Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficient Dynamic Resource Allocation in Parallel Data Processing for Cloud using PACT

Mr. Abhijit Adhikari
Assistant Prof. RMD Sinhgad COE,
Computer Department, Pune University, India

Gandhali Upadhye
RMD Sinhgad COE,
Computer Department, Pune University, India

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. Majority of the Cloud computing companies have started to integrate frameworks for parallel data processing in their product portfolio, making it easier for customers to access these services and to deploy their programs. The static and homogeneous cluster setups that are designed as processing frameworks which are currently used, ignore the precise nature of a cloud. Consequently, the allocation of compute resources may be inadequate for big parts of the submitted job and may unnecessarily increases the processing time and cost. In this paper, the challenges and opportunities for efficient parallel data processing in clouds are discussed and research project Nephele is presented. The first data processing framework for explicitly exploiting the dynamic resource allocation offered by today's IaaS clouds for both, task scheduling and execution is Nephele. 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, extended evaluations of Map Reduce-inspired processing jobs on an IaaS cloud system are performed and the results to the popular data processing framework Hadoop are compared.

Keywords— IaaS (Infrastructure-as-a-service), PACTs- Parallelization Contracts, Cloud Computing, Parallel Data Processing, Dynamic resource allocation.

I. Introduction

Cloud computing is a colloquial expression used to describe a variety of different types of computing concepts that involve a large number of computers connected through a real-time communication network. Cloud computing is a terminology without a commonly accepted unequivocal scientific or procedural definition. It is a synonym for distributed computing over a network which means the ability to run a program on many connected computers at the same time. The phrase is also; a lot of unremarkably wont to consult with network-based services that seem to be provided by real server hardware, however that if truth be told are served up by virtual hardware, simulated by software system running on one or a lot of real machines. Such virtual servers, physically don't exist and thus will be scaled up or down and affected around and on the fly while not poignant the top user-arguably, rather sort of a cloud. Cloud computing is having following types:

- Platform as a Service (PaaS),
- Infrastructure as a Service (IaaS),
- Software as a Service (SaaS).

Users can rent application software and databases, using Software as a Service. The infrastructure and platforms on which the applications run is managed by cloud providers.

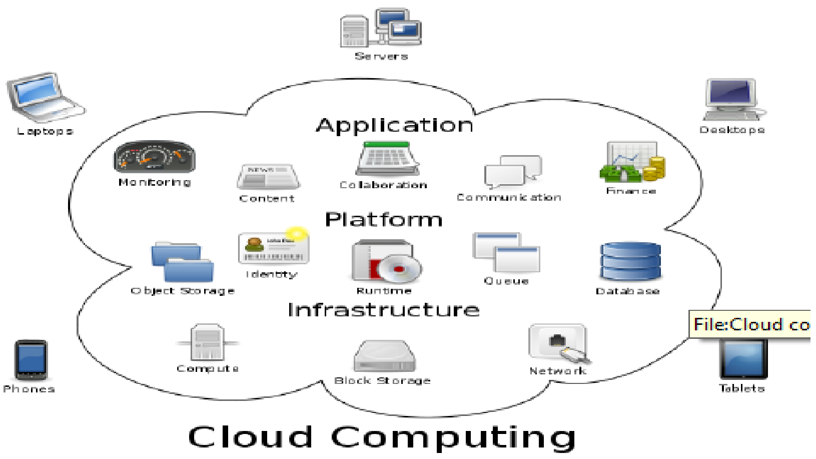

Fig 1. The architecture of cloud storage

Cloud-based applications can be accessed through a web browser or a light-weight desktop or mobile app by an end user while the business software and user's data are stored on servers at a remote location. To meet unsteady and unpredictable business demands, proponents of cloud computing claim that it permits enterprises to urge their applications up and running quicker, with improved tractableness and fewer maintenance, and therefore permits IT to sooner modify the resources. Cloud computing depends on sharing of resources to realize coherence and economies of scale similar to a utility (like the electricity grid) over a network (typically the Internet). The foundation of cloud computing is the comprehensive concept of converged infrastructure and shared services.

