Analysis of Mixed Workloads from Shared Cloud Infrastructure

Dalibor Klusáček and Boris Parák

CESNET a.l.e., Brno, Czech Republic
{klusacek,parak}@cesnet.cz

Abstract. Modern computing environments such as clouds, grids or HPC clusters are both complex and costly installations. Therefore, it has always been a major challenge to utilize them properly. Workload scheduling is a critical process in every production system with an unwanted potential to hamper overall performance if the given scheduler is not adequate or properly configured. Therefore, researchers as well as system administrators are frequently using historic workload traces to model/analyze the behavior of real systems in order to improve existing scheduling approaches. In this work we provide such real-life workload traces from the CERIT-SC system. Importantly, our traces describe a "mixed" workload consisting of both cloud VMs and grid jobs executed over a shared computing infrastructure. Provided workloads represent an interesting scheduling problem. First, these mixed workloads involving both "grid jobs" and cloud VMs increase the complexity of required (co)scheduling necessary to efficiently use the underlying physical infrastructure. Second, we also provide a detailed description of the setup of the system, its operational constraints and unresolved issues, putting the observed workloads into a broader context. Last but not least, the workloads are made freely available to the scientific community allowing for further independent research and analysis.

Keywords: Cloud, Grid, Workloads, Scheduling, Simulation

1 Introduction

Workload traces from various real-life systems have been used by researchers for decades. Notable examples represent the Parallel Workloads Archive [4] and the Grid Workloads Archive [7] that contains plethora of historic workloads, mainly from HPC-like systems. Similarly, there are several publicly available traces[1] from large cloud/hadoop/cluster installations including, e.g., Eucalyptus IaaS cloud workload [25], Facebook Hadoop traces [24] or Google cluster workload [22].

The increasing popularity of resource virtualization introduces new scheduling problems. For example, different types of applications/frameworks can now be hosted simultaneously in a shared physical infrastructure. This was not a

[1] Nice overview can be found at: http://bit.ly/2kLf44d

usual scenario 15 years ago. Today, applications and services can be relatively easily encapsulated as, e.g., VMs or containers and run in an isolated fashion within a data-center. Therefore, it is very important to collect information about such installments, where different computing paradigms meet in a single infrastructure in order to understand the nature and influence of such co-existence and its potential impact on existing scheduling approaches, both local (e.g., batch scheduling system) and global (e.g., Mesos-like inter-application scheduling [6]).

The traces presented in this paper come from the Czech *CERIT Scientific Cloud (CERIT-SC)* installation [2] and represent such mixed workloads. Simply put, they capture a mixture of two different types of applications — standard computational jobs and cloud VMs — that each use their own resource manager and scheduling approaches to handle their jobs and VMs, respectively. Both of them use the same fully virtualized infrastructure to execute the workloads. Moreover, we present detailed information about system configuration and its usage policies, maintenance periods and operational constraints, including details concerning applied scheduling approaches and system performance objectives. We also discuss current problems with existing system setup and provide several examples of future research and development. Of course, these workload traces are freely available to other researchers [12].

This paper is organized as follows. Section 2 describes the shared computing environment of CERIT-SC, giving details about the hardware, middleware and system constraints including its scheduling policies. Next, the workload traces are presented in Section 3 and their major characteristics are discussed and compared. Section 4 describes the optimization criteria that the CERIT-SC uses when optimizing its performance. Existing open problems are discussed in Section 5 where several unresolved issues with the current system configuration are presented. Finally, Section 6 describes the formats of our workload traces, introduces some available workload parsers and concludes the paper.

2 System Description

Workload traces described in this paper were collected during the year 2016 and come from the CERIT-SC site, which is the largest partition of the Czech national grid and cloud infrastructure *MetaCentrum* [18]. MetaCentrum has some 13,288 CPU cores out of which 5,224 belong to the CERIT-SC partition. Within CERIT-SC, 3,912 CPU cores (75%) are fully virtualized using OpenNebula framework [19] and can be used by various applications, while the remaining 1,312 CPU cores are not virtualized and are exclusively used for "bare metal" grid-like computations. In this paper, we will only concentrate on the mixed workloads coming from the shared and fully virtualized partition.

The shared partition is managed by OpenNebula and allows for simultaneous execution of two major classes of workloads. The first type is represented by classic virtual machines (VM) that are submitted by the users of the system and serve for various purposes, e.g., they host a database or a web server, or they encapsulate some "exotic" software or operating system (OS) which cannot be

executed on the "bare metal" nodes that all use Debian 8 OS. Second, there is a special type of VM which we call a "grid worker" VM. Once deployed and started, this VM behaves as a "normal" node of the grid infrastructure, i.e., computational jobs from the CERIT-SC's batch resource manager (RM) can be executed within such VMs. The scheme of the current system configuration is shown in Fig. 1.

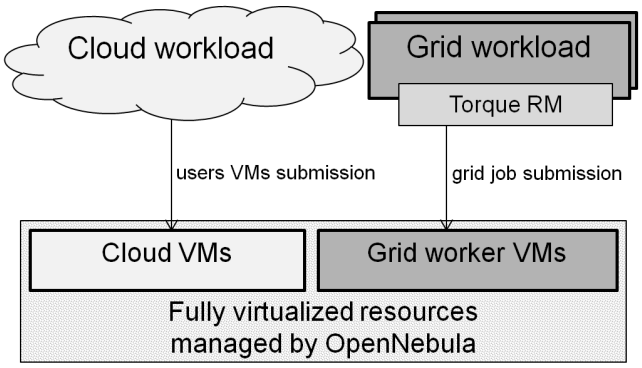

Fig. 1. The scheme of the shared virtualized infrastructure in CERIT-SC.

The biggest advantage of this mechanism is that the actual amount of resources available either to the grid or to the cloud can be easily and dynamically adapted, simply by changing the number of running grid worker VMs. This allows for greater flexibility and better distribution of resources compared to the standard situation where resources are statically allocated either to the cloud or to the grid and cannot be easily reallocated.

