TimeCap: Methodology for comparing IT infrastructures based on time and capacity metrics

Toni Mastelic, Ivona Brandic
Institute of Information Systems, Vienna University of Technology, Vienna, Austria
{toni, ivona}@infosys.tuwien.ac.at

***Abstract*—Scientific community is one of the major driving forces for developing and utilizing IT technologies such as Supercomputers and Grid. Although, the main race has always been for bigger and faster infrastructures, an easier access to such infrastructures in recent years created a demand for more customizable and scalable environments. However, introducing new technologies and paradigms such as Cloud computing requires a comprehensive analysis of its benefits before the actual implementation.**

In this paper we introduce the TimeCap, a methodology for comparing IT infrastructures based on time requirements and resource capacity wastage. We go beyond comparing just the execution time by introducing the Time Complexity as part of TimeCap, a methodology used for comparing arbitrary time related tasks required for completing a procedure, i.e., obtaining the scientific results. Moreover, a resource capacity wastage is compared using the Discrete Capacity, a second methodology as part of TimeCap used for analyzing resource assignment and utilization. We evaluate our methodology by comparing a traditional physical infrastructure and a Cloud infrastructure using our local IT resources. We use a real world scientific application for calculating plasma instabilities for analyzing time and capacity required for computation.

I. Introduction

A scientific community has always required state of the art computing infrastructures in order to benefit from rapidly advancing technologies. Today these infrastructures are based on large-scale distributed systems such as TeraGRID [1] used by 4000 users around 200 universities, and Open Science Grid[2] hosting up to 25000 machines used for research work in molecular bioscience, mathematics, neuroscience, physics etc. Implementing new paradigms such as Cloud computing on large IT infrastructures obviously requires analysis of the current state and of the benefits that a new approach would bring.

Comparing IT infrastructures used by scientific community is usually all about performance [3], [4], [5], and thus notion of Cloud with its virtualization layer becomes undesirable due to a performance loss. However, besides prolonged execution time, there are other time consuming prerequisites that a scientist has to do before actually executing his application [6], e.g., some scientific applications may not fit existing execution environments and many of them may require complete redesigning and reprogramming in order to be run [7]. Even after successfully adapting the application, due to constant system upgrades, hardware changes, etc., further adaptation may still be required [8]. Since reprogramming a modern scientific application represents a complex task [8], this can create even bigger time overhead than the one of a prolonged execution time. Additionally, scaling up or down these execution environments also takes time since each node has to be reconfigured separately.

Another aspect that is usually overlooked is resource assignment and utilization. IT capacity of a scientific cluster is rarely fully utilized [9], while job queues are commonly long [9]. This is usually due to unfair resource assignment where certain applications do not utilize all the reserved resources, while other applications are waiting in the queue [10].

In this paper we introduce the TimeCap, a methodology for comparing IT computing infrastructures considering two basic aspects important for scientific community: **time** and **capacity**. For comparing time requirements we present the Time Complexity (TC) methodology, a novel approach for comparing time related tasks based on *story points* [11], a methodology used by agile development teams for estimating project complexity. In order to compare a *story points* abstract metric and execution time we introduce a mapping function that provides a generic metric for comparing time related tasks. Moreover, we add possibility of modeling task parallelization using the Amdahl's law [12]. For analyzing IT capacity utilization and assignment, we present the Discrete Capacity (DC) methodology. DC methodology introduces intuitive resource slicing approach that divides IT resources to smallest usable configurations called *instances*. Based on these *instances*, a resource assignment and utilization is analyzed.

We evaluate our TimeCap methodology by setting up and comparing a traditional physical infrastructure with a Cloud infrastructure using our local IT resources. As a testing application we utilize a real-world scientific application for calculating Weibel instabilities within a quark-gluon plasma during particle collision like those in Large Hardon Collider at CERN [13].

The rest of this paper is organized as follows. Section II discusses related work. Section III provides two use cases describing time and capacity issues when using a certain type of infrastructure. Sections IV and V present two novel comparison methodologies, the Time Complexity and the Discrete Capacity methodology. Infrastructures and the execution environment, along with monitoring tools are described in Section VI, while the evaluation of our approach using a real-world scientific application is presented in Section VII. Finally, Section VIII gives the conclusion and the future work.

II. Related Work

In [14], a comparison is made between in-house clusters and clouds. However, the authors focus on public cloud, specifically Amazon Cluster Compute Instances. Therefore, aspects such as resource assignment and utilization are neglected, as well as setup of the infrastructure. In [15], the authors compare costs and performance issues when using a cloud versus physical infrastructure. However, their work focuses only on data-intensive applications using Google App Engine, so they do not consider setup procedure nor resource utilization. In [16], the authors analyze the performance in a Xen-based virtual cluster environment. They consider resource consumption and introduce a model for measuring the performance overhead for network latency and bandwidth. However, utilization and assignment of these resources is not covered. Resource consumption overhead of virtualization technology is evaluated in [17], and the performance in [5]. The authors compare different hypervisors against bare metal execution by benchmarking the three environments. However, only performance and resource overheads are measured, while resource dependance are neglected. The authors in [18] compare different public clouds with a local infrastructure focusing on deployment, performance and cost efficiency. They measure deployment time only for public clouds without comparing them to a local setup, since their measurement is based on actual time. Also, they do not compare all time related tasks such as execution time. Moreover, resource utilization is neglected. [19] presents a methodology for trade-off analysis when moving scientific applications to Cloud based on four models: complexity, performance, resource and energy. However, models do not provide a methodology for comparing setup procedure and the execution time, nor do they consider dependance between the resource types.

