Power aware reconfigurations of parallel applications

Daniele De Sensi[*,1],
Marco Danelutto[*,1],
Massimo Torquati[*,1]

[*] *Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo, 3, 56127 Pisa, Italy*

ABSTRACT

Current architectures provide many possibilities for the reduction of power consumption of applications, such as reducing the number of used cores or scaling down their frequency. However, the amount of resources allocated to an application is usually static and fixed by the programmer or by the runtime. While there are cases where such a static choice may be appropriate, other scenarios may require to dynamically change the amount of resources during the application execution. Choosing the right amount of resources to use in order to satisfy requirements on performance and/or power consumption is a complex task and testing all the possible configurations is an unfeasible solution since it would require too much time.

We show some solutions to this problem that, by acting on the number of cores used by the application an on the frequency of these cores are able to provide guarantees on maximum power consumption or on a minimum performance level. We then outline the main results achieved by applying these techniques to some real applications.

KEYWORDS: power aware; self adaptive; parallel applications;

1 Introduction

Recently, increasing attention has been paid on finding mechanisms and techniques for implementing energy efficient applications. In fact, the energy provisioning cost is quickly going to overcome the cost of the physical system itself. Moreover, high power consumption rises the temperature of the system, increasing the probability of failure of the physical components and requiring advanced heat management systems, that increase the cost by one dollar for each dollar spent in electricity. On the other hand, energy consumption has a considerable impact on the environment, since during 2010 the CO_2 emissions of US' data centers were on par with those of an entire country like Argentina. Nevertheless, according to [Ran10], the average utilisation of many systems is usually in the range $10\% - 50\%$. This opens many possibilities for energy saving by increasing the average utilisation of these systems. This solution is also supported by manufacturers, which provide architectural mechanisms to control and adapt the amount of used physical resources to the real application

[1]E-mail: {desensi,marcod,torquati}@di.unipi.it

needs, for example by scaling the frequency of the CPUs or by turning off cores, cache or RAM modules when they are not used. Moreover, even when the system is fully utilised, the user may decide to sacrifice some performance in favour of a reduction in power consumption. We present some techniques that can be used to provide guarantees on power consumption or performance, outlining the main achieved results.

2 Possible solutions

The main idea behind this work is to start the application execution in a given configuration and, if requirements in term of utilisation, performance or power consumption are not satisfied, try to predict performance and power consumption in all the possible configurations and pick the optimal one. For example when a minimum level of performance is required, among all the configurations characterised by a performance greater than or equal to that required, we would like to pick the configuration with the lowest power consumption. To perform such prediction, we proposed two different types of mechanisms, the first one is based on a simple heuristic, while the other one exploits some basic machine learning techniques.

2.1 Heuristics

A first possible solution [DST] consists in assuming that the amount of elements μ that can be processed by the application per unit of time proportionally decreases when the number of cores n used by the application or their frequency f increases. Accordingly, if $\overline{n}$ is the current number of cores used by the application and $\overline{f}$ is the current frequency of the cores, we have, for any n, f: $\mu(n, f) = \mu(\overline{n}, \overline{f})\frac{n \times f}{\overline{n} \times \overline{f}}$. Concerning the power consumption, it is proportional to $n \times f \times v^2$, where v is the operating voltage of the cores. Since the voltage depends from the frequency, we can easily estimate the power consumption for all the possible configurations. However, this is not an exact value but just a proportional estimation. Consequently, we can use it to determine if a configuration is more power consuming than another, but this value cannot be used to provide explicit guarantees on the power consumption.

We used such solution over a network monitoring application, showing that we are able to adapt the amount of used resources, obtaining a power consumption proportional to the actual rate of data received by the application (Fig. 1), with a maximum overhead of $\sim 4\%$.

2.2 Machine learning

Another possibility [DS] consists in collecting information about different configurations and use such data to train a machine learning model. This would allow a more precise estimation of performance and power consumption. To perform this task, we used a *multiple linear regression* approach. We model the relationship between two or more independent variables (called *predictors*) and a dependent variable (called *response*) by fitting a linear equation to observed data. In our case, the predictors are the number of cores used by the application and their frequency, while the responses are the execution time or the power consumption. To apply this technique, we must express performance and power consumption as a linear combination of number of cores and frequency. As far as performance is concerned the performance we extended the Amdhal's law, considering also the impact of

