On the Impact of Advance Reservations for Energy-Aware Provisioning of Bare-Metal Cloud Resources

Marcos Dias de Assuncao, Laurent Lefèvre, Francois Rossigneux

On the Impact of Advance Reservations for Energy-Aware Provisioning of Bare-Metal Cloud Resources

Marcos Dias de Assunção, Laurent Lefèvre, François Rossigneux
Inria Avalon, LIP Laboratory, ENS de Lyon, France
{marcos.dias.de.assuncao, laurent.lefevre}@ens-lyon.fr

***Abstract*—This work investigates factors that can impact the elasticity of bare-metal resources. We analyse data from a real bare-metal deployment system to build a deployment time model, which is used to evaluate how provisioning time impacts the reservation of bare-metal resources. Climate/Blazar, a reservation framework designed for OpenStack, is discussed. Simulation results show that reservations can help reduce the time to deliver a provisioned cluster to its customer while achieving energy savings similar to those of strategies that switch-off idle resources.**

I. Introduction

The workload consolidation that clouds provide by virtualising resources and enabling customers to share the underlying physical infrastructure brings benefits such as energy efficiency and better system utilisation. Customers can request resources on demand and pay for their use on a per-hour basis. Such elasticity allows for adjusting the allocated capacity dynamically to meet fluctuating demands.

Though this model suits several use cases, certain applications are not fully portable to this scenario as they are resource intensive and sensitive to performance variations. The means used by cloud providers to offer customers with high and predictable performance mostly consist in deploying bare-metal resources or grouping Virtual Machines (VMs) where high network throughput and low latency can be guaranteed. This model contrasts with traditional cloud use cases as it is costly and provides little flexibility in terms of workload consolidation and resource elasticity.

Over the past, High Performance Computing (HPC) users for instance have been tolerant to resource availability as they generally share large clusters to which exclusive access is made by submitting a job that may wait in queue for a period often longer than the job execution itself. Users of bare-metal services also commonly accept provisioning delays that can vary from hours to several days. Although we do not think clouds should adopt a similar queuing model, we believe that a compromise between wait time and on-demand access could be exploited for bare-metal resources in the cloud via resource reservations. Reservations provide means for reliable allocation and allow customers to plan the execution of their applications, which is key to many use cases that require bare-metal and specialised resources.

In this work, we analyse historical data on the provision of bare-metal resources from a real system to model the time required by bare-metal deployment. Using this model, results from discrete-event simulations demonstrate (i) the energy-saving potential of strategies that switch unused resources off and (ii) how reservations can help reduce the time to deliver a provisioned cluster to its customer.

II. The Reservation System

Climate, conceived during the FSN XLCloud project[1] and renamed Blazar when incorporated into OpenStack[2], enables reserving and deploying bare-metal resources whilst considering their energy efficiency. Its architecture (Figure 1) comprises the following components:

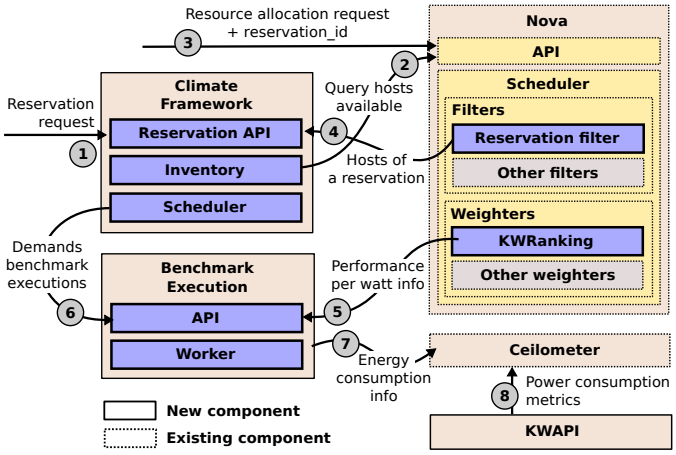

Fig. 1: Architecture of the proposed reservation framework.

- *Reservation API*: used by client applications and users to reserve resources and query the status of reservations.
- *Climate Inventory*: a service that stores information about physical nodes used for reservations.
- *Climate Scheduler*: responsible for scheduling reservation requests on available nodes.
- *Energy-Consumption Monitoring Framework*: monitors the energy consumption of physical resources and interfaces with OpenStack telemetry infrastructure.

