DAGSTUHL
REPORTS

Volume 11, Issue 4, May 2021

ISSN 2192-5283

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/2192-5283

Publication date
November, 2021

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at `http://dnb.d-nb.de`.

Digital Object Identifier: 10.4230/DagRep.11.4.i

Aims and Scope
The periodical *Dagstuhl Reports* documents the program and the results of Dagstuhl Seminars and Dagstuhl Perspectives Workshops.
In principal, for each Dagstuhl Seminar or Dagstuhl Perspectives Workshop a report is published that contains the following:

- an executive summary of the seminar program and the fundamental results,
- an overview of the talks given during the seminar (summarized as talk abstracts), and
- summaries from working groups (if applicable).

This basic framework can be extended by suitable contributions that are related to the program of the seminar, e. g. summaries from panel discussions or open problem sessions.

Contact
Schloss Dagstuhl – Leibniz-Zentrum für Informatik
Dagstuhl Reports, Editorial Office
Oktavie-Allee, 66687 Wadern, Germany
reports@dagstuhl.de
http://www.dagstuhl.de/dagrep

Report from Dagstuhl Seminar 21181

Computational Geometry

Edited by

Siu-Wing Cheng[1], Anne Driemel[2], and Jeff M. Phillips[3]

1 **HKUST – Kowloon, HK,** scheng@cse.ust.hk
2 **Universität Bonn, DE,** driemel@cs.uni-bonn.de
3 **University of Utah – Salt Lake City, US,** jeffp@cs.utah.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 21181 "Computational Geometry". The seminar was held from May 2 to May 7, 2021. Because of COVID, the seminar was held online in a virtual manner, and 36 participants from various countries attended it. New advances and directions in computational geometry were presented and discussed. The report collects the abstracts of talks and open problems presented in the seminar.

Seminar May 2–7, 2021 – http://www.dagstuhl.de/21181
2012 ACM Subject Classification Theory of computation → Computational geometry
Keywords and phrases algorithms, computational geometry, Computational topology, data structures, Discrete geometry
Digital Object Identifier 10.4230/DagRep.11.4.1
Edited in cooperation with Conradi, Jacobus

1 Executive Summary

Siu-Wing Cheng
Anne Driemel
Jeff M. Phillips

Computational Geometry

The field of computational geometry is concerned with the design, analysis, and implementation of algorithms for geometric and topological problems, which arise naturally in a wide range of areas, including computer graphics, CAD, robotics, computer vision, image processing, spatial databases, GIS, molecular biology, sensor networks, machine learning, data mining, scientific computing, theoretical computer science, and pure mathematics. Computational geometry is a vibrant and mature field of research, with several dedicated international conferences and journals and strong intellectual connections with other computing and mathematics disciplines.

In the early years mostly theoretical foundations of geometric algorithms were laid and fundamental research remains an important issue in the field. Meanwhile, as the field matured, researchers have started paying close attention to applications and implementations of geometric and topological algorithms. Several software libraries for geometric computation (e.g. leda, cgal, core) have been developed. Remarkably, this emphasis on applications and implementations has emerged from the originally theoretically oriented computational geometry community itself, so many researchers are concerned now with theoretical foundations as well as implementations.

Seminar Topics

The emphasis of this seminar was on presenting recent developments in computational geometry, as well as identifying new challenges, opportunities, and connections to other fields of computing. In addition to the usual broad coverage of new results in the field, the seminar included broad survey talks on Computational Topology on Surfaces and Graphs as well as Combinatorial Complexity of Geometric Structures.

Computational Topology on Surfaces and Graphs

Computational topology has seen exciting advances in a number of topics. Indeed, best paper awards in several recent SoCGs went to papers on these topics. In 2019, Cohen-Addad et al. give a lower bound to a cutting problem in embedded graphs, essentially matching the running time of the fastest algorithm known and settling a 17-year old question. In 2018, Goaoc et al. proved that it is NP-complete to decide if a d-dimensional simplicial complex is shellable for $d \geq 2$, resolving a question of Danaraj and Klee in 1978. In 2017, Despré and Lazarus presented simple quasi-linear algorithms for questions regarding geometric intersection number of a curve on a surface. Progress in these and related topics have had influences in problems on graphs embedded on surfaces, maximum flows and multiple-source shortest paths in planar graphs, collapsibility of simplicial complexes, metric learning, etc. The seminar highlighted these topics with two overview talks. The first by Hsien-Chih Chang was on Tightening Curves on Surfaces, and provided a overview of recent advancements in this area, and exciting directions for future work on flipping triangulations and morphing planar multicurves using electrical moves. The second by Uli Wagner discussed Embeddability of Simplicial Compexes, and also the flurry of recent research in this area, and pinpointed the several remaining questions and where the community has not yet been able to resolve the embeddability and why the challenges remain. These talks, and other on recent advances, helped summarize the state of this area, and generate new avenues towards moving the field further forward.

Combinatorial Complexity of Geometric Structures

The understanding of the combinatorial properties of geometric structures is at the core of computational geometry. A lot of these structures such as union of shapes, cuttings, arrangements, Delaunay triangulation, Voronoi diagram have found numerous applications in algorithm design. For example, the analysis of the complexity of the union of translates of a convex body allows us to understand the complexity of the free space in planning the motion of that convex body under translation. Their studies have also triggered the development of new theoretical tools such as the polynomial method that has been gaining a lot of attention lately. There are also new applications that require the modeling of uncertain data and hence call for a study of many geometric structures under a stochastic setting. The seminar promoted these topics via two overview talks. The first overview talk was by Mikkel Abrahamsen on Minimum Fence Enclosure and Separation Problems; this line of work generalizes the notion of convex hull by identifies other minimally enclosing structures called fences, and the interesting combinatorial properties that arise. The second overview talk by Evanthia Papadopoulou was on Problems in Voronoi and Voronoi-like diagrams. This talk discussed the advancement in generalizations of the classic geometric object of Voronoi diagrams to be defined among geometric objects beyond points, and to higher-order complexes. In addition to providing snapshots of these exciting subareas, they provided future directions for research within these topics and in how they can interact across the broader computational geometry landscape.

Participants and Participation

Dagstuhl seminars on computational geometry have been organized in a two year rhythm since a start in 1990. They have been extremely successful both in disseminating the knowledge and identifying new research thrusts. Many major results in computational geometry were first presented in Dagstuhl seminars, and interactions among the participants at these seminars have led to numerous new results in the field. These seminars have also played an important role in bringing researchers together, fostering collaboration, and exposing young talent to the seniors of the field and vice versa. They have arguably been the most influential meetings in the field of computational geometry. The organizers held a lottery for the fifth time this year; the lottery allows to create space to invite younger researchers, rejuvenating the seminar, while keeping a large group of senior and well-known scholars involved. The seminar has now a more balanced attendance in terms of seniority and gender than in the past. This year, 36 researchers from various countries and continents attended the seminar, despite the virtual nature due to COVID-19, showing the strong interest of the community for this event.

Due to the COVID-19 pandemic, the seminar was held entirely virtually. Talks were held over four days. Each day had 2 two-hour blocks of talks, separated by a 2-hour meal break. They were held in the late-afternoon and evening in Europe, which allowed for participants from North America to join in during their morning hours. Unfortunately, this timing was late for those in Asia. The talks were held on Zoom, a Slack server was set up for a more persistent text-based discussion, and a Wonder.me instance was arranged for dynamic forming of group discussions before and after each session. All of these settings were used to communicate research, form collaborations, and attack open problems. Although not as wonderful as actually being at Schloss Dagstuhl, these online mechanisms provided for a workable replacement for what a normal Dagstuhl seminar provides in this abnormal time.

The feedback from participants was very positive. The participants viewed the composition of the group positively, remarking how it was well-balanced in terms of seniority and gender. They also praised the quality of the talks as of very high quality – making the virtual-only participation worthwhile.

We warmly thank the scientific, administrative and technical staff at Schloss Dagstuhl! Dagstuhl made virtual hosting possible and easy in a time filled with complications. Despite not providing a physical space to meet, socialize, and collaborate, their help in organizing the event made it a success despite the less than ideal circumstances.

2 Table of Contents

3 Overview of Talks

3.1 Minimum Fence Enclosure and Separation Problems

Mikkel Abrahamsen (University of Copenhagen, DK)

The classical problem of computing the convex hull of a given set of points in the plane can be formulated in a natural way as a fence enclosure problem: Find the shortest fence that encloses the points. In this talk, we survey our recent work on related problems that appear when the formulation is changed slightly. We will touch upon the following problems:

1. Given a set of points in the plane, find the two fences of minimum total length that together enclose all the points. We will outline an algorithm with $O(n \log^2 n)$ running time.
2. Given a set of points and a number k, find a system of at most k fences of minimum total length that enclose the points. We will explain how this problem can be attacked via dynamic programming, which leads to a polynomial-time, although very slow, algorithm.
3. Given a set of unit disks, find a system of fences of minimum total length that enclose all the disks (with no restriction on the number of fences). We report on a near-linear time algorithm for this and related problems.
4. Finally, we consider the problem where the input consists of pairwise interior-disjoint polygons in the plane, and each polygon has a color. We want to compute the fence of minimum total length that separates all pairs of polygons of different colors. This problem can be solved in polynomial time when there are just two colors, but it becomes NP-hard already for three colors. We report on an approximation algorithm. During the talk, we will suggest directions for future research.

3.2 On the Union of Cubes in 3D

Pankaj Kumar Agarwal (Duke University – Durham, US)

Joint work of Pankaj K. Agarwal, Micha Sharir, Alex Steiger
Main reference Pankaj K. Agarwal, Micha Sharir, Alex Steiger: "Decomposing the Complement of the Union of Cubes in Three Dimensions", in Proc. of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021, pp. 1425–1444, SIAM, 2021.
URL http://dx.doi.org/10.1137/1.9781611976465.86

Let C be a set of n axis-aligned cubes of arbitrary sizes in 3D, let $K = \mathbb{R}^3 \setminus U(C)$ be the complement of their union (i.e. free space). The complexity of K, denoted by k, can vary between $O(1)$ and $O(n^2)$. This talk presents two main results: (i) An output-sensitive algorithm to compute K in time $O(n\text{polylog}(n) + k)$ time; and (ii) an output-sensitive algorithm to partition K into $O((n + k)\text{polylog}(n))$ boxes in the same time bound. These results can be slightly improved if the cubes in C have roughly the same size or if they have bounded depth (i.e. any point in $\mathbb{R}^3$ lies in $O(1)$ cubes).

3.3 Fine-grained Complexity of Nearest Neighbors for Fréchet Distance

Karl Bringmann (Universität des Saarlandes – Saarbrücken, DE)

Joint work of Karl Bringmann, Anne Driemel, André Nusser, Ioannis Psarros

Fine-grained complexity theory is the area of theoretical computer science that proves conditional lower bounds based on the 3-SUM Hypothesis, the Strong Exponential Time Hypothesis, and similar conjectures. This talk is an introduction to recent fine-grained lower bounds in computational geometry, with a focus on lower bounds for polynomial-time problems based on the Orthogonal Vectors Hypothesis. Specifically, we discuss conditional lower bounds for nearest neighbor search under the Euclidean distance and Fréchet distance. We see that lower bounds for the Bichromatic Closest Pair problem follow from the Orthogonal Vectors Hypothesis by simple embeddings. This implies a near-linear lower bound for the query time of nearest neighbor data structures. Then we see Unbalanced Orthogonal Vectors, a simple trick to even rule out any polynomial preprocessing time and near-linear query time for nearest neighbors. Finally, we discuss recent, unpublished work on approximate nearest neighbor data structures for the Fréchet distance.

3.4 Around k-fold filtrations

Mickaël Buchet (TU Graz, AT)

Joint work of Mickaël Buchet, Bianca B. Dornelas, Michael Kerber

Given a point set P, a number k and a radius r, the k-fold cover is defined as the union of all intersections of k balls of radius r around points of P. This cover has two parameters (k and r) and defines a very natural bi-filtration of the space. This bi-filtration represents one natural occurrence for multi-parameter persistent homology. Unfortunately, the usual combinatorial objects used to represent the bi-filtration are not bi-filtration themselves and are of larger size. I will talk about several ways to tackle this issue through various approximations, constructions and sparsifications, mostly adapted from techniques used in the one-parameter case but where the nature of the k-fold cover raises new challenges and interesting open questions.

3.5 Computing the inverse geodesic length in graphs of bounded treewidth

Sergio Cabello (University of Ljubljana, SI)

Main reference Sergio Cabello: "Computing the inverse geodesic length in planar graphs and graphs of bounded treewidth", CoRR, Vol. abs/1908.01317, 2019.
URL https://arxiv.org/abs/1908.01317

The inverse geodesic length of a graph G is the sum of the inverse of the distances between all pairs of distinct vertices of G. In some domains it is known as the Harary index or the global efficiency of the graph. We show that, if G has n vertices and constant treewidth,

then the inverse geodesic length of G can be computed in near-linear time. To achieve this we use techniques developed for computing the sum of the distances, which does not have "inverse" component, together with batched evaluations of rational functions.

3.6 Tightening Curves on Surfaces

Hsien-Chih Chang (Dartmouth College – Hanover, US)

In this talk we survey the recent advancements in tightening curves on surfaces under different categories of curves and deformations in the past few decades. We then present two possible directions for future research, one on computing geodesics by flipping triangulations and the other on the complexity of morphing planar multicurves using electrical moves. Open questions are provided for the curious to ponder.

3.7 Multicuts in planar and surface-embedded graphs

Éric Colin de Verdière (CNRS, LIGM, Marne-la-Vallée, FR)

Joint work of Vincent Cohen-Addad, Vincent, Éric Colin de Verdière, Dániel Marx, Arnaud de Mesmay

The Multicut problem is defined as follows. Given an edge-weighted graph G and pairs of vertices (s_1, t_1), ..., (s_k, t_k), compute a minimum-weight subset of edges whose removal disconnects each pair (s_i, t_i).

This problem is NP-hard, APX-hard, and W[1]-hard in the number of pairs of terminals, even in very simple cases, such as planar graphs.

We will survey some recent results on this problem, on planar graphs and more generally on graphs embedded on a fixed surface: An exact algorithm, whose running time is a polynomial in the genus and the number of terminals [1]; a matching lower bound assuming ETH [2]; and an approximation scheme with running time $O(n \log n)$ if the approximation factor, the genus, and the number of terminals are fixed [3].

All these results rely on topological methods: The subgraph of the dual of G, made of the edges dual to a multicut, has nice properties, which can be exploited using classical tools from algebraic topology such as homotopy, homology, and universal covers of surfaces.

References

1 Éric Colin de Verdière. Multicuts in planar and bounded-genus graphs with bounded number of terminals. *Algorithmica*, 78:1206–1224, 2017.
2 Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay. Almost tight lower bounds for hard cutting problems in embedded graphs. In *Proc. Int. Symp. on Computational Geometry*, pages 27:1–27:16, 2019. Full version to appear in *J. ACM*.
3 Vincent Cohen-Addad, Éric Colin de Verdière, and Arnaud de Mesmay. A near-linear approximation scheme for multicuts of embedded graphs with a fixed number of terminals. *SIAM J. Comput.*, 50(1):1–33, 2021.

3.8 Fine-grained Complexity of the k-Shortcut Fréchet distance

Jacobus Conradi (Universität Bonn, DE)

Joint work of Jacobus Conradi, Anne Driemel

The Fréchet distance is a popular measure of dissimilarity for polygonal curves. It is defined as a min-max formulation that considers all direction-preserving continuous bijections of the two curves. Because of its susceptibility to noise, Driemel and Har-Peled introduced the shortcut Fréchet distance in 2012, where one is allowed to take shortcuts along one of the curves, similar to the edit distance for sequences. We analyse the parametrized version of this problem, where the number of shortcuts is bounded by a parameter k. The corresponding decision problem can be stated as follows: Given two polygonal curves T and B of at most n vertices, a parameter k and a distance threshold δ, is it possible to introduce k shortcuts along B such that the Fréchet distance of the resulting curve and the curve T is at most δ? We study this problem for polygonal curves in the plane. We provide a complexity analysis for this problem with the following results: (i) assuming the exponential-time-hypothesis (ETH), there exists no algorithm with running time bounded by $n^{o(k)}$; (ii) there exists a decision algorithm with running time in $O(kn^{2k+2}\log n)$. In contrast, we also show that efficient approximate decider algorithms are possible, even when k is large. We present a $(3+\varepsilon)$-approximate decider algorithm with running time in $O(kn^2\log^2 n)$ for fixed ε. In addition, we can show that, if k is a constant and the two curves are c-packed for some constant c, then the approximate decider algorithm runs in near-linear time.

3.9 Contractibility on 3-manifold boundaries and compressed problems on surfaces

Arnaud de Mesmay (University Paris-Est – Marne-la-Vallée, FR)

Joint work of Erin W. Chambers, Arnaud de Mesmay, Francis Lazarus, Salman Parsa
Main reference Erin Wolf Chambers, Francis Lazarus, Arnaud de Mesmay, Salman Parsa: "Algorithms for Contractibility of Compressed Curves on 3-Manifold Boundaries", CoRR, Vol. abs/2012.02352, 2020.
URL https://arxiv.org/abs/2012.02352

We show that the problem of deciding whether a closed curve on the boundary of a 3-manifold is contractible is in NP, and furthermore we provide an algorithm that is FPT in the complexity of the manifold. This relies on techniques to solve various topological problems for curves on surfaces with compressed inputs. The talk assumes no topological background and focuses on explaining why issues with compression appear naturally in this line of work.

3.10 Practical volume approximation of H, V, and Z-polytopes

Ioannis Emiris (University of Athens & Athena Research Center, GR)

Joint work of Apostolos Chalkis, Ioannis Z. Emiris, Vissarion Fisikopoulos
Main reference Apostolos Chalkis, Ioannis Z. Emiris, Vissarion Fisikopoulos: "Practical Volume Estimation by a New Annealing Schedule for Cooling Convex Bodies", CoRR, Vol. abs/1905.05494, 2019.
URL http://arxiv.org/abs/1905.05494

We tackle the problem of efficiently approximating the volume of convex polytopes, when these are given in 3 different representations: H-polytopes, which have been studied extensively, V-polytopes, and zonotopes (Z-polytopes). We design a novel practical Multiphase Monte Carlo (MMC) algorithm that leverages geometric random walks. Our algorithmic contributions include: (i) a uniform sampler employing billiard walk for the first time in volume computation, showing it mixes much faster than Hit-and-Run variants, (ii) a new simulated annealing schedule, generalizing existing MMC, by introducing adaptive convex bodies which, moreover, (iii) probabilistically restricts volume ratios to a target interval, thus drastically reducing the number of bodies in MMC. Extensive experiments indicate that our method requires about $O(d^2)$ oracle calls compared to the best theoretical bound of $O^*(d^3)$, where d is the dimension. For zonotopes, we appropriately use centrally symmetric polytopes which yield an MMC with constant number of phases, when the ratio of generators over dimension is small. We present a detailed experimental evaluation of our algorithm using Birkhoff polytopes and polytopes of all 3 classes. Our open-source C++ software offers the first method that scales up to thousands of dimensions for H-polytopes and in the hundreds for V- and Z-polytopes on moderate hardware. We illustrate it on SDP optimization, by means of sampling spectrahedra, and on sampling structured polytopes obtained from modeling financial portfolios.

3.11 Consistent Digital Line Segments

Matias Korman (Siemens EDA – Wilsonville, US)

Joint work of Man-Kwun Chiu, Matias Korman, Martin Suderland, Takeshi Tokuyama
Main reference Man-Kwun Chiu, Matias Korman, Martin Suderland, Takeshi Tokuyama: "Distance Bounds for High Dimensional Consistent Digital Rays and 2-D Partially-Consistent Digital Rays", in Proc. of the 28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference), LIPIcs, Vol. 173, pp. 34:1–34:22, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
URL http://dx.doi.org/10.4230/LIPIcs.ESA.2020.34

In this talk I will introduce the concept of "consistent digital segments": In short, we look for an axiomatic construction of segments in discrete spaces, akin to the construction that we have in Euclidean segments. After discussing movivation, we will focus on known results in two and higher dimensions. Each result will be followed with discussion on what are the big open problems that remain and what are possible lines of research that could be followed.

3.12 Approximating Maximum Independent Set in the Plane

Joseph S. B. Mitchell (Stony Brook University, US)

Main reference Joseph S. B. Mitchell: "Approximating Maximum Independent Set for Rectangles in the Plane", CoRR, Vol. abs/2101.00326, 2021.
URL https://arxiv.org/abs/2101.00326

We give a polynomial-time constant-factor approximation algorithm for maximum independent set for (axis-aligned) rectangles in the plane. Using a polynomial-time algorithm, the best approximation factor previously known is $O(\log \log n)$. The results are based on a new form of recursive partitioning in the plane, in which faces that are constant-complexity and orthogonally convex are recursively partitioned in a constant number of such faces.

3.13 Efficient Near-Neighbor Search via Average Distortion Embeddings

Aleksandar Nikolov (University of Toronto, CA)

Joint work of Deepanshu Kush, Aleksandar Nikolov, Haohua Teng
Main reference Deepanshu Kush, Aleksandar Nikolov, Haohua Tang: "Near Neighbor Search via Efficient Average Distortion Embeddings", CoRR, Vol. abs/2105.04712, 2021.
URL https://arxiv.org/abs/2105.04712

A recent series of papers by Andoni, Naor, Nikolov, Razenshteyn, and Waingarten (STOC 2018, FOCS 2018) has given approximate near neighbour search (ANN) data structures for a wide class of distance metrics, including all norms. In particular, these data structures achieve approximation on the order of p for ℓ_p norms with space complexity nearly linear in the dataset size n and polynomial in the dimension d, and query time sub-linear in n and polynomial in d. The main shortcoming is the exponential in d pre-processing time required for their construction. In this talk, we describe a more direct framework for constructing ANN data structures for general norms. More specifically, we show via an algorithmic reduction that an efficient ANN data structure for a given metric is implied by an efficient average distortion embedding of the metric into the Manhattan norm or into Euclidean space. In particular, the resulting data structures require only polynomial pre-processing time, as long as the embedding can be computed in polynomial time. As a concrete instantiation of this framework, we give an ANN data structure for ℓ_p with efficient pre-processing that matches the approximation factor, space and query complexity of the aforementioned data structure of Andoni et al.

3.14 Problems in Voronoi and Voronoi-like diagrams

Evanthia Papadopoulou (University of Lugano, CH)

Joint work of Kolja Junginger, Evanthia Papadopoulou
Main reference Kolja Junginger, Evanthia Papadopoulou: "Deletion in Abstract Voronoi Diagrams in Expected Linear Time", in Proc. of the 34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary, LIPIcs, Vol. 99, pp. 50:1–50:14, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
URL http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.50
Main reference Kolja Junginger, Evanthia Papadopoulou: "Deletion in abstract Voronoi diagrams in expected linear time", CoRR, Vol. abs/1803.05372, 2018.
URL http://arxiv.org/abs/1803.05372

Differences between classical Voronoi diagrams of points, versus segments, circles, or polygons are often forgotten or underestimated. Abstract Voronoi diagrams (AVDs) offer a unifying framework for many such Voronoi diagrams in the plane; however, diagrams of points are not a representative concrete structure for AVDs. In this talk, I will first survey fundamental differences between higher order Voronoi diagrams of points and their counterparts of segments or AVDs. I will then address the problem of site-deletion in abstract Voronoi diagrams in expected linear time. Although linear-time algorithms for site-deletion in planar point Voronoi diagrams had been well-known to exist since the late 80's, the corresponding problems for non-point Voronoi diagrams remained open, until recently. As a byproduct, I will introduce *abstract Voronoi-like diagrams*, a relaxed Voronoi structure of independent interest, which leads to a very simple randomized incremental technique to perform site-deletion in abstract Voronoi diagrams. The technique extends to computing various tree-like Voronoi diagrams such as constructing the farthest abstract Voronoi diagram, after the order of its regions at infinity is known, constructing the order-$(k+1)$ subdivision within an order-k Voronoi region, and others. The time analysis introduces a simple alternative to backwards analysis applicable to order-dependent structures.

3.15 Stronger Bounds for Weak Epsilon-Nets in Higher Dimensions

Natan Rubin (Ben Gurion University – Beer Sheva, IL)

Main reference Natan Rubin: "Stronger Bounds for Weak Epsilon-Nets in Higher Dimensions", CoRR, Vol. abs/2104.12654, 2021.
URL https://arxiv.org/abs/2104.12654

Given a finite point set P in $\mathbb{R}^d$, and $\epsilon > 0$ we say that $N \subseteq \mathbb{R}^d$ is a weak ϵ-net if it pierces every convex set K with $|K \cap P| \geq \epsilon |P|$.

Let $d \geq 3$. We show that for any finite point set in $\mathbb{R}^d$, and any $\epsilon > 0$, there exist a weak ϵ-net of cardinality $O\left(\frac{1}{\epsilon^{d-1/2+\gamma}}\right)$, where $\gamma > 0$ is an arbitrary small constant.

This is the first improvement of the bound of $O^*\left(\frac{1}{\epsilon^d}\right)$ that was obtained in 1993 by Chazelle, Edelsbrunner, Grigni, Guibas, Sharir, and Welzl for general point sets in dimension $d \geq 3$.[1]

[1] $O^*(\cdot)$-notation hides multiplicative factors that are polylogarithmic in $\log 1/\epsilon$.

3.16 Terrain prickliness: theoretical grounds for low complexity viewsheds

Maria Saumell (The Czech Academy of Sciences – Prague & Czech Technical University in Prague, CZ)

Joint work of Ankush Acharyya, Ramesh K. Jallu, Maarten Löffler, Gert G. T. Meijer, Maria Saumell, Rodrigo I. Silveira, Frank Staals, Hans Raj Tiwary
Main reference Ankush Acharyya, Ramesh K. Jallu, Maarten Löffler, Gert G. T. Meijer, Maria Saumell, Rodrigo I. Silveira, Frank Staals, Hans Raj Tiwary: "Terrain prickliness: theoretical grounds for low complexity viewsheds", CoRR, Vol. abs/2103.06696, 2021.
URL https://arxiv.org/abs/2103.06696

An important task when working with terrain models is computing viewsheds: the parts of the terrain visible from a given viewpoint. When the terrain is modeled as a polyhedral terrain, the viewshed is composed of the union of all the triangle parts that are visible from the viewpoint. The complexity of a viewshed can vary significantly, from constant to quadratic in the number of terrain vertices, depending on the terrain topography and the viewpoint position.

