Dynamically Building Energy Proportional Data Centers with Heterogeneous Computing Resources

Violaine Villebonnet*†, Georges Da Costa*, Laurent Lefevre†, Jean-Marc Pierson* and Patricia Stolf*
*IRIT, University of Toulouse, France
†Inria Avalon LIP - Ecole Normale Superieure of Lyon, University of Lyon, France

***Abstract*—As the number of data centers increases, it is urgent to reduce their energy consumption. Although servers are becoming more energy-efficient, their idle consumption remains high, which is an issue as data centers are often over-provisioned. This work proposes a novel approach for building data centers with heterogeneous machines carefully chosen for their performance and energy efficiency ratios. We focus on web applications whose load varies over time, and design a scheduler that dynamically reconfigures the infrastructure, by migrating applications and switching machines on or off, so that the energy consumed by the data center is proportional to the load. Experiments evaluate the approach and show the energy savings achieved by our heterogeneous data center design and management while satisfying Quality of Service (QoS) constraints.**

I. Introduction

IT infrastructures, especially data centers, are responsible for a substantial amount of the global energy consumption and greenhouse emissions. While data centers consume a lot of energy, they are often over-provisioned and contain numerous servers that are not fully utilized. An Uptime Institute survey [1] suggests that close to 30% of servers in US enterprises' data centers are *comatose*, meaning that they are consuming power without doing any work. It has been shown that a typical server, when idle, can consume up to 50% of the amount drawn at peak utilization [2]. Therefore we need novel approaches for conceiving and managing data centers more efficiently.

This work provides an approach for designing data centers whose energy consumption is proportional to their load. It focuses on services with variable load, and offers means for adjusting the computational capacity to service requirements by dynamically modifying the number and type of machines hosting the service. The goal is to minimize the energy consumed by allocating the minimum number of resources required to meet application demands. We advocate the use of multiple server architectures that feature heterogeneous ranges of performance and energy consumption. Each machine type is profiled by running the target application, and the best resources combinations for all application performance rates are computed. Our previous work introduced the concept of heterogeneous energy proportional infrastructure, named "Big,Medium,Little" (BML) [3]. The present work extends it by providing a scheduler that handles dynamic reconfiguration decisions, which consist in dynamic resources management with switch on and off actions, whose time and energy overheads are taken into account, to achieve energy proportionality while respecting QoS requirements.

II. Related Work on Energy Proportionality

Barroso and Holzle [2] introduced the goal of energy proportionality in 2007. They discovered that servers' average utilization was between 10 and 50% in a Google data center. The issue is that a typical server is not very energy efficient under low utilization as its idle power consumption can amount up to 50% of its peak consumption. In [4], Varsamopoulos *et al.* define two metrics to quantify energy proportionality: IPR, for Ideal to Peak Ratio, which measures the dynamic power range, and LDR, for Linear Deviation Ratio, to evaluate the linearity of the consumption. They studied the evolution of energy proportionality and found that recent servers feature better characteristics, but most time it only concerns one aspect: a larger dynamic power range or an improved linearity.

Attempts have been made to improve power management of servers, such as Running Average Power Limit (RAPL) by Intel. Via this mechanism a user can specify a power consumption threshold that the processor will not exceed for a given period. Energy savings achieved by RAPL have been evaluated for data stores applications [5], and latency critical workloads [6]. This power capping tool offers better energy proportionality, but does not help reducing idle consumption.

An example of heterogeneity is ARM big.LITTLE [7] that gathers on the same board two different processors:a low power processor that delivers low level performance, and a more powerful one, consequently consuming more, to process intensive tasks. ARM developed this technology to extend mobile devices' battery life. The concept has been adapted to server scale in [8]. Authors designed a motherboard containing a server processor, called primary server, and a low power processor, called the Knight, which is always on and wakes up the primary server in case of high load.

Our work aims at achieving energy proportionality at the data center scale by combining existing heterogeneous architectures. The strong aspect is that it does not rely on a specific processor design. This idea has been introduced previously [9], and its feasibility has been studied [3]. The present work implements scheduling policies that take into account both benefits and drawbacks of using independent machines.

