Maximizing Utilization in Private IaaS Clouds with Heterogenous Load

Tomáš Vondra, Jan Šedivý
Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27
Prague, Czech Republic
vondrto6@fel.cvut.cz, sedivja2@fel.cvut.cz

***Abstract*—This document presents ongoing work on creating a computing system that can run two types of workloads on a private cloud computing cluster, namely web servers and batch computing jobs, in a way that would maximize utilization of the computing infrastructure. The idea stems from the experience with the Eucalyptus private cloud system, which is used for cloud research at the Dept. of Cybernetics. This cloud lets researchers use spare computing power of lab computers with the help of our in-house queue engine called Cloud Gunther. This application improves upon current practices of running batch computations in the cloud by integrating control of virtual machine provisioning within the job scheduler. In contrast to other similar systems, it was built with the capacity restrictions of private clouds in mind. The Eucalyptus system has also been evaluated for web server use, and the possibility of dynamically changing the number of servers depending on user demand, which changes throughout the day, has been validated. Although there are already tools for running interactive services in the cloud and tools for batch workloads, there is no tool that would be able to efficiently distribute resources between these two in private cloud computing environments. Therefore, it is difficult for the owners of private clouds to fully exploit the potential of running heterogenous load while keeping the utilization of the servers at optimal levels. The Cloud Gunther application will be modified to monitor the resource consumption of interactive traffic in time and use that information to efficiently fill the remaining capacity with its batch jobs, therefore raising the utilization of the cluster without disrupting interactive traffic.**

Keywords-Cloud Computing; Automatic Scaling; Job Scheduling; Real-time Infrastucture.

I. INTRODUCTION

According to Gartner [1], private cloud computing is currently at the top of the technology hype; but, its popularity is bound to fall due to general disillusionment.

Why? While the theoretical advantages of cloud computing are widely known – private clouds build on the foundations of virtualization technology and add automation, which should result in savings on administration while improving availability, they provide elasticity, which means that an application deployed to the cloud can dynamically change the amount of resources it uses, which is connected to agility, meaning that the infrastructure can be used for multiple purposes depending on current needs. Lastly, the cloud should provide self-service, so that the customer can provision his infrastructure at will, and pay-per-use, so he will pay exactly for what he consumed.

The problem is that not all of these features are present in current products that are advertised as private clouds. Specifically, this document will deal with the problem of infrastructure agility.

A private cloud can be used for multiple tasks, which all draw resources from a common pool. This heterogenous load can basically be broken down into two parts, interactive processes and batch processes. An example of the first are web applications, which are probably the major way of interactive remote computer use nowadays, the second could be related to scientific computations or, in the corporate world, data mining.

When building a data center, which of course includes private clouds, the investor will probably want to ensure that it is utilized as much as possible. The private cloud can help achieve that, but not when the entire load is interactive. This is due to the fact that interactive load depends on user activity, which varies throughout the day, as seen in Figure 1.

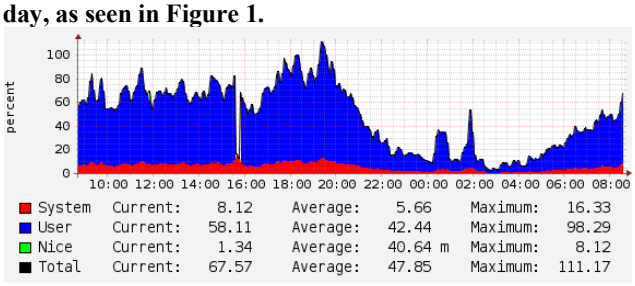

Figure 1. Daily load graph of an e-business website [2]

In our opinion, the only way to increase the utilization of a private cloud is to introduce non-interactive tasks that will fill in the white parts of the graph, i.e., capacity left unused by interactive traffic (which of course needs to have priority over batch jobs).

HPC (High Performance Computing) tasks are traditionally the domain of grid computing. Lately, however, they also began to find their way into the cloud. Examples may be Google's data mining efforts in their private cloud or Amazon's Elastic MapReduce public service [16]. The grid also has the disadvantage that it is only usable for batch and parallel jobs, not interactive use.