In this work we study a new topographic attribute, the *prickliness*, that measures the number of local maxima in a terrain from all possible perspectives. We show that the prickliness effectively captures the potential of 2.5D terrains to have high complexity viewsheds, and we present near-optimal algorithms to compute the prickliness of 1.5D and 2.5D terrains. We also report on some experiments relating the prickliness of real word 2.5D terrains to the size of the terrains and to their viewshed complexity.

3.17 Guarding Problems

Christiane Schmidt (Linköping University, SE)

Joint work of Sarah Cannon, Ovidiu Daescu, Thomas Fai, Stephan Friedrichs, Justin Iwerks, Undine Leopold, Hemant Malik, Bengt J. Nilsson, Valentin Polishchuk, Christiane Schmidt

The classical Art Gallery Problem (AGP) asks for the minimum number of guards that are necessary to visually cover a polygon P, where visibility between two points is defined as the line segment between these points being fully contained in P. In this talk, we highlight some recent works and open problems for different variants of the AGP.

First, we consider k-transmitters, for which the definition of visibility is altered: two points, p, q, can see each other if the line segment $\overline{pq}$ intersects P's boundary at most k times. We review results on stationary point and edge $k-/2$-transmitters – Art Gallery theorems and complexity results; we present several properties of $k-/2$-transmitters and recent complexity results on 2-transmitter watchman routes. Finally, we highlight an open problem on Art Gallery theorems for 2-transmitters in simple polygon: we have a lower bound of $\lfloor n/5 \rfloor$ guards, but the best known upper bound essentially stems from "normal" guards/0-transmitters: $\lfloor (n-1)/3 \rfloor$.

We then consider guarding problems in special polygon classes (altering the environment to be guarded rather than the capabilities of the guards). We show that we can find an optimal guard set for uni-monotone polygons in linear time and that the size of a minimum cardinality guard set equals the size of a maximum cardinality witness set for this class –

uni-monotone polygons are perfect. We survey for which guarding problems discretizations have been obtained. Finally, we review under which types of visibility definition and for which polygon classes similar results on perfectness have been obtained and give the basic idea of these results. This leads to an open question of AGP in monotone polygons: can we show perfectness for, e.g., staircase or O-visibility. Can we discretize in this special polygon class?

3.18 Sketching Persistence Diagrams

Donald Sheehy (North Carolina State University – Raleigh, US)

Joint work of Donald R. Sheehy, Siddharth Sheth
Main reference Donald R. Sheehy, Siddharth Sheth: "Sketching Persistence Diagrams", in Proc. of the 37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference), LIPIcs, Vol. 189, pp. 57:1–57:15, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
URL http://dx.doi.org/10.4230/LIPIcs.SoCG.2021.57

Given a persistence diagram with n points, we give an algorithm that produces a sequence of n persistence diagrams converging in bottleneck distance to the input diagram, the ith of which has i distinct (weighted) points and is a 2-approximation to the closest persistence diagram with that many distinct points. For each approximation, we precompute the optimal matching between the ith and the $(i+1)$st. Perhaps surprisingly, the entire sequence of diagrams as well as the sequence of matchings can be represented in $O(n)$ space. The main approach is to use a variation of the greedy permutation of the persistence diagram to give good Hausdorff approximations and assign weights to these subsets. We give a new algorithm to efficiently compute this permutation, despite the high implicit dimension of points in a persistence diagram due to the effect of the diagonal. The sketches are also structured to permit fast (linear time) approximations to the Hausdorff distance between diagrams – a lower bound on the bottleneck distance. For approximating the bottleneck distance, sketches can also be used to compute a linear-size neighborhood graph directly, obviating the need for geometric data structures used in state-of-the-art methods for bottleneck computation.

3.19 Optimal bounds for the colorful fractional Helly theorem

Martin Tancer (Charles University – Prague, CZ)

Joint work of Denys Bulavka, Afshin Goodarzi, Martin Tancer
Main reference Denys Bulavka, Afshin Goodarzi, Martin Tancer: "Optimal Bounds for the Colorful Fractional Helly Theorem", in Proc. of the 37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference), LIPIcs, Vol. 189, pp. 19:1–19:14, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
URL http://dx.doi.org/10.4230/LIPIcs.SoCG.2021.19

The well known fractional Helly theorem and colorful Helly theorem can be merged into the so called colorful fractional Helly theorem. It states: for every $\alpha \in (0,1]$ and every non-negative integer d, there is $\beta = \beta(\alpha, d) \in (0,1]$ with the following property. Let $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ be finite nonempty families of convex sets in $\mathbb{R}^d$ of sizes $n_1, \ldots, n_{d+1}$, respectively. If at least $\alpha n_1 n_2 \cdots n_{d+1}$ of the colorful $(d+1)$-tuples have a nonempty intersection, then there is

$i \in [d+1]$ such that $\mathcal{F}_i$ contains a subfamily of size at least βn_i with a nonempty intersection. (A colorful $(d+1)$-tuple is a $(d+1)$-tuple $(F_1, \ldots, F_{d+1})$ such that F_i belongs to $\mathcal{F}_i$ for every i.)

The colorful fractional Helly theorem was first stated and proved by Bárány, Fodor, Montejano, Oliveros, and Pór in 2014 with $\beta = \alpha/(d+1)$. In 2017 Kim proved the theorem with better function β, which in particular tends to 1 when α tends to 1. Kim also conjectured what is the optimal bound for $\beta(\alpha, d)$ and provided the upper bound example for the optimal bound. The conjectured bound coincides with the optimal bounds for the (non-colorful) fractional Helly theorem proved independently by Eckhoff and Kalai around 1984.

We verify Kim's conjecture by extending Kalai's approach to the colorful scenario. Moreover, we obtain optimal bounds also in a more general setting when we allow several sets of the same color.

3.20 Light Euclidean Spanners

Csaba Tóth (California State University – Los Angeles, US)

Main reference Sujoy Bhore, Csaba D. Tóth: "On Euclidean Steiner (1+ϵ)-Spanners", in Proc. of the 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference), LIPIcs, Vol. 187, pp. 13:1–13:16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
URL http://dx.doi.org/10.4230/LIPIcs.STACS.2021.13

Lightness is a fundamental parameter for Euclidean spanners; it is the ratio of the spanner weight to the weight of the minimum spanning tree of a finite set of points in $\mathbb{R}^d$. In a recent breakthrough, Le and Solomon (2019) established the precise dependencies on $\varepsilon > 0$ and $d \in \mathbb{N}$ of the minimum lightness of a $(1+\varepsilon)$-spanner, and observed that additional Steiner points can substantially improve the lightness. Le and Solomon (2020) constructed Steiner $(1+\varepsilon)$-spanners of lightness $O(\varepsilon^{-1} \log n)$ for n points in the plane. They also constructed spanners of lightness $\tilde{O}(\varepsilon^{-(d+1)/2})$ in dimensions $d \geq 3$.

We established a lower bound of $\Omega(\varepsilon^{-d/2})$ for the lightness of Steiner $(1+\varepsilon)$-spanners in $\mathbb{R}^d$, for all $d \geq 2$. We also prove that this bound is the best possible for $d = 2$, that is, for every finite set of points in the plane and every $\varepsilon > 0$, there exists a Euclidean Steiner $(1+\varepsilon)$-spanner of lightness $O(\varepsilon^{-1})$. We generalize the notion of shallow light trees, which may be of independent interest, and use directional spanners and a modified window partitioning scheme to achieve a tight weight analysis. (Joint work with Sujoy Bhore.)

3.21 Embeddability of Simplicial Complexes

Uli Wagner (IST Austria – Klosterneuburg, AT)

Consider the following decision problem in computational topology, which we refer to as the *embeddability problem*: Given a finite k-dimensional simplicial complex K, does K admit a (piecewise-linear) embedding into $\mathbb{R}^d$? More generally, the *extendability of embeddings problem* asks: Given K, a subcomplex $A \subseteq K$, and an embedding $f\colon A \to \mathbb{R}^d$, can f be extended to an embedding $F\colon K \to \mathbb{R}^d$? (The embeddability problem is the special case $A = \emptyset$.)

We survey what is known about the decidability and computational complexity of these problems in higher dimensions (for fixed positive integers $k \leq d$). Some of the main results and open questions are:

- For $d = 3$, the embeddability problem is known to be algorithmically decidable [7] as well as NP-hard [2]; the exact complexity of the problem (including whether it lies in NP) remains unknown.
- In the so-called *metastable range* $d \geq \frac{3(k+1)}{2}$, both embeddability and extendability of embeddings can be decided in polynomial time (for fixed k and d). Indeed, in this dimension range, by classical work of Haefliger and Weber [4, 8], both problems reduce to questions about the existence of equivariant maps from the *deleted product* $(K \times K) \setminus \{(x, x) \colon x \in K\}$ to S^{d-1}, and the latter can be decided in polynomial time by a series of works on computational homotopy theory culminating in [1].
- Outside the metastable range, it is known that the embeddability problem is NP-hard if $d \geq 4$, and algorithmically undecidable if $k + 1 \geq d \geq 5$ [6]. Moreover, extendability of embeddings is algorithmically undecidable for almost all dimensions outside the metastable range, namely for $8 \leq d < \lfloor \frac{3(k+1)}{2} \rfloor$ [3]. In [3], it is claimed that this also implies undecidability of the embeddability problem in the same range of dimensions, but the proof of this implication contains a gap [5]. Fixing this gap would require constructing suitable so-called *linking gadgets*. E.g., in the special case $k = 5, d = 8$, this would require constructing a 5-dimensional complex L containing copies of S^5 and S^2 as vertex-disjoint subcomplexes, such that in any embedding of G, the images of S^5 and S^2 are linked with linking number ± 1. (Currently, it is only known that there are examples of complexes L that force an odd linking number.)

References

1 M. Čadek, M. Krčál, and L. Vokřínek. Algorithmic solvability of the lifting-extension problem. *Discrete Comput. Geom.*, 57(4):915–965, 2017.

2 A. de Mesmay, Y. Rieck, E. Sedgwick, and M. Tancer. Embeddability in $\mathbb{R}^3$ is NP-hard. In *Proc. 20th Ann. ACM-SIAM Symp. Discr. Algorithms (SODA)*, pp. 1316–1329, 2018.

3 M. Filakovský, S. Zhechev, and U. Wagner. Embeddability of Simplicial Complexes is Undecidable. In *Proc. 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 767–785, 2020.

4 A. Haefliger. Plongements différentiables dans le domaine stable. *Comment. Math. Helv.*, 37:155–176, 1962/1963.

5 R. Karasev and A. Skopenkov. Some 'converses' to intrinsic linking theorems. Preprint, https://arxiv.org/abs/2008.02523.

6 J. Matoušek, M. Tancer, and U. Wagner. Hardness of embedding simplicial complexes in *mathbbR*^{d}. *J. Eur. Math. Soc.*, 13(2):259–295, 2011.

7 J. Matoušek, E. Sedgwick, M. Tancer, and U. Wagner. Embeddability in the 3-sphere is decidable. *J. ACM*, 65(1):Art. 5, 49, 2018.

8 C. Weber. Plongements de polyedres dans le domaine metastable. *Comment. Math. Helv.*, 42:1–27, 1967.

3.22 Comparing Embedded and Immersed Graphs

Carola Wenk (Tulane University – New Orleans, US)

Data in the form of one-dimensional structures, embedded or immersed in an ambient space, arise in a variety of applications, including GIS analysis, trajectory clustering, protein alignment, plant morphology, commodity networks such as electrical grids, and geographic networks of roads or rivers. Often one is interested in comparing such structures. But since data collection introduces noise and errors, distance measure need to be robust to different issues in the data.In this talk we will focus on graphs and provide a survey of distance measures suitable for comparing embedded or immersed graphs. Oftentimes these graphs are not isomorphic, nor is one interested in true subgraph isomorphism. However, it is desirable to have a mapping of one graph to the other, which should fulfill certain properties such as continuity. And the distances should ideally measure differences in geometry and topology. We will examine several distances from mathematical, algorithmic, and applied point of views, and pose open problems for comparing embedded or immersed graphs.

4 Open problems

4.1 Twin-width of String Graphs

Édouard Bonnet (ENS – Lyon, FR)

A tri-graph is a graph consisting of vertices, edges and red edges. Contractions on a tri-graph of two vertices u and v recolour edges according to the following rules depending on the sets of neighbours $N(u)$ and $N(v)$ of u and v: 1) edges from $N(u) \triangle N(v)$ to the new vertex are always red 2) edges from $N(u) \cap N(v)$ to the new vertex are not red, only if both original edges were not red. The twin-width of any graph G is then definied as the smallest integer d, that allows a contraction sequence on G, where the maximum red degree during the sequence is bound by d. Do $K_{t,t}$-free string graphs have bounded twin-width? Similarly, do $H_{t,t}$-free string graphs have bounded twin-width?

4.2 Expected Volume of Stochastic Bounding Box

Sergio Cabello (University of Ljubljana, SI)

Given n points in $\mathbb{R}^d$ together with probabilities for each point, we want to compute the expected volume of the bounding box of these points. Is this problem fixed parameter tractible with respect to d? Is it $\#W[1]$-hard w.r.t. d? Is the dependency on d in the degree needed?

4.3 Average Distortion Embeddings

Aleksandar Nikolov (University of Toronto, CA)

Joint work of Deepanshu Kush, Aleksandar Nikolov, Haohua Tang
Main reference Deepanshu Kush, Aleksandar Nikolov, Haohua Tang: "Near Neighbor Search via Efficient Average Distortion Embeddings", CoRR, Vol. abs/2105.04712, 2021.
URL https://arxiv.org/abs/2105.04712

Suppose (M, dist) is a metric space, and P is a finite set of points in M. A function $f : M \to \ell_2$ is called an embedding with average distortion D w.r.t. P, if $\|f(x)-f(y)\|_2 \leq \mathrm{dist}(x,y)$ for all $x, y \in M$, and $\sum_{x\in P}\sum_{y\in P} ||f(x) - f(y)||_2^2 \geq D^{-2}\sum_{x\in P}\sum_{y\in P}\mathrm{dist}(x,y)^2$. Naor [1] showed, that for any d-dimensional normed space $(X, \|\cdot\|)$, defining the metric $\mathrm{dist}(x,y) = \sqrt{||x-y||_X}$, then any $P \subset X$ embeds into ℓ_2 with average distortion $O(\sqrt{\log d})$. Naor's proof proceeds via duality, and does not give an explicit embedding f. Can we find an explicit f given M and P, and, in particular, can we find an f that is efficiently computable from P?

References

1 Naor, Assaf. An average John theorem. arXiv, 1905.01280, 2020

Remote Participants

- Mikkel Abrahamsen
University of Copenhagen, DK
- Pankaj Kumar Agarwal
Duke University – Durham, US
- Helmut Alt
FU Berlin, DE
- Elena Arseneva
St. Petersburg State University, RU
- Édouard Bonnet
ENS – Lyon, FR
- Karl Bringmann
Universität des Saarlandes – Saarbrücken, DE
- Mickaël Buchet
TU Graz, AT
- Maike Buchin
Ruhr-Universität Bochum, DE
- Sergio Cabello
University of Ljubljana, SI
- Hsien-Chih Chang
Dartmouth College – Hanover, US
- Siu-Wing Cheng
HKUST – Kowloon, HK
- Man-Kwun Chiu
FU Berlin, DE
- Jinhee Chun
Tohoku University – Sendai, JP
- Éric Colin de Verdière
University Gustav Eiffel – Marne-la-Vallée, FR
- Jacobus Conradi
Universität Bonn, DE
- Arnaud de Mesmay
University Paris-Est – Marne-la-Vallée, FR
- Anne Driemel
Universität Bonn, DE
- Ioannis Emiris
University of Athens & Athena Research Center, GR
- Matias Korman
Siemens EDA – Wilsonville, US
- Joseph S. B. Mitchell
Stony Brook University, US
- Aleksandar Nikolov
University of Toronto, CA
- Eunin Oh
POSTECH – Pohang, KR
- Steve Y. Oudot
INRIA Saclay – Île-de-France, FR
- Evanthia Papadopoulou
University of Lugano, CH
- Zuzana Patáková
Charles University – Prague, CZ
- Jeff M. Phillips
University of Utah – Salt Lake City, US
- Benjamin Raichel
University of Texas – Dallas, US
- Natan Rubin
Ben Gurion University – Beer Sheva, IL
- Maria Saumell
The Czech Academy of Sciences – Prague & Czech Technical University in Prague, CZ
- Lena Schlipf
Universität Tübingen, DE
- Christiane Schmidt
Linköping University, SE
- Donald Sheehy
North Carolina State University – Raleigh, US
- Kolay Sudeshna
Indian Institute of Technology – Kharagpur, IN
- Martin Tancer
Charles University – Prague, CZ
- Csaba Tóth
California State University – Los Angeles, US
- Uli Wagner
IST Austria – Klosterneuburg, AT
- Carola Wenk
Tulane University – New Orleans, US
- Sue Whitesides
University of Victoria, CA

Report from Dagstuhl Seminar 21192

Approaches and Applications of Inductive Programming

Edited by

Andrew Cropper[1], Luc De Raedt[2], Richard Evans[3], and Ute Schmid[4]

1 **University of Oxford, GB,** andrew.cropper@cs.ox.ac.uk
2 **KU Leuven, BE,** luc.deraedt@cs.kuleuven.be
3 **DeepMind – London, GB,** richardevans@google.com
4 **Universität Bamberg, DE,** ute.schmid@uni-bamberg.de

Abstract

In this report the program and the outcomes of Dagstuhl Seminar 21192 "Approaches and Applications of Inductive Programming" is documented. The goal of inductive programming (IP) is to induce computer programs from data, typically input/output examples of a desired program. IP interests researchers from many areas of computer science, including machine learning, automated reasoning, program verification, and software engineering. Furthermore, IP contributes to research outside computer science, notably in cognitive science, where IP can help build models of human inductive learning and contribute methods for intelligent tutor systems. Building on the success of previous IP Dagstuhl seminars (13502, 15442, 17382, and 19202), the goal of this new edition of the seminar is to focus on IP methods which integrate learning and reasoning, scaling up IP methods to be applicable to more complex real world problems, and to further explore the potential of IP for explainable artificial intelligence (XAI), especially for interactive learning. The extended abstracts included in this report show recent advances in IP research. The included short report of the outcome of the discussion sessions additionally point out interesting interrelation between different aspects and possible new directions for IP.

Seminar May 9–12, 2021 – http://www.dagstuhl.de/21192
2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Machine learning; Software and its engineering → Compilers; Human-centered computing → Human computer interaction (HCI)
Keywords and phrases Interpretable Machine Learning, Explainable Artificial Intelligence, Interactive Learning, Human-like Computing, Inductive Logic Programming
Digital Object Identifier 10.4230/DagRep.11.4.20
Edited in cooperation with Jaimovitch López, Gonzalo

1 Executive Summary

Andrew Cropper
Luc De Raedt
Richard Evans
Ute Schmid

The goal of Inductive Programming (IP) is to provide methods for induction of computer programs from data. Specifically, IP is the automated (or semi-automated) generation of a computer program from an incomplete information, such as input-output examples, demonstrations, or computation traces.IP offers powerful approaches to learning from relational

Approaches and Applications of Inductive Programming, *Dagstuhl Reports*, Vol. 11, Issue 04, pp. 20–33
Editors: Andrew Cropper, Luc De Raedt, Richard Evans, and Ute Schmid

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

data and to learning from observations in the context of autonomous intelligent agents. IP is a form of machine learning, because an IP system should perform better given more data (i.e. more examples or experience). However, in contrast to standard ML approaches, IP approaches typically only need a small number of training examples. Furthermore, induced hypotheses are typically represented as logic or functional programs, and can therefore be inspected by a human. In that sense, IP is a type of interpretable machine learning which goes beyond the expressivity of other approaches of rule learning such as decision tree algorithms. IP is also a form of program synthesis. It complements deductive and transformational approaches. When specific algorithm details are difficult to determine, IP can be used to generate candidate programs from either user-provided data, such as test cases, or from data automatically derived from a formal specification. Most relevant application areas of IP techniques is end-user programming and data wrangling.

This seminar has been the fifth in a series – building on seminars 13502, 15442, 17383, and 19202. In the wake of the recent interest in deep learning approaches, mostly for end-to-end learning, it has been recognized that for practical applications, especially in critical domains, data-intensive blackbox machine learning must be complemented with methods which can help to overcome problems with data quality, missing or errouneous labeling of training data, as well as providing transparency and comprehensibility of learned models. To address these requirements, on the one hand, explainable artificial intelligence (XAI) emerged as a new area of research and on the other hand, there is a new interest in bringing together learning and reasoning. These two areas of research are in the focus of the 2021 seminar. Futhermore, recent developments to scale up IP methods to be more applicable to complex real world domains has been taken into account. Based on outcomes of the fourth seminar (19202), the potential of IP as powerful approach for explainable artificial intelligence ("IP for XAI") has been be elaborated. Bringing together IP methods and deep learning approaches contributes to neural-symbolic intergration research. While two years ago (seminar 19202) focus has been on IP as interpretable surrogate model, in the 2021 seminar explainability of different addressees of explanations and their need to different types of explanations (e.g. verbal or example-based) are considered. For many real world applications, it is necessary to involve the human as teacher and judge for the machine learned models. Therefore, a further topic of the seminar has been to explore IP in the context of new approaches to interactive ML and their applications to automating data science and joint human-computer decision making.

2 Table of Contents

3 Overview of Talks

3.1 Beneficial and harmful explanatory machine learning

Lun Ai (Imperial College London, GB), Mark Gromowski, Céline Hocquette (Imperial College London, GB), Stephen H. Muggleton (Imperial College London, GB), and Ute Schmid (Universität Bamberg, DE)

Main reference Lun Ai, Stephen H. Muggleton, Céline Hocquette, Mark Gromowski, Ute Schmid: "Beneficial and harmful explanatory machine learning", Mach. Learn., Vol. 110(4), pp. 695–721, 2021.
URL https://doi.org/10.1007/s10994-020-05941-0

Given the recent successes of Deep Learning in AI there has been increased interest in the role and need for explanations in machine learned theories. A distinct notion in this context is that of Michie's definition of ultra-strong machine learning (USML). USML is demonstrated by a measurable increase in human performance of a task following provision to the human of a symbolic machine learned theory for task performance. A recent paper demonstrates the beneficial effect of a machine learned logic theory for a classification task, yet no existing work to our knowledge has examined the potential harmfulness of machine's involvement for human comprehension during learning. This paper investigates the explanatory effects of a machine learned theory in the context of simple two person games and proposes a framework for identifying the harmfulness of machine explanations based on the Cognitive Science literature. The approach involves a cognitive window consisting of two quantifiable bounds and it is supported by empirical evidence collected from human trials. Our quantitative and qualitative results indicate that human learning aided by a symbolic machine learned theory which satisfies a cognitive window has achieved significantly higher performance than human self learning. Results also demonstrate that human learning aided by a symbolic machine learned theory that fails to satisfy this window leads to significantly worse performance than unaided human learning.

3.2 A Declarative Framework for Knowledge-Based Explainable Link Analysis

Martin Atzmüller (Universität Osnabrück, DE)

Joint work of Martin Atzmüller, Cicek Guven, Dietmar Seipel

Generating explanations is a prominent topic in artificial intelligence and data science, in order to make methods and systems more transparent, interpretable and understandable for humans. We focus on link analysis: Here, link prediction and anomalous link discovery are challenging, e.g., in cold-start scenarios or when only sparse historic data is available [2, 3]. We discuss how to apply answer set programming (ASP) in a declarative framework for (1) formalizing knowledge-augmented link analysis in feature-rich networks [3], with (2) explanation generation using ASP [1, 4]. We exemplify this via simple link predictors on real-world network datasets.

References

1 Martin Atzmueller, Cicek Guven, Dietmar Seipel (2019) Towards Generating Explanations for ASP-Based Link Analysis using Declarative Program Transformations. Proc. International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2019). CoRR abs/1909.03404
2 Martin Atzmueller and Cicek Guven (2019) A Multi-Perspective View on Model-Based Exceptional Link Analysis on Complex Interaction Networks, NPCS
3 Cicek Guven, Martin Atzmueller (2019) Applying Answer Set Programming for Knowledge-Based Link Prediction on Social Interaction Networks. Frontiers in Big Data
4 Cicek Guven, Dietmar Seipel, Martin Atzmueller (2020) Applying ASP for Knowledge-Based Link Prediction with Explanation Generation in Feature Rich Networks. IEEE Transactions on Network Science and Engineering 8(2):1305-1315

3.3 Inductively inferring human problem solving strategies from observed behavior

Thea Behrens (TU Darmstadt, DE) and Frank Jäkel (TU Darmstadt, DE)

When people solve Sudokus, they can apply many different inference rules. In a series of think-aloud studies we inferred these rules from participants' behavior and their verbalizations manually and implemented them as Prolog programs. In our studies just one general rule would have been enough to fill all cells of a Sudoku puzzle, but we still saw a lot of rule variability and flexibility in our participants. From the data we could estimate the preferences for each rule and each participant. We found that these preferences differ markedly between participants. The rules and rule preferences together form a probabilistic program that is a good description of a participant's problem solving strategy. Unfortunately, in our studies the observed behavior alone was not enough to allow us to infer the rules that participants used and we had to rely on think-aloud data. These natural language data are too unstructured to serve as input to available inductive programming systems. Therefore, we developed a user-interface for solving Sudokus that elicits all the information that participants use when they apply a rule. Our hope is that these new data will allow for a more formal approach to infer the rules that underlie the observed behavior.

3.4 Abductive Knowledge Induction from Raw Data

Wang-Zhou Dai (Imperial College London, GB) and Stephen H. Muggleton (Imperial College London, GB)

Main reference Wang-Zhou Dai, Stephen H. Muggleton: "Abductive Knowledge Induction From Raw Data", CoRR, Vol. abs/2010.03514, 2020.
URL https://arxiv.org/abs/2010.03514

For many reasoning-heavy tasks involving raw inputs, it is challenging to design an appropriate end-to-end learning pipeline. Neuro-Symbolic Learning, divide the process into sub-symbolic perception and symbolic reasoning, trying to utilise data-driven machine learning and

knowledge-driven reasoning simultaneously. However, they suffer from the exponential computational complexity within the interface between these two components, where the sub-symbolic learning model lacks direct supervision, and the symbolic model lacks accurate input facts. Hence, most of them assume the existence of a strong symbolic knowledge base and only learn the perception model while avoiding a crucial problem: where does the knowledge come from? In this paper, we present Abductive Meta-Interpretive Learning (MetaAbd) that unites abduction and induction to learn neural networks and induce logic theories jointly from raw data. Experimental results demonstrate that MetaAbd not only outperforms the compared systems in predictive accuracy and data efficiency but also induces logic programs that can be re-used as background knowledge in subsequent learning tasks. To the best of our knowledge, MetaAbd is the first system that can jointly learn neural networks from scratch and induce recursive first-order logic theories with predicate invention.

