Scheduler hierarchies to aid peta-scale cloud simulations with DISSECT-CF

Gabor Kecskemeti

Laboratory of Parallel and Distributed Systems at the Institute for Computer Science and Control
of the Hungarian Academy of Sciences (MTA SZTAKI), Kende u. 13-17, Budapest 1111, Hungary
kecskemeti.gabor@sztaki.mta.hu

Abstract

IaaS cloud simulators are frequently used for evaluating new scheduling practices. Unfortunately, most of these simulators scarcely allow the evaluation of larger-scale cloud infrastructures (i.e., with physical machine counts over a few thousand). Thus, they are seldom applicable for evaluating infrastructures available in commercial cloud settings (e.g., users mostly do not wait for simulations to complete in such settings). DISSECT-CF was shown to be better scaling than several other simulators, but peta-scale infrastructures with often millions of CPU cores were out of scope for DISSECT-CF as well. This paper reveals a hierarchical scheduler extension of DISSECT-CF that not only allows its users to evaluate peta-scale infrastructure behaviour, but also opens possibilities for analysing new multi-cloud scheduling techniques. The paper then analyses the performance of the extended simulator through large-scale synthetic workloads and compares its performance to DISSECT-CF's past behaviour. Based on the analysis, the paper concludes with recommended simulation setups that will allow the evaluation of new schedulers for peta-scale clouds in a timely fashion (e.g., within minutes).

Keywords Cloud computing, simulator, petascale, hierachical scheduling

I. Introduction

Cloud computing infrastructures [1] have rapidly developed into a commodity. Based on virtualisation technologies, they offer simple and straightforward management capabilities for virtual infrastructures. As a result, users of Infrastructure as a Service (IaaS) clouds can more rapidly respond to their ever changing demand patterns, while they do not have to face the everyday issues that would arise with the maintenance of physical infrastructures. Because of the many benefits of IaaS clouds, their adoption has became widespread.

Unfortunately, this widespread use limits research on the internal behaviour of IaaS clouds. To overcome these limits, researchers often turn to cloud simulators to analyse their new ideas [2]. These simulators allow rapid evaluation of many new scenarios; however, they frequently have scaling issues of their own. Thus, they restrain those scenarios that can be evaluated with them. And even in cases when they scale well for the larger-scale simulation needs of recent cloud infrastructures, they are too specialised for general research (e.g., they do not allow simultaneous evaluation of both cloud internals and user side behaviour).

DISSECT-CF (DIScrete event baSed Energy Consumption simulaTor for Clouds and Federations [3]) was proposed as a general purpose, compact, highly customisable open source cloud simulator with special focus on the internal organisation and behaviour of IaaS systems. Compared to other state-of-the-art cloud simulators its performance is already amongst the best. However, even DISSECT-CF has scaling issues when it needs to simulate such large-scale computing infrastructures as the front entries in the top500 supercomputers list[1].

This paper analyses past DISSECT-CF behaviour when simulating infrastructures similar to the ones listed amongst the top500 supercomputers. Based on

[1] http://top500.org

the analysis, the paper concludes that the large amount of physical machines (handled by a single virtual machine – VM – placement technique) cause the scaling issues in the past simulator. Therefore, this paper proposes a generic technique to organise scheduler hierarchies in DISSECT-CF. These hierarchies allow the reduction of the number of machines handled by a single VM placement technique. In order to overcome the inefficiencies that could be caused by the newly introduced hierarchies, the simulator now introduces several ways for interacting between the various levels of the scheduler hierarchy: (*i*) automated high-level VM request revocation, (*ii*) VM request rejection, (*iii*) automated hierarchy setup and (*iv*) VM request propagation though cloud boundaries.

Although the introduced hierarchies are good to increase the scalability of the simulator and allow the evaluation of larger-scale systems, the newly proposed technique is still limited by several factors: (*i*) it cannot support nodes with mixed accelerator-CPU constructs – accelerator-CPU interactions cannot be handled with the new hierarchical model because the new model is limited to a hierarchy of a single kind of resource (because of a limitation in DISSECT-CF)–, (*ii*) similarly, inhomogeneous multi-cloud systems are still out of scope, (*iii*) the automated hierarchy setup is dependent on the kind of simulated workload – with improper hierarchy setup, the simulation might still face scalability issues–, and finally (*iv*) the relation between the actual layout of the simulated infrastructure and the real one can become very detached (the automatically introduced hierarchies usually have different layout than the actual racks, clusters and data centres in an IaaS).