III. Use cases

We consider two use cases concerning two major issues when executing a scientific application on an IT infrastructure.

TIME use case: Scientists A and B want to deploy their applications on an already existing execution environment (i.e., Grid) that is usually implemented as bag of tasks (BoTs), workflows or Message Passing Interface (MPI) applications, as shown in Figure 1.

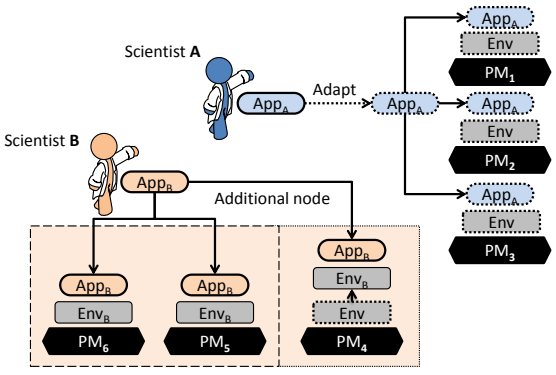

Fig. 1. Time consuming tasks required before executing an application.

Scientist A has developed his application targeting the existing environment, and thus only has to deploy it on an infrastructure. However, due to some changes and updates on the environment, scientist A will still have to adapt his application in order to run it.

On the other hand scientist B requires completely different execution environment. Instead of rewriting his application to fit the existing environment, the scientist decides to create a new execution environment spreading over dozens of machines. Obviously, he has to configure all the machines to support his application. However, in case he has to scale up or down his environment, he has to manually reconfigure all extra machines. This also includes software upgrades that have to be done on each node separately as well.

Utilizing Cloud technologies such as virtualization would provide standard execution environments such as one used by scientist A, as well as fully customizable environments fitted for a specific application required by scientist B, where no adaptations or compensations are required. Both environments would be extremely scalable, simply by shutting down existing and deploying new virtual machines [20]. Finally, managing an environment would require managing a single virtual image. However, due to a virtualization overhead, a scientist can expect longer execution times, where he can perhaps lose all the benefits of a quicker deployment.

CAPACITY use case: A scientist A wants to deploy his application on a cluster composed of 4 servers with 4 cores and 8 GB each, as shown in Figure 2. In the meanwhile, scientists B and C are already running their applications on the cluster. On the one hand scientist B is utilizing 2 machines at 100% of the cpu and only 6 GB of memory per each machine. On the other hand, scientist C is utilizing only 3 out of 4 cores on each of those two machines, and also only 50% of memory due to limitations of his application. Although, the cpu utilization of the cluster is only 75% with 4 unused cores, and only 62,5% of memory that equals to 12 GB of unused memory, scientist A is unable to run his application as there are no free machines.

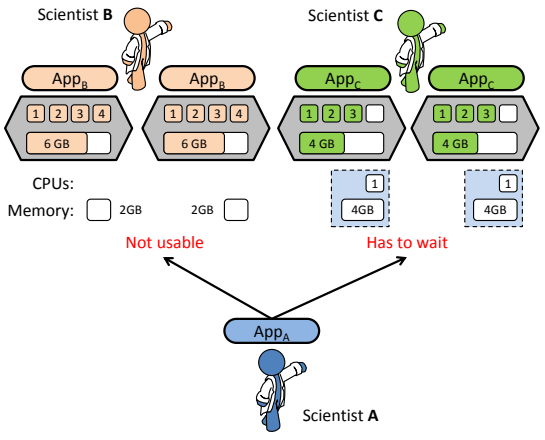

Fig. 2. Resource assignment use case.

Better assignment of these resources could be done using a virtualization, where each scientist could be assigned a

TABLE I
COMPLEXITY SCALE FOR GRADING TASKS, FOLLOWED BY MAPPED GRADES DEPENDING ON UNCERTAINTY FACTOR x.

Task	Complexity
Register on the site	13
Login to the site	2
Configure your profile and design	50

perfect-fit size virtual machines [21]. Not only that a resource utilization would improve, it would also improve resource assignment and lower the waiting queues. However, if applications are optimized for a targeted environment such as the application of the scientist B, and thus use almost all the resources reserved for them, resource consumption overhead due to a virtualization layer can finally lead to an even greater resource wastage. Additionally, it is incorrect to assume that by running two additional VMs in order to use those 4 GB that are currently not being used by the scientist B is of no purpose, since no spare cpu cycles are available that could utilize those RAMs.