2.1 Physical Clusters

The physical infrastructure of CERIT-SC's virtualized environment consists of six major clusters that vary heavily by means of their size and per-node parameters. The largest cluster is *zapat* (1760 CPU cores, 16 cores and 128 GB RAM per node), followed by *zebra* (960 CPU cores, 40 cores and 256 GB RAM per node), *zegox* (576 CPU cores, 12 cores and 90 GB RAM per node), *zefron* (320 CPU cores, 40 cores and 1 TB RAM per node), *zigur* (256 CPU cores, 8 cores and 128 GB RAM per node) and *zorg* (40 CPU cores, 40 cores and 512 GB RAM per node). The availability of nodes and clusters varied slightly during 2016 as shows Fig. 2.

2.2 Resource Managers

CERIT-SC is using two independent resource managers within its shared infrastructure. First, it is the OpenNebula software stack [19] which is responsible for

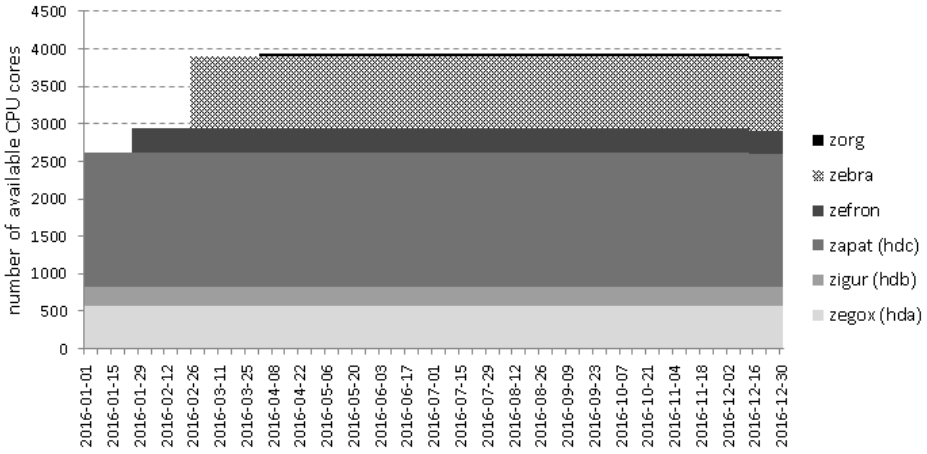

Fig. 2. The number of CPU cores available in CERIT-SC during 2016.

virtualization and VM life-cycle management. Using OpenNebula, both regular users' VMs are deployed as well as those special "grid worker" VMs that serve for running computational batch jobs from our grid-like batch system. For this purpose, we use the Torque resource manager [1] with a custom built advanced job scheduler that has been throughly described at JSSPP 2016 [13]. While every user of our system can run a regular VM, only the system administrator can manipulate with those "grid worker" VMs.

2.3 Operational Constraints and Policies

In our system, most operational constraints are related to the batch scheduling system, i.e., the Torque resource manager. This system is heavily optimized with quite a complex set of system policies and usage constraints. For example, jobs are automatically assigned into system queues, the amount of resources available to a given queue or user is carefully selected while a complex job scheduling approach based on conservative backfilling [20] is used in order to efficiently use the available infrastructure as well as to meet several user-oriented criteria [13]. Importantly, job scheduling is subject to an advanced multi-resource aware fair-sharing mechanism that guarantees that resources are used in a fair fashion with respect to system users. Beside that, additional metrics like expected job slowdown and wait time are also used when prioritizing users' jobs. More details about the system configuration and applied constraints can be found in [15] and [13].

In contrast to the batch system, the cloud-operating OpenNebula framework represents a rather simple environment. We use the default VM scheduler in OpenNebula (`mm_sched`), which uses a simple VM-matching approach. No advanced methods like VM prioritization, fair-sharing or automatic VM migrations/re-scheduling are applied because they are not currently supported.

The exact amount of physical resources that are delegated to the grid-like computations using "grid worker" VMs is selected by the system administrator, i.e., there is no automatic load-balancing feature applied in the system.

3 Workload Description

In this section we describe the main characteristics of the mixed workloads collected during 2016 in CERIT-SC. During this one year period there were 11,382 running VMs. Out of these 11,147 represented normal user cloud VMs while 235 represented "grid worker" VMs. Concerning the grid workload from the Torque resource manager, there were 472,328 jobs computed inside those 235 grid worker VMs during 2016.

3.1 Main Characteristics of Cloud VMs and Grid Jobs

We now proceed to the analysis of the cloud and grid workload in CERIT-SC, starting with some major characteristics of cloud VMs and grid jobs. Fig. 3 shows scatter plots of all cloud VMs (left) and grid jobs (right). A "dot" represents one job/VM and its x and y coordinates represent the number of requested CPU cores and RAM in GB, respectively. Therefore, Fig. 3 allows us to see the differences among those two workloads by means of their resource requirements.

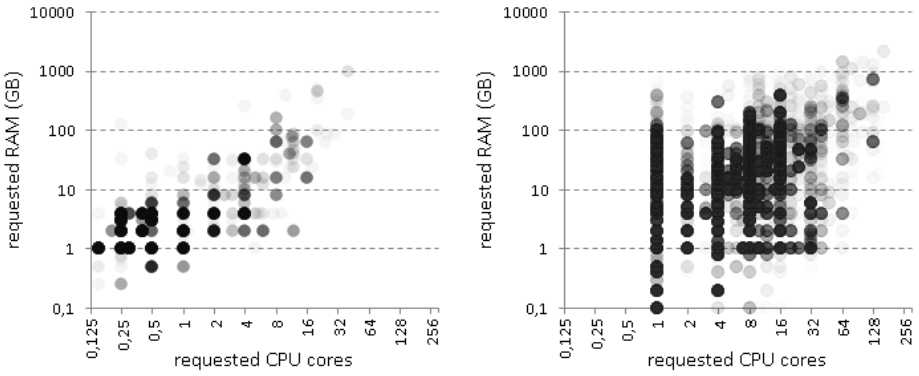

Fig. 3. CPU and RAM requirements for cloud VMs (left) and grid jobs (right).