3.5 From Statistical Relational to Neuro-Symbolic AI

Luc De Raedt (KU Leuven, BE)

Joint work of Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, Giuseppe Marra
Main reference Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, Giuseppe Marra: "From Statistical Relational to Neuro-Symbolic Artificial Intelligence", in Proc. of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 4943–4950, ijcai.org, 2020.
URL https://doi.org/10.24963/ijcai.2020/688

Neural-symbolic and statistical relational artificial intelligence both integrate frameworks for learning with logical reasoning. This survey identifies several parallels across seven different dimensions between these two fields. These cannot only be used to characterize and position neural-symbolic artificial intelligence approaches but also to identify a number of directions for further research.

3.6 Knowledge Refactoring for Inductive Program Synthesis

Sebastijan Dumancic (KU Leuven, BE), Andrew Cropper (University of Oxford, GB)

Joint work of Sebastijan Dumancic, Andrew Cropper, Tias Guns
Main reference Sebastijan Dumancic, Tias Guns, Andrew Cropper: "Knowledge Refactoring for Inductive Program Synthesis", Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35(8), pp. 7271–7278, 2021.
URL https://ojs.aaai.org/index.php/AAAI/article/view/16893

Humans constantly restructure knowledge to use it more efficiently. Our goal is to give a machine learning system similar abilities so that it can learn more efficiently. We introduce the knowledge refactoring problem, where the goal is to restructure a learner's knowledge base to reduce its size and to minimise redundancy in it. We focus on inductive logic programming, where the knowledge base is a logic program. We introduce Knorf, a system that solves the refactoring problem using constraint optimisation. We evaluate our approach on two program induction domains: real-world string transformations and building Lego structures. Our experiments show that learning from refactored knowledge can improve predictive accuracies fourfold and significantly reduce learning times.

3.7 On Conditional Teaching Size and Minimal Curricula

Manuel Garcia-Piqueras (University of Castilla-La Mancha, ES) and José Hernández-Orallo (Technical University of Valencia, ES)

Main reference Jan Arne Telle, José Hernández-Orallo, Cèsar Ferri: "The teaching size: computable teachers and learners for universal languages", Mach. Learn., Vol. 108(8-9), pp. 1653–1675, 2019.
URL https://doi.org/10.1007/s10994-019-05821-2

Machine teaching, under certain integrative prior knowledge, enables the instruction of any concept expressed in a universal language. Latest experiments show that there are instructional sets surprisingly shorter than the concept description itself [1]. We delineate a border for these remarkable experimental findings through *teaching size* and concept complexity. Also, we study teaching curricula and find a new phenomenon that we call *interposition*: certain prior knowledge generates simpler compatible concepts which increase the teaching size of the concept that we want to teach. Far beyond, we provide an algorithm which builds *optimal curricula* based on interposition avoidance. These results reveal innovative curriculum design strategies for machines, but also for animals and humans.

References

1 Telle, Jan Arne and Hernández-Orallo, José and Ferri, Cèsar. The teaching size: computable teachers and learners for universal languages, Machine Learning, 108(8), 1653–1675 (2019).

3.8 IP vs Humans: Learning from Machine Teaching Examples

Gonzalo Jaimovitch López (Technical University of Valencia, ES), Cesar Ferri Ramirez (Technical University of Valencia, ES), and José Hernández-Orallo (Technical University of Valencia, ES)

Main reference Gonzalo Eduardo Jaimovitch López: "Comparison between machine learning and human learning from examples generated with machine teaching", 2020.
URL http://hdl.handle.net/10251/152771

Inductive programming has been singled out as one important approach to learning, where background knowledge and simplicity priors together play a key role to infer patterns, including algorithmic ones, from very few examples. Other machine learning techniques, especially deep learning, require thousands, if not millions of examples, to reach the same inference. This duality seems to be challenged by new massive models based on transformers that are able to be pretrained from large datasets. These models capture vast amounts of knowledge with very abstract representations and then make inferences from very few examples, with little tuning or no retraining needed. In particular, large language models have shown an impressive ability for few-shot learning. It seems relevant to ask now what kind of patterns these models can capture and how many examples they need in their prompts. We present this question as a machine teaching problem with strong priors [1, 2], and test whether language models can learn simple algorithmic concepts from small witness sets. In particular, we explore how several GPT architectures, inductive programming systems (the inductive functional programming system MagicHaskeller and the inductive logic programming system Louise) and humans perform in terms of the complexity of the concept and the number of examples provided, and how much their behaviour diverge [3]. This first joint analysis of

machine teaching and language models can address key questions for artificial intelligence and machine learning, such as whether strong priors, and Occam's razor in particular, can be distilled from data, making learning from a few examples possible without the need of providing domain knowledge or common sense knowledge about the world.

References

1 Telle, J. A., Hernández-Orallo, J., & Ferri, C. (2019). *The teaching size: computable teachers and learners for universal languages.* Machine Learning, 108(8), 1653-1675.

2 Hernández-Orallo, J., & Telle, J. A. (2020). *Finite and Confident Teaching in Expectation: Sampling from Infinite Concept Classes.* In 24th European Conference on Artificial Intelligence (ECAI2020)

3 Jaimovitch López, G. E. (2020). *Comparison between machine learning and human learning from examples generated with machine teaching* (Degree disertation). Universitat Politècnica de València

3.9 Analyzing massive biomedical datasets with graph-based rule mining for drug repurposing

Tomáš Kliegr (University of Economics – Prague, CZ)

Main reference Václav Zeman, Tomáš Kliegr, Vojtech Svátek: "RDFRules: Making RDF rule mining easier and even more efficient", Semantic Web, Vol. 12(4), pp. 569–602, 2021.
URL https://doi.org/10.3233/SW-200413

Large biomedical datasets containing structured data could contain hidden knowledge useful for the response to the pandemic. In this talk, we present our attempt to apply a graph-based rule mining system to the KG-Covid-19 dataset [2]. This RDF knowledge graph is a result of ingestion of multiple specialized knowledge sources, such as DrugBank, as well as extraction from COVID-19 research literature selected for their relevance for drug repurposing efforts. The published version of the dataset contains 377,482 nodes and 21,433,063 edges. Due to the large size of the dataset, we initially chose the AMIE+ rule mining algorithm, which was shown to be orders of magnitude faster than the previous approaches. Due to the need to search for highly specific patterns, our initial attempts to apply "vanilla" AMIE+ were not successful due to the combinatorial explosion associated with mining of low support rules and associated memory issues. We also experimented with AnyBURL [3], which had excellent performance, but also lacked the possibility to finely define the sought pattern. We finally settled on RDFRules [5], which is a comprehensive set of extensions for the AMIE+ that also includes the possibility to apply fine-grained patterns for constraining the search space. Using RDFRules, it was possible to find rules with low support even on the full KG-Covid-19 dataset (without metadata) on a single computer using less than 64 GB of RAM. As an example use case, we described a mining task performed in Summer 2020 to Fall 2020, using a Spring 2020 release of KG-Covid-19 as described in detail in [4]. This task aimed to find drugs that "molecularly interact with" the ACE 1 receptor (UniProtKB ID P12821) and at the same time they are connected through an arbitrary predicate to an intermediary resource, which is using the "interact with" predicate connected to the ACE 2 receptor (UniProtKB ID Q9BYF1). Note that "molecularly interact with" and "interact with" are Biolink Model predicates. The task was thus to find rules complying to the following RDFRules pattern: (<Any> <interacts_with> <Q9BYF1>) ∧ [(<Any>

<Any> <Any>)] ⇒ (<Any> <molecularly_interacts_with> <P12821>). This task executed with minsupp= 1 and mining with constants led to the discovery of a single logical rule (?b <interacts_with> <Q9BYF1>) ∧ (?a <molecularly_interacts_with> ?b) → (?a <molecularly_interacts_with> <P12821>), whose instantiation resulted in five drugs. These included a widely used antihypertensive drug "Telmisartan" for which a subsequent literature search showed that it is a viable drug repurposing target. A recent open multicenter randomized clinical trial has shown that Telmisartan could, through anti-inflammatory effects, reduce mortality and morbidity in hospitalized patients infected with SARS-CoV-2 [1]. We conclude that rule mining is a viable approach for finding "nuggets" in large knowledge graphs. Our work was limited in that we have not explored the possibility to use the embeddings-based approaches and we have not performed comparison with more direct methods of analyzing graph data.

References

1 Duarte, M., Pelorosso, F., Nicolosi, L. N., Salgado, M. V., Vetulli, H., Aquieri, A., ... & Rothlin, R. P. (2021). *Telmisartan for treatment of Covid-19 patients: An open multicenter randomized clinical trial.* EClinicalMedicine, 37, 100962.

2 Reese, J. T., Unni, D., Callahan, T. J., Cappelletti, L., Ravanmehr, V., Carbon, S., ... & Mungall, C. J. (2021). *KG-COVID-19: A framework to produce customized knowledge graphs for COVID-19 response.* Patterns, 2(1), 100155.

3 Meilicke, C., Chekol, M. W., Ruffinelli, D., & Stuckenschmidt, H. (2019). *Anytime Bottom-Up Rule Learning for Knowledge Graph Completion.* In IJCAI (pp. 3137-3143).

4 Šimečková, J. *Extrakce pravidel ze znalostních grafů.* Bachelor thesis. University of Economics, Prague, 2020

5 Zeman, Václav, Tomáš Kliegr, and Vojtěch Svátek. *RDFRules: Making RDF rule mining easier and even more efficient.* Semantic Web. 2021

3.10 Towards Robust, Data-Efficient, and Explainable Deep Learning

Pasquale Minervini (University College London, GB)

Deep Learning models are a class of Machine Learning models that use multiple processing layers to progressively extract higher-level features from raw inputs. Over the past decade, it has become one of the most impactful research areas in Artificial Intelligence, with many notable commercially important applications. However, Deep Learning models still fall short in terms of data efficiency, out-of-distribution generalisation, interpretability, and complexity. We discuss several ways of overcoming such limitations, by increasing their statistical robustness [1, 2, 3], incorporating prior knowledge [4, 5, 6], combining symbolic and sub-symbolic computation models [7, 8, 9, 10], and developing more computationally efficient neural models [11, 12, 13].

References

1 Joe Stacey, Pasquale Minervini, Haim Dubossarsky, Sebastian Riedel, Tim Rocktäschel: Avoiding the Hypothesis-Only Bias in Natural Language Inference via Ensemble Adversarial Training. EMNLP (1) 2020: 8281-8291

2 Johannes Welbl, Pasquale Minervini, Max Bartolo, Pontus Stenetorp, Sebastian Riedel: Undersensitivity in Neural Reading Comprehension. EMNLP (Findings) 2020: 1152-1165

3 Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, Phil Blunsom: Make Up Your Mind! Adversarial Generation of Inconsistent Natural Language Explanations. ACL 2020: 4157-4165
4 Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, Sebastian Riedel: Adversarial Sets for Regularising Neural Link Predictors. UAI 2017
5 Pasquale Minervini, Luca Costabello, Emir Muñoz, Vít Novácek, Pierre-Yves Vandenbussche: Regularizing Knowledge Graph Embeddings via Equivalence and Inversion Axioms. ECML/PKDD (1) 2017: 668-683
6 Pasquale Minervini, Sebastian Riedel: Adversarially Regularising Neural NLI Models to Integrate Logical Background Knowledge. CoNLL 2018: 65-74
7 Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, Tim Rocktäschel: NLProlog: Reasoning with Weak Unification for Question Answering in Natural Language. ACL (1) 2019: 6151-6161
8 Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, Edward Grefenstette: Differentiable Reasoning on Large Knowledge Bases and Natural Language. AAAI 2020: 5182-5190
9 Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, Tim Rocktäschel: Learning Reasoning Strategies in End-to-End Differentiable Proving. ICML 2020: 6938-6949
10 Erik Arakelyan, Daniel Daza, Pasquale Minervini, Michael Cochez: Complex Query Answering with Neural Link Predictors. ICLR 2021
11 Yuxiang Wu, Sebastian Riedel, Pasquale Minervini, Pontus Stenetorp: Don't Read Too Much Into It: Adaptive Computation for Open-Domain Question Answering. EMNLP (1) 2020: 3029-3039
12 Yuxiang Wu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel: Training Adaptive Computation for Open-Domain Question Answeringwith Computational Constraints. ACL 2021
13 Patrick S. H. Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, Sebastian Riedel: PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them. CoRR abs/2102.07033 (2021)

3.11 Advances in Meta-Interpretive Learning/ILP and Cognitive Artificial Intelligence

Stephen H. Muggleton (Imperial College London, GB)

Joint work of Stephen Muggleton, Andrew Cropper, Sebastijan Dumancic, Richard Evans
Main reference Andrew Cropper, Sebastijan Dumancic, Stephen H. Muggleton: "Turning 30: New Ideas in Inductive Logic Programming", in Proc. of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 4833–4839, ijcai.org, 2020.
URL https://doi.org/10.24963/ijcai.2020/673

Inductive logic programming (ILP) is a form of logic-based machine learning. The goal of ILP is to induce a hypothesis (a logic program) that generalises given training examples and background knowledge. As ILP turns 30, we survey recent work in the field. In this survey, we focus on (i) new meta-level search methods, (ii) techniques for learning recursive programs that generalise from few examples, (iii) new approaches for predicate invention, and (iv) the use of different technologies, notably answer set programming and neural networks. We conclude by discussing some of the current limitations of ILP and discuss directions for future research.

3.12 Generating Contrastive Explanations for Inductive Logic Programming Based on a Near Miss Approach

Johannes Rabold (Universität Bamberg, DE), Ute Schmid (Universität Bamberg, DE)

Joint work of Johannes Rabold, Ute Schmid, Michael Siebers

In recent research, human-understandable explanations of machine learning models have received a lot of attention. Often explanations are given in the form of model simplifications or visualizations. However, as shown in cognitive science as well as in early AI research, concept understanding can also be improved by aligning a given instance for a concept with a similar counterexample. Contrasting a given instance with a structurally similar example which does not belong to the concept highlights what characteristics are necessary for concept membership. Such near misses have been proposed by Winston (1970) as efficient guidance for learning in relational domains. We introduce an explanation generation algorithm for relational concepts learned with Inductive Logic Programming (GENME). The algorithm identifies near miss examples from a given set of instances and ranks them by their degree of closeness to a specific positive instance. A modified rule which covers the near miss but not the original instance is given as an explanation. We illustrate GENME with the well known family domain consisting of kinship relations, the visual relational Winston arches domain and a real-world domain dealing with file management. We also present a psychological experiment comparing human preferences of rule-based, example-based, and near miss explanations in the family and the arches domains.

3.13 Learning Episodic Memory Retrieval Procedures Using First-Order Ripple-Down Rules

Claude Sammut (UNSW – Sydney, AU)

Joint work of Claude Sammut, Colm Flanagan, Eric Martin, Michael Bain

The long term memory of an agent can be separated into three categories: procedural, declarative and episodic [4]. The distinguishing features of episodic memories is that they store knowledge of specific events, including contextual information such, time, location and participating agents. It is useful for an agent, such as a robot, to be able to recall past events that are similar to one that has just been observed. The problem here is to define similarity and how to retrieve similar events. A typical approach taken in many case-based reasoning systems [3, 6, 5] is to create a geometric distance measure, where the dimensions in the event space are the observed features. Devising such as measure becomes difficult when the observations are complex, as is the case for a robot operating a complex environment such as a real home or work place or in search and rescue operations. Flanagan [2] describes a system that learns as matching procedure that is customised for each type of event and is capable of matching structured object descriptions. It associates a Ripple-Down Rule [1]

References

1. Compton, P. and Kang, B. (2021). *RIPPLE-DOWN RULES: The Alternative to Machine Learning*. Chapman and Hall/CRC.
2. Flanagan, C. (2021). *Episodic and Semantic Memory in Cognitive Robots*. Phd thesis, School of Computer Science and Engineering, The University of New South Wales.
3. Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, San Mateo.
4. Laird, J. E., Kinkade, K. R., Mohan, S., and Xu, J. Z. (2012). Cognitive Robotics Using the Soar Cognitive Architecture. In *8th International Conference on Cognitive Robotics, (Cognitive Robotics Workshop, Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 46–54, Toronto. ACM Press.
5. Riesbeck, C. and Schank, R. (1989). *Inside Case-based Reasoning*. Erlbaum, Northvale, NJ.
6. Veloso, M. and Aamodt, A., editors (1995). *Case-Based Reasoning Research and Development: Proceedings of the First International Conference on Case-Based Reasoning*, Berlin. Springer Verlag.

3.14 Ultra-strong machine learning with explanatory dialogs

Ute Schmid (Universität Bamberg, DE)

Joint work of Bettina Finzel, Ute Schmid

At Dagstuhl AAIP 2017, Stephen Muggelton reminded us about Donald Michie's criteria which require a machine learning system (1) to show improved predictive performance with increasing amounts of data (weak criterion), (2) additionally to present the learned model in symbolic –human understandable– form (strong criterion), which (3) teach the model to a human in such a way that the performance is increased to a level beyond that of the human studying the training data alone (ultra-strong criterion). Michie's ultra-strong criterion corresponds to the main claim of current research on explainable AI (XAI). Our discussion at Dagstuhl inspired us to empirically research whether models learned with inductive logic programming (ILP) fulfill the ultra-strong criterion [1, 2]. We extended this work to applying ILP to teaching best moves in game playing be generating verbal explanations from the learned Prolog rules [3]. However, in this work, explanations are given once and in one specific way. In contrast, when one human teaches another, explanations are often a process based on a dialog between generator and receiver of the explanation. Currently, we realize an approach to generate such explanatory dialogs from reasoning traces from Prolog and combine such verbal explanations with prototypes and near miss examples.

References

1. Ute Schmid, Christina Zeller, Tarek R. Besold, Alireza Tamaddoni-Nezhad, Stephen Muggleton: How Does Predicate Invention Affect Human Comprehensibility? ILP 2016: 52-67
2. Stephen H. Muggleton, Ute Schmid, Christina Zeller, Alireza Tamaddoni-Nezhad, Tarek R. Besold: Ultra-Strong Machine Learning: comprehensibility of programs learned with ILP. Mach. Learn. 107(7): 1119-1140 (2018)
3. Lun Ai, Stephen H. Muggleton, Céline Hocquette, Mark Gromowski, Ute Schmid: Beneficial and harmful explanatory machine learning. Mach. Learn. 110(4): 695-721 (2021)

3.15 Generalized Planning as Heuristic Search

Javier Segovia-Aguas (UPF – Barcelona, ES)

Joint work of Javier Segovia-Aguas, Sergio Jiménez, Anders Jonsson
Main reference Javier Segovia Aguas, Sergio Jiménez Celorrio, Anders Jonsson: "Generalized Planning as Heuristic Search", CoRR, Vol. abs/2103.14434, 2021.
URL https://arxiv.org/abs/2103.14434

Generalized planning is a booming research topic in the automated planning community, which aims at computing algorithm-like plans, e.g. plans that branch and loop, that solve a possibly infinite set of planning instances of a given domain. Generalized planning is then a fascinating meeting point for automated planning and program synthesis since both pursue a well-founded integration of (i), knowledge representation with human-comprehensible models (ii), model-based reasoning and (iii), the learning of such models from examples. In this work we show that the heuristic search paradigm, that has traditionally shown successful for classical planning, applies also to the computation of generalized plans using a Random Access Machine, a best-first search algorithm, and different evaluation/heuristic functions for guiding the search in a tractable (though combinatorial) solution space. We believe this is a promising research direction for achieving a tighter integration of the representation, reasoning and learning facets in Artificial Intelligence.

Remote Participants

- Lun Ai
Imperial College London, GB
- Nada Amin
Harvard University – Allston, US
- Martin Atzmüller
Universität Osnabrück, DE
- Feryal Behbahani
DeepMind – London, GB
- Thea Behrens
TU Darmstadt, DE
- Andrew Cropper
University of Oxford, GB
- James Cussens
University of Bristol, GB
- Wang-Zhou Dai
Imperial College London, GB
- Luc De Raedt
KU Leuven, BE
- Thomas Demeester
Ghent University, BE
- Amit Dhurandhar
IBM TJ Watson Research Center – Yorktown Heights, US
- Sebastijan Dumancic
KU Leuven, BE
- Kevin Ellis
Cornell University – Ithaca, US
- Richard Evans
DeepMind – London, GB
- Cesar Ferri Ramirez
Technical University of Valencia, ES
- Bettina Finzel
Universität Bamberg, DE
- Peter Flach
University of Bristol, GB
- Johannes Fürnkranz
Johannes Kepler Universität Linz, AT
- Artur Garcez
City – University of London, GB
- Manuel Garcia-Piqueras
University of Castilla-La Mancha, ES
- José Hernández-Orallo
Technical University of Valencia, ES
- Céline Hocquette
Imperial College London, GB
- Frank Jäkel
TU Darmstadt, DE
- Gonzalo Jaimovitch López
Technical University of Valencia, ES
- Susumu Katayama
University of Miyazaki, JP
- Tomáš Kliegr
University of Economics – Prague, CZ
- Stefan Kramer
Universität Mainz, DE
- Maithilee Kunda
Vanderbilt University, US
- Sara Magliacane
University of Amsterdam, NL
- Roman Manevich
Facebook – London, GB
- Fernando Martinez-Plumed
European Commission – Sevilla, ES
- Pasquale Minervini
University College London, GB
- Stephen H. Muggleton
Imperial College London, GB
- Stassa Patsantzis
Imperial College London, GB
- Johannes Rabold
Universität Bamberg, DE
- Claude Sammut
UNSW – Sydney, AU
- Stephan Scheele
Universität Bamberg, DE
- Ute Schmid
Universität Bamberg, DE
- Javier Segovia-Aguas
UPF – Barcelona, ES
- Gustavo Soares
Microsoft Corporation – Redmond, US
- Stefano Teso
University of Trento, IT
- Jan Tinapp
bidt – München, DE

Report from Dagstuhl Seminar 21201

Serverless Computing

Edited by

Cristina Abad[1], Ian T. Foster[2], Nikolas Herbst[3], and Alexandru Iosup[4]

1 **ESPOL – Guayaquil, EC,** cristina.abad@gmail.com
2 **Argonne National Laboratory – Lemont, US,** foster@anl.gov
3 **Universität Würzburg, DE,** nikolas.herbst@uni-wuerzburg.de
4 **VU University Amsterdam, NL,** alexandru.iosup@gmail.com

Abstract

In the backbone of our digital society, cloud computing enables an efficient, utility-like ecosystem of developing, composing, and providing software services. Responding to a trend to make cloud computing services more accessible, fine-grained, and affordable, *serverless computing* has gained rapid adoption in practice, and garnered much interest from both industry and academia.

However successful, serverless computing manifests today the opportunities and challenges of emerging technology: a rapidly growing field but scattered vision, plenty of new technologies but no coherent approach to design solutions from them, many simple applications but no impressive advanced solution, the emergence of a cloud continuum (resources from datacenters to the edge) but no clear path to leverage it efficiently, and overall much need but also much technical complexity.

Several related but disjoint fields, notably software and systems engineering, parallel and distributed systems, and system and performance analysis and modeling, aim to address these opportunities and challenges. Excellent collaboration between these fields in the next decade will be critical in establishing serverless computing as a viable technology.

We organized this Dagstuhl seminar to bring together researchers, developers, and practitioners across disciplines in serverless computing, to develop a vision and detailed answers to the timely and relevant, open challenges related to the following topics:

- Topic 1: design decisions for serverless systems, platforms, and ecosystems,
- Topic 2: software engineering of serverless applications, but also systems, platforms, and ecosystems
- Topic 3: applications and domain requirements for serverless computing,
- Topic 4: evaluation of serverless solutions,

and beyond (privacy, cyber-physical systems, etc.).

In this document, we report on the outcomes of Dagstuhl Seminar 21201 "Serverless Computing" by integrating diverse views and synthesizing a shared vision for the next decade of serverless computing.

Seminar May 16–21, 2021 – http://www.dagstuhl.de/21201

2012 ACM Subject Classification General and reference → Performance; Computer systems organization → Cloud computing; Computer systems organization → Grid computing; Software and its engineering → Distributed systems organizing principles; Software and its engineering → Software organization and properties

Keywords and phrases Cloud computing, Cloud continuum, data-driven, design patterns, DevOps, experimentation, model-driven, serverless computing, simulation, software architecture, systems management, vision

Digital Object Identifier 10.4230/DagRep.11.4.34

1 Executive Summary

Cristina Abad (ESPOL – Guayaquil, EC)
Ian T. Foster (Argonne National Laboratory – Lemont, US)
Nikolas Herbst (Universität Würzburg, DE)
Alexandru Iosup (VU University Amsterdam, NL)

Serverless computing holds a significant promise for the modern, digital society. For the past seven decades, our society has increasingly required ever-cheaper, ever-more convenient, and ever-faster computing technology. In the late-1950s, leasing time on an IBM 701 cost $15,000 per month ($135,000 in 2020 dollars). Today, we can lease many times this computing power for mere pennies but need to be careful about the actual cost of doing so. Cloud computing, that is, the utility providing IT as a service, on-demand and pay-per-use, is a widely used computing paradigm that offers large economies of scale and promises extreme environmental efficiency. Born from a need to make cloud computing services more accessible, fine-grained, and affordable, *serverless computing* has garnered interest from both industry and academia. In our vision, serverless computing can meet this need, but to do this it will have to overcome its current status of emergent technology or risk its demise.

Cloud computing is already an established technology. Today, more than three-quarters of the US and European companies, and many private individuals, use cloud computing services[1]. The serverless market is blooming[2] and has already exceeded $200 billion in 2020[3]. The cost of one hour on a cloud computer leased on-demand can be lower than a cent[4] and all the major cloud providers offer inexpensive access to diverse and state-of-the-art hardware. However cheap, cloud computing still poses daunting operational challenges to software professionals, in particular, how to manage the selection, operation, and other aspects of using cloud infrastructure (in short, *servers*). Correspondingly, it poses significant challenges to systems designers and administrators, related to keeping the cloud infrastructure efficient and sustainable.