The paper concludes with the analysis of the new hierarchical scheduling. Using synthetic traces a comparison is shown between the past and current simulator with infrastructures ranging from a few thousand to almost two million CPU cores. The behaviour of the extended simulator is also compared to CloudSim [4], revealing that DISSECT-CF has a performance advantage between 5-136× over CloudSim. Even compared to its past self, the new DISSECT-CF performs significantly better and its hierarchical scheduling mechanism could provide up to four-fold performance increase in smaller-scale infrastructure simulations, and a performance improvement of over 92× is observable for large-scale simulations.

The rest of the paper is structured as follows: the paper continues with studying state-of-the-art simulators to reveal their problems. Then, in Section III, the paper introduces a new hierarchical VM scheduling technique for DISSECT-CF. Next, the paper presents a performance evaluation for the improved simulator in Section IV. Finally, the paper provides its conclusive thoughts in Section V.

II. Related Works

CloudSim [4] is amongst the most popular IaaS cloud simulators. It introduces the simulation of virtualised data centres mostly focusing on computational intensive tasks and data interchanges between data centres. An extension called NetworkCloudSim [5], improved its support for in-data-centre network communications. There is also an extension that simulates the energy consumption of the physical machines in a data centre based on specpower benchmarks [6]. CloudSim also ignited an ecosystem around it adding performance improvements, inter-cloud operations and GUI wrappers for teaching [7, 8, 9, 10]. Despite its widespread use and its healthy ecosystem, research done with CloudSim is mostly limited to clouds with a few thousand CPU cores. This limitation severely affects the applicability of the results of CloudSim based simulations.

The SimGrid framework [11] is another widely used simulator for analysing distributed systems (e.g., grids, peer-to-peer systems). This simulator is not focused on clouds and only includes constructs to support virtualisation (e.g., hypervisors and live migration – [12]). Unfortunately, the lack of higher-level cloud related constructs reduces the applicability of SimGrid in most cloud simulation scenarios. Its users would need significant expertise in every cloud management issue so they can build and evaluate complete cloud-like scenarios.

Next, an analysis of GroudSim, which is a simulator developed at the University of Innsbruck [13, 14], was performed. This simulator aims at runtime performance, while it also integrates with the ASKALON workflow system. Until recently this simulator followed a black box model (i.e., it did not simulate any internal details of the cloud management behaviour). Nowadays, GroudSim incorporates the DISSECT-CF simulator to enable the simulation of internal IaaS behaviour [15]. However, the complex cross-simulation synchronisation and workflow orientation of GroudSim makes it less scalable than DISSECT-CF alone.

While the previously mentioned simulators were heavily influenced by past simulators of grids and/or distributed systems, for performance reasons, they also made compromises on the simulation of networking functionalities. Such issues are resolved by simulators like iCanCloud [16] and GreenCloud [17]. These are built on network simulators (e.g., OMNeT++ or NS2) to most accurately simulate network communications in cloud systems. Other than their networking improvements, GreenCloud also offers precise energy estimates for networking and computing components, while iCanCloud is also user oriented and thus offers support in the decision making regarding the use of IaaS systems [18]. As these simulators are network oriented, their use cases are different from the rest of the simulators discussed in this section (e.g., they are mostly used to evaluate localised phenomena thus their scaling capabilities are not relevant).

Finally, there are some specialised simulators like xSim [19] and SimMatrix [20]. These simulators are proven to perform well for large-scale systems, but their scope is limited. For example xSim is focusing on the analysis of MPI workloads in exa-scale systems, while SimMatrix is focused on many task computing. Because of their over-specialisation these simulators are not suitable for analysing general problems in large-scale systems.

II.1 Problem Statement

After analysing the prior art, it can be concluded that existing simulators have many drawbacks for those planning to investigate scheduling in large-scale IaaS systems (e.g., they do not provide foundations for constructing scheduling hierarchies instead they expect their users to construct the hierarchies on their own). To fulfil the needs of such scheduling scenarios, the rest of this paper reveals a new hierarchical virtual machine scheduling technique to be applied by the DISSECT-CF simulator. With the use of this technique researchers will have better insights on infrastructure behaviour even if significantly larger-scale systems are simulated than it was previously possible by past simulators.