Clearly, there are large differences among the two workloads. Generally speaking, the grid workload is much more variable, with lots of jobs that request more CPUs and/or larger chunks of RAM. In fact, a large amount of grid jobs is using only few CPUs but lots of available RAM. On the other hand, cloud workloads are less CPU and RAM demanding. Also, VMs frequently request only a fraction of one physical CPU which never happens in the grid. When a VM is submitted

in OpenNebula, its owner can specify the amount of both physical and virtual CPUs (vCPUs). While the minimal vCPU value is 1.0, i.e., the guest OS inside a VM always "sees" at least 1 CPU, the user or the system administrator can specify that this virtual CPU(s) corresponds to a fraction of physical CPU(s), allowing for explicit *CPU overcommitting*.

Fig. 4 shows the actual impact of such overcommitting by comparing the number of allocated CPUs and vCPUs during 2016. Although the average overcommitment of a VM[2] was 3.6, physical CPUs in CERIT-SC were — on average — overcommitted only by the factor of 1.42 during 2016. The latter value is lower since the dominant part of infrastructure's physical CPUs has been utilized by VMs having relatively low vCPUs to CPUs ratio.

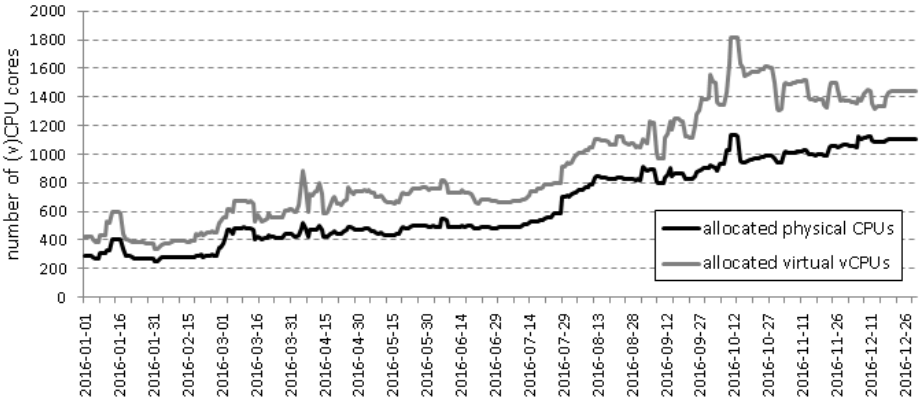

Fig. 4. The level of overcommitment (allocated CPUs vs. corresponding vCPUs).

Another important parameter of the workload is the VM's and job's execution time (duration) and their wait times, i.e., the time before a VM or a job is actually started. For this purpose, we present cumulative distribution functions (CDF) of VM and job durations and wait times. These CDFs are shown in Fig. 5, where the duration is on the left and the wait time is on the right. In the CDF, the y-axis represents the fraction of jobs that have duration less than or equal to a given duration/wait time, which is shown on the x-axis.

Not surprisingly, there are significant differences between the cloud and grid workloads. Concerning execution time, all grid jobs have some upper bound on their maximum execution time, which is specified either by a user or by a default queue limit. This is not the case for cloud VMs whose maximum runtime is unbounded and unknown in general. In CERIT-SC, the maximum allowed runtime for grid jobs is 2 months. In grid, jobs are terminated once reaching their maximum execution time. This causes the "staircase-like" shape of the CDF for grid jobs, where several such "stairs" can be observed (e.g., at 2, 4 and

[2] VM overcommitment factor is computed as vCPUs/CPUs.

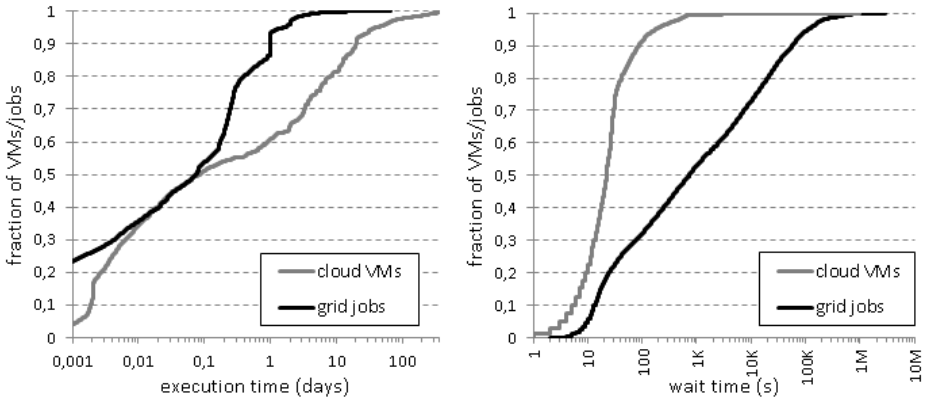

Fig. 5. The CDFs of VM/job execution time (left) and wait time (right).

24 hours) which correlate to the most popular user-provided runtime estimates. In cloud, the runtime is not bounded and (as can be seen in the CDF) there are VMs that ran throughout the whole one year period (i.e., the CDF reaches maximum as x approaches 365 days). Fig. 5 (left) also reveals that several cloud VMs and quite a few grid jobs actually terminated right after their start, most probably due to a misconfiguration or some form of failure.