An emerging class of cloud-based software architectures, *serverless computing*, focuses on providing software professionals the ability to execute arbitrary functions with low or even no overhead in server management. Serverless computing leverages recent developments in the miniaturization of software parts through *microservice-based architectures*, in the operationalization of small self-contained execution units through *containers*, and in their integration in service models such as *Function-as-a-Service (FaaS)*. Truly, serverless is more [12]. Early research successes [6, 15, 17, 18, 22] complement numerous industrial applications [9], from business-critical to scientific computing, from DevOps to side-tasks. Already, IT spending on serverless computing should exceed $8 billion per year, by 2021.[5]

However promising, serverless computing has yet to mature and presents many hard, open challenges. There are numerous signs and reports [11, 14] that serverless computing poses critical challenges in software engineering, parallel and distributed systems operation,

[1] European Commission, Uptake of Cloud in Europe, Digital Agenda for Europe report by the Publications Office of the European Union, Luxembourg, Sep 2014. and Flexera, State of the Cloud Report, 2020.
[2] Gartner Inc. Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17% in 2020. Press Release.
[3] Frank Gens. Worldwide and Regional Public IT Cloud Services 2019–2023 Forecast. Tech. Rep. by IDC, Doc. #US44202119, Aug 2019.
[4] Amazon AWS, Microsoft Azure, and Google Compute Engine offer VMs in this price range.
[5] "Function-as-a-Service Market - Global Forecast to 2021," marketsandmarkets.com, Feb 2017.

and performance engineering [10]. For example, software engineering could help overcome challenges in the developer experience [23], including testing, tooling, functionality, and training and education. The systems side requires, among others, new approaches for deployment, monitoring, and general operation, and also specific advances in security, cost predictability, and life-cycle management for cloud functions. Performance engineering raises many hard aspects, such as performance optimization, engineering for cost-efficiency, and various forms of fast online scheduling. These combined challenges are distinctive from the general challenges of cloud computing, for example, because the fine-grained, often event-driven nature of serverless computing typically requires approaches that are lightweight and able to respond without delay.

The goal of the seminar is to *combine the views of a diverse and high-quality group of researchers spanning three disciplines*: software engineering, parallel and distributed systems, and performance engineering. The Dagstuhl Seminar will be a catalyst. Attendees discussed the open challenges and opportunities of serverless computing for the next decade, with a focus on at least the following crucial aspects and questions:

- Envision serverless systems and applications in the next decade. How to leverage the freedom from operational concerns? How to overcome the challenge and enjoy the benefits of fine granularity?
- How to properly engineer serverless software and systems? What are the emerging architectural patterns for serverless systems and applications? How to test and debug serverless systems and applications?
- How to characterize, model, and analyze serverless systems and applications? How to understand the diverse serverless workloads?
- How to manage the resources used in serverless operations? How to schedule and orchestrate in this environment? How to manage specific application classes, such as computer vision, enterprise workflows, HPC, DevOps?
- How to deploy and manage the full lifecycle of serverless applications? How to add ML-capabilities to feedback loops? How to break through the operational silos?
- How to support privacy, security, dependability, and other desirable operational properties for serverless applications and systems?
- Beyond computer systems, how to consider serverless systems and applications from a holistic, cyberphysical perspective?

Core topics

The seminar focussed on the following key topics related to serverless computing:

Topic 1. Design decisions for serverless systems, platforms, and ecosystems. As the serverless model is increasingly being adopted in industry [9], the challenges of properly designing these systems and the platforms on which they run are becoming more apparent. These challenges include important problems [10], such as: how to reduce the serverless overhead added by the platform to the (commonly lightweight) functions representing the business logic of the application (e.g., see [20]), how to ensure proper performance isolation while making efficient use of the shared infrastructure (e.g., see [1]), how to partition the functions [5, 6], and how to properly schedule functions and route requests to these functions (e.g., see [2]), in such a way that the service level objectives (SLO's) are adequately met, among other important challenges. There is also the question of running serverless workloads alongside conventional applications, e.g., HPC, big data, machine learning. The experiences

of the attendees to the seminar, some of which have already started working in these domain and others with established experience in prior technologies from which we may learn and transfer knowledge (e.g., grid computing), will enable us to focus on determining which of these decisions the community should be focusing on, and how to establish adequately prioritized research agendas.

Topic 2. Software engineering of serverless applications, but also systems, platforms, and ecosystems. To increase the domain of application for serverless computing, the functionality it can express needs to become increasingly more complex, which contrasts with the perceived simplicity of the model [23]. What is the trade-off between simplicity and expressiveness? Which composition models can ensure that serverless workflows can be maintained and developed (and updated) long term? Serverless functions should become increasingly interoperable, and applications should become able to leverage the services of any serverless platform [6]. How to make serverless functions vendor-agnostic and how to run serverless applications across cloud federations? Which architectural patterns are useful for serverless applications? How to consider and support the legacy part of serverless applications? The development processes, from the macro-view of how teams coordinate to deliver applications that could operate in an external ecosystem, to the micro-view of how to develop and test a serverless function, will have to consider the new aspects raised by serverless computing. What are effective development processes? What tools and IDE features are needed? What versioning and testing, and what CI/CD protocols should be used? How to evolve legacy software toward serverless-native applications? How to ensure open-source software becomes FAIR software [13]?

Topic 3. Applications and domain requirements for serverless computing. Preliminary studies of serverless applications at large [9] have shown that there is a wide variety of scenarios for which industry and academia are adopting serverless approaches. From business-critical workloads, to automating DevOps, scientific computing, and beyond, the diversity of the applications and domains for which serverless is being applied poses significant challenges when attempting to optimally manage the resources and infrastructure on which these applications depend. It is important to properly understand the variety of these applications and domain requirements, engaging both academia and industry in the discussion.

These requirements should relate to various aspects in software engineering, parallel and distributed systems, and performance engineering. For example, a domain-based approach could help increase scalability [3]; considering the structure of packages in composing a deployable serverless application could improve scheduling performance [2]; and serverless functions and architectures should be considered during performance tests [8, 28].

Topic 4. Evaluation of serverless computing systems, platforms, and ecosystems. The performance trade-offs of serverless systems are not yet well understood [28], thus highlighting the importance of proper evaluation and benchmarking of these systems. However, the high level of abstraction and the opaqueness of the operational-side make evaluating these platforms particularly challenging. As recent efforts are starting to focus on this topic [24, 28], it is important to engage the community on an early discussion on the best approaches to tackle this problem. How to understand and engineer the performance of serverless systems? How to translate the findings, when serverless systems are opened to external developers (as platforms) or take part in much larger systems of systems (and even ecosystems)? How to account for parts of the ecosystem being closed-source and even acting as black-boxes? How to identify and even explain the performance bottlenecks such systems, platforms, and

ecosystems experience? How to use evaluation results with other performance engineering techniques to control and improve the performance of serverless systems, platforms, and ecosystems?

An important focus of inquiry has recently become prominent in computer systems research: the reproducibility of evaluation results and of experiments in general [19, 21]. Not doing so can result in misleading results [26], and in results that cannot be obtained again [25] sometimes even under identical circumstances and by their original authors [7]. This leads to a possible loss of faith in the entire field [4, 27]. "How to benchmark serverless solutions reproducibly?" is an important question to address with diverse expertise and fresh ideas.

Synopsis and Planned Actions

We would like to thank the Dagstuhl staff and sponsors for this unique seminar opportunity even under the constraints of the pandemic. During the seminar, we had almost 45h of online meetings (not counting sub-meetings): some 9-10h of online meetings each seminar day. Three 3h sessions per day were spread around the clock to allow participation from various timezones. Even under these constraints, we experienced enormous participation and active discussion involvement. In brief, the seminar week was structured as follows:

After each participant presented her/himself to the plenary, we formed four working groups according to the topics above. The discussions were kick-started by four distinguished keynotes, in plenary, with the respective talk abstracts included in this report:

- "Serverless Predictions: 2021-2030" given jointly by Pedro García López (Universitat Rovira i Virgili – Tarragona, ES) and Bernard Metzler (IBM Research-Zurich, CH)
- "Developer Experience for Serverless: Challenges and Opportunities" given by Robert Chatley (Imperial College London, GB)
- "Federated Function as a Service" given jointly by Kyle Chard (University of Chicago, US) and Ian T. Foster (Argonne National Laboratory – Lemont, US)
- "Characterizing Serverless Systems" given by Mohammad Shahrad (University of British Columbia – Vancouver, CA)

Each of the four working groups held five 3h sessions with their teams, including three 1h one-on-one meetings with the other groups. The four working groups report individually on their outcomes and list identified research challenges. In a consolidation phase, we identified and planned nine focused topics for future joint research among the participants.

Complemented by a Slack workspace for the seminar participants, a focused continuation of discussions beyond the seminar week was enabled: Among others, a discussion initiated and led by Samuel Kounev on the notion of serverless computing, started during the seminar, continued well beyond. We include the outcome of this "panel discussion" in Section 5.1 of this report.

The organizers and participants decided to jointly work toward at least one high-profile magazine article reporting on the seminar outcome and research agenda.

Furthermore, during the seminar the motion was raised to establish a conference series on serverless computing. We see good potential for a new conference on "Serverless Software and Systems" as a cross-community event embracing, at least, the disciplines of software engineering, system engineering, and performance engineering. Working potentially in concert with an existing workshop series in the field, we plan to initiate this step in the coming months. We hope that one day in the future, we can proudly look back and say that this Dagstuhl seminar 21201 was an important trigger event.

References

1. Z. Al-Ali, S. Goodarzy, E. Hunter, S. Ha, R. Han, E. Keller, and E. Rozner. Making serverless computing more serverless. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018.
2. G. Aumala, E. Boza, L. Ortiz-Avilés, G. Totoy, and C. Abad. Beyond load balancing: Package-aware scheduling for serverless platforms. In 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2019.
3. Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, André van Hoorn, Henning Schulz, Daniel S. Menasché, and Vilc Queupe Rufino. Scalability assessment of microservice architecture deployment configurations: A domain-based approach leveraging operational profiles and load tests. J. Syst. Softw., 165:110564, 2020.
4. Monya Baker. Is there a reproducibility crisis? Nature, 533(7604):452–454, 2016.
5. Edwin F. Boza, Xavier Andrade, Jorge Cedeno, Jorge R. Murillo, Harold Aragon, Cristina L. Abad, and Andres G. Abad. On implementing autonomic systems with a serverless computing approach: The case of self-partitioning cloud caches. Comput., 9(1):14, 2020.
6. Ryan Chard, Yadu N. Babuji, Zhuozhao Li, Tyler J. Skluzacek, Anna Woodard, Ben Blaiszik, Ian T. Foster, and Kyle Chard. funcx: A federated function serving fabric for science. In HPDC '20: The 29th International Symposium on High-Performance Parallel and Distributed Computing, pages 65–76. ACM, 2020.
7. Christian S. Collberg and Todd A. Proebsting. Repeatability in computer systems research. Commun. ACM, 59(3):62–69, 2016.
8. Simon Eismann, Cor-Paul Bezemer, Weiyi Shang, Dusan Okanovic, and André van Hoorn. Microservices: A performance tester's dream or nightmare? In ICPE '20: ACM/SPEC International Conference on Performance Engineering, 2020, pages 138–149. ACM, 2020.
9. Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, and Alexandru Iosup. A review of serverless use cases and their characteristics. Technical Report SPEC-RG-2020-5, SPEC RG Cloud Working Group, May 2020.
10. Erwin Van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, and Simon Eismann. A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures. In Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, ICPE 2018, pages 21–24, 2018.
11. Erwin Van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. The SPEC cloud group's research vision on FaaS and serverless architectures. In Proceedings of the 2nd International Workshop on Serverless Computing, WOSC@Middleware 2017, Las Vegas, NV, USA, December 12, 2017, pages 1–4, 2017.
12. Erwin Van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uta, and Alexandru Iosup. Serverless is more: From PaaS to present cloud computing. IEEE Internet Comput., 22(5):8–17, 2018.
13. Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather S. Packer, and Thanassis Tiropanis. From FAIR research data toward FAIR and open research software. it Inf. Technol., 62(1):39–47, 2020.
14. Joseph M. Hellerstein, Jose M. Faleiro, Joseph Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. Serverless computing: One step forward, two steps back. In CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings. www.cidrdb.org, 2019.
15. Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Serverless computation with openlambda. In 8th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2016, Denver, CO, USA, June 20-21, 2016., 2016.

16 Alexandru Iosup, Catia Trubiani, Anne Koziolek, José Nelson Amaral, Andre B. Bondi, and Andreas Brunnert. Flexibility is key in organizing a global professional conference online: The ICPE 2020 experience in the COVID-19 era. CoRR, abs/2005.09085, 2020.

17 Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Jayant Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson. Cloud programming simplified: A berkeley view on serverless computing. CoRR, abs/1902.03383, 2019.

18 Ana Klimovic, YawenWang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. Pocket: Elastic ephemeral storage for serverless analytics. In 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018, pages 427–444. USENIX Association, 2018.

19 Ravi Madduri, Kyle Chard, Mike D'Arcy, Segun C. Jung, Alexis Rodriguez, Dinanath Sulakhe, Eric Deutsch, Cory Funk, Ben Heavner, Matthew Richards, Paul Shannon, Gustavo Glusman, Nathan Price, Carl Kesselman, and Ian Foster. Reproducible big data science: A case study in continuous fairness. PLOS ONE, 14(4):1–22, 04 2019.

20 Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. SOCK: Rapid task provisioning with serverlessoptimized containers. In USENIX Annual Technical Conference (USENIX ATC 18), July 2018.

21 A. V. Papadopoulos, L. Versluis, A. Bauer, N. Herbst, J. Von Kistowski, A. Ali-eldin, C. Abad, J. N. Amaral, P. Tuma, and A. Iosup. Methodological principles for reproducible performance evaluation in cloud computing. IEEE Trans. Software Eng., pages 1–1, 2019.

22 Qifan Pu, Shivaram Venkataraman, and Ion Stoica. Shuffling, fast and slow: Scalable analytics on serverless infrastructure. In Jay R. Lorch and Minlan Yu, editors, 16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019, pages 193–206. USENIX Association, 2019.

23 Mike Roberts. Serverless architectures. https://martinfowler.com/articles/ serverless.html, 2016. Continuous development of the material.

24 Joel Scheuner and Philipp Leitner. Function-as-a-service performance evaluation: A multivocal literature review. Journal of Systems and Software, 2020.

25 Dag I. K. Sjøberg, Jo Erskine Hannay, Ove Hansen, Vigdis By Kampenes, Amela Karahasanovic, Nils-Kristian Liborg, and Anette C. Rekdal. A survey of controlled experiments in software engineering. IEEE Trans. Software Eng., 31(9):733–753, 2005.

26 Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan S. Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. Is big data performance reproducible in modern cloud networks? In 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020, pages 513–527, 2020.

27 Erik van der Kouwe, Gernot Heiser, Dennis Andriesse, Herbert Bos, and Cristiano Giuffrida. Benchmarking flaws undermine security research. IEEE Secur. Priv., 18(3):48–57, 2020.

28 Erwin van Eyk, Joel Scheuner, Simon Eismann, Cristina L. Abad, and Alexandru Iosup. Beyond microbenchmarks: The SPEC-RG vision for a comprehensive serverless benchmark. In Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE), page 26–31, 2020.

2 Table of Contents

Working groups

Panel discussions

3 Overview of Talks

3.1 Serverless Predictions: 2021-2030 (Keynote Abstract – Topic 1)

Pedro García López (Universitat Rovira i Virgili – Tarragona, ES) and Bernard Metzler (IBM Research-Zurich, CH)

Joint work of Pedro García López, Aleksander Slominski, Michael Behrendt, Bernard Metzlert
Main reference Pedro García López, Aleksander Slominski, Michael Behrendt, Bernard Metzler: "Serverless Predictions: 2021-2030", CoRR, Vol. abs/2104.03075, 2021.
URL https://arxiv.org/abs/2104.03075

Within the next 10 years, advances on resource disaggregation will enable full transparency for most Cloud applications: to run unmodified single-machine applications over effectively unlimited remote computing resources. In this article, we present five serverless predictions for the next decade that will realize this vision of transparency – equivalent to Tim Wagner's Serverless SuperComputer or AnyScale's Infinite Laptop proposals.

The major hypothesis is that transparency will be achieved in the next ten years thanks to novel advances in networking, disaggregation, and middleware services. The huge consequence is the unification of local and remote paradigms, which will democratize distributed programming for a majority of users. This will realize the old and ultimate goal of hiding the complexity of distributed systems. The projected developments to reach the ultimate goal (Serverless End Game) include the following:

- Prediction 1: Serverless Clusters (Multi-tenant Kubernetes) will overcome the current limitations of direct communication among functions, hardware acceleration, and time limits.
- Prediction 2: Serverless Granular computing will offer 1-10 µs microsecond latencies for remote functions thanks to lightweight virtualization and fast RPCs.
- Prediction 3: Serverless memory disaggregation will offer shared mutable state and coordination at 2-10 µs microsecond latencies over persistent memory.
- Prediction 4: Serverless Edge Computing platforms leveraging 6G's ms latencies and AI optimizations will facilitate a Cloud Continuum for remote applications.
- Prediction 5: Transparency will become the dominant software paradigm for most applications, when computing resources become standardized utilities.

We discuss the basis of these predictions as well as technical challenges and risks. The predictions are mapped to phases to reach the final goal.

In conclusion, we argue that full transparency will be possible soon thanks to low latency and resource disaggregation. The Serverless End Game will unify local and remote programming paradigms, changing completely the way we currently create distributed applications. This is the ultimate goal of distributed systems, to become invisible using transparent middleware, and to simplify how users access remote resources.

3.2 Developer Experience for Serverless: Challenges and Opportunities (Keynote Abstract – Topic 2)

Robert Chatley (Imperial College London, GB)

Joint work of Robert Chatle, Goko, Adzic, Thomas Allerton, Hang Li Li
Main reference Robert Chatley, Thomas Allerton: "Nimbus: improving the developer experience for serverless applications", in Proc. of the ICSE '20: 42nd International Conference on Software Engineering, Companion Volume, Seoul, South Korea, 27 June – 19 July, 2020, pp. 85–88, ACM, 2020.
URL https://doi.org/10.1145/3377812.3382135

In this keynote talk we present a number of industrial case studies of building serverless systems, from the developer point of view. We examine how economic aspects – for example billing models – affect architectural design decisions for serverless applications, and also how available tooling inhibits or enhances developer experience when building, running and evolving serverless systems. We look at some current challenges, and propose some possible future directions aiming to address these.

After completing his PhD, Robert spent many years working in industry as a senior software engineer and a consultant before returning to university life. His work now bridges industry and academia, focussing on developing skills and knowledge in software engineers to build technical competence and improve developer productivity. His role at Imperial combines a strong focus on education with industry-focussed research. Robert's main interests are in developer experience – trying to support and improve developer productivity through advances in tools, technologies and processes.

3.3 Federated Function as a Service (Keynote Abstract – Topic 3)

Kyle Chard (University of Chicago, US) and Ian T. Foster (Argonne National Laboratory – Lemont, US)

Main reference Ryan Chard, Yadu N. Babuji, Zhuozhao Li, Tyler J. Skluzacek, Anna Woodard, Ben Blaiszik, Ian T. Foster, Kyle Chard: "funcX: A Federated Function Serving Fabric for Science", in Proc. of the HPDC '20: The 29th International Symposium on High-Performance Parallel and Distributed Computing, Stockholm, Sweden, June 23-26, 2020, pp. 65–76, ACM, 2020.
URL https://doi.org/10.1145/3369583.3392683

Introduction

The serverless paradigm has revolutionized programming by allowing programmers to develop, run, and manage scalable *applications* without needing to build and operate the *infrastructure* that would normally be required to host those applications [5]. In particular, the popular function as a service (FaaS) model reduces application development to two tasks: defining and registering (or discovering) high-level programming language functions, and invoking those functions. The underlying FaaS platform, traditionally operated by a cloud provider, then deals with the complexities of provisioning and managing the servers, virtual machines, containers, and programming environments needed to run those functions.

Most current FaaS offerings adopt the powerful simplifying assumption that functions run on a single, centralized, and homogeneous platform, whether a commercial (public) cloud like AWS, Azure, or Google, or a dedicated (private) cluster as in the case of OpenWhisk. In such environments, FaaS systems provide simple and intuitive APIs that democratize access to seemingly unlimited remote elastic computing capacity. But modern computing environments are increasingly distributed and heterogeneous.

For example, quasi-ubiquitous machine learning methods require access to specialized hardware (e.g., AI accelerators) and introduce new workload patterns (e.g., large-memory training and short-duration inference) and interactive and event-based computing models (e.g., from automated laboratories and robots) that require instantaneous access to specialized computing capabilities. Sensor network applications often require that data be processed near to data sources to reduce bandwidth needs and/or enable rapid response.

To address these concerns, we propose a new *federated FaaS model* in which function executions can be dispatched to arbitrary computing resources, chosen for example on the basis of latency, cost, data locality, security, or other concerns. This new model preserves and leverages powerful features of conventional FaaS (e.g., simple function registration and invocation APIs, abstraction of infrastructure) while also allowing programmers to operate effectively in a distributed computational continuum [2]. In effect, federated FaaS aims to allow computation to flow to wherever makes sense for a particular purpose.

funcX: early experiences with federated FaaS

funcX [4] is a federated FaaS platform designed to address some of the challenges outlined above. funcX adapts the traditional cloud-hosted FaaS model by enabling users to route function invocations to a distributed set of user-deployed funcX endpoints. Thus, users can add their own computing system (e.g., cluster, cloud, laptop) to the funcX ecosystem by deploying an endpoint and they may then use those endpoints to execute functions. From a user's perspective, funcX looks like any other FaaS system: users register functions with the cloud-hosted funcX service, they may then invoke that function by specifying input arguments and the target endpoint. funcX manages the complexity of execution, authenticating with the remote endpoint, reliably executing the function (optionally inside a container), and caching results (or exceptions) until retrieved by the user.

Over the past year we have applied funcX to a range of research applications and as the basis for building other services (e.g., DLHub [3]). We have found that funcX can effectively abstract the complexity of using diverse computing resources, simplify authentication and authorization, reduce the difficulties associated with scaling resources to support workloads, remove the challenge of porting applications between different systems and data centers, and enable new application modes such as event-based and interactive computing.

We have also identified limitations of the federated FaaS model as realized in our work to date. For example, many applications cannot easily be mapped to the FaaS paradigm; funcX's centralized data and state management restrict the application patterns that can be implemented, and require that the ratio of data size to compute must be reasonable to keep transfer overheads manageable; containers fail to solve portability problems in HPC environments; and the coarse-grained allocation models of HPC systems do not lend themselves well to function execution. These are all topics that we are addressing in current work.

Open challenges

The federated FaaS model introduces fascinating research challenges, including the following.

Data and State. Traditionally, FaaS functions are stateless. However, many applications require that data be passed to, from, and between functions. (Indeed, data locality is one of the main reasons to apply a federated model.) Conventional cloud-hosted FaaS platforms

meet these needs via universal object storage; however, such storage is not generally accessible in federated settings. There is a need to explore FaaS application communication patterns, data-centric programming models for FaaS, transparent wide-are data staging, and shared data substrates for low-latency data sharing.

Environment management. FaaS systems leverage containerized environments that enable dependencies to be met while sandboxing execution in multi-tenant environments. Cloud FaaS systems have developed new software container technologies with rapid startup time and low cold start overheads [1]. The heterogeneous environments found in federated FaaS create more challenges, such as diverse container technologies and slow resource provisioning [7].

Scheduling. Increasingly heterogeneous computing environments create a continuum of computing capacity: from edge computing devices through to supercomputers. Federated FaaS makes it easy to route functions to execute anywhere, and thus exposes a fabric on which broad scheduling policies can be explored. Such scheduling polices may consider data locations, transfer costs, resource provisioning time, resource costs (monetary and/or availability), hardware performance, and function needs [6].

Security, policies, regulations. Federated FaaS is distinguished from conventional FaaS by a quite different security model. In a federated environment, each computing endpoint may be located in a distinct administrative domain with unique authentication and authorization systems, policies, and regulations. Centralized FaaS systems typically operate within a single administrative domain. Federated FaaS requires methods for bridging domains and for ensuring that policies and regulations are enforced.

Summary

Federated FaaS provides a potential solution to long-standing remote computing challenges. In so doing, it enables a range of new application scenarios and moves us closer to a truly integrated treatment of local and remote computing. It also exposes fascinating new research challenges that will only grow in importance as both application demands and technologies continue to develop.

References

1 A Agache et al., Firecracker: Lightweight virtualization for serverless applications, 17th USENIX Symposium on Networked Systems Design and Implementation, 2020, pp. 419–434.
2 P Beckman et al.,Harnessing the computing continuum for programming our world, Fog Computing: Theory and Practice (2020), 215–230.
3 R Chard et al., Dlhub: Model and data serving for science, 35rd IEEE International Parallel and Distributed Processing Symposium, 2019, pp. 283–292.
4 R Chard et al., FuncX: A federated function serving fabric for science, 29th International Symposium on High-Performance Parallel and Distributed Computing, ACM, 2020, p. 65–76.
5 S Eismann et al., Serverless applications: Why, when, and how?, IEEE Software 38(2020), no. 1, 32–39.
6 R Kumar et al., Coding the computing continuum: Fluid function execution in heterogeneous computing environments, IEEE International Parallel and Distributed Processing Symposium Workshops, 2021, pp. 66–75.
7 T Shaffer et al., Lightweight function monitors for fine-grained management in large scale Python applications, IEEE International Parallel and Distributed Processing Symposium, 2021, pp. 786–796

3.4 Characterizing Serverless Systems (Keynote Abstract – Topic 4)

Mohammad Shahrad (University of British Columbia – Vancouver, CA)

This keynote is dedicated to understanding the importance of characterizing serverless systems from different perspectives. To make the case, two characterization studies will be presented: 1) a cluster-wide characterization of the entire serverless workload at Azure Functions [1], and 2) a detailed micro-architectural study of Apache OpenWhisk [2]. The insights gained by the first study lead to designing an adaptive scheduling policy reducing cold starts and resource wastage, and the observations in the second study reveal inefficiencies in cloud-grade processors in serving serverless workloads. The talk also emphasizes the importance of reproducibility through open-sourcing traces or tools.

References

1 Mohammad Shahrad, Rodrigo Fonseca, and Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. *Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider*. USENIX Annual Technical Conference (USENIX ATC), 2020

2 Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. *Architectural implications of function-as-a-service computing*. 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2019.

