Hochschule Offenburg
University of Applied Sciences

Masterarbeitspräsentation

Fraunhofer IWM

Distributed Flow Control and Intelligent Data Transfer in High Performance Networks

Masterstudium:
Communication and Media Engineering

Mehdi Sadeghi

Hochschule für Technik, Wirtschaft und Medien Offenburg
Fakultät Medien und Informationswesen
Professorin: Dr. Katharina Mehner-Heindl
Betreuer: Dr. Adham Hashibon

Introduction

Scientific experiments are the source of many high performance computing (HPC) problems. Here are some highlights:

- Multiple simulation programs run during one operation
- Huge data is being used by these programs
- Data is being produced and used on the fly
- Simplest form of workflow management is writing scripts
- Users have to manage input/output files manually
- Job schedulers have control over execution process

This thesis is an effort to:

- Accomplish operations on a distributed network collectively
- Make users neglectful of data location
- Propose a decentralized approach toward workflow management
- Avoid unnecessary data transfer

Requirements

The main points according to our context are these:

- **Data location is unknown** to users
- User should have same experience regardless of the peer to which she connects.
- System should have **no central brokers**.
- Required data are distributed on the network.
- **Runtime control** over task execution.
- The solution should be **distributed**.
- Easily deployable.
- **User space** solution
- Light weight

Related Work

There are similar solutions, some very localized and some too big for small groups to handle:

- UNICORE: a grid computing technology for resources such as supercomputers or cluster systems and information stored in databases.
- Distributed File Systems, centralized or decentralized
- Porto: The Porto platform will provide a lightweight and flexible system for data and workflow management.

Main Scenarios

We categorize every possible operation into two main groups:

1. **Linear**. These operations could be run in parallel. The algebraic notation would be:
$$\boldsymbol{Operation}(\boldsymbol{A}+\boldsymbol{B}) = \boldsymbol{Operation}(\boldsymbol{A}) + \boldsymbol{Operation}(\boldsymbol{B})$$
2. **Non-linear**. These operations are different, they can't be run in parallel:
$$\boldsymbol{Operation}(\boldsymbol{A}+\boldsymbol{B}) \neq \boldsymbol{Operation}(\boldsymbol{A}) + \boldsymbol{Operation}(\boldsymbol{B})$$

Proposed Solution

We need only to solve two problems:

1. How to solve an operation on one dataset?
2. How to solve a non-linear operation on two datasets?

We break down all others into combinations of these two scenarios.

We propose to make a peer-to-peer collaborative application which will run on every resource (computer) in our network, this computer (which we call it peer) will then become one part of the network. In the beginning there is a network with any given number of peers, new ones can join and current ones can leave the network. Here are the main characteristics:

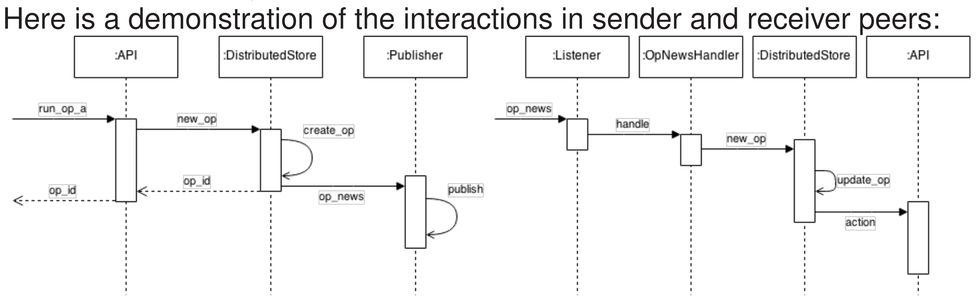
- The solution is **peer-to-peer**
- **Not a client/server** notion
- Based on **publish-subscribe** pattern
- Operations are **asynchronous**
- API calls return a **unique operation ID**
- Distributed operation store will be synchronized with peers
- **Peers subscribe to each other**
- Every **pear publishes news** on every **state changes**
- Peer can launch operations **recursively** on itself or others
- Peers keep state for **Operations**, **Datasets** and other **peer list**
- Peers and application components are **loosely coupled**

In our design the simplest operation is not dependent on any other operation, and we know how to handle it. Other operations are made from these atomic ones.

