MANY TASK AND HIGH THROUGHPUT COMPUTING USING INTELLIGENT CLOUDS

Ch.Swathi*[1]*, *P.Veeresh*[2]*, *V. Chandra Sekhar*[3]

1. M.Tech (CSE) Student, Dept of CSE, ST.Johns College of Engg, Yemmiganur, Kurnool

2. Assoc.Prof, Dept of CSE, ST.Johns College of Engg, Yemmiganur, Kurnool

3. Professor, Dept of CSE, ST.Johns College of Engg, Yemmiganur, Kurnool

Keywords:
IaaS, Ad-hoc Parallel data processing, Virtual machines, Job Scheduling

Abstract: *Ad-hoc parallel data processing has proven to be a critical paradigm for companies processing large volumes of unstructured data.ad-hoc parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) clouds. Customers to access the services and to deploy their programs, major Cloud computing companies have on track to integrate frameworks for parallel data processing in their product group. In this paper we discuss the algorithms and architecture for efficient Ad-hoc parallel data processing in clouds. It is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's Infrastructure-as-a-Service clouds for both, task scheduling and execution*.

1. INTRODUCTION

Ad-hoc data processing is a powerful abstraction for mining terabytes of data. Systems for massive parallel data processing, such as Map Reduce [5] and Dryad [8], allow Internet companies, e.g., Google, Yahoo, and Microsoft, to mine large web crawls, click streams, and system logs across shared-nothing clusters of unreliable servers. While traditionally the job of parallel databases [6], these systems give programmers a familiar procedural interface to process unstructured data, and, since the data is often relatively transient, avoid the overhead of importing the data into a traditional RDBMS. Today, such systems manage parallel processing tasks across tens of thousands of machines in a single data center.

Cloud computing has the potential to dramatically change the landscape of the current IT industry (Armbrust et al., 2009; Goldberg, 1989) 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.

*** Ch.Swathi***
M.Tech (CSE) Student, Dept of CSE, ST.Johns College of Engg, Yemmiganur, Kurnool

Operators of so-called IaaS clouds, like Amazon EC2, (AWSLLC, 2011a), 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) and cost. Since the VM abstraction of IaaS clouds fits the architectural paradigm assumed by the data processing frameworks described above, projects like Hadoop The Apache Software Foundation, 2011 (White, 2010), a popular open source implementation of Google's MapReduce framework, already have begun to promote using their frameworks in the cloud (White, 2010) Only recently, Amazon has integrated Hadoop as one of its core infrastructure services (AWSLLC, 2011b). However, instead of embracing its dynamic resource allocation, current data processing frameworks rather expect the cloud to imitate the static nature of the cluster environments (Dornemann et al., 2009) they were originally designed for, e.g., at the moment the types and number of VMs allocated at the beginning of a compute job cannot be changed in the course of processing, although the tasks the job consists of completely different demands on the environment.

As a result, rented resources may be inadequate for big parts of the processing job, which may lower the overall processing performance and increase the cost. One of an IaaS cloud's key feature is the provisioning of compute resources on demand. The computer resources available in the cloud are highly dynamic and possibly heterogeneous. Nephele is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for both task scheduling and execution. Particular tasks of a processing a job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution. Nephele is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for both, task scheduling and execution. Particular tasks of a processing job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution.

EXISTING SYSTEM

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. 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.

PROPOSED SYSTEM

In recent years a variety of systems to facilitate MTC has been developed. Although these systems typically share common goals (e.g. to hide issues of parallelism or fault tolerance), they aim at different fields of application. MapReduce is designed to run data analysis jobs on a large amount of data, which is expected to be stored across a large set of share-nothing commodity servers.

Once a user has fit his program into the required map and reduce pattern, the execution framework takes care of splitting the job into subtasks, distributing and executing them. A single Map Reduce job always consists of a distinct map and reduce program.

3.SYSTEM ARCHITECTURE

The system architecture of the proposed system is given in fig 1. A brief description of the components within the given system follows.

3.1Components of cloud

The different components of Eucalyptus Cloud are

- Node Controller (NC)
- Cluster Controller (CC)
- Walrus Storage Controller (WS3)
- Storage Controller (SC)
- Cloud Controller (CLC)

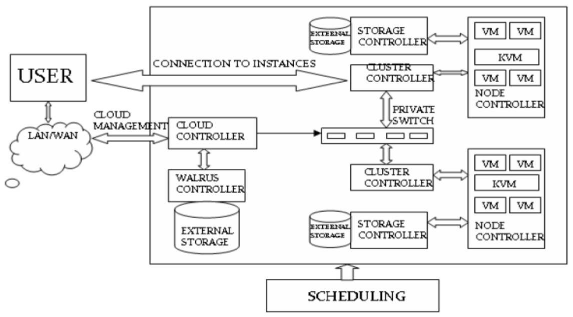

Fig 1: System Architecture

Node Controller (NC). Node Controller runs on each node and controls the life cycle of instances running on the node. The NC interacts with the OS and the hypervisor running on the node on one side and the Cluster Controller (CC) on the other side. NC queries the Operating System running on the node to discover the node's physical resources - the number of cores, the size of memory, the available disk space and also to learn about the state of VM instances running on the node and propagates this data up to the CC. Certain functions like Collection of data related to the resource availability and utilization on the node and reporting the data to CC, Instance life cycle management can be performed.

