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

Mehdi Sadeghi

Supervisors:
Prof. Dr. Katharina Mehner-Heindl
Dr. Adham Hashibon

MASTER'S THESIS

On
Master's Degree Program
Communication and Media Engineering
in Offenburg

March 2015

Declaration of Authorship

I declare in lieu of an oath that the Master Thesis submitted has been produced by me without illegal help from other persons. I state that all passages which have been taken out of publications of all means or un-published material either whole or in part, in words or ideas, have been marked as quotations in the relevant passage. I also confirm that the quotes included show the extend of the original quotes and are marked as such. I know that a false declaration will have legal consequences.

February 28, 2015

Mehdi Sadeghi

Abstract

This document contains my master's thesis report, including the problem definition, an overview of state of the art, discussions and proposed solution. During this work we have proposed a solution to run various types of operations in a network collaboratively without any permanent broker or orchestrator. We have defined and analysed a number of scenarios according to our requirements and we have implemented the solution to address those scenarios using a distributed workflow management and data transfer approach. During this report we first introduce the problem and its context and then we go through existing solutions, our workflow and data transfer methods, application architecture and discussing different aspects of our approach.

Contents

1 Introduction **5**
- 1.1 Thesis Context . . . 5
- 1.2 Thesis Objectives . . . 5
- 1.3 Terminology . . . 6
- 1.4 Problem Context . . . 6
- 1.5 Assumptions . . . 7
  - 1.5.1 Collaborating Network . . . 7
  - 1.5.2 Data Characteristics . . . 7
  - 1.5.3 Data Transfer . . . 7
  - 1.5.4 Workflow . . . 7

2 Related Work **9**
- 2.1 Parameters . . . 9
- 2.2 Data Storage Systems . . . 9
- 2.3 Distributed File Systems . . . 10
  - 2.3.1 Hadoop Distributed File System (HDFS) . . . 10
  - 2.3.2 XTREEMFS . . . 11
- 2.4 Distributed Objects . . . 11
  - 2.4.1 Concoord . . . 11
  - 2.4.2 Distributed Hash Tables (DHT) . . . 11
- 2.5 Distributed Workflows . . . 11

4 Proposed Architecture **13**
- 4.1 Internals . . . 13
- 4.2 Network View . . . 13

4 Proposed Architecture **13**
- 4.1 Internals . . . 13
- 4.2 Network View . . . 13

5 Discussions **14**
- 5.1 Possible Issues . . . 14
  - 5.1.1 High Load . . . 14
  - 5.1.2 Orphan Operations . . . 14
  - 5.1.3 Complexity Growth . . . 14
- 5.2 Future Work . . . 14
- 5.3 Network Discovery . . . 14

5.4 Bootstrapping 14
5.5 Data Popularity 15
5.6 Security 15
5.7 Fault Tolerance 15

Preface

European scientific communities launch many experiments everyday, resulting in huge amounts of data. Specifically in molecular dynamics and material science fields there are many different simulation softwares which are being used to accomplish multi-scale modeling tasks. These tasks often involve running multiple simulation programs over the existing datasets or the data which is produced by other simulation software. It's common to run multiple programs on existing datasets during one operation to produce the desired results. The order to run simulation software is normally defined within scripts written by users.

Moreovers users have to provide the required data manually and copy all required files to a working directory to submit their job, and they might have to login to different machines to prepare files, submit the script, monitor the status of the job and finally collect the output files. This type of work routine is a common form of workflow management in above mentioned communities.

While simpler and smaller experiments could be handled this way, larger and more complicated experiments require different solutions.

Such experiments are the source of many high performance computing (HPC) problems, specially workflow management and data transfer. This thesis is an effort accomplish such operations in a distributed manner with a collective but decentralized approach toward workflow management and to minimized data transfer during such operations.

This thesis is an indirect research and development effort in the field of high performance computing (HPC) and scientific simulations. This has not been an implementation task nor a purely theoratical work. It means that I was not supposed to create an application or develop a software from ground up (even though eventually I did), instead I have been responsible to study about and define the problem of my client and assist them either with finding a suitable solution and helping them to integrate it into their development process or propose a new approach to address their needs. My activities include but not limited to analysing the problem, collecting requirements, studying state of the art software frameworks and related products, analysing them against the defined requirments, proposing a solution and developing a prototype.