 ISBN: 978-1-61208-216-5

Currently, there is not much support for running of batch jobs on private clouds. The well known scheduling engines Condor [17] and SGE (Sun Grid engine) [18] both claim Amazon EC2 (Elastic Compute Cloud) [19] compatibility, they however cannot control the cloud directly, they only use resources provisioned by other means (See Section II.). (SGE seems to be able to control cloud instances in a commercial fork by Univa, though [3].)

That is why the Cloud Gunther project was started. It is a web application that can run batch parallel and pseudoparallel jobs on the Eucalyptus private cloud [4]. The program does not only run tasks from its queue; it can also manage the VM (virtual machine) instances the tasks are to be run on.

What the application currently lacks is support for advanced queuing schemes (only Priority FCFS (First Come First Served) has been implemented). Further work will include integration of a better queuing discipline, which will be capable of maximizing utilization of the cloud computing cluster by reordering the tasks as to reduce the likelihood of one task waiting for others to complete, while there are unused resources in the cluster, effectively creating a workflow of tasks (see Section IV).

The scheduler will be fed with data about the average amount of free resources left on the cluster by interactive processes. This will ensure that the cluster is always fully loaded, but the interactive load is never starved for resources.

This document has five sections. After Section I, Introduction, comes Section II, Related Work, which will present the state of the art in the area of grid schedulers and similar cloud systems. Section III, Completed Work, summarizes progress done in cloud research at the Dept. of Cybernetics, mainly the Cloud Gunther job scheduler. Section IV, Future Work, outlines the plans for expansion of the scheduler, mainly to accommodate heterogenous load on the cloud computing cluster. Section V, Conclusion, ends the paper.

II. RELATED WORK

As already stated, the most notable job control engines in use nowadays are probably SGE [18] and Condor [17]. These were developed for clusters and thus lack the support of dynamic allocation and deallocation of resources in cloud environments.

There are tools that can allocate a complete cluster for these engines, for example StarCluster for SGE [9]. The drawback of this solution is that the management of the cloud is split in two parts – the job scheduler, which manages the instances currently made available to it (in an optimal fashion, due to the experience in the grid computing field), and the tool for provisioning the instances, which is mostly manually controlled.

This is well illustrated in an article on Pandemic Influenza Simulation on Condor [10]. The authors have written a web application which would provision computing resources from the Amazon cloud and add them to the Condor resource pool. The job scheduler could then run tasks on them. The decision on the number of instances was however left to the users.

A similar approach is used in the SciCumulus workflow management engine, which features adaptive cloud-aware scheduling [11]. The scheduler can react to the dynamic environment of the cloud, in which instances can be randomly terminated or started, but does not regulate their count by itself.

The Cloud Gunther does not have this drawback, as it integrates job scheduling with instance provisioning. This should guarantee that there is no unused time between the provisioning of a compute resource and its utilization by a task, and that the instances are terminated immediately when they are no longer needed.

A direct competitor to Cloud Gunther is Cloud Scheduler [13]. From the website, it seems to be a plug-in for Condor which can manage VM provisioning for it. Similarly to Cloud Gunther, it is fairly new and only features FCFS queuing.

An older project of this sort is Nephele [14], which focuses on real-time transfers of data streams between jobs that form a workflow. It provisions different-sized instances for each phase of the workflow. In this system, the number and type of machines in a job are defined upfront and all instances involved in a step must run at once, so there is little space for optimization in the area of resource availability and utilization.

Aside from cluster-oriented tools, desktop grid systems are also reaching into the area of clouds. For example, the Aneka platform [12] can combine resources from statically allocated servers, unused desktop computers and Amazon Spot instances. It can provision the cloud instances when they are needed to satisfy job deadlines. This system certainly seems more mature than Cloud Gunther and has reached commercial availability.

None of these systems deals with the issue of resource availability in private clouds and fully enjoy the benefits of the illusion of infinite supply. To the best of our knowledge, no one has yet dealt with the problem of maximizing utilization of a cloud environment that is not fully dedicated to HPC and where batch jobs would have the status of “filler traffic”.