The Climate Scheduler manages reservations and extends Nova's filtering scheduler with a set of resource filters and

[1]http://xlcloud.org
[2]https://wiki.openstack.org

ranking (or weighting) criteria. Nova filter accepts a scheduling hint, here used to provide a reservation ID. When a Climate-created ID is provided, the filter uses the reservation API with an admin Keystone token to retrieve the list of hosts associated with the reservation.

Two Nova weighters were created. The first weighter, ignored by reservation requests, ranks machines by their free time until the next reservation. If handling a request for a non-reserved instance, the weighter tries to place the instance on a host that is available for the longest period. This helps minimise the chance of having to migrate the instance later to vacate its host for a reservation. The second weighter, called KiloWatt Ranking (KWRanking), ranks machines by their power efficiency (*i.e.* FLOPS/Watt) and relies on: a software infrastructure [1] to monitor the power consumed by resources of a data centre and to interface with Ceilometer; and a benchmark executed on the machines to determine their performance per Watt.

III. Modelling Bare-Metal Deployment

To model the time required for bare-metal deployment, we use traces from Grid'5000, an experimental platform comprising several sites in France and Luxembourg [2]. The traces were generated by Kadeploy3 [3], a disk imaging and cloning tool that takes a file containing the operating system to deploy (*i.e.* an environment) and copies it to target nodes. An environment deployment consists of three phases:

1) Minimal environment setup, where nodes reboot into an OS with tools for partitioning disks.
2) Environment installation, when the environment is broadcast and copied to all nodes, and post-copy operations are performed.
3) Reboot of nodes using the deployed environment.

We gathered several years of Kadeploy3 traces from five clusters on three Grid'5000 sites (Table I) and evaluated the time to execute the three phases described above. All deployments from Jan. 2010 through Dec. 2013 were considered. The first step towards building a model consisted in creating time histograms and visually examining probability distributions that were likely to fit the data. Scott's method was used to determine the size of histogram bins [4]. After considering a number of distributions, we found that log-normal, gamma and generalised gamma were most likely to fit the data. Figure 2 depicts the results of fitting these distributions to the deployment time information of each cluster. In general, deployment presents an average completion time with eventual failures overcome by executing extra routines and performing additional server reboots.

The goodness of fit of the distributions has also been submitted to the Kolmogorov–Smirnov test (KS test), whose D-statistic quantifies the distance between the distribution function of empirical values and the cumulative distribution function of the reference distribution. As shown in Table II, log-normal provides slightly better fit to most clusters, and is therefore used to model deployment time.

TABLE I: Grid'5000 clusters whose deployment logs were considered.

Cluster Name	# Nodes	Node Characteristics
parapluie	40	2 12-core CPUs AMD 1.7GHz, 48GB RAM, 232GB DISK
parapide	25	2 4-core CPUs Intel 2.93GHz, 24GB RAM, 465GB DISK
paradent	64	2 4-core CPUs Intel 2.5GHz, 32GB RAM, 298GB DISK
stremi	44	2 12-core CPUs AMD 1.7GHz, 48GB RAM, 232GB DISK
sagittaire	79	2 CPUs AMD 2.4GHz, 1GB RAM, 68GB DISK

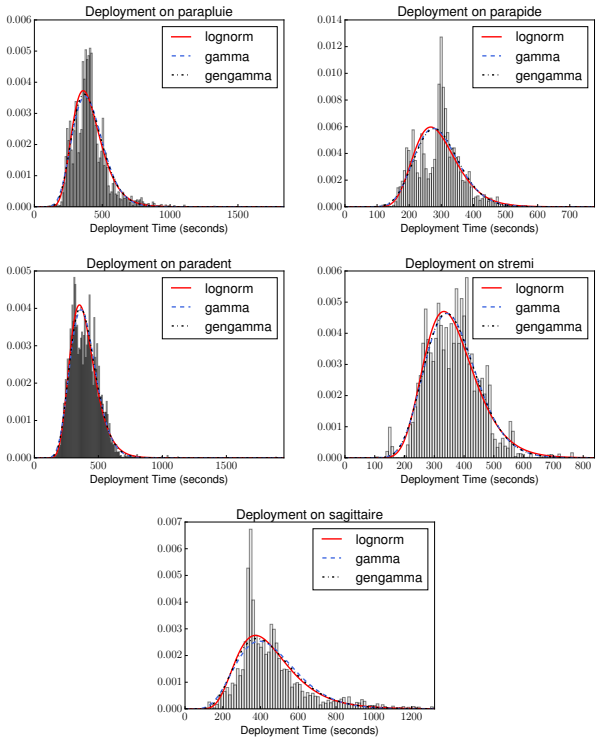

Fig. 2: Deployment time histograms and distribution fitting.