The CDFs of wait times are shown in Fig. 5 (right) and show major differences between the workloads. In the cloud, 70% of VMs start within 30 seconds and 99% of all VMs are running no later than 10 minutes after their submission. This is natural as cloud is used for more interactive work and system administrators periodically check that there are free resources available for the newly coming VMs. On the other hand, grid job wait times are more spread with many jobs waiting more than 10 minutes (52% of jobs). The nature of batch computations is different from the cloud, as user-to-job interactions are not very common, i.e., it is natural to have a significant backlog of waiting jobs in the system. Therefore, job wait times may be as high as two weeks in extreme cases ($\approx$ 1.2M seconds).

3.2 Infrastructure Utilization by Cloud and Grid Workloads

Let us briefly describe how the system resources were used by these two workloads (cloud VMs vs. "grid worker" VMs). Fig. 6 shows the distribution of available CPU cores to the cloud and grid, respectively. It reveals that the amount of resources available for each particular framework was changing throughout time. Especially, the continuous increase of cloud VMs is nicely visible. While in January 2016 the cloud VMs only required 284 CPUs, by the end of the year their total allocation was over 1,100 CPUs. This was only possible by reducing the allocation for "grid worker" VMs, as shown in Fig. 6.

The figure also reveals some minor inconsistencies — the number of available CPUs is sometimes smaller than the number of allocated CPUs. These anomalies are caused by the imperfect accounting of OpenNebula, which only stores

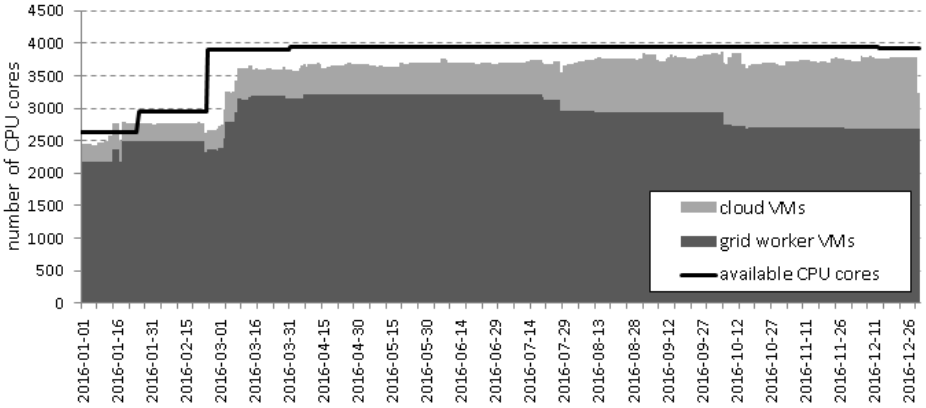

Fig. 6. The number of CPU cores allocated to the cloud VMs and "grid worker" VMs.

timestamps when a VM was running in the past, but it does not record the actual amount of resources used during the VM's "offline" periods. It is therefore impossible to accurately measure the real system resource usage. Unlike a common grid job which has a well defined start and completion time, a VM can be launched and paused repeatedly while (sometimes) still blocking resources during its "offline" period. This is the case for VMs in "suspend/power-off" states while VMs that are "undeployed" or "shutdown" do not consume any resources [21]. Since this VM state-related information is not available, we have approximated the real resource consumption as if all VMs used their resource allocations during their "offline" periods, which leads to the observed minor inconsistencies.

The allocation of free CPU cores by the cloud VMs varies in time and is 72% on average[3]. For better visibility, the actual allocation of "grid worker" VMs by the grid jobs from the Torque batch system is shown in separate Fig. 7, both for CPU (top) and RAM (bottom). As can be seen, the allocation levels vary from high to rather low, which is quite common behavior in grid-like system and relates to several factors including limited job runtime and daily/weekly cycles of user submissions. The average CPU utilization in CERIT-SC's grid partition is 77%. Frequently, CPUs/RAM cannot be used to their full capacity due to the fragmentation of system resources. Simply put, although the overall (theoretical) capacity is sufficient to accommodate further workload, job requirements (e.g., its user-defined topology) cannot be satisfied with the current placement of already running jobs. Fig. 7 also reveals that in CERIT-SC's grid workload the most constraining resource is usually the CPU, while plenty of free RAM is generally available most of the time.

[3] As discussed in Section 2.2, "grid worker" VMs are started/stopped by the system administrator, so the 72% utilization of cloud VMs is computed with respect to the remaining (i.e., available) capacity in the system (see Fig. 6).

