Workflow Scheduling on Power Constrained VMs

David Shepherd, Ilia Pietri, Rizos Sakellariou
School of Computer Science, University of Manchester, UK

***Abstract*—With energy consumption being an issue of growing concern in large-scale cloud data centers, providers may wish to impose restrictions on the power usage of the hosts. This raises the challenge of operating cloud resources under power limits which may vary over time. Motivated by such a constraint, this paper considers the problem of scheduling scientific workflows in an environment where the number of VMs available is limited by a time-varying power cap. A simple scheduling algorithm for such cases is proposed and experimentally evaluated.**

I. Introduction and Problem Description

Cloud computing platforms offer a flexible environment to provide user resources on demand. However, the scale of existing cloud data centers may lead to high energy consumption, making electricity cost a significant element in their operation. To manage such costs cloud providers may set power budgets [1], which in turn may lead to power caps for specific users or applications. This implies that the latter will need to limit the number of VMs they use to the extent that they do not exceed the power cap.

This paper focuses on scheduling in scientific workflow applications under a power cap. Scientific workflows [2] consist of inter-related tasks with data dependencies between them. They can be modelled as a Directed Acyclic Graph (DAG) where the nodes represent the computational tasks and the edges represent the data dependencies between them. We assume that information on task runtime and data transfer is known and users are interested in maximizing the number of VMs provisioned to take advantage of the inherent parallelism in the workflow structure. As the power budget limits the number of VMs that can be provisioned over time the problem that arises is how to schedule tasks onto VMs while taking into account fluctuations in the number of available VMs. This paper suggests a scheduling algorithm to address this problem.

In contrast to related work making elaborate assessments of the power/performance correlation, such as [3], we use a simple power model and focus instead on the scheduling challenge. We assume that a single VM type is available, and that the VMs have a fixed power consumption (*i.e.* no dynamic voltage or frequency scaling). We also assume that information about the power cap is available ahead of time for use in scheduling decisions. Thus the number of VMs available at any given time is entirely determined by the power cap: we allow the use of as many VMs as possible while remaining below the cap. So the problem is reduced to scheduling jobs on a time-varying number of VMs.

II. The Algorithm

Our algorithm is based on the well known Heterogeneous Earliest Finish Time (HEFT) scheduling algorithm [4]. HEFT is a static, heuristic scheduling algorithm which operates in two steps. First the tasks are sorted by their "upward rank", a combination of factors based on the job's mean computation time (over the available VMs), mean communication time, and the upward rank of its successors in the workflow DAG. Second the tasks are scheduled, in the above order, to the VM that results in the earliest finish time for that job. Due to space constraints we refer to the paper [4] for details of the HEFT algorithm.

In standard HEFT initially all VMs are available at all times, and portions of time on the VMs are occupied by jobs as the algorithm progresses. In our power-capped modification we instead begin with some portions of VM time occupied by dummy jobs at times when there is insufficient power to run that VM. Following this initial modification of the available slots, the algorithm proceeds identically to standard HEFT. The key observation is that this modification does not interfere with the running of HEFT except to prevent jobs from being scheduled to unavailable VMs. The pseudocode for this algorithm is given in Algorithm 1.

Note that, despite the name, our algorithm currently assumes homogeneous VMs. However it could be easily extended for heterogeneous VMs as follows: no VMs are spawned initially, but HEFT is allowed to schedule jobs to free slots on existing VMs (as normal) *or* to spawn any type of new VM for which there is sufficient power. This could be thought of as reusing the HEFT scheduling heuristics to decide which VM types to spawn.

Algorithm 1 Power-capped HEFT scheduling algorithm.

1: Determine the number of VMs available over time within the power cap.
2: Schedule dummy jobs at times when VMs are unavailable.
3: Run HEFT with the modified initial schedule.

III. Experimental Evaluation

The proposed algorithm was implemented within the cloud workflow simulator in order to evaluate its performance on large workflows [5], [6]. Performance is tested on synthetic workflow data, corresponding to real scientific applications, generated using WorkflowGenerator [7], [8]. In our implementation communication costs are not included for simplicity and due to a lack of support in cloud workflow simulator. However we experiment with CPU-intensive workflows [9] so that the impact of this simplification is minimal.

The performance of the power-capped HEFT algorithm was compared to a simple dynamic scheduling algorithm with no

978-0-7695-5697-0/15 $31.00 © 2015 IEEE
DOI 10.1109/UCC.2015.74

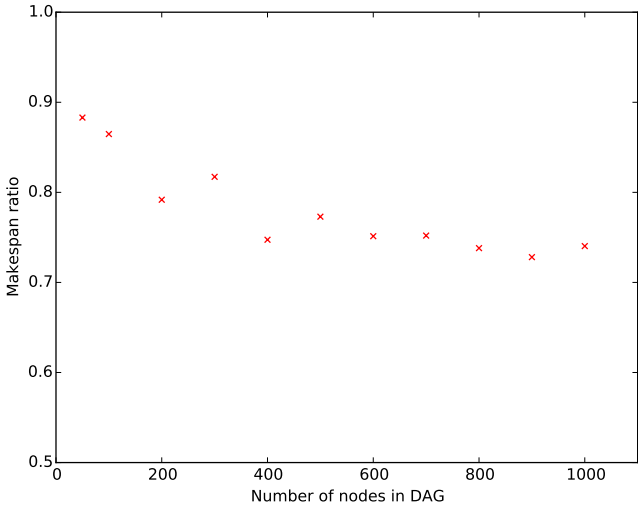

Fig. 1: The ratio of the HEFT-based makespan to the FCFS-based makespan against the number of nodes in the workflow DAG for SIPHT workflows [10].

