Scheduling of Functions on Serverless Platforms

A. A. Da Silva, Y. Georgiou, M. Mercier, G. Mounié, D. Trystram,
Université Grenoble Alpes, Ryax Technologies

3 of December of 2021

Agenda

- Serverless Computing
  - Micro-Services and Containers
  - Cloud, Edge and Serverless Platforms
  - Serverless Computing in Practice
- Methodology
  - Scheduling of Functions
  - Benchmarks
  - A Simulated Environment
- Experimental Results
- Proposed Allocation Policies
- Future Work

Serverless Computing

Serverless Computing has emerged as a new paradigm of abstraction, platform and implementation of cloud functions [1,2], presenting an evolution of the Cloud Computing model in the sense of use of micro-services and containers.

Serverless Computing

Serverless Computing has emerged as a new paradigm of abstraction, platform and implementation of cloud functions [1,2], presenting an evolution of the Cloud Computing model in the sense of use of micro-services and containers. It has been characterized by:

- a simplified programming model, abstracting the operational concerns.
- a billing model based on the functions execution time instead of the resource allocation.

Serverless Computing

Serverless Computing has emerged as a new paradigm of abstraction, platform and implementation of cloud functions [1,2], presenting an evolution of the Cloud Computing model in the sense of use of micro-services and containers. It has been characterized by:

- a simplified programming model, abstracting the operational concerns.
- a billing model based on the functions execution time instead of the resource allocation.

In this context, it is possible to deploy rapidly stateless functions which respond to events. They can be coordinated to behave as micro-services and to be statefull.

Serverless Computing

In practice:

- It was introduced by AWS when they launched AWS Lambda[3] with the Function as a Service concept.
- After that vendors as Google, Microsoft and IBM followed AWS introducing Google Cloud Functions[4], Microsoft Azure Functions[5] and IBM Cloud Function[6]. The last one turned to Open Source, with the name of OpenWhisk[7], and has been used in this work.

Micro-services and Containers

A micro-service is an architectural pattern of development of applications. It consists on splitting an application (or a service) in pieces (micro services) grouped by functionalities, and it provides a lot of benefits, such as, fast developtment, easy scaling, update, readability, and etc.

- This architecture contrasts with standard monolithic architectured applications, which has all code together in an unique or few files.

Micro-services and Containers

A micro-service is an architectural pattern of development of applications. It consists on splitting an application (or a service) in pieces (micro services) grouped by functionalities, and it provides a lot of benefits, such as, fast developtment, easy scaling, update, readability, and etc.

- This architecture contrasts with standard monolithic architectured applications, which has all code together in an unique or few files.

A container is a package of code and dependencies that allow applications to be executed easily. They are composed by layers, which contains separeted parts of the dependencies or the code.

Micro-services and Containers

A simple example of a monolithic architectural designed application can be:

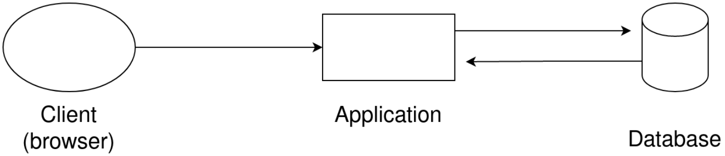


Figure 1: Example of a monolithic architectural designed application.

Micro-services and Containers

In a micro-service architectural approach this application can turn to:

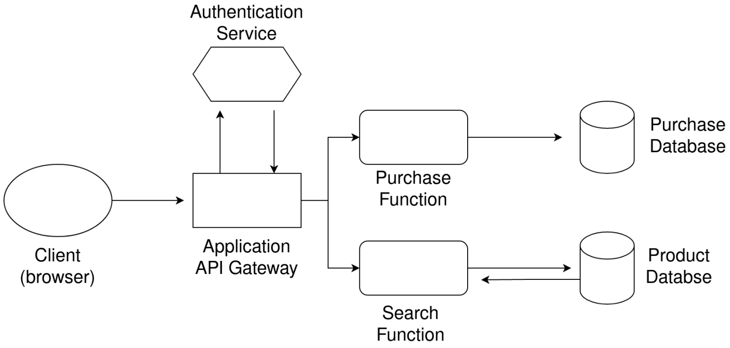


Figure 2: A micro-services approach.