The main goal of this paper is to decrease the overloads of the main cloud and increase the performance of the cloud by segregating all the jobs of the cloud by cloud storage, task manager and job manager that performs the different task using different resources as the infrastructure needed. The companies to run with minimal resources and hire the resources needed on demand and pay only when there is a use is enabled by IaaS (Infrastructure as a Service). This becomes achievable by the Virtual Machines (VM) which runs at the data center to calculate the usage and necessary billing is done. Since the abstraction of VM, IaaS clouds fits the architectural paradigm assumed by the data processing frameworks described above, projects like Hadoop. This paper talks about the new framework for data processing called as “Nephele”. This framework enables the possibility of allocating and reallocating resources dynamically in a cloud during the execution of a job.

II. **LITERATURE SURVEY**

Recent data processing frameworks like Google’s Map Reduce or Microsoft’s Dryad engine have been designed for cluster environments [1] [2]. This is echoed in a number of assumptions they make which are not essentially valid in cloud environments. This section describes how these assumptions gives path for new framework. Today’s processing frameworks assume the resources they manage contain of a static set of homogeneous compute nodes. The numbers of accessible machines area unit thought-about to be constant, particularly once programing the process job’s execution and this can be designed to traumatize individual nodes failures. The flexibility of IaaS clouds remains unused while IaaS clouds can undoubtedly be used to create such cluster-like setups. One of the key features of an IaaS cloud’s the provisioning of compute resources on demand. New VMs can be allocated at any time and can become available in a matter of seconds through a well-defined interface. Machines that are no longer used can be terminated instantly and the customer of cloud will be charged for them no more. Cloud operators or VMs of different types, i.e. with different sizes of main memory, different computational power, and storage like Amazon is rented to customers. Dynamic and possibly heterogeneous computer resources are hence available in a cloud. Flexibility ends up in a spread of recent potentialities, notably for programing processing jobs, with relation to parallel processing.

The compute nodes are characteristically interconnected through a physical high-performance network, in a cluster. The way the compute nodes are physically wired to each other should be well known and more important is that it remains constant over a period of time; this is the topology of network. To improve the overall throughput of the cluster and to avoid bottlenecks current data processing frameworks offer to influence 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].

III. **THE SYSTEM MODEL**

Nephele, a new data processing framework for cloud environments. Many ideas of previous processing frameworks are taken by Nephele but refine them to better match the dynamic and opaque nature of a cloud. The Parallelization Contracts (PACTs) and the scalable parallel execution engine Nephele is centered on a programming model of parallel data processor. The PACT programming model is a generalization of the well-known map/reduce programming model, extending it with additional second-order functions, as well as with Output Contracts that guarantees about the behavior of a function. The complete system is meant to be as basic as (and compatible to) map/reduce systems, whereas overcoming many of their major weaknesses: 1) to express many data processing tasks both naturally and efficiently the functions map and reduce alone are not sufficient. 2) Execution strategy which is robust but highly suboptimal for many tasks ties the map/reduce to a single fixed program. 3) No assumptions about the behavior of the functions are made by Map/reduce.

The actual execution of tasks with Nephele job consists is carried out by a set of instances and this instance runs so-called Task Manager(TM). Task Manager receives one or additional tasks from the task Manager at a time, executes them and informs the task Manager concerning their completion or attainable errors. The set of instances expected is to be empty if the job is not submitted to the Job Manager. After job reception, the Job Manager decides, depending on job what type and how many instances the job should be executed on, and the instances must be allocated/de-allocated to ensure a continuous but cost-efficient processing.

Newly allocated instances boot with a previously compiled virtual machine image, image is configured to automatically start a Task Manager and register it with the Job Manager. Execution of the scheduled job is triggered when all Task Managers have contacted Job Managers.