Cluster Controller (CC). CC manages one or more Node Controllers and deploys/manages instances on them. CC also manages the networking for the instances running on the Nodes under certain types of networking modes of Eucalyptus. CC communicates with Cloud Controller (CLC) on one side and NCs on the other side. Certain functions like receiving requests from CLC to deploy instances, deciding which NCs to use for deploying the instances, controlling the virtual network available to the instances, collecting information about the NCs registered with it and report it to the CLC

Walrus Storage Controller (WS3). WS3 provides a persistent simple storage service using REST and SOAP APIs compatible with S3 APIs. Certain functions like Storing the machine images, Storing snapshots, Storing and serving files using S3 API can be performed in this controller. WS3 should be considered as a simple file storage system.

Storage Controller (SC). SC provides persistent block storage for use by the instances. This is similar to the Elastic Block Storage (EBS) service from AWS. Certain functions like Creation of persistent EBS devices, providing the block storage over AoE or iSCSI protocol to the instances, allowing creation of snapshots of volumes

Cloud Controller (CC). The Cloud Controller (CLC) is the front end to the entire cloud infrastructure. CLC provides an EC2/S3 compliant web services interface to the client tools on one side and interacts with the rest of the components of the Eucalyptus infrastructure on the other side. CLC also provides a web interface to users for managing certain aspects of the UEC infrastructure. Certain functions like monitoring the availability of resources on various components of the cloud infrastructure, including hypervisor nodes that are used to actually provision the instances and the cluster controllers that manage the hypervisor nodes, Resource arbitration - Deciding which clusters will be used for provisioning the instances and monitoring the running instances.

Cloud Setup

The functional architecture of the private cloud setup is given in fig 3.1.Two servers (server 1 and server 2) will run a 64-bit

server version and third server will run a Desktop 64-bit version (client 1). Then install the Desktop version on client 1 so that

Firefox or other browsers can be used to access the web interface.

The following modifications are to be made for the cloud setup.

The gateway for Server2 is set to the IP of the CC IP 192.168.20.1.This will enable the Server2 to connect to the enterprise

network through Server1 (CC).

Server1 is a 64-bit server and Server2 is a 64-bit VT-enabled server.

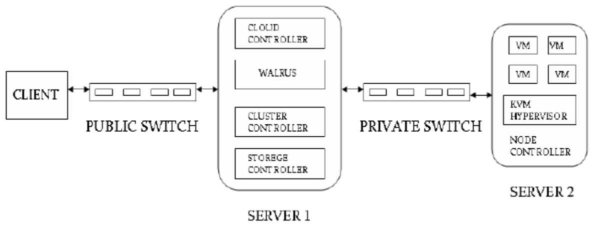

Fig.2. Functional Architecture for Cloud Setup

1.1. ALGORITHMS:

1.Job Scheduling and Execution:

After having received a valid Job Graph from the user, Nephele's Job Manager transforms it into a so-called Execution Graph. An Execution Graph is Nephele's primary data structure for scheduling and monitoring the execution of a Nephele job. Unlike the abstract Job Graph, the Execution Graph contains all the concrete information required to schedule and execute the received job on the cloud.

2.Parallelization and Scheduling Strategies:

If constructing an Execution Graph from a user's submitted Job Graph may leave different degrees of freedom to Nephele. The user provides any job annotation which contains more specific instructions we currently pursue simple default strategy: Each vertex of the Job Graph is transformed into one Execution Vertex. The default channel types are network channels. Each Execution Vertex is by default assigned to its own Execution Instance unless the user's annotations or other scheduling restrictions (e.g. the usage of in-memory channels) prohibit it.

CONCLUSION

In this paper, a clear view of how the cloud can be setup and how an instance i.e. a virtual machine can be created and thus new Operating System can be boot from the virtual machine. It basically used to implement infrastructure as a service (IaaS). Thus it helps for the organization to create their own cloud structure which eliminates renting from the public cloud providers. It also offers flexible infrastructure services that can be easily utilized and managed by end users according to their needs. It enables enterprises and government agencies to establish their own cloud computing environments. In the future, a proper CPU Scheduling and security management algorithms can also be incorporated in the cloud setup.

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. H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters. In SIGMOD ’07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029–1040, New York, NY, USA, 2007. ACM.

7. M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements. SIGMETRICS Perform. Eval. Rev., 30(1):11–20, 2002.

8. R. Davoli. VDE: Virtual Distributed Ethernet. Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on, 0:213–220, 2005.

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.,

11. (3):219–237, 2005.