ISSN: 2321-7782 (Online)
Volume 3, Issue 12, December 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Nephele as Efficient Framework in the Cloud

Mukku Sindhu[1]
M.Tech Scholar
Department of CSE
St.Mary's Group of Institutions Guntur
Chebrolu(V&M),Guntur(Dt),
Andhra Pradesh, India

Subhani Shaik[2]
M.Tech,(PhD)
Assoc Prof & Head of the Department
Department of CSE,
St.Mary's Group of Institutions Guntur
Chebrolu(V&M),Guntur(Dt),Andhra Pradesh, India

Abstract: ***Today, Infrastructure-as-a-Service (IaaS) cloud providers have incorporated parallel data processing framework in their clouds for performing Many-task computing (MTC) applications. Parallel data processing framework reduces time and cost in processing the substantial amount of users' data. Nephele is a dynamic resource allocating parallel data processing framework, which is designed for dynamic and heterogeneous cluster environments. The existing framework does not support to monitor resource overload or under utilization, during job execution, efficiently. Consequently, the allocated compute resources may be inadequate for big parts of the submitted job and unnecessarily increase processing time and cost. Nephele's architecture offers for efficient parallel data processing in clouds. It is the first data processing framework for 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.***

Index Terms: Parallel Data Processing, Dynamic resource allocation, IaaS, high-throughput computing, Nephele, Map Reduce

I. INTRODUCTION

GROWING organizations have been processing immense amount of data in a cost effective manner using cloud computing mechanism. More of the cloud providers like Google, Yahoo, Microsoft and Amazon are available for processing these data. Instead of creating large data centers which are expensive, these providers move into architectural paradigm with commodity servers, to process these huge data [3]. Problems such as processing crawled documents or web request logs are divided into subtasks and these subtasks are distributed into the available nodes for parallel processing. Cloud computing is the delivery of computing and storage capacity as a service to a community of end-recipients. The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts services with a user's data, software and computation over a network. It is difficult for him to maintain all the software's and data in local servers which costs a lot. His business is constantly running out of storage space and fixing broken servers. Then he came to know about cloud computing, which gives a solution to his problem. Instead of maintaining all the data and software's at local machines, he can now depend on another company which provides all the features that he requires and are accessible through the Web. **"This is called Cloud Computing".**

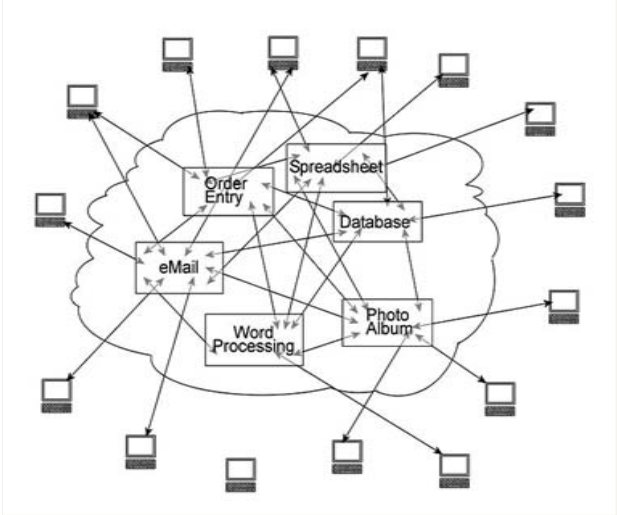


Fig1:Cloud Computing

II. Cloud Computing Layers

Cloud computing is made up of a variety of layered elements, starting at the most basic physical layer of storage and server infrastructure and working up through the application and network layers.