IV. TIME COMPLEXITY

Traditional approach when estimating time needed for performing scientific tests usually only includes execution time of the application. However, as shown in the *Time use case*, time required to obtain scientific test results should also include all prerequisites before the actual execution, such as application development or adaptation, input data formatting, environment setup, etc. Prerequisites will depend on a scenario.

In order to support and compare different scenarios, an unified metric has to be used that is able to describe all the tasks required for obtaining scientific results. Thus, we present the Time Complexity (TC) methodology based on the *story points* [11], a methodology widely used in agile software development for estimating project complexity. *Story points* break the procedure (i.e., a story) to a number of points, which we refer to as tasks. Each task is assigned with a complexity grade that abstractly describes the (1) complexity, (2) effort and (3) uncertainty for executing a certain task. Grades are represented with integers defined relatively to one another and are defined by a team making the estimate. Moreover, *story points* guidelines suggest that grades should have a steep increase, since higher grades assume more (1) complex tasks that require more (2) effort and finally have more (3) uncertain deadline. For example, if we have *story points* grades 2, 13 and 50, we can use them to relatively describe complexity of tasks listed in Table I.

However, *story points* provides an abstract comparison metric defined as a task complexity that cannot be directly compared with execution time, which is on the other hand measured in seconds. Additionally, some tasks can be done in parallel, and thus would require less time than initially shown by a *story point* complexity metric. Both issues are discussed and solved in following two subsections IV-A and IV-B.

A. Mapping function

In order to compare task's complexity with time, a mapping function is required that can map time deadline with a specific complexity grade, while still preserving the characteristics of a *story points*.

Table II in column *real time* contains complexity scale defined by following the guidelines of a *story points* methodology previously described. However, instead of choosing arbitrary complexity grades, we built a complexity scale using number of minutes, followed by a user-friendly naming convention as shown by column *name* in Table II. Number of minutes for a grade is defined as a maximum number of minutes until the next grade, e.g., *hours* is set to 1440 minutes or 24 hours, which equals to one day that is represented by a following grade *days*. *Days* is defined as 10080 minutes or 7 days, which equals to one week that is represented by a following grade *weeks*, and so fort. This way, when a certain task takes, for example, 5 hours, it is referred to as *hours*, which follows the standard use of an English language.

With such direct mapping the grade's characteristics (1) complexity and (2) effort are preserved. However, (3) uncertainty is lost since we can now exactly determine a deadline for a certain task since grades are directly mapped to time. In order to preserve (3) uncertainty of a *story points* grades, but still be able to map them with time, two goals have to be achieved: (a) lower resolution of the grades, i.e., one grade should map to several time moments, i.e., a time interval; (b) those time intervals should overlap following the nature of *story points* methodology, i.e., distance between the grades should tend to be smaller.

$$c_t = x^{log(time_t)} \tag{1}$$

We achieve this by introducing a mapping function (1) with a customizable uncertainty factor x ranging from 10 to 1, where mapped values are referred to as TC grades and are shown in Table II for our complexity scale. 10 represents a direct one-to-one mapping between TC grades and *real time*, which indicates complete certainty, e.g., if we are traveling by hypothetically most punctual train, then a time distance between two stations could be graded with the TC grade 2 if the real time is actually two hours.

By reducing the uncertainty factor x, resolution of the TC grades goes down and thus a single TC grade maps to a time interval due to rounding of a TC grade as an integer. Additionally, distance between the TC grades becomes smaller as shown in Figure 3 as the uncertainty factor x gets reduced.

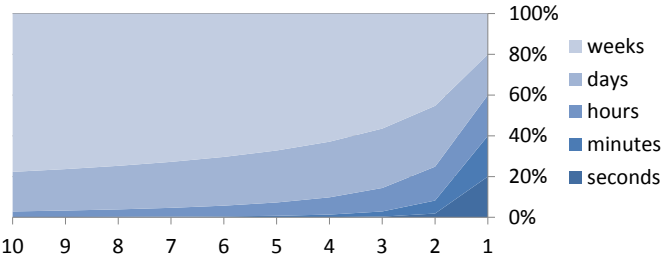

Fig. 3. By reducing the uncertainty factor x TC grades start to overlap by becoming more and more closer to each other.

TABLE II
COMPLEXITY SCALE FOR GRADING TASKS, FOLLOWED BY MAPPED VALUES, I.E., TC GRADES WITH UNCERTAINTY FACTOR x.

		TC grades with uncertainty factor x									
Name	Real time (min)	x = 10	x = 9	x = 8	x = 7	x = 6	x = 5	x = 4	x = 3	x = 2	x = 1
seconds	1	1	1	1	1	1	1	1	1	1	1
minutes	60	60	50	40	32	24	17	12	7	3	1
hours	1440	1440	1032	712	467	287	161	80	32	9	1
days	10080	10080	6611	4126	2417	1304	628	257	81	16	1
weeks	40320	40320	24819	14428	7800	3835	1656	593	158	24	1
months	483840	483840	265818	136081	63699	26519	9407	2646	516	51	1

Therefore uncertainty factor 1 would be all-to-one mapping, where all grades have the same value and thus represent complete uncertainty, e.g., trying to compare building construction with searching for a new planet in the Solar system. Although, both tasks could be underway, uncertainty of the latter makes it impossible to predict which one will finish sooner since comparison is relative to one another. Therefore, uncertainty factor x is always selected for the most uncertain task in the process.