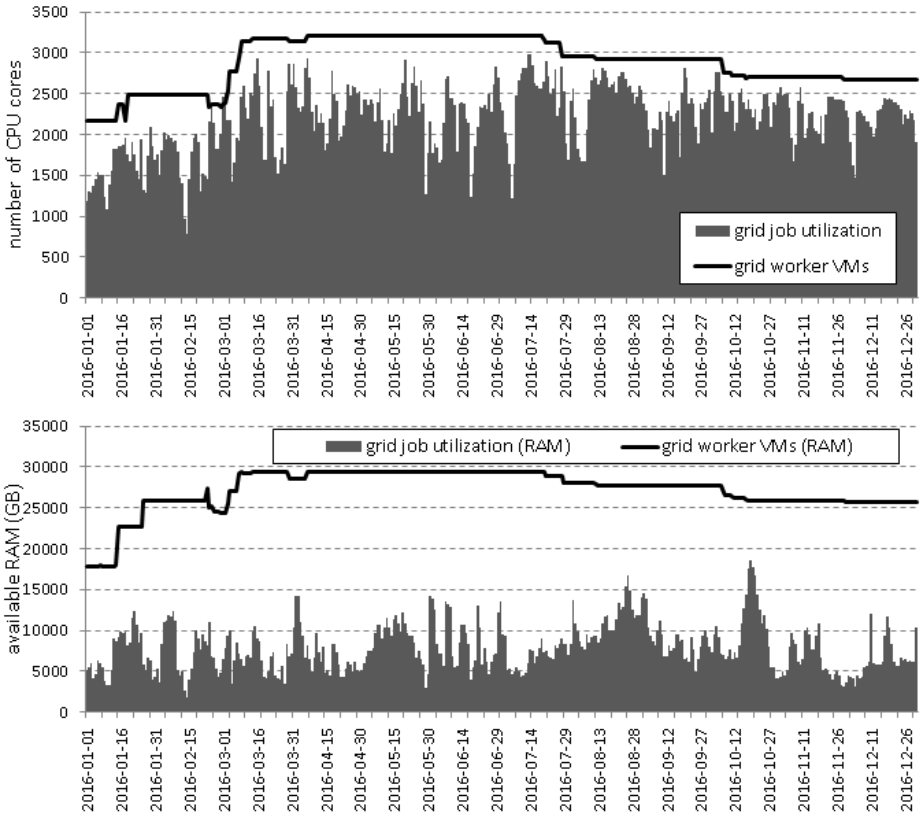

Fig. 7. The allocation of "grid worker" VMs by jobs from the Torque batch system showing CPU (top) and RAM (bottom) allocations.

Unfortunately, the lack of accounting data both in the OpenNebula and the Torque does not allow us to construct similar charts of actual CPU/RAM load neither for the cloud VMs nor the grid jobs. Therefore, we do not know exactly to what extent the resources allocated for cloud VMs and/or grid jobs were actually used throughout the time. For grid jobs, we only know the average CPU load per job. Using this data, we see that the CPU load is usually quite high (78% on average). We have no accounting data concerning real CPU/RAM load for cloud VMs in CERIT-SC system. Fortunately, we have such data from a separate cluster that is fully dedicated for cloud VMs[4]. Fig. 8 shows the actual CPU load in this cluster, suggesting that the CPU load of cloud VMs is typically much smaller than for grid jobs, being only 5–30% in most cases. Although this data come from a different cluster, we expect analogous behavior in CERIT-SC, given the same user-base and similar workload.

[4] It is the *dukan* cluster which is not part of the CERIT-SC infrastructure but it executes similar workloads from the same user-base.

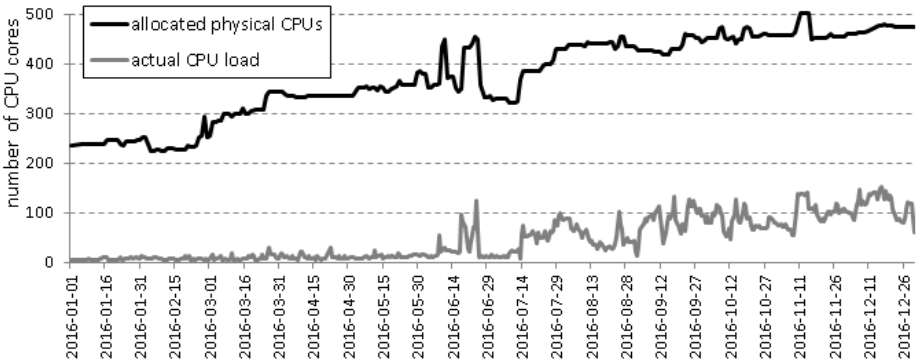

Fig. 8. The actual load of allocated CPUs by cloud VMs in *dukan* cluster during 2016.

3.3 User-Oriented View of the Workloads

So far, the major differences among the workloads have been discussed by focusing on the characteristics of individual VMs and/or jobs. In the following text, we provide another complementary view of the workload, which analyzes the workload on a per-user level. For this purpose, we have prepared stacked charts in Fig. 9 that show the amount of allocated CPU cores per-user during 2016, both for cloud VMs (top) and grid jobs (bottom).

Fig. 9 complements our previous observations concerning typical VM and job durations in Section 3.1. In the cloud environment, the majority of CPU cores is consumed by $\sim$ 15 users with long running (continuous) workloads. In the grid, the situation is exactly opposite. Most CPU cores are consumed by many different users that execute rather time-constrained workloads, which is natural since every grid job has a firm runtime limit. This strict limitation combined with typical day-to-day and weekly cycles makes the overall utilization curve of the grid workload much more "noisy".

3.4 Maintenance Periods

The final part of this section focuses on maintenance periods in CERIT-SC system. It is fair to say that the following description and presented data is not based on some advanced monitoring tool, instead we only rely on the system administrator's log of planned upgrades/repairs. Minor offline periods and minor unplanned failures may not be captured here, however all major interventions and downtimes are likely documented in this log[5].

Using this data we were able to reconstruct major system upgrades and maintenance periods that are shown in Fig. 10. As the chart shows, the overall number of offline CPUs is quite low during the year ($\sim$ 1.3% of available CPUs), except for one event at the beginning of May, where a total of 416 CPUs (28