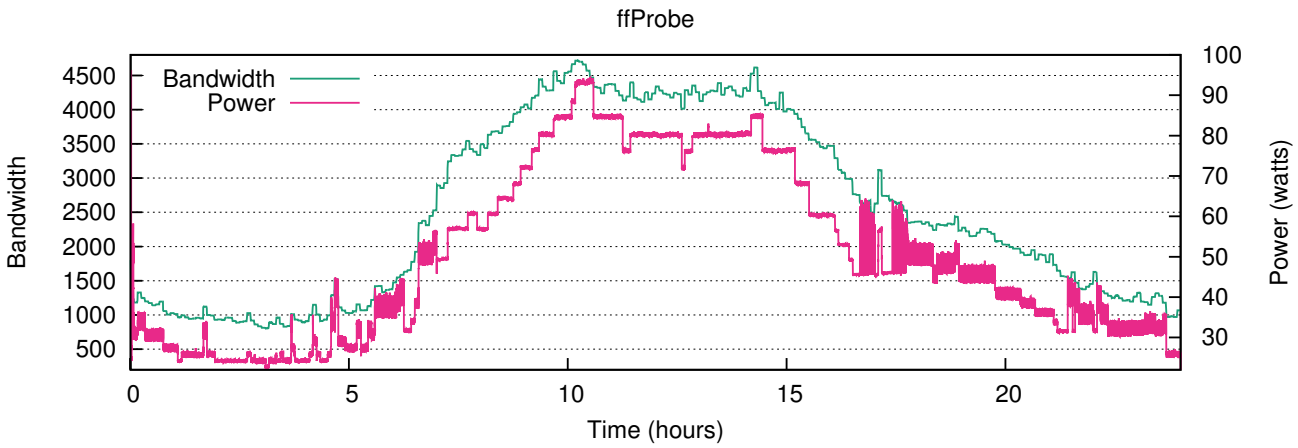

Figure 1: Network monitoring application adaptivity.

frequency scaling. Interested readers can find in [DS] more details about the exact models we used. We applied this technique over all the applications of the PARSEC benchmark suite. Our simulations shown that is sufficient to interpolate 4 configurations in order to achieve an average accuracy of 96% in predicting both performance and power consumption.

Recently, we implemented this technique on a real runtime system, achieving results comparable with those obtained by state of the art solutions and experiencing minimal differences in terms of overhead and accuracy with respect to the optimal algorithm. Differently from the heuristic approach, by using this solution is possible to specify explicit constraints on the power consumption (Figure 2).

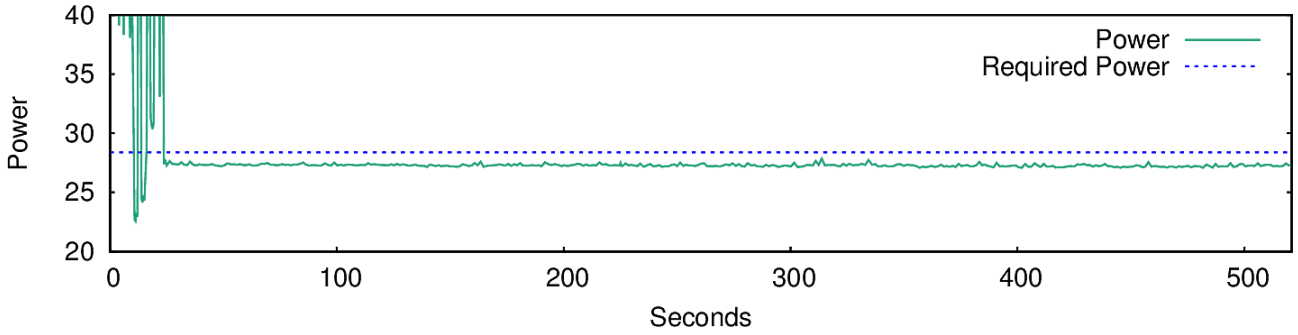

Figure 2: Maximum power consumption constraint on data compression application. In the initial part the application visits different configurations to collect data for building prediction models.

3 Implementation

The techniques we described led to the implementation of NORNIR[2], a self-adaptive, power-aware extension of FASTFLOW[3] a C++ parallel programming framework. By using NORNIR

[2] http://danieledesensi.github.io/nornir/
[3] http://calvados.di.unipi.it/

it is possible to write parallel applications and to express requirement on their maximum power consumption or the minimum performance they should provide. At the moment, we provide *embarrassingly parallel*, *map*, *reduce*, *pipeline* and *dataflow* parallel patterns. The following code snippet shows how to build an embarrassingly parallel video processing application, that should not consume more than 80 Watts:

```
1  class Sch: Scheduler<I>{
2  public:
3      Frame* schedule(){
4          // Reads a video frame. Each frame will be automatically
5          // scheduled to a different Worker
6          return readFrame();
7      }
8  };
9  class Wrk: Worker<Frame>{
10 public:
11     void compute(Frame* in){
12         // Compute something on the frame received by the scheduler.
13         W(in);
14     }
15 };
16 void main(){
17     Parameters par;
18     par.contractType = CONTRACT_TYPE_POWER_BUDGET;
19     par.powerBudget = 80; // Fixed power budget
20     Farm farm<Sch, Wrk>(par);
21     farm.startAndWait();
22 }
```

To read energy consumption, shutdown cores, set frequencies, gathering topology information and for all the other hardware related aspects, we implemented an open-source C++ library (MAMMUT[4]). By using MAMMUT, is possible to perform these operations both on local or remote machines in an object-oriented fashion and for different types of architectures.

We plan to study other mechanisms beside number of cores and clock frequency, finding new prediction algorithms. Moreover, we will investigate control theory techniques to improve the stability of our approach.

References

[DS] Daniele De Sensi. Predicting performance and power consumption of parallel applications. In *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016*.

[DST] Marco Danelutto, Daniele De Sensi, and Massimo Torquati. Energy driven adaptivity in stream parallel computations. In *23rd Euromicro Int'l Conf. on Parallel, Distributed, and Network-Based Processing, PDP 2015*.

[Ran10] Parthasarathy Ranganathan. Recipe for efficiency: Principles of power-aware computing. *Commun. ACM*, 53(4):60–67, April 2010.

[4] http://danieledesensi.github.io/mammut/