TABLE II: Kolmogorov-Smirnov test for goodness of fit.

Cluster Name	D-Statistics		
	Log-normal	Gamma	Gen. Gamma
parapluie	0.051	0.066	0.059
parapide	0.111	0.095	0.091
paradent	0.041	0.046	0.043
stremi	0.051	0.036	0.039
sagittaire	0.067	0.076	0.070

IV. Reservation Strategies

Power-Off Idle Resources: this strategy checks resources periodically, and if a resource remains idle during a given time (*i.e. idleness interval*), it is powered off.

Reservation-Based Power-Off: similar to powering-off idle resources, but when assessing a resource idleness it also determines whether the resource is likely to remain unused over a time *horizon*. The strategy also boots resources in advance to serve previously scheduled reservations. The av-

erage deployment time and a small safety margin are used to determine how long in advance resources must be deployed. The length of the idleness interval and time horizon over which the server must remain unused to become a candidate for switch off are 5 and 30 minutes respectively.

Reservation With Minimum Capacity Estimation: in addition to delaying when resources are made available to users, frequent server initialisation and shut down can be detrimental to energy efficiency [5]. This strategy seeks to avoid frequent reinitialisation by using a technique proposed in our previous work to configure a minimum resource pool, with a capacity below which decisions to switch resources off are ignored [6].

Resource utilisation is used to determine when the minimum capacity is adjusted. Utilisation at time t, denoted by υ_t, is the ratio between the number of resource hours effectively used to handle requests and the number of hours resources were powered on. The provider sets parameters H and L, $0 \leq L \leq H \leq 1$, indicating utilisation lower (L) and upper (H) thresholds according to which additional capacity is required or powered-on resources are not needed, respectively. The minimum pool capacity should be modified before utilisation reaches undesired levels, which requires a prediction on future resource utilisation. The estimation is based on the measurements performed over the past i measurement intervals. Namely, after measuring υ_t at time t, weighted exponential smoothing is used to predict the utilisation for step $t+1$. If the past $v \leq i$ measurements (*i.e.*, $\upsilon_{t-v}, \upsilon_{t-v+1}, \ldots, \upsilon_t$) and the forecast utilisation are below (above) the lower (upper) threshold L (H), the minimum capacity must be adjusted. In this work, $L = 0.4$, $H = 0.9$, $i = 5$ and $v = 10$.

V. Performance Evaluation

A discrete-event simulator is used to model and simulate resource allocation and request scheduling[3]. We adapted the Google workload logs [7] to model cloud users' resource demands[4]. The trace contains a log of job submissions, their schedule and execution, where each job comprises one or multiple tasks that are executed on containers deployed on one or multiple machines. The original resource demands (*e.g.* memory, CPU, disk) are normalised by the configuration of the largest machine. To determine the number of physical machines m_j that a job j requires, we obtain the maximum set of simultaneous tasks in execution T_j over the duration of job j and compute $m_j = min\{a, c * \sum_{t \in T_j} t/mac_t^{mem}\}$, where mac_t^{mem} is the normalised capacity of the machine that executed task t; c is a constant representing the available host's memory capacity allocated to containers – here set to 0.85; and a is a constant that specifies the maximum number of machines per request, set to 50 to prevent creating workloads that are extremely bursty.

Certain jobs are very short and probably part of submission bursts for which we consider a user would make a single reservation. Hence jobs are grouped using a technique proposed for bag-of-tasks applications [8]. The *continued submission* grouping scheme is applied with $\Delta = 180$ seconds. From the original trace that contains jobs submitted by a total of 933 services, we crafted five different workloads, each comprising job submissions from 250 randomly selected services; these traces, depicted in Figure 3, are taken as the *cloud* workloads. As the original trace does not contain reservations, we create *reservation* workloads by randomly selecting requests that require reservations, where the reservation ratio varies as described later. Original job start time is used as the reservation start time and how long in advance the reservation is made is uniformly drawn from an interval of 0 to 24h.

