ulm university universität
uulm

Predictability of Resource Intensive Big Data and HPC Jobs in Cloud Data Centres

Christopher B. Hauser, Jörg Domaschka, Stefan Wesner
Institute of Information Resource Management
Ulm University
Albert-Einstein-Allee 43, 89081 Ulm, Germany
Email: {christopher.hauser, joerg.domaschka, stefan.wesner}@uni-ulm.de

© *2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.*
DOI: 10.1109/QRS-C.2018.00069

Predictability of Resource Intensive Big Data and HPC Jobs in Cloud Data Centres

Christopher B. Hauser, Jörg Domaschka, Stefan Wesner
Institute of Information Resource Management
Ulm University
Albert-Einstein-Allee 43, 89081 Ulm, Germany
Email: {christopher.hauser, joerg.domaschka, stefan.wesner}@uni-ulm.de

***Abstract*—Cloud data centres share physical resources at the same time with multiple users, which can lead to resource interferences. Especially with resource intensive computations like HPC or big data processing jobs, neighbouring applications in a cloud data centre may experience less performance of their assigned virtual resources. This work evaluates the predictability of such resource intensive jobs in principle. The assumption is, that the execution behaviour of such computations depends on the computation and the environment parameters. From these two influencing factors, the predictability is the outcome of removing the hardware dependent environment parameters from the observed execution behaviour, in order to compute any other execution behaviour for computations with similar computation parameters but on a different environment. The assumptions are analysed and evaluated with the HPC application Molpro.**

***Keywords*-Resource Prediction, Resource Intensive Jobs, Cloud Computing, Resource Interference**

I. Introduction

Cloud computing is becoming more popular recently, due to its virtually unlimited computation and storage resources, and the flexible on demand resource provisioning. High performance computing (HPC) on the other hand is still the most relevant data centre operation model for science and engineering, when large and intensive computations have to be scheduled among distributed physical resources. Anyway HPC jobs are slowly moving towards cloud computing, since Infrastructure as a Service (IaaS) offers a dynamic payment model for virtual servers instead of buying and hosting physical servers for a data centre. Additionally, with the rise of big data applications, data and hence compute intensive applications similar to HPC jobs appear in cloud computing.

Moving HPC jobs or similar resource intensive jobs to cloud computing brings some new challenges to cloud providers. The fact that physical resources are shared amongst customers as virtual resources, in addition with overbooking, introduces the so called *Noisy Neighbour Effect*: virtual machines experience performance degradations whenever shared physical resources are heavily demanded by other virtual machines. The *Quality of Experience* (QoE) in this case decreases. It is hence necessary for cloud providers to improve the resource reliability for their customers to have a deep understanding of the resource demands of resource intensive jobs. Figure 1 shows the impact

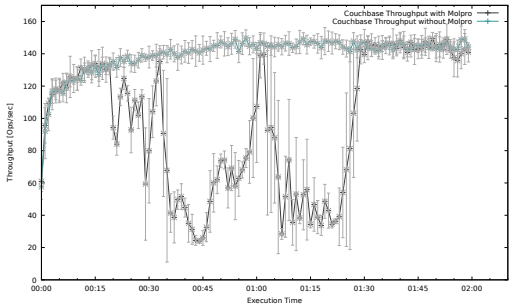


Figure 1. Couchbase Throughput with and without HPC computation

of running an HPC job on the shared physical resources in parallel to a distributed cloud database. The disk I/O intensive Molpro computation as an representative of an HPC job in this example has visible impacts on the throughput of the Couchbase database cluster. In this example the bottleneck is the storage (hard drive disks in RAID 1), which is shared by a "Kernel-based Virtual Machine" (KVM) hypervisor.

The presented work analyses the predictability of the workload characteristics of resource intensive jobs to help cloud data centre managers, who provide private or public clouds as a service provider, to reduce or even avoid the noisy neighbour effect. Applications in this setup hence run on shared physical resources, yet we analyse the resource utilisation of each single virtual machine with one application each by itself. The work presents the foundation of a tool set for cloud resource scheduling with utilisation awareness - where predictability is the first important factor and is analysed in the following.

The paper is organized as follows. Section II clarifies the problem statement considered in our work. Section III reviews related work in this area. Section IV describes our approach towards predictability of resource intensive computations in cloud computing, which is then practically analysed in section V with one exemplary application. From this analysis we evaluate with a proof of concept approach in section VI the approach presented in section IV. Section VII finally concludes the paper.

II. Problem Statement

The following work analyses the predictability of the workload characteristics of resource intensive jobs. Therefore we first introduce main characteristics of cloud and HPC data centres and workloads. From there, research questions are derived.