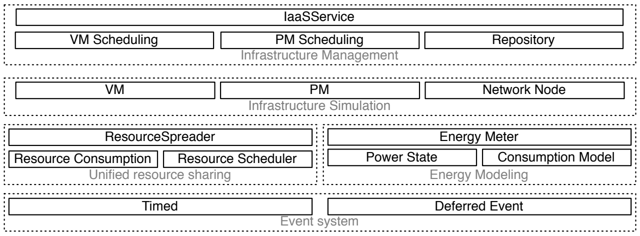

Figure 1: The architecture of DISSECT-CF

III. Generic Hierarchical Scheduling

III.1 Overview

DISSECT-CF [3] is a compact, customisable open-source simulator with focus on the internal organisation and behaviour of IaaS systems. Figure 1 presents its architecture. The figure groups the major components with dashed lines into subsystems. Each subsystem is implemented as independently from the others as possible. There are five major subsystems each responsible for a particular aspect of internal IaaS functionality: (i) event system – for a primary time reference; (ii) unified resource sharing – to resolve low-level resource bottleneck situations; (iii) energy modelling – for the analysis of energy-usage patterns of individual resources (e.g., network links, CPUs) or their aggregations; (iv) infrastructure simulation – to model physical and virtual machines as well as networked entities; and finally (v) infrastructure management – to provide a real life cloud like API and encapsulate cloud level scheduling (the target of this paper's improvements).

III.2 Scaling Bottleneck

Although the simulator was designed from the beginning with high scalability in mind, the performance of its VM placement mechanisms is dependent on the number of physical machines registered at an IaaS service. Thus, to reduce this bottleneck, but to allow simulator developers to still design simple schedulers, a new hierarchical solution was needed. This new solution must allow the old scheduling mechanisms to work without even knowing that they only play a small role in a large-scale simulation, but the simulator's users should also have a chance to alter the hierarchy and the way higher-level schedulers interact with the original scheduling mechanisms.

III.3 Hierarchical Scheduling

III.3.1 Generic Scheduling

The new hierarchical scheduler concept of DISSECT-CF is built around the following functionalities: (*i*) schedulers should be able to propagate VM requests amongst each others, (*ii*) higher-level schedulers should be able to cancel requests, (*iii*) independently from the level of the scheduler, the same scheduling interface must be used (allowing to implement even cross-cloud scheduling mechanisms). In the following paragraphs, these functionalities are detailed.

VM propagation The hierarchical scheduler interface is expected to be implemented by every entity in the system who is planning to accept VM requests (ranging from Physical Machines, to low-level VM schedulers to even IaaS systems). This interface contains the following operations: (*i*) VM request, (*ii*) VM termination, (*iii*) VM migration, (*iv*) VM request cancellation and (*v*) VM resource reallocation. With the "standardised" interface it is not only possible to migrate VMs across various VM managers but also possible to create federations of multiple IaaS systems. In the new hierarchy, VM requests are propagated until they reach a physical machine that can serve them. If there are no physical machine that can serve a request in a low-level scheduler (one that directly interacts with physical machines), then the scheduler is allowed to queue the VM request. If the VM request is handled by an entity that does not directly interact with physical machines, then based on a scheduling policy it must decide to which lower-level scheduler it propagates the VM request (it cannot queue the request on its own). The selection of the lower-level scheduler can be accomplished by a range of techniques. The simulator currently delivers a random, a round robin and a weighted probabilistic scheduler selection technique (where VM requests are propagated to lower level schedulers that are less likely to queue them).

Automated request cancellation and resubmission As with many hierarchical schedulers in the past, it could frequently happen that one of the low-level schedulers gets overloaded with VM requests while others are left with very little work to do (this is a likely scenario with weighted probabilistic techniques). In order to automatically avoid bottlenecks, DISSECT-CF contains an automated request cancellation and resubmission technique. Upon submitting a VM request, higher-level schedulers will tag the request with an expiration time. This tag will be a timestamp after which time the lower-level scheduler is not allowed to serve the VM request, instead the request should be sent back to the higher-level scheduler who sent it. This tagging mechanism allows the low-level schedulers to prioritise the almost expired requests (as a measure of keeping service level objectives). Also, the high-level schedulers could penalise those lower-level schedulers that did not succeed in completing their designated VM requests within the expected period of time. In order to refrain VM requests from getting cancelled too frequently, the high-level schedulers analyse the VM throughput of each of their underlying schedulers and they set up VM request termination times so requests will expire with a small likelihood (in the current simulation setup, the resubmission rate is set up to be around a single request out of every thousand).

VM request rejection As VM requests received by lower-level schedulers are tagged with their expiration times, the low-level schedulers can decide if they are willing to queue such requests. When a tagged request arrives, the scheduler will automatically reject the request if its queue is significantly longer than the queues of other schedulers with similarly sized managed infrastructures.

