Future Generation Computer Systems 28 (2012) 184–192

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adapting scientific computing problems to clouds using MapReduce

Satish Narayana Srirama, Pelle Jakovits *, Eero Vainikko

Distributed Systems Group, Institute of Computer Science, University of Tartu, J. Liivi 2, 50409 Tartu, Estonia

ARTICLE INFO

Article history:
Received 19 January 2011
Received in revised form
10 May 2011
Accepted 28 May 2011
Available online 12 June 2011

Keywords:
Scientific computing
Cloud computing
MapReduce
Hadoop
Iterative algorithm
Twister

ABSTRACT

Cloud computing, with its promise of virtually infinite resources, seems to suit well in solving resource greedy scientific computing problems. To study this, we established a scientific computing cloud (SciCloud) project and environment on our internal clusters. The main goal of the project is to study the scope of establishing private clouds at the universities. With these clouds, students and researchers can efficiently use the already existing resources of university computer networks, in solving computationally intensive scientific, mathematical, and academic problems. However, to be able to run the scientific computing applications on the cloud infrastructure, the applications must be reduced to frameworks that can successfully exploit the cloud resources, like the MapReduce framework. This paper summarizes the challenges associated with reducing iterative algorithms to the MapReduce model. Algorithms used by scientific computing are divided into different classes by how they can be adapted to the MapReduce model; examples from each such class are reduced to the MapReduce model and their performance is measured and analyzed. The study mainly focuses on the Hadoop MapReduce framework but also compares it to an alternative MapReduce framework called Twister, which is specifically designed for iterative algorithms. The analysis shows that Hadoop MapReduce has significant trouble with iterative problems while it suits well for embarrassingly parallel problems, and that Twister can handle iterative problems much more efficiently. This work shows how to adapt algorithms from each class into the MapReduce model, what affects the efficiency and scalability of algorithms in each class and allows us to judge which framework is more efficient for each of them, by mapping the advantages and disadvantages of the two frameworks. This study is of significant importance for scientific computing as it often uses complex iterative methods to solve critical problems and adapting such methods to cloud computing frameworks is not a trivial task.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Scientific computing is a field of study that applies computer science to solve typical scientific problems. It should not be confused with just computer science. Scientific computing is usually associated with large scale computer modeling and simulation and often requires a large amount of computer resources. Cloud computing [1] suits well in solving these scientific computing problems, with its promise of provisioning virtually infinite resources.

In the adaptation of resource-intensive applications for the clouds, the applications must be reduced to frameworks that can successfully exploit the cloud resources, which is the approach we are studying in our Scientific Computing on the Cloud (SciCloud) project [2]. Generally, cloud infrastructure is based on commodity computers, which are cost effective but are bound to fail regularly. This can cause serious problems as the software has to adapt to failures. To deal with hardware or network failures in a distributed system, the best course usually is to replicate important data and retry computations which fail. There are also distributed computing frameworks that provide fault tolerance by design and one such framework is the MapReduce [3] framework.

MapReduce was first developed by Google as a parallel computing framework to perform distributed computing on a large number of commodity computers. Since then, it has gained popularity as a cloud computing framework on which to perform automatically scalable distributed applications. Google MapReduce implementation is proprietary and this has resulted in the development of open source counterparts like Hadoop [4] MapReduce. Hadoop is a Java software framework inspired by Google's MapReduce and Google File System [5] (GFS). The Hadoop project is being actively developed by Apache and is widely used both commercially and for research, and as a result has a large user base and adequate documentation.

While the automatic scalability is very attractive when working with distributed applications, the structure of a MapReduce application is very strict. It is not trivial to reduce complex algorithms to the MapReduce model and there is no guarantee that the resulting

* Corresponding author. Tel.: +372 55638589.
E-mail addresses: srirama@ut.ee (S.N. Srirama), jakovits@ut.ee, jakovits@smail.ee (P. Jakovits), eero@ut.ee (E. Vainikko).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.05.025

MapReduce algorithms are effective. Previous work has shown that MapReduce is well suited for simple, often embarrassingly parallel problems. Google show in their paper [3] that they use MapReduce for a wide variety of problems like large-scale indexing, graph computations, machine learning and extracting specific data from a huge set of indexed web pages. Other related work [6] shows that MapReduce can be successfully used for graph problems, like finding graph components, barycentric clustering, enumerating rectangles and enumerating triangles. MapReduce has also been tested for scientific problems [7]. It performed well for simple problems like the Marsaglia polar method for generating random variables and integer sort.

However, MapReduce has also been shown to have significant problems [7] with more complex algorithms, like conjugate gradient, fast Fourier transform and block tridiagonal linear system solver. Moreover, most of these problems use iterative methods to solve them, indicating that MapReduce may not be well suited for algorithms that have an iterative nature. However, there is more than one type of iterative algorithm. To study if MapReduce model is unsuitable for all iterative algorithms or only a certain subset of them, we devised a set of classes for scientific algorithms. Algorithms are divided between these classes by how difficult it is to adapt them to the MapReduce model and their resulting structure. To be able to compare the classes to each other, we selected and adapted algorithms from each class to the MapReduce model and studied their efficiency and scalability. Such a classification allows us to precisely judge which algorithms are more easily adaptable to the MapReduce model and what kind of effect belonging to a specific class has on the parallel efficiency and scalability of the adapted algorithms.

