IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 2, Issue 3 (July-Aug. 2012), PP 01-05
www.iosrjournals.org

Efficient Parallel Data Processing For Resource Sharing In Cloud Computing

R.Balasubramanian[1], Dr.M.Aramuthan [2]
[1](*Research Scholar, Manonmanium Sundaranar University,India)*
[2](*Asst.Professor, Perunthalaivar Kamarajar Institute of Engineering& Technology,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. 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. However, 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. 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.*
Key words: *Cloud computing, Nephele, Sharing, Fish Algorithm, Feistel Networks, VMs.*

I. Introduction

In recent years 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 Drya or Yahoo!'s Map-Reduce-Merge. They can be classified by terms like high throughput. 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.

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 (IaaS) clouds like Amazon EC2, let their customers allocate, access, and control a set of virtual machines (VMs) which run inside their data centers and only charge them for the period of time the machines are allocated. The VMs are typically offered in different types, each type with its own characteristics (number of CPU cores, amount of main memory, etc.) and cost. Since the VM abstraction of IaaS clouds fits the architectural paradigm assumed by the data processing frameworks described above, projects like Hadoop a popular open source implementation of Google's Map Reduce framework, already have begun to promote using their frameworks in the cloud .Hadoop as one of its core infrastructure services. However, instead of embracing its dynamic resource allocation, current data processing frameworks rather expect the cloud to imitate the static nature of the cluster environments they were originally designed for.

In this project 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 along with the security of the data. Most notably, Nephele is the first data processing framework to include the possibility of dynamically allocating and reallocating different compute resources from a cloud in its scheduling and during job execution. It includes further details on scheduling strategies and cryptographic algorithms.

A. Implementation:

1) Resource Sharing

This allows different customers to share the same resource at the same time. There are three models you can use:

a) Sharing is disabled (use in most cases) -- this will let only one customer reserve given resource at one time

b) Allowed -- it's up to the customer to decide whether other customers can share the resource with them or if they wish to use the resource exclusively. An example where this can be used is a shuttle bus rental which normally would also take other customers but sometimes a customer may wish to use the entire shuttle van exclusively. In pricing manager you can specify a rule which would update the price in such case.

c) Always possible -- this forces all customers to share the resource with the others. This mode should be used whenever you sell tickets or seats or any other kind of resource which is used on per-person basic (including bikes, personal gear, dormitory bed etc.). If you select this option, two things will happen: planyo will add a new reservation form item Number of persons where the customer can set the number of tickets/seats they wish to reserve. Another change is that by selecting always possible the monthly price in case of Planyo PRO will be lower (for shared resources planyo counts the number of resources as a half of the total seats/tickets available).

II. Algorithms

A. Job Scheduling and Execution:

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

B. Parallelization and Scheduling Strategies:

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

C. Two fish Algorithm:

Two fish is a block cipher by Counterpane Labs. It was one of the five Advanced Encryption Standard (AES) finalists. Two fish is unpatented, and the source code is uncopyrighted and license-free; it is free for all uses.

D. General Description:

Two fish is a 128-bit block cipher that accepts a variable-length key up to 256 bits. The cipher is a 16-round Feistel network with a bijective F function made up of four key-dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix over GF(28), a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key schedule. A fully optimized implementation of Two fish encrypts on a Pentium Pro at 17.8 clock cycles per byte, and an 8-bit smart card implementation encrypts at 1660 clock cycles per byte. Two fish can be implemented in hardware in 14000 gates. The design of both the round function and the key schedule permits a wide variety of tradeoffs between speed, software size, key setup time, gate count, and memory. We have extensively crypt analyzed Two fish; our best attack breaks 5 rounds with 222.5 chosen plaintexts and 251 effort.

128-bit block
128-, 192-, or 256-bit key
16 rounds
Works in all standard modes
Encrypts data in:
18 clocks/byte on a Pentium
16.1 clocks/byte on a Pentium Pro

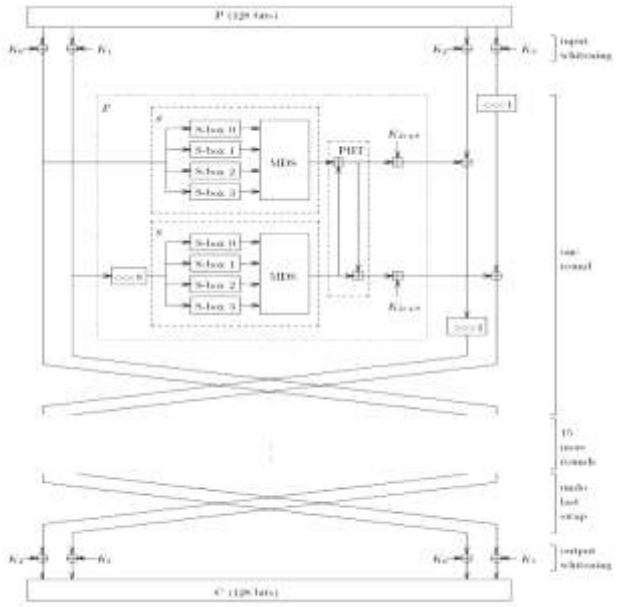


Figure 1: Twofish

E. Blow fish Algorithm:

The data transformation process for Pocket Brief uses the Blowfish Algorithm for Encryption and Decryption , respectively. The details and working of the algorithm are Given below.

Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses.

Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors. Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches.

F. Feistel Networks:

A Feistel network is a general method of transforming any function (usually called an Function) into a permutation. It was invented by Horst Feistel and has been used in many block cipher designs. The working of a Feistal Network is given below:

Split each block into halves

Right half becomes new left half

New right half is the final result when the left half is XOR'd with the result of

applying f to the right half and the key.

Note that previous rounds can be derived even if the function f is not invertible.

G. The Blowfish Algorithm:

Manipulates data in large blocks

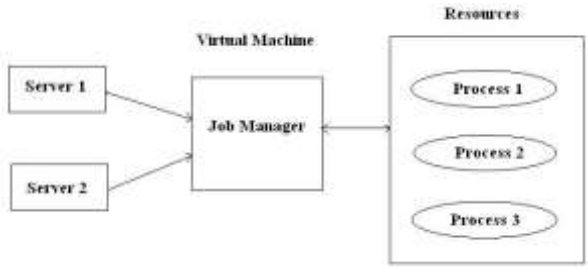
Has a 64-bit block size.

Has a scalable key, from 32 bits to at least 256 bits.

Uses simple operations that are efficient on microprocessors.e.g., exclusive-or, addition, table lookup, modular-multiplication. It does not use variable-length shifts or bit-wise permutations, or conditional jumps.

Employs precomputable subkeys. On large-memory systems, these subkeys can be precomputed for faster operation. Not precomputing the subkeys will result in slower operation, but it should still be possible to encrypt data without any precomputations consists of a variable number of iterations. For applications with a small key size, the trade-off between the complexity of a brute-force attack system. Thus, in analysis it is possible to produce the correct model of existing system.

H. System architecture:



III. Database Design

The database design is a must for any application developed especially more for the data store projects. Since the chatting method involves storing the message in the table and produced to the sender and receiver, proper handling of the table is a must. In the project, admin table is designed to be unique in accepting the username and the length of the username and password should be greater than zero.

IV. Module Description

A. 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 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.

B. Allocate Task:

In this Module the service ask the processing data for the encryption process. Client user has to give the related data for task scheduling. Here we can see the given datas in text area and also have to select the encryption model's orders for processing. Then have to precede the data to task scheduling process.

C. Scheduled Task:

Here the tasks send by the multiple clients are scheduled in processing area. This module arranged the each and every task in FIFO manner. Here it shows the each and every task in list area. After scheduled the task it will precede the each and every data to the Virtual machine process.

V. Processing In Virtual Machine

Virtual machine receives the each and every process and arranges it for the processing servers. It is used to search the free server for processing job. Suppose if every server is in working process, it makes the remaining task to wait stage. It is only arrange the job to the server and respond the output to the client.

A. Processing Task:

This is the area of server to processing the encryption for each and every request given from the client. Here some more servers are available for large number of processing task. Each server will do the prober encryption process. After processing the job it will forward that to the virtual machine.

VI. Conclusion

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/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. 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.

VII. Future Enhancements

We discussed the resource sharing in the cloud and also access the resource the in parallel time. Here we are using the Encryption and decryption process as a resource and input job as a user given data. Here the input job is the word document. In the future we can make it to the input design in any type of document like word, txt, jpeg and mpeg and so on. Also here we using the three encryption process such as Tea, Two fish and Blow fish algorithm and in the future we can use much more algorithm for the sharing process. And also we can do the compression and decompression itself with that encryption process.

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. ttp://aws.amazon.com/elasticmapreduce/, 2009.
[3] AmazonWeb Services LLC. Amazon Simple Storage Service. http://aws.amazon.com/s3/, 2009.
[4] D. Battr´e, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke.Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In SoCC '10: Proceedingsof the ACM Symposium on Cloud Computing 2010, pages 119–130, New York, NY, USA, 2010. ACM.
[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib,S. Weaver, and J. Zhou. SCOPE: Easy and EfficientParallel Processing of Massive Data Sets. Proc. VLDB Endow., 1(2):1265–1276, 2008.
[6] H. Chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Man