Nornir: A Power-Aware Runtime Support for Parallel Applications

Daniele De Sensi

1 The Power Efficiency Challenge

Power consumption is becoming a key factor in designing applications and computing systems.

On mobile devices, having an efficient power management translates directly into how long the battery lasts while using the device. The highest ranked metric in mobile devices evaluation is usually the battery life and improving it will lead to even increased functionality and richer applications.

Power management is also relevant for tethered devices (connected to a power supply). For example, in current datacenters, the energy cost is quickly going to overcome the cost of the physical system itself [2]. Increased power consumption leads to more complex power supplies that also add costs. More importantly, energy consumption in Information Technology (IT) industry has a considerable impact on the environment, since during 2010 the CO_2 emissions of U.S. data centers were on par with those of an entire country like Argentina or Netherlands[1] .

Most likely, the importance of power management techniques is going to increase in the future. On mobile devices, the increase in performance requirements due to new functionality will overcome the advances in battery capacity. Concerning datacenters, in 2010, the energy consumed in U.S. by data centers reached the 3% of the overall energy production [14], and this power demand is estimated rapidly growing 10% – 12% a year [15].

2 Facing the Challenge

This thesis focuses on runtime, software level solutions on multicore architectures. Such approaches often operate on *control knobs* to reduce or optimize the number of resources allocated to an application, thus decreasing its power consumption. This implies reducing the number of cores used by the application, scaling down their frequency or acting on other software or hardware mechanisms. A specific combination of the knobs values is often referred as a "*configuration*". However, each configuration is characterized by a different performance and power consumption and, in general, those having a lower power consumption are also characterized by lower performances. Accordingly, the best configuration depends on the application's user requirements. In the following, we describe two scenarios addressed in this thesis:

[1]http://www.hpcwire.com/2010/06/15/the_coming_c_change_in_datacenters/

Power Capping To improve data centers efficiency, over-subscription of data center power is recently becoming a common practice [13]. Data center power demand is intentionally allowed to exceed the power supply, under the assumption that correlated spikes in servers' power consumption are infrequent. However, this exposes data centers to the risk of power outages, caused by unpredictable power spikes (e.g. due to an increase in the power consumption of more servers at the same time). Such an event would have catastrophic effects, since it would lead to degraded user experience or to a service outage. For these reasons, to achieve power safety and to avoid having under-utilized power provisioning, *power capping* techniques have been recently proposed [17]. These techniques monitor data center power consumption and, when it is close to the available capacity, they force some server to reduce its power consumption.

Performance Requirements In some cases, applications don't need to run at maximum performance. Nevertheless, the user would still like to have some guarantees on minimum application's performance. This scenario provides power saving possibilities, since it would be possible to reduce the number of resources used by the application, thus decreasing application's performance (still keeping it above the requirement specified by the user) and reducing its power consumption [16]. For example, suppose to have a data backup application that, each night, collects and compresses the data produced by the employees of a company during the day and sends them to a remote storage. We could require this application to complete this process in a 12 hours time frame. The application could probably finish the process in a shorter time frame. However, doing so would not necessarily have an advantage from the user perspective and by slowing it down we could reduce its power consumption while at the same time be completely functional.

3 Thesis Contributions

In this thesis, we propose different algorithms to decide the best configuration to apply to a parallel application according to the requirement expressed by its user. These algorithms differ in the type of knobs they operate on, on the type of requirements they can satisfy, on their accuracy and on the overhead they have on application execution. In detail, we designed the following algorithms: **i)** Two heuristics [7, 6, 5], able to provide explicit guarantees while minimizing power consumption; **ii)** Two self-aware algorithms [8, 11], able to autonomously model the performance and power consumption behavior of the application. Differently from most existing algorithms, they only use data collected online, while the application is running, not requiring any previous training phase; **iii)** A concurrency control algorithm [1], i.e. an algorithm that decides which is the best way to accesses data structures shared among the threads in a parallel application. The algorithm autonomously switches between different access modes according to the user's requirements.