Since TC methodology is still based on *story points*, it is important to notice that the final output of the mapping function is not actual time, but rather an abstract metric referred to as TC (time complexity). Thus, it is not advisable to use the TC methodology to estimate time needed to complete certain tasks, but only to compare different approaches that relate to some deadlines.

B. Parallelization

Calculating a TC of a task t that has to be repeated n times can be done using the Equation (2).

$$C_t = n * c_t \tag{2}$$

However, if a task can be partly parallelized, than the TC output of the Equation (2) is not realistic. In order to describe scenarios where parts of a task can be parallelized, we extend the Equation (2) using Amdahl's law [12], which states that in parallelization, if P is the proportion of a system or a program that can be made parallel, and $1-P$ is the proportion that remains serial, then the maximum speedup that can be achieved using N number of processors is equal to Equation (3).

$$\frac{1}{(1-P) + \frac{P}{N}} \tag{3}$$

Although this law relates to systems and programs, it can be used in our scenario as well. After defining parallelization factor P for a certain task t with time complexity c_t, total time complexity C_t for performing this task n times would be equal to Equation (4), where N from Equation (3) is equal to n, assuming that all tasks can be parallelized at the same time.

$$C_t = ((1-P) + \frac{P}{n}) * n * c_t \tag{4}$$

C. Usage

TC methodology can be used for arbitrary scenarios where a procedure can be broken into smaller serial tasks, where repeatable tasks may or may not be parallelized. More generally, TC methodology can be used for any scenario where relative time comparison between two approaches is necessary.

For the purpose of comparing different IT infrastructures for executing scientific applications, additional overheads are included such as the execution time overhead of the "slower" infrastructure, e.g., if an execution time on an infrastructure A is 168 hours (7 days), and on an infrastructure B is 170 hours (7 days and 2 hours), both execution times would get the same TC grade *days* from Table II. Thus, it would look like that both execution times are equal. For this reason, only an overhead is taken that equals 2 hours for the infrastructure B, and thus infrastructure B has the additional overhead in range of *hours*.

V. DISCRETE CAPACITY

When it comes to the IT resource consumption, common review of virtualization technology usually focuses only on a resource consumption overhead created by a hypervisor. Moreover, unused resources of physical machines described in our *Capacity use case* are usually overlooked and not considered as wasted, while benefits of virtualization technology such as increased utilization due to a better resource assignment possibilities are somehow neglected.

We present Discrete Capacity (DC) methodology that takes these issues into account and provides more realistic view of resource usage and utilization. DC methodology considers utilization U of a specific resource as a ratio of used resources R_{used} and reserved resources $R_{reserved}$ as defined by the Equation (5) [19].

$$U = \frac{R_{used}}{R_{reserved}} \tag{5}$$

However, in order to model dependence between resource types described in the *Capacity use case* in Section III, a systematic approach is required on based which a utilization U can be calculated, as well as the resource wastage.

A. Discretization

DC methodology slices down IT resources to a smallest usable configurations called *instances* composed of four main type of resources: cpu, memory, disk and network, similar to Amazon instances [22]. Figure 4 shows example of how IT

equipment composed of 2 physical servers could be sliced if a) bare physical machines are used, and if b) a Cloud paradigm is applied. A single instance represents a reserved amount of all types of resources for a single application, assuming that resource reservation is done on an instance basis.

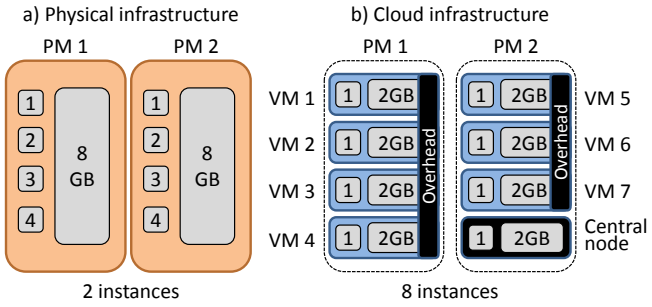

Fig. 4. Slicing the IT capacity to a discrete instances.

B. Wastage

As shown in Figure 4, the Cloud scenario includes resource consumption overhead on all instances due to a hypervisor (e.g., Domain 0 in Xen hypervisor [23]) and virtualization overhead, as well as the central node running Cloud management system (e.g., OpenStack central node [24]). Combining the data on overheads and the resource utilization we can calculate resource wastage W using the Equation (6).

$$W = 1 - \frac{R_{used} - R_{overhead}}{R_{reserved}} \tag{6}$$

C. Usage

DC methodology is used for comparing IT resource assignment and wastage between different infrastructures. After slicing the resources to a smallest usable *instances* for a targeted infrastructure, *instances* are assigned to users and/or different execution environments. Applications are then analyzed for their resource consumption and matched to the instances in order to estimate resource utilization. Finally, using the utilization data and resource overheads, total resource wastage is calculated using Equation (6).