The three cloud layers are:

» Infrastructure cloud: Abstracts applications from servers and servers from storage

» Content cloud: Abstracts data from applications

» Information cloud: Abstracts access from clients to data

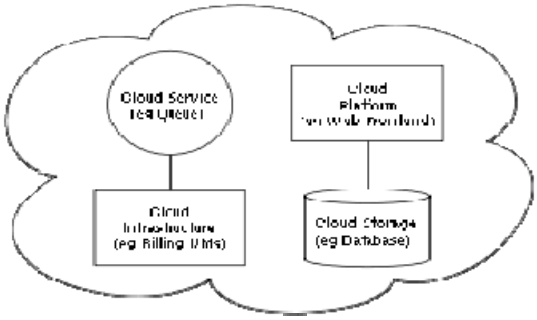


Fig 2:Cloud Computing Layers

III. Cloud Computing Models

» Private cloud: Created and run internally by an organization or purchased and stored within the organization and run by a third party

» Hybrid cloud: Outsources some but not all elements either internally or externally

» Public cloud: No physical infrastructure locally, all access to data and applications is external

3.1 Infrastructure cloud:

Includes the physical components that run applications and store data. Virtual servers are created to run applications, and virtual storage pools are created to house new and existing data into dynamic tiers of storage based on performance and reliability requirements. Virtual abstraction is employed so that servers and storage can be managed as logical rather than individual physical entities.

3.2 Content cloud:

Implements metadata and indexing services over the infrastructure cloud to provide abstracted data management for all content. The goal of a content cloud is to abstract the data from the applications so that different applications can be used to access the same data, and applications can be changed without worrying about data structure or type. The content cloud transforms data into objects so that the interface to the data is no longer tied to the actual access to the data, and the application that created the content in the first place can be long gone while the data itself is still available and searchable.

3.3 Information cloud:

The ultimate goal of cloud computing and the most common from a public perspective. The information cloud abstracts the client from the data. For example, a user can access data stored in a database in Singapore via a mobile phone in Atlanta, or watch a video located on a server in Japan from his a laptop in the U.S. The information cloud abstracts everything from everything. The Internet is an information cloud.

IV. PROBLEM STATEMENT

The IaaS cloud providers integrate the processing frame-work to reduce the processing time and provide simplicity to the users. Reducing process time leads to reduction in the cost for attracting the users to use their cloud services. Several frameworks have been developed with some specific features (e.g. To reduce cost or increase performance) for cloud which reduce the complexities for the user. However, the existing well known frameworks like Google's Map Reduce, Yahoo's Map Reduce Merge need the job to be written in a distinct map and reduce program by the developer. 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. Most notably, Nephele is the first data processing framework to include the possibility of dynamically allocating/ de-allocating different compute resources from a cloud in its scheduling and during job execution.

V. SYSTEM ANALYSIS

5.1 Economical Feasibility:

In this project the Sun java language is used for developing the piracy protection software, java is the product from sun Microsystems and it is provided for free of cost. So, it is bought without spending money. It can be downloaded from the sun web site directly from the Internet itself. So, it is not needed to look from any third party or in the market. It is freely download it from the Internet. This Exact Knowledge Hiding through Database Extension Software will be cost effective.

5.2 Technical Feasibility

Technical feasibility is important, but business need is even more important. It does no good to build a high tech system or product that no one really wants. The KTR developed using for minimum key management for the End user and is readily available with everyone so no need to search for any requirements and this software will work only in windows XP environment in order to pick up speed and also it is the advanced version of windows which available with everyone. The technical requirements like hardware and software requirements are available so it easily worked by other users.

5.3 Operational Feasibility

The Software that is developed is very user friendly. The user needs not to be a computer programmer. Even a computer literate who knows very basic things about the computer can work with this piracy protection software. This software itself will assist us in working with multimedia contents. This software is developed in fast growing language with updated features.

VI. Security And Privacy

It's impossible to develop a single data-protection solution for the cloud because the term means too many different things. Any progress must first occur in a particular domain—accordingly, our work focuses on an important class of widely used applications that includes e-mail, personal financial management, social networks, and business tools such as word processors and spreadsheets. The following criteria define this class of applications:

» provide services to a large number of distinct end users, as opposed to bulk data processing or workflow management for a single entity;

» use a data model consisting mostly of sharable units, where all data objects have access control lists (ACLs) with one or more users; developers could run the applications on a separate computing platform that encompasses the physical infrastructure, job scheduling, user authentication, rather than implementing the platform themselves.

VII. 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.

7.1 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. 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 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 [6].

7.2 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. All throughput of the cluster can be improved. In a cloud

this topology information is typically not exposed to the customer [7]. 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.