III.3.2 Automated Hierarchy Formation

The simulator allows the precise definition of the hierarchy of the schedulers during the construction of

IaaSServices, ensuring that it matches the real life hierarchies set up in large scale cloud systems. However, to allow investigations about the effect of different hierarchies, this definition is not obligatory. If the user prefers, the hierarchy can be automatically constructed. The automatic hierarchies can even evolve during the entire runtime of a simulation (allowing the user to investigate several hierarchy adaptation scenarios and their effect on runtime). Also, the simulator can save an automatically determined hierarchy for later use, so users will have a chance to reuse previously efficiently working hierarchies (this option also allows users to utilise the auto constructed hierarchies outside of the simulated world and check the correctness of their simulations in real life).

The simulator allows the creation of the following kinds of schedulers (all these schedulers are also exemplified in Figure 2):

- *Regular schedulers.* These schedulers have an IaaSService as their parents and they manage physical machines directly. These are the original kinds of schedulers possible in the simulator.
- *Low-level schedulers.* They manage physical machines directly, but their parents are not IaaSServices. Instead they are operating with a high-level scheduler.
- *High-level schedulers.* They do not directly operate with physical machines, they handle the VM requests as discussed in the previous sub-section. They can be classified into two subtypes:
  - *Mid-level schedulers.* They are placed in the middle of the hierarchy, they forward VM requests from their parents to their directly managed schedulers. They can either manage both high and low-level schedulers.
  - *Ingress schedulers.* They have an IaaSService as their parents, and they can act as either mid-level schedulers (if acting as an entry to a hierarchy) or as regular schedulers (if no hierarchy is needed).

Creating scheduler profiles. Before running the simulator with automated scheduling hierarchy management for realistic workloads, the hierarchy creation technique needs a profile for the user provided low-level schedulers (for the schedulers integrated in DISSECT-CF these profiles are already done). The profile creation starts with the specification of the total number of CPU cores – C_{max} – the simulated infrastructures should have during the profiling session. Next, several IaaSServices are created with varying number of CPUs in each physical machine – $c_{pm} \in C_{pm}$, where $C_{pm} = \{\forall x : (x \in \mathbb{N}) \wedge (1 < x < C_{max})\}$ – ensuring that the complete size of the infrastructure is matching the previously given total (e.g., with 8 CPU physical machines the number of physical machines should be $C_{max}/8$). Then, on each IaaSService the same randomly generated VM utilisation trace is executed (the trace's properties can be user defined). The execution time – $t_{ex} : C_{pm} \rightarrow \mathbb{R}$ – is recorded for all cases and they are written to the profile for the user provided low-level scheduler (examples of such profiles can be seen in Figures 4 and 5). The profiling executed on this way to ensure that the simulated infrastructure's size does not unnecessarily prolong the profiling period, and instead, only the PM count related scaling properties of the low-level scheduler do.

From the collected profile, the automated hierarchy creator determines the optimal amount of physical machines:

$$\begin{aligned} pm_{opt} &:= C_{max}/c_{opt} \qquad (1) \\ \text{where} \quad t_{ex}(c_{opt}) &= \min_{\forall(c_{pm})}(t_{ex}(c_{pm})) \end{aligned}$$

The optimal amount of machines is calculated as the ratio between the maximum amount of CPU cores used during the profiling – C_{max} – and the specific number of CPU cores per machine – c_{opt} – which resulted in the smallest profiling execution time. Each low-level scheduler has a specific optimum value. Users can also specify an operationally acceptable pm count range for their scheduler relative to the pm_{opt} value. The range is specified with $\tau \in]0..1]$ which shows the difference allowed in the percentage of the execution time entries in the profile compared to the minimum execution time recorded for pm_{opt} (i.e., $t_{ex}(c_{opt})$). Thus:

$$C_{acc} := \{\forall c_x : c_x \in C_{pm} \wedge |1 - \frac{t_{ex}(c_x)}{t_{ex}(c_{opt})}| < \tau\} \tag{2}$$