VI. Infrastructure

Evaluation of our TimeCap methodology is done by comparing the physical and Cloud infrastructure using our local cluster (called Haley) consisting of 8 machines listed in Table III. We focus here only on cpu and memory resources, but the same approach applies on network and disk resources. Both for physical and virtual machines we use Ubuntu Server 12.04 [25] as an operating system. Futhermore, for cloud infrastructure we use OpenStack[24] as a cloud management system and Xen [23] for a virtualization layer, which uses a paravirtulization technology. Xen is automatically deployed on all physical nodes using a dodai-deploy puppet [26].

Requirements over the IT infrastructure are given by the groups using the cluster and are listed in Table IV. Student group uses their resources only during a semester, while all other groups use their resources all the time. Our evaluation follows the Physics department group that will be running their application on the Haley cluster.

TABLE III
Haley cluster.

No.	CPU model	Cores	Clock (GHz)	RAM (GB)
1, 2	Intel Xeon E31240	4	3,3	4
3, 4	Intel Xeon X3430	4	2,4	8
5, 6	Intel Xeon E31220	4	3,1	8
7, 8	AMD Opteron 4130	4	2,6	8

A. M4Cloud monitoring tool

For monitoring resource and energy consumption we use an improved version of our M4Cloud monitoring tool [27] with an agent-server architecture. It consists of three main components as shown in Figure 5. Application Deployer is used for deploying applications and plugins for additional metrics. It also registers a deployed application within a monitoring system. Metric Plugin Container supports the concept of a dynamic plugin loader for additional metrics. It is implemented within a lightweight agent running on a monitored machine, while server runs on a separate machine. Application Level Monitoring component serves as a central component and is implemented within a server. It manages all the agents and stores the metric data within a database.

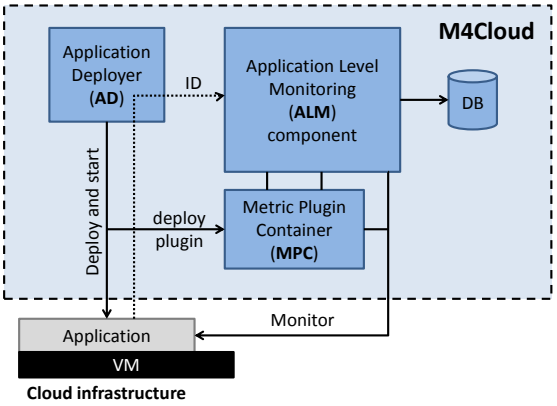

Fig. 5. Architecture of the M4Cloud monitoring tool showing its components.

B. Quark-gluon plasma simulator

In order to correctly asses what type of execution environment should be set up on an IT infrastructure, it is important to understand the types of applications that will be running on it. Here we focus on a single scientific application used for creating a real-world scenario based on our use cases in Section III. We explain the implementation and the attributes of our test application for simulating instabilities of a specific plasma type.

The simulation [28] is based on Boltzmann-Vlasov equations for color electric and magnetic fields. The evolution of instabilities is solved in three dimensional configuration space which is discretized on a grid. The contribution of particle distributions is furthermore discretized for each grid point in a two-dimensional velocity space. Therefore the simulation

TABLE IV
RESOURCE REQUIREMENTS BY GROUPS USING THE CLUSTER.

Group	Requirements	Instances
Students	At least four instances for testing distributed environment	4+
Scientist A	One instance for programming and compiling	1
Scientist B	One dedicated machine for energy consumption measurements	1 PM
Physics department	As much cpu cores as possible	1+

covers effectively a five-dimensional space which means that memory requirements scale accordingly with increasing system size. In order to deal with the large memory requirements, the application is parallelized using the MPI library. Execution time of the simulation depends on the number of steps the simulation should calculate before terminating, which is defined with a parameter $STEP$. For a comparison reference the $STEP$ parameter is set to provide an execution time of exactly 10 days when running on a faster infrastructure of those that are being compared. Additionally, size of the simulated system is defined with the internal parameter $NUMT$. Number if MPI workers depends directly on this parameter, where number of workers w should not exceed $NUMT + 1$. We use $NUMT$ parameter for creating different application requirements/limitations.

VII. EVALUATION

A. Discrete capacity results

Based on the user requirements from Table IV, we assign resources as shown in Figure 6: a) on physical infrastructure students get their minimum of 4 instances as requested. Scientists A and B are assigned with machines 7 and 8 respectively. Finally, physics department is left with machines 5 and 6. b) on Cloud infrastructure students get machines 1 and 2, running total of 4 instances as requested. Scientist B is assigned with a full machine for energy consumption measurements, while scientist A gets one out of two instances on the machine 7. At last, physics department is left with total of 5 instances, 4 on machines 3 to 6, and a smaller one on machine 7. Latter is selected as a central node for OpenStack and thus won't be used for running the application.