VIII. Related Work

In this section we mainly discuss with some existing frameworks that were already implemented for efficient parallel data processing.

8.1 SCOPE ((Structured Computations Optimized for Parallel Execution)

For Companies providing cloud-scale services of Easy and Efficient Parallel Processing of Massive Data Sets have an increasing need to store and analyze massive data sets such as search logs and click streams. For cost and performance reasons, processing is typically done on large clusters of with a huge amount needed to deploy large servers (I.e. shared-nothing commodity machines). It is imperative to develop a programming model that hides the complexity of the underlying system but provides flexibility by allowing users to extend functionality to meet a variety of requirements.

8.2 SWIFT

We present Swift, a system that combines a novel scripting language called Swift Script with a powerful runtime system based on Falkon, and Globus to allow for the concise specification, and reliable and efficient execution, of large loosely coupled computations. Swift adopts and adapts ideas first explored in the virtual data system, improving on that system in many regards. But this was not at all useful as this script was not in usage for the current developers, so that's the reason this framework is not popular in usage.

8.3 FALKON

Fast And Light-Weight Task Execution Framework .To enable the rapid execution of many tasks on compute clusters, we have developed Falkon. Falkon integrates multi-level scheduling to separate resource acquisition (via, e.g., requests to batch schedulers) from task dispatch, and a streamlined dispatcher. Falkon's integration of multi-level scheduling and streamlined dispatchers delivers performance not provided by any other system. We describe Falkon architecture and implementation, and present performance results for both micro benchmarks and applications.

8.4 HADOOP

Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage in static mode. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. This was not suitable in parallel processing as this was not suitable for dynamic nature, hence we came with a proposed new framework called Nephele Framework.

IX. Nephele Framework

9.1 Structural Overview of Nephele

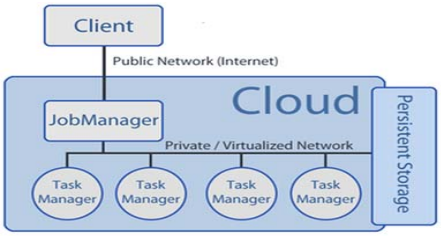


Fig:3 Structural overview of Nephele running in an Infrastructure-as-a-Service (IaaS) in cloud

Nephele is a massively parallel data flow engine dealing with resource management, work scheduling, communication, and fault tolerance. Nephele can run on top of a cluster and govern the resources itself, or directly connect to an IaaS cloud service to allocate computing resources on demand. Before submitting a Nephele compute job, a user must start an instance inside the cloud which runs the so called Job Manager. The Job Manager receives the client's jobs, is responsible for scheduling them and coordinates their execution. It can allocate or deallocate virtual machines according to the current job execution phase. The actual execution of tasks is carried out by a set of instances. Each instance runs a local component of the Nephele framework (Task Manager). A Task Manager receives one or more tasks from the Job Manager at a time, executes them and 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.

9.2. Nephele Architecture:

A system architecture or systems architecture is the conceptual design that defines the structure and/or behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. This may enable one to manage investment in a way that meets business needs. The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The composite of the design architectures for products and their life cycle processes. A representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components. An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline. Architecture is the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.

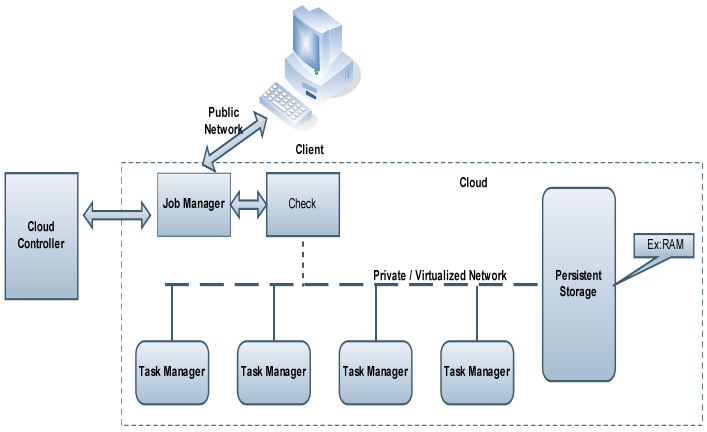


Fig 4: Architecture Diagram for Proposed Framework for Parallel data Processing in the Cloud