[5] This log is available at: `https://github.com/CERIT-SC/cerit-maintenance`

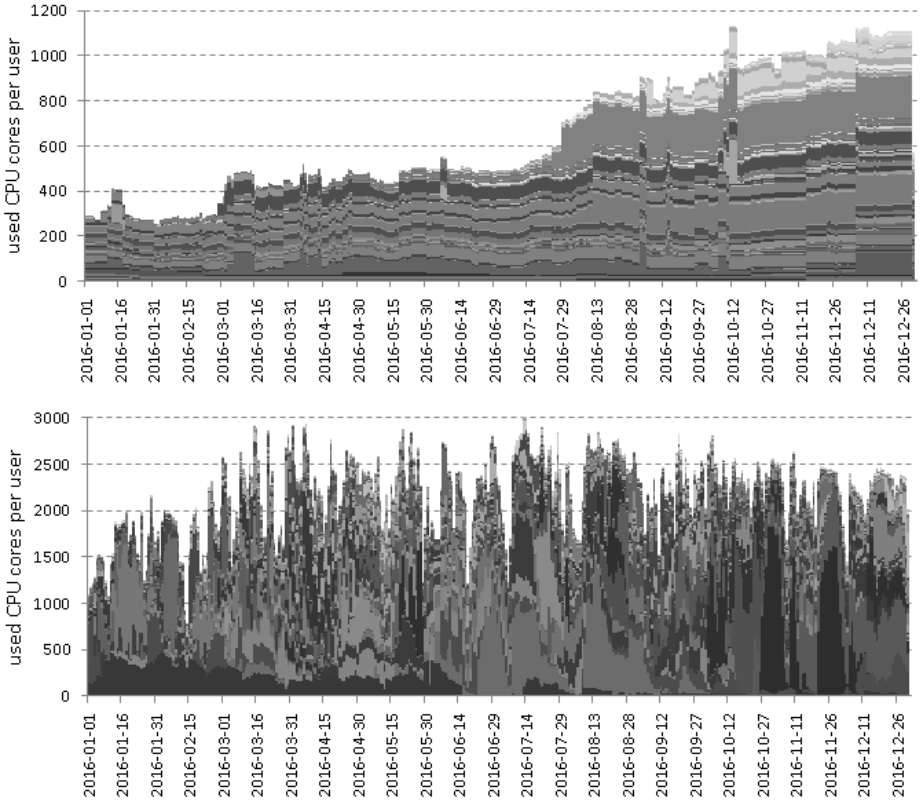

Fig. 9. Users' CPU usage over the time for cloud VMs (top) and grid jobs (bottom).

nodes) was switched off for a week due to the planned HDD firmware upgrade. This was the only case, when the total number of offline CPUs exceeded 500, causing 13.3% of the infrastructure being unusable for a week.

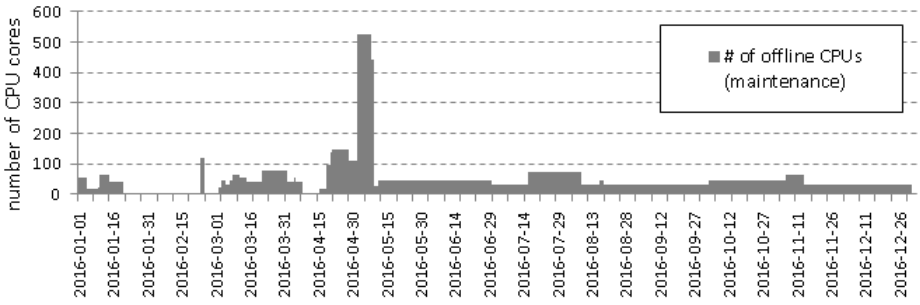

Fig. 10. Number of offline CPUs undergoing maintenance during 2016.

4 Optimization Criteria used in the System

Those two systems operating over the CERIT-SC infrastructure have different goals, i.e., different sets of optimization criteria. In both the cloud and the grid partition, these goals are only enforced in a "best-effort" manner, i.e., no formal Service Level Agreements are established, although there are various Service Level Indicators/Objectives [10] that are optimized/targeted.

We now describe these major goals (i.e., objectives and their indicators) in order to allow other researchers to use our workloads to analyze new/existing scheduling approaches using realistic optimization goals. We would like to emphasize that it is not a good idea to compare simulation-based results with those of the original workload, since there are many unknown constraints that influence the quality of the original schedule (as seen in the original workload). Typically, simulation-based results tend to show much better results than those original (real) schedulers. In reality, real schedulers employ various considerations that limit their options, and lead to sub-optimal scheduling [16].

Sadly, many of these "considerations" cannot be reconstructed as they are not recorded anywhere[6]. For more details please refer to [16] and [14]. Let us start with the grid-related criteria system before proceeding to the cloud-related and global criteria.

4.1 Grid Optimization Criteria

The grid partition is subject to four major optimization criteria, which we describe in the following text.

First, we aim to minimize the *avg. wait time* [3] which is the mean time that jobs spend waiting before their executions start. Second, the *avg. bounded slowdown* [5] is minimized, which is the mean of jobs bounded slowdowns. The bounded slowdown is the ratio of the actual response time of the job (the time from its submission to its termination) to the response time if executed without any waiting. To avoid huge slowdowns of extremely short jobs, the minimal job runtime is bounded by some predefined time constant (e.g., 10 second), sometimes called a "threshold of interactivity" [5].

We also focus on *resource utilization*, i.e., the goal is to minimize the number of idle CPUs throughout the time. The system does not use any "green goals", i.e., no methods to reduce the carbon footprint/energy consumption are used or even planned to be used in the future.

Last but not least, *user-to-user fairness* is considered as one of the most important goals in CERIT-SC grid partition and is managed by a fair-sharing approach that uses so-called *Normalized User Wait Time (NUWT)* metric [13]. For a given user, NUWT is the total user wait time divided by the amount

[6] For example, real schedulers must limit the number of concurrently running licensed applications (jobs using licensed SW) with respect to the number of available software licenses, i.e., even if resources are free some jobs must wait until a license is available. Such information is not usually recorded in the workload.

of previously consumed system resources by that user. Then, the user-to-user fairness is optimized by minimizing the mean and the standard deviation of all NUWT values. It follows the classical fair-share principles, i.e., a user with lower resource usage and/or higher total wait time gets higher priority over more active users and vice versa [8]. The calculation of NUWT reflects consumptions of multiple resources (CPU and RAM utilization), representing a solution suitable for systems having heterogeneous workloads and/or infrastructures.

4.2 Cloud Optimization Criteria

Just like in the grid, there are no "green" objectives and no formal SLAs and the service is provided in a best-effort fashion. While the grid partition have quite robust optimization goals, the cloud partition (currently) uses only two simple objectives and further criteria are likely to be included/developed in the future.