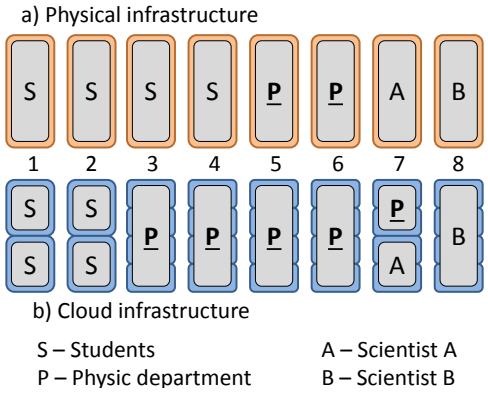

Fig. 6. Resource assignment of the Haley cluster.

It is important to notice that the student group and scientist A use their instances for programming and running non-intensive applications. Hence, their machines are always underutilized, below 40% for all types of resources on a bare physical machines. Therefore, there is no performance loss when sliced to smaller instances on Cloud infrastructure. Moreover, extra resource consumption overhead created by the central node on the Cloud infrastructure is avoided, since those resources were already unutilized on the physical infrastructure, and thus were already considered as wasted.

In order to evaluate resource wastage of the scientific application, internal parameter $NUMT$ described in Section VI-B is used. By changing this parameter, we created scenarios where a single instance can run 1 to 10 workers, going thus from the lowest to the highest utilization. Number of maximum 10 workers is obtained by evaluating the application's performance with different number of workers per machine/cpu. Both for physical and Cloud infrastructure the application performs best with 10 workers per PM, or 2.5 workers per cpu as shown in Figure 7. Obviously, although running in parallel, not all workers consume cpu cycles all the time.

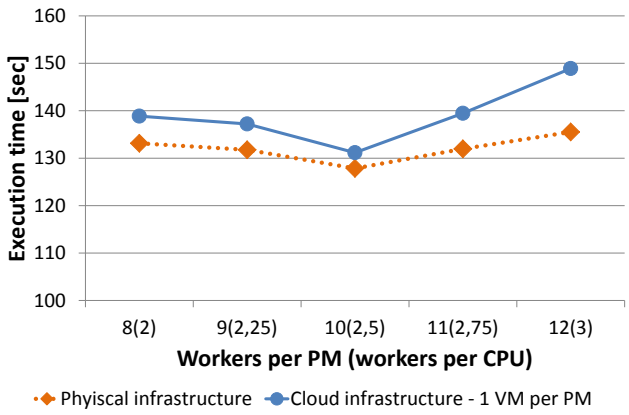

Fig. 7. Average execution time for different application parameters.

Figures 8 and 9 show different cpu and memory utilization scenarios for both infrastructures where one instance runs 1 to 10 MPI workers. Running the application in the smaller size instances using the virtualization is done so the performance is not affected, i.e., execution time on the Cloud infrastructure does not fall below reference times shown in Figure 7. Obviously, resource wastage is greater on physical infrastructure when there is lower resource demand (1 to 5 workers) due to a bigger instances, while the Cloud infrastructure is unable to provide full amount of the machine's capacity due to its virtualization overhead. However, in our case the application does not actually use the entire cpu or memory capacity, thus resource wastage is equal for both infrastructures even for a greater resource demand (6 to 10 workers).

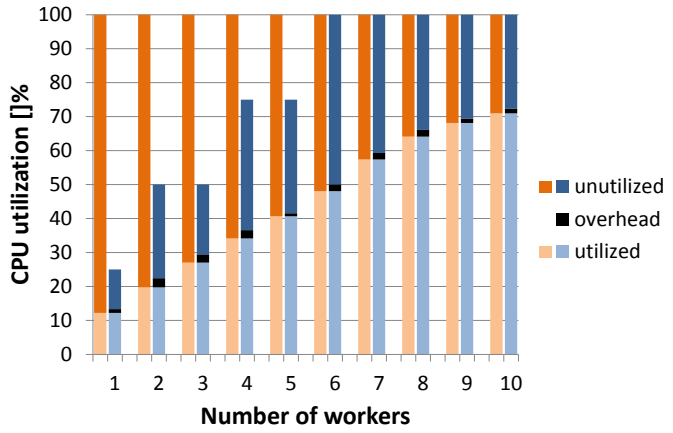

Fig. 8. CPU utilization and wastage for physical (left column) and cloud (right column) infrastructures.

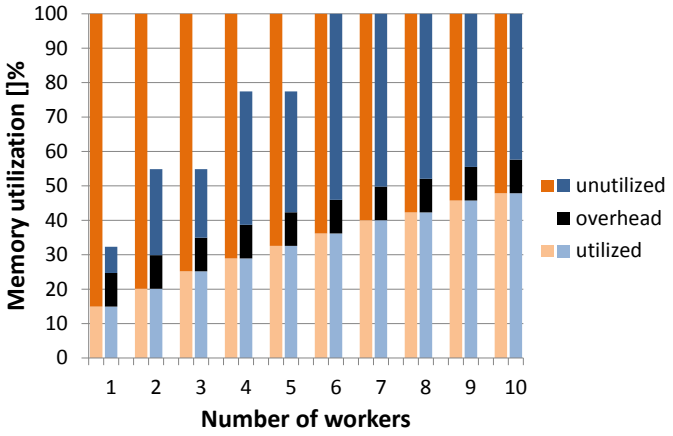

Fig. 9. Memory utilization and wastage for physical (left orange column) and cloud (right blue column) infrastructures.