9.3 Advantage:

Unlike existing systems (Building a high level language layer that abstracts from underlying massively parallel data processing systems is a quite common concept. For Hadoop there exist several such layers like Apache Pig, Jaql, and Hive.) We chose Nephele for the following two reasons even in principle the presented PACT programming model could be set up to run on top of other DAG-based execution engines as well since the Nephele is benefits in the followings:

» First, Nephele offers a rich set of parameters which allow influencing the physical execution schedule of the received data flow program in a versatile manner.

» Second, Nephele is highlighted by its ability to deal with dynamic resource allocation. The system can independently request new compute nodes from a commercial cloud like Amazon EC2 in order to match the occurring workload. Although this feature is currently only used in a very limited fashion, we plan to leverage it for load balancing and dynamic re-optimization.

9.4 Working principle

Nephele supports three different types of communication channels: Network, in-memory, and file channels. While network and in-memory channels allow the PACT compiler to construct low-latency execution pipelines in which one task can immediately consume the output of another, file channels collect the entire output of a task in a temporary file before passing its content on to the next task .

9.5 Job graph:

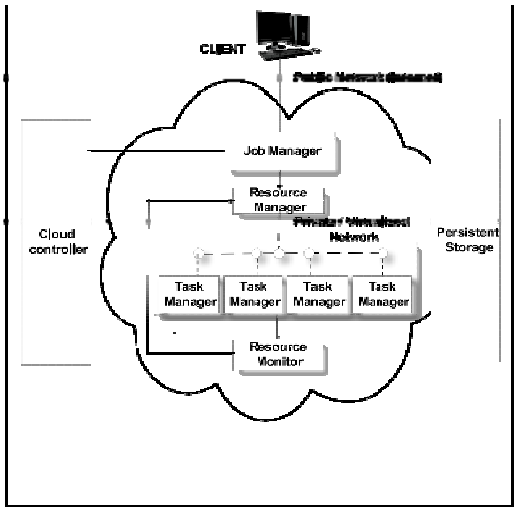


Fig:5 An example of a Job Graph in Nephele

The simplest Job graph which consists of one input, one Task and one output vertex. For generating the job graph user must have some ideas about aspects like number of subtasks, number of subtasks per instances, sharing instances between tasks, channel types and instance types for job descriptions

9.6 Job Execution and Scheduling:

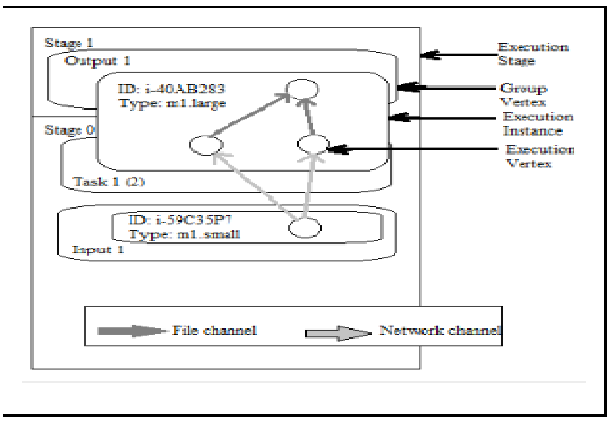


Fig:6 An Execution Graph created from the original Job Graph

After receiving the valid Job Graph the JM converts it into the Execution Graph which is the primary data structure for scheduling and monitoring the execution of the extended Nephele job. It contains all the concrete information required to schedule and execute the tasks in the cloud. Fig. 3. Shows the Execution Graph for the given Job Graph (i.e, Fig. 2.). Here Task 1 is, e.g., Split into two parallel subtasks which are both connected to the task Output 1 using file channels and are all scheduled to run on the same instance.

9.7 Nephele System Configuration:

Nephele computes a job, 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 a128 GB disk while the instance type "c1.xlarge" could refer to machines with 8 CPU cores, 18 GB RAM, and a 512 GB disk.