First, the goal is to *minimize the wait time* of (newly submitted) pending VMs, thus maximizing the number of concurrently running VMs. In other words, the goal is to *minimize the number of VMs that have to wait* for their deployment. This two criteria (number of waiting VMs and their wait time) are currently used by the system administrators when (de)allocating physical nodes for the cloud partition. The "algorithm" is to keep a decent part of the cloud nodes free for newly arriving VMs. The utilization criterion is used as an auxiliary indicator, i.e., low utilization implies that some nodes should be returned to the grid and vice versa. No further indicators/objectives are currently (actively) measured and optimized/enforced.

A notable difference with respect to the grid partition is the (current) absence of any fairness-related objective and/or technique to enforce fair use of resources which is becoming an apparent problem. This is a general problem in those private clouds that do not use the "pay-per-use" model (or some other equivalent of money/credit). Further details are provided in the following Section 5 which discusses open scheduling problems in CERIT-SC.

4.3 Global Criteria

Except for the overall CPU utilization (total allocated CPUs throughout the time), there are currently no global criteria used to measure the performance of the whole system, i.e., how well the cloud and grid partitions co-exist together.

5 Open Problems

Although the CERIT-SC system is production-grade and currently operates without any major problems, there are still several unresolved (scheduling) problems that must be addressed in the (near) future. In the following text we summarize these problems and their origins.

5.1 Advanced VM-packing and Adaptive Re-scheduling

In the cloud environment, we would like to improve the quality of VM scheduling. Especially, we want to investigate whether VMs can be efficiently "packed" on the physical nodes such that their combined resource requests (e.g., CPU, RAM, disk space, disk I/O and network I/O) are reasonably balanced *throughout the time*. Here we are facing the limits of the current VM scheduler used in the OpenNebula framework which does not provide any intelligent VM (re)scheduling heuristic that would adapt VMs allocations in time. By default, it only schedules VMs upon their deployment based on their predefined resource requests. No further optimization — based on an actual performance of a running VM — is done during a VM lifetime.

Clearly, this leaves an open space for improvements as currently some nodes may be occupied by idle VMs (resource wasting) while other nodes may be overloaded with VMs competing for resources such as CPUs, I/O, etc. As was demonstrated in Fig. 8, there seem to be many opportunities how to improve, e.g., the CPU load. For example, the scheduler should be able to dynamically reschedule idle VMs, possibly migrating them to heavily overbooked nodes, thus freeing their original hosts for more demanding or new (pending) VMs. Similarly, when a node becomes overloaded by its VMs, some of them should be rescheduled/migrated to decrease the host's contention.

In order to actually enable such functionality one must however not only develop a new advanced scheduler but also invest in the underlying infrastructure to allow such live VM migrations. In this case, a dedicated distributed storage facility (e.g., the *Ceph* [23]) is needed to allow for live migrations, which is not currently fully operational in CERIT-SC.

5.2 Resource Reclaiming

As discussed in Section 2, the major benefit of resource virtualization in CERIT-SC is that the actual amount of resources available either to the grid or to the cloud can be easily and dynamically adapted, simply by changing the number of running "grid worker" VMs. In practice however, this mechanism does not work that easily due to the nature of our cloud workload. In fact, it works flawlessly when more resources are required for the cloud VMs. In that case, several "grid worker" nodes are first drained, i.e., all grid jobs running inside the worker are completed and new jobs are not allowed to start there. Then the given "grid worker" VM is terminated and its host becomes available to the classic cloud VMs. The problem is that the same mechanism does not work so easily in the opposite direction, i.e., it is not always easy to drain a host that is hosting running cloud VMs. As discussed in Section 3.1, in CERIT-SC the runtime of a VM is unbounded and unknown, in general. Therefore it is impossible to drain a node by the same mechanism that works in the grid. Theoretically, VMs that execute on a node can be migrated, but this may not be always possible, e.g., when the cloud infrastructure is already saturated by running VMs.

This is currently our major threat, since the cloud allocations in our system are increasing very quickly (see Fig. 9 (top)), yet we do not have any "automated" resource-reclaiming mechanism. Another problem is that CERIT-SC *provides its resources for free* to anyone who is affiliated with the scientific/academia community in the Czech Republic (university students/teachers, academic researchers, etc.). Therefore, it does not use some form of the "pay-per-use" model which is otherwise very suitable to motivate users to stop their VMs once they are not needed anymore. At the same time, CERIT-SC's budget is fixed, i.e., we cannot just buy another cluster whenever the demand is approaching the available capacity.

5.3 Fair-Sharing in Cloud

The absence of the "pay-per-use" model together with the rather poor resource-reclaiming in our cloud brings another problem — the resources are allocated to the users without considering some overall fairness. This is in great contrast with the grid installation, where fairness is one of the major optimization goals and is managed by the fair-sharing approach [13]. As was shown in Fig. 9 (top), the majority of cloud resources is consumed by few users over a long time period, yet there is no automated mechanism that would force them to decrease their allocations, letting other users to use the system.

Apparently, we should adopt some analogy of the fair-sharing in our cloud installation. Perhaps a good starting point would be to prioritize users (based on their resource usage) and automatically decrease allocations for long running VMs of low priority users (i.e., increase VMs overcommitment factor). Solving this problem will however require major changes in the current, rather naive, scheduler used in the OpenNebula framework.

5.4 Load-Balancing

Upon solving the problems mentioned in Sections 5.1–5.3 we start to focus on the global scheduling problems such as load-balancing among the applications/frameworks. Certainly, our workloads indicate that there are many opportunities to use temporarily idle resources. For example, short jobs from the grid can probably "steal the cycles" when cloud VMs are idle, because the average load of allocated CPU cores in the cloud is currently bellow 30% in most cases. For this purpose a "sleeping grid worker" VM could be launched on every host with a huge overcommitment factor during its idle phase. Then, upon the request of the batch system, such a sleeping grid worker VM can be woken up by increasing its resource allocations and used for (short) grid jobs. In this way, many short and/or narrow grid jobs can use even very time-limited opportunities, e.g., when cloud VMs are idle during the night.

