International Journal of Emerging Engineering Research and Technology
Volume. 2, Issue 4, July 2014, PP 139-145
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Dynamic Resource Allocation for Efficient Parallel Data Processing Using RMI Protocol

M.Kishore Kumar

M.Tech Scholor
CSE DEPT, SIT, PUTTUR, INDIA
kishorekumarm000@gmail.com

K.Narayana

Assistant professor
Head of the Departement of Cse, SIT, PUTTUR, INDIA
karnatham.narayana@gmail.com

Abstract: *In recent years ad-hoc parallel data processing has emerged to be one of the killer applications for Infrastructure-as-a-Service (IaaS) clouds. Major Cloud computing companies have started to integrate frameworks for parallel data processing in their product portfolio, making it easy for customers to access these services and to deploy their programs. the processing frameworks which are currently used have been designed for static, homogeneous cluster setups and disregard the particular nature of a cloud. Consequently, 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 discuss the opportunities and challenges for efficient parallel data processing in clouds and present our research project Nephele. 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. In this paper we discuss the opportunities and challenges for efficient parallel data processing 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. Based on this new framework, we perform extended evaluations of Map Reduce-inspired processing jobs on an IaaS cloud system and compare the results to the popular data processing framework Hadoop.*

Keywords: ***Many****-Task Computing, High-Throughput Computing, Loosely Coupled Applications, Cloud Computing.*

1. INTRODUCTION

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 Map Reduce, 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 [7]. Although these systems differ in design, their programming models share similar objectives, namely hiding the hassle of parallel programming, fault tolerance, and execution optimizations from the developer. Developers can typically continue to write sequential programs. The processing framework then takes care of distributing the program among the available nodes and executes each instance of the program on the appropriate fragment of data.

In this paper we want to discuss the particular challenges and opportunities for efficient parallel data processing in clouds and present Nephele, a new processing framework explicitly designed for cloud environments. Most notably, Nephele 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. This paper is an extended version of [9]. It includes further details on scheduling strategies and extended experimental results.

2. CHALLENGES AND OPPORTUNITIES

Current data processing frameworks like Google's Map Reduce or Microsoft's Dryad engine have been designed 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.

2.1 Opportunities

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 compute resources available in a cloud are highly dynamic and possibly heterogeneous.

With respect to parallel data processing, this flexibility leads to a variety of new possibilities, particularly for scheduling data processing jobs. The question a scheduler has to answer is no longer "Given a set of compute resources, how to distribute the particular tasks of a job among them?", but rather "Given a job, what compute resources match the tasks the job consists of best?". This new paradigm allows allocating compute resources dynamically and just for the time they are required in the processing workflow. E.g., a framework exploiting the possibilities of a cloud could start with a single VM which analyzes an incoming job and then advises the cloud to directly start the required VMs according to the job's processing phases. After each phase, the machines could be released and no longer contribute to the overall cost for the processing job.First, the scheduler of such a framework 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.

Finally, the scheduler of such a processing framework must be able to determine which task of a job should be executed on which type of VM and, possibly, how many of those. This information could be either provided externally, e.g. as an annotation to the job description, or deduced internally, e.g. from collected statistics, similarly to the way database systems try to optimize their execution schedule over time [8].

2.2 Challenges

The cloud's virtualized nature helps to enable Promising new use cases for efficient parallel data processing. However, 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 [6]. 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 [10]. 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 network 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. [4]), it is unclear if these techniques are applicable to IaaS clouds. For security reasons clouds often incorporate network virtualization techniques (e.g. [5]) which can hamper the inference process, in particular when based on latency measurements.

As a result, the only way to ensure locality between tasks of a processing job is currently to execute these tasks on the same VM in the cloud. This may involve allocating fewer, but more powerful VMs with multiple CPU cores. E.g., consider an aggregation task receiving data from seven generator tasks. Data locality can be ensured by scheduling these tasks to run on a VM with eight cores instead of eight distinct single-core machines. However, currently no data processing framework includes such strategies in its scheduling algorithms.

3. DESIGNING & IMPLEMENTATION

Based on the challenges and opportunities outlined in the previous section we have designed Nephele, a new data processing framework for cloud environments. Nephele takes up many ideas of previous processing frameworks but refines them to better match the dynamic and opaque nature of a cloud.

