RESEARCH REVIEW JOURNALS
Volume-03
Issue-01
January-2018

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com [UGC Listed Journal]

Design and Development of Metrics and Graphical Model in Clouds

[1]Syed Azahad & [2]Dr. R. P. Singh

[1]*Research Scholar, Sri Satya Sai University ,Bhopal (India)*
[2]*Vice Chancellor. SSSUTMS, Bhopal (India)*

ARTICLE DETAILS

Article History

Published Online: 28 January 2018

Keywords
probabilistic graphical model; Cloud Metric

ABSTRACT

Cloud manufacturing (CM) is an emerging manufacturing model based on collaboration among manufacturing enterprises in a cloud computing environment. Naturally, collaboration is one of main factors that impacts performance in a variety of ways such as quality, lead time, and cost. Therefore, collaboration levels should be considered when solving operational issues in CM. However, there has been no attempt to estimate these levels between enterprises participating in CM. The collaboration level among enterprises in CM is defined as the ability to produce a manufacturing service that satisfies a customer by means of collaborative production amongst enterprises. We measure it as the conditional probability that collaborative performances are high given collaborative performance factors (e.g., resource sharing, information sharing, etc.). In this paper, we propose a framework for estimating collaboration levels. We adopt a probabilistic graphical model (PGM) to develop the framework, since the framework includes a lot of random variables and complex dependencies among them.

INTRODUCTION

Cloud and Grid computing build the pillars of today's modern scientific compute environments. Batch computing has traditionally supported high performance computing centers to better utilize their compute resources with the goal to satisfy the many concurrent users with sophisticated batch policies utilizing a number of well managed compute resources. Grid Computing and its predecessor meta-computing elevated this goal by not only introducing the utilization of multiple queues accessible to the users, but by establishing virtual organizations that share resources among the organizational users. This includes storage and compute resources and exposes the functionality that users need as services. Recently, it has been identified that these models are too restrictive, as many researchers and groups tend to develop and deploy their own software stacks on computational resources to build the specific environment required for their experiments. Cloud computing provides here a good solution as it introduces a level of abstraction that lets the advanced scientific community assemble their own images with their own software stacks and deploy them on large numbers of computational resources in clouds. Since a number of Infrastructure as a Service (IaaS) exist, our experience [1] tells us the importance of offering a variety of them to satisfy the various user community demands. In addition, it is important to support researchers that develop such frameworks further and may need more access to the compute and storage hardware resources than is provided by the current IaaS frameworks. For this reason, it is also important to provide users with the capabilities of staging their own software stack. This feature has also been introduced by other test-beds. This includes Open Cirrus [2], Emu Lab [3], Grid5000 [4] and Future Grid [5]. Within Future Grid we developed a sophisticated set of services that simplify the instantiation of images that can be deployed on virtualized and non-virtualized resources contrasting our efforts. The work described here significantly enhances the services developed and described in our publications about Future Grid focusing on dynamic provisioning supported by image management, generation, and deployment [1] [6].

we enhance our services in the following aspects

a) Implementation of a uniform cloud metric framework for Eucalyptus 3 and Open Stack Essex.
b) Design of a flexible framework that allows resource reallocation between various IaaS frameworks, as well as bare-metal.
c) Design of a meta-scheduler that re-allocates resources based on metric data gathered from the usage of different frameworks.
d) Targeted prototype development and deployment for Future Grid.

DESIGN

features of the resource and service fabric that are an integral part of our design. We assume that the Resource Fabric consists of a resource pool that contains a number of compute services. Such services are provided either as a cluster or as part of a distributed network of workstations (NOW). The resources are grouped based on network connectivity proximity. This will allow the creation of regions within cloud IaaS environments to perform more efficiently among its servers. We assume a rich variety of services offered in the Service Fabric. This includes multiple IaaS, PaaS frameworks, and HPC environments. Instead of assuming that there can only be one cloud for a particular IaaS framework, we envision multiple independent clouds. This assumption potentially allows users to host their own privately managed clouds and also integrate them with public clouds. We have already deployed such an infrastructure as part of the