A Sample Scenario

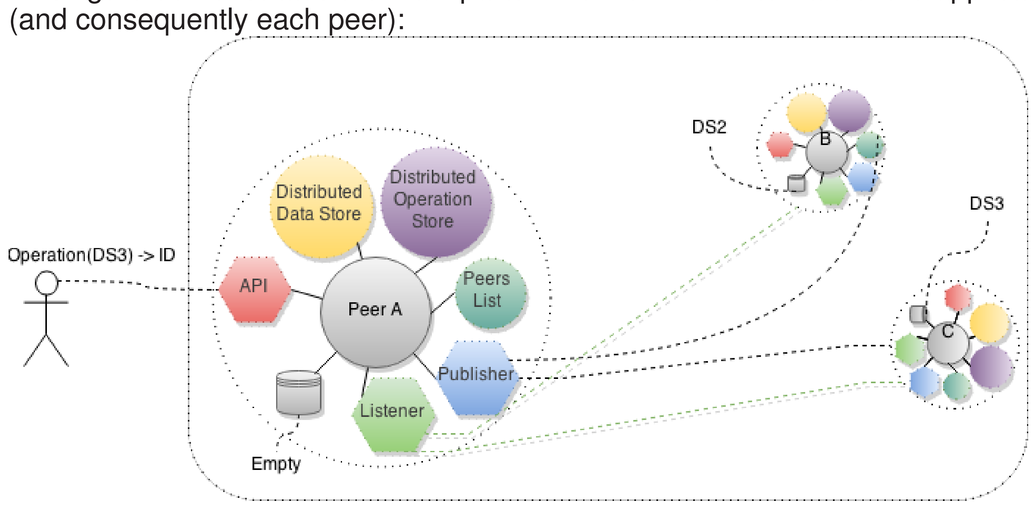
1. User sends $\boldsymbol{Op}(\boldsymbol{A}+\boldsymbol{B})$ to a **random** peer and gets an ID
2. The peer stores this operation in a **distributed store**
3. System breaks this to $\boldsymbol{Op}(\boldsymbol{A}) + \boldsymbol{Op}(\boldsymbol{B})$ and **self-launches** them **asynchronously**.
4. Upon completion of sub-operations a random peer will store the result on the network and notify others

Here is a demonstration of the interactions in sender and receiver peers:



Internal Components

This figure demonstrates the main parts which form the internals of the application (and consequently each peer):



The above illustration shows:

- A network with three peers, labeled left to right as A, B and C respectively
- Publish/subscribe links between the peer in the left with the other two
- Local data store of each peer
- State objects of the peers including
  - DistributedDataStore
  - DistributedOperationStore
  - Peers list
- A basic submission scenario
- API, Publisher and Subscriber components

Conclusion

Even though there are many solutions designed for HPC problems, still there are requirements for smaller groups which are not satisfied, such as:

- Making scientific applications user friendly
- Providing *smarter* solutions which get out of users way, i.e. hiding the systems complexity from ordinary users
- The system manages data endpoints, not users
- Less deployment and maintenance cost
- More flexibility to control application at runtime

During this work we addressed some of these needs:

- The problem was defined and requirements where defined
- We went through the state of the art
- A solution approach was proposed
- A prototype was developed
  - Based on open technologies
  - Runs in user space
  - Open source and freely available on Github

Our approach is very flexible to be extended and it is easy to build new services on top of the existing framework which provides the distributed operation and storage mechanisms to applications.

References

- UNICORE article on Wikipedia http://en.wikipedia.org/wiki/UNICORE
- The Porto platform https://github.com/NanoSim/Porto
- The Konsensus project https://github.com/mehdisadeghi/konsensus

Kontakt: msadeghi@stud.hs-offenburg.de