III. COMPLETED WORK

A. Eucalyptus

Eucalyptus [4] is the cloud platform that is used for experiments at the Dept. of Cybernetics. It is an open-source implementation of the Amazon EC2 industry standard API (Application Programming Interface) [19]. It started as a research project at the University of California and evolved to a commercial product.

It is a distributed system consisting of five components. Those are the Node Controller (NC), which is responsible of running virtual machines from images obtained from the Walrus (Amazon S3 (Simple Storage Service) implementation). Networking for several NCs is managed by a Cluster Controller (CC), and the Cloud

 ISBN: 978-1-61208-216-5

Controller (CLC) exports all external APIs and manages the cloud's operations. The last component is the Storage Controller (SC), which exports network volumes, emulating the Amazon EBS (Elastic Block Store) service. The architecture can be seen in Figure 2.

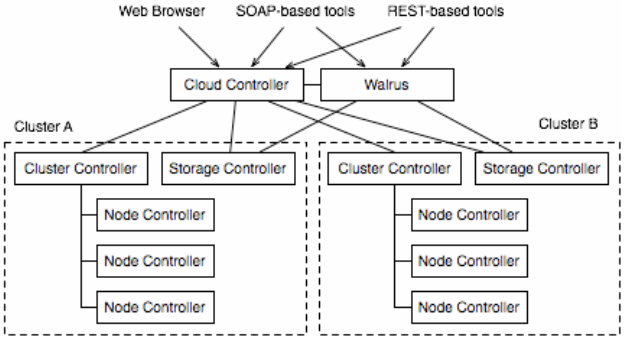

Figure 2. Eucalyptus architecture [4]

Our Eucalyptus setup consists of a server that hosts the CLC, SC and Walrus components and is dedicated to cloud experiments. The server manages 20 8-core Xeon workstations, which are installed in two labs and 1/4 of their capacity can be used for running VM instances through Eucalyptus NCs. A second server, which is primarily used to provide login and file services to students and is physically closer to the labs, is used to host Eucalyptus CC.

The cloud is used for several research projects at the Cloud Computing Center research group [5]. Those are:

- Automatic deployment to PaaS (Platform as a Service), a web application capable of automatic deployment of popular CMS (Content Management Systems) to PaaS.
- Effective scaling in private IaaS (Infrastructure as a Service), a diploma thesis on adding automatic scaling and load balancing support for web applications in private clouds.
- Cloud Gunther, a web application that manages a queue of batch computational jobs and runs them on Amazon EC2 compatible clouds.

Aside from this installation of Eucalyptus, we also have experience deploying the system in a corporate environment. An evaluation has been carried out in cooperation with the Czech company Centrum. The project validated the possibility of deploying one of their production applications as a machine image and scaling the number of instances of this image depending on current demand. A hardware load-balancer appliance from A10 Networks was used in the experiment and the number of instances was controlled manually as private infrastructure clouds generally lack the autoscaling capabilities of public clouds.

B. Cloud Gunther

While the Effective scaling in private IaaS project will also be instrumental for further research, it is only just starting. In contrast, the Master's thesis on Cloud Gunther has already been defended; the possibilities for its further development are the main topic of this article.

The application is written in the Ruby on Rails framework and offers both interactive and REST (Representational State Transfer) access. It depends on Apache with mod_passenger, MySQL and RabbitMQ for operation.

It can control multiple Amazon EC2 [19] compatible clouds. The queuing logic resides outside the MVC (Model, View, Controller) scheme of Rails, but shares database access with it. The communication scheme is on Fig. 3.