$$PM_{acc} := \{C_{max}/c_x : c_x \in C_{acc}\} \tag{3}$$

Where the set of C_{acc} defines all the CPU core num-

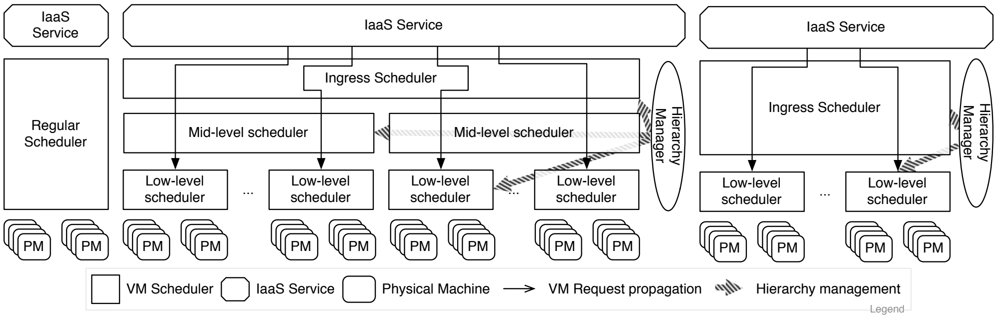

Figure 2: The kinds of scheduling hierarchies and schedulers possible in the new extended simulator

bers that resulted in profiling execution times within the user specified range. While PM_{acc} is the set of PM numbers for which the profiling suggests acceptable scheduling performance according to the user specified range. The hierarchy creator will use only the minimum $pm_l = \min PM_{acc}$ and maximum $pm_h = \max PM_{acc}$ values from the PM_{acc} set to determine under what conditions it does not need to alter the previously constructed hierarchy.

Managing scheduler hierarchies using profiling results. When a new PM is registered in the infrastructure under the control of a high-level scheduler, the hierarchy manager will register the PM in a round robin fashion to the underlying schedulers. Once the supervised PM count for a particular low-level scheduler is reached (i.e., it reaches over pm_h), the hierarchy manager realigns the PMs amongst the currently registered schedulers (ensuring a uniform PM count distribution amongst all its managed low-level schedulers). If it is not possible to realign the PM set without violating the maximum PM count limit set by the user, then the hierarchy manager will create a new low-level scheduler and start the realignment process again. This process is repeated until the number of physical machines registered under a particular low-level scheduler reaches pm_{opt}.

As PMs belong to a particular low-level scheduler, when they are deregistered, their past low-level scheduler will have a less balanced pm count. The hierarchy manager will not react to this unbalance until the PM count in one of the low-level schedulers reach pm_l. In such case, first the hierarchy manager tries to realign the PM set of all low-level schedulers so their managed PM set will be equally sized. If this approach still leaves some low-level schedulers with too low PM counts then those schedulers are eliminated from the system and their PMs are distributed amongst the still remaining low-level schedulers by the hierarchy manger. The elimination process is continued until the PM set of each managed low-level scheduler is sized around pm_{opt}.

So far we have discussed the situation for high-level schedulers that are directly in contact with low-level ones. In some cases the number of their directly managed schedulers reach the boundaries of the optimally operated scheduler set size. In those cases they get in touch with the hierarchy manager. The manager will either create secondary schedulers alongside the scheduler in question or eliminate one with similar techniques discussed for low level schedulers. The difference: the profile created for these high-level schedulers are defined in terms of the number of directly managed schedulers instead of the number of supervised PMs in a low-level scheduler. In both cases the hierarchy manager tries to equalise the number of PMs under a particular scheduler before creating or eliminating a high-level scheduler. The physical machine counts are also automatically determined and realigned if some low-level schedulers are preferred more than the others.

Finally, the hierarchy manager is operated alongside the ingest scheduler, the top scheduler in the hierarchy. This scheduler is the one that will be responsible for receiving the VM management operations from the `IaaSService`. Also, as future work, the automated

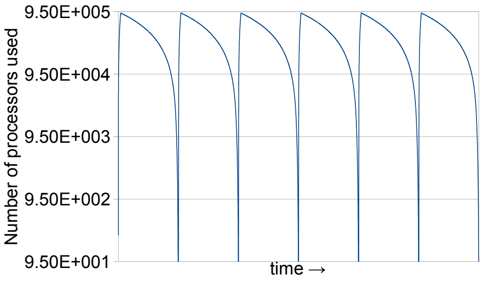

Figure 3: Example synthetic workload

hierarchies can follow the rules of software defined networking to have a more realistically constructed data centre layout.