FutureGrid, allowing users to access a variety of preconfigured clouds to conduct interoperability experiments among the same IaaS and also different IaaS frameworks, as well as the inclusion of dedicated HPC services. Having access to such a comprehensive environment opens up a number of interesting design challenges. We observe that our operational mode is significantly enhanced in contrast to other academic clouds that typically only install a single IaaS framework on their resource. Thus, such environments cannot offer by themselves the comprehensive infrastructure needed to conduct many of the topics that arise in cloud federation. One of the questions we need to answer is how we can best utilize such an environment that supports inter-cloud and bare metal demands posed by the users as we have practically observed in Future Grid and how we can integrate these requirements into a software architecture. We have designed a software architecture to address the requirements presented earlier. We distinguish the user layer allowing administrators, but also users (and groups of users) to interact with the framework. In addition, we point out that Web services can interact with it to develop third party automated tools and services leveraging the capabilities. Access to the various functions is provided in a secure fashion. Due to the diverse user communities wishing to use the environment, our design supports a variety of access interfaces including command line, dashboard, web services, as well as libraries and APIs.[7] An important aspect is to be able to integrate existing and future information services to provide the data to guide dynamic and automatic resource provisioning, cloud–bursting, cloud-seeding, and cloud-shifting. Due to this reason, we allow in our design the integration of events posted by services such as Inca, Ganglia, and Nagios. Moreover, we obtain information from the running clouds and, when the provided information is not sufficient, we will be able to ingest our own information by analyzing log files or other information obtained when running a cloud. For clouds, we also host an instance archive that allows us to capture traces of data that can be associated with a particular virtual machine instance. A metric archive allows the registration of a self-contained service that analyses the data gathered while providing a data series according to the metric specified. Metrics can be combined and can result in new data series. At the center of this design is a comprehensive RAIN service Layer. Rain is an acronym for Runtime Adaptable INsertion service signifying services that on the one hand adapt to runtime conditions and on the other allow inserting or dynamically provisioning software environments and stacks. We use the terms rain and raining to refer to the process of instantiating services on the resource and service fabrics. In this analogy, we can rain onto a cluster services that correspond to an IaaS, a PaaS, or a HPC batch system. Rain can be applied to virtualized and non-virtualized machine images and software stacks. In addition, Rain can also be used to move resources between already instantiated environments, hence supporting cloudshifting. The most elementary operation to enable cloud-seeding[8]

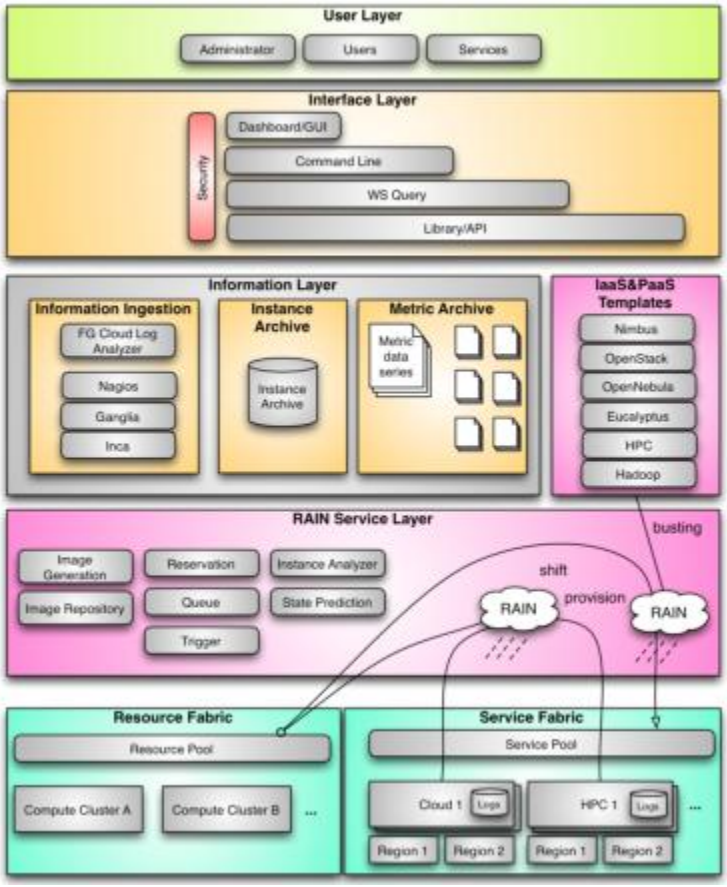

Figure: Design of the rain-based federated cloud management services and cloud-shifting is to provision the software and services onto the resources. We have devised this elementary operation and introduced in [6] and can now build upon it. In our past effort, we took on the problem of image management. In this work we focus on cloud-shifting, which is a logical extension in order to satisfy our users' needs[9]

CLOUD METRIC

cloud computing in the mainstream, there is a preponderance of cloud based services in the market and the choices for consumers increase daily. ... To describe a "measured service", one needs to identify the cloud service properties that have to be measured and what their standards of measurement or metrics are

Top 7 Cloud Metrics for Value-Based Practices

1. **Monthly Recurring Revenue (MRR):** Accountants sum the monthly fee paid by every customer to calculate the MRR. This is unquestionably one of the most crucial metrics to be tracked over time because once a new customer is acquired, there is no ongoing cost to acquire new customers or sales expenditure. Generally, MRR is a very predictable source of ongoing subscription revenue and a major source of revenue for major accounting businesses.
2. **Cost to Acquire (CAC):** The CAC is the price paid to acquire a new customer. It can be worked out by dividing the total costs associated with acquisition by total new customers within a specific time period. This metric is generally used in conjunction with the lifetime value of the customers to calculate how much money is being made from each customer, relative to the cost of acquisition. Accountants, therefore, usually also work out a cost to acquire versus lifetime value ratio. Generally, this can be displayed by the following:

   A ratio of less than 1:1 means that the business is actually losing money for each customer it acquires.
   A ratio of 1:1 means that the business is only breaking even with each customer acquisition.
   A ratio of 3:1 means the business is thriving and has a solid business underlying business model.
   Greater than 4:1 the business is performing well, but under investing. More clients could be acquired.[10]

3. **Lifetime Value (LTV):** The lifetime value is the total financial benefit from the future relationship with a customer. It can be summarized as the dollar value of each customer. This is an extremely important metric because it allows business to determine the maximum they they should spend to attract new customers. Customer lifetime value also emphasizes the importance of having a long, healthy and stable consumer relationship, rather than trying to focus on short-term earnings. One of the most important components of identifying customer lifetime value is identifying that not all customers are created equally. Segmenting the lifetime value of specific groups of clients can allow the company to predict its most profitable group of customers and focus on them more, rather than on less profitable customers. Accountants increasingly find that focusing on high-value customers can enable them to bolster their business relationships and reduce customer churn.
4. **Non-Recurring Revenue:** This comprises of earnings that are once-off events, which are unusual in nature for a company. Extraordinary earnings are said to be non-recurring when they will not generally take place again in the future. These are separately disclosed as infrequent items, which are unrelated to the typical operating activities of a firm, or of a contract nature (a one-off deal, for example).[11]
5. **Churn:** With regard to accounting services, this refers to the number of subscribers who discontinue/downgrade their services within a specific period. Churn is inherently negative, as it reduces a firm's revenue stream and total earning power. For a company to sustain its earnings, the number of new clients that it acquires must exceed the churn rate.
6. **Average Revenue Per Client:** The average revenue per client is a measure of the average revenue generated per client. This helps businesses to segment their client base and work out which clients are generating the most/least revenue and the respective sources of revenue. This helps accounting firms measure the effectiveness of their service offerings.
7. **Net Promoter Score:** The net promoter score measures customer business experience and serves to predict business growth. The net promoter score currently commonly serves as a metric for customer experience management around the world.[12]

CONCLUSION

cloud environment that is not focused singly on supporting just an IaaS framework. Our understanding of federation includes various IaaS frameworks on potentially heterogeneous compute resources. In addition, we are expanding our federated cloud environment to include and integrate traditional HPC services. This work is a significant enhancement to our earlier work on dynamic image generation and provisioning in clouds and baremetal environments by addressing challenges arising in cloud seeding and cloud shifting.

REFERENCES

[1] von Laszewski, G., Diaz, J., Wang, F. and Fox, G. C. Comparison of Multiple Cloud Frameworks. In Proceedings of the IEEE CLOUD 2012, 5th International Conference on Cloud Computing (Honolulu, HI, USA, 24-29 June, 2012). Doi 10.1109/CLOUD.2012.104.

[2] Avetisyan, A. I., Campbell, R., Gupta, I., Heath, M. T., Ko, S. Y., Ganger, G. R., Kozuch, M. A., O'Hallaron, D.,

Kunze, M., Kwan, T. T., Lai, K., Lyons, M., Milojicic, D. S., Hing Yan, L., Yeng Chai, S., Ng Kwang, M., Luke, J. Y. and Han, N. Open Cirrus: A Global Cloud Computing Testbed. Computer, 43, 4 2010), 35-43. Doi 10.1109/mc.2010.111.

[3] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C. and Joglekar, A. An Integrated Experimental Environment for Distributed Systems and Networks. In Proceedings of the Proceedings of the 5th Symposium on Operating Systems Design & Implementation (December 2002, 2002).

[4] Grid5000 Home Page, http://www.grid5000.fr/.

[5] G. von Laszewski, G. C. Fox, Fugang Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Vöckler, R. J. Figueiredo, J. Fortes and K. Keahey. Design of the FutureGrid experiment management framework. In Proceedings of the Gateway Computing Environments Workshop (GCE) at SC10 (New Orleans, LA, 14-14 Nov. 2010, 2010). IEEE. Doi 10.1109/GCE.2010.5676126

[6] Diaz, J., von Laszewski, G., Wang, F. and Fox, G. Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures. In Proceedings of the IEEE CLOUD 2012, 5th International Conference on Cloud Computing (Honolulu, HI, 2012). IEEE. Doi 10.1109/cloud.2012.94.

[7] Yoo, A., Jette, M. and Grondona, M. SLURM: Simple Linux Utility for Resource Management. in Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol 2862, p. 44-60, Springer Berlin / Heidelberg, 2003.

[8] Univa Grid Engine, http://www.univa.com/products/grid-engine/.

[9] Genias CODINE: Computing in distributed networked environments (1995), http://www.genias.de/genias/english/ codine.html.

[10] Altair PBS, http://www.pbsworks.com/.

[11] Moab, http://www.adaptivecomputing.com/products/hpcproducts/.

[12] Bright-Computing Cluster Manager, http://www.brightcomputing.com/Bright-Cluster-Manager.php