3.5 Beyond Load Balancing: Package-Aware Scheduling for Serverless Platforms

Cristina Abad (ESPOL – Guayaquil, EC)

Joint work of Gabriel Aumala, Edwin F. Boza, Luis Ortiz-Avilés, Gustavo Totoy, Cristina L. Abad
Main reference Gabriel Aumala, Edwin F. Boza, Luis Ortiz-Avilés, Gustavo Totoy, Cristina L. Abad: "Beyond Load Balancing: Package-Aware Scheduling for Serverless Platforms", in Proc. of the 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2019, Larnaca, Cyprus, May 14-17, 2019, pp. 282–291, IEEE, 2019.
URL https://doi.org/10.1109/CCGRID.2019.00042

Fast deployment and execution of cloud functions in Function-as-a-Service (FaaS) platforms is critical, for example, for user-facing services in microservices architectures. However, functions that require large packages or libraries are bloated and start slowly. An optimization is to cache packages at the worker nodes instead of bundling them with the functions. However, existing FaaS schedulers are vanilla load balancers, agnostic to packages cached in response to prior function executions, and cannot properly reap the benefits of package caching. We study the case of package-aware scheduling and propose PASch, a novel Package-Aware Scheduling algorithm that seeks package affinity during scheduling so that worker nodes can re-use execution environments with preloaded packages. PASch leverages consistent hashing and the power of two choices, while actively avoiding worker overload. We implement PASch in a new scheduler for the OpenLambda framework and evaluate it using simulations and real experiments. We evaluated PASch with varying cluster sizes and skewness of package popularity distribution, and found that it outperforms a regular balancer by as much as 318x (median speedup). Furthermore, for the workloads studied in this paper, PASch can outperform consistent hashing with bounded loads – a state-of-the-art load balancing algorithm – by 1.3x (mean speedup), and a speedup of 1.5x at the 80th percentile.

3.6 Accelerating Reads with In-Network Consistency-Aware Load Balancing

Samer Al-Kiswany (University of Waterloo, CA)

Joint work of Hatem Takruri, Ibrahim Kettaneh, Ahmed Alquraan, Samer Al-Kiswany
Main reference Hatem Takruri, Ibrahim Kettaneh, Ahmed Alquraan, Samer Al-Kiswany: "FLAIR: Accelerating Reads with Consistency-Aware Network Routing", in Proc. of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), pp. 723–737, USENIX Association, 2020.
URL https://www.usenix.org/conference/nsdi20/presentation/takruri

Replication is the main reliability technique for many modern cloud services that process billions of requests each day. Unfortunately, modern strongly-consistent replication protocols – such as multi-Paxos, Raft, Zab, and Viewstamped replication (VR) – deliver poor read performance. This is because these protocols are leader-based: a single leader replica (or leader, for short) processes every read and write request, while follower replicas (followers for short) are used for reliability only.

I present FLAIR, a novel approach for accelerating read operations in leader-based consensus protocols. FLAIR leverages the capabilities of the new generation of programmable switches to serve reads from follower replicas without compromising consistency. The core of the new approach is a packet-processing pipeline that can track client requests and system replies, identify consistent replicas, and at line speed, forward read requests to replicas that can serve the read without sacrificing linearizability. An additional benefit of FLAIR is that it facilitates devising novel consistency-aware load balancing techniques. Following the new approach, the research team designed FlairKV, a key-value store atop Raft. FlairKV implements the processing pipeline using the P4 programming language. We evaluate the benefits of the proposed approach and compare it to previous approaches using a cluster with a Barefoot Tofino switch. The evaluation indicates that, compared to state-of-the-art alternatives, the proposed approach can bring significant performance gains: up to 42% higher throughput and 35-97% lower latency for most workloads.

3.7 A tool set for serverless

Ahmed Ali-Eldin Hassan (Chalmers University of Technology – Göteborg, SE)

Joint work of Ahmed Ali-Eldin Hassan, Prashant Shenoy

Designing full stack serverless edge applications is a challenge. System dynamics of most edge applications poses challenges to what application can be developed using the serverless model to run on the edge. Specific challenges include function startup times, and state management. In this line of research, our aim is to develop models, tools, and frameworks that can enable programmers and system owners to harness the power of serverless computing for edge systems. We will initially focus on the problem of startup times and state management. Our aim is to eventually build an entire tool-base that enables for optimizes compiling applications into serverless functions, optimizes the deployment of serverless based applications, and optimizes the runtime on the edge.

3.8 Serverless execution of scientific workflows

Bartosz Balis (AGH University of Science & Technology – Krakow, PL)

Joint work of Maciej Malawski, Adam Gajek, Adam Zima, Bartosz Balis, Kamil Figiela
Main reference Maciej Malawski, Adam Gajek, Adam Zima, Bartosz Balis, Kamil Figiela: "Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions", Future Gener. Comput. Syst., Vol. 110, pp. 502–514, 2020.
URL https://doi.org/10.1016/j.future.2017.10.029

Scientific workflows, consisting of a large number of tasks structured as a graph, are an important paradigm for automation in scientific computing. We discuss the applicability of serverless infrastructures to compute- and data-intensive workflows, and options for designing serverless workflow execution architecture. We also present cost analysis and implications with regard to resource management for scientific applications in the serverless paradigm. The approach is experimentally evaluated using the HyperFlow workflow management system and real workflow applications. Our findings indicate that the simple mode of operation makes the serverless approach attractive and easy to use, although for larger workflows traditional IaaS infrastructure is more cost-efficient. We conclude that a hybrid approach combining VMs with cloud functions for small tasks could be a good execution model for scientific workflows.

3.9 Using Severless Computing for Streamlining the Data Analytic Process

André Bauer (Universität Würzburg, DE)

The discipline of data analytics has grown significantly in recent years as a means to make sense of the vast amount of data available. It has permeated every aspect of computer science and engineering and is heavily involved in business decision-making. However, data analytics projects are often done manually. To accelerate and improve such projects, there are, for example, Federated Learning and the best practices of DataOps. Since such approaches need a high degree of flexibility and should generate as little overhead as possible, I am interested in how far Serverless Computing can be used to guarantee these conditions.

3.10 Challenges for Serverless Databases

A. Jesse Jiryu Davis (MongoDB – New York, US)

Joint work of Judah Schvimer, A. Jesse Jiryu Davis, Max Hirschhor
Main reference Judah Schvimer, A. Jesse Jiryu Davis, Max Hirschhorn: "eXtreme Modelling in Practice", Proc. VLDB Endow., Vol. 13(9), pp. 1346–1358, 2020.
URL https://doi.org/10.14778/3397230.3397233

Academic research into serverless platforms has focused primarily on FaaS, not on the backend services such as databases that support FaaS applications. Furthermore, the literature mostly discusses how to use serverless platforms from the application developer's perspective, rather

than how to implement them from the provider's perspective. My research goal is to review the state of the art for implementing serverless platforms, particularly serverless databases. I am interested in methods for scaling and balancing tenants in multi-tenant serverless databases, and moving tenants between servers efficiently and without disruption. I am also interested in testing and validation methods for distributed systems algorithms, including formal methods such as TLA+.

3.11 Using Serverless to Improve Online Gaming

Jesse Donkervliet (VU University Amsterdam, NL) and Alexandru Iosup (VU University Amsterdam, NL)

Main reference Jesse Donkervliet, Animesh Trivedi, Alexandru Iosup: "Towards Supporting Millions of Users in Modifiable Virtual Environments by Redesigning Minecraft-Like Games as Serverless Systems", in Proc. of the 12th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2020, July 13-14, 2020, USENIX Association, 2020.
URL https://www.usenix.org/conference/hotcloud20/presentation/donkervliet

Serverless computing offers potential for the simpler and more cost-efficient deployment of large-scale systems. Online games are a billion dollar industry supported by large-scale distributed systems. How can these systems benefit from serverless computing? How to design real-time online games for serverless platforms? How to meet the QoS requirements of these systems? How to guarantee sufficient consistency between users? How to schedule its components cost- and energy-efficiently? I am interested in learning more about these questions and exploring their answers.

3.12 Understanding and optimizing serverless applications

Simon Eismann (Universität Würzburg, DE)

Joint work of Simon Eismann, Joel Scheuner, Erwin Van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup
Main reference Simon Eismann, Joel Scheuner, Erwin Van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup: "Serverless Applications: Why, When, and How?", IEEE Softw., Vol. 38(1), pp. 32–39, 2021.
URL https://doi.org/10.1109/MS.2020.3023302

Serverless application are a novel computing paradigmn with rapidly growing industry adoption. However, there are still many open questions about the characteristics of serverless applications, such as how many serverless functions does a typical serverless microservice consist of. Additionally, there are still a number of manual configurations that developers need to fine-tune in order to optimize their applications.

We collected 89 serverless applications from white literature, grey literature, open-source projects, and scientific computing and analyze their characteristics to provide insight into the current state of serverless applications. Further, we present approaches to model the performance of serverless workflows and serverless functions with different sizes to enable the automated optimization of serverless functions and workflows.

3.13 Autonomous resource allocation methods for serverless systems

Erik Elmroth (University of Umeå, SE)

With the overall research direction being on how to build autonomous or semi-autonomous resource management systems for IT systems, we have done a lot of work on resource allocation topics such as scaling, orchestration, scheduling, service differentiation and all kind of techniques trying to control performance, efficiency, reliability, etc. And we have been doing this for systems spanning from servers and clusters to rack-scale systems, datacenters, edge environments, and so on. Serverless systems are obviously within scope for this.

In the past few years we have also spent more and more efforts on handling the situations that cannot be controlled, by focusing on anomaly detection, primarily trying to identify performance issues and their root causes but also considered functional and security anomalies, which are not always easy to distinguish.

When looking more specifically into serverless systems, we have recently initiated a project where we on one hand try to determine in advance what resources are needed and how they should be allocated to meet particular performance requirements. As these systems are increasingly building on machine learning models, we are also digging deeper into the questions of when to retrain the models, what data to use for retraining, and ultimately what data to save for future retraining of the machine learning models used in the management systems.

3.14 Is Serverless an Opportunity for Edge Applications?

Nicola Ferrier (Argonne National Laboratory, US)

URL www.sagecontinuum.org

Deploying AI at the edge creates an opportunity to develop software defined sensors, using cameras and microphones, along with appropriate software to enable scientists to obtain measurements specific to their application. Some processing methods may require resources that exceed available edge resources. In addition, having multiple scientists seeking to use the same edge device might require off-loading some computations. Serverless architecture for these applications could support a seamless method to have methods run on the edge, cloud, or high-performance computing centers.

3.15 HyScale into Serverless: Vision and Challenges

Hans-Arno Jacobsen (University of Toronto, CA)

Joint work of Yuqiu Zhang, Hans-Arno Jacobsen
Main reference Anthony Kwan, Jonathon Wong, Hans-Arno Jacobsen, Vinod Muthusamy: "HyScale: Hybrid and Network Scaling of Dockerized Microservices in Cloud Data Centres", in Proc. of the 39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019, pp. 80–90, IEEE, 2019.
URL https://doi.org/10.1109/ICDCS.2019.00017

Microservices, in contrast to traditional monolithic architectures, consist of several smaller dedicated processes working together to provide services to users and are widely adopted in industry due to better flexibility and reliability. Container technologies, such as Docker, provide a lightweight environment for deploying the microservices in computing clusters. Containers are independent units that package software and its dependencies together. Similar to virtual machines (VMs), containers are a virtualization technology that allows a single computing resource to be shared among multiple microservices. However, different from VMs which virtualize resources at the hardware level, containers are virtualized at the operating system level. Containers provide weaker isolation, but are much smaller in size, take much less time to start/stop, and bear lower overhead. This results in improving resource utilization in terms of the number of machines required to host a given workload on the host.

The large-scale adoption of containers for hosting microservices requires the use of container orchestration middleware, such as Kubernetes, to efficiently manage and deploy them. Therefore, an important issue arises, which is to schedule and place containerized applications on available hosts. When submitting an application for deployment, the container orchestration middleware must place it on one of the available resources, considering the limitations of the application and aiming to maximize the use of computing resources. From a cost-efficient perspective, the container orchestration middleware should consider factors such as the capacity of available machines, application performance, quality of service, energy consumption, and operation costs. Typically, a container orchestration middleware provides a unified control interface that is responsible for the whole cluster. The control interface, which we call an autoscaler, runs the autoscaling algorithm to automatically scale up or down the number of allocated resources of containers based on system usage, user requirements, and costs. Our HyScale project is dedicated to solve this issue by building a cost-efficient, SLO-aware autoscaler for container orchestration systems (service level objective-aware).

Going into the serverless era, we envision HyScale to have even more impact and practical use. First, since serverless services are widely adopting containers as the underlying infrastructure, HyScale should be able to seamlessly work with any container-based serverless frameworks to provide elasticity and scalability improvements. Second, the principle of 'scaling from zero to infinity' intrinsic to serverless computing and the fact that serverless function executions are mostly short-lived and small in size, finer-grained and faster-reacting autoscaling policies are required to meet the specific needs for this new paradigm. This also puts challenges upon HyScale design to account for the faster autoscaling decision making needs. Moreover, the problem of cold start becomes even more inevitable in the serverless context, where higher requirements of the cooldown period are expected. This needs more meticulous thinking in autoscaler design as it is usually difficult to find a balance between cost efficiency and cooldown period reduction. All in all, autoscaling in serverless is definitely an interesting and promising research area where HyScale can be devoted to in the near future.

3.16 Serverless Workflows for Sustainable High-Performance Data Analytics

Nikolas Herbst (Universität Würzburg, DE)

Main reference Simon Eismann, Joel Scheuner, Erwin Van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup: "Serverless Applications: Why, When, and How?", IEEE Softw., Vol. 38(1), pp. 32–39, 2021.
URL https://doi.org/10.1109/MS.2020.3023302

With the current serverless technologies like FaaS as early enabling technology, we see huge potential for the next generation of serverless computing. Current technical limitations can be overcome: among the current major limitations, we see in accordance with [1] [2] (1) missing ways for direct low-latency communication of functions, (2) efficient state transfer of intermediate results via dis-aggregated memory, (3) in-transparency in terms of real resource consumption and apriori cost estimates, and (4) missing intelligence in placement and scheduling of distributed serverless workflows in the cloud to edge continuum.

We envision a resource-efficient serverless computing platform enabling the specification, management, and automated execution of high-performance data analytic workflows for experts as well as non-experts. A low entry-barrier (NoOps) and flexibility (fine-granular PayPerUse including scale-to-zero) of the envisioned platform could foster interdisciplinary research across research domains based. Besides a sustainable serverless compute infrastructure, we envision a data analytic workflow engine that can leverage serverless technology for ease of assembly, configuration, and efficient operation with a high degree of reusability for distributed data sources. It should support end-to-end data analysis including steps like initial data-quality assessment for a result confidence rating, feature selection, model federation, tuning, method chaining, model-(re-)training, and more.

References

1 Pedro García López, Aleksander Slominski, Michael Behrendt, Bernard Metzler: Serverless Predictions: 2021-2030. CoRR abs/2104.03075 (2021)
2 Joseph M. Hellerstein, Jose M. Faleiro, Joseph Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, Chenggang Wu: Serverless Computing: One Step Forward, Two Steps Back. CIDR 2019

3.17 Massivizing Computer Systems: Science, Design, and Engineering for Serverless Computing

Alexandru Iosup (VU University Amsterdam, NL) and Erwin van Eyk (VU University Amsterdam, NL)

Main reference Alexandru Iosup, Alexandru Uta, Laurens Versluis, Georgios Andreadis, Erwin Van Eyk, Tim Hegeman, Sacheendra Talluri, Vincent van Beek, Lucian Toader: "Massivizing Computer Systems: A Vision to Understand, Design, and Engineer Computer Ecosystems Through and Beyond Modern Distributed Systems", in Proc. of the 38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018, pp. 1224–1237, IEEE Computer Society, 2018.
URL https://doi.org/10.1109/ICDCS.2018.00122

The idea of enabling businesses, governments, scientific labs, and society at-large to use IT infrastructure for fine-grained, daily operations, without detailed management of operational logic, emerged in the 1950s and remains a grand challenge in computer science. After a

hiatus between roughly the 1970s through the 2000s, in the 2010s the cloud has picked up the challenge. In the 2020s instance of this challenge, serverless computing, the cloud provider manages the resources, lifecycle, and execution of user-provided functions, all packaged together into fine-grained applications (and fine-grained monitoring, accounting, and billing). But, beyond the low-hanging fruits of this models, and the early emergence of Function-as-a-Service (FaaS), the challenge remains largely untouched.

In this talk, we posit that the principles, challenges, and approach of **massivizing computer systems** [1] could help. Massivizing computer systems is a multi-disciplinary, mixed-methods approach, spanning at least distributed computer systems, performance engineering, and software engineering. We highlight in this talk several points:

1. We do not have to start from scratch to understand why serverless is more (than PaaS cloud) [2]: Using a historiographical approach focusing on technology, we can *explain the origins of serverless computing, and predict (envision) its evolution* focusing on the most important aspects and avoiding the common pitfalls of the past.
2. Building the serverless systems and applications of the future depends – much like the containerization of transport did in its early decades – on *modeling the architecture of serverless operations*. The SPEC RG reference architecture for FaaS [3] is an example of this.
3. Designing new parts and composites is essential for serverless computing, because the current technology raises many technical issues, and the interplay between non-functional properties such as performance, elastic scalability, dependability, security, and sustainability (in particular, energy-efficiency) is complex. The task is daunting and will require many different designers to be able to share and work together, so we need to also design processes, in other words, to *design the design of serverless systems and applications* [4]. A similar argument can be made about *optimization and tuning*.
4. Understanding and analyzing serverless ecosystems is necessary – paraphrasing every scientist and engineer ever, we cannot hope to use what we do not understand, lest it collapses when we least expect it. *Real-world experiments and benchmarking* are important activities here [5]. But real-world experimentation is too costly and time-consuming for large-scale, long-term operations. Instead, *simulation*-based approaches, e.g., based on simulators such as OpenDC [6], are important. Much like a single model cannot capture entirely complex real-life situations, the community should provide multiple simulators (models), and consider *predictions based on ensembles of models*. *Reproducibility* is another important aspect of this line of work [7].
5. Last, but not least, we need a forum to discuss serverless-related topics, especially focusing on the interplay between non-functionals. *The SPEC RG Cloud Group* provides such a forum and is inclusive. Developing its flagship workshop, HotCloudPerf, and merging it with others to form a serverless conference, could provide an annual selective event. Sharing data and software artifacts, FAIRly, would benefit all and be greatly facilitated by such a community/conference. We also envision here a *Memex-like approach to preserve diverse operational traces representative of serverless computing*.

References

1 Alexandru Iosup, Alexandru Uta, Laurens Versluis, Georgios Andreadis, Erwin Van Eyk, Tim Hegeman, Sacheendra Talluri, Vincent van Beek, Lucian Toader: *Massivizing Computer Systems: A Vision to Understand, Design, and Engineer Computer Ecosystems Through and Beyond Modern Distributed Systems*. ICDCS 2018: 1224-1237.

2 Erwin Van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uta, Alexandru Iosup: *Serverless is More: From PaaS to Present Cloud Computing*. IEEE Internet Comput. 22(5): 8-17 (2018)

3 Erwin Van Eyk, Alexandru Iosup, Johannes Grohmann, Simon Eismann, André Bauer, Laurens Versluis, Lucian Toader, Norbert Schmitt, Nikolas Herbst, Cristina L. Abad: *The SPEC-RG Reference Architecture for FaaS: From Microservices and Containers to Serverless Platforms*. IEEE Internet Comput. 23(6): 7-18 (2019)

4 Alexandru Iosup, Laurens Versluis, Animesh Trivedi, Erwin Van Eyk, Lucian Toader, Vincent van Beek, Giulia Frascaria, Ahmed Musaafir, Sacheendra Talluri: *The AtLarge Vision on the Design of Distributed Systems and Ecosystems*. ICDCS 2019: 1765-1776

5 Erwin Van Eyk, Joel Scheuner, Simon Eismann, Cristina L. Abad, Alexandru Iosup: *Beyond Microbenchmarks: The SPEC-RG Vision for a Comprehensive Serverless Benchmark*. ICPE Companion 2020: 26-31

6 Fabian S. Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin van Eyk, Laurens Versluis, Vincent van Beek, Alexandru Iosup (2021) *OpenDC 2.0: Convenient Modeling and Simulation of Emerging Technologies in Cloud Datacenters*. CCGrid 2021

7 Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim von Kistowski, Ahmed Ali-Eldin, Cristina L. Abad, José Nelson Amaral, Petr Tuma, Alexandru Iosup: *Methodological Principles for Reproducible Performance Evaluation in Cloud Computing*. IEEE TSE.

3.18 Machine Learning to enable Autonomous Serverless Systems

Pooyan Jamshidi (University of South Carolina – Columbia, US)

Joint work of Nabor Chagas Mendonça, Pooyan Jamshidi, David Garlan, Claus Pahl
Main reference Nabor Chagas Mendonça, Pooyan Jamshidi, David Garlan, Claus Pahl: "Developing Self-Adaptive Microservice Systems: Challenges and Directions", IEEE Softw., Vol. 38(2), pp. 70–79, 2021.
URL https://doi.org/10.1109/MS.2019.2955937

Serverless computing offers cloud functions, a new type of cloud service that offers fine granularity and lower latency. However, building systems with this new computing platform comes with its challenges: (1) functions are stateless and may need to download large amounts of code/data when they boot up, (2) functions have very limited runtime before they are killed, (3) Storage is limited, but much faster comparing with outside services, (4) the number of available cloud workers depends on the overall load of service providers and the load can only be predicted, (5) node failures occur when running at a large scale, (6) the dependencies differ in functions comparing with an on-premise machine, and (7) latency to the cloud makes roundtrips costly. (8) the cost of acquiring and running a function may vary over time and across providers.

Although researchers have addressed some of these challenges, I am, in particular, interested in developing a vendor-agnostic framework that application developers can build their serverless systems with functionalities such as load balancing between cloud providers and reconfiguring the serverless pipeline to optimize the performance and reliability of the system. The framework can also dynamically map functions to compute nodes based on performance, reliability, and cost trade-off. In addition, automated fault detection and repair will enable resilient and robust serverless application development. We rely on our recent advancement in machine learning, particularly Causal AI (Causal Structure Learning, Causal Inference, Counterfactual Reasoning, Causal Transfer Learning), to enable the proposed capabilities in the serverless framework.

3.19 Self-Aware Platform Operations and Resource Management

Samuel Kounev (Universität Würzburg, DE)

Joint work of Samuel Kounev, Simon Eismann, Johannes Grohmann, Erwin Van Eyk, Nikolas Herbst
Main reference Simon Eismann, Johannes Grohmann, Erwin Van Eyk, Nikolas Herbst, Samuel Kounev: "Predicting the Costs of Serverless Workflows", in Proc. of the ICPE '20: ACM/SPEC International Conference on Performance Engineering, Edmonton, AB, Canada, April 20-24, 2020, pp. 265–276, ACM, 2020.
URL https://doi.org/10.1145/3358960.3379133

In serverless computing, the responsibility for operation aspects, including application resource management, is offloaded to the Cloud provider. This includes, for example, managing virtual machines and containers, managing function execution runtimes (e.g., a Python runtime environment with respective libraries), elastic scaling, reliability/fault tolerance, monitoring, and logging. To manage such aspects, novel mechanisms for automated and proactive resource management are required. We focus on the development of techniques for self-aware platform operations including online learning and reasoning capabilities for efficient and scalable workflow execution.

3.20 From design to migration and management: FaaS platforms for application porting to optimized serverless implementation and execution

Georgios Kousiouris (Harokopion University – Athens, GR)

Joint work of George Kousiouris, Dimosthenis Kyriazis
Main reference George Kousiouris, Dimosthenis Kyriazis: "Functionalities, Challenges and Enablers for a Generalized FaaS based Architecture as the Realizer of Cloud/Edge Continuum Interplay", in Proc. of the 11th International Conference on Cloud Computing and Services Science, CLOSER 2021, Online Streaming, April 28-30, 2021, pp. 199–206, SCITEPRESS, 2021.
URL https://doi.org/10.5220/0010412101990206

The availability of decentralized edge computing locations as well as their combination with more centralized Cloud solutions enables the investigation of various trade-offs for application component placement in order to optimize application behaviour and resource usage. Key functionalities and operations needed by a middleware layer so that it can serve as a generalized architectural and computing framework in the implementation of a Cloud/Edge computing continuum are presented. As a primary middleware candidate, FaaS frameworks are taken under consideration, given their significant benefits such as flexibility in execution, event driven nature and enablement of incorporation of arbitrary and legacy application components triggered by diverse actions and rules. Gaps and enablers for three different layers (application design and implementation, semantically enriched runtime adaptation/configuration and deployment optimization) are highlighted. The goal is to enable abstracted application design and porting to the serverless paradigm, based on ready-made, reusable and self-regulating pattern prototypes, semantic annotation of functions in order to dictate deployment or runtime needs (based on goals and constraints), used by the underlying management mechanisms, as well as runtime optimization of the candidate services selection based on performance and configuration trade-offs. The talk highlights the main approach and goals of the H2020 PHYSICS project (https://physics-faas.eu/).

3.21 Software Development Using Serverless Systems

Philipp Leitner (Chalmers University of Technology – Göteborg, SE)

Function-as-a-Service, and more generally serverless, is a massive area of research interest at the moment. Most of this research deals with how to build, maintain, and scale serverless infrastructure – a problem that few companies outside of a few large cloud providers actually have. However, orders of magnitude more, from small start-ups to billion-dollar industries, face the challenge of how to best make use of this new wave of cloud services to ideally serve their customers. My interest is studying how software engineering research can best support practitioners in this new world.

3.22 Running and Scheduling Scientific Workflows on Serverless Clouds: From Functions to Containers

Maciej Malawski (AGH University of Science & Technology – Krakow, PL)

Main reference Krzysztof Burkat, Maciej Pawlik, Bartosz Balis, Maciej Malawski, Karan Vahi, Mats Rynge, Rafael Ferreira da Silva, Ewa Deelman: "Serverless Containers – rising viable approach to Scientific Workflows", CoRR, Vol. abs/2010.11320, 2020.
URL https://arxiv.org/abs/2010.11320

Scientific workflows are an important class of applications, which consist of computing tasks and data transfers connected into a dependency graph. Traditionally, they are executed on HPC clusters, distributed infrastructures such as grids or clouds. Recent emergence of serverless infrastructures drives us to explore the applicability of these platforms to scientific workflows and associated research problems related to resource management.