During this thesis an open source prototype application has been developed which is available online[1]. The source files of the current document are also available[2]. If there are any comments and improvements regarding this document, I appreciate an email to **sadeghi@mehdix.org**.

[1] https://github.com/mehdisadeghi/konsensus
[2] https://github.com/mehdisadeghi/cme-thesis

Chapter 1

Introduction

1.1 Thesis Context

This thesis is an indirect research and development effort in the field of high performance computing (HPC) and scientific simulations. This has not been an implementation task nor a purely theoratical work. It means that I was not supposed to create an application or develop a software from ground up (even though eventually I did), instead I have been responsible to study about and define the problem of my client and assist them either with finding a suitable solution and helping them to integrate it into their development process or propose a new approach to address their needs.

My activities include but not limited to analysing the problem, collecting requirements, studying state of the art software frameworks and related products, analysing them against the defined requirments, proposing a solution and developing a prototype.

1.2 Thesis Objectives

There are two main objectives in these thesis as the title suggests. First is to distribute the workflow and eliminate central brokers. Second is minimizing the amount of transferred data in the network.

Both of these objectives are tailored toward the context that this work is done. There are existing workflow management tools and data transfer solutions out there but this work is

This data belongs to operations which might be linear or non-linear and might involve other operations as well. Each operation can be initiated in any participating peer but the required data is not necessarily available on that computer even though the operation result might be delivered back to the same peer. We will focus on two important topics in this work. First one is data transfer problems between multiple computers which are doing a task collaboratively. Second one is the collaboration itself, i.e. how multiple computers will manage to finish the task in a distributed and decentralized environment. To accomplish these objectives we will discuss our specific distributed workflow management and data transfer methods. We define our requirements regarding the above mentioned topics and we will extract the

parameters which we are going to assess other solutions with them. Then we will go through the currently available solutions and we will discuss them shortly to see whether they are applicable to our problem domain with regard to our requirements.

1.3 Terminology

We will use a number of terms through this report. Here are the meaning for each.

Node Refers to one computer in the network.

Data When we refer to data we mean the output of scientific applications.

Dataset Same as data with more emphasize on it as collection e.g. NumPy array.

Application The demo software which has been developed to show case the proposed solution.

Peer One instance of the network application which is in collaboration with others.

Instance An instance of the application running on a node.

Operation Linear or non-linear functions which users want to run on datasets.

Task Same as the operation with more emphasize on the output rather than the functionality.

Service A scientific operation being provided by the application which could be called remotely.

System The combination of nodes, data, application, instances, operations and services as a whole.

User A scientist, researcher or student who uses the system to run a task.

1.4 Problem Context

Whole this work is an effort to address issues of a scientific environment. Some particular characteristics are running multiple scientific programs on different computers which need to exchange data in order to accomplish one operation. Another task which is often done is visualization. Visualizing the operation results ,depending on the requested visualization, might require heavy computational tasks i.e. average

or comparison on data which might not be available on the same machine or might be residing partially on different computers. The produced data often exceeds 1 GB in many experiments and it should be moved back and forth every few minutes, therefore it is cheaper to transfer the operation rather than the data.

The problem here is not about distributing the stored data rather exchanging it between instances of the application talking together in runtime while doing one global task and keeping this workflow distributed. In this terms each application instance takes care of its own data and provides a set of services. Some operations require data from another node, therefore we have to transfer the data or run the operation on the node which contains the data. There are a number of scenarios which we will discuss.

1.5 Assumptions

During this work we have a number of assumptions. We have a certain problem which we want to focus on rather than reintroducing solutions that already exist. For this reason we discuss regarding our needs.

1.5.1 Collaborating Network

We assume there is a network of computers which are available to run the tasks, each node is running an instance of the application. We will propose our collaboration and data transfer algorithm between them later.

1.5.2 Data Characteristics

We need to discuss more about the data. In our scientific context data is mostly numerical and explains characteristics of physical particles such as atoms and molecules. These data is being used to simulate collections of particles called models. Although our work is not dependent on these, they help us to understand the definition of the data that we often refer to in this report. One important aspect of the data that we are interested in is that it is not critical and we can reproduce it.

1.5.3 Data Transfer

We assume a data transfer approach is already in place. This could be any file system which supports network storage. Rather than going into details of how data could be transferred more efficiently, we will focus on finding which data to be transferred and from which computer to which destination.