III. Characterizing the Application and its Load

Our system considers applications with variable load and it adapts the infrastructure to load conditions so that energy consumption more closely matches resource utilization. For such,

the application performance is characterized using an *application metric* that represents the amount of work performed over a given time unit. This metric is used to assess the application performance independent of the underlying architecture and to determine the QoS. With respect to performance, applications can be classified as *critical*, having strict performance requirements, and *tolerant*, applications with soft QoS requirements. *Critical* applications can be found in banking and medical areas where delays have serious consequences. More *tolerant* applications are found in, for instance, enterprise services, or services with flexible deadlines. Applications can lie in between these classes, and hence intermediate classes can be required depending on the use-case.

Applications are also classified on whether they can be migrated across machines, and whether they can run on multiple architectures. The former is determined by how the application maintains state and on the amount of data to transfer. We must evaluate the application's migration overhead, both in terms of duration and energy consumption. Another important characteristic is the application malleability, *i.e.*, its ability to be distributed across several machines. If not, the minimum and maximum number of instances should be specified. This criterion poses a constraint when computing the possible hosting machine combinations.

The knowledge of how load evolves, an important parameter in our system, can be *perfect*, when the load can be determined with precision; *partial*, where certain characteristics are known, such as weekly, diurnal, hourly patterns, but the accuracy of variations is unknown; and *unknown* when no a priori information is available, and the load must be predicted.

IV. Building BML Infrastructure in 5 Steps

This section lays out the steps towards selecting hardware for our heterogeneous energy proportional data center. Four illustrative architectures are used as input, however this methodology is not dependent on the number of architectures. Results with real hardware are presented in Section V.

A. Step 1: Characterizing Each Architecture Profile

The first step consists in building the energy and performance profiles for all available hardware considering the target application. The profile of an architecture i provides:

- $idlePower_i$: average idle power of architecture i, in Watts.
- $maxPerf_i$: maximum performance rate, expressed with the application metric (*e.g.*, nb of requests processed per second).
- $maxPower_i$: average power consumed when it reaches $maxPerf_i$ rate, expressed in Watts.

A function is created using this data to compute the power consumed by architecture i for a given performance rate $perfRate$. We make the assumption of linear power consumption, even if we know it might lead to small under- or over-estimation, as studied by Rivoire *et al.* [10]. Yet, this approximation is precise enough for our solution, and eases the profiling phase. Although acquiring more intermediate data points, if the application allows, would enable more precision, our methodology would not be affected.

We consider that enough machines of each type are available to choose from when building machine combinations, which enables creating ideal combinations. With minor changes, this work can consider cases of existing heterogeneous infrastructure where there is limited numbers of machines of each type.

B. Step 2: Sort Architectures to Keep Only BML Candidates

Building a BML infrastructure starts by sorting machines by decreasing maximum performance. Then we check that their respective maximum power consumption respects this initial ordering. We proceed by comparing sorted architectures in pairs; if one has lower performance than another while consuming more energy, we remove it from the BML candidates as it would not improve energy proportionality.

Figure 1 gathers the profiles of illustrative architectures A, B, C, and D. Beyond the point $(maxPerf_i, maxPower_i)$ of each architecture, its profile is repeated to picture multiple nodes. In this case, only three architectures are kept as good candidates for a BML infrastructure, namely A, B and C. Architecture D is discarded because its maximum power consumption is greater than A's, the most powerful machine. Once the initial filtering is done, architectures can be labeled according to their performance as *Big*, *Medium* or *Little*. In this example: A $\leftarrow Big$, B $\leftarrow Medium$, and C $\leftarrow Little$.

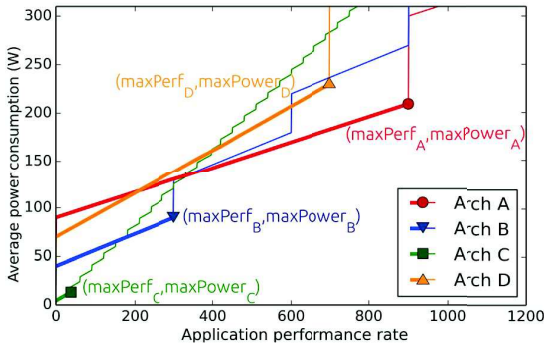

Fig. 1. Architectures A, B and C are good candidates for BML infrastructure, but D will be removed due to its poor energy efficiency compared to A.