IV. Evaluation

IV.1 Simulation Setup

Throughout the evaluation section, the following machine was used for executing the simulations: AMD Athlon FX 8120, 16 GB RAM, 128 GB SSD. The simulations were all set up and run in a controlled environment, the machine was never executing anything else just the simulators themselves. As workload traces for peta scale infrastructures are not yet public, in all the following cases the simulators were running a synthetic workload with similar characteristics than the one shown in Fig. 3. This synthetic workload has a peak utilisation with the same number of nodes as the simulated infrastructure has. This utilisation is the result of a randomly generated amount of virtual machine requests at a given time, and filling up the VMs with as many jobs as many needed to reach the expected utilisation profile at the given time instance (e.g., a particular VM in the load could host no jobs at all, but in some cases they can be filled up with several hundred if the VM's resources would not get exhausted by that many jobs). Also, the number of utilisation peaks can be configured and during the evaluation runs it was set up to be between 4-10 (the actual number of peaks was determined so it has had a stabilising effect on the simulation runtime – i.e., the number of peaks was set so the consecutive simulation runs have had small variance in their runtimes). Throughout this paper the following simulators were used: (i) DISSECT-CF 0.9.5 – as the old reference point that represents the latest stable release of the simulator without any peta-scale optimisations, (ii) DISSECT-CF 0.9.6 – as the last released version, (iii) DISSECT-CF 0.9.7* – the simulator incorporating the hierarchical extensions of this paper, and (iv) CloudSim 3.0.3 DVFS extensions – the independent reference point. Finally, it was ensured that independently from the simulator used, the program that set up and run the simulations was never consuming more CPU than 1% of the total CPU consumption of a simulation (this step assured that the below presented results are influenced mainly by the investigated simulators and not by the additional code used for the evaluation).

The simulated results of the extended DISSECT-CF simulator were *validated* by comparing its results to the past validated DISSECT-CF 0.9.5 results as well as to the results obtained from CloudSim 3.0.3 DVFS. After executing a workload in a simulation, the termination time of the last virtual machine was noted. These times were compared with both past simulators. The new simulations yielded final termination times within 0.05% of the older simulators, while significantly improving on simulation performance.

IV.2 Measurements

During the first measurements, basic scaling properties were investigated for all simulators. For this experiment, a 5000 core infrastructure was set-up in the simulators. The composition of the infrastructure was changed to range from a single node (with 5000 cores) to 5000 nodes (with a single core each). This experiment was designed to show the bottleneck situations in both the simulator's resource sharing mechanism and in its default virtual machine scheduler. Also, this experiment actually replicates the profiling technique introduced in Section III.3.2.

Figure 4 reveals the results of this first experiment. In the experiments utilising simulated infrastructures with less than 10 hosts, the resource sharing mechanism of the simulator is more dominant (i.e., the mechanism that determines how the resources of a single physical machine are shared amongst the VMs it is hosting). Similarly, experiments, with infrastructures consisting of over 1000 hosts, were dominated by the

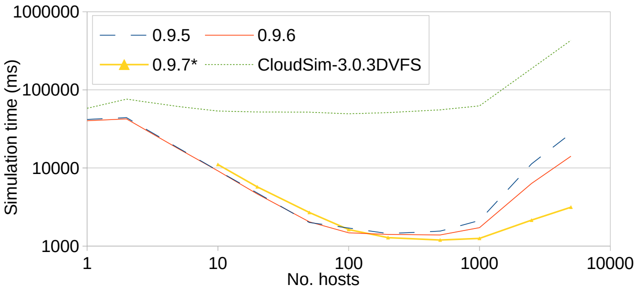

Figure 4: Small-scale measurement utilising an infrastructure with 5000 CPU cores (distributed amongst a varying number of hosts)

virtual machine scheduler's performance (i.e., in these cases it was very unlikely to have multiple virtual machines hosted in a single physical machine, thus the assignment mechanism of the virtual machines to physical ones become more dominant in execution time). As it can be seen, DISSECT-CF has a bigger overhead for resource sharing than CloudSim: DISSECT-CF previously had a 33% performance loss if the 5000 cores needed to be scheduled by the resource sharing mechanism of the simulator, while in case of CloudSim a virtual machine scheduler has a 86% performance loss. Next, the evaluation of DISSECT-CF 0.9.7 and its hierarchical extensions show that an over 8.8× improvement is achievable by just switching to the hierarchical scheduler. Remarks: the hierarchy built up by the new automated mechanism consisted of 10 low-level virtual machine schedulers and one high-level scheduler acting as ingress. Also the number of utilisation peaks was set to 10 as that gave the most stable measurements after evaluating the sample standard deviation of the measured runtimes of simulations.

Because of performance issues, only the first experiment was evaluated with CloudSim. The rest of the simulations are compared to past DISSECT-CF versions only as CloudSim based results were not obtainable in feasible time.