3.1 Architecture

Nephele's architecture follows a classic master-worker pattern as illustrated in Fig 1.

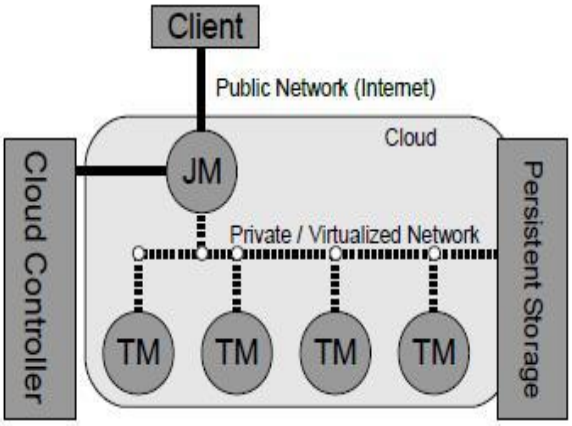

Fig 1. *Structural overview of Nephele running in an Infrastructure-as-a-Service (IaaS) cloud.*

Before submitting a Nephele 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 actual execution of tasks which a Nephele job consists of is carried out by a set of instances. Each instance runs a so-called Task Manager (TM). A Task Manager receives one or more tasks from the Job Manager at a time, executes them, and after that informs the Job Manager about their completion or possible errors. Unless a job is submitted to the Job Manager, we expect the set of instances (and hence the set of Task Managers) to be empty. Upon job reception the Job Manager then decides, depending on the job's particular tasks, how many and what type of instances the job should be executed on, and when the respective instances must be allocated/ deallocated to ensure a continuous but cost-efficient processing.The newly allocated instances boot up with a previously compiled VM image. The image is configured 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 VM 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 offer persistent storage. 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.

4. MODULES DESCRIPTION

4.1 Network Module

Server - Client computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients. Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with

servers which await (listen to) incoming requests.

4.2 Scheduling Task

The client initiates the task to be processes to the job manager, the job manager reads the task dispatches task, it coordinates and schedules the task to the task manager and allocates resource for processing it.

4.3 Client Module

He client which sends the request to the job manager for the execution of task the job manager will schedule the process and coordinates the task and wait for the completion response. The client is the one who initiates the request to the job manager.

4.4 Job Manager Module

The job manager will wait for the task from client, coordinates the process and it checks the availability of the server, if the server is available for the task to be done, it allocates the resource for execution and wait for the completion response. Figure shows the flow chart of job manager.

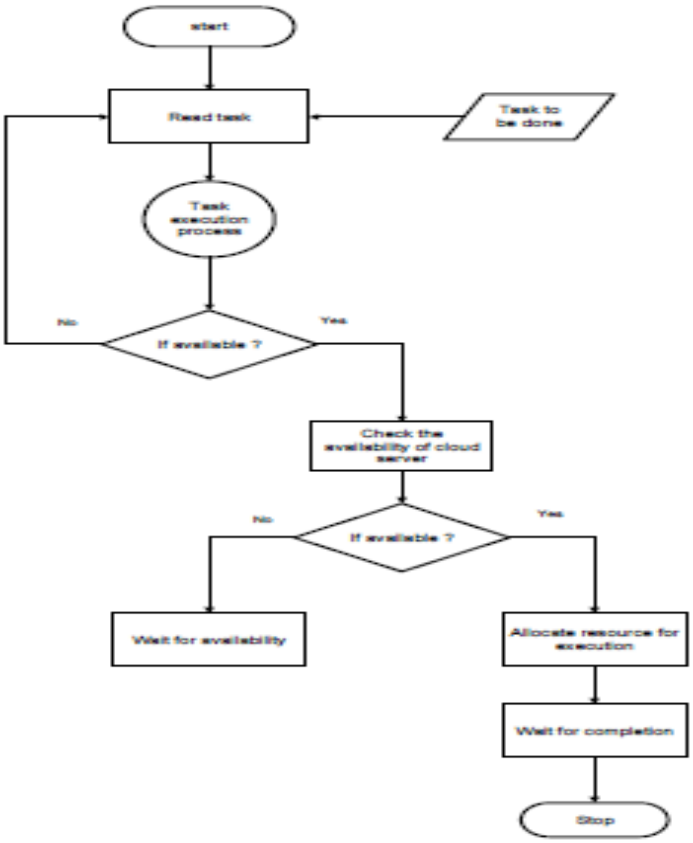

Figure 2. *Flowchart for Job Scheduling*

4.5 Cloud Controller Module

This acts a interface between the job manager and task manager and provides the control and initiation of task managers. It is also responsible for coordinating and manages the execution and also dispatches the task. It checks for the availability of task managers and allocate the resource for the task to be executed.