C. Step 3: Finding Crossing Points between Architectures

This step determines how architectures should be combined for more power proportionality. We define the *minimum utilization threshold* for each architecture, expressed with the application performance metric. For two architectures, i as $Little$ and j as Big, the minimum threshold of architecture j corresponds to the point from which utilization of j becomes more relevant than i's considering power consumption. Except for the *Little* architecture whose minimum threshold is 1, a function is needed to compute thresholds for other architectures. The points where an architecture becomes preferable over another are termed as *crossing points* as they represent the points where power profiles meet.

Left part of Figure 2 illustrates this step with architectures A, B and C, now denoted Big, $Medium$ and $Little$. Utilization threshold of $Medium$ starts around a performance rate of 150. Before this point, it is more efficient to use up to five $Little$ nodes. The minimum utilization threshold of Big corresponds

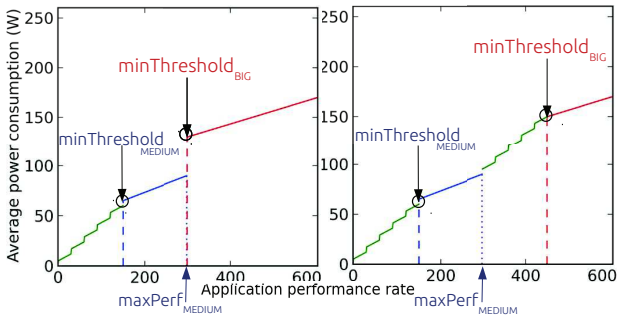

Fig. 2. On left: *Step 3* - First step of crossing points computation between $Little$ and $Medium$, and between $Medium$ and Big. On right: *Step 4* - Second step of crossing points computation between $Little$ and $Medium$, and between combinations of $Medium - Little$ and Big

to the maximum performance rate of a $Medium$ node. This results in a substantial jump in power consumption when switching from $Medium$ to Big since this crossing point is not optimal, and next step is needed to improve it.

D. Step 4: Finding Crossing Points between Architectures and Combinations of Smaller Architectures

This step is required for more than two architectures. The previous step computes the crossing points between homogeneous combinations of machines, but with three architectures one must determine whether adding $Little$ nodes to $Medium$ combinations help improve power proportionality and reduce the gap between $Medium$ and Big architectures. Right part of Figure 2 pictures the re-evaluated crossing points and shows that minimum threshold of Big has consequently increased.

E. Final step: Computing Ideal BML Combination

This step computes the ideal machine combinations and their corresponding power consumption for a given performance rate. Building a BML combination is similar to a bin-packing problem where architectures and their maximum performance represent bins of different sizes. The singularity of our problem is that there is only one object to pack, *i.e.*, target performance, but it can be divided into any pieces of any size. Bins are now sorted by size and cost, so what is left to perform is to divide the desired performance into pieces to fill them. Firstly, we consider architectures sorted decreasingly and seek to fill completely Big nodes, then $Medium$, and so on. Architectures are the most energy efficient when fully loaded. Secondly, we use minimum thresholds to choose the right architecture to process the remaining performance.

V. Experimental Evaluation

A. Experimental Setup and Profiling Results

The processors we have chosen to profile are the following: *Paravance*: x86 Intel Xeon E5-2630v3 (2x8 cores); *Taurus*: x86 Intel Xeon E5-2630 (2x6 cores); *Graphene*: x86 Intel Xeon X3440 (1x4 cores); *Chromebook*: ARM Cortex-A15 (1x2 cores); *Raspberry*: ARM Cortex-A7 (1x4 cores). A *WattsUp?Pro* wattmeter monitors power consumption of the *Samsung Chromebook* and *Raspberry Pi2B+*. The x86 servers are available at

TABLE I
Performance and Power Profiles of Each Architecture.

