AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Distributed Data-Parallel Programs From Sequential Data Processing In Cloud

C. Vijayalakshmi, C. Vivek@Meenatchisundaram, S. Yamuna Devi and P. Abitha

Dept of CSE, Panimalar Engineering College Chennai, India

Address For Correspondence:
C. Vijayalakshmi, Dept of CSE, Panimalar Engineering College Chennai, India.
E-mail: vijidurai88@gmail.com

ARTICLE INFO

Article history:
Received 10 December 2015
Accepted 28 January 2016
Available online 10 February 2016

Keywords:
Many-task computing, high-throughput computing, loosely coupled applications, cloud computing

ABSTRACT

Map-Reduce are a programming model that enables easy development of scalable parallel applications to process vast amounts of data on large clusters of commodity machines. 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. Through a simple interface with two functions, map and reduce, this model facilitates parallel implementation of many real-world tasks such as data processing for search engines and machine learning. 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. 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.

INTRODUCTION

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. 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. 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. 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 allocated. 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 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.

Challenges And Opportunities:

Current data processing frameworks like Google's MapReduce or Microsoft's Dryad engine have been designed for cluster environments. This is reflected in a number of assumptions they make which are not

To Cite This Article: C. Vijayalakshmi, C. Vivek@Meenatchisundaram, S. Yamuna Devi and P. Abitha., Distributed Data-Parallel Programs From Sequential Data Processing In Cloud. ***Aust. J. Basic & Appl. Sci.,*** *10(1): 354-357, 2016*

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.

A. 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 computer 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. For, 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. Facilitating such 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 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 consists 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 de allocate it. The MapReduce 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.

B. 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. 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. 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 it is unclear if these techniques are applicable to IaaS clouds. For security reasons clouds often incorporate network virtualization techniques which can hamper the inference process, in particular when based on latency measurements. Even if it was possible to determine the underlying Network hierarchy in a cloud and use it for topology-aware.

Design:

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.

A. Architecture:

Nephele's architecture follows a classic master-worker pattern as illustrated in Fig. 1. 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 de allocate 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. For example, 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/de allocated to ensure a continuous but cost-efficient processing. Our current strategies for these decisions are highlighted at the end of this section.

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 (like, e.g., Amazon S3 2009). 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.

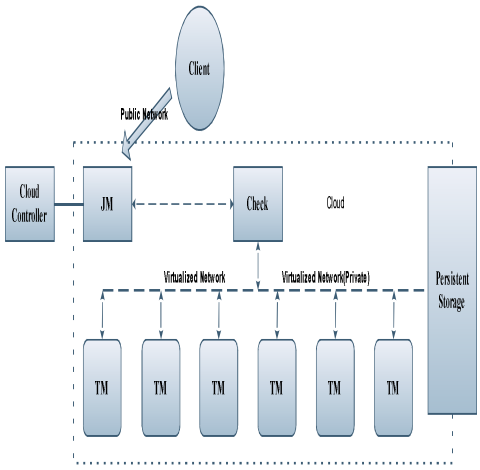

Fig. 1: Structural overview of Nephele running in an Infrastructure-as-a-Service (IaaS) cloud.

Evaluation:

In this section, we want to present first performance results of Nephele and compare them to the data processing framework Hadoop. We have chosen Hadoop as our competitor, because it is an open source software and currently enjoys high popularity in the data processing community. We are aware that Hadoop has been designed to run on a very large number of nodes (i.e., several thousand nodes). However, according to our observations, the software is typically used with significantly fewer instances in current IaaS clouds. In fact, Amazon itself limits the number of available instances for their MapReduce service to 20 unless the respective customer passes an extended registration process.

The challenge for both frameworks consists of two abstract tasks: Given a set of random integer numbers, the first task is to determine the k smallest of those numbers. The second task subsequently is to calculate the average of these k smallest numbers. The job is a classic representative for a variety of data analysis jobs whose particular tasks vary in their complexity and hardware demands. While the first task has to sort the entire data set, and therefore, can take advantage of large amounts of main memory and parallel execution, the second aggregation task requires almost no main memory and, at least eventually, cannot be parallelized. We

implemented the described sort/aggregate task for three different experiments. For the first experiment, we implemented the task as a sequence of MapReduce programs and executed it using Hadoop on a fixed set of instances. For the second experiment, we reused the same MapReduce programs as in the first experiment but devise a special MapReduce wrapper to make these programs run on top of Nephele. The goal of this experiment was to illustrate the benefits of dynamic resource allocation/ deallocation while still maintaining the MapReduce processing pattern. Finally, as the third experiment, we discarded the MapReduce pattern and implemented the task based on a DAG to also highlight the advantages of using heterogeneous instances. For all three experiments, we chose the data set size to be 100 GB. Each integer number had the size of 100 bytes. As a result, the data set contained about 109 distinct integer numbers. The cut-off variable k has been set to 2 _ 108, so the smallest 20 percent of all numbers had to be determined and aggregated.

Conclusion:

In this paper, we have discussed the challenges and opportunities for efficient parallel data processing in cloud environments 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 and presented a performance comparison to the well-established data processing framework Hadoop.

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/de allocate virtual machines in the course of a job execution, can help to improve the overall resource utilization and, consequently, reduce the processing cost. This template, modified in MS Word 2007 and saved as a "Word 97-2003 Document" for the PC, provides authors with most of the formatting specifications needed for preparing electronic versions of their papers.

All standard paper components have been specified for three reasons: (1) ease of use when formatting individual papers, (2) automatic compliance to electronic requirements that facilitate the concurrent or later production of electronic products, and (3) conformity of style throughout conference proceedings. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

REFERENCES

Amazon Web Services, L.L.C., 2009. "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/.

Amazon Web Services, L.L.C., 2009. "Amazon Elastic MapReduce," http://aws.amazon.com/elasticmapreduce/.

Amazon Web Services, L.L.C., 2009. "Amazon Simple Storage Service,"http://aws.amazon.com/s3/.

Battre´, D., S. Ewen, F. Hueske, O. Kao, V. Markl, D. Warneke, 2010. "Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing," Proc. ACM Symp. Cloud Computing (SoCC '10), 119-130.

Chaiken, R., B. Jenkins, P.A. Larson, B. Ramsey, D. Shakib, S. Weaver, J. Zhou, 2008. "SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets," Proc. Very Large Database Endowment, vol. 1, no. 2, pp. 1265-1276.

Chih Yang, H., A. Dasdan, R.L. Hsiao, D.S. Parker, 2007. "Map- Reduce-Merge: Simplified Relational Data Processing on Large Clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data.