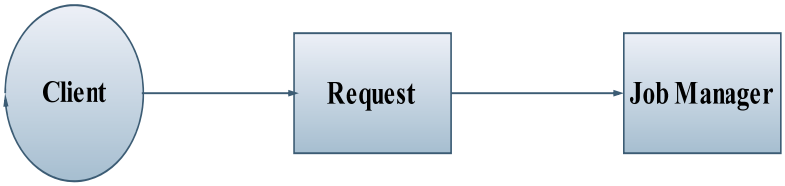
X. IMPLEMENTATION MODULES

For implementing this proposed framework, we divide the paper into four modules for implementing in Java Technology. The following are the four modules:

1. Job Manager (JM).
2. Task Manager(TM).
3. Parallel Data Processing.
4. Cloud Controller.

10.1 Job Manager (JM)

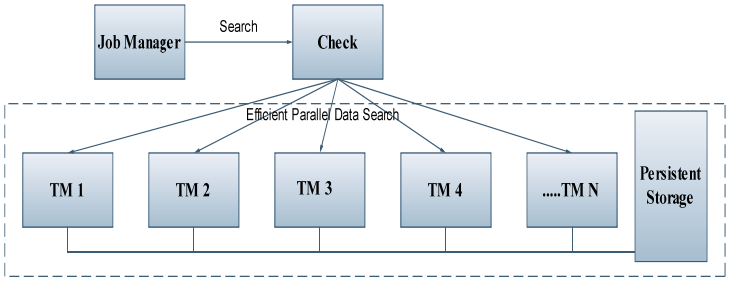
The Job Manager receives the client's jobs, is responsible for scheduling them, and coordinates their execution. It is capable of communicating with the interface the cloud operator provides to control the instantiation of VMs. We call this interface the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or deallocate VMs according to the current job execution phase. This module provides the tools for managing job openings and job applications. The functionality is divided into two distinct sections. The first one allows the administration staff to create/edit jobs, manage job applications and set the appearance of the public page. The second section provides the job applicant with information about the job openings and with tools to apply for a job of interest.



10.2 Task Manager(TM)

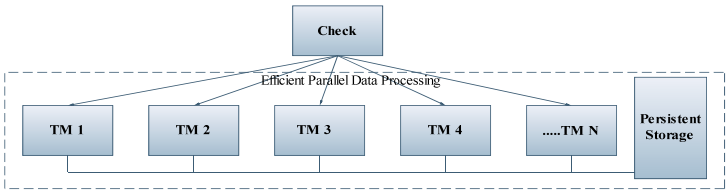
A task manager is a program used to provide information about the processes and programs running on a computer, as well as the general status of the computer. It can also be used to terminate processes and programs, as well as change the processes priority. Task managers can display currently running services (processes) as well as those that were stopped. They can display information about the services. 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.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. 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. 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.



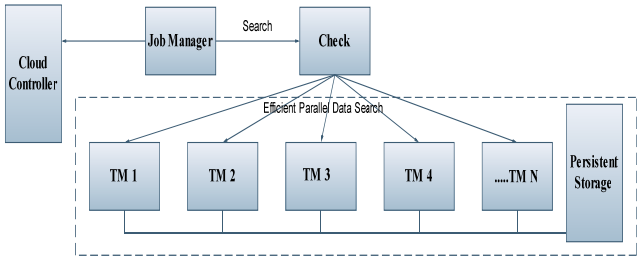
10.3 Parallel Data Processing

In order to simplify parallel data processing on large clusters the simple technique that interface the Job manager and Task manager with parallel data processing. To reduce the amount of data needed to transfer across network, a Combiner function is run on the same machine that ran a Map task. The Combiner merges the intermediate results on the local disk, before it is transferred to the corresponding Reduce task. Map tasks often produce many key/value pairs with the same key. This way those key/value pairs with the same key are merged, instead of transferring them all individually. Map-Reduce can be considered as an extension to the Map Reduce programming model, rather than an implementation of Map Reduce. Original Map Reduce programming model does not directly support processing multiple related heterogeneous datasets. For example, relational operations, like joining multiple heterogeneous datasets, can be done with Map Reduce by adding extra Map Reduce steps. Map-Reduce are an improved model that can be used to express relational algebra operators and join algorithms.