A.Architecture

Job manager means a user must start an instance inside the cloud which runs before submitting a Nephele compute job. The client's job is responsible for scheduling and coordinating their execution which is received by Job Manager. The cloud controller through a web service interface is capable of communicating. According to the current job execution phase virtual machines can be allocated or deallocated. The term instance type will be used to differentiate between virtual machines with different hardware characteristics. It follows a simple master Slave Configuration which is shown in fig 2.

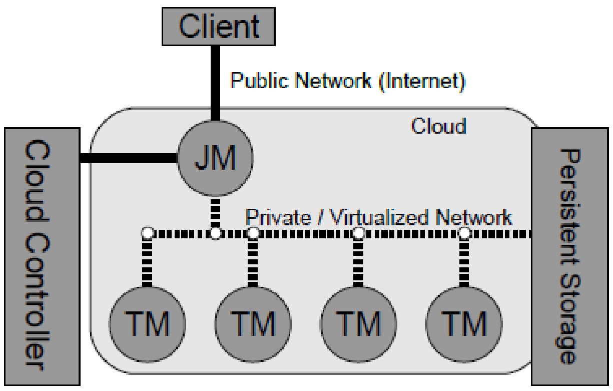

Fig 2. Structural overview of Nephele running in an Infrastructure-as-a-Service (IaaS) cloud

B. Job Description

Jobs in Nephele are expressed as a directed acyclic graph (DAG). A task of overall processing job is represented by each vertex in the graph and the communication flow between these tasks is defined by edges. DAG's are used to describe processing jobs for two reasons: first is that DAG allows tasks to have multiple inputs and output edges which simplify implementation of classic data combining functions [3]. Second, DAG edges explicitly model communication paths of processing job. In this case, the track of what instance requires data from what other instances and instance can be shut down and de-allocated kept by Nephele.Defining a Nephele job comprises following binding steps:

1. The user must connect to virtual machine and start his task
2. The task program must be assigned to a vertex
3. Finally, the vertices must be connected by edges to define the communication paths of the job.

Fig.3 shows the simplest possible Job Graph that consists of one task, one input, and one output vertex.

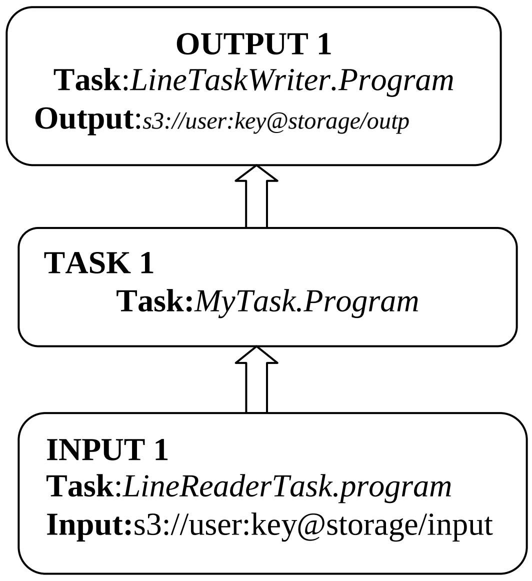
OUTPUT 1
Task:LineTaskWriter.Program
Output:s3://user:key@storage/outp
TASK 1
Task:MyTask.Program
INPUT 1
Task:LineReaderTask.program
Input:s3://user:key@storage/input

Fig.3. An example of Job Graph in Nephele

Expected tasks should contain sequential code and process called as records, the primary data unit in Nephele. Records can be of arbitrary types defined by Programmer's. These records enter and leave the task program via input or output gates, which are endpoints of DAG's edges. Regular tasks have at least one or more input and output gates. Tasks that represent the source or the sink of the data flow must not contain any input or output gates. After specifying code for tasks, user must define DAG to connect these tasks, call this DAG as Job Graph. Job Graph maps each task to vertex and determines communication paths between them. Number of incoming and outgoing edges must comply with number of input and output gates defined inside tasks. For executing task, input and output vertices can be associated with URL pointing to external storage facilities to read or write input or output data.