Using HyperFlow, our workflow engine developed at AGH, we have evaluated the scientific workflow execution using FaaS (AWS Lambda, Google Cloud Functions) and CaaS platforms (AWS Fargate, Google Cloud Run). We have also performed performance evaluation of serverless cloud infrastructures with a focus on scientific workflows. Based on these experiences, we have recently started addressing scheduling challenges on highly-elastic infrastructures, including cloud functions and containers. Moreover, we have also approached solving simple scheduling problems using D-Wave quantum annealer, achieving quite promising preliminary results for small graphs of tasks fitting entirely in the computer architecture.

Current experience with severless platforms leads to the conclusion that they provide a viable solution for scientific applications, not only scientific workflows but also for large-scale data processing tasks, which come, e.g., from High Energy Physics domain. Serverless infrastructures provide excellent scalability, elasticity and high level of automation of resource management, but as there are many decisions regarding selection of function of container memory and CPU allocation, research on scheduling and performance optimization is still needed.

3.23 The case for a hybrid cloud model for serverless computing

Vinod Muthusamy (IBM TJ Watson Research Center – Yorktown Heights, US)

Distributed applications have traditionally been architected to run on a single cloud vendor, using a combination of compute, storage, messaging, load balancing, orchestration, monitoring, authentication, analytics, and numerous other platform capabilities offered by the cloud vendor. Relying on a single vendor's platform has the benefits of tight integration of these capabilities but leads to vendor lock-in, making it difficult for application owners to migrate to another cloud vendor, and challenging for new cloud vendors to compete without building their own portfolio of services.

Hybrid cloud or multi-cloud architectures address the drawbacks of single-vendor cloud platforms, building applications and tooling to allow distributed application components to run on a mixture of private, on-premise, dedicated, and public cloud environments. Application developers have the flexibility to easily migrate their entire applications to another cloud vendor, or make fine-grained deployment decisions based on the performance, cost, regulatory compliance, security policies, and other capabilities of the cloud vendor, matched with the requirements of each application component.

Seen in this light, serverless computing is still in its infancy, with most serverless applications developed for and run on a single serverless platform. There are a class of enterprise applications that aren't amenable to run fully on a public cloud due to regulatory constraints, and the vendor and platform lock-in in today's most popular serverless platforms is holding back these applications from being rearchitectured on serverless principles. As well, geo-distributed applications, such as those architected for edge computing platforms, will benefit from taking advantage of a variety of edge vendors; relying on a single vendor to offer edge servers at all desired locations severely constrains the choice of vendors.

A hybrid serverless model brings with it a number of challenges across the stack, including addressing the impedance mismatch when bridging across serverless platforms from multiple providers, including the non-functional properties such as latency, scalability, availability, and cost. For example, it is not clear what is the emergent cold-start behavior when a serverless function running on one platform calls a function on another. There are also functional mismatches, such as security policies, and messaging semantics that need to be reconciled.

As in conventional cloud applications, supporting serverless applications to run across a multi-cloud or hybrid could environment will give developers more flexibility, enable a new class of serverless applications held back by vendor lock-in constraints, support truly geo-distributed serverless applications, and offer an opportunity for new serverless platform vendors to compete with novel platform capabilities.

3.24 Performance Evaluation in Serverless Computing

Alessandro Vittorio Papadopoulos (Mälardalen University – Västerås, SE)

Joint work of Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim von Kistowski, Ahmed Ali-Eldin, Cristina Abad, José Nelson Amaral, Petr Tuma, Alexandru Iosup
Main reference Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim Von Kistowski, Ahmed Ali-eldin, Cristina Abad, José Nelson Amaral, Petr Tůma, Alexandru Iosup: "Methodological Principles for Reproducible Performance Evaluation in Cloud Computing", IEEE Transactions on Software Engineering, pp. 1–1, 2019.
URL https://doi.org/10.1109/TSE.2019.2927908

In the last few years, obtaining reproducible performance in distributed systems is gaining a lot of attention. This is due to the rapid adoption and diversification of cloud computing technology. The emergence of serverless computing poses additional challenges to such a problem.

Two opposite approaches can be adopted to assess the performance of these kinds of systems. On the one hand, empirical approaches focus on the analysis of the measurable performance of an existing system performing a series of experiments. In empirical studies, sound experimental methodology, and in particular reliable, consistent, and meaningful performance evaluation, is challenging but necessary [2]. On the other hand, theoretical approaches can create reliable models of the system under study, allowing for a deeper understanding of it [1]. Theoretical approaches typically require a design effort and may abstract from certain parts of the system that may be difficult to model.

I am interested in discussing what type of guarantees can be provided on serverless computing applications, and how such guarantees can be obtained through sound performance evaluation.

References

1 V. Gulisano, A. V. Papadopoulos, Y. Nikolakopoulos, M. Papatriantafilou, and P. Tsigas. Performance modeling of stream joins. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (DEBS)*, pages 191–202, New York, NY, USA, Jun. 2017. ACM.

2 A. V. Papadopoulos, L. Versluis, A. Bauer, N. Herbst, J. von Kistowski, A. Ali-Eldin, C. L. Abad, J. N. Amaral, P. Tuma, and A. Iosup. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Transactions on Software Engineering*, Jul. 2019.

3.25 Federated AI on Serverless Edge Clusters Powered by Renewable Energy

Panos Patros (University of Waikato, NZ)

Artificial Intelligence (AI) applications for agritech, such as robotic harvest and pollination, cannot be implemented without reliable and secure access to computing power. Adding extra hardware on robots increases design complexity, power requirements and weight. Outsourcing to unreliable and off-shore cloud providers increases operational risk and threatens data and economic sovereignty.

The proposed solution is to offer AI services locally via interconnected clusters powered by locally generated renewable energy. Crucially, these Rural AI clusters will maintain a reliable connection with robots, and will leverage advanced federated-learning algorithms and a serverless architecture to store/compute sensitive data locally; thus, only connecting to the cloud for low-risk operations.

A serverless architecture for federated edge learning will provide a seamless transition between edge and cloud computation, while offering a much needed fine-grain allocation (and costing) of scarce edge resources. Because of the limited resources of edge systems, platform innovations will be required to enable these technologies, leveraging prior experience in cloud computing [1, 2, 3, 4, 5, 6].

References

1 P. Patros, D. Dayal, K.B. Kent, M. Dawson, and T. Watson. *Multitenancy benefits in application servers*. Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering, 111-118, 2015

2 P. Patros, K.B. Kent, and M. Dawson. *SLO request modeling, reordering and scaling*. Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, 180-191, 2017

3 P. Patros, K.B. Kent, and M. Dawson. *Mitigating garbage collection interference on containerized clouds*. 2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 168-173, 2018

4 P. Patros, K.B. Kent, and M. Dawson. *Why is garbage collection causing my service level objectives to fail?*. International Journal of Cloud Computing, 7, 3-Apr, 282-322, 2018, Inderscience Publishers (IEL)

5 V. Podolskiy, Vladimir; M. Mayo; A. Koay; M. Gerndt; P. Patros. *Maintaining SLOs of cloud-native applications via self-adaptive resource sharing*. 2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 72-81, 2019

6 V. Podolskiy, M. Patrou, P. Patros, M. Gerndt, and K.B. Kent. *The weakest link: revealing and modeling the architectural patterns of microservice applications*, Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering, 2020, ACM

3.26 Is serverless computing the holy grail of fog computing application design paradigms?

Guillaume Pierre (University & IRISA – Rennes, FR)

Joint work of Guillaume Pierre, Arif Ahmed, Ali Fahs, Hamidreza Arkian, Paulo Souza junior, Mulugeta Ayalew Tamiru, Mozhdeh Farhadi

Main reference Arif Ahmed, HamidReza Arkian, Davaadorj Battulga, Ali J. Fahs, Mozhdeh Farhadi, Dimitrios Giouroukis, Adrien Gougeon, Felipe Oliveira Gutierrez, Guillaume Pierre, Paulo R. Souza Jr., Mulugeta Ayalew Tamiru, Li Wu: "Fog Computing Applications: Taxonomy and Requirements", CoRR, Vol. abs/1907.11621, 2019.

URL http://arxiv.org/abs/1907.11621

My research mostly focuses on the design of fog computing platforms. To process massive volumes of data being produced far from the data centers, fog computing extends cloud platforms with additional compute/storage/communication resources in the vicinity of the main sources of data, where these data can be (pre-)processed before reaching the cloud. Although this extension may seem trivial, it brings major new challenges in the way we

design these platforms. In particular, fog computing resources are located close to the main sources of data but necessarily far from each other. This means that it becomes much more difficult to share state between multiple fog nodes taking part in the same application. In this context, serverless computing provides an interesting programming paradigm which neatly separates stateless functions from stateful data services. In the Serverless workshop I tried to better understand the benefits and challenges brought about by this upcoming paradigm shift.

3.27 Performance Evaluation of Serverless Applications

Joel Scheuner (Chalmers and University of Gothenburg, SE)

Joint work of Simon Eismann, Joel Scheuner, Erwin Van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup
Main reference Simon Eismann, Joel Scheuner, Erwin Van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, Alexandru Iosup: "Serverless Applications: Why, When, and How?", IEEE Softw., Vol. 38(1), pp. 32–39, 2021.
URL https://doi.org/10.1109/MS.2020.3023302

Serverless applications typically combine event-triggered functions (i.e., FaaS) with scalable backend services (i.e., BaaS). However, such event-based integrations can lead to long delays that are difficult to debug in a distributed system. Therefore, my research aims to capture and explain application-level serverless performance through detailed tracing and reproducible experimentation.

My prior work consolidates 112 FaaS performance studies [1] and characterizes 89 serverless applications [2] both from academic and industrial sources. In the future, I am interested in performance-aware programming models where developers can indicate their performance-cost trade-off preferences and serverless applications optimize themselves accordingly.

References

1 J. Scheuner, P. Leitner, Function-as-a-Service Performance Evaluation: A Multivocal Literature Review. In *Journal of Systems and Software (JSS)*, Dec. 2020.
2 S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup. Serverless applications: Why, when, and how? *IEEE Software*, vol. 38, pp. 32–39, Jan 2021.

3.28 FaaS orchestration

Mina Sedaghat (Ericsson – Stockholm, SE)

Container orchestrators are often influenced by the application architectural models and their requirements. Modern applications (and their architectures) are getting more complex, often distributed geographically over a continuum of resources, and have stricter performance demands, i.e., on latency and data transfer. The evolution of the application architectures, from monoliths, to microservices and recently to Function as a Service (FaaS), puts new requirements on the orchestration systems and how the container deployment models should look like.

The current FaaS frameworks can only efficiently support a certain class of workloads, such as serving static content, time-based batch jobs, and ETL [6] jobs. They currently have a hard time supporting stateful applications with fine grain state sharing requirements. The basic assumption in the FaaS model is that functions are stateless, and if needed, they store their state using external storage. Therefore, stateful applications are currently constrained by limitations on existing cloud storage services, e.g. due to limited IO throughput and access latencies. I am, personally, interested in simplifying orchestration of Functions in a FaaS framework, providing support for a stateful applications, answering questions around data management, and finding solutions for a seamless orchestration of functions over a continuum of resources.

3.29 LaSS: Running Latency Sensitive Serverless Computations at the Edge

Prashant Shenoy (University of Massachusetts – Amherst, US) and Ahmed Ali-Eldin Hassan (Chalmers University of Technology – Göteborg, SE)

Joint work of Bin Wang, Ahmed Ali-Eldin, Prashant Shenoy
Main reference Bin Wang, Ahmed Ali-Eldin, Prashant J. Shenoy: "LaSS: Running Latency Sensitive Serverless Computations at the Edge", in Proc. of the HPDC '21: The 30th International Symposium on High-Performance Parallel and Distributed Computing, Virtual Event, Sweden, June 21-25, 2021, pp. 239–251, ACM, 2021.
URL https://doi.org/10.1145/3431379.3460646

Serverless computing has emerged as a new paradigm for running short-lived computations in the cloud. Due to its ability to handle IoT workloads, there has been considerable interest in running serverless functions at the edge. However, the constrained nature of the edge and the latency sensitive nature of workloads result in many challenges for serverless platforms. In this paper, we present LaSS, a platform that uses model-driven approaches for running latency-sensitive serverless computations on edge resources. LaSS uses principled queuing-based methods to determine an appropri- ate allocation for each hosted function and auto-scales the allocated resources in response to workload dynamics. LaSS uses a fair-share allocation approach to guarantee a minimum of allocated resources to each function in the presence of overload. In addition, it utilizes resource reclamation methods based on container deflation and termination to reassign resources from over-provisioned functions to under-provisioned ones. We implement a prototype of our ap- proach on an OpenWhisk serverless edge cluster and conduct a detailed experimental evaluation. Our results show that LaSS can accurately predict the resources needed for serverless functions in the presence of highly dynamic workloads, and reprovision con- tainer capacity within hundreds of milliseconds while maintaining fair share allocation guarantees.

References

1 Bin Wang, Ahmed Ali-Eldin and Prashant Shenoy *LaSS: Running Latency Sensitive Serverless Computations at the Edge*. Proceedings of ACM Symposium on High Performance Distributed Computing (HPDC) 2021.

[6] Extract, Transform and Load

3.30 Fitting Serverless Abstractions and System Designs to Next-Generation Application Needs

Josef Spillner (ZHAW – Winterthur, CH)

Main reference Josef Spillner: "Resource Management for Cloud Functions with Memory Tracing, Profiling and Autotuning", in Proc. of the WoSC@Middleware 2020: 2020 Sixth International Workshop on Serverless Computing, Virtual Event / Delft, The Netherlands, December 7-11, 2020, pp. 13–18, ACM, 2020.
URL https://doi.org/10.1145/3429880.3430094

Are today's serverless systems appropriate for emerging applications such as nation-scale digital services or massive IoT data stream processing? To answer that question, we need to reconsider system designs, programming abstractions and development tools.

On the system level, we investigate more light-weight isolation techniques including zero-coldstart microthreads. Software engineers can leverage these with syntactic constructs they already know in terms of coroutines, asynchronous processing and workers/tasklets. The aim is to reach beyond a few thousand instances per second to tens or hundreds of thousands of invocations, including light-weight state handling like with function-level ring buffers. We also study insights into application execution profiling and subsequent autotuning of memory allocation and other configuration parameters. Such techniques can help to reduce the overallocation of memory from the application engineer's perspective, to some extent with current statically allocated function instances and to an even greater extent with container isolations permitting dynamic memory updates. This is technically possible even with Docker containers, however the necessary APIs are not exposed by commercial FaaS/CaaS providers.

On the abstraction and tooling level, we explore the use of declarative code annotations to extract functions suitable for offloading computation. The function requirements are then matched as part of a FaaSification process against the cross-provider deployment and execution constraints. Furthermore, we observe static and dynamic characteristics of software artefacts representing serverless software – such as AWS SAM – to convey to software engineers whether there will be any problems or flaws especially when the artefacts originate from third-party dependencies.

3.31 Architectural Patterns for Serverless-Based applications

Davide Taibi (Tampere University, FI)

Main reference Davide Taibi, Nabil El Ioini, Claus Pahl, Jan Raphael Schmid Niederkofler: "Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review", in Proc. of the 10th International Conference on Cloud Computing and Services Science, CLOSER 2020, Prague, Czech Republic, May 7-9, 2020, pp. 181–192, SCITEPRESS, 2020.
URL https://doi.org/10.5220/0009578501810192

Companies are increasingly adopting Serverless, by migrating existing applications to this new paradigm. Different practitioners proposed patterns for composing and managing serverless functions. However, some of these patterns offer different solutions to solve the same problem, which makes it hard to select the most suitable solution for each problem.

In this work, we aim at supporting practitioners in understanding the different architectural patterns adopted by different companies, reporting benefits and issues of their applications.

This work proposal was initiated by a previous literature review [1] and is aimed at collecting experiences directly from practitioners by means of interviews and surveys, and to validate the resulting patterns with different collaborative empirical studies.

References

1 Taibi D., El Ioini N., Pahl C., Niederkofler J.R.S. Patterns for serverless functions (function-as-a-service): A multivocal literature review Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER'20) (2020), 10.5220/0009578501810192

3.32 Continuous testing of serverless applications

André van Hoorn (Universität Stuttgart, DE)

Joint work of André van Hoorn, Thomas F. Düllmann
Main reference Giuliano Casale, Matej Artac, Willem-Jan van den Heuvel, André van Hoorn, Pelle Jakovits, Frank Leymann, M. Long, V. Papanikolaou, D. Presenza, A. Russo, Satish Narayana Srirama, Damian A. Tamburri, Michael Wurster, Lulai Zhu: "RADON: rational decomposition and orchestration for serverless computing", SICS Softw.-Intensive Cyber Phys. Syst., Vol. 35(1), pp. 77–87, 2020.
URL https://doi.org/10.1007/s00450-019-00413-w

Quality assurance is a key software engineering activity to deliver high-quality software. The way software is being developed has changed dramatically over the past years, due to emerging cloud-native architectural styles, such as microservices and serverless, in combination with modern software engineering paradigms such as DevOps. The frequency and velocity of changes impose challenges to quality assurance, particularly for assessing runtime quality attributes such as performance and resilience. On the other hand, the new developments provide opportunities for novel quality assurance approaches, e.g., due to established technologies, a high degree of automation, and operational feedback from production. We investigate the interplay of the mentioned topics in the DevOps Performance Working Group of the SPEC RG. Concerning the seminar topic, my particular interest is in the question of "How to seamlessly integrate quality-of-service assurance for serverless into the DevOps ecosystem?".

Over the last year, I was involved in the EU Horizon 2020 project RADON on "Rational decomposition and orchestration for serverless computing". RADON provides an end-to-end framework to develop serverless applications, building on the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA). To assess whether applications developed via the RADON methodology and framework meet their quality requirements, RADON includes the continuous testing workflow, which particularly aims to support software developers, QoS engineers, and release managers in producing high-quality applications. The core component implementing the continuous testing workflow is the Continuous Testing Tool (CTT). CTT enriches the TOSCA ecosystem by end-to-end support for continuous testing of microservice-based (including FaaS) and data pipeline applications in DevOps. CTT supports the whole workflow – from test specification over execution and reporting to automated updates based on production data – that is also extensible to custom needs, e.g., integrating other types of tests or tools. A particular innovation lies in the integrative test generation features for obtaining tailored tests, which fits into the constraints of DevOps-based development settings with separate teams and delivery pipelines, and the goal of fast and frequent releases.

3.33 Serverless Compute Primitives as a Compilation Target

Soam Vasani (Stripe – San Francisco, US)

FaaS is a compelling compute primitive: it has the best elasticity that cloud compute has so far offered, and it abstracts away more infrastructure than any other compute primitive has so far. However application developers must account for FaaS limitations on timing, networking, artifact size, etc; these limitations have fundamental effects on application architectures.

This raises the question: can we have the elasticity and abstraction of FaaS without having to learn new application architecture patterns? To this end I'm interested in borrowing ideas from compilers: can we use FaaS and other serverless technologies (such as object stores and workflow runtimes) as a compilation target? In other words, can we transform a source program that is not serverless-specific to a set of functions, objects, and workflows? If this is not universally possible, then is there a set of source programs for which this is both possible as well as useful?

As a prototype, I'm exploring this question for the specific technologies of Python and AWS serverless, transforming functions written in a subset of Python into a set of Lambdas, Step Functions and S3 buckets.

3.34 Network Challenges in Serverless Computing

Florian Wamser (Universität Würzburg, DE)

Joint work of Nguyen Huu Thanh, Nguyen Trung Kien, Ngo Van Hoa, Truong Thu Huong, Florian Wamser, Tobias Hoßfeld

Main reference Nguyen Huu Thanh, Nguyen Trung Kien, Ngo Van Hoa, Truong Thu Huong, Florian Wamser, Tobias Hossfeld: "Energy-Aware Service Function Chain Embedding in Edge-Cloud Environments for IoT Applications", IEEE Internet of Things Journal, pp. 1–1, 2021.

URL https://doi.org/10.1109/JIOT.2021.3064986

The serverless computing paradigm promises a number of advantages over conventional cloud- or server-centered computing. Serverless computing offers the developer greater scalability and more flexibility at a lower cost. From the developer's point of view, one does not have to worry about the dimensioning, provision and administration of backend servers and hosts.

To provide flexibility, scalability, and developer-friendliness, a serverless platform typically manages and maintains the underlying resources. In addition to the computing resources, the *network* also plays a decisive role here, connecting the computing resources and transporting application requests to the serverless functions.

At the *University of Würzburg* we investigate the challenges for networks in connection with serverless computing. The most important points are:

1. Elasticity and scalability of network resources
2. Dynamic addressing and forwarding of requests to computing resources
3. Provision of network resources for functionality and adaptability

3.35 Decision Support for Modeling and Deployment Automation of Serverless Applications

Vladimir Yussupov (Universität Stuttgart, DE)

The term "serverless" gains more and more attention in the context of cloud-native application development. Frequently being associated exclusively with the Function-as-a-Service (FaaS) cloud service model, the idea of what a serverless application is keeps evolving, resulting in more issues to decide on when engineering serverless applications. I am interested in the topic of decision support for modeling and deployment of serverless architectures comprising various kinds of components such as FaaS platforms, function orchestrators, serverless databases and message queues. In particular, I am investigating which decisions need to be considered (also, decisions captured in the form of patterns), and how to use them to support practitioners in transitioning from abstract serverless application models to refined, provider-specific deployment models that can be enacted using deployment automation technologies of choice.

Some related publications:

- Yussupov, V.; Soldani, J.; Breitenbücher, U.; Brogi, A. and Leymann, F. (2021). From Serverful to Serverless: A Spectrum of Patterns for Hosting Application Components. In Proceedings of the 11th International Conference on Cloud Computing and Services Science – CLOSER
- Yussupov, V.; Soldani, J.; Breitenbücher, U.; Brogi, A.; Leymann, F. (2021). FaaSten your decisions: A classification framework and technology review of function-as-a-Service platforms, In Journal of Systems and Software, Volume 175

4 Working groups

4.1 Design of Serverless Systems, Platforms, and Ecosystems (Topic 1)

Samer Al-Kiswany (University of Waterloo, CA), Ahmed Ali-Eldin Hassan (Chalmers University of Technology – Göteborg, SE), André Bauer (Universität Würzburg, DE), André B. Bondi (Software Performance and Scalability Consulting LL, US), Ryan L. Chard (Argonne National Laboratory – Lemont, US), Andrew A. Chien (University of Chicago, US), A. Jesse Jiryu Davis (MongoDB – New York, US), Erik Elmroth (University of Umeå, SE), Alexandru Iosup (VU University Amsterdam, NL), Hans-Arno Jacobsen (University of Toronto, CA), Samuel Kounev (Universität Würzburg, DE), Vinod Muthusamy (IBM TJ Watson Research Center – Yorktown Heights, US), Guillaume Pierre (University & IRISA – Rennes, FR), Mina Sedaghat (Ericsson – Stockholm, SE), Prashant Shenoy (University of Massachusetts – Amherst, US), Davide Taibi (Tampere University, FI), Douglas Thain (University of Notre Dame, US), Erwin van Eyk (VU University Amsterdam, NL), and Soam Vasani (Stripe – San Francisco, US)

Organization

- Co-chairs: Alexandru Iosup and Samer Al-Kiswany
- Rapporteurs: Mina Sedaghat and Doug Thain

Opening Statement

This topic focuses on the design of serverless systems, platforms, and ecosystems. We organized the discussion for this topic around sessions, aiming to first obtain a diverse set of sub-topics, then to refine our own views about a focused set of sub-topics, then to take in new perspectives and share our own in discussion with the groups working on other topics, and, last, to refine our views toward a vision. Our sessions proceeded as follows:

- First session: We focused on scoping, discussing possible sub-topics for the design topic and trying to have as many ideas represented as possible.
- Second session: We focused on choosing and discussing 3 sub-topics, and on finalizing the scoping effort with ideas that arrived from the cross-pollination with other topics. We discussed requirements, including geo-distributed operation, the impact of state and data streams, predictable performance, energy awareness, security, and auditability through provenance provisions; we further discussed the actual complexity when automating the operational concerns for the user. We further discussed operational techniques for workload and resource management at runtime, and ensuring SLAs and SLOs while still being able to "make it easy for the user". We discussed lessons learned from cloud computing, especialy from PaaS (e.g., "We wanted all applications to be equally supported, with one simple and unified interface, but every significant application has at least something different").

- Third session: We analyzed existing serverless definitions, benefiting from discussion with other topics, and concluded on the key aspects of a good serverless definition. This allowed us to focus on a programming model, a reference architecture, non-functional requirements, patterns and anti-patterns in serverless applications in practice, and on toolchains. We also discussed programmability, portability, and interoperation.
- Fourth session: We had joint discussions with Topics 2 (software engineering) and 3 (application requirements). Main aspects discussed: What are the application domains and domain verticals? How to think about the user? What applications are the most important? What is the lifecycle of a serverless application? What sort of application architecture should be adopted? What is a good definition for serverless? How to think about managerial, policy, cloud, and resource-level metrics?
- Fifth and sixth sessions: concluding on the core definition, vision, challenges, etc.

Link to other topics

There is a strong link to the other topics, both conceptually and, following the joint sessions, practically:

- Questions related to representative use cases and applications, and to requirements related to them, are essential for the design topic and link strongly to Topic 3.
- Questions related to implementing and realizing the software of serverless systems, platforms, and ecosystems, including both the software patterns and the engineering process, are linked strongly with Topic 2.
- Questions related to testing and evaluating serverless systems, platforms, and ecosystems are linked strongly to topic 4.

A reference architecture for serverless computing

We discussed and agreed on a reference architecture for serverless computing. The reference architecture considers the following main layers:

1. Compute, memory, storage, and networking infrastructure, consisting primarily of (programmable) hardware devices and of corresponding virtualized devices.
2. Operating services, providing foundational services such as messaging, coordination, and authentication.
3. Resource managers, providing collections of (distributed) resources, physical and/or virtual, with pre-configured operating services, under a convenient programming interface.
4. Runtime engines, providing capabilities for executing simple and composite functions, up to orchestrating entire dataflows and workflows, and automating the back-end management of transient state and persistent data.
5. Front-end core, providing a programming model for serverless applications, specializations of this programming model for specific application domains, high-level programming languages for convenient programming, and portal and command-line high-level interfaces.
6. Across all layers, a toolchain of compilers, monitors, profilers, and benchmarks for serverless computing, helping optimize each aspect and making all levels observable.