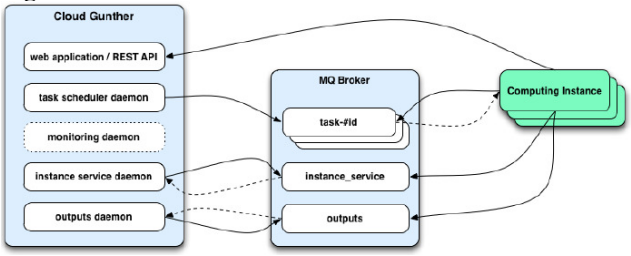

Figure 3. Communication scheme in Cloud Gunther [6]

The Scheduler daemon contains the Priority FCFS queuing discipline and is responsible for launching instances and submitting their job details to the message broker. The Agent on the instance then retrieves these messages and launches the specified user algorithm with the right parameters. It is capable of running multiple jobs from the same user, thus saving the overhead of instance setup and teardown.

The two other daemons are responsible for collecting messages from the queue, which are sent by the instances. The Instance Service serves to terminate instances, which have run out of jobs to execute; the Outputs daemon collects standard and error outputs of user programs captured by the launching Agent. A Monitoring daemon is yet to be implemented.

The web application itself fulfills the requirement of multitenancy by providing standard user login capabilities. The users can also be categorized into groups, which have different priorities in the scheduler.

The cloud engine credentials are shared for each cloud (for simpler cloud access via API and instance management via SSH (Secure Shell)).

Each cloud engine has associated images for different tasks, eg. image for Ruby algorithms, image for Java, etc. The images are available to all users, however when launched, each user will get his own instance.

The users can define their algorithm's requirements, i.e., which image the algorithm runs on and what instance size it needs. There is also support for management of different versions of the same algorithm. They may only differ in command line parameters, or each of them may have a binary program attached to it, which will be uploaded to the instance before execution.

Individual computing tasks are then defined on top of the algorithms. The task consists of input for the algorithm, which is interpolated into its command line

 ISBN: 978-1-61208-216-5

with the use of macros, as well as the instance index and total count of instances requested. These values are used by pseudoparallel algorithms to identify the portion of input data to operate on, and by parallel algorithms for directing communication in message passing systems.

As one can see in Figure 4., the system is ready for private clouds. It can extract the amount of free resources from Eucalyptus and the scheduler takes it into account when launching new instances.

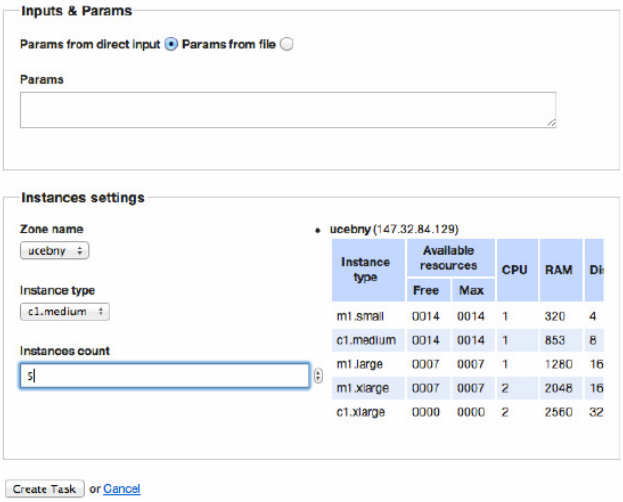

Figure 4. Cloud Gunther – part of the New Task screen [6]

The Cloud Gunther has been tested on several real workloads from other scientists. Those were production planning optimization, recognition of patterns in images and a multiagent simulation. They represented a parameter sweep workflow, a pseudoparallel task and a parallel task, respectively.

VM images for running the tasks were prepared in cooperation with the users. Usability was verified by having the users set up algorithm descriptions in the web interface. The program then successfully provisioned the desired number of VM instances, executed the algorithms on them, collected the results and terminated the instances.

The main drawback, from our point of view, is that when there are jobs in the queue, the program consumes all resources on the cluster.

This is not a problem in the experimental setting, but in a production environment, which would be primarily used for interactive traffic, and would attempt to exploit the agility of cloud infrastructure to run batch jobs as well, this would be unacceptable.

In such a setting, the interactive traffic needs to have absolute priority. For example, if there was a need to increase the number of web servers due to a spike in demand, then in the current state, the capacity would be blocked by Cloud Gunther until some of its tasks finished. It would be possible to terminate them, but that would cause loss of hours of work. A proactive solution to the heterogenous load situation is needed.