B. Time complexity results

In order to compare which infrastructure requires more time to obtain simulation results, we must consider time required for setting up an infrastructure and the execution environment, as well as the execution time. Therefore, we consider two scenarios: (a) initial setup for the semester, and (b) reconfiguration after the semester.

(a) Initial setup is performed by slicing the resources as described in Section VII-A. Table V lists the tasks required for building both physical and Cloud infrastructure on these machines. As shown in Figure 11, initial setup of the Cloud infrastructure takes longer due to installation of CMS and hypervisor. However, as shown in Figure 10, Cloud infrastructure performes more than 2 times better than the physical infrastructure, due to a better resource slicing capabilities. Taking that the reference execution time is 10 days as defined in Section VI-B, execution time for physical infrastructure would be slightly over 20 days, which makes the overhead of 10 days. Applying the TC grade it would make it in range of *days*. Therefore, total time needed for obtaining simulation results was much shorter on a Cloud infrastructure as shown by Figure 11.

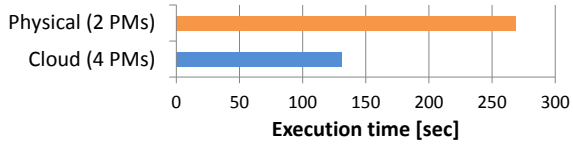

Fig. 10. Average execution time for different scenarios on both infrastructures.

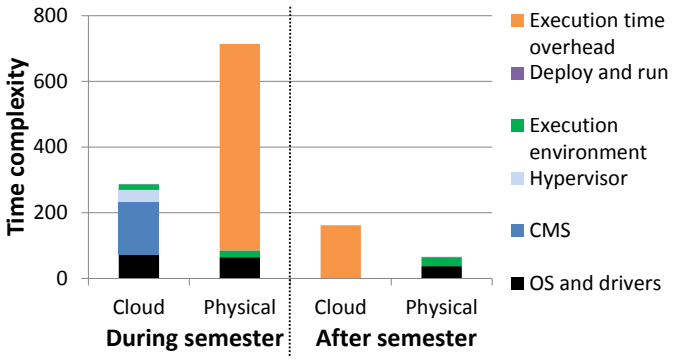

Fig. 11. Comparison of Time complexity for running the scientific application on the Haley cluster.

(b) In order to get the same amount of resources on a physical as on a Cloud infrastructure, physics department has to wait the end of the semester in order to use additional machines 1 to 4. However, before using them, they have to reconfigure them first, which includes tasks 2, 5 and finally 6 for running the application. On the other hand, on a Cloud infrastructure, scientists only have to shut down VMs on machines 1 to 4, and run already existing VM image, using a CMS from a central node, which requires a single task 6. However, when using the same amount of resources, a Cloud infrastructure gives 2.58% longer execution time than the physical infrastructure, as shown in Figure 7. With a reference execution time of 10 days, a Cloud infrastructure would provide execution time of 10 days and 6,19 hours, making a time overhead in range of *hours* by using TC grades. Due to a small number of instances that had to be reconfigured, total time required for obtaining simulation results was shorter on a physical infrastructure as shown in Figure 11. Obviously, for a bigger cluster where much larger number of instances would have to be reconfigured, setup time would exceed the execution time overhead of the Cloud infrastructure. On the other hand, for multiple experiments the execution time overhead would be multiplied by number of experiments made.

VIII. Conclusion and Future Work

In this paper, we introduced the TimeCap, a methodology for analyzing benefits and trade-offs of introducing new technologies and paradigms on an IT infrastructure. We focused on the time needed to obtain scientific results that includes all tasks prior to the actual execution of the application, as well as the execution time. We achieved this by developing a novel Time Complexity methodology that can be used for comparing

TABLE V
LIST OF TASKS AND THEIR TC GRADES WITH UNCERTAINTY FACTOR $x = 5$, INCLUDING PARALYZATION FACTOR P. TOTAL TIME COMPLEXITIES ARE SHOWN FOR TWO SCENARIOS FOR BOTH INFRASTRUCTURES.