Cloud,Edge and Serverless Platforms

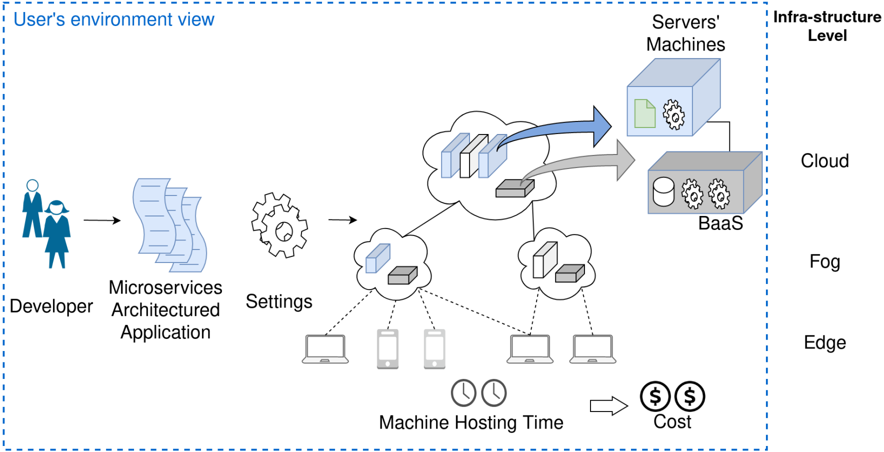


Figure 3: Cloud/ Edge Platform illustration.

Cloud,Edge and Serverless Platforms

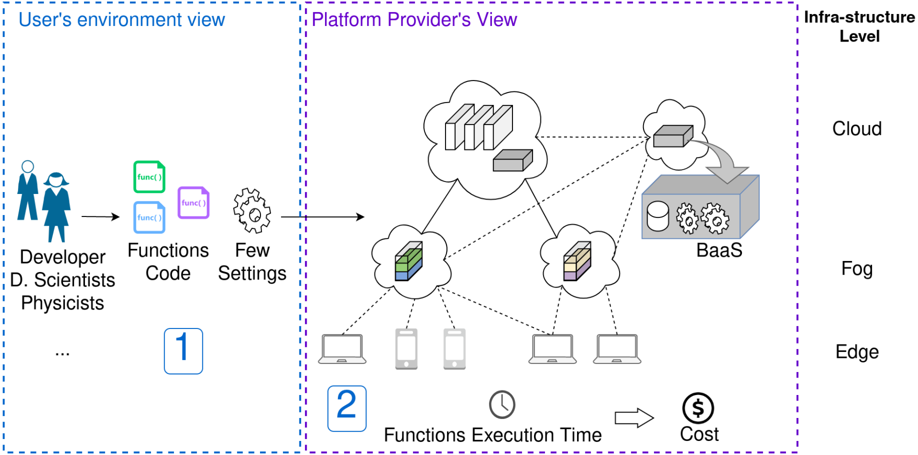


Figure 4: Serverless Platform Illustration.

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

Action	Actor

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

	Action	Actor
1	Send the function and dataset credentials (if needed) to the platform,	User

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

	Action	Actor
1	Send the function and dataset credentials (if needed) to the platform,	User
2	Create a package (container) with the dependencies needed,	Platform
3	Inject the function inside the container,	Platform
4	Allocate the container to a machine to be executed,	Platform
5	Execute the container,	Platform

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

	Action	Actor
1	Send the function and dataset credentials (if needed) to the platform,	User
2	Create a package (container) with the dependencies needed,	Platform
3	Inject the function inside the container,	Platform
4	Allocate the container to a machine to be executed,	Platform
5	Execute the container,	Platform
6	Download the inputs though the credentials passed to external databases,	Function
7	Upload the inputs though the credentials passed to external databases,	Function

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

	Action	Actor
1	Send the function and dataset credentials (if needed) to the platform,	User
2	Create a package (container) with the dependencies needed,	Platform
3	Inject the function inside the container,	Platform
4	Allocate the container to a machine to be executed,	Platform
5	Execute the container,	Platform
6	Download the inputs though the credentials passed to external databases,	Function
7	Upload the inputs though the credentials passed to external databases,	Function
8	Receive a completion signal.	User

Serverless Computing in Practice

What happens when a function is submitted to such platforms?