A. Cloud Data Centres

Cloud computing consists of management software which controls a large amount of physical resources in a data centre and offers abstracted access to a Cloud user. Typical Cloud applications running in such a setup are continuously running services like web servers or web based applications. The encapsulation for multiple users in parallel and sharing of physical resources is realised by virtualisation. Yet, the encapsulation limits user access securely but does not isolate the performance demands. Hence, in parallel running applications might influence each other in an intransparent and undeterministic way.

A Cloud middleware offers interfaces to spawn new or remove existing virtual resources. The scheduling of new virtual machines takes place immediately after a user issued a request. The VM has a set of maximum hardware properties assigned, consisting of CPU cores, amount of memory and disk space. A typical Cloud scheduler puts new VMs on the host with the most free memory available [1], as long as the overbooking factors for each resource is not exceeding. The Cloud schedulers usually take not into account the heterogeneity of physical servers or the applications' hardware affinity [2].

The effect of experienced resource degradation - as shown in figure 1 - can happen on any resource type like CPU, memory, disk or network. Yet, the impact differs, depending on the parallelism the physical resource can handle: *i)* Modern CPUs usually come with multiple cores, servers even have multiple CPUs, which makes parallel execution physically possible. *ii)* the allocation of memory is rather static and is (compared to CPU cycles or I/O operations) slowly changing. Still swapping to disk leads due to heavy memory allocation leads to the noisy neighbour effect as well. Memory bandwidth utilisation are not considered, since monitoring memory access performance is usually not part of monitoring software. *iii)* Disk I/O operations are usually transferred towards the computation unit using a small number of physical connections (e.g. SATA, SAS or Fiberchannel). This connection and the sequential access handling of disks introduce a bottleneck for parallelism. RAID levels partially introduce parallel disk access, yet limited to a maximum amount of disks handled by a RAID controller. Disk allocation is out of scope of this work, since running out of disk space will break most operating systems and applications anyway, why we assume enough capacity. *iv)* Similarly to disk I/O, the network is usually designed to be hierarchical. Eventually on the path towards the public network from within a data centre, the network capacity and hence physical network connections will be shared by multiple consumers. Link aggregation can increase the parallelism of physical network connections to some limited extent.

B. Resource Intensive Computations

In HPC, machines are not directly and immediately available to the user - instead, HPC clusters are typically overbooked, as they are constrained in size compared to the number and size of submitted jobs. Therefore, the user will have to submit his job to a "waiting queue" which will select jobs according to resource availability. Typical schedulers for this task include the Portable Batch System (PBS) in its commercial or open source flavour OpenPBS[1], MOAB[2] or Slurm[3], to name a few. They all try to place jobs so as to maximize resource utilization, but keep a common fair share goal. HPC data centres are typically consisting of a large number of (comparably homogeneous) compute resources. An HPC application typically is distributed over many servers and runs for a long time (in the order of hours to weeks). Due to high demand of hardware resources and the sensitivity of the application towards resource utilisation, HPC servers are dedicated to one user at a time. The overall cluster is hence fully used by only a few applications, which might not even fully use their resources at all time. This leads to a high wait time for running a new job and an overall inefficient data centre utilisation. Further, the efficiency of the job scheduling mechanism depends on the correctness of the resource requirements specification of the submitted jobs. Since the requirements are submitted by the users themselves, there is a strong tendency to overestimate the resource needs, in order to avoid running out of resources or time.

Similarly, resource intensive computations appear in big data applications. Frameworks with map-reduce approaches or using the Apache Spark[4] framework process up to petabytes of data. In this context, big data job scheduler like Apache Mesos[5] appeared, to schedule containers and jobs on data centres to solve big data challenges. A single map-reduce job has by design changing resource demands over time, due to the split in map phase and reduce phase. Further, each job has its resource utilisation depending on the task it fulfils. Since these big data computations usually run on cloud data centres, the resource interference occurs eventually.

Both HPC and Big Data computations when running in Cloud data centres eventually lead to resource degradations for neighbouring virtual machines. While their historic

[1] http://pbspro.org/
[2] http://www.adaptivecomputing.com/products/hpc-products/
[3] https://slurm.schedmd.com/
[4] https://spark.apache.org/
[5] http://mesos.apache.org/

origin differs, the conceptional appearance from a Cloud provider's perspective is the same: batch job workloads, with intense resource demands, with repeating patterns in their resource utilisations.

