Distributed Flow Control and Intelligent Data Transfer in High Performance Computing Networks

Hochschule Offenburg

Mehdi Sadeghi

13. Feb 2015

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Offenburg, February 28, 2015
Mehdi Sadeghi

Abstract

This document contains my master's thesis report, including the problem definiation, an overview of state of the art, discussions and my suggestions.

Acknowledgement

Here will come acknowledgement

Contents

Preface

European scientific communities launch many scientific experiments daily, resulting in huge amounts of data. Specifically in molecular dynamics and material science fields there are many different simulation software which are being used to accomplish multiscale modelling tasks. These tasks often involve running multiple simulation programs over the existing data or the data which is produced by other simulation software. Depending on the amount of the data and the desired type of simulation these tasks could take many days to finish. The order to run simulation software and providing the input data are normally defined in scripts written by researchers which is the simplest form of workflow management.

While small experiments could be handled with simple scripts and normal computers, larger scale experiments require different solutions. These type of experiments demand huge computing power which are made available by computer clusters and super computers. An important chracteristic of larger experiments is the amount of produced data. This data which needs to be transferred back and forth between computers many times grow by an order of magnitute, resulting in terabytes of data.

Large scientific experiments are the source of many high performance computing (HPC) problems, specially data transfer and workflow management. Moreover HPC resources are expensive and should be used efficiently, therefore making data transfer more efficient is important.

This thesis is an effort to know the main data transfer scenarios in HPC experiments and try to address them in a distributed manner with a collective but decentralized approach toward workflow management.

If there are any comments and improvements regarding this document, the author appreciates an email to the following address:

`msadeghi@stud.hs-offenburg.de`
Hochschule Offenburg
Mehdi Sadeghi

Chapter 1

Introduction

1.1 Objectives

We will cover two important characteristics of HPC at this work. First is data transfer between multiple computers which are doing a task collaboratively. Second the collaboration itself, i.e. how multiple computers will accomplish a collaborative task. To do this we will discuss distributed workflow management. To accomplish these objectives we will first define our requirments clearly regarding the above mentioned topics. We will extract the parameters which we will then assess the other solutions using them. Then we will go through the existing solutions which are currently available. Afterwards we will discuss their applicability according to our defined requirements and whether they answer our needs or not.

1.2 Structure

This document is structured in three main conceptual parts.

Problem In this part we will introduce the problem domain, the requirements and the main problem itself.

Prior Art We will discuss prior art in two parts. First is about distributed data transfer and data access technologies. The second part is about distributed workflow management.

Idea In this part we will focus on our approach to address the problems which have been discussed in previous chapters.

Chapter 2

State of the Art

In this chapter we will go through a number of existing solutions and we will disscuss their efficiency and deployment complexity. Before that we introduce the parameters which are important for us. Then we will assess each solution against the introduced parameter set.

2.1 Parameters

There are many factors that we need to take into account before introducing our parameters. To name a few:

- Data transfer cost
- Data reproduction cost
- Type of operation on data, linear or non-linear

2.2 Data Storage Systems

SAME POINTS SHOULD BE DISCUSSED FOR EACH APPROUACH (FROM OUR POINT OF VIEW) e.g. EFFICIENCY, COMPLEXITY, DISTRIBUTED APPLICATION ACCESS, APPLICATION AWARENESS, POSSIBLE DATA ACCESS SCENARIOS, SCALIBILITY, DATA TRANSFER, FAULT TOLERANCE, ACCESS CONTROL

2.3 Distributed File Systems

One way to achieve fault tolerant and reliable data storage and access is to use distributed file systems (DFS). In this case the data will be replicated

over a network of storage servers with different magnitutes based on the underlying file system. We will discuss a number of free and open source solutions.

2.3.1 Hadoop Distributed File System (HDFS)

"The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware." [5, tp. 3]

"Hadoop1 provides a distributed filesystem and a framework for the analysis and transformation of very large data sets using the MapReduce [2] paradigm."[4]

"HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes."[4]

Deployment Complexity src:http://hadoop.apache.org/docs/r0.18.3/quickstart.html needs Java 1.5.x ssh and sshd and rsync. Three basic modes are available: Local, Pseudo-Distributed and Fully Distributed mode. XML configuration, installation of Local and Pseudo Distributed modes are almost straight forward, for fully distributed note extra steps are required (offical doc link is dead).