	Action	Actor
1	Send the function and dataset credentials (if needed) to the platform,	User
2	Create a package (container) with the dependencies needed,	Platform
3	Inject the function inside the container,	Platform
4	Allocate the container to a machine to be executed,	Platform
5	Execute the container,	Platform
6	Download the inputs though the credentials passed to external databases,	Function
7	Upload the inputs though the credentials passed to external databases,	Function
8	Receive a completion signal.	User
9	The container is destroyed,	Platform
10	The resources are free,	Platform

Scheduling of Functions

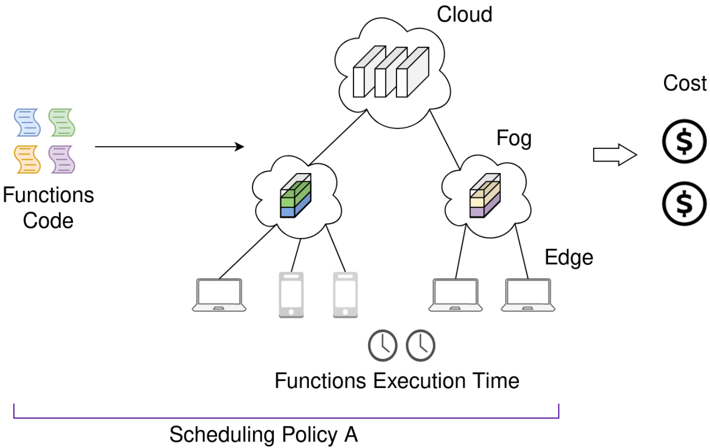


Figure 5: Scheduling of Functions on Serverless Platforms - Policy A.

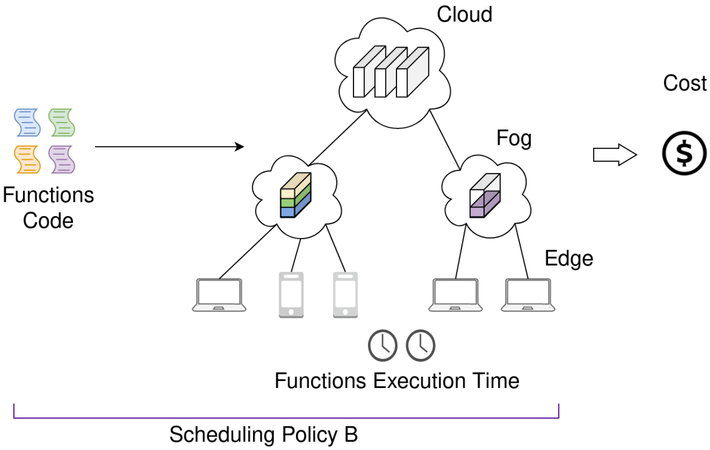


Figure 6: Scheduling of Functions on Serverless Platforms - Policy B.

Benchmarks

We adapted Servelss Functions from FunctionBench[8, 9, 10]. Some examples are: linpack, image processing, video processing, matmul, model training, and others.

Benchmarks

We adapted Servelss Functions from FunctionBench[8, 9, 10]. Some examples are: linpack, image processing, video processing, matmul, model training, and others. We adapted such functions to be executed in the open source Serveless Platform OpenWhisk. We deployed OpenWhisk on GRID5000[11] and executed our functions there.

Benchmarks

We adapted Servelss Functions from FunctionBench[8, 9, 10]. Some examples are: linpack, image processing, video processing, matmul, model training, and others. We adapted such functions to be executed in the open source Serveless Platform OpenWhisk. We deployed OpenWhisk on GRID5000[11] and executed our functions there. Then, we extracted:

- Resources usage:
  - cpu,
  - memory,
  - bandwidth.
- Different measurements:
  - functions execution time,
  - donwload/ upload time of inputs/ outputs,

Benchmarks

Through our benchmarks we identified that the creation of containers can take considerable time, depending on the functions depenendecies. But, due to the containers composition - layers - they can share common data.

Benchmarks

Through our benchmarks we identified that the creation of containers can take considerable time, depending on the functions depenendecies. But, due to the containers composition - layers - they can share common data. For instance, we can imagine the container layers of video processing, image processing, and linpack as the following:

Benchmarks

Through our benchmarks we identified that the creation of containers can take considerable time, depending on the functions depenendecies. But, due to the containers composition - layers - they can share common data. For instance, we can imagine the container layers of video processing, image processing, and linpack as the following:

video processing	image processing	linpack
cv2	Image, ImageFilter	
uuid	uuid	
boto3	boto3	numpy
python3	python3	python3
OpenWhisk	OpenWhisk	OpenWhisk
OS Distribution	OS Distribution	OS Distribution

Benchmarks

Through our benchmarks we identified that the creation of containers can take considerable time, depending on the functions depenendecies. But, due to the containers composition - layers - they can share common data. For instance, we can imagine the container layers of video processing, image processing, and linpack as the following:

video processing	image processing	linpack
cv2	Image, ImageFilter	
uuid	uuid	
boto3	boto3	numpy
python3	python3	python3
OpenWhisk	OpenWhisk	OpenWhisk
OS Distribution	OS Distribution	OS Distribution

In practice, we can estimate a matching of layers between video processing and image processing of 80%. While, we have about 10% among them and linpack.

A Simulated Environment

To study different scheduling policies, we have used the simulator Batsim/ SimGrid[12], that allows the usage of different platforms, workloads and scheduler policies.

Then we have:

- Targeted a platform to describe the Cluster GRID5000,
- Modeled workloads using the results of our Benchmarks,
- Created different scheduling policies. As a first approach: cache locality.

Experimental Results

Priorities: 1. Resource availability.

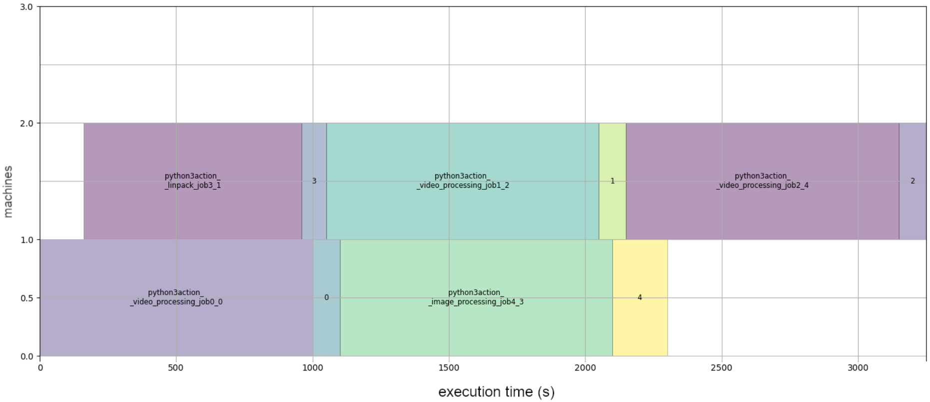


Figure 7: Always Downloading Container Scheduling Policy (Worst Case).

Priorities: 1. Resource availability, 2. Cache locality.

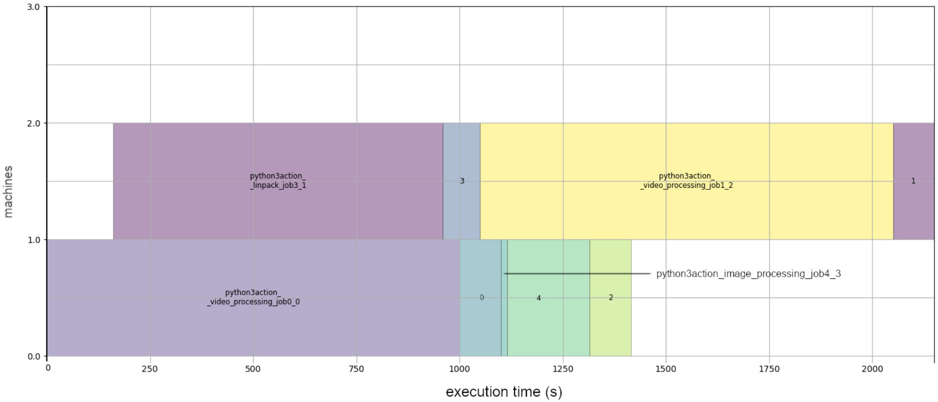


Figure 8: Cache Locality with Container Layers Scheduling Policy.