10.4 Cloud Controller

Cloud Controller is an administration tool for private cloud deployments. It manages cloud applications through their entire life cycle, from provisioning to monitoring, metering, and billing. Cloud Controller also supports hybrid clouds with cloud bursting to public clouds. It manages and verifies the both Job manager and Task manager in this controller to define the received tasks. Also by only paralyzing the scheme not serialize as literature concept. 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.



XI. RESULTS

Performance results of our three experiment, respectively. All three plots illustrate the average instance utilization over time, i.e., the average utilization of all CPU cores in all instances allocated for the job at the given point in time. The utilization of each instance has been monitored with the Unix command "top" and is broken down into the amount of time the CPU cores spent running the respective data processing framework (USR), the kernel and its processes (SYS), and the time waiting for I/O to complete (WAIT). In order to illustrate the impact of network communication, the plots additionally show the average amount of IP traffic flowing between the instances over time. We begin with discussing Experiment 1 (Map Reduce and Hadoop): For the first Map Reduce job, Tera Sort, shows a fair resource utilization. During the map (point (a) to (c)) and reduce phase (point (b) to (d)) the overall system utilization ranges from 60 to 80 percent.

XII. CONCLUSIONS AND FUTURE WORK

In this project, 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 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/deallocate virtual machines in the course of a job execution, can help to improve the overall resource utilization and, consequently, reduce the processing cost. In particular, we are interested in improving Nephele's ability to adapt to resource over load or underutilization during the job execution automatically. Our current profiling approach builds a valuable basis for this, however, at the moment the system still requires a reasonable amount of user annotations. In general, we think our work represents an important contribution to the growing field of Cloud computing services 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),",htthttp://aws.amazon.com/ec2 /, 2012
2. Amazon Web Services LLC, "Amazon Simple Storage Service,"http://aws.amazon.com/s3/ , 2012.
3. 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. Very Large Database
4. H. Chih Yang, A. Dasdan, R.-L. Hsiao, and D.S. Parker, "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2007
5. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Conf. Symp. Operating Systems Design and Implementation (OSDI '04), p. 10, 2004
6. 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," Scientific Programming, vol. 13, no. 3, pp. 219-237, 2005.
7. T. Dornemann, E. Juhnke, and B. Freisleben, "On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud," Proc. Ninth IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGRID '09), pp. 140-147, 2009
8. J. Frey, T. Tannenbaum, M. Livonia, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi- Institutional Grids," Cluster Computing, vol. 5, no. 3, pp. 237-246, 2002
9. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," Proc. Second ACM SIGOPS/EuroSys European Conf.Computer Systems (EuroSys '07), pp. 59-72, 2007[10]
10. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-So-Foreign Language for Data Processing," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 1099-1110, 2008.

11. R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, “Interpreting the Data: Parallel Analysis with Sawzall,” Scientific Programming, vol. 13, no. 4, pp. 277-298, 2005.

12. I. Raicu, I. Foster, and Y. Zhao, “Many-Task Computing for Grids and Supercomputers,” Proc. Workshop Many-Task Computing on Grids and Supercomputers, pp. 1-11, Nov. 2008.

13. L. Ramakrishnan, C. Koelbel, Y.-S. Lee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T.M. Huang, K. Thyagaraja, and D. Zagorodnov, “VGrADS: Enabling e-Science Workflows on Grids and Clouds with Fault Tolerance,” Proc. Conf. High Performance Computing Networking, Storage and Analysis (SC ’09), pp. 1-12, 2009.

14. The Apache Software Foundation “Welcome to Hadoop!” http://hadoop.apache.org/,2012

15. D. Warneke and O. Kao, “Nephele: Efficient Parallel Data Processing in the Cloud,” Proc. Second Workshop Many-Task Computing on Grids and Supercomputers (MTAGS ’09), pp. 1-10, 2009

16. T. White, Hadoop: The Definitive Guide. O’Reilly Media, 2009. Endowment, vol. 1, no. 2, pp. 1265-1276, 2008.