The question is to how to actually implement such an inter-application scheduling. One way is to try to build upon existing frameworks like Apache Mesos [6], or use some form of a module in the existing underlying platform — in

this case inside the OpenNebula SW stack. The decision process is further complicated by the fact that new frameworks like OpenStack [9] and/or Docker [17] are currently tested and will be probably offered in the near future to our users. Therefore we must not only focus on the "optimal result" but also keep in mind that our development team has a limited capacity. Thus, every new framework and/or functionality then brings not only opportunities but problems too.

6 Workload Formatting and Conclusion

In this paper we have provided a detailed analysis of the mixed workload traces from the CERIT-SC system. Presented workloads can be freely obtained at the JSSPP's public workload archive [12]. Since the workload logs come from two different systems, they are provided in two different formats.

The grid-based workload is formatted according to the well-know Standard Workload Format (SWF) which is adopted in the Parallel Workloads Archive [4]. SWF formatting rules are described at the Parallel Workloads Archive's website[7]. Additional information that are not supported by the original SWF format but may be useful for some simulations are added at the end of each job entry (i.e., at the end of a line) into newly defined fields and are separated by whitespaces. These additional fields include the specifications of used computing node(s) (i.e., hostname(s)), human-readable name of the queue, so-called "nodespec" (i.e., detailed job description taken from the `qsub` command, including per-node specified number of requested resources) and the number of requested GPUs.

The cloud-based workload describing all VMs is formatted in a JSON (JavaScript Object Notation) format [11]. Unlike the jobs in the grid workload that are non-preemptible, cloud VMs can be executed repeatedly and thus require more complex data structure to describe their (repeated) executions. For this purpose the array feature of JSON is very suitable.

Although there are many SWF and JSON parsers available [4, 11], we provide a set of simple java parsers that can be used to parse the aforementioned workload traces. Moreover, they calculate and print several statistics that were used in this paper. These parsers are provided along with the workloads at the JSSPP's workload archive [12].

Acknowledgments. We kindly acknowledge the support and computational resources provided by the MetaCentrum under the program LM2015042 and the CERIT Scientific Cloud under the program LM2015085, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" and the project Reg. No. CZ.02.1.01/0.0/0.0/16_013/0001797 cofunded by the Ministry of Education, Youth and Sports of the Czech Republic. We also highly appreciate the access to CERIT Scientific Cloud workload traces.

[7] SWF format: `http://www.cs.huji.ac.il/labs/parallel/workload/swf.html`

References

1. Adaptive Computing Enterprises, Inc. *Torque 6.1.0 Administrator Guide*, February 2017. `http://docs.adaptivecomputing.com`.
2. CERIT Scientific Cloud, February 2017. `http://www.cerit-sc.cz`.
3. C. Ernemann, V. Hamscher, and R. Yahyapour. Benefits of global Grid computing for job scheduling. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 374–379. IEEE, 2004.
4. D. G. Feitelson. Parallel workloads archive, February 2017. `http://www.cs.huji.ac.il/labs/parallel/workload/`.
5. D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1291 of *LNCS*, pages 1–34. Springer Verlag, 1997.
6. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 295–308, Berkeley, CA, USA, 2011. USENIX Association.
7. A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema. The Grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.
8. D. Jackson, Q. Snell, and M. Clement. Core algorithms of the Maui scheduler. In D. G. Feitelson and L. Rudolph, editors, *Job Sched. Strategies for Paral. Proc.*, volume 2221 of *LNCS*, pages 87–102. Springer, 2001.
9. K. Jackson. *OpenStack Cloud Computing Cookbook*. Packt Publishing, 2012.
10. C. Jones, J. Wilkes, N. Murphy, C. Smith, and B. Beyer. Service level objectives. In B. Beyer, C. Jones, J. Petoff, and N. Murphy, editors, *Site Reliability Engineering: How Google Runs Production Systems*, chapter 4. O'Reilly Media, 2016. `https://landing.google.com/sre/book.html`.
11. Introducing JSON, February 2017. `http://www.json.org/`.
12. D. Klusáček. Workload traces from CERIT Scientific Cloud, February 2017. `http://jsspp.org/workload/`.
13. D. Klusáček and V. Chlumský. Planning and metaheuristic optimization in production job scheduler. In *Job Scheduling Strategies for Parallel Processing*, volume 10353 of *LNCS*. Springer, 2017. To appear.
14. D. Klusáček and Š. Tóth. On interactions among scheduling policies: Finding efficient queue setup using high-resolution simulations. In F. Silva, I. Dutra, and V. S. Costa, editors, *Euro-Par 2014*, volume 8632 of *LNCS*, pages 138–149. Springer, 2014.
15. D. Klusáček, Šimon Tóth, and G. Podolníková. Real-life experience with major reconfiguration of job scheduling system. In *Job Scheduling Strategies for Parallel Processing*, volume 10353 of *LNCS*. Springer, 2017. To appear.
16. D. Krakov and D. G. Feitelson. High-resolution analysis of parallel job workloads. In W. Cirne, N. Desai, E. Frachtenberg, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 178–195. Springer, 2013.
17. D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239), March 2014.
18. MetaCentrum, February 2017. `http://www.metacentrum.cz/`.

19. R. S. Montero, I. M. Llorente, and D. Milojii. OpenNebula: A cloud management tool. *IEEE Internet Computing*, 15(2):11–14, 2011.
20. A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.
21. Managing virtual machines, February 2017. `https://archives.opennebula.org/documentation:rel4.4:vm_guide_2`.
22. C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2014-11-17 for version 2.1. Posted at `https://github.com/google/cluster-data`.
23. K. Singh. *Ceph Cookbook*. Packt Publishing, 2016.
24. SWIM workload repository, February 2017. `https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository`.
25. R. Wolski and J. Brevik. Using parametric models to represent private cloud workloads. *IEEE Transactions on Services Computing*, 7(4):714–725, 2014.