Using architectural simulation models to aid the design of data intensive application

Javier Fernández¹, Liangxiu Han², Alberto Nuñez¹, Jesus Carretero¹, Jano van Hemert²

¹Architecture Group, Computer Science Department
Universidad Carlos III de Madrid, Leganés. Spain
{jfmunoz, anunez, jcarrete}@inf.uc3m.es
²National eScience Center, School of Informatics, University of Edinburgh
{liangxiu.han, j.vanhemert}@ed.ac.uk

Abstract—Performance is an open issue in data intensive applications. Finding the best implementation and influential performance factors of hardware and software platforms for the data intensive applications requires trial and error. However, it is very difficult and costly to perform these trials in a real large-scale environment. In this paper we use a generic simulation framework SIMCAN to simulate hardware and software platforms of data intensive applications for investigating the influential performance factors, and thereby making decisions on the design of data intensive application architectures. We have employed a typical use case of a data mining application, in which the architecture has been proposed using a pipeline model. We have simulated various scenarios to investigate factors that affect the system performance to assist the architecture design and the simulation results provide useful information for this decision-making.

Keywords: Simulation framework; Computing architecture; Storage management; Data intensive application

I. INTRODUCTION

A high performance data intensive application requires a good design and implementation of its hardware and software platforms, such as increasing computing power, storage, reducing network communication overhead and moving computation and data close to each other. To obtain and optimize influential factors, it is necessary to perform various experiments that contain various scenarios as many as possible. However, it is costly and difficult to do this in a real large-scale environment. The alternative way is to use a simulation framework to model the different hardware components and the different application models in order to execute as many combinations as possible and find the best match.

Data mining and data integration applications are typical examples of data intensive applications. One proposal here to parallelize this kind of applications is to model the task as a directed graph of processing elements that can be assembled and composed together in a pipeline manner with data streaming between them [10]. This philosophy leads to a simple scalable architecture design. However, this scalability has to be checked for all of possible designs using simulation techniques before a realization of the architecture.

To date, various simulators at different levels have been developed. At the architecture level, Parsec [1] has provided a generic framework for sequential and parallel execution. OMNET++ [2] was originally developed for the simulation of communication networks. However, because of its generic and flexible architecture, it is also used in simulations at the architecture level. At the component level, NS [3] and INET [4] are especially for the simulation of the network and DiskSim [5] for the disk. Moreover, SIMLAB [6] provides a scalable and resource efficient storage network and MPI-SIM [7] for MPI applications.

For this purpose, we developed a generic Simulator Framework for Computer Architectures and Storage (SIMCAN [8]). The SIMCAN framework is developed as a modular and flexible framework to simulate various architectures and applications. By using the basic modules such as computing, storage and network components, we can construct simulated hardware and software platforms to model the proposed architecture and to investigate the factors that affect performance.

The rest of paper is structured as follows: Section 2 presents the reason to use simulations models, outlines the architecture of the SIMCAN and briefly a use case of data mining. In Section 3, