Laboratoire de l'Informatique du Parallélisme
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668
École Normale Supérieure de Lyon, site Monod, 46 Allée d'Italie, Lyon.

Report on the thesis of Paweł Żuk

Resource allocation methods for serverless computing platforms

Severless computing, also known as Function as a Service (or FaaS) has appeared as an interesting alternative to classical Cloud computing paradigm to ease the use of distributed resources. Programmers do not need to worry anymore about scaling the platform to the incoming load and about the deployment of new nodes; they just declare stateless functions that will be called when new requests are issued: the system takes care of providing a dedicated environnement for each function call and of scaling the platform as needed. This also leaves more optimization opportunities to the system, in particular when mapping and scheduling functions. Optimizing the operation of such systems by taking into account specificities in the FaaS workload is the subject of the thesis of Paweł Żuk, which proposes both theoretical contributions and practical implementations of some of the proposed strategies.

The thesis starts with a clear **Introduction** (chapter 1, 10 pages) of the subject. It presents the specificities of the FaaS abstraction, the literature covering various works around FaaS and related mapping and scheduling problems, and the outline of the thesis. It is followed by a short chapter on **Preliminaries** (chapter 2, 4 pages) that presents common definitions as well as the architecture of OpenWhisk, the FaaS system chosen for the practical contributions.

Chapter 3 (25 pages) explains the first contribution of the thesis on scheduling for functions with dependencies. FaaS allows to compose functions: after a specific event, such as an HTTP request, a function is called, followed by another one, etc. Paweł Żuk notices that for now, only chains of functions are common, but anticipates the use of more complex patterns, forming general Directed Acyclic Graphs. He accurately notices that knowing in advance the structure of composite functions allows to optimise the scheduling of such functions. This chapter presents a model of the problem, taking into account the limited memory of the machines and the size of the function environnements. He

then proposes a number of scheduling heuristics, starting from the classical strategies and incorporating interesting ideas in order to well reuse existing environnements, to avoid the cost of deploying new environments. These heuristics are tested through a wide simulation campaign, using a simulator written by Paweł Żuk, both on synthetic data and on data from an existing trace of a FaaS system. The simulator is first validated by comparing its results to a real execution, which is remarkable. The simulations show that some of the heuristics proposed in the chapter lead to a large improvement in response time compared to existing strategies, especially the ones taking into account the knowledge of composition and thus some of the future function calls. The chapter concludes on the ability to benefit from information on future load in case of composite functions, and on how the proposed strategies could be implemented in real FaaS systems.

In **Chapter 4**, Paweł Żuk concentrates on the allocation of applications to machines to cope with the incoming load and under memory constraints: as each application is in charge of many incoming requests, one may distribute its execution over several machines to cope with a larger load. However, this requires to deploy the applications and thus takes up memory on several machines. Paweł Żuk proposes a simple theoretical model and outlines two problems: (i) minimizing the maximum load on a given set of machines, under memory constraint, and (ii) minimizing the number of used machines, under both memory and load constraint. He carefully studies the complexity of each problem, by proposing optimal polynomial algorithms for the homogeneous case and by proving that adding heterogeneity on either the memory size or the load of applications renders the problem NP-complete. For the second problem, he also shows that using several instances for each application potentially reduces the number of applications by a factor 2. Then, the chapter proposes several low-cost heuristics based on existing strategies for bin packing, which are adapted to take into account specificities of the present problem, such as the use of several instances for an application. These heuristics are compared through simulation on instances of FaaS functions coming from an existing dataset. The simulations show that using several instances allows to fulfill the load of each application while reducing the number of required machines.

Chapter 5 (22 pages) focuses on the problem of scheduling applications on a single machine (assuming the allocation of applications to machine has already been done earlier). The objective is to improve the response time in case of a burst of requests, as platforms are currently largely over-provisioned in order to cope with such short bursts: deploying new environments on available machines is usually too long. Paweł Żuk proposes a classical model of the problem, based on non-clairvoyant online scheduling. The novelty comes from the fact that all requests target a limited number of functions. This allows

to predict the running time of a request based on the previous execution of the same function. Paweł Żuk thus proposes to adapt classical clairvoyant scheduling heuristics, such as Shortest Processing Time (SPT) or Shortest Remaining Processing Time (SRPT) in the case of preemptive scheduling, by using the expected (remaining) processing time. He also designs two additional heuristics focusing on fairness among functions in addition to performance. The proposed strategies are compared to classical ones (FIFO for non-preemptive scheduling, Round-Robin for preemptive scheduling) in a large simulation campaign using trace of an actual FaaS system. Paweł Żuk carefully analyse the results and show that (i) simple running-time predictions are sufficient to help with scheduling and (ii) prediction-aware scheduling heuristics largely outperform classical strategies.

The good behavior of the strategies proposed in Chapter 5 motivates their implementation in a real FaaS system, which is the goal of **Chapter 6** (20 pages). Paweł Żuk opts for an open-source FaaS, OpenWhisk, and adapts it in order to include some of the new heuristics of the previous chapter, namely Shortest Expected Processing Time (SEPT) and Fair Choice (FC), as well as two other heuristics based on a similar intuition but with an additional guarantee that no starvation can happen (all requests are eventually processed). This requires a significant implementation effort in OpenWhisk, among other in order to include the use of prediction for scheduling and to avoid preemption as much as possible (as the chosen scheduling strategies are designed for the non-preemptive case). Experiments are conducted using the SeBS benchmark of FaaS systems. A short burst is obtained by generating many requests in one minute. The obtained results are interesting, well presented and analysed. They first show that SEPT and FC are able to largely reduce the flow-time and stretch in all scenarios. Heuristics guaranteed against starvation are a bit less efficient, but still better than system-oriented strategy (FIFO). FC gives good performance even among very different functions. Finally, in a multi-node execution, the experiments show that the proposed strategies are able to reduce the average response time, which would allow to decrease the size of the platform without loosing on the quality of service.

The thesis ends on a **Conclusion** (3 pages) that summarized the problem that was adressed as well as the contributions.

In conclusion, this thesis concentrates on a new and timely scheduling problem and makes very relevant contributions. Paweł Żuk is able to take into account specificities of the problem (such as semi-flexible bin-packing, possible use of running time predictions) to design innovative and interesting mapping and scheduling heuristics. The description of the proposed strategies is clear and guided with well-described intuitions. Theoretical contri-

butions also include a complete complexity study of the semi-flexible mapping problem, with NP-completeness results and optimal polynomial algorithms for simpler cases. It has to be noted that the thesis also includes a large "practical" contribution: many simulation results, mostly based on actual traces or datasets to validate the proposed strategies, the validation of one of the simulators by a comparison with real executions, and last but not least, the implementation of some scheduling heuristics in a real systems. Given this large implementation effort, it would have been interesting for the community to make the code publicly available, which is currently not the case and limits the reproducibility of the results, even if the usage of traces and the generation of instances is well documented in the thesis.

In all his thesis, Paweł Żuk has demonstrated its ability to cover all parts of scheduling research, from theory to practice. Therefore, I firmly believe that Paweł Żuk meets all the requirements to defend his thesis.

Loris Marchal,
Senior researcher, CNRS.
Loris.Marchal@ens-lyon.fr