All the algorithms have been implemented and tested over all the applications from the PARSEC benchmark. We were able to improve state-of-the-art solutions in terms of accuracy, optimality, and overhead on application execution. To be able to implement and test them with relatively low programming effort we designed Nornir [9], a customizable runtime support, which can be used by application's users to express performance and power consumption requirements on their application and by algorithm designers to implement new strategies or to customize existing ones. By using the set of actuators and the application monitoring infrastructure provided by Nornir, the designer can easily implement new algorithms. Moreover, new monitoring and actuation mechanisms can be easily added. At best of our knowledge, this is the first attempt to build a framework for reconfiguration of parallel applications which allows to plug in different reconfiguration algorithms. The source code of the framework is publicly available[2]. To realize the actuators (and part of the monitoring infrastructure), we have designed, implemented and released as open source[3] Mammut [12], a novel framework for integrated management of system knobs and sensors. Differently from existing tools, it allows to easily integrate information coming from different subsystem, thanks to a high-level abstraction of the underlying architecture.

Concerning the applications used for testing, some of the knobs used by our algorithms require the application to have some specific characteristics. A particularly interesting class of applications which falls in this category is that of *parallel design patterns*-based applications (e.g. parallel loops, map-reduce, thread pools, etc...). Despite the diffusion of parallel programming patterns and despite many applications have been implemented by using such programming abstraction, no extensive and coherent benchmark was available, limiting the validation possibilities for reconfiguration strategies targeted towards such applications. We addressed this issue by creating P^{3}ARSEC[4] [3], a benchmark for parallel pattern based applications. We expressed 12 out of 13 applications from the PARSEC benchmark by using parallel patterns and we implemented them with two different parallel pattern based frameworks. This is a contribution per se since we proved, for the first time at this scale, that parallel patterns ease parallel applications development while not decreasing the performance with respect to a handwritten application. This benchmark allowed us to validate our reconfiguration algorithms on a wide set of different and realistic applications.

Eventually, we performed an extensive comparison of the efficiency of some commonly used knobs [4], to understand which one is better under which situation. Moreover, to use some knobs, some support from the application programmer may be required. We analyzed this aspect in [10].

References

[1] M. Aldinucci, M. Danelutto, D. De Sensi, G. Mencagli, and M. Torquati. Towards power-aware data pipelining on multicores. In *Proc. of HLPP2017.*

[2] `http://danieledesensi.github.io/nornir/`
[3] `http://danieledesensi.github.io/mammut/`
[4] `http://github.com/ParaGroup/p3arsec`

[2] L. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, Dec 2007.

[3] M. Danelutto, T. De Matteis, D. De Sensi, G. Mencagli, and M. Torquati. P^{3}ARSEC: Towards parallel patterns benchmarking. In *Proc. of SAC 2017*.

[4] M. Danelutto, T. De Matteis, D. De Sensi, and M. Torquati. Evaluating concurrency throttling and thread packing on smt multicores. In *Proc. of PDP 2017*.

[5] M. Danelutto, D. De Sensi, and M. Torquati. Energy driven adaptivity in stream parallel computations. In *Proc. of PDP 2015*, pages 103–110.

[6] M. Danelutto, D. De Sensi, and M. Torquati. A power-aware, self-adaptive macro data flow framework. In *Proc. of HLPP2016*.

[7] M. Danelutto, D. De Sensi, and M. Torquati. A power-aware, self-adaptive macro data flow framework. *Parallel Processing Letters*, 27(01), 2017.

[8] D. De Sensi. Predicting performance and power consumption of parallel applications. In *Proc. of PDP 2016*, pages 200 – 207, Feb.

[9] D. De Sensi and T. De Matteis. Nornir: A customisable framework for autonomic and power-aware applications. In *Euro-Par 2017: Parallel Processing Workshops*, 2017. To Appear.

[10] D. De Sensi, P. Kilpatrick, and M. Torquati. State-aware concurrency throttling. In *Proc. of ParCo 2017 conference*, 2017. To Appear.

[11] D. De Sensi, M. Torquati, and M. Danelutto. A reconfiguration algorithm for power-aware parallel applications. *ACM Trans. Archit. Code Optim.*

[12] D. De Sensi, M. Torquati, and M. Danelutto. Mammut: High-level management of system knobs and sensors. *SoftwareX*, 6:150 – 154, 2017.

[13] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News.*

[14] J. G. Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3):034008, July 2008.

[15] P. Kurp. Green computing. *Commun. ACM*, 51(10):11–13, Oct. 2008.

[16] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann. A Probabilistic Graphical Model-based Approach for Minimizing Energy Under Performance Constraints. *ACM SIGARCH Computer Architecture News*, 43(1):267–281, Mar. 2015.

[17] Q. Wu, Q. Deng, L. Ganesh, C. H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, and Y. J. Song. Dynamo: Facebook's data center-wide power management system. In *Proc. of ISCA 2016*.