Uncertainty factor $x = 5$

Task	P	c_t	During semester				After semester			
			Cloud		Physical		Cloud		Physical	
			n	C_t	n	C_t	n	C_t	n	C_t
OS and drivers	0,6	17	9	71,4	8	64,6	-	-	4	37,4
Hypervisor	0,8	17	7	37,4	-	-	-	-	-	-
Cloud management system	0	161	1	161	-	-	-	-	-	-
Execution environment	0	17	1	17	2	20,4	-	-	4	27,2
Deploy and run	1	1	4	1	1	1	2	1	1	1
Execution time overhead	-	-	-	-	1	628	1	161	-	-
TOTAL			287,8		714		162		65,6	

any time related tasks. By customizing the time complexity scale, TC methodology can be adapted to any scenario. Additionally, we introduced a Discrete Capacity methodology for analyzing IT resource assignment and utilization. As we demonstrated on a single application, the same approach would be used for plethora of applications using a statistical data of the cluster.

For our future work we plan to extend the methodology by adding a cost analysis of different approaches, including public clouds. This would provide a comprehensive methodology that could be used for general purposes, and not only for scientific applications.

REFERENCES

[1] C. Catlett, "TeraGrid: A Foundation for US Cyberinfrastructure," in *Network and Parallel Computing* (H. Jin, D. Reed, and W. Jiang, eds.), vol. 3779 of *Lecture Notes in Computer Science*, ch. 1, p. 1, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2005.

[2] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Wrthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, "The open science grid," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012057, 2007.

[3] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar, "Exploring the performance fluctuations of hpc workloads on clouds," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 383 –387, 30 2010-dec. 3 2010.

[4] M. Ahmadi and D. Maleki, "Performance evaluation of server virtualization in data center applications," in *Telecommunications (IST), 2010 5th International Symposium on*, pp. 638 –644, dec. 2010.

[5] A. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, and G. Fox, "Analysis of virtualization technologies for high performance computing environments," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 9 –16, july 2011.

[6] P. Kang, E. Tilevich, S. Varadarajan, and N. Ramakrishnan, "Maintainable and reusable scientific software adaptation: democratizing scientific software adaptation," in *Proceedings of the tenth international conference on Aspect-oriented software development*, AOSD '11, (New York, NY, USA), pp. 165–176, ACM, 2011.

[7] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pp. 4 –16, dec. 2009.

[8] B. Konning, C. Engelmann, S. Scott, and G. Geist, "Virtualized environments for the harness high performance computing workbench," in *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, pp. 133 –140, feb. 2008.

[9] The National Energy Research Scientific Computing Center (NERSC), "NERSC - cluster statistics, utilization graphs." http://www.nersc.gov/users/computational-systems/pdsf/cluster-statistics/utilization-graphs/, 2013.

[10] G. Stiehr and R. Chamberlain, "Improving cluster utilization through intelligent processor sharing," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, p. 8 pp., april 2006.

[11] M. Cohn, *Agile Estimating and Planning*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[12] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), (New York, NY, USA), pp. 483–485, ACM, 1967.

[13] CERN, "LHC - large hadron collider.." http://public.web.cern.ch/public/, 2012.

[14] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen, "Cloud versus in-house cluster: Evaluating amazon cluster compute instances for running mpi applications," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pp. 1 –10, nov. 2011.

[15] A. Angabini, N. Yazdani, T. Mundt, and F. Hassani, "Suitability of cloud computing for scientific data analyzing applications; an empirical study," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, pp. 193 –199, oct. 2011.

[16] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, pp. 273 –280, sept. 2010.

[17] Y. Jin, Y. Wen, and Q. Chen, "Energy efficiency and server virtualization in data centers: An empirical investigation," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 133 –138, march 2012.

[18] E. Roloff, M. Diener, A. Carrisimi, and O. A. N. Philippe, "High performance computing in the cloud: Deployment, performance and cost efficiency," in *CloudCom 2012, IEEE 4th International Conference on Cloud Computing Technology and Science*, 2012.

[19] T. Mastelic, D. Lucanin, A. Ipp, and I. Brandic, "Methodology for trade-off analysis when moving scientific applications to cloud," in *???*, 2012.

[20] Y. Simmhan, C. van Ingen, G. Subramanian, and J. Li, "Bridging the gap between desktop and the cloud for escience applications," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 474 –481, july 2010.

[21] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation in clouds," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 418 –425, july 2010.

[22] Amazon - Web Services, "Amazon ec2 instance types." http://aws.amazon.com/ec2/instance-types/, 2013.

[23] I. Citrix Systems, "Xen - hypervisor.." http://xen.org/, 2012.

[24] OpenStack, "OpenStack - open source software for building private and public clouds.." http://www.openstack.org/, 2012.

[25] Canonical, "Ubuntu - debian based linux distribution." http://www.ubuntu.com/, 2012.

[26] N. I. of Informatics Cloud Team, "dodai-deploy - software management tool.." https://github.com/nii-cloud/dodai-deploy/wiki, 2012.

[27] T. Mastelic, V. Emeakaroha, M. Maurer, and I. Brandic, "M4cloud - generic application level monitoring for resource-shared cloud environments," in *CLOSER 2012, 2st International Conference on Cloud Computing and Services Science*, 2012.

[28] A. Ipp, A. Rebhan, and M. Strickland, "Non-abelian plasma instabilities: Su(3) versus su(2)," *Phys. Rev. D*, vol. 84, p. 056003, Sep 2011.