C. Research Questions & Contribution

To recap the main problem with resource intensive jobs in cloud computing: the physical resources are not dedicated to one customer at time but shared, leading to unpredictable performance degradations. Cloud schedulers usually place VMs when their creation is requested and only some schedulers replace at runtime. To optimize the resource scheduling in cloud data centres, this work focuses on the analysis, if it is possible to reliably analyse the resource demands of resource intensive computations. This analysis as an output for cloud providers can be used to schedule resources in a cloud data centre at runtime. The research questions of our analysis are as follows:

1) What are influencing factors of the execution behaviour of resource intensive computations?
2) How can a prediction model depending on influencing factors be for HPC jobs?

III. Related Work

The general idea of moving HPC workloads to Cloud data centres is widely discussed in literature, and is for some types of computations a reasonable alternative to HPC clusters [3]. Especially for applications of smaller scale, the payment model of public clouds is with its elastic and virtually unlimited computation resources economically interesting. Cloud computing can further help to optimise costs by using cloud features like snapshotting and resuming [4].

Research focused on the the performance of HPC workloads, like by Nawaz et al. in [5], where I/O intensive workloads are evaluated on two of the most frequently used public cloud offerings on the market recently. This study compared the various different storage models offered by Google Cloud[6] and Amazon Web Services[7]. The main message is the importance of data locality, especially when moving to clouds. The public cloud providers recently push their offerings towards HPC, like promoting HPC on Amazon EC2[8] with self deploying workload schedulers, special virtual machine types or even with bare metal hosts. Similar approaches have been made by Brandt et al., to enable the HPC cluster management softwate OVIS [9] on cloud computing environments in [6].

A even more sophisticated study including the cloud internals of a provider perspective has been done by Gupta et al. This study first evaluates HPC applications on clouds [7] to identify issues, solved by a scheduler to place virtual machines with HPC workloads accordingly on available physical resources in a private cloud infrastructure [8].

While these researches either focus on public clouds or on dedicated HPC-aware private clouds, our approach is motivated from a practical point of view: a cloud provider with very mixed resource demanding virtual machines. While our approach will eventually lead to virtual machine placement and hence scheduling decisions on various levels like in the CACTOS project [2] with a cloud monitoring solution [9], this paper steps back to first elaborate on the feasibility of reliably predicting workloads for cloud data centre optimisation in principal.

Analysing resource intensive jobs is core functionality of tools and frameworks like from Allinea[10], Vampir[11], Paraver[12], Scalasca[13], and many more. All these tools have in common to help users to optimize the runtime of their HPC jobs. While increasing the efficiency on code level is an important part of developing HPC jobs, our approach assumes the operational level. We assume that HPC jobs run inside virtual machines on a private cloud, with only little knowledge about the internals of the virtual machines. Our approach rather relates with task characteristics estimation in [10], where sophisticated monitoring data is used to automatically characterize resource needs and detect correlations using clustering techniques. While this approach is evaluated on a set of scientific workloads, our work has to deal with black box virtual machines and a high diversity of resource utilisation traces.

Performance interference in cloud computing, known as the "noisy neighbour" effect, has been discussed in research and by public cloud providers like Amazon. Approaches exist like in [2] or [11]. Their goal is a smart placement of virtual machines on physical resources, to avoid performance interferences at runtime. Another approach to control the performance interference is to limit the virtual resources according to the current utilisation and the available physical resources. For networking a software defined networking approach can be considered [12]. Since placement and resource scheduling is based on knowledge, this work first elaborates the predictability to eventually provide a reliable input to such algorithms.

IV. Predictability Models for Resource Intensive Jobs

We assume that the considered resource intensive jobs are batch jobs without any interaction while running. In fact, a job consists of a mix of an application and input parameters to compute with. The application is either written to fit a specific use case or is a general purpose application,

[6] https://cloud.google.com
[7] https://aws.amazon.com/
[8] https://aws.amazon.com/hpc/
[9] https://ovis.ca.sandia.gov/
[10] https://www.allinea.com
[11] https://www.vampir.eu/
[12] http://www.vi-hps.org/Tools/Paraver.html
[13] http://www.scalasca.org/

which strengthens the impact of the input parameters. We call the set of application and input parameters *computation parameters* of a resource intensive job in the following. The assumption in the following is, since resource intensive jobs are running without any human interaction, that their execution behaviour is predictable in principle, since the computation is defined only by its computation parameters.

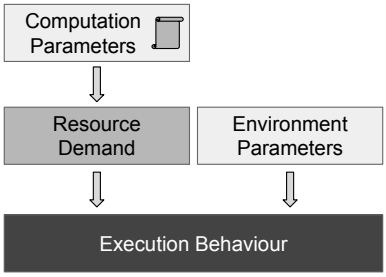


Figure 2. Influencing Factors for the Execution Behaviour

From the computation parameters the *resource demand* can be derived. Some jobs specify resource utilisation like the maximal usage of memory or cpu cores in the input parameters. While these parameters are directly influencing specific resource types, more generally, the computation complexity defined by the computation parameters implicitly defines the resource types and their utilisation. To name an example, a computation might be on a single node and a single CPU only, without any communication. Other examples are highly distributed computations with a high amount of network and disk operations. The evaluation in this paper will show, that this resource demand stays the same with static computation parameters. We are aware that some applications behave differently depending on the used hardware, for e.g. caching intermediate results versus recomputing them.

Finally, the execution behaviour of a job depends on the used resources to run the job. The set of used resources is called *environment parameters* in the following. These set of parameters describe the hardware setup, like CPU model with frequency and cores, the underlying storage like SSDs or HDDs, the network infrastructure and the memory. The resource demand, which directly depends on the computation parameters, and the environment parameters define the *execution behaviour*, like represented in figure 2.

A. From Execution Behaviour to Prediction

From figure 2, the conclusion for a hardware independent predictability model *Resource Demand* follows from extracting the environment parameters from the execution behaviour. The compilation of these abstract resource demands can be considered as the training phase in machine learning context. At runtime these resource demand models are then used to predict the execution behaviour on different hardware but with similar computation parameters. While the first modelling step to compile the abstract resource demand takes place after each computation from monitoring data, the prediction takes place whenever new or running high performance computations have to be considered in a cloud data centre. Our assumptions with $Computation\ Parameters\ C$, $Resource\ Demand\ D$, $Environment\ Parameters\ E$, and $Execution\ Behaviour\ B$ are as follows: as defined in figure 3 $D \wedge E \rightsquigarrow B$, with $C \rightsquigarrow D$. From monitoring data the execution behaviour can be extracted to calculate the resource demand with $B \setminus E \rightsquigarrow D$. From this resource demand model, any behaviour can be predicted on alternative hardware with $D \wedge E' \rightsquigarrow B'$. Figure 3 depicts this approach: from this abstract resource demand any other execution behaviour can be compiled by aligning the abstract resource demand with a set of execution parameters.

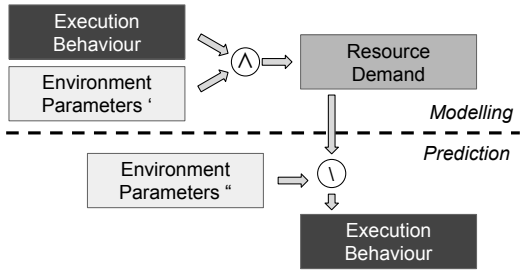


Figure 3. Approach for Predictability

While this work provides an evaluation of the conceptual feasibility with practical evaluations, the integration of the resource demand for resource scheduling is out of scope of this paper.

V. Analysis of Resource Intensive Computations

To proof the concept of predictability models for resource intensive jobs in cloud data centres, we analysed an exemplary resource intensive computation in detail. The analysis is designed to reveal three insights: *i)* the predictability of resource intensive jobs in general, *ii)* the impact of computation parameters on the execution behaviour and *iii)* the impact of environment parameters on the execution behaviour.

The approach to analyse the predictability of resource intensive jobs for cloud computing is based on evaluation runs of the Molpro application. To understand the impact of computation and environment parameters on the execution behaviour (cf. figure 2), we first varied the computation parameters with static environment parameters, and second varied the environment parameters with static computation parameters. This way, the two sets of parameters can be isolated and analysed separately.

The evaluation first shows that the resource utilisation with static computation parameters and static environment

parameters stays identical. This first insight is important to allow for predictability. From there, the computation parameters are first analysed with varying computation complexities, meaning the computation parameters are slightly changed. Last, the environment parameters are changed to identify the execution behaviour with varying hardware resources.

A. Exemplary Use Case

To further analyse resource intensive computations and their application behaviour we focus on low scalable but disk I/O intensive exemplary workloads as a starting point. This limitation in first instance leaves out networking on purpose but starts with the analysis of CPU, disk operations and memory. Computations in the field of our focus can be found in simulations for structural mechanics or quantum chemistry (QC). We chose the `Molpro`[14] application package that stands for a full class of applications in the field of QC [13]. Within the Molpro application a set of different methods is implemented. This leads to a very different application behaviour depending on the input data, what makes this application interesting as an exemplary workload. Two well-established methods are DFT and LCCSD: *Density Functional Theory (DFT)* based algorithms have a high dependency on the CPU speed and memory bandwidth but little dependency on I/O. *Conventional coupled-cluster (CCSD)* methods and their local variant LCCSD [14] on the other hand show a clear dependency on I/O speed. Since we also want to include disk I/O, we analyse the (L)CCSD method in the following in more detail, although the methodology will apply for any other method and even applications as well.

The LCCSD computations to evaluate the predictability in the following are designed to run in an acceptable time frame (less than ten hours) in order to run enough repetitions. In a real world scenario, these computations are expected to run even longer, up to several days.

B. Predictability: Constant Resource Utilisation

To show that resource intensive jobs are principally predictable, we isolated one server node of our OpenStack cloud testbed and placed a single virtual machine on this node to host the experiment. The physical server has a dual socket Intel Xeon CPU E5-2670 with 2.60GHz frequency, and in total 128GB of DDR3 memory. For storage, the physical server has two HDDs in RAID1 with 1TB capacity and 7200 rpm. The virtual machine on top of a KVM hypervisor on this physical server has four virtual cpu cores, 17GB memory and a 300GB qcow2 file as virtual disk. The local virtual machine storage is comparably the fastest storage solution [15]. The exemplary workload is issued by Molpro with a LCCSD computation.

[14]http://www.molpro.net

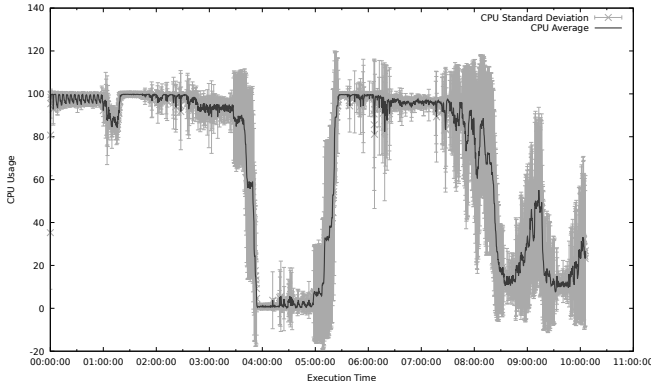


Figure 4. CPU Utilisation (Average and Standard Deviation)

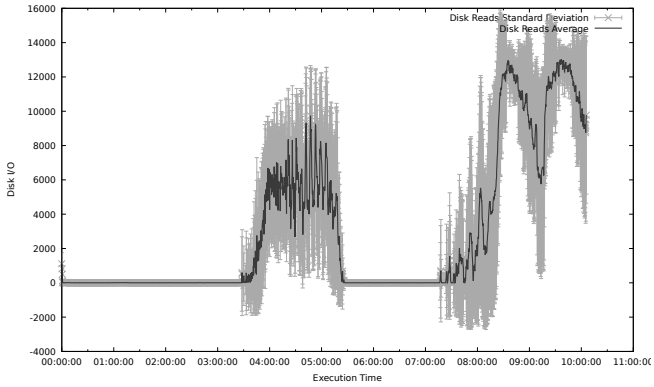


Figure 5. Disk Reads (Average and Standard Deviation)

The results of this experiment are based on 23 runs. Figure 4 and 5 show two graphs: the CPU utilisation and the disk reads, each with average and standard deviation over the 23 runs. The main message of this result is the predictable appearance of read requests to the disk and a reduction of CPU utilisation. The graphs also show the changing resource demands over time, depending on the current application phase. While at the beginning the computation is the major workload, writing some temporary results to the disk, the behaviour changes within minutes at approx. 4:00h. At this time, the disk becomes the bottleneck, while the computation is randomly reading previously written data from disk. Since the disk is a shared physical resource for all hosted VMs in a productive cloud infrastructure with local storage, the read heavy phases of this Molpro computation will harm the quality of other VMs on the same host. Yet, with the knowledge about the exact occurrences of these peaks, the cloud provider's management software can avoid customer shortcomings by scheduling the resources in advance accordingly.

C. Impact of Computation Parameters

Obviously, depending on the application under consideration, the resource demand will vary. The same applies

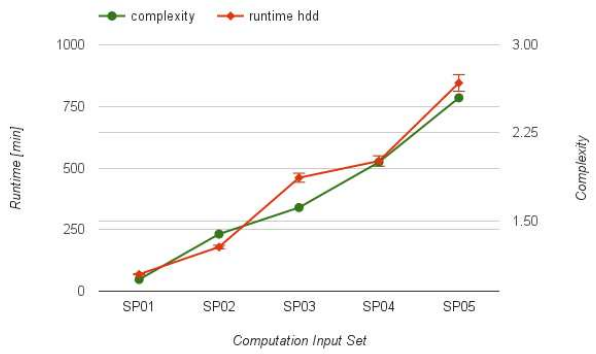


Figure 6. Linear dependency of computation complexity to execution time.

for input parameters of general purpose applications of a specific field, like Molpro for quantum chemistry. Still, the dependency between the complexity of these computation parameters and the execution time is of importance to predict a computation's execution behaviour for cloud infrastructures.

To elaborate on the impact of computation parameters, we analysed different commands, which can be specified as input parameters for Molpro and lead to different computations. From analysing these input parameters, the type of computation and leading to the type of resource demands can be derived. For example a DFT algorithm behaves completely different than a LCCSD computation: while DFT jobs run from minutes to a few hours on CPU and memory, LCCSD jobs run several hours and use CPU, memory and disk. Understanding the exact input parameters as part of the computation parameters hence leads to a prediction of the resource demand, and how the resource demand changes during execution time.

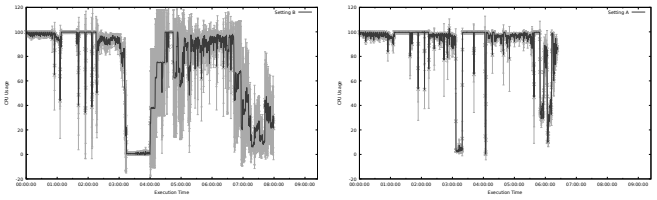
To further detail the impact of computation parameters to the execution behaviour, we evaluated runs of Molpro with an LCCSD computation with a varying problem size. The problem size in this context is defined by the amount of molecules and the type of electron approximation. Elaborated in many different evaluations, the problem size is directly influencing the minimum of required memory and disk space for the computation. The CPU time is affected as well, since more molecules lead to more computation time. Figure 6 shows the mostly linear dependency of computation complexity and computation runtime, where computation complexity is a factor expressing the amount of molecules.

D. Impact of Environment Parameters

After analysing the computation parameters as one influencing factor of the execution behaviour (cf. figure 2), the environment parameters and their impact are analysed. To understand the change of behaviour, we fixed the computation parameters to Molpro with a LCCSD computation

Table I
VARYING ENVIRONMENT PARAMETERS

		16 GB RAM		32 GB RAM	
		setup	avg. (std.)	setup	avg. (std.)
Intel i5 3.2GHz	SSD	A	385.14 (0.36)	E	381.29 (0.20)
	HDD	B	504.16 (22.11)	F	487.23 (21.04)
Intel e5 2.6GHz	SSD	C	452.88 (7.07)	G	450.74 (4.28)
	HDD	D	578.60 (24.63)	H	597.18 (35.02)



(a) Setup B with HDD (b) Setup A with SSD

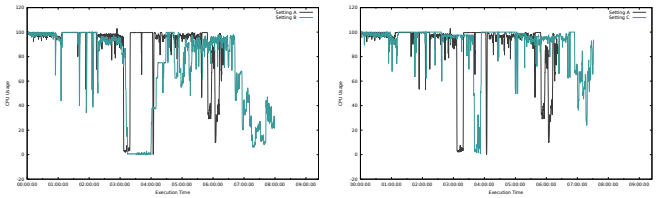
Figure 7. Average CPU Utilisation with HDD and SSD storage

and evaluated the execution behaviour on different physical hardware. Table I lists the eight setups for this evaluation, with two different CPUs (Intel i5 with 3.2GHz and Intel e5 with 2.6GHz), different storage technologies (HDD and SSD), and different amount of DDR3 memory (16GB and 32GB).

The first evaluation insight of the eight variations is the changing standard deviation depending on the used type of storage. Since the chosen computation parameter leads to a high amount of random disk reads, the standard deviation depends on the access time (or seek time) of the disk. Figure 7a and 7b shows the average and standard deviation of cpu usage with HDD (figure 7a) and SSD (figure 7b). The much faster access time of SSD as storage device leads to much lower standard deviations and hence a much higher accuracy for predicting the execution behaviour. The standard deviations and how they differ between SSD and HDD are also stated in table I. Since additional memory is used for caching disk requests, the standard deviation changes also depending on the amount of memory. Apart from that, additional memory shows only minor effect on the overall execution behaviour.

The difference for the execution behaviour of SSD and HDD as storage is also visible in figure 8a, where the average CPU utilisation is printed for setting A (SSD) and setting B (HDD). While the first two hours of execution is mostly identical, starting at 3:00h the impact of storage becomes clearly visible. While the disk I/O phase for setting A takes approx. 20 minutes, this phase starts slightly later for setting B and takes approx. one hour. In total, the execution behaviour leads to two hours longer execution time for setting B, with similar CPU phases but much longer and less accurate disk I/O phases.

Similar to the evaluation with different storage types is the



(a) Setup A & B with SSD / HDD (b) Setup A & C with 3.2/2.6 GHz

Figure 8. Average CPU Utilisation with Changing Hardware Setup

evaluation for different CPU frequencies. Figure 8b shows the average CPU utilisation for settings A (3.2GHz) and C (2.6GHz). The disk I/O phase as a reference point shows the impact of CPU frequency: while for setting A it starts at about 3:00h and takes approx. 20 minutes, this phase starts at 3:40h and hence 40 minutes later. The overall execution time is more than one hour more with the slower CPU frequency, while the CPU and disk I/O phases predictably occur with a factor relative to the CPU frequency.

E. Concluding the Analysis

The evaluation of predictability of high performance computations first shows that with static computation and environment parameters the execution behaviour for computations stays with only low deviations predictably static. This first enabling conclusion allows to have a detailed analysis of the two influencing parameters for describing the computation and the environment. Changing the computation algorithm obviously leads to a different execution behaviour, while changing only the complexity of the computation linearly scales with the execution behaviour. These insights conclude a static behaviour of jobs, which allows a prediction in general. Yet the prediction of the execution behaviour strongly relates with the underlying hardware. The evaluation shows that the amount of memory is comparably less important, while storage access time and cpu frequency has a reasonable impact. From the evaluation, the next step is to derive a prediction model for high performance computations in cloud computing.

VI. Proof of Concept Calculations with Prediction Models

In order to proof the concept for predicting resource intensive computations, the conceptual approach presented in section IV is exemplary calculated with the results from the analysis in section V. To simplify the proof of concept, this example validates the whole execution time of a job, while accepting a loss of accuracy. In an optimised system, the overall execution time should be divided in its phases like in [16].

To proof the prediction of one execution behaviour to another with varying environment parameters, the proof of concept takes the two most influencing environment parameters: the type of disk and the CPU frequency. The calculations are based on the setups A,B and A,C from section V-D. First, the environment parameters from A are extracted from the execution behaviour A. From this hardware independent factor, the execution behaviour for setups B and C are calculated and compared with the measured execution behaviours. When considering the monitoring data, the calculations use the 95th percentile of the collected data to reduce noise and outliers. The statistical computations may be reconsidered, when phases are considered instead of the whole execution time.

A. Prediction with changing type of disk (HDD vs.SSD)

The measured overall execution time for setup A (SSD) is 385.14 min., while for setup B (HDD) it takes 504.16 min. The difference regarding the overall execution time is hence $\frac{A}{B}$. The difference for SSD and HDD is the overall amount of reads and writes, which needs to be analysed in more detail: Setup A has a 95th percentile for writes of 73048 ops and for reads 964632 ops (sum: 1037680 ops), Setup B has 95th percentile for writes of 20091 ops and for reads 296900 ops (sum: 316991 ops). The factor of overall disk I/O operations per second between both setups is hence $f_{disk} = \frac{A_{disk}}{B_{disk}} = 3.2735$. The predictability of execution behaviour A but with environment parameters from B is hence $(1 + \frac{1}{f_{disk}}) * A$ which is 502.73, and hence - due to accuracy issues with this proof of concept approach - approximately equal to the execution behaviour B with 504.16.

B. Prediction with changing CPU frequency

Similarly to the predictability with changing the type of disk, the factor of the overall execution time for setup A and setup B is $\frac{A}{C} = \frac{385.14min}{452.88min}$. The changed environment parameter in this example is the CPU frequency, hence this factor has to be related to the the product of CPU utilisation and CPU frequency. For setup A the 95th percentile for CPU utilisation is 99.7122144253 with a frequency of 3.2GHz (product: 319.079), and for setup C with the 95th percentile of CPU utilisation 100.0 and frequency 2.6Hz (product: 260.0). For predicting the execution behaviour C out of A, the computed resource demand is 472.65 - which is with inaccuracy approximately the expected 452.88 of execution behaviour C.

VII. Conclusion

In this work, the predictability of resource intensive computations in cloud computing data centres has been analysed. We first introduce the problem statement of resource interference in cloud data centres, especially of resource intensive computations like HPC or big data processing jobs. From the problem statement, our approach is described as predictability models, where the two influencing factors of computations' execution behaviour are the computation parameters and the environment parameters. A detailed

analysis shows the constant resource utilisation, which is the first required assumption for predictability. The analysis also shows the impact of computation and environment parameters to the execution behaviour. From this analysis, the predictability models are exemplary evaluated.

This work focuses on the very first requirements for resource scheduling in data centres: the predictability of resource consumptions. While related work in research is solving this bin packing problem, the knowledge about the applications' resource traces are the key aspect. Recent research projects like CACTOS [2] show that the overall data centre utilisation and experience can be optimised. With a monitoring system for controlling the overbooking [9], the profiling for reliably predicting execution behaviours can be achieved with the analysis and suggested approach in this work. The predictability models can then be added to job descriptions or images like addressed in [17].

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the federal state of Baden-Württemberg (Germany), under the Project "ViCE - Virtual Open Science Collaboration Environment", and from the European Commission's Seventh Framework Programme under grant agreement number 610711 (CACTOS).

REFERENCES

[1] OpenStack Foundation, "Openstack configuration reference - scheduling," Dec. 2017. [Online]. Available: https://docs.openstack.org/nova/latest/user/filter-scheduler.html

[2] P.-O. Ostberg, H. Groenda, S. Wesner, J. Byrne, D. Nikolopoulos, C. Sheridan, J. Krzywda, A. Ali-Eldin, J. Tordsson, E. Elmroth, C. Stier, K. Krogmann, J. Domaschka, C. Hauser, P. Byrne, S. Svorobej, B. McCollum, Z. Papazachos, D. Whigham, S. Ruth, and D. Paurevic, "The cactos vision of context-aware cloud topology optimization and simulation," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, Dec 2014, pp. 26–31.

[3] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman, "Experiences using cloud computing for a scientific workflow application," in *Proceedings of the 2nd international workshop on Scientific cloud computing*. ACM, 2011, pp. 15–24.

[4] Y. Gong, B. He, and A. C. Zhou, "Monetary cost optimizations for mpi-based hpc applications on amazon clouds: Checkpoints and replicated execution," in *High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for*. IEEE, 2015, pp. 1–12.

[5] H. Nawaz, G. Juve, R. F. da Silva, and E. Deelman, "Performance analysis of an i/o-intensive workflow executing on google cloud and amazon web services."

[6] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "Resource monitoring and management with ovis to enable hpc in cloud computing environments," in *IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–8.

[7] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *Open Cirrus Summit (OCS), 2011 Sixth*. IEEE, 2011.

[8] A. Gupta, L. V. Kale, D. Milojicic, P. Faraboschi, and S. M. Balle, "Hpc-aware vm placement in infrastructure clouds," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. IEEE, 2013, pp. 11–20.

[9] A. Tsitsipas, C. B. Hauser, J. Domaschka, and S. Wesner, *Towards Usage-Based Dynamic Overbooking inIaaS Clouds*. Cham: Springer International Publishing, 2017, pp. 263–274.

[10] R. F. Da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and M. Livny, "Toward fine-grained online task characteristics estimation in scientific workflows." in *WORKS@ SC*, 2013, pp. 58–67.

[11] F. Caglar, S. Shekhar, A. Gokhale, and X. Koutsoukos, "Intelligent, performance interference-aware resource management for IoT cloud backends," *Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pp. 95–105, 2016.

[12] C. B. Hauser and S. R. Palanivel, "Dynamic network scheduler for cloud data centres with sdn," in *Proceedings of the10th International Conference on Utility and Cloud Computing*, ser. UCC '17. New York, NY, USA: ACM, 2017, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/3147213.3147217

[13] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, and M. Schtz, "Molpro: a general-purpose quantum chemistry program package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 2, no. 2, pp. 242–253, 2012.

[14] M. Schütz and H.-J. Werner, "Low-order scaling local electron correlation methods. iv. linear scaling local coupled-cluster (lccsd)," *The Journal of Chemical Physics*, vol. 114, no. 2, pp. 661–681, 2001.

[15] H. Nawaz, G. Juve, R. F. Da Silva, and E. Deelman, "Performance analysis of an i/o-intensive workflow executing on google cloud and amazon web services," in *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, 2016, pp. 535–544.

[16] A. Bhattacharyya, S. Sotiriadis, and C. Amza, "Online phase detection and characterization of cloud applications," in *Cloud Computing Technology and Science (CloudCom), 2017 IEEE 9th International Conference on Cloud Computing Technology and Science*, Dec 2017.

[17] C. Hauser and J. Domaschka, "Vice registry: An image registry for virtual collaborative environments," in *Cloud Computing Technology and Science (CloudCom), 2017 IEEE 9th International Conference on Cloud Computing Technology and Science*, Dec 2017.