Architecture Codename	MaxPerf (reqs/s)	Idle-Max Power (Watts)	On_t (s)	On_E (Joules)	Off_t (s)	Off_E (Joules)
Paravance	1331	69.9 - 200.5	189	21341	10	657
Taurus	860	95.8 - 223.7	164	20628	11	1173
Graphene	272	47.7 - 123.8	71	4940	16	760
Chromebook	33	4 - 7.6	12	49.3	21	77.6
Raspberry	9	3.1 - 3.7	16	40.5	14	36.2

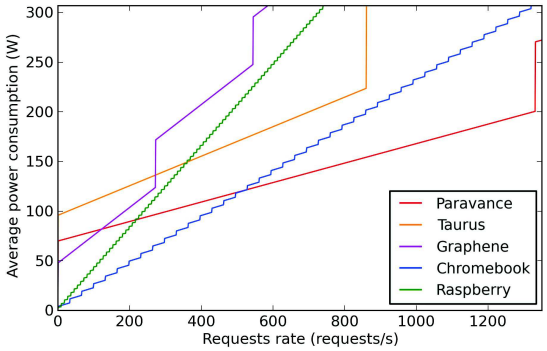

Fig. 3. Power and performance profiles of web servers acquired from experiments on 5 different architectures

Grid'5000 [11], a French experimental testbed for research, whith power monitoring data accessible via *Kwapi* [12].

A stateless web server is our target application because:
- a load balancer could allow the load to be distributed among several web server instances;
- being stateless, it can be easily migrated by stopping a server instance and launching a new one on the destination machine, and then updating the load balancer;
- it is a perfect example of application with variable load over time, and its performance can be characterized with an application metric: number of requests processed per second.

We use *lighttpd* as web server and *Siege* as web benchmark tool. The content of the web server is a python cgi script. Each request consists in a loop of random number generation, while loop iterations is also chosen randomly between 1000 and 2000. The request response is a static html page containing this later integer. We execute the benchmark with an increasing number of concurrent clients in order to find the maximum request rate that can be processed. Each test runs for 30 seconds and the maximum performance is the average of 5 results. We also measure On/Off durations and energy consumption. Table I and Figure 3 present the results.

B. Results at Server-Scale

Figure 3 is the result of *Step 1* described in Section IV. *Step 2* results in the removal of *Taurus* architecture as its maximum power consumption is higher than *Paravance*'s while delivering lower performance. *Step 3* realizes that the profile of *Graphene* never crosses any other architecture's profile, and consequently it is removed from the list of candidates. Our final heterogeneous infrastructure comprises *Raspberry* ($Little$), *Chromebook* ($Medium$) and *Paravance* (Big). Their minimum utilization thresholds are respectively 1, 10 and 529 requests per second. The resulting ideal BML combination is depicted in Figure 4. Big architecture's profile is also

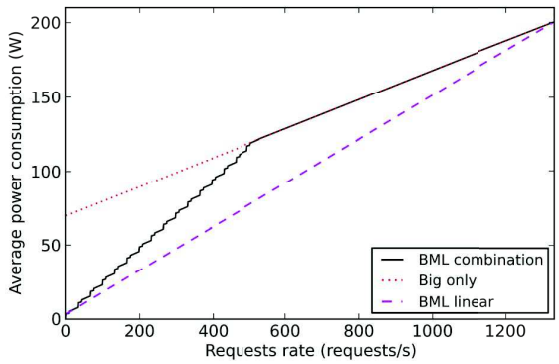

Fig. 4. Consumption of BML combination over an increasing performance rate, until $maxPerf_{Big}$, compared to Big and $BML\ linear$

represented to demonstrate the gains of the heterogeneous combination. In addition, we introduce $BML\ linear$ whose idle power is equal to $Little$'s and maximum power and performance is equal to Big's. It represents an achievable goal, and how our solution approaches it.

C. Comparison with Lower and Upper Bounds

To evaluate our approach, we developed a simulator in Python, which takes as input the experimental machine profiles, and a trace file describing the application load variation over time. We emulate a load prediction mechanism by considering a sliding look-ahead window. We use as prediction the maximum load value over a window of 378 seconds, equivalent to 2 times the longest On duration. The scheduling policy is pro-active; at each prediction it computes the corresponding BML combination, and if this leads to a new hardware configuration, a decision of reconfiguration is taken. During the reconfiguration, no other decision can be made, ensuring the completion of On/Off actions before a new decision. The next prediction window starts from reconfiguration completion time, but if there is no combination changes, the window just slides one time step forwards, a second in this case.