Efficiency

Fault Tolerance "Hardware failure is the norm rather than the exception." "Each DataNode sends a Heartbeat message to the NameNode periodically." "The DataNodes in HDFS do not rely on data protection mechanisms such as RAID to make the data durable. Instead, like GFS, the file content is replicated on multiple DataNodes for reliability." [4]

Portability "HDFS has been designed to be easily portable from one platform to another."

Robustness "The primary objective of HDFS is to store data reliably even in the presence of failures."

Accessibility

1. FS Shell
2. DFSAdmin
3. Browser

Applicability There is a good document here: http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf Hints: HADOOP is for big data and the programming should be different (map/reduce) and it does not look suitable for our use cases and requirements. The burden would be so high that we will waste a lot of resources. I have to put these in scientific words with more logic and references to sizes that we need and more numbers.

Users have to program their applications using Java and Hadoop to take advantage of distributed computing features in Hadoop MapReduce and HDFS. Cites? Hadoop website? https://infosys.uni-saarland.de/publications/BigDataTutorial.pdf

2.3.2 XTREEMFS

Chapter 3

Rough Ideas

This chapter contains very raw ideas to address main requirements i.e. distributed workflow management and intelligent data transfer.

3.1 Intelligent Data Transfer: Use Case One

By *Intelligent Data Transfer* we mean an approach that minimizes required data transfers between application[1] instances.

In the most basic use case[2] we run a script [3] which consists of two linear operations. Each operation consumes data and generates data. A third operation needs both generated data two operate on and generate the third and final data.

The script is data driven. It means that it contains a number of steps and for every step it needs appropriate data to run the desired operations[4]. We assume that the script will run on *Node 1* and required data *DataSet1* and *DataSet2* are located on *Node 2* and *Node 3* respectively. Therefor *Node 1* have to initiate operations on the other machines.

3.1.1 Identical Instances

We assume that on each machine of the network the same instance of our imaginary program is running which is capable of running all operations including A, B and C. The only consideration is the availability of DatSets, they are not available on all machines.

[1] To be defined
[2] To be added later and referenced here
[3] To be defined and added to the terminology, terminology itself has to be defined
[4] To be defined

A linear operation (not clear to my self how to write it):

$$Operation(A, B) = Operation(A) + Operation(B)$$

$$DataSet^A = Operation^A(DataSet^1)$$

$$DataSet^B = Operation^B(DataSet^2)$$

$$DataSet^C = Operation^C(DataSet^A, DataSet^B)$$

Assuming that operations A and B will run on the machines which contain the required data, a number of questions arise here:

1. On which machine operations C should run? A, B or C?
2. On How to transfer the required data to that machine in an optimized way?

3.1.2 The Idea

First of all we assume that we have the information about the DataSets available on all of the mamchines i.e. in form of a distributed table with entries containing the node address and DataSet id. Based on this information the application can decide if it has the required data or not.

Based on this algorithm (to be defined) the initial application delegates operations to the other nodes (instances of the same program), where the data is available. Our distributed workflow manager (to be defined) will synchronize the information on these running operation and will lable the output data and will add it to the distributed data table.

After finishing operations A and B we will run operaion C in either of these nodes, because the required data is partially available on these nodes. Then we have to transfer the rest of the data to one of these nodes to run the operation C which needs both parts simultaniously.

Using Prior Art At this point we can take advantage of existing Distributed File Systems (DFS) to make the data available for operation C. We can then eliminate the complexity of data transfer between these two nodes and delegate it to existing distributed file systems. The main point is we don't rely on DFS for all of our decistion making part but we explicitely make the decition to run operation A and B on specific nodes and then for the last part we use a meta disk or universal disk concept to deliver the remaining data for operation C.

References

[1] URL: http://www.plyojump.com/classes/mainframe_era.html.

[2] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. Proc. Sixth Symposium on Operating System Design and Implementation, 2004.

[3] *History of Computers*. 2008. URL: http://www.youtube.com/watch?v=LvKxJ3bQRKE.

[4] *The Hadoop Distributed File System*. URL: http://www.aosabook.org/en/hdfs.html.

[5] *The Hadoop Distributed File System: Architecture and Design*. 2007. URL: http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf.