ISSN:2320-0790

An International Journal of Advanced Computer Technology

Dynamic Resource Allocation Using Virtual Machines and Parallel Data Processing in the Cloud

Y.Bharath Bhushan[1], V.Bhavani[2]
Department of Computer Science and Engineering
Sri Venkateswara Engineering College, Suryapet

ABSTRACT: The main enabling technology for cloud computing is virtualization which generalize the physical infrastructure and makes it easy to use and manage. Virtualization is used to allocate resources based on their needs and also supports green computing concept. Parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) clouds. The processing frameworks which are currently used have been designed for static, homogeneous cluster setups and disregard the particular nature of a cloud. The allocated compute resources may be inadequate for big parts of the submitted job and unnecessarily increase processing time and cost.

In this paper we are applying the concept of **"SKEWNESS"** to measure the unevenness in the multi-dimensional resource utilization of a server. By minimizing skewness, we can combine different types of workloads and improve the overall utilization of server resources and discuss the opportunities and challenges for efficient parallel data processing in clouds using **"NEPHELE'S ARCHITECTURE"**. Nephel's architecture offers efficient parallel data processing in clouds. It is the first data processing framework for the dynamic resource allocation offered by today's IaaS clouds for both, task scheduling and execution

Index Terms- Cloud Computing, Parallel Data Processing, Dynamic resource allocation, High-Throughput Computing, Loosely Coupled Applications

1. INTRODUCTION

Cloud computing is the delivery of computing and storage capacity as a service to a community of end recipients. The name comes from the use of a cloud shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams.

1.1 Resource Allocation:

Cloud computing entrusts services with a user's data, software and computation over a network. The remote accessibility enables us to access the cloud services from anywhere at any time. To gain the maximum degree of the above mentioned benefits, the services offered in terms of resources should be allocated optimally to the applications running in the cloud. In this paper, we discuss how the cloud service provider can best multiplex the available virtual resources onto the physical hardware. And perform parallel data processing using Nephele's architecture. This is important because much of the touted gains in the cloud model come from such multiplexing. Virtual Machine Monitors (VMMs) like Xen provide a mechanism for mapping Virtual Machines (VMs) to Physical Resources [1]. This mapping is hidden from the cloud users. It is up to the Cloud Service Provider to make sure the underlying Physical Machines (PMs) has sufficient resources to meet their needs. VM live migration technology makes it possible to change the mapping between VMs and PMs While applications are running [2] This is challenging when the resource needs of VMs are heterogeneous due to the diverse set of applications they run and vary with time as the workloads grow and shrink. The capacity of PMs can also be heterogeneous because multiple generations of hardware co-exist in a data center. To achieve the overload avoidance that is the capacity of a PM should be sufficient to satisfy the resource needs of all VMs running on it. The two main goals that we achieve here is

1) The capacity of PM should be able to satisfy the needs of the VM's running. Thus we should maintain the utilization of PM's low as possible.
2) The number of PM's should be minimized. Thus in this case we have to maintain the utilization of Pm's high.

There is an in depth tradeoff between the two goals in the face of changing resource needs from all VMs. To avoid the overload, should keep the utilization of PMs Low to reduce the possibility of overload in case the resource needs of VMs increase later. For green computing, should keep the utilization of PMs reasonably high to make efficiency in energy. A VM Monitor manages and multiplexes access to the physical resources, maintaining isolation between VMs at all times. As the physical resources are virtualized, several VMs, each of which is self-contained with its own operating system, can execute on a physical machine (PM).

The three main contributions we have made in this paper for dynamic resource allocation are

1. To avoid the overload, we develop a resource allocation system is maintained thus by minimizing the total number of servers used.
2. To measure the utilization of the server we introduce concept "skewness" and by minimizing this we can find the utilization of the servers.
3. We also design a load prediction algorithm to encounter the future resource usages.

1.2 Parallel Data Processing:

Similarly today a growing number of companies have to process huge amounts of data in a cost-efficient manner. Classic representatives for these companies are operators of Internet search engines, like Google, Yahoo, or Microsoft. The vast amount of data they have to deal with every day has made traditional database solutions prohibitively expensive. Instead, these companies have popularized an architectural paradigm based on a large number of commodity servers. Problems like processing crawled documents or regenerating a web index are split into several independent subtasks, distributed among the available nodes, and computed in parallel. In order to simplify the development of distributed applications on top of such architectures, many of these companies have also built customized data processing frameworks. Examples are Google's MapReduce, Microsoft's Dryad , or Yahoo!'s Map-Reduce-Merge .They can be classified by terms like high throughput computing (HTC) or many-task computing (MTC), depending on the amount of data and the number of tasks involved in the computation [3]. For companies that only have to process large amounts of data occasionally running their own data center is obviously not an option. Instead, Cloud

computing has emerged as a promising approach to rent a large IT infrastructure on a short-term pay-per-usage basis. Operators of so-called Infrastructure-as-a-Service (IaaS) clouds, like Amazon EC2 [4], let their customers allocate, access, and control a set of virtual machines (VMs) which run inside their data centers and only charge them for the period of time the machines are allocated. The VMs are typically offered in different types, each type with its own characteristics (number of CPU cores, amount of main memory, etc.) and cost. Here i want to discuss the particular challenges and opportunities for efficient parallel data processing in clouds and present Nephele's, a new processing framework explicitly designed for cloud environments. Most notably, Nephele's is the first data processing framework to include the possibility of dynamically allocating/deallocating different compute resources from a cloud in its scheduling and during job execution. The paper is structured as follows: Section 2 starts with analyzing the above mentioned opportunities and challenges and derives some important design principles for our new framework. In Section 3 we present Nephele's basic architecture and outline how jobs can be described and executed in the cloud. Section 4 provides some first figures on Nephele's performance and the impact of the optimizations we propose.

2. RELATED WORK

In **resource allocation**, using feedback control theory, for adaptive management of virtualized resources, which is based on VM. In this VM-based architecture all hardware resources are pooled into common shared space in cloud computing infrastructure so that hosted application can access the required resources as per there need to meet Service Level Objective (SLOs) of application. The adaptive manager use in this architecture is multi-input multi-output (MIMO) resource manager, which includes 3 controllers: CPU controller, memory controller and I/O controller, its goal is regulate multiple virtualized resources utilization to achieve SLOs of application by using control inputs per-VM CPU, memory and I/O allocation. The seminal work of Walsh et al [3], Proposed a general two-layer architecture that uses utility functions, adopted in the context of dynamic and autonomous resource allocation, which consists of local agents and global arbiter. The responsibility of local agents is to calculate utilities, for given current or forecasted workload and range of resources, for each AE and results are transfer to global arbiter. The dynamic resource allocation based on distributed multiple criteria decisions in computing cloud explain in [6]. In it author contribution is tow-fold, first distributed architecture is adopted, in which resource management is divided into independent tasks, each of which is performed by Autonomous Node Agents (NA) in ac cycle of three activities: (1) VM Placement, in it suitable physical machine (PM) is found which is capable of running given VM and then assigned VM to that PM, (2) Monitoring, in its total resources use by hosted VM are monitored by NA, (3) In VM selection, if local accommodation is not possible, a VM need to migrate at another PM and process loops back to into placement and second, using PROMETHEE method,

For **Parallel Data Processing**, the Current data processing frameworks like Google's Map Reduce or Microsoft's Dryad engine have been de-signed for cluster environments. This is reflected in a number of assumptions they make which are not necessarily valid in cloud environments. In this section we discuss how abandoning these assumptions raises new opportunities but also challenges for efficient parallel data processing in clouds. Today's processing frameworks typically assume the resources they manage consist of a static set of homogeneous compute nodes. Although designed to deal with individual nodes failures, they consider the number of available machines to be constant, especially when scheduling the processing job's execution. While IaaS clouds can certainly be used to create such cluster-like setups, much of their flexibility remains unused.

One of an IaaS cloud's key features is the provisioning of compute resources on demand. New VMs can be allocated at any time through a well-defined interface and become available in a matter of seconds. Machines which are no longer used can be terminated instantly and the cloud customer will be charged for them no more. Moreover, cloud operators like Amazon let their customers rent VMs of different types, i.e. with different computational power, different sizes of main memory, and storage. Hence, the computer resources available in a cloud are highly dynamic and possibly heterogeneous.

Facilitating use cases imposes some requirements on the design of a processing framework and the way its jobs are described. First, the scheduler of such a frame-work must become aware of the cloud environment a job should be executed in. It must know about the different types of available VMs as well as their cost and be able to allocate or destroy them on behalf of the cloud customer.

Second, the paradigm used to describe jobs must be powerful enough to express dependencies between the different tasks the jobs consist of. The system must be aware of which task's output is required as another task's input. Otherwise the scheduler of the processing framework cannot decide at what point in time a particular VM is no longer needed and deallocate it. The Map Reduce pattern is a good example of an unsuitable paradigm here: Although at the end of a job only few reducer tasks may still be running, it is not possible to shut down the idle VMs, since it is unclear if they contain intermediate results which are still required.

The cloud's virtualized nature helps to enable promising new use cases for efficient parallel data processing. How-ever, it also imposes new challenges compared to classic cluster setups. The major challenge we see is the cloud's opaqueness with prospect to exploiting data locality:

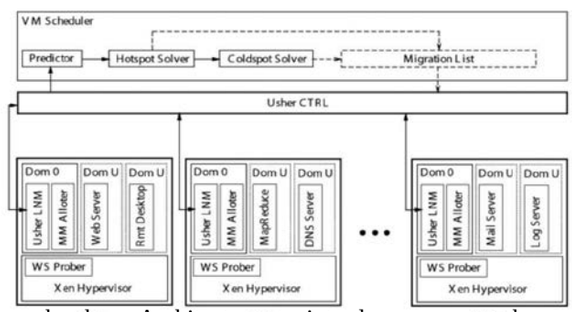
In a cluster the compute nodes are typically interconnected through a physical high-performance network. The topology of the network, i.e. the way the compute nodes are physically wired to each other, is usually well-known and, what is more important does not change over time. Current data processing frameworks offer to leverage this knowledge about the network hierarchy and attempt to schedule tasks on compute nodes so that data sent from one node to the other has to traverse as few network switches as possible [9]. That way network bottlenecks can be avoided and the overall throughput of the cluster can be improved.

In a cloud this topology information is typically not exposed to the customer [29]. Since the nodes involved in processing a data intensive job often have to transfer tremendous amounts of data through the network, this drawback is particularly severe; parts of the net-work may become congested while others are essentially unutilized. Although there has been research on inferring likely network topologies solely from end-to-end measurements (e.g. [7]), it is unclear if these techniques are applicable to IaaS clouds. For security reasons clouds often incorporate network virtualization techniques (e.g. [8]) which can hamper the inference process, in particular when based on latency measurements.

3. DESIGN & ARCHITECTURE

This proposed system, Set of servers used for running different applications. Predictor is used to execute periodically to evaluate the resource allocation status based on the predicted future demands of virtual machines.

3.1 SKEWNESS ALGORITHM:



From the above Architecture, we introduce a concept skewness which would be useful to measure the variable utilization of the server. By minimizing skewness we can find the various utilization of the servers. Hot spot is a small area in which there

is relatively higher temperature than the surroundings. Cold spot is the area in which there is a decrease in ambient temperature. Here we use the hot spot and cold spot to just explain the way in which the green computing algorithm has been used .The threshold technology is thus maintained here to make it clearer. The overload avoidance and the green computing concept is being used to make the resource Management precise.

Our algorithm evaluates the allocation of resources based on the demands of VM. Here we define the server a hotspot and if the utilization exceeds the above the hot threshold then it symbolizes that the server is overloaded and Vm's are moved away. The temperature is zero when the server is not a hot spot. We define a cold spot when the utilization of the resources are below the clod threshold which indicates that the server is idle and it has to be turned off in order to save energy. This is done when mostly all servers are actively used below the green computing threshold else it is made inactive. Hot spot mitigation the sorted lists of hot spots are arranged in a order so that we can eliminate them else keep the temperature low. Our goal is to move away the VM's that can reduce the server's temperature. Among all we select the one which can reduce skewness. Green computing Green computing aims to attain economic viability and improve the way computing devices are used. It is the environmentally responsible and eco-friendly use of computers and their resources. When the resources utilization of servers are low in such cases they are turned off wherein we use this green computing algorithm. The very important challenge here is to reduce the number of actively participating servers. Thus we have to avoid oscillation in the system. Our algorithm is used when utilization of all active servers are below the green computing threshold. Dynamic resource management has become an active area of research in the Cloud Computing paradigm. Cost of resource varies significantly depending on configuration for using them. Hence efficient management of resource is of prime interest to both Cloud Provider and Cloud Users. The success of any cloud management software critically depends on the flexibility; scale and efficiency with which it can utilize the underlying hardware resource while providing necessary performance isolation. Successful resource management solution for cloud environments needs to provide a rich set of resource controls for better isolation, while doing initial placement and load balancing for efficient utilization of underlying resource. VM live migration is widely used technique for dynamic resource allocation in a virtualized environment. The process of running two or more logical computer system so on one set of physical hardware.

Let n be the number of resources we consider and r_i be the utilization of the i-th resource. We define the resource skewness of a server p as where 'r' is the average utilization of all resources for server 'p'

$$\text{Skewness (p)} = \sqrt{\Sigma^{n}_{i=1}[(ri/r)-1]^{2}}$$

Skewness algorithms consist of three steps:

1: Hotspot Migration

2: Green Computing

1. Hotspot Mitigation

We handle the hottest one first I.e. sort the list of hot spots in the system Otherwise, keep their temperature as low as possible. Our aim is to migrate the VM that can reduce the server's temperature. In case of ties, the VM whose removal can reduce the skewness of the server the most is selected. We first decide for each server p which of its VMs should be migrated away. Based on the resulting temperature we sort list the VMs of the server if that VM is migrated away. We see if we can find a destination server to accommodate it for each list of in the VM. After accepting this VM the server should not become hot spot. We select one skewness which can be reduced the most by accepting this VM among all servers. We record the migration of the VM to that server and update the predicted load of related servers when the destination server is found. Else we move on to the next VM in the list and try to find a destination server for it.

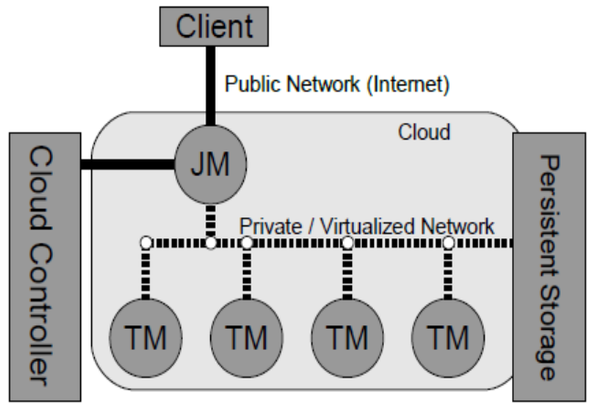
2 .Green Computing

When the resource utilization of active servers is too low, some of them can be turned off to save energy. This is handled in our green computing algorithm. Our green computing algorithm is invoked when the average utilizations of all resources on active servers are below the green computing threshold. We check if we can migrate all its VMs somewhere else for a cold spot p. For each VM on p, we try to find a destination server to accommodate it. The utilizations of resources of the server after accepting the VM must be below

The warm threshold. Section 7 in the supplementary file explains why the memory is a good measure in depth. We try to eliminate the cold spot with the lowest cost first. We select a server whose skewness can be reduced the most. If we can find destination servers for all VMs on a cold spot, we record the sequence of migrations and update the predicted load of related servers. Otherwise, we do not migrate any of its VMs.

3.2 NEPHELE'S ARCHITECTURE:

Nephele's architecture follows a classic master-worker pattern as illustrated below:



Structural overview of Nephele's running in an Infrastructure-as-a-Service (IaaS) cloud

Before submitting a Nephele's compute job, a user must start a VM in the cloud which runs the so called Job Manager (JM). The Job Manager receives the client's jobs, is responsible for scheduling them, and coordinates their execution. It is capable of communicating with the interface the cloud operator provides to control the instantiation of VMs. We call this interface the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or deallocate VMs according to the current job execution phase. We will comply with common Cloud computing terminology and refer to these VMs as instances for the remainder of this paper. The term instance type will be used to differentiate between VMs with different hardware characteristics. E.g., the instance type "m1.small" could denote VMs with one CPU core, one GB of RAM, and a 128 GB disk while the instance type "c1.xlarge" could refer to machines with 8 CPU cores, 18 GB RAM, and a 512 GB disk

The newly allocated instances boot up with a previously compiled virtual machine image. The image is con_gured to automatically start a Task Manager and register it with the Job Manager. Once all the necessary Task Managers have successfully contacted the Job Manager, it triggers the execution of the scheduled job. Initially, the virtual machines images used to boot up the Task Managers are blank and do not contain any of the data the Nephele job is supposed to operate on. As a result, we expect the cloud to over persistent storage (like e.g. Amazon S3 [3]). This persistent storage is supposed to store the job's input data and eventually receive its output data. It must be accessible for both the Job Manager as well as for the set of Task Managers, even if they are connected by a private or virtual network

3.3 SCHEDULING STRATEGIES:

The Basic idea to refine the scheduling strategy for recurring jobs is to use feedback data. We develop a system for Nephele's which continuously monitor's running tasks and the instances.

Based on the Java Management Extensions (JMX) the system is capable of breaking down its processing time that a task spends processing user code and the time it waits for data. With the collected data Nephele's is able to detect computational and I/O bottlenecks. The computational bottlenecks suggests that higher degree of parallelization for the tasks, I/O bottlenecks provides hints to switch to faster channel types and reconsider the instance. Then Nephele's generates a cryptographic signature for every task and recurring tasks can be identified and already recorded data can be exploited. Now, we use the profiling data to detect the bottlenecks and provide the user to choose annotations for the job A user can use the feedback to improve the job's annotations. In advanced versions of Nephele's, the system can automatically adjust to detected bottlenecks between continuous executions of the same job or at job's execution at runtime. The allocation time of cloud instances is determined by the start times of the subtasks, there are different strategies for deallocation. Nephele's can track the instances' allocation times. An instance of a each type in the execution stage is not immediately deallocated if same instance type is required in an upcoming execution Stage. So, Nephele's retains the instance allocated till the end of current lease period. If the preceding execution phase has begun before the end of the previous period, it is reassigned to an execution of preceding stage, else it deallocates early enough not to cause any additional cost.

3.4 ALGORITHMS:

Algorithm 1: Non - Preemptive Scheduling

Consider K accepted task in ready queue and the current time t

-Parameters

1. Accepted task in the queue level. Let {t1, t2, t3... tk} Ar be the arrival time AT[T= 1 to K].
2. Let currently running task may be at T=0. Show the task with T and the threshold value ThAT= A0.
3. Conditions the current job is in critical, then abort the execution of T0.
4. Otherwise new task enrolled in the end process.
5. Calculation of efficiency of task and reschedule the task based on the utility value and load into the ready queue.
6. Start the execution from T1. The utility value is less than the threshold value then removes the process from ready queue else the current process and start its execution

Description:

This scheduling algorithm works at scheduling points that include: the arrival of a new task, the completion of the current task and the critical point of the current task. In algorithm 1, when the time reaches the critical point of the current task, the current active task is immediately discarded and the task with the highest expected efficiency is selected to be executed. Upon the finish of the current task, the task with the highest expected efficiency is selected for execution. After the selection of the new task in both of the two cases, the expected efficiency for the rest of the tasks is recalculated. The tasks with the expected efficiency smaller than the threshold value are discarded.

Algorithm 2: Sort the Ready Queue based on Recalculated Expected Gain

1. Input: Let T_r= {t_1, t2... tk} be the accepted tasks in the ready queue, let $tri_{,i}$ = 1, ..., k represent their specific arrival times. Let current time be t and T0 be the task currently being executed.
2. Output: The list of tasks in the ready queue is given as T‟r= {T‟1, T‟2... T‟k} sorted based on their expected gain.
3. T start = expected finishing time of T0–t.
4. for i=0 to k do.
5. T‟ $_i$= T $_j$ where T $_j \in T_r$ is the task with the largest expected gain assuming it starts at Tstart.
6. Remove T_j from T_r.
7. Tstart = Tstart + expected execution time of T‟$_j$.
8. Calculate the following tasks expected utility at time Tstart.
9. End for

Description:

When a new job comes, it is first inserted at the head of the ready queue, assuming its expected starting time would be the expected finishing time of the current active task. Based on this starting time, we then can compare its expected utility with the rest of the tasks in the queue. If its expected utility is less than that of the one following it, we reinsert this job to the queue according to its new expected utility. We calculate the new expected utility according to Algorithm 2, by estimating its new expected starting time as the sum of the expected executing time of the leading tasks‟ in the ready queue. This procedure continues until the entire ready queue becomes a list ordered according to their expected utilities. We remove the ones with expected utility lower than the threshold.

The feasibility check is one more part deserves detail description. In this part, scheduling simulates the real execution sequence for the left tasks in ready queue and check following this sequence, if all of them can satisfy the requirement or not. The thing needs to be discussed is how to determine the sequence of the left tasks. From equation (1), (2) and (3), we can clearly see that the expected utility of running a task depends heavily on variable T, i.e., the time when the task can start. If we know the execution order and thus the expected starting time for tasks in the ready queue, we will be able to quantify the expected utility density of each task more accurately.

4. CONCLUSION

In this paper we have presented the design, implementation, and evaluation of a resource management system for cloud computing services. We use the skewness metric to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. Our algorithm achieves both overload avoidance and green computing for systems with multi resource constraints and also discussed the challenges and opportunities for efficient parallel data processing in cloud environments and presented Nephele's, the first data processing framework to exploit the dynamic resource provisioning offered by today's IaaS clouds. The performance evaluation gives a first impression on how the ability to assign specific virtual machine types to specific tasks of a processing job, as well as the possibility to automatically allocate/deallocate virtual machines in the course of a job execution, can help to improve the overall resource utilization and, consequently, reduce the processing cost. With a framework like Nephele's at hand, there are a variety of open research issues, which we plan to address for future workOur current profiling approach builds a valuable basis for this; however, at the moment the system still requires a reasonable amount of user annotations. In general, we think our work represents an important contribution to the growing field of Cloud computing

5. REFERENCES

[1] Amazon Web Services LLC. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/, 2009.

[2] Amazon Web Services LLC. Amazon Elastic MapReduce. http://aws.amazon.com/elasticmapreduce/, 2009.

[3] AmazonWeb Services LLC. Amazon Simple Storage Service. http://aws.amazon.com/s3/, 2009.

[4] D. Battr´e, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In SoCC '10: Proceedings of the ACM Symposium on Cloud Computing 2010, pages 119–130, New York, NY, USA, 2010. ACM.

[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib,S. Weaver, and J. Zhou. SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. Proc. VLDB Endow., 1(2):1265–1276, 2008.

[6] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," Proc. USENIX Ann. Technical Conf., 2005.M.Young, *The Techincal Writers Handbook.* Mill Valley, CA:University Science, 1989.

[7] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," Proc. IFIP/IEEE Int'l Symp.Integrated Network Management (IM '07), 2007.

[8] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," Proc. Symp. Networked Systems Design and Implementation (NSDI '07), Apr. 2007.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[10] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman,G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob,and D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Sci. Program.,13(3):219–237, 2005.

[11] T. Dornemann, E. Juhnke, and B. Freisleben. On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud. In CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 140–147, Washington, DC, USA, 2009. IEEE Computer Society.

[12] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. Intl. Journal of Supercomputer Applications, 11(2):115– 128, 1997.