We compare this placement algorithm against a theoretical BML lower bound and two homogeneous upper bounds corresponding to existing data center management. We run the simulations for days 6 to 92 of 1998 World Cup traces (available online). The resulting scenarios are as follows:

- *UpperBound_Global*: a homogeneous data center with constant number of servers, computed according to the maximum request rate. Here it contains 4 *Big* machines always On. This is an example of a classical over-provisioned data center.
- *UpperBound_PerDay*: a homogeneous data center, but dimensioned each day according to the daily maximum rate. This is an example of coarse grain capacity planning.
- *Big-Medium-Little*: our BML infrastructure and placement algorithm. The total consumption per day contains the energy consumed by computation and by On/Off reconfigurations.
- *LowerBound_Theoretical*: the minimum computing energy achievable with BML infrastructure if it is dimensioned every second with the ideal combination. This is an unreachable lower bound considering no On/Off latency and energy costs, picturing the best energy proportionality we could reach.

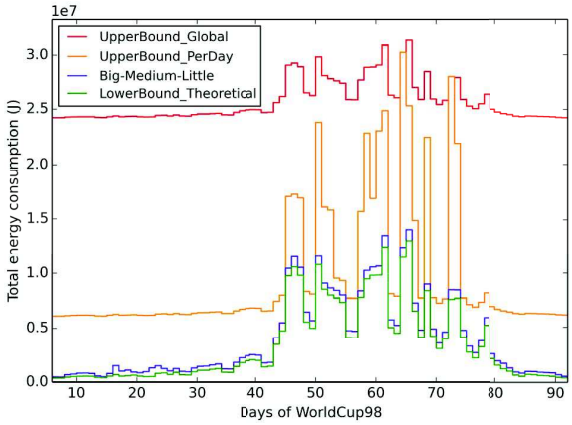

Fig. 5. Energy consumption comparison with lower and upper bounds.

Figure 5 shows that our solution is very close to the theoretical lower bound: on average over 86 days, it consumes 32% more energy than the lower bound, minimum 6.8% and maximum 161.4%. This graph demonstrates the high static costs of classical over-provisioned data center, and shows the energy proportionality achieved by our solution.

VI. Conclusion and Perspectives

We demonstrated the feasibility of BML infrastructure with existing hardware, and evaluated its performance when hosting stateless web servers. We showed that compared to classical data center management, BML drastically reduces static costs and consequently achieves energy proportionality. As future work we will investigate the impact of load prediction errors on reconfiguration decisions. It is also worth considering other hardware combinations than pre-computed BML combinations as reconfiguration possibilities, and take in account their corresponding overheads when taking reconfiguration decisions.

Acknowledgments

Experiments presented in this paper were carried out on Grid5000 testbed, supported by a scientific interest group hosted by Inria, including CNRS, RENATER, several Universities and organizations (*www.grid5000.fr*). This research is partially supported by the French ANR MOEBUS project.

References

[1] Uptime Institute. 2014 Data Center Industry Survey. 2014.
[2] L.A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.
[3] V. Villebonnet *et al.* "Big, Medium, Little": Reaching Energy Proportionality with Heterogeneous Computing Scheduler. *PPL*, 2015.
[4] G. Varsamopoulos *et al.* Trends and Effects of Energy Proportionality on Server Provisioning in Data Centers. *Int. Conf. on HPC*, 2010.
[5] B. Subramaniam and W. Feng. On the Energy Proportionality of Distributed NoSQL Data Stores. *Int. Workshop in HPC Systems*, 2014.
[6] D. Lo *et al.* Towards Energy Proportionality for Large-scale Latency-critical Workloads. *SIGARCH Comput. Archit. News*, 2014.
[7] ARM big.LITTLE: The Future of Mobile. *ARM White Paper*, 2013.
[8] D. Wong and M. Annavaram. Scaling the Energy Proportionality Wall with KnightShift. *IEEE Micro*, 2013.
[9] G. Da Costa. Heterogeneity: The Key to Achieve Power-Proportional Computing. *IEEE CCGrid*, 2013.
[10] S. Rivoire *et al.* A Comparison of High-level Full-system Power Models. *Conference on Power Aware Computing and Systems (HotPower)*, 2008.
[11] R. Bolze et al. Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *Int. Journal of HPC Applications*, 2006.
[12] F. Rossigneux *et al.* Generic and Extensible Framework for Monitoring Energy Consumption of OpenStack Clouds. *IEEE SustainCom*, 2014.