International Journal of Computer Applications Technology and Research
Volume 1– Issue 3, 79-81, 2012

EXPLOITING DYNAMIC RESOURCE ALLOCATION FOR EFFICIENT PARALLEL DATA PROCESSING IN CLOUD-BY USING NEPHEL'S ALGORITHM

Kavya Jakkula
Department of Computer Science and Engineering
Kottam College of Engineering
Chinnatekur, Kurnool, A.P, 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. Based on this new framework, we perform extended evaluations of Map Reduce-inspired processing jobs on an IaaS cloud system and compare the results to the popular data processing framework Hadoop.

.**Keywords**: Iaas-Infrasturcture-as-a-service, PACTs- Parallelization Contracts.

1. INTRODUCTION

The main goal of our project is to decrease the overloads of the main cloud and increase the performance of the cloud. 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. The main objective of our project 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, job manager and task manager. and perform the different task using different resources as the infrastructure needed.

2. Companies providing cloud-scale services 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 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. In this paper, we present a new declarative and extensible scripting language, SCOPE (Structured Computations Optimized for Parallel Execution), targeted for this type of massive data.

2. NEPHELE/PACT ALGORITHM

We present a parallel data processor centered around a programming model of so called Parallelization Contracts (PACTs) and the scalable parallel execution engine Nephele. The PACT programming model is a generalization of the well-known map/reduce programming model, extending it with further second-order functions, as well as with Output Contracts that give guarantees about the behavior of a function. We describe methods to transform a PACT program into a data flow for Nephele, which executes its sequential building blocks in parallel and deals with communication, synchronization and fault tolerance. Our definition of PACTs allows applying several types of optimizations on the data flow during the transformation. The system as a whole is designed to be as generic as (and compatible to) map/reduce systems, while overcoming several of their major weaknesses: 1) the functions map and reduce alone are not sufficient to express many data processing tasks both naturally and efficiently. 2) Map/reduce ties a program to a single fixed execution strategy, which is robust but highly suboptimal for many tasks. 3) Map/reduce makes no assumptions about the behavior of the functions. Hence, it offers only very limited optimization opportunities. With a set of examples and experiments, we illustrate how our system is able to naturally represent and efficiently execute several tasks that do not fit the map/reduce model well.The term Web-Scale Data Management has been coined for describing the challenge to develop systems that scale to data volumes as they are found in search indexes, large scale warehouses, and scientific applications like climate research. Most of the recent approaches build on massive parallelization, favoring large numbers of cheap computers over expensive servers. Current multicore hardware trends support that development. In many of the mentioned scenarios,

Parallel databases, the traditional workhorses, are refused. The main reasons are their strict schema and the missing scalability, elasticity and fault tolerance required for setups of 1000s of machines, where failures are common. Many new architectures have been suggested, among which the map/reduce paradigm and its open source implementation Hadoop have gained the most attention. Here, programs are written as map and reduce functions, which process key/value pairs and can be executed in many data parallel instances. The big advantage of that programming model is its generality: Any problem that can be expressed with those two functions can be executed by the framework in a massively parallel way. The map/reduce execution model has been proven to scale to 1000s of machines. Techniques from the map/reduce execution model have found their way into the design of database engines and some databases added the map/reduce programming model to their query interface .The map/reduce programming model has however not been designed for more complex operations, as they occur in fields like relational query processing or data mining. Even implementing a join in map/reduce requires the programmer to bend the programming model by creating a tagged union of the inputs to realize the join in the reduce function. Not only is this a sign that the programming model is somehow unsuitable for the operation, but it also hides from the system the fact that there are two distinct inputs. Those inputs may be treated differently, for example if one is already partitioned on the key. Apart from requiring awkward programming, that may be one cause of low performance .Although it is often possible to force complex operations into the map/reduce programming model, many of them require to actually describe the exact communication pattern in the user code, sometimes as far as hard coding the number and assignment of partitions. In consequence, it is at least hard, if not impossible, for a system to perform optimizations on the program, or even choose the degree of parallelism by itself, as this would require modifying the user code. Parallel data flow systems, like Dryad [10], provide high flexibility and allow arbitrary communication patterns between the nodes by setting up the vertices and edges correspondingly. But by design, they require that again the user program sets up those patterns explicitly. This paper describes the PACT programming model for the Nephele system. The PACT programming model extends the concepts from map/reduce, but is applicable to more complex operations. We discuss methods to compile PACT programs to parallel data flows for the Nephele system, which is a flexible execution engine for parallel data flows (cf. Figure 1).The contributions of this paper are summarized as follows:• We describe a programming model, centered around key/value pairs and Parallelization Contracts (PACTs). The PACTs are second-order functions that define properties on the input and output data of their associated first-order functions (from here on referred to as “user function”, UF). The system utilizes these properties to parallelize the execution of the UF and apply optimization rules. We refer to the type of the second-order function as the Input Contract. The properties of the output data are described by an attached Output Contract.• We provide an initial set of Input Contracts, which define how the input data is organized into subsets that can be processed independently and hence in a data parallel fashion by independent instances of the UF. Map and Reduce are representatives of these contracts, defining, in the case of Map, that the UF processes each key/value pair independently, and, in the case of Reduce, that all key/value pairs with equal key form an inseparable group. We describe additional functions and demonstrate their applicability. • We describe Output Contracts as a means to denote some properties on the UF’s output data.

3. PERFORMANCE EXPERIMENTS

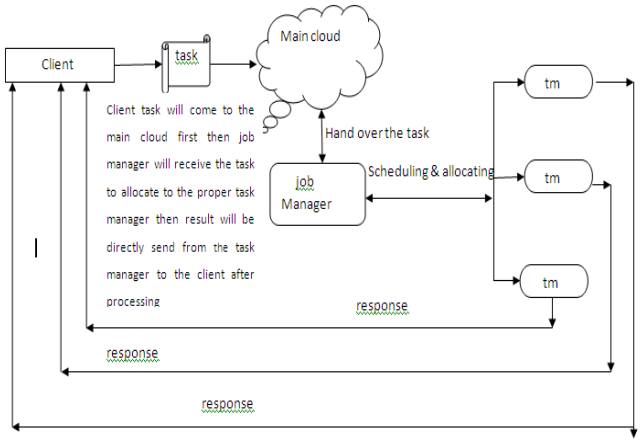

Figure 1. Flow of Dynamic resource Allocation

Figure 2. Dynamic resource Allocation

Figure 3. Technical resource Allocation

4. CONCLUSIONS

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.

5. ACKNOWLEDGMENTS

I would like to thank my HOD Srinivas Murthy and to my beloved guide B Krishna Veni for their continuous support and guidance.

REFERENCES

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

[2] Amazon Web Services LLC, "Amazon Elastic MapReduce," http://aws.amazon.com/elasticmapreduce/, 2009.

[3] Amazon Web Services LLC, "Amazon Simple Storage Service," http://aws.amazon.com/s3/, 2009.

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