The rest of the paper is structured as follows. Section 2 briefly introduces the SciCloud project. Section 3 describes the Hadoop MapReduce model on our SciCloud infrastructure and Section 4 describes the different classes for iterative algorithms. Section 5 outlines the algorithms that were implemented and analyzed. Section 6 describes an alternative MapReduce framework called Twister and produces the analysis of the algorithms on the framework. Section 7 mentions the related work and Section 8 concludes the paper and describes the future research directions in the context of the SciCloud project.

2. SciCloud

The main goal of the scientific computing cloud (SciCloud) project [2] is to study the scope of establishing private clouds at universities. With these clouds, students and researchers can efficiently use the already existing resources of university computer networks, in solving computationally intensive scientific, mathematical, and academic problems. Traditionally, such computationally intensive problems were targeted by batch-oriented models of the GRID computing domain. SciCloud tries to achieve this with more interactive and service oriented models of cloud computing that fit a larger class of applications. It targets the development of a framework, including models and methods for establishment, proper selection, state and data management, auto scaling and interoperability of the private clouds. Once such clouds are feasible, they can be used to provide better platforms for collaboration among interested groups of universities and in testing internal pilots, innovations and social networks. SciCloud also focuses on finding new distributed computing algorithms and tries to reduce some of the scientific computing problems to MapReduce algorithm.

While there are several public clouds on the market, Google Apps (examples include Google Mail, Docs, Sites, Calendar etc.), Google App Engine [8] (which provides an elastic platform for Java and Python applications with some limitations) and Amazon EC2 [9] are probably the most known and widely used. Amazon EC2 allows full control over the virtual machine, starting from the operating system. It is possible to select a suitable operating system and platform (32 and 64 bit) from many available Amazon Machine Images (AMI) and several possible virtual machines, which differ in CPU power, memory and disk space. This functionality allows us to freely select suitable technologies for any particular task. In the case of EC2, the price for the service depends on the machine size, its uptime, and the used bandwidth in and out of the cloud.

There are also free implementations of cloud infrastructure e.g. Eucalyptus [10]. Eucalyptus allows the creation of private clouds compatible with Amazon EC2. Thus the cloud computing applications can initially be developed in private clouds and can later be scaled to the public clouds. This is of great help for the research and academic communities, as the initial expenses of experiments can be reduced by a great extent. With this primary goal we have set up SciCloud on a cluster consisting of 8 nodes of SUN FireServer Blade system with 2-core AMD Opteron Processors, using Eucalyptus technology. The cluster was later extended by 2 nodes with double quad-core processors and 32 GB memory per node, plus 4 more nodes with a single quad-core processor and 8 GB of memory each.

While several applications are obvious from such a private cloud setup, we have used it in solving some of our research problems in distributed computing and mobile web services domains [2]. In the mobile web services domain, we scaled our Mobile Enterprise [11] to the loads possible in cellular networks. A Mobile Enterprise can be established in a cellular network by participating Mobile Hosts, which act as web service providers from smart phones, and their clients. Mobile Hosts enable seamless integration of user-specific services to the enterprise, by following web service standards [12], also on the radio link and via resource constrained smart phones. Several applications were developed and demonstrated with the Mobile Host in health care systems, collaborative m-learning, social networks and multimedia services domains [11]. We shifted some of the components and load balancers of Mobile Enterprise to the SciCloud and proved that the Mobile Web Services Mediation Framework [13] and components are horizontally scalable. More details of the analysis are available at [14]. Apart from helping us in our research, SciCloud also has several images supporting in data mining and bio-informatics domains.

3. SciCloud Hadoop framework

With the intent of having a setup for experimenting with MapReduce based applications, we have set up a dynamically configurable SciCloud Hadoop framework. We used the Hadoop cluster to reduce some of the scientific computing problems like CG to MapReduce algorithms. The details are addressed in this section.

MapReduce is a programming model and a distributed computing framework. It was first developed by Google to process very large amounts of raw data that it has to deal with on a daily basis, like indexed Internet documents and web requests logs, which grows every day. Google uses MapReduce to process data across hundreds or thousands of commodity computers. MapReduce applications get a list of key-value pairs as an input and consist of two main methods, Map and Reduce. The Map method processes each key-value pair in the input list separately, and outputs one or more key-value pairs as a result.

$$\text{map}(\text{key}, \text{value}) \Rightarrow [(\text{key}, \text{value})].$$

The Reduce method aggregates the output of the Map method. It gets a key and a list of all values assigned to this key as an input, performs user defined aggregation on it and outputs one or more key-value pairs.

$$\text{reduce}(\text{key}, [\text{value}]) \Rightarrow [(\text{key}, \text{value})].$$

Download English Version:

https://daneshyari.com/en/article/426134

Download Persian Version:

https://daneshyari.com/article/426134

Daneshyari.com