Experimental Results

Priorities: 1. Cache locality, 2. Resource availability.

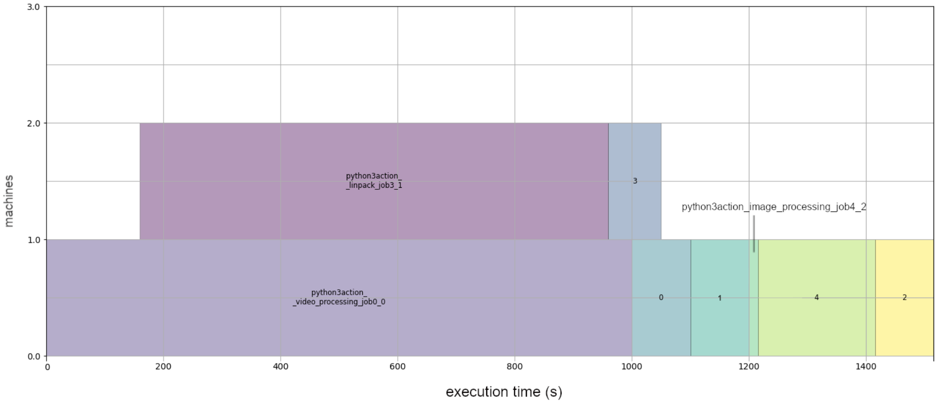


Figure 9: Hard Cache Locality with Container Layers Scheduling Policy.

Proposed Scheduling Model: Variables

- T : Maximum Completion Time
- C : Maximum Cost of Solution
- M : Nb of Machine
- N : Nb of Jobs
- K : Nb of Environment
- c [M][N] : Cost of a Job on a Machine
- p [M][N] : Time of a Job on a Machine
- d [M][N] : Cost of an Env on a Machine
- b [M][N] : Time of an Env on a Machine
- env [N] : Env needed by a Job

Outputs:

- x [M][N] : Job Placement on Machines
- e [M][N] : Env Placement on Machines

Proposed Scheduling Model: Constraints

$$\textit{Objective function} : \min\{ \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij} * x_{ij} + \sum_{i=1}^{M}\sum_{k=1}^{K} d_{ik} * e_{ik}\}$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij} * x_{ij} + \sum_{i=1}^{M}\sum_{k=1}^{K} d_{ik} * e_{ik} \leq C \qquad (1)$$

$$\sum_{i=1}^{M} x_{ij} = 1 \qquad \forall j \leq N \qquad (2)$$

$$\sum_{j=1}^{N} p_{ij} * x_{ij} + \sum_{k=1}^{K} b_{ik} * e_{ik} \leq T \qquad \forall i \leq M \qquad (3)$$

$$x_{ij} \geq 0 \qquad \forall i \leq M, \forall j \leq N \qquad (4)$$

$$x_{ij} = 0 \text{ if } p_{ij} + b_{i,env[j]} > T \qquad \forall i \leq M, \forall j \leq N \qquad (5)$$

$$x_{ij} \leq e_{i,env[j]} \qquad \forall i \leq M, \forall j \leq N \qquad (6)$$

$$e_{ik} \leq 1 \qquad \forall i \leq M, \forall k \leq K \qquad (7)$$

$$e_{ik} \geq 0 \qquad \forall i \leq M, \forall k \leq K \qquad (8)$$

Proposed Scheduling Model: Example

Let's consider the following input:

- $m = 3$
- $n = m(m-1) + 1 = 7$
- $p_{i1} = m, i = 1, ..., m$
- $p_{ij} = 1, i = 1, ..., m, j = 2, ..., n$
- $c_{ij} = 0, i = 1, ..., m, j = 1, ..., n$
- $C = 0$
- $T = 3$

It will provide the following solution:

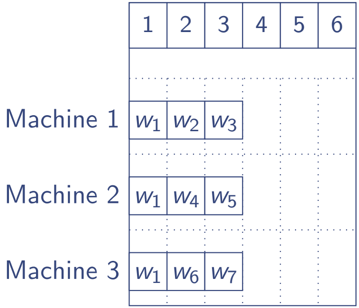
$$X = \begin{bmatrix} 1/3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Please, notice that the solution provides fractional allocations.