1.5.4 Workflow

In contrast to data we are interested in workflow. We want to find a reliable approach to access and update state of our workflow on any arbitrary node which is part of our collaborative network.

Approaches

During this work we consider three different approaches toward preparing required data for operations.

Conventional Approach in this approach we put the required data on a network file system and all application instances will access it there. We will utilize an NFS mounted file system.

Centralized Approach in this approach we will have a central instance which will orchestrate operation delegation and operation output forwarding to other nodes.

Decentralized Approach in this approach we will eliminate the orchestrator node and the network of application instances should collaborate in a decentralized fashion to keep track of data and control flow for each task.

For every approach we will run performance tests and we will compare the results.

Method

We will discuss scenarios in **??**. For each scenario we will analyze the possible combinations of data and operations and we will discuss how to deliver the input data and where to store output data. We will discuss workflow management in chapter **??** and data transfer in **??**.

Chapter 2

Related Work

In this chapter we will go through a number of existing solutions and we will discuss 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
- Replication
- Deployment complexity
- Fault tolerance
- Portability
- Performance according to number of distributed nodes and size of files

2.2 Data Storage Systems

SAME POINTS SHOULD BE DISCUSSED FOR EACH APPROACH (FROM OUR POINT OF VIEW) e.g. EFFICIENCY, COMPLEXITY, DISTRIBUTED APPLICATION ACCESS, APPLICATION AWARENESS, POSSIBLE DATA ACCESS SCENARIOS, SCALABILITY, 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 magnitudes 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." [**HDFSDocuments**]

"Hadoop1 provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce [**DG04**] paradigm."[**TheHDFS**]

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

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 (official 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." [**TheHDFS**]

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

2.4 Distributed Objects

In this section we go through a number of existing methods to distributed an object or in another terms to distribute the state.

2.4.1 Concoord

Describe why it is not suitable for us. It allows single object sharing.

2.4.2 Distributed Hash Tables (DHT)

DHTs are known to be a distributed key/value storage.

Kademlia

Kademlia is a p2p DHT algorithm introduced in 2002.

2.5 Distributed Workflows

In this section we introduce a number of existing scientific workflow systems.

Chapter 3

Proposed Architecture

3.1 Internals

3.2 Network View

Chapter 4

Proposed Architecture

4.1 Internals

4.2 Network View

Chapter 5

Discussions

5.1 Possible Issues

5.1.1 High Load

5.1.2 Orphan Operations

5.1.3 Complexity Growth

5.2 Future Work

During this work we have focused on the aspects of the problem which were important in the context domain and we left aside many other small and big problems without considering them during this project. The main reason was that we wanted to work on problems which were new and genuine because for other aspects there are already many well-defined solutions available, so we did not spend our time for them. Moreover one should consider that this project is not solely an implementation rather a research on finding ways to embed distributed solutions into other projects.

In the following sections we talk shortly about the topics which we have not covered but this work can be extended to include them as well.

5.3 Network Discovery

Currently the peers are configured in the beginning and there is no dynamic peer recognition. This might be done in a number of ways such as sending broadcasts or using third party projects such as Zyre [**Zyre**].

5.4 Bootstrapping

With having address of only one peer we would be able to configure and a new peer and join the network. There should be a mechanism among peers to identify joining and leaving peers. But our context is different than a peer-to-peer applications

which peers join and leave frequently. In our case most of peers run a long time and bootstrapping is more a way to get the state of currently running workflows and let others know about the new peer.

5.5 Data Popularity

There are algorithms developed to calculate data popularity over time and then replicate them over peers for easier access. If we want to move toward any type of data replication we would need to use this algorithms.

5.6 Security

There is no user management and secure communication in our initial requirements however this would be required if we want to manage user rights or introduce limitations or simply to keep a history of activities for each user. Moreover to secure inter-peer communications we might use X.509 certificates. Further more since we've used ZeroMQ[**ZeroMQ**] as underlying transport channel we can use its more advanced security features such as Elliptic curve cryptography[**Curve**] based on Curve25519[**Curve25519**] to add perfect forward secrecy, arbitrary authentication backends and so on.

5.7 Fault Tolerance

In current work there is no failure recovery mechanism, since it was not part of the requirements. In case of a failure or exception in any collaborating peer not only the failed instance should be able to recover itself into a correct state, moreover the other peers should maintain a valid state for on-going distributed workflows and keep their internal state up-to-date.