Next, the above discussed experiment was repeated with 50000 cores. This experiment was executed to show how the simulator scales in a well defined and controlled environment. The number of utilisation peaks was kept at 10 in order to allow a more direct comparison with the results from the previous experiment. The hierarchy built up by the new automated mechanism consisted of 50 low-level virtual machine schedulers. The results of this second experiment are shown in Fig. 5. As it can be seen, the hierarchical extension now improves with over 111× compared to the original DISSECT-CF 0.9.5 version. The figure also reveals, that it is not recommended to use hierarchical virtual machine schedulers in case there are too few physical machines on which these schedulers can place the requested VMs (e.g., performance degradation is observable for simulations with less than 600 hosts – the worst case degradation reaches 33%). In terms of scaling, the 5000 node infrastructure performs 1.72× better when there are more CPU cores supervised by a single node (the 50000 core experiment executed with 64VMs/ms – virtual machines processed per millisecond – while the 5000 core experiment achieved a performance of 37VMs/ms).

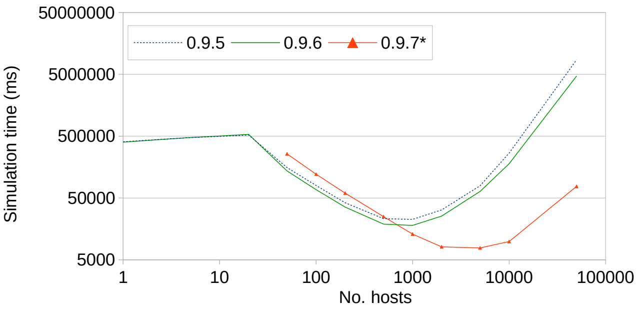

Figure 5: Medium-scale measurement using an infrastructure consisting of 50000 cores (distributed amongst a varying number of hosts)

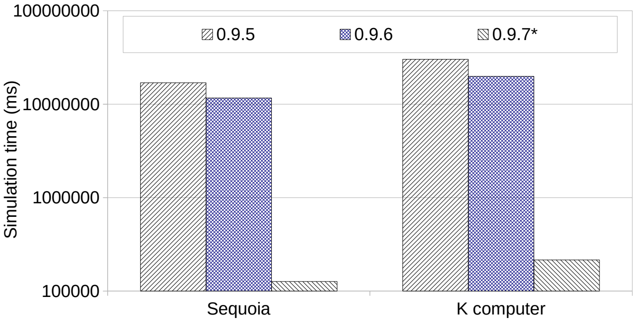

Figure 6: Peta-scale measurement

Finally, two peta-scale experiments were also executed. For these experiments, two of the top supercomputers were selected and their infrastructures were constructed in the simulator. The selected supercomputers were chosen with two criteria: they should be listed amongst the top 10 from the top500 supercomputers list, and they should not have accelerators augmenting their computing power (this second requirement is needed because the current simulator cannot handle tasks running simultaneously on an accelerator and on a CPU core). The two supercomputers fulfilling these requirements:

- Sequoia – with 1,572,864 CPU cores in 98,304 nodes.
- K Computer – with 705,024 CPU cores in 88,128 nodes.

These two large-scale systems were simulated in the various versions of DISSECT-CF with 4 utilisation peaks for Sequoia and 6 utilisation peaks for the K Computer. Figure 6 reveals the measured results. As it can be seen, without the hierarchical extensions of DISSECT-CF, the simulations complete in over 3 hours (best case with the fastest 0.9.6 version without hierarchical scheduling). Thus it is not realistic to expect users of the simulators that they can execute hundreds or even thousands of scenarios for their research. However, with the hierarchical extensions even the peta-scale simulation reaches a manageable less than 4 minutes runtime. The automatically created hierarchy is still only two levels deep for these large-scale infrastructures (the third level of the hierarchy would appear for exa-scale simulations), but now consists of a little over 100 smaller infrastructure fragments. The resulting performance increase is between 91-139× compared to the original hierarchy-less simulations.

V. Conclusions and Future Works

This paper presented a new hierarchical VM scheduling technique for the DISSECT-CF simulator. The new scheduling technique is aimed at large-scale simulations (in the current paper it is focused on simulating peta-scale systems). The paper shows that the new technique can successfully reduce the time required for simulations on such scale. The observable performance improvements are enabling the application of the DISSECT-CF simulator during the evaluation of even peta-scale cloud systems (e.g., systems similarly constructed as the Sequoia or K computer present in recent top500 lists).