Job Graph consists of one input, one task and one output vertex. Goal of Job Graph is that users should be able to describe tasks and their relationships on abstract level. Job Graph does not explicitly model task parallelization and mapping of tasks to instances. Users who wish to influence these aspects provide annotations to their job description:

i. Number of Subtasks:-Developer can declare his task to be suitable for parallelization. Users include such tasks in their Job Graph can specify parallel subtasks. Nephele splits these tasks at runtime.

ii. Number of Subtasks per instance: - By default, each subtask is assigned to separate instance. Same instances are supposed to be shared by subtasks.

iii. Sharing instances between tasks: - Subtasks of different tasks are assigned to different instances unless prevented by another scheduling restriction.

iv. Channel Types: - Depending on each edge connection of two vertices, user can determine channel type. All edges of original Job Graph should be replaced by at least one channel of specific type before executing job in Nephele. Nephele supports network, file and in-memory channels. Choice of channel type has several implications on entire job schedule.

v. Instance Type: - Subtask can be executed on different instance types which may be more or less suitable for considered program.

Once Job Graph is specified, user submits it to Job Manager, together with credentials obtained from cloud operator.

C. Job Scheduling and Execution

After receiving valid Job Graph from user, Nephele's [3] Job Manager transforms it into Execution Graph. Nephele's primary data structure for scheduling and monitoring execution of Nephele Job is an Execution Graph that contains all concrete information required to schedule and execute received job on cloud. The proposed framework allows splitting the job into tasks and subtasks and also takes care of creating and terminating VM instances besides managing them for efficiency. In order to overcome the problem of availability of VMs, the framework allows the dividing the execution graph into many execution stages. VM instances are associated with stages so that it is easy to manage those VM instances with stages. These stages act as checkpoints that ensure the complete execution of tasks without causing problems.

All the information required to schedule and executes the received job on the cloud contains the Execution Graph. It explicitly models the mapping of tasks to instances and task parallelization. Nephele may have different degrees of freedom in constructing the Execution Graph, depending on the level of annotations the user has provided with Job Graph. Task 1 is e.g. split into two parallel subtasks which are both connected to the task Output 1 via file channels and are all scheduled to run on the same instance. The structure of the Execution Graph is explained in the Fig.4:

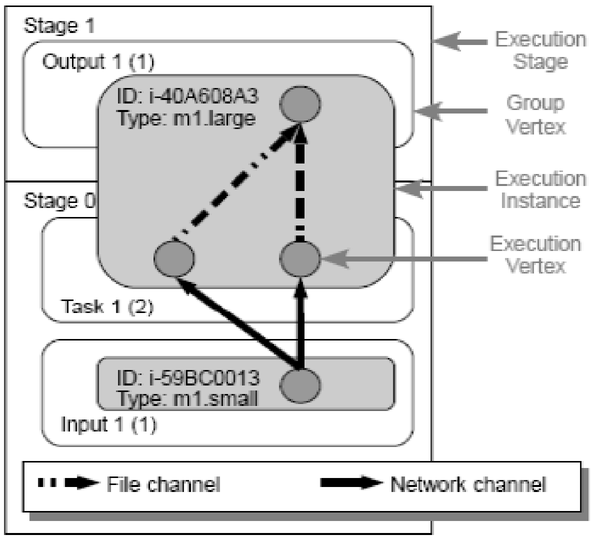

Fig.4. Execution Graph created from the Job Graph

Before processing can begin, Nephele requires all edges of an Execution Graph to be replaced by a channel. How records are transported from one subtask to the other is determined by the type of channel. Currently, three different types of channels are featured by Nephele, which all put different constrains on the Execution Graph [4] [5].