IV. FUTURE WORK

Future work planned on the Cloud Gunther can be split into two categories. First and more important is the consideration of interactive load also present on the cluster, see Subsection A. Second is integration of better queuing disciplines to bring it up to par with existing cluster management tools. Two ideas for that are presented in Subsections B and C.

A. Estimation of the amount of interactive load in time

The interactive traffic needs to have priority over the batch jobs. Therefore, once work is completed on the general purpose autoscaler for private IaaS, it will be possible to record the histogram of the number of instances that the autoscaler is managing. From this histogram, data on daily, weekly and monthly usage patterns of the web servers may be extracted and used to set the amount of free resources for Cloud Gunther.

The vision on the extraction method is that it will employ machine learning techniques to approximate the statistical distribution of the number of web server instances at any hour of the year, probably breaking it up to yearly, monthly, weekly and daily curves.

Instead of seeing only the current amount of free resources in the cloud, the batch job scheduler could be able to ask: "May I allocate 10 large instances to a parallel job for the next 4 hours with 90% probability of it not being killed?"

A similar problem exists in desktop grids. Article [15] illustrates the collection of availability data from a cluster of desktop machines and presents a simulation of predictive scheduling using this data. The abstraction of the cloud will shield away the availability of particular machines or their groups, the only measured quantity will be the amount of available VM slots of a certain size.

B. Out-of-order scheduling

This of course assumes a scheduler that will be capable of using this information. Our vision is a queue discipline that internally constructs a workflow out of disparate tasks. The tasks, each with an associated estimate of duration, will be reordered so that the utilization of the cloud is maximized.

For example, when there is a job currently running on 20 out of 40 slots and should finish in 2 hours, and there is a 40 slot job in the queue, it should try to run several smaller 2 hour jobs to fill the free space, but not longer, since that would delay the large job.

These requirements almost exactly match the definition of the Multiprocessor scheduling problem (see [8]). Since this is a NP-hard class problem, solving it for the whole queue would be costly. The most feasible solution seems to come from the world of out-of-order microprocessor architectures, which re-order instructions to fully utilize all execution units, but only do so with the first several instructions of the program. The batch job scheduler will be likewise able to calculate the exact solution with the first several jobs in the queue, which will otherwise remain Priority FCFS.

 ISBN: 978-1-61208-216-5

C. Dynamic priorities

The estimation of job duration is a problem all for itself. At first, the estimate could be done by the user. Later, a system of dynamic priorities could be built on top of that.

The priorities would act at the level of users, penalizing them for wrong estimates, or better, suspending allocation of resources to users whose tasks have been running for longer time than the scheduler thought.

Inspiration for this idea is taken from the description of the Multilevel Feedback Queue scheduler used historically in Linux [7]. However, the scheduler will set priorities for users, not processes, and allocate VMs to tasks, not jiffies to threads. It also will not have to be real-time and preemptive, making the design simpler.

The scheduler's estimate of process run time could be based on the user estimates, but also on the previous run time of processes from the same task or generally those submitted by the same user for the same environment. That would lead to another machine learning problem.

V. CONCLUSION

The cloud presents a platform that can join two worlds that were previously separate – web servers and HPC grids. The public cloud, which offers the illusion of infinite supply of computing resources, will accommodate all the average user's needs, however, new resource allocation problems arise in the resource-constrained space of private clouds.

We have experience using private cloud computing clusters both for running web services and batch scientific computations. The challenge now is to join these two into a unified platform.

Currently, Cloud Gunther, although not ready for commercial deployment, already has some state of the art features, like the automatic management of cloud computing instances and a REST-compliant web interface. It also differs from other similar tools by its orientation towards private cloud computing clusters.

In the future, it could become a unique system for managing batch computations in a cloud environment primarily used for web serving, thus allowing to exploit the dynamic nature of private cloud infrastructure and to raise its overall utilization.

ACKNOWLEDGMENTS

Credit for the implementation of Cloud Gunther, mainly the user friendly and cleanly written web application goes to Josef Šín.