Regarding future works, the results presented in this paper is just the first step in many to enable a generic cloud and distributed systems simulator to cope with such large-scale systems like the commercially available Amazon EC2. In the following, the simulator is intended to be improved with GPGPU support to follow the trends of the accelerator based systems. Also, further research is needed to identify the peculiarities of the various interconnects in these large-scale systems. This new research will allow DISSECT-CF based simulations to handle infrastructures with mixed interconnects or federation of infrastructures with different interconnect technologies.

Acknowledgement

This work was partially supported by the European Union's Horizon 2020 programme under grant agreement No 644179 (ENTICE), by the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS) and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Software availability

This paper described the behaviour and features of DISSECT-CF version 0.9.7. Its source code is open and available (under the licensing terms of the GNU LGPL 3) at the following website:
`https://github.com/kecskemeti/dissect-cf`

References

[1] M. Armbrust, A. Fox, R. Griffith *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California at Berkley, Tech. Rep. UCB/EECS-2009-28, February 2009.

[2] G. Sakellari and G. Loukas, "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing," *Simul. Model. Pract. Th.*, vol. 39, pp. 92–103, 2013.

[3] G. Kecskemeti, "DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds," *Simulation Modelling Practice and Theory*, to appear, DOI: 10.1016/j.simpat.2015.05.009, pp. 1–28, 2015.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, January 2011.

[5] S. K. Garg and R. Buyya, "NetworkCloudSim: modelling parallel applications in cloud simulations," in *Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*. Victoria, NSW: IEEE, December 2011, pp. 105–113.

[6] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, Sept 2012.

[7] X. Li, X. Jiang, P. Huang, and K. Ye, "DartCSim: An enhanced user-friendly cloud simulation system based on CloudSim with better performance," in *2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*, vol. 1. Hangzhou: IEEE, Oct 2012, pp. 392–396.

[8] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum, "SimIC: Designing a new inter-cloud simulation platform for integrating large-scale resource management," in *27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 90–97.

[9] Y. Shi, X. Jiang, and K. Ye, "An energy-efficient scheme for cloud resource provisioning based on CloudSim," in *IEEE International Conference on Cluster Computing (CLUSTER)*. Austin, TX: IEEE, Sept 2011, pp. 595–599.

[10] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, and M. Alsaleh, "Teachcloud: a cloud computing educational toolkit," *Int. J. of Cloud Computing*, vol. 2, no. 2/3, pp. 237–257, 2012.

[11] H. Casanova, "SimGrid: A toolkit for the simulation of application scheduling," in *First IEEE/ACM International Symposium on Cluster Computing and the Grid*. Brisbane, Qld.: IEEE, May 2001, pp. 430–437.

[12] T. Hirofuchi, A. Lèbre, L. Pouilloux *et al.*, "Adding a live migration model into SimGrid, one more step toward the simulation of infrastructure-as-a-service concerns," in *5th IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom)*, Bristol, UK, December 2013, pp. 96–103.

[13] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "Groudsim: an event-based simulation framework for computational grids and clouds," in *Euro-Par 2010 Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, vol. 6586. Springer, 2011, pp. 305–313.

[14] S. Ostermann, K. Plankensteiner, D. Bodner, G. Kraler, and R. Prodan, "Integration of an event-based simulation framework into a scientific workflow execution environment for grids and clouds," in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science. Poznan, Poland: Springer, 2011, vol. 6994, pp. 1–13.

[15] G. Kecskemeti, S. Ostermann, and R. Prodan, "Fostering energy-awareness in simulations behind scientific workflow management systems," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, Dec 2014, pp. 29–38.

[16] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, J. Carretero, and I. M. Llorente, "Design of a new cloud computing simulation platform," in *Computational Science and Its Applications-ICCSA 2011*, ser. Lecture Notes in Computer Science. Santander, Spain: Springer, 2011, vol. 6784, pp. 582–593.

[17] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware

cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[18] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "iCanCloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.

[19] C. Engelmann, "Scaling to a million cores and beyond: Using light-weight simulation to understand the challenges ahead on the road to exascale," *Future Gener. Comp. Sy.*, vol. 30, pp. 59–65, 2014.

[20] K. Wang, K. Brandstatter, and I. Raicu, "Simmatrix: Simulator for many-task computing execution fabric at exascale," in *Proceedings of the High Performance Computing Symposium*. Society for Computer Simulation International, 2013, p. 9.