Proposed Scheduling Model: Fractional and Integer Solution

We used a Bipartite Graph to convert the fractional into an integer solution. For example:

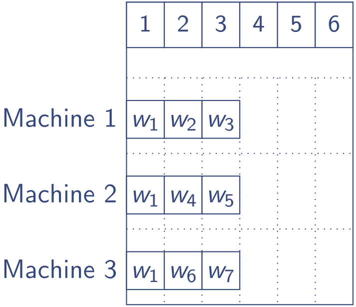
$$X = \begin{bmatrix} 1/3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



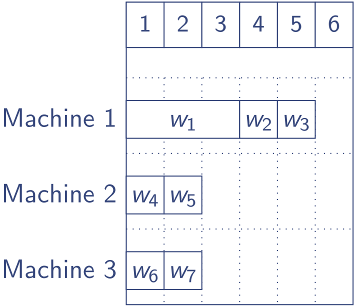
Proposed Scheduling Model: Fractional and Integer Solution

We used a Bipartite Graph to convert the fractional into an integer solution. For example:

$$X = \begin{bmatrix} 1/3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



$$X' = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



Future Work

Papers proposition:

- To present an allocation policy to schedule heterogenous functions to heterogeneous machines, based on the allocation policy of Shmoys and Tardos[13]. In this scenario, we want to reduce the cost of bandwidth of downloading containers and input data, as well as uploading output data, at the same time as reducing the makespan of the platform.
- To present how to model Serverless Platforms in a simulated environment to study scheduling policies.

Thank you for your attention!
Any question?

Scheduling Function on Serverless Platforms

A. A. Da Silva, Y. Georgiou, M. Mercier, G. Mounié, D. Trystram

Université Grenoble Alpes, Ryax Technologies

Contact email: anderson-andrei.da-silva@inria.fr

https://gitlab.com/andersonandrei/scheduling-functions-on-serverless-computing

UGA Université Grenoble Alpes · ryax technologies · MIAI Grenoble Alpes · PHYSICS

References

[1] D. Barcelona-Pons, M. Sánchez-Artigas, G. Parıs, P. Sutra, and P. Garcia López, "On the FaaS Track: Building Stateful Distributed Applications with Serverless Architectures," in Proceedings of the 20th International Middleware Conference. Davis CA USA: ACM, Dec. 2019, pp. 41–54. [Online]. Available: https://dl.acm.org/doi/10.1145/3361525.3361535

[2] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, "Serverless Computing: Current Trends and Open Problems," arXiv:1706.03178[cs], Jun. 2017, arXiv: 1706.03178. [Online]. Available:http://arxiv.org/abs/1706.03178

[3] "Aws lambda." [Online]. Available: https://aws.amazon.com/lambda/

[4] "Cloud functions." [Online]. Available: https://cloud.google.com/functions

[5] "Azure functions." [Online]. Available: https://azure.microsoft.com/en-us/services/functions

[6] "Ibm cloud functions." [Online]. Available: https://www.ibm.com/cloud/functions

References

[7] "OpenWhisk." [Online]. Available: https://openwhisk.apache.org/

[8] Jeongchul Kim and Kyungyong Lee, 'Function Bench : A Suite of Workloads for Serverless Cloud Function Service', IEEE International Conference on Cloud Computing 2019, 07/2019

[9] Jeongchul Kim and Kyungyong Lee, 'Practical Cloud Workloads for Serverless FaaS, ACM Symposium on Cloud Computing 2019, 11/2019

[10] "Function Bench". Available: https://github.com/kmu-bigdata/serverless-faas-workbench

[11] "Grid5000". Available: https://www.grid5000.fr

[12] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," Journal of Parallel and Distributed Computing, vol. 74, no. 10, pp. 2899–2917, Jun. 2014. [Online]. Available: http://hal.inria.fr/hal-01017319

[13] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. Math. Program., 62(1–3):461–474, February 1993.

Scheduling of Functions on Serverless Platforms

A. A. Da Silva, Y. Georgiou, M. Mercier, G. Mounié, D. Trystram,
Université Grenoble Alpes, Ryax Technologies
3 of December of 2021