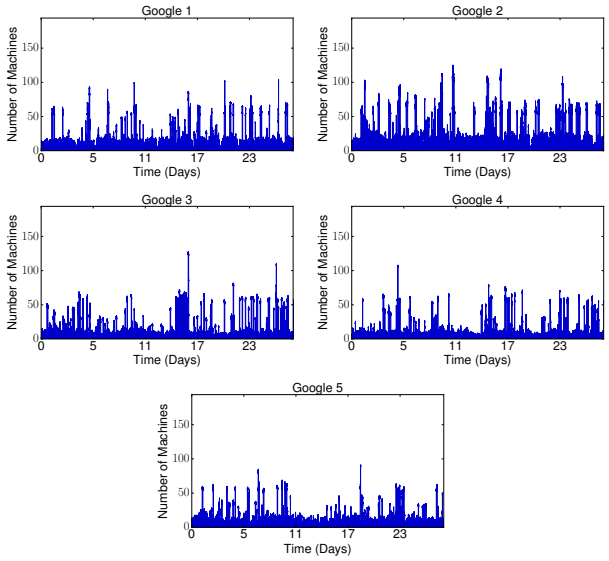

Fig. 3: Overview of Google cloud workloads.

Infrastructure capacity and resource requests are expressed in number of machines. The maximum number of machines available at a site is computed by simulating the request scheduling of its corresponding *cloud* workload under a large number of machines, so that each request is treated immediately as it arrives and no request is rejected. The maximum number of machines used during this evaluation is taken as the site capacity. Based on the deployment information from Kadeploy, we model the time in seconds to boot powered-off machines using a log-normal distribution whose scale is 6 and shape is 0.4. We take 25 minutes as the time a machine must remain idle to be a candidate for switch off and 30 minutes as the future horizon to check whether it is committed to reservations. The evaluation of candidates for switch off is performed every 5 minutes. Moreover, the following schemes for resource provisioning are considered:

- **Cloud Always On:** baseline scenario that uses the *cloud* workloads and maintains all servers constantly on. It is also used to determine the resource capacity to handle requests in an on-demand, cloud-like manner.
- **Cloud Switch Off:** does not consider reservations, employs the *cloud* workloads and the policy that switches servers off if they remain idle for a given interval.
- **Reservation Switch Off:** uses the reservation traces and

[3]Available at: https://github.com/assuncaomarcos/servsim

[4]The original trace provides data over a month-long period in May 2011 from a 12k-machine set used by the Exploratory Testing Architecture.

the reservation policy that switches off servers that remain idle for an interval and that are not committed to requests over a time horizon. It also boots servers in advance to fulfil previously scheduled reservations.

- **Reservation Minimum Pool:** similar to reservation with switch off, but keps a minimum resource pool.

A. Performance Metrics

Energy Saving Potential: shows how much of the server idleness time is used for switch-off. The total server idleness si under the *Cloud Always On* scenario is the maximum time during which servers could potentially be switched off, and it is hence considered the upper bound on potential energy savings. The si of a site is: $si = \int_{t_0}^{t_{last}} s_{total} - s_{used}\, dt$ where t_0 is the start of the evaluation, t_{last} is when the last request is submitted, s_{total} is the total number of servers available at any time, and s_{used} is the number of machines in use at time t. The potential for energy saving is the percentage of si during which servers are switched off.

Request Aggregate Delay: the time users have to wait to have their requests serviced. The aggregate delay ad of requests whose Quality of Service (QoS) has been impacted (R_{delay}) is given by: $ad = \sum_{r \in R_{delay}} (r_{dep\_start} + r_{dep\_end}) - r_{start\_time}$ where r_{dep_start} is when the deployment of the servers required by request r started, r_{dep_end} is when the last server became ready to use, and r_{start_time} is when the request was supposed to start; that is, when the user expected the servers to be available. The distributions of deployment time obtained while inspecting Kadeploy traces give a conservative estimate to evaluate ad. Certain public cloud providers publicise provisioning times of bare-metal resources much higher than those found in Grid'5000.

B. Evaluation Results

Figures 4 and 5 show the potential energy savings and request aggregate delay where 40% of requests of each workload are randomly selected to require reservations. As *Cloud Switch Off* switches servers off almost immediately once they become idle, it is able to achieve higher energy saving potential. This simple policy, however, does not consider the cost of powering off/on resources and hence presents the largest request aggregate delay. *Reservation Switch Off* exploits less idle time, but leads to smaller QoS degradation.

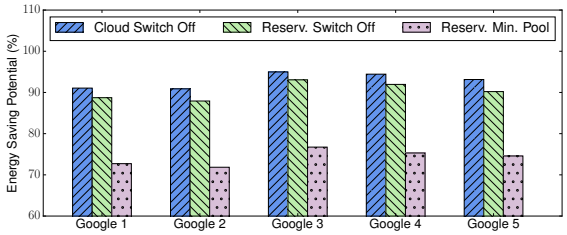

Fig. 4: Energy saving potential for the various scenarios.