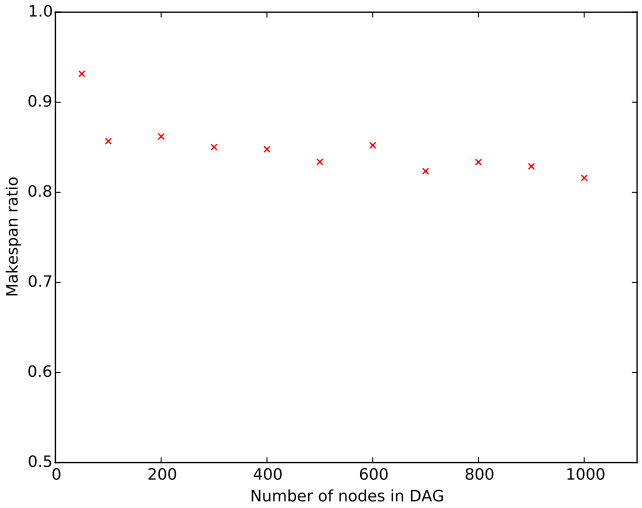

Fig. 2: The ratio of the HEFT-based makespan to the FCFS-based makespan against the number of nodes in the workflow DAG for GENOME workflows [11].

prior knowledge of the power cap. Tasks are scheduled using a first-come-first-served (FCFS) approach: tasks are scheduled to the first available VM in the order that they are ready (*i.e.* when all predecessors in the workflow DAG are complete). The power cap is enforced by limiting the number of VMs; when the power cap is reduced VMs are killed as needed. Note that standard HEFT could not be used as a baseline for comparison as it assumes a predefined and fixed list of VMs.

In the experiments the power cap $P(t)$ is chosen to be a simple piecewise-constant function of time. It begins with power P_0, drops to power $P_0/2$ at time $\tau/3$, then jumps back to power P_0 at time $2\tau/3$. It remains to choose reasonable values for τ and P_0 for a general workflow. We choose τ to be the computation time for the critical path of the workflow DAG, *i.e.* a lower bound for the makespan. We then choose P_0 to be $P_0 = \frac{\alpha C}{\tau}$, where α is the instructions executed per unit of power consumption for the VMs and C is the total number of instructions needed for all tasks in the workflow. This corresponds to the constant power that would be needed to finish all tasks by time τ if the constraints imposed by the workflow DAG and the granularity of the VMs could be ignored. For the experiments $\alpha = 20,000$ was chosen arbitrarily.

We plot the ratio of the HEFT-based makespan to the FCFS-based makespan in order to compare the performance of the algorithms over a wide range of workflow sizes. Figure 1 shows the ratio of the two makespans against the number of nodes in the DAG for SIPHT workflows [10]. Figure 2 shows the same plot for GENOME workflows [11]. We see that in both cases the HEFT-based scheduling algorithm outperforms the naive approach (FCFS) by a margin between roughly $0.9\times$ and $0.7\times$. We also note that HEFT-based scheduling performs better as the size of the workflow DAG increases.

IV. Conclusion

We have introduced an algorithm for resource provisioning and workflow scheduling under time-varying power constraints. Our algorithm offers a reasonable improvement in the makespan over a naive dynamic scheduling approach.

References

[1] Y. Zhang, Y. Wang, and X. Wang, "Capping the electricity cost of cloud-scale data centers with impacts on power markets," in *Proceedings of the 20th international symposium on High performance distributed computing*. ACM, 2011, pp. 271–272.

[2] I. J. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer, 2007.

[3] P. E. Bailey, D. K. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. R. De Supinski, "Adaptive configuration selection for power-constrained heterogeneous systems," in *43rd International Conference on Parallel Processing*. IEEE, 2014, pp. 371–380.

[4] H. Topcuoglu and S. Hariri, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[5] Cloud Workflow Simulator, Available:https://github.com/malawski/cloudworkflowsimulator.

[6] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IAAS clouds," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[7] Workflow Generator, Available:https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

[8] R. F. da Silva, W. Chen, G. Juve, K. Vahi, and E. Deelman, "Community Resources for Enabling Research in Distributed Scientific Workflows," in *10th International Conference on e-Science*. IEEE, Oct. 2014, pp. 177–184.

[9] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Third Workshop on Workflows in Support of Large-Scale Science*. IEEE, Nov. 2008, pp. 1–10.

[10] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-Throughput, Kingdom-Wide Prediction and Annotation of Bacterial Non-Coding RNAs," *PLoS ONE*, vol. 3, no. 9, p. e3197, Sep. 2008.

[11] USC Epigenome Center, Available:http://epigenome.usc.edu.