Our main insight from the reference architecture is that the automated operation for serverless applications is an ecosystem, with many different parts developed and operated by different and autonomous organizations; this is very different from a single integrated system and leads to different design challenges and practices.

Another insight is that the designs of all the systems at different layers are highly influenced by the programming model, but currently no single programming model exists for serverless computing, and it is likely such a programming model will only be possible if it is very abstract and generic. We expect high levels of specialization and that serverless applications will use a special runtime stack as well as will often rely on back end services offered by the cloud providers.

We acknowledge that the serverless computing paradigm is in its infancy and many of the layers, especially the runtime engines, front-end core, and toolchains, will provide many radically different alternatives that will take time to mature and perhaps not converge.

Vision on the design of serverless systems, platforms, and ecosystems

We envision that serverless systems, platforms, and ecosystems should aim to:

- Ensure (nearly) complete automation of operational concerns and
- high programmability, portability, and interoperability, for
- diverse application domains and use cases, where
- many parts are defined once but used many times, with
- on-demand deployment, (geo-)distributed operation, and utilization-proportional cost, by
- offering diverse operational techniques (Rethinking resource management and scheduling),
- supporting and enforcing non-functional requirements, controlling for variability,
- rethinking observability and providing serverless-related monitoring,
- rethinking the static and dynamic toolchain, and
- ensuring integration with a diverse, evolving technology ecosystem.

Next to many challenges, which we list in the following, there are also *uncomfortable questions*, such as:

1. The question is not can we make remote execution as easy as local execution, but can we make it easier and more beneficial?
2. Industry is ahead and facing immediate challenges, so how should academics engage so they have impact in this field?
3. What is new, over the problems of full automation of 70 years ago (utility computing), 20 years ago (grid), 10 years ago (cloud)? (Serverless is not the entire cloud and should not try to do everything.)

Challenges in the design of serverless systems, platforms, and ecosystems

Related to the reference architecture:

C1 Capturing the multi-level architectural features and emerging architectural patterns of this rapidly evolving serverless computing field

C2 Predicting which architectural features and patterns will succeed, and explaining why (and why not others)

Related to full automation of operational concerns:

C3 Agreeing on a serverless definition and making it operational

Can we make it easy to run applications remotely? Can we achieve full transparency, in Coulouris' sense, a sort of "cloud button"? Can we achieve near-zero waste, scale to zero (cost)? (See also Section 5.1.)

C4 Understanding system-level, operational requirements

This includes understanding the stakeholders of serverless operations, the users of specific applications, and the systems-level requirements raised by industry verticals, application domains, and applications. Focus on both functional and non-functional requirements, and for non-functionals consider metrics at different levels of interest, from hardware resources to organization-wide managerial decisions. Focus on energy efficiency, but also on sustainability awareness (e.g., how the electricity used for serverless workloads is produced and consumed, how much greenhouse gas emissions and water consumption occurs here).

C5 Programming model from a systems perspective

What sort of application architecture should be adopted? What is the right granularity of the function? How to trade-off between providing control and simplicity? How to express and manage workflows (and is there a new way needed, or are workflow abstractions already sufficient)? What can we learn from decades of programming parallel and distributed systems?

C6 Workload and resource management for serverless, and overall routing and scheduling

How to extend and apply traditional techniques for workload and resource management? How to consider the full compute continuum (i.e., IoT/fog/edge/cloud)? How to replicate, cache, partition, consolidate, migrate, offload, etc. the functions and/or the data? How to provision, allocate, elastically scale, load-balance the resources? How to schedule and route across the whole (eco)system? How to consider resource provisioning over short periods of time (e.g., auto-scaling) and also long-term (e.g., capacity planning)?

C7 Practical needs in serverless orchestration

How to mix serverless with other computational models, i.e., run mixtures of workloads instead of merely serverless? How to achieve near-zero waste, even under complex deployment scenarios (e.g., geo-distributed scenarios)? How to reduce the serverless overhead added by the platform to the (commonly lightweight) functions representing the business logic of the application? How to ensure proper performance isolation while making efficient use of the shared infrastructure?

C8 Manage ecosystem instability

How to limit the impact of, e.g., performance variability, the impact of (correlated, even cascading) component downtime, multiple versions of the same service, service continuity under transience of various providers?

Related to the toolchain:

C9 Create the serverless toolchain

How does the traditional toolchain – Compiler, Linker, Loader Static Analysis, Dynamic Analysis, Dependency Detection, Testing, Debugging, Mocking, Tracing, Replaying – need to change for serverless, starting with FaaS? How to use self-descriptive metadata to improve (i) *safety* (type signature, semantics, version, dependencies, non-functional requirements, etc.), and (ii) *efficiency* (performance, resource requirements, co-location with other functions, etc.)? Process-wise, how to engage both toolchain and application developers, to motivate incremental deployment and interoperation of both metadata and tools, while accepting incomplete information?

C10 Support for patterns and anti-patterns, both functional and non-functional

What are the serverless patterns and anti-patterns, both functional and non-functional, that systems designers can work with? For example, what are the performance patterns and anti-patterns for serverless operations? How to support enterprise patterns, e.g., for integration or for distributed operation? How to support specific industry verticals or application domains, e.g., matching Topic 3: for scientific computing, for machine learning and artificial intelligence, for online gaming, and for mobile and telco operations?

Next steps and takeaway for the community

We have discussed the topic of design for serverless computing systems, platforms, and ecosystems. Linking to the other topics in this Dagstuhl Seminar, we have considered a definition for serverless computing, requirements from various application domains and use cases, software engineering concepts and processes, etc. We have provided in this section a summary of several critical aspects for design, including a reference architecture, a vision, and several uncomfortable questions and challenges.

The main takeaway for the community is that serverless computing poses hard, even grand challenges, related to full automation of operational concerns under hard constraints. The design of serverless systems, platforms, and ecosystems is an essential part of achieving the promise of serverless computing. The challenges we have listed here shape the task ahead, but there is more on the horizon.

As indicated by the value of the discussion we had with other topics in this seminar, designers should make sure the collaboration between computer systems, software engineering, performance engineering, and beyond to cross-disciplinary collaborations.

4.2 Software Engineering of Serverless Applications, but also Systems, Platforms, and Ecosystems (Topic 2)

Simon Eismann (Universität Würzburg, DE), Robert Chatley (Imperial College London, GB), Nikolas Herbst (Universität Würzburg, DE), Georgios Kousiouris (Harokopion University – Athens, GR), Philipp Leitner (Chalmers University of Technology – Göteborg, SE), Pedro García López (Universitat Rovira i Virgili – Tarragona, ES), Bernard Metzler (IBM Research-Zurich, CH), Davide Taibi (University of Tampere, FI), Vincent van Beek (Solvinity, Amsterdam and Delft University of Technology, NL), André van Hoorn (Universität Stuttgart, DE), Guido Wirtz (Universität Bamberg, DE), and Vladimir Yussupov (Universität Stuttgart, DE)

Opening Statement

The group discussed the topic of serverless from the perspective of software engineers in DevOps teams that are responsible for the development and operation of software systems running on serverless cloud platforms. The group decided to use the established stages of the software life-cycle to guide the discussion. For each stage, the group discussed how this stage is different from traditional software development when building serverless applications and what the resulting challenges are.

Changes to the Software Engineering Lifecycle

This section highlights the major differences in the engineering process when building serverless applications, compared to engineering traditional software architectures. Based on these differences, the group collected a number of software engineering challenges for serverless applications, which are discussed in the next section.

Planning

During the planning phase, the requirements of the application need to be collected and based on them the fundamental decisions about the application are made. For serverless applications, two additional decisions need to be made during the planning phase. The first one is whether serverless is actually well suited for this task. As there are still several limitations to serverless, it is not a suitable solution for every application, yet. The second decision that needs to be made is the selection of a cloud provider. While this was also a decision for traditional cloud applications, it's impact is far larger for serverless applications. The IaaS and container offerings of most cloud providers offer very similar features. However there are significant differences in the serverless offerings of different providers. While they all offer a function-as-a-service solution, there are large differences when it comes to the managed services. As cloud providers work to increase the number of specialized managed services, these differences will increase further.

Design

The key objective of the design phase is to come up with a suitable software architecture for the planned application. In this phase, the serverless application is split into coarse-grained, individual units (called microservices, components, or service). Within such a serverless microservice, there is a second, explicit architecture layer that describes the separation of code into serverless functions, incorporated external services, and the triggers that define the control flow within the application. This architecture within a service also implicitly exists within a traditional application in the form of software classes. However serverless makes this architecture explicit and forces developers to think of this low-level architecture before the implementation. This change increases the awareness of developers for the architecture of their application, and makes architecture diagrams for this second architecture level commonplace (in contrast to the often neglected UML diagrams).

Implementation

In the implementation phase, developers start implementing according to the requirements and the software architecture discussed in previous phases. A serverless application contains significantly less code than a traditional application, as much of the control flow and business logic is handled by managed services. However, these managed services and triggers need to be configured in the form of infrastructure-as-code (IaC) files. Therefore, developers spend a lot of time working on IaC files when building serverless applications. This means developers frequently need to context-switch between the actual code and the IaC file, as the functionality of the application is spread across both. While the tooling around code development is mature, the tooling around IaC and the integration of IaC and code is quite immature. This currently makes the development of serverless applications quite cumbersome.

Testing

The testing phase for serverless architectures must cover both functional and non-functional aspects. Unit tests can be implemented with relative ease due to the smaller granularity of functions and tested as usual for the chosen programming language. Integration tests become more important for serverless applications as the majority of behavior to test is located outside of the functions. However, integration testing also becomes more difficult as serverless applications rely on integration of multiple fine-grained components hosted using provider-managed services. Integration tests can be executed in a local environment using service emulators and available tooling, testing remotely on the provider's side, or a combination of both options. In practice, the hybrid testing option is currently quite common, since the local testing is limited w.r.t. available tooling and is not representative enough, whereas only remote testing incurs additional costs and takes longer as applications need to be redeployed with each update.

Deployment

During the deployment phase, all components and configurations of their interactions must be deployed to a target environment, meaning that not only the packaged source code has to be deployed, but also required event bindings need to be created, security policies configured, etc. As a result, a large part of the deployment requirements is at least partially addressed during the design and implementation phases: required component bindings are established either

in the source code or configuration files of the chosen deployment automation technology. The choice of the underlying deployment technology also defines which components can be deployed by it and in certain cases a combination of several technologies needs to be used, e.g., infrastructure deployment and configuration management using different automation tools.

Identified Challenges

Based on the changes to the software engineering lifecycle, the group identified a number of software engineering challenges for serverless applications. The table below shows how the identified challenges map to the software engineering lifecycle phases.

Table 1 Identified Challenges in the Lifecycle.

Lifecycle Phase	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Planning	X			X					X	X
Design				X	X	X	X		X	
Implementation			X			X	X	X		
Testing		X	X							
Deployment										

C1 Identifying whether a use case is serverless-ready

Serverless is quickly evolving and more and more use cases are becoming suitable. However, there are still limitations to serverless, which means it is not a suitable solution for every application, yet. There are currently no guidelines on how to identify whether serverless is suitable for a specific use case, which hinders the adoption of serverless.

C2 Testing serverless applications

Integration testing of serverless applications includes testing the configuration of managed services and function triggers. Local emulators for managed services and function integrations are difficult to build and maintain as serverless platforms are quickly evolving. Running integration tests directly on the cloud platform requires time-intensive deployments, which slows down the feedback cycle for developers.

C3 Debugging serverless applications

Triaging the cause of bugs in serverless applications is currently quite difficult. As a feature is often implemented by multiple functions and managed services, understanding what happened for a single request requires the logs from the multiple functions and services involved in the processing of the request. The non-standardized logging schemas of managed services and immature observability tooling makes this quite cumbersome.

C4 Predicting the costs of serverless applications

On the surface, the serverless billing model – pay-per-use – seems predictable. However, estimating the cost per request for a serverless API requires developers to understand the pricing models of all involved services. This is further complicated by the fact that many of the costs are dependent on data volumes and execution times which are often challenging to estimate.

C5 Determining function size

With serverless computing, applications are broken down into many serverless functions that are connected via managed services and event triggers. Developers often need to make the decision if a function is too large and should be split into two or more functions. There are currently few, and often conflicting guidelines on how to determine the appropriate size of a serverless function.

C6 Managing state in serverless functions

In their current state, serverless functions are stateless, which means that applications that require state can not be built purely from serverless functions. Instead, stateful information is currently stored in managed services such as databases or messaging services. Enabling functions to have some fast, shared state would not eliminate the need for databases or messaging services, but simplify the development of serverless applications and enable new use cases.

C7 Finding suitable abstract languages/models

Serverless applications are currently designed as architecture diagrams and implemented in the form of code and infrastructure as code definitions. Serverless could benefit from an intermediate language or model that could bridge the gap between the very coarse-grained, non-standardized architecture diagrams and the hard to understand combination of code and infrastructure as code.

C8 Reusing serverless functions

Serverless applications are broken down into small parts (functions) which can enable the reuse of existing functions in new contexts. However, managing functions within an organization at scale is currently challenging. Open questions here include how to determine what requirements and assumptions an existing function makes and whether reusing a function should include a separate deployment or the routing of requests to the existing function deployment.

C9 Migrating existing applications to serverless

Many existing applications could benefit from a partial or full migration towards serverless. However, many of these applications are not migrated, as developers are unaware of how to structure the migration, which parts to migrate first, and how to manage a serverless/serverful hybrid application. Additionally, this poses the challenge of how to train developers that are used to the serverful model in the skill required to build serverless applications.

C10 Vendor lock-in

Migrating a serverless application from one cloud provider to another cloud provider is very time-intensive and often requires partial rearchitecting of the application. Serverless offerings are mostly built on top of proprietary software instead of open-source solutions. This means that there is little to no compatibility between, e.g., the blob storage offerings of two cloud providers, which leads to the commonly reported vendor lock-in for serverless applications.

Closing Statement

The group discussed the changes to the traditional software engineering lifecycle from the perspective of software engineers that are responsible for the development and operation of software systems running on serverless cloud platforms. Based on these changes, the group identified a number of challenges for the planning, design, implementation and operation of serverless applications. While the discussion focussed mostly on these challenges, the group is confident that they can be overcome by a combined effort from industry and academia.

4.3 Serverless Applications and Requirements (Topic 3)

Josef Spillner (ZHAW – Winterthur, CH), Bartosz Balis (AGH University of Science & Technology – Krakow, PL), Jesse Donkervliet (VU University Amsterdam, NL), Nicola Ferrier (Argonne National Laboratory, US), Ian T. Foster (Argonne National Laboratory – Lemont, US), Maciej Malawski (AGH University of Science & Technology – Krakow, PL), Panos Patros (University of Waikato, NZ), Omer F. Rana (Cardiff University, GB), and Florian Wamser (Universität Würzburg, DE)

Why/when should applications be serverless?

As it stands, Serverless Computing expands on state-of-the-art cloud computing by further abstracting away software operations (ops) and larger parts of the hardware/software stack. One could consider functions, the execution unit of serverless computing, as "lightweight" containers, invoked with a set of inputs and expected to produce a set of outputs, when triggered.

From a user perspective, Serverless reduces system operation effort, simplifies development, supports highly variable and unpredictable workload patterns, enables the complete removal from dynamic memory of applications not in use – referred to as Scale to zero – and, under the right circumstances, can reduce software operation cost. From an operator perspective, it reduces costs by increasing resource efficiency and crucially, it incentivizes innovation for sustainability because the operator bears the cost of idleness.

Considering both the current state-of-the-art of serverless computing as well as its expected evolution over the year, this report aims to identify the types of applications that are currently well-supported by today's serverless platforms, and then, move on to discuss novel and upcoming applications with challenging characteristics, which would require serverless to evolve in order to satisfy them.

What are the unique characteristics?

We identified the following four unique characteristics (UC) for current serverless:

- UC-1 Stateless/Idempotency, which describes the pure-function behavior of invocations;
- UC-2 Fixed Memory, which limits the amount of resident memory an invocation is allowed to have;
- UC-3 Short Running, enforcing a short time limit on the execution of invocations;
- UC-4 Little Control, referring to the abstraction of ops, such as scheduling and autoscaling, from the user.

The transparency trade-off

However, are software engineers and problem owners ready to relinquish all this control of their applications? We identified a tradeoff between resource abstraction and resource control, essentially a tug-of-war between ease of programming vs. efficiency and cost.

This could be mitigated by exposing tuning knobs, such as resource management, to use by FaaS developers, a concept inspired by "open implementation analysis and design" [Maeda, Murphy, Kizales, 1997]. For a more technical example, consider the developer passing hints from the application through an interface exposed by the backend stack. This could be incarnated by pragmas, event interface, rate-limiting contracts, etc.

All in all, is it worthwhile to exchange ease of programming, deployment, maintenance and operation, to enable fine-grained control for developer customization? From a platform design perspective, such a requirement endangers efficiency in application and backend. However, it could help FaaS providers with resource allocation and scheduling decisions, while saving cost. Thus, the question persists if serverless should be even enabling any type of ops.

Scoping applications on their path towards serverless

The suitability of serverless computing concepts to deliver application functionality opens a maturity-chronology spectrum associated with application enablement. This spectrum can be roughly divided into three serverless phases: "Serverless 1.0" starting around 2014, "Serverless 1.5" representing the state of commercially available technology in 2021, "Serverless 2.0", bringing finer granularity and control in the coming years. Potentially more phases (3.0, 4.0, ...) will follow that are currently unclear but may nevertheless still not be sufficient for certain types of applications.

For some early adopter applications in the "serverless 1.0" phase, the initial serverless concepts around FaaS (λ, OW/ICF, GCF, AF) in the mid-2010s were already suitable. Further applications have been enabled recently by an expanded set of serverless computing offerings including FaaS-alike flavours of CaaS [GCR, Fargate, IBM CodeEngine, ACS/Dapper that permit stateful tasks/inter-instance communication/multiple CPUs, academic approaches like funcX], relaxed limits in FaaS invocations, and low-latency BaaS that characterise "serverless 1.5".

In the near future, based on recent scientific progress, "serverless 2.0" will make it easier to build applications that currently require unaffordable effort [e.g. ExCamera, deep learning, NumPyWren], and will furthermore allow for new classes that are currently unreachable [e.g.

online gaming, federated learning, agritech on the edge]. This trend will be driven primarily by four factors:

1. Hardware advances such as disaggregation and continuums; including better heterogeneity, specialised hardware (GPUs, TPUs), storage, networking (mobile radio heads)
2. Changes in BaaS, primarily the ubiquitous ability to run functions next to data (fusing the concepts of FaaS, stored procedures in DBs, UDFs in big data tools)
3. Autonomic middleware assisting the decomposition and placement of application code into managed services
4. Improvements in the design of serverless platforms, including
   a. selected control knobs for applications (possibly with some declarative language),
   b. including "only" a maximum runtime instead of a fixed short runtime (see "elastic" execution time for applications),
   c. differentiated and guaranteed quality (QoS or QoE guarantees) including real-time constraints (see also [RTserverless]), as well as
   d. aligned mature toolsets to convey the platform benefits directly to software engineers including testing, tracing and debugging, covering the entire DevOps cycle.

Application domains

The main application domains we discussed are:

1. Scientific Computing
2. Machine Learning and Artificial Intelligence
3. Online Gaming
4. Mobile Serverless and Telecommunication
5. Big Data Analysis
6. IoT, Agriculture and Cyber Physical Systems
7. Web Services

Below we focus on the selected four domains as representing the key challenges for current and future serverless platforms.

Domain: Scientific computing

Why use serverless?

Modern science relies on large scale experiments, simulations and data-driven analysis methods. Scientists analyze time series of global archives, often on the order of hundreds of Terabytes up to Petabytes, and hundreds of thousands of data images in order to generate a single layer of global, sometimes geospatial, information with added value. For this processing, extreme performance and often local processing is required. Computational requirements cover the full spectrum – from functions providing "support" for HPC environments to initial (approximate) analysis closer to the point of data capture. Many tasks are available as functions, can be reused and are available in R, R-Shiny Apps, Python, while we can think of larger HPC-jobs as "fat" functions that can be also considered serverless.

What can serverless provide today?

The serverless model is appealing for scientists because of the ease of programming, whereby scientists can focus on implementing scientific procedures, easily collaborate on function development, and reuse existing functions. Many scientific applications include fine-grained tasks, e.g. high-throughput scientific workflows, machine learning tasks, or interactive analytics.

Scheduling of tasks

Resource allocation associated with serverless platforms is highly dynamic and elastic, so the scientists can gain quick on-demand access to computing resources suitable for exploratory interactive data analysis, processing of streaming data from instruments etc. It is noteworthy that in the context of scientific computing the cold-start problem or high start-up times typical, e.g., for serverless containers, are less significant in comparison to job queue wait times in HPC systems. Accelerated time-to-science is thus another potential advantage of serverless computing applied to scientific use cases. Scientific applications are diverse in terms of software, complex dependencies on libraries, and packages, often requiring legacy software, so current approach to containerisation, deployable to serverless CaaS, is a perfect solution to these problems.

Challenging application requirements

Dynamic provisioning (on-demand access) is radically different from the typical batch-queue model used in scientific computing. Moreover, scientific computing often involves long-running tasks with high memory usage, while cloud functions currently are not suited to run as long as batch jobs and have fixed memory limits. Scientific applications often rely on specialized hardware, including all types of accelerators (GPU, potentially TPU for tensor tasks) and fast I/O (burst buffers, nvram) which are available in state-of-the-art HPC systems but not in cloud datacenters. For large-scale tightly-coupled parallel simulations fast interconnects and communication substrates are required (MPI), for which workarounds like using cloud storage or other means are now developed (NumPyWren), but need better solutions in the future. Scientific pipelines (workflows) benefit from data locality, difficult to achieve with stateless functions.

Ultimate vision

We envision that with Serverless 2.0 some of these requirements will be soon fulfilled, but the ultimate goal of "Serverless Supercomputer" will be possible not earlier than with the advent of "Serverless 4.0" era, where intra-datacenter latency will match the current leadership HPC interconnects and the distinction between a datacenter and supercomputer will disappear.

Domain: Machine learning and artificial intelligence

What are the characteristics of this domain?

Machine Learning is an emerging area in function-based processing, combining both learning on edge devices combined with inference-based models (MobileNetV1, MobileNetV2 and Faster R-CNN – i.e. pretrained models on cloud systems) that can be deployed on such

devices.[7] These computationally reduced versions of machine learning algorithms provide great opportunities for deploying function-based processing. Conversely, a number of hardware vendors (e.g. NVidia, Huawei, Intel, etc) are increasingly developing hardware accelerators aimed at improving the performance of machine learning algorithms, these range in complexity from support for specialist data structures (e.g. matrices and matrix/vector manipulation), to inclusion of specialist dedicated hardware that can be used to improve processing of data associated with machine learning algorithms (e.g. video analysis). Understanding how serverless approaches can be used to deploy (sustainably – combining both energy and economic efficiency) ML functions can be used to support a variety of different types of applications. To provide an example: the size of the models, number of parameters and computational complexity of these two MobileNet models include: MobileNetV1 (570M Multiply-Accumulate (MACs) and 4M parameters (which can include weights connecting layers and other model parameters such as learning rate)); MobileNetV2 (300M MACS, 3M parameters). Understanding benchmarks that can be used to characterise performance (and accuracy) of ML algorithms, realised as functions on edge devices is also being undertaken within the MLCommons Consortium (bringing together academia and industry). The benchmarks being proposed in this work could directly be used to undertake capacity planning for serverless implementations of ML functions.

ML functions can also vary in their computational time requirements – from algorithms that need to execute over long time frames (e.g. multiple days) to process different input data, to others that can be used to pre-process data prior to processing (<1min). Additionally, other ML pre-processing functions can be triggered by events observed in the environment (e.g. availability of sensor data, movement of people etc). Understanding where the serverless paradigm aligns with ML function implementation is an important consideration – as not all of these functions may be suitable to be realised using the serverless paradigm (especially when considering the economics of deployment). The following diagram demonstrates the possible mechanisms for distributing ML functions using serverless approaches: (i) we can partition the data (sharding of a data stream); (ii) partitioning a model (e.g. with the use of federated learning, where multiple models are independently constructed, and then integrated at a central site); (iii) aggregating the outcome of multiple functions and combining this with additional parameter optimisation using a cloud-based backend server.

What could serverless provide today?

Today's serverless can or, with modest system tweaks, could support ML and open up a number of opportunities in providing:

- Function-based implementation of ML algorithms at different levels of complexity, from ML that can be deployed within a data centre to functions at the edge;
- Programming support for implementing ML functions and developing software libraries that can be used to realise functions. A variety of libraries already exist – such as TF-Lite, use of "distillation" and quantization approaches to reduce the complexity of learned models to deploy over resource constrained environments. Another similar approach is the ability to migrate functions between edge and cloud resources – e.g. use of Osmotic computing approaches that enable migration of functions as lightweight containers;
- Deployment mechanisms that can be used to place ML functions across the IOT-edge-cloud continuum. Function placement driven by performance, cost and energy constraints can provide a useful basis for making more effective use of these within other application areas;

[7] https://dl.acm.org/doi/10.1145/3398020

- Serverless utilising increasingly available hardware accelerators – support for "hardware aware" function optimisation
- Specialist compilers that are able to create serverless functions that can be adapted to hardware characteristics

Challenging application requirements

Some of the characteristics and limitations of available serverless systems remain important open challenges, such as:

- Need to support often long-running functions that may have high memory and I/O requirements. Understanding whether a serverless approach would be most relevant in his context, and where alternative approaches may be more suitable for such deployments. Another challenge in this context would be understanding how to partition machine learning algorithms or general workloads across the iot-edge-cloud continuum.
- Need to support observability and manageability of functions, especially if these ML functions are part of other applications, for instance using a learning algorithm that is used as a component within a larger workflow. In this context, understanding the level of "control" a user has on configuring and deploying these ML functions remains an important overall consideration
- A deployment environment, e.g. as used in funcX/Parsl to dynamically deploy ML functions based on user demand, and aligned with the characteristics of the hardware platform. Matchmaking between function characteristics and hardware device properties also remains an important research challenge to increase adoption.
- Secure and privacy-aware ML functions, especially when dealing with sensitive data that may have GDPR/data privacy constraints, is also an important requirement for serverless deployment. Using encrypted data (e.g. using fully or partially homomorphic encryption) or utilising functions that carry particular security credentials also remains an important requirement for some ML applications. The research challenge here lies in identifying mechanisms for certifying ML functions based on "certificate servers" prior to their use.