As shown in Figure 4, reservation with a minimum resource pool presents smaller energy savings compared to the simple reservation strategy, but it reduces the request aggregate delay.

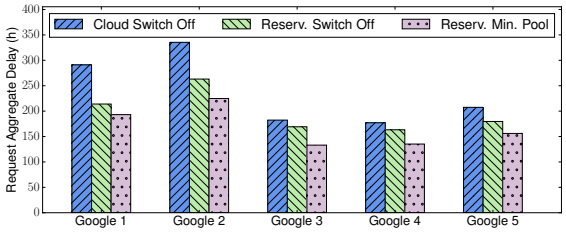

Fig. 5: Request aggregate delay in resource/hour.

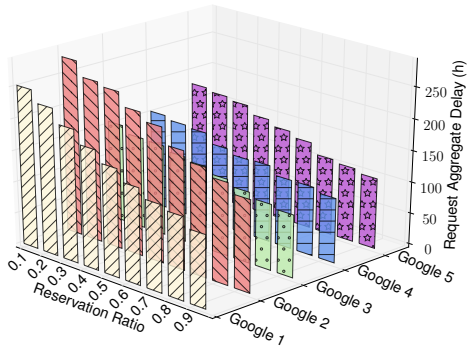

Fig. 6: Request delay in resource/hour for Google workloads.

Although reservation reduces the aggregate request delay as shown in Figure 5, this reduction results in smaller energy saving potential. Further investigation of this issue revealed that the exponential smoothing applied to forecast required server capacity leads to a minimum number of resources being kept on during longer periods than under other strategies. Although at first one may assume that other strategies can always better exploit the bursty behaviour of the Google workloads, it is important to note that we use conservative bare-metal deployment times. We believe that when considering the times reported by the industry – where servers take several hours to be provisioned or recycled – the smoother behaviour of reservation with minimum pool is preferable.

To evaluate the impact of reservations on the request aggregate delay, we varied the reservation ratio from 0.1 to 0.9 (*i.e.* from 10% to 90% of requests require reservations). Figure 6 shows that in general the aggregate delay is inversely proportional to the reservation ratio. We highlight that the impact of reservations on the aggregate request delay would be higher if jobs from the original trace had not been grouped to build what we believe is a more realistic scenario of bare-metal deployment.

VI. Conclusion

We analysed historical information on bare-metal deployment and evaluated strategies for switching servers off in a cloud data centre. Results show that under the evaluated scenarios, reservations reduce the time to deliver resources to users when compared to allocation strategies that naively switch off idle servers.

ACKNOWLEDGEMENTS

This research was partially supported by the FSN XLCloud and the CHIST-ERA STAR projects. Some experiments were carried out using Grid'5000 (see https://www.grid5000.fr).

REFERENCES

[1] F. Rossigneux *et al.*, "A generic and extensible framework for monitoring energy consumption of OpenStack clouds," in *SustainCom 2014*, Dec. 2014.

[2] R. Bolze *et al.*, "Grid'5000: a large scale and highly reconfigurable experimental Grid testbed," *Int. J. of High Perf. Comp. Applications*, vol. 20, no. 4, pp. 481–494, Nov. 2006.

[3] E. Jeanvoine *et al.*, "Kadeploy3: Efficient and Scalable Operating System Provisioning," *USENIX ;login:*, vol. 38, no. 1, pp. 38–44, Feb. 2013.

[4] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[5] A.-C. Orgerie *et al.*, "Save watts in your Grid: Green strategies for energy-aware framework in large scale distributed systems," in *ICPADS'08*, Melbourne, Australia, Dec. 2008, pp. 171–178.

[6] M. D. Assuncao *et al.*, "Impact of user patience on auto-scaling resource capacity for cloud services," *Future Generation Computer Systems*, vol. 55, pp. 41–50, 2016.

[7] C. Reiss *et al.*, "Google cluster-usage traces: Format + schema," Google Inc., Mountain View, USA, Tech. Report, Nov. 2011.

[8] A. Iosup *et al.*, "The characteristics and performance of groups of jobs in grids," in *Euro-Par 2007 Parallel Processing*, ser. LNCS, A.-M. Kermarrec *et al.*, Eds. Springer, 2007, vol. 4641, pp. 382–393.