Summary

When Amazon announced EC2, its Elastic Computing Cloud, the era of cloud computing started to offer a different computing paradigm. EC2 and other commercial cloud providers (Rackspace, GoGrid, IBM) offer compute resources with defined quality of service (CPU type and clock speed, size of main memory, etc.) These computers can be rented at certain prices per given time intervals, typically per hour. While each of these commercial offerings comes with clear machine specifications, users still need guidance when choosing among them, simply because most of the time these specifications say little about the performance of a given application.

Bags of tasks are an important intrinsically elastic application class. In this work, we introduce BaTS, our budget-constrained scheduler for bag-of-tasks applications. BaTS requires no a-priori information about task execution times. It uses statistical methods to execute samples of tasks on all cloud platforms that are available to a user. BaTS monitors the progress of the tasks and dynamically reconfigures the set of machines, based on the expected budget consumption and completion time.

We first propose the general architecture of the BaTS scheduler. We also describe the core scheduling algorithm. Our real-life experiments with a BaTS prototype show that a cost-aware scheduler may be efficiently and effectively implemented, and that the real-time reconfigurations manage to maintain the cost of execution within a specified budget.

Next, we introduce a user guidance component, the estimator, which takes advantage of mathematical tools to present the user with feasible schedules for the execution of the bag after inspection of only a small number of tasks. Our results show that BaTS is able to guide a user through the jungle of cloud offerings and/or schedule within a user-defined budget (if such a schedule is possible at all.)

We then proceed to model the general architecture of BaTS as a stochastic mathematical construct which we use to evaluate the strengths and limitations of
our proposed scheduler. For this purpose, we implement a simulator that we use to provide an extensive set of evaluations, involving a wide set of schedules and real-life, traces-inspired workloads. Our simulation experiments show that the prediction mechanisms implemented in BaTS may effectively deal with different types of workloads.

We also introduce a number of stochastic tail-phase optimizations that we evaluate on the mathematical construct introduced earlier. Our simulation experiments show that replication strategies outperform migration strategies and that imperfect information coupled with random selection deliver the closest performance to perfect information.

Finally, we evaluate BaTS in a real-world environment, by actual execution of a parameter-sweep application on Amazon EC2 instances of different types. Our experiments show that BaTS is capable of successfully selecting the most efficient resources for the workload at hand and consequently executing the application within the user-specified requirements.

Our results also suggest possible directions for future work. One such direction is introducing an adaptive profitability criterion with respect to the phase of the computation. Here the faster-to-cheaper ratio used for the high-throughput phase should be replaced by a least-variance ratio during the tail-phase. Also, an interesting theoretical question is how much our proposed CLT model accounts for the execution performance variability of machines pertaining to the same type. Another interesting research direction stems from the intrinsic fault-tolerance of the BaTS scheduler, given that one of the task assumptions is that tasks may be re-executed. This would enable checkpointing the application to stable storage, leading to further research on whether the theoretical model put forward in this work could already deal with the intricacies of storage and file transfer cost models.