Domain: Online Gaming

Gaming is a massive industry, generating a revenue of $180 billion in 2020.[8] But despite its size, developing and operating games and their surrounding ecosystems is challenging. We envision today's and future serverless technology addressing these challenges. In this section, we argue why online gaming can benefit from serverless, how today's serverless technology can help, and in which direction serverless technology needs to develop to better support the online gaming domain.

Benefits

The characteristics of serverless is promising for the online gaming domain. Without being comprehensive, we discuss here three areas where serverless can help. First, online games typically have large workload variance over time.[9] The popularity of games is difficult to

[8] https://www.marketwatch.com/story/videogames-are-a-bigger-industry-than-sports-and-movies-combined-thanks-to-the-pandemic-11608654990
[9] https://atlarge-research.com/pdfs/2011-nae-dynamic.pdf

predict, but can be the difference between attracting tens or tens of millions of players. Importantly, the number of players in an online game typically goes down over time, and many games fail to attract a significant number of players. To manage this risk, game developers need the ability to scale to zero. At the same time, successful games that attract large numbers of users have significant daily, weekly, and yearly workload variation patterns. Games and their ecosystem services require strong scalability to support this. Second, Successful online games require continuous operational support to provide a service to players and meet QoS and other NFR constraints. Because such support is labor intensive, it requires risky and costly investments from development companies and individual developers. Third, online games operate as parts of a large ecosystem, and as such require good integration (e.g., high availability, scalability, fault tolerance) with other services. Using serverless applications can simplify development effort required to meet these goals.

Technology assessment

Today's serverless platforms are a promising technology for several areas of the online gaming ecosystem. Using the house-like metaphor from Iosup et al.,[10] we envision serverless technology to automatic content generation (e.g., generating worlds on demand), game analytics (e.g., analyzing player behavior and detecting toxicity), the social meta game (e.g., web apps where users share player-created content), and the virtual world (e.g., player authentication, matchmaking).

Future directions

While these applications are promising, we identify several challenges that require serverless to develop beyond its current capabilities. We briefly describe three such challenges here. First, games can have stringent QoS requirements such as low jitter and latency in the range of tens of milliseconds, which requires low (cold) start times and guarantees on tail latencies, and good performance isolation to prevent performance variability in areas that will result in reduced player experience. Second, using an architecture with large numbers of small components can make keeping consistent state between players will become more difficult. Third, the gaming ecosystem contains parts (such as game server instances) that do not use a programming model that fits easily with the request/reply model used by today's FaaS platforms.

Domain: Mobile and telco serverless

Overview

The mobile networking area is currently seeing a significant increase in connected devices, including IoT devices and smart mobile devices. Due to the operators' business models and the potential for additional revenue models for network and telcos, operators are forced to act efficiently and in line with the demands. In particular, this means being efficient in the direction of scaling and elasticity. More precisely, mobile networks are typically geographically distributed and have to deal with a highly variable amount of device messages at the edge and on central entities. This requires a massive scaling in both directions, spatially and in terms of

[10] https://arxiv.org/pdf/1802.05465.pdf

resources – all things that Serverless entails. Contrary to requirements, Mobile Serverless also has some inherent functionality that conforms to the serverless paradigm: many network core functions are sold today as software to avoid large, rigid hardware boxes. Cellular functions are already available today as separate functions (virtual network functions, especially with the use of Software-Defined Networking (SDN) environments). These functions are usually already short-running and often even already stateless.

Benefits

The most important points that can be envisioned for Mobile Serverless are: it leverages the efficient and scalable architecture, which provides benefits from reduced operational costs and function isolation. Mobile core network functions are expected to be fully integrated into or partially merged with user functions. Besides these specific points, Mobile Serverless can generally benefit from better maintainability and updatability as mobile functions are encapsulated in atomic tasks or functions, including better resilience with the replication of functions for fallback purposes and chaos monkey functions.

Challenges

For the next generation of Serverless Computing the challenging application requirements include runtime and latency guarantees for network and signaling processing functions, the cold start problem (especially with distributed deployment), and locality, since some functions must be performed in specific locations. Ultimately, more and more security and privacy requirements play a role in the discussion with Serverless, as mobile networks typically offer larger attack surfaces and have many common elements where sensitive information about users is stored.

Anti-Hypothesis: What is "not" serverless, and why/when should applications sometimes not be serverless?

Having the above mentioned application domains in mind, significant assumptions exist in current vendor-based serverless models regarding function execution time, memory constraints, "cold start" overhead, and execution costs (e.g. per unit time execution costs $\gg$ VM/container execution costs).

The question arises if these constraints are inherent to the serverless model or just technical limitations which will be relaxed in the future?

- Are they really constrained or just driven by economic models?
- What is not serverless, i.e. when should we just not use serverless?
- What is the anti-pattern equivalent for serverless? (Definition: "An anti-pattern is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.")

From a general point of view:

1. The first obvious thing that strikes against the use of serverless for some applications is the fact that partitioning an application into functions may take too long or can even be counterproductive.

2. Sometimes applications depend on a strong requirement of I/O or other hardware components like shared or fast memory, which is hardly possible to achieve in serverless at the moment.
3. In addition to the last point, heterogeneity given by functions across multiple hardware and infrastructure can also lead to severe challenges in resilience and coordination.
4. There is furthermore limited reproducibility in serverless computing, which also applies to the performance of these functions.
5. It is also important to note that applications can require particular QoS and QoE guarantees (e.g. gaming) that have to be supported across multiple executions of these functions.
6. Finally, there are also restrictions from the architecture and platform side in direction to elasticity for applications – consider for example that the "unlimited resource" assumption does not completely hold for edge devices in case your platform spans over heterogeneous computing devices.

From a technical point of view, typical restrictions and the ones above arise from the fact that Serverless Computing is based on a number of paradigms, including the fact that simple direct communication between functions is not normally possible. Often there is also only a limited amount of synchronization possible and serverless commonly only allows few modifications and control of the workflow and scheduling of functions, which is required by some applications. One severe problem, regarding the performance point of view, is also that shared memory capabilities and heavy memory optimization is not possible. Ideas like OpenMP will not be possible since such approaches require strict locality of the memory.

Next steps – Takeaway for the wider community

We ask interested research communities (cloud and systems, software engineering, performance) to reflect on better application enablement. This encompasses concrete actions such as:

1. Helping to complete the transition to "Serverless 2.0" by measuring and optimising the recently introduced prototypes, both from industry and from academia, to overcome current limits in massive scalability and startup latency.
2. Understanding the cost and economics of using Serverless functions in applications, and developing a "cost calculator" that is able to make effective assessment of potential costs for an application user (along similar lines to AWS Cost Calculator).
3. Performing more empirical research with companies to also learn from failures and hesitation in addition to success cases. This will uncover current system limitations and turn that into common knowledge, not confined to the serverless platform product owners.
4. Support for serverless functions that can co-exist and meet different types of application requirements – such as security, performance, usability etc. Security remains an important challenge and will become increasingly important as new platforms and applications communities make use of Serverless.
5. Answering fundamental questions such as the "greenness" and cost efficiency of serverless computing in a way that practical advice for application engineers can be derived.
6. Co-designing forward-looking serverless architectures that abstract from the underlying isolation layers (container, µ-VM, WASM) and are prepared to work in heterogeneous hardware environments. The co-design should be conducted in conjunction with "borderline applications" that are not just yet enabled but might be with the new design.

This way, progress into the next stages of serverless computing can be documented with timestamped examples. Examples include smart edge-based systems with dynamic resource autodiscovery, uniform management interfaces, and awareness about end-to-end characteristics such as networked invocation duration to account for latency variations, interrupted connections and other QoS concerns, for instance in connected vehicles.

4.4 Evaluation of Serverless Systems (Topic 4)

Cristina Abad (ESPOL – Guayaquil, EC), Kyle Chard (University of Chicago, US), Pooyan Jamshidi (University of South Carolina – Columbia, US), Alessandro Vittorio Papadopoulos (Mälardalen University – Västerås, SE), Robert P. Ricci (University of Utah – Salt Lake City, US), Joel Scheuner (Chalmers and University of Gothenburg, SE), Mohammad Shahrad (University of British Columbia – Vancouver, CA), and Alexandru Uta (Leiden University, NL)

Opening Statement

The group discussed the topic of serverless computing from the perspective of performance evaluation and benchmarking of the serverless platforms and applications that can be built on those platforms.

Reproducibility in serverless

One of the overarching challenges in computer science is reproducibility [1, 2, 3]. Previous studies focusing on cloud computing [4] have already shown that results, especially performance data [5] are difficult to reproduce across studies. We expect this behavior to be exacerbated in serverless scenarios: On the one hand, from the client perspective, the underlying system is opaque. On the other hand, cloud providers have a clear view of the system design and implementation, but the client workloads are opaque to them. We believe this is an opportunity for the two parties to work together toward achieving better experimental reproducibility.

Directions

The performance evaluation of serverless systems can be classified into six types of evaluations, according to their goal. We describe each, next.

Evaluation of existing platforms and reverse-engineering:

Performed when we want to know well a serverless platform performs; for example, as in [6]. This type of evaluation primarily employs micro-benchmarks to measure a very specific resource such as CPU speed for floating-point operations. The evaluation results can be used to choose a suitable service, optimize configurations, guide design decisions of serverless applications, or parametrize a theoretical performance model.

Application-level benchmarks:

Application benchmarks focus on explaining the performance behavior of a known application under a serverless system. These benchmarks select representative applications motivated by real-world use cases and test them under realistic workloads. An example evaluation of a serverless application is presented in the ExCamera paper [7].

Middleware/frameworks:

Researchers and developers need a way to evaluate the performance of middlewares or frameworks, layers that are built on top of Function-as-a-Service platforms but are not user-focused (e.g., a workflow manager). The goal being, frequently, to preserve performance and reliability, while decreasing cost. An example of this type of evaluation can be seen in [8].

Workload characterization:

Like other cloud services, serverless offerings host a wide range of users running various applications. Characterizing the serverless workload enables discovering usage patterns, modeling resource consumption, and understanding the composition of serverless applications. For example, Microsoft's characterization study [9] came with open-source traces on invocation times alongside function duration and memory usage distributions for each application.

Systems design / development / solution evaluations:

Frequently, changes are made to the inner workings of serverless platforms like OpenWhisk or OpenFaaS. These studies seek to improve parts of the stack, like the scheduler (maybe while stubbing/simulating other parts); for example, as in [10]. Such inner working can be at different layers in the computer system stack including software, computer architecture, and hardware.

Other:

In the other category, we include any study that does not fit in the prior five categories; for example, software engineering papers decomposing monoliths, or papers that use serverless to test/validate something else. The former includes case studies showing how (typically monolithic) applications can be re-architected to work with a serverless design. For a concrete study that illustrates this category, consider [11].

Performance evaluation approaches

In addition to the classification of performance evaluation of serverless systems according to their goal, the group also discusses the different performance evaluation approaches that researchers can take. **Empirical** evaluations employ an observation-based approach in which the system is deployed in a testbed, a workload issued, and results observed and analyzed. **Theoretical** evaluations start with a model and try to reason about a system through analysis or simulations: What are the inputs and how do these inputs affect the performance? Theoretical approaches are particularly useful for predictions, and real-time decisions (e.g., scheduling, offloading). **Hybrid** approaches that combine the two prior approaches can be used for sensitivity analysis, to identify the top x parameters to configure or tune.

Next steps and takeaway for the community

To enable better evaluations, industry should release traces that can be analyzed, modeled, and replayed. The community would also benefit significantly if cloud providers were to publish how their serverless systems work internally. We need platform and application benchmarks, and we need these to be based on a solid understanding of how actual applications use serverless frameworks. Less explored and equally important are analytical frameworks that can be used to explore pricing policies that can be beneficial for providers and consumers.

References

1 X. Chen, S. Dallmeier-Tiessen, R. Dasler, S. Feger, P. Fokianos, J. B. Gonzalez, H. Hirvonsalo, D. Kousidis, A. Lavasa, S. Mele *et al.*, "Open is not enough," *Nature Physics*, vol. 15, no. 2, pp. 113–119, 2019.

2 B. Haibe-Kains, G. A. Adam, A. Hosny, F. Khodakarami, L. Waldron, B. Wang, C. McIntosh, A. Goldenberg, A. Kundaje, C. S. Greene *et al.*, "Transparency and reproducibility in artificial intelligence," *Nature*, vol. 586, no. 7829, pp. E14–E16, 2020.

3 E. van der Kouwe, G. Heiser, D. Andriesse, H. Bos, and C. Giuffrida, "Sok: Benchmarking flaws in systems security," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 310–325.

4 A. V. Papadopoulos, L. Versluis, A. Bauer, N. Herbst, J. Von Kistowski, A. Ali-Eldin, C. Abad, J. N. Amaral, P. Tuma, and A. Iosup, "Methodological principles for reproducible performance evaluation in cloud computing," *IEEE Transactions on Software Engineering*, 2019.

5 A. Uta, A. Custura, D. Duplyakin, I. Jimenez, J. Rellermeyer, C. Maltzahn, R. Ricci, and A. Iosup, "Is big data performance reproducible in modern cloud networks?" in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 2020, pp. 513–527.

6 L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift, "Peeking behind the curtains of serverless platforms," in *USENIX Annual Technical Conference (USENIX ATC)*, 2018, pp. 133–146.

7 S. Fouladi, R. S. Wahby, B. Shacklett, K. V. Balasubramaniam, W. Zeng, R. Bhalerao, A. Sivaraman, G. Porter, and K. Winstein, "Encoding, fast and slow: Low-latency video processing using thousands of tiny threads," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 363–376.

8 S. Fouladi, F. Romero, D. Iter, Q. Li, S. Chatterjee, C. Kozyrakis, M. Zaharia, and K. Winstein, "From laptop to lambda: Outsourcing everyday jobs to thousands of transient functional containers," in *USENIX Annual Technical Conference (USENIX ATC)*, 2019, pp. 475–488.

9 M. Shahrad, R. Fonseca, Í. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *USENIX Annual Technical Conference (USENIX ATC 2020)*, 2020, pp. 205–218.

10 E. Oakes, L. Yang, D. Zhou, K. Houck, T. Harter, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "SOCK: Rapid task provisioning with serverless-optimized containers," in *USENIX Annual Technical Conference (USENIX ATC)*, 2018, pp. 57–70.

11 A. Goli, O. Hajihassani, H. Khazaei, O. Ardakanian, M. Rashidi, and T. Dauphinee, "Migrating from monolithic to serverless: A fintech case study," in *Companion of the ACM/SPEC International Conference on Performance Engineering*, 2020, pp. 20–25.

5 Panel discussions

5.1 Toward a Definition for Serverless Computing

Samuel Kounev (Universität Würzburg, DE), Cristina Abad (ESPOL – Guayaquil, EC), Ian T. Foster (Argonne National Laboratory – Lemont, US), Nikolas Herbst (Universität Würzburg, DE), Alexandru Iosup (VU University Amsterdam, NL), Samer Al-Kiswany (University of Waterloo, CA), Ahmed Ali-Eldin Hassan (Chalmers University of Technology – Göteborg, SE), Bartosz Balis (AGH University of Science & Technology – Krakow, PL), André Bauer (Universität Würzburg, DE), André B. Bondi (Software Performance and Scalability Consulting LL, US), Kyle Chard (University of Chicago, US), Ryan L. Chard (Argonne National Laboratory – Lemont, US), Robert Chatley (Imperial College London, GB), Andrew A. Chien (University of Chicago, US), A. Jesse Jiryu Davis (MongoDB – New York, US), Jesse Donkervliet (VU University Amsterdam, NL), Simon Eismann (Universität Würzburg, DE), Erik Elmroth (University of Umeå, SE), Nicola Ferrier (Argonne National Laboratory, US), Hans-Arno Jacobsen (University of Toronto, CA), Pooyan Jamshidi (University of South Carolina – Columbia, US), Georgios Kousiouris (Harokopion University – Athens, GR), Philipp Leitner (Chalmers University of Technology – Göteborg, SE), Pedro García López (Universitat Rovira i Virgili – Tarragona, ES), Martina Maggio (Universität des Saarlandes – Saarbrücken, DE), Maciej Malawski (AGH University of Science & Technology – Krakow, PL), Bernard Metzler (IBM Research-Zurich, CH), Vinod Muthusamy (IBM TJ Watson Research Center – Yorktown Heights, US), Alessandro Vittorio Papadopoulos (Mälardalen University – Västerås, SE), Panos Patros (University of Waikato, NZ), Guillaume Pierre (University & IRISA – Rennes, FR), Omer F. Rana (Cardiff University, GB), Robert P. Ricci (University of Utah – Salt Lake City, US), Joel Scheuner (Chalmers and University of Gothenburg, SE), Mina Sedaghat (Ericsson – Stockholm, SE), Mohammad Shahrad (University of British Columbia – Vancouver, CA), Prashant Shenoy (University of Massachusetts – Amherst, US), Josef Spillner (ZHAW – Winterthur, CH), Davide Taibi (Tampere University, FI), Douglas Thain (University of Notre Dame, US), Animesh Trivedi (VU University Amsterdam, NL), Alexandru Uta (Leiden University, NL), Vincent van Beek (Solvinity, Amsterdam and Delft University of Technology, NL), Erwin van Eyk (VU University Amsterdam, NL), André van Hoorn (Universität Stuttgart, DE), Soam Vasani (Stripe – San Francisco, US), Florian Wamser (Universität Würzburg, DE), Guido Wirtz (Universität Bamberg, DE), and Vladimir Yussupov (Universität Stuttgart, DE)

A definition is the first principle of any field of human inquiry. As for many other complex issues, for serverless computing the semantics have become source of commentary and debate. So, what is the object of our inquiry, *what is serverless computing?*

Many of this Dagstuhl Seminar attendees engaged in early discussions around this question. Early definitions include aspects such as: the deployment model of Function-as-a-Service and Backend-as-a-Service being key to operate complex serverless applications [1]; granular billing matching actual use, event-driven operation, and (almost) complete lack of operational

logic [1]; (almost) no concerns about operation for the user, function lifecycle management including events as triggers, operations including performance isolation and prediction, operations to trade-off cost and performance under guidance from the user [2]; the details of FaaS operation that users can expect to encounter, as a reference model spanning functions to workflows [3]; etc.

With so many aspects to consider, a definition remained elusive. During the seminar, an intense discussion thread started resulting in an improved common understanding of the notion of "serverless computing", around notions such as:

1. **NoOps:** Hiding/abstracting complexity of *execution environment* (physical and virtual machines, hypervisors, operating systems, containers, etc.) as well as *system/operation aspects*, such as resource management, component/instance deployment, instance lifecycle, elasticity/autoscaling, reliability/fault-tolerance, ...
2. **Utilization-based billing:** a billing model that only charges for the *resources actually used* both with respect to time and space, for example, for Function-as-a-Service (FaaS) this translates into "pay only for function execution (space dimension) in fine-granular time units (time dimension)".

Based on this, we formulate the following definition of serverless computing:

Definition: *Serverless computing* is a cloud computing paradigm offering a high-level application programming model that allows one to develop and deploy cloud applications without allocating and managing virtualized servers and resources or being concerned about other operational aspects. The responsibility for operational aspects, such as fault tolerance or the elastic scaling of computing, storage, and communication resources to match varying application demands, is offloaded to the cloud provider. Providers apply utilization-based billing: they charge cloud users in proportion to the resources that applications actually consume from the cloud infrastructure, such as computing time, memory, and storage space.

Today, Function-as-a-Service (FaaS) platforms are the most prominent example of serverless computing offerings. Current FaaS platforms focus on the function as a unit of computation assumed to be small, stateless, and event-driven (i.e., executed asynchronously in response to certain triggers or events). The short runtime and stateless nature of FaaS functions makes it easier for FaaS cloud providers to implement autoscaling in a generic manner, while applying a fine-granular utilization-based cost model that bills customers based on the actual time functions are running. Given its popularity and rapid adoption, today FaaS is often used interchangeably with serverless computing.

We believe that the current assumptions of the FaaS model (small, stateless, and event-driven units of computation) might eventually be relaxed, as platforms evolve to support a wider set of applications. In addition to FaaS platforms, our broad notion of serverless computing also explicitly includes modern Backend-as-a-Service (BaaS) offerings, which are focused on specialized cloud application components, such as object storage, databases, or messaging. Finally, some Software-as-a-Service (SaaS) platforms support the execution of user-provided functions tightly coupled to the specific application domain. In summary, the serverless ecosystem includes a growing set of technologies and evolving programming models (e.g., FaaS, BaaS, some PaaS and SaaS), which, taken together, will provide the basis for building (end-to-end) next-generation serverless cloud applications.

Please cite as:

Samuel Kounev et al., Toward a Definition for Serverless Computing, in Serverless Computing (Dagstuhl Seminar 21201) (Cristina Abad, Ian T. Foster, Nikolas Herbst, and Alexandru Iosup, eds.), vol. 11(4), Chapter 5.1, p.34–93, Schloss Dagstuhl Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021.

or using

```
@Incollection{kounev_et_al:DagRep.11.4.34:ServerlessNotion,
  author = {Samuel Kounev and Cristina Abad and Ian T. Foster and
Nikolas Herbst and Alexandru Iosup and Samer Al-Kiswany and
Ahmed Ali-Eldin Hassan and Bartosz Balis and Andr\'e Bauer and
Andr\'e B. Bondi and Kyle Chard and Ryan L. Chard and
Robert Chatley and Andrew A. Chien and A. Jesse Jiryu Davis and
Jesse Donkervliet and Simon Eismann and Erik Elmroth and
Nicola Ferrier and Hans-Arno Jacobsen and Pooyan Jamshidi and
Georgios Kousiouris and Philipp Leitner and Pedro Garcia Lopez and
Martina Maggio and Maciej Malawski and Bernard Metzler and
Vinod Muthusamy and Alessandro V. Papadopoulos and
Panos Patros and Guillaume Pierre and Omer F. Rana and
Robert P. Ricci and Joel Scheuner and Mina Sedaghat and
Mohammad Shahrad and Prashant Shenoy and Josef Spillner and
Davide Taibi and Douglas Thain and Animesh Trivedi and
Alexandru Uta and Vincent van Beek and Erwin van Eyk and
Andr\'e van Hoorn and Soam Vasani and Florian Wamser and
Guido Wirtz and Vladimir Yussupov},
  title = {{Toward a Definition for Serverless Computing}},
  booktitle = {{Serverless Computing (Dagstuhl Seminar 21201)}},
  pages = {34--93},
  journal = {Dagstuhl Reports},
  year = {2021},
  volume = {11},
  issue = {4},
  editor = {Cristina Abad and Ian T. Foster and Nikolas Herbst and
Alexandru Iosup},
  publisher = {Schloss Dagstuhl -- Leibniz-Zentrum f{\"u}r Informatik},
  address = {Dagstuhl, Germany},
  doi = {10.4230/DagRep.11.4.34},
}
```

References

1 Erwin Van Eyk, Alexandru Iosup, Simon Seif, Markus Thömmes: *The SPEC cloud group's research vision on FaaS and serverless architectures*. WOSC@Middleware 2017: 1-4

2 Erwin Van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, Simon Eismann: *A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures*. ICPE Companion 2018: 21-24

3 Erwin Van Eyk, Alexandru Iosup, Johannes Grohmann, Simon Eismann, André Bauer, Laurens Versluis, Lucian Toader, Norbert Schmitt, Nikolas Herbst, Cristina L. Abad: *The SPEC-RG Reference Architecture for FaaS: From Microservices and Containers to Serverless Platforms*. IEEE Internet Comput. 23(6): 7-18 (2019)

Participants

- André Bauer
Universität Würzburg, DE
- Simon Eismann
Universität Würzburg, DE
- Nikolas Herbst
Universität Würzburg, DE

Remote Participants

- Cristina Abad
ESPOL – Guayaquil, EC
- Samer Al-Kiswany
University of Waterloo, CA
- Ahmed Ali-Eldin Hassan
Chalmers University of Technology – Göteborg, SE
- Bartosz Balis
AGH University of Science & Technology – Krakow, PL
- André B. Bondi
Software Performance and Scalability Consulting LL, US
- Kyle Chard
University of Chicago, US
- Ryan L. Chard
Argonne National Laboratory – Lemont, US
- Robert Chatley
Imperial College London, GB
- Andrew A. Chien
University of Chicago, US
- A. Jesse Jiryu Davis
MongoDB – New York, US
- Jesse Donkervliet
VU University Amsterdam, NL
- Erik Elmroth
University of Umeå, SE
- Nicola Ferrier
Argonne National Laboratory, US
- Ian T. Foster
Argonne National Laboratory – Lemont, US
- Alexandru Iosup
VU University Amsterdam, NL
- Hans-Arno Jacobsen
University of Toronto, CA
- Pooyan Jamshidi
University of South Carolina – Columbia, US
- Samuel Kounev
Universität Würzburg, DE
- Georgios Kousiouris
Harokopion University – Athens, GR
- Philipp Leitner
Chalmers University of Technology – Göteborg, SE
- Pedro García López
Universitat Rovira i Virgili – Tarragona, ES
- Martina Maggio
Universität des Saarlandes – Saarbrücken, DE
- Maciej Malawski
AGH University of Science & Technology – Krakow, PL
- Bernard Metzler
IBM Research-Zurich, CH
- Vinod Muthusamy
IBM TJ Watson Research Center – Yorktown Heights, US
- Alessandro Vittorio Papadopoulos
Mälardalen University – Västerås, SE
- Panos Patros
University of Waikato, NZ

- Guillaume Pierre
University & IRISA – Rennes, FR
- Omer F. Rana
Cardiff University, GB
- Robert P. Ricci
University of Utah – Salt Lake City, US
- Joel Scheuner
Chalmers and University of Gothenburg, SE
- Mina Sedaghat
Ericsson – Stockholm, SE
- Mohammad Shahrad
University of British Columbia – Vancouver, CA
- Prashant Shenoy
University of Massachusetts – Amherst, US
- Josef Spillner
ZHAW – Winterthur, CH
- Davide Taibi
University of Tampere, FI
- Douglas Thain
University of Notre Dame, US
- Animesh Trivedi
VU University Amsterdam, NL
- Alexandru Uta
Leiden University, NL
- Vincent van Beek
Solvinity, Amsterdam and Delft University of Technology, NL
- Erwin van Eyk
VU University Amsterdam, NL
- André van Hoorn
Universität Stuttgart, DE
- Soam Vasani
Stripe – San Francisco, US
- Florian Wamser
Universität Würzburg, DE
- Guido Wirtz
Universität Bamberg, DE
- Vladimir Yussupov
Universität Stuttgart, DE