1. Network channels: A network channel rents two subtasks exchange data via a TCP connection and allows pipelined processing, so that the emission of records is done by the producing subtask that are instantaneously transported to the consuming subtask [7]. As a result, two subtasks that are connected via a network channel may be executed on different instances.

2. In-Memory channels: Similarly, pipelined processing is also allowed in in-memory channel. Nonetheless, instead of using a TCP connection, the individual subtasks exchange data using the instance's main memory. The fastest way to transport records in Nephele is represented by in-memory channel; however, it also suggests most scheduling restrictions: The two connected subtasks must be scheduled to run on the same instance and run in the same Execution Stage.

3. File channels: By using local file system a file channel allows two subtasks to exchange records. The records of the producing task are first written to an intermediary file and later read into the consuming subtask. Two such subtasks need to be assigned in Nephele to the same instance. Additionally, the consuming Group Vertex must be scheduled to run in a higher Execution Stage than the producing Group Vertex [8]. Over-all, Nephele only permits subtasks to exchange records across diverse stages via file channels since they are the only channel types which store the intermediate records in a persistent manner.

i. Input Design

The link between the information system and the user is the input design. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. Controlling the amount of input required, controlling the errors, avoiding extra steps, avoiding delay and keeping the process simple is the main focus of the input design. The main considerations for designing the input are security and ease of use with holding the privacy. While designing Input following things are to be considered:

- What information ought to be given as input?
- How the data ought to be organized or coded?
- The dialog to guide the in operation personnel in providing input.
- Methods for making ready input validations and steps to follow once an error occurs.

ii. Objectives

1. The process of converting a user-oriented description of the input into a computer-based system is Input Design, which is important to avoid errors in the data input process and form computerized system it shows the correct direction to the management for getting correct information.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of input is to be free from errors and to make data entry easier. The screen of data entry is designed in such a manner that all the data manipulations can be accomplished. Also there is a provision for record viewing facilities.

3. It will check for its validity when the data is entered. With the help of screens data can be entered. Appropriate messages are provided as and when required so that the user will not be in maize of instant. Therefore the aim of input design is to create an input layout which is easy to follow.

iii. Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. The results of processing are communicated to the users and to other system through outputs in any system. The hard copy output and also how the information is to be displaced for immediate need are determined in output design. It is the most important and direct source information to the user. The system's relationship improves efficient and intelligent output design to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the proper output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. Once analysis designs computer output, they should recognize the precise output that is required to fulfill the requirements.
2. Select methods for presenting information.
3. Create document, report, or alternative other formats that contain information made by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information concerning past activities, current standing or projections of the longer term.
- Signal key, problems, events, warnings, or opportunities.
- Trigger an action.
- Confirm an action.

IV. CONCLUSIONS

The challenges and opportunities for efficient parallel data processing in cloud environments are analyzed using Nephele, processing framework to take the advantage of the dynamic resource provisioning offered by today's IaaS clouds job, similarly the possibility to automatically allocate or deallocate virtual machines within the course of a job execution, that consequently reduce the processing cost and helps to improve the overall resource utilization. In general, the work above 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] D. Battr´e, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/PACTs: A Programming Model andExecution Framework for Web-Scale Analytical Processing. In SoCC '10: Proceedings of the ACM Symposium onCloud Computing 2010, pages 119– 130, New York, NY, USA, 2010. ACM.

[2] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. SCOPE: Easy and EfficientParallel Processing of Massive Data Sets. Proc. VLDB Endow., 1(2):1265– 1276, 2008.

[3] H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on LargeClusters. In SIGMOD '07: Proceedings of the 2007 ACM SIGMO Dinternational conference on Management of data, pages 1029–1040,New York, NY, USA, 2007. ACM.

[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] Amazon Web Services LLC. Amazon elastic Compute Cloud Amazon EC2) http://aws.amazon.com/ec2/,2009

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

[7] I. Raicu, I. Foster, and Y. Zhao. Many-Task Computing for Grids and Supercomputers. 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.