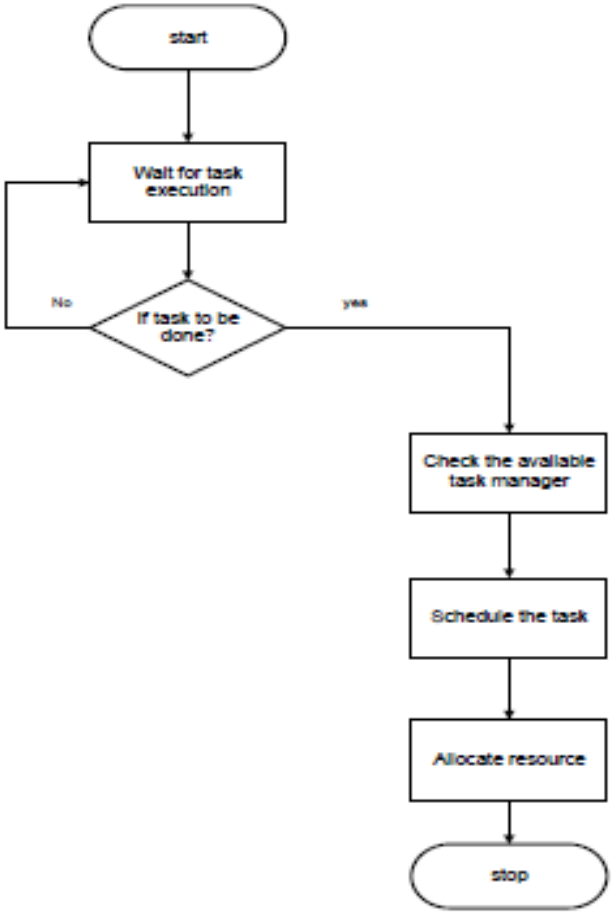

Figure 3. *Flowchart for Job Scheduling*

4.6 Task Manager Module

The task manager will wait for task to be executed; it then executes it and sends the complete response to the job manager in turn to the client. The actual execution of the task it done in the task manager.

4.7 RMI Protocol

RMI (Remote method invocation) use this simple Multiplexing protocol to allow a client to connect to an RMI server object in some situations.RMI objects and service remote calls from direct socket connections. The multiplexing protocol facilitates the use of virtual connections, which are themselves bidirectional, reliable byte streams, representing a particular session between two endpoints.

5. RESULT

This section will present the results of task scheduling for resource allocation and parallel processing by using RMI protocol and Nephele's architecture and process the task and allocate the resource for the executed task.

6. CONCLUSION

In this paper we have discussed the challenges and opportunities for scheduling and efficient parallel data processing and presented Nephele, the first data processing framework to exploit the dynamic resource provisioning offered by today's IaaS clouds. We have described Nephele's basic architecture which represents 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 at hand, there are a variety of open research issues, which we plan to address for future work.In particular,we are interested in improving Nephele's ability to adapt to resource overload or underutilization during the job execution automatically. In general, we think our work represents an important contribution to the growing fields and points out exciting new opportunities in the field of parallel data processing.

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] 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.

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

[6] 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.

[7] I. Raicu, I. Foster, and Y. Zhao. Many-Task Computing for Grids and Super computers. In Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on, pages 1–11, Nov.2008.

[8] M. Stillger, G. M. Lohman, V. Markl, and M.Kandil. LEO-DB2's LEarning Optimizer. In VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, pages 19–28, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[9] D. Warneke and O. Kao. Nephele: Efficient Parallel Data Processing in the Cloud. In MTAGS '09: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, pages 1–10, New York, NY, USA, 2009. ACM.

[10] T. White. Hadoop: The Definitive Guide. O'Reilly Media, 2009.