We thank the company Centrum for providing material for our experiment and insights on private clouds from the business perspective.

REFERENCES

[1] D. M. Smith, "Hype Cycle for Cloud Computing," Gartner, 27 July 2011, G00214915.

[2] T. Vondra and J. Šedivý, "Od hostingu ke cloudu," Research Report GL 229/11, CTU, Faculty of Electrical Engineering, Gerstner Laboratory, Prague, 2011, ISSN 1213-3000.

[3] T. P. Morgan, "Univa skyhooks grids to clouds: Cloud control freak meets Grid Engine," The Register, 3rd June 2011, <http://www.theregister.co.uk/2011/06/03/univa_grid_engine_cloud/> 19 March 2012.

[4] "Installing Eucalyptus 2.0," Eucalyptus, <http://open.eucalyptus.com/wiki/EucalyptusInstallation_v2.0> 19 March 2012.

[5] J. Šedivý, "3C: Cloud Computing Center," CTU, Faculty of Electrical Engineering, dept. of Cybernetics, Prague, <https://sites.google.com/a/3c.felk.cvut.cz/cloud-computing-center-preview/> 19 March 2012.

[6] J. Šín, "Production Control Optimization in SaaS," Master's Thesis, CTU, Faculty of Electrical Engineering and University in Stavanger, Department of Electrical and Computer Engineering, Supervisors J. Šedivý and C. Rong, Prague, 20 December 2011.

[7] T. Groves, J. Knockel, E. Schulte, "BFS vs. CFS - Scheduler Comparison," 11 December 2011 < http://slimjim.cs.unm.edu/~eschulte/data/bfs-v-cfs_groves-knockel-schulte.pdf > 11 May 2012.

[8] "Multiprocessor scheduling," in Wikipedia: the free encyclopedia, San Francisco (CA): Wikimedia Foundation, 12 March 2012, <http://en.wikipedia.org/wiki/Multiprocessor_scheduling> 19 March 2012.

[9] "StarCluster," Massachusetts Institute of Technology, < http://web.mit.edu/star/cluster/index.html> 11 May 2012.

[10] H. Eriksson, et al., "A Cloud-Based Simulation Architecture for Pandemic Influenza Simulation," AMIA Annu Symp Proc. 2011; 2011: 364–373, pp. 364–373.

[11] D. de Oliveira, E. Ogasawara, K. Ocaña, F. Baião and M. Mattoso, "An adaptive parallel execution strategy for cloud-based scientific workflows," Concurrency Computat.: Pract. Exper. (2011), doi: 10.1002/cpe.1880.

[12] R. N. Calheiros, C. Vecchiola, D. Karunamoorthya and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds," Future Generation Computer Systems 28 (2012), pp. 861-870, doi: 10.1016/j.future.2011.07.005.

[13] "Cloud Scheduler," University of Victoria, <http://cloudscheduler.org/> 11 May 2012.

[14] D. Warneke and O. Kao, "Nephele: efficient parallel data processing in the cloud," MTAGS '09: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, November 2009, doi: 10.1145/1646468.1646476.

[15] K. Ramachandran, H. Lutfiyya and M. Perry, "Decentralized approach to resource availability prediction using group availability in a P2P desktop grid," Future Generation Computer Systems 28 (2012), pp. 854–860, doi: 10.1109/CCGRID.2010.54.

[16] R. Grossman and Y. Gu, "Data mining using high performance data clouds: experimental studies using sector and sphere," In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, 2008, pp. 920-927, doi: 10.1145/1401890.1402000.

[17] M. J. Litzkow, M. Livny and M. W. Mutka, "Condor-a hunter of idle workstations," 8th International Conference on Distributed Computing Systems (1988), pp. 104-111.

[18] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," Proceedings of the first IEEE/ACM International Symposium on Cluster Computing and the Grid (2001), pp. 35-36.

[19] "Amazon Elastic Compute Cloud (EC2) Documentation," Amazon, <http://aws.amazon.com/documentation/ec2/> 27 May 2012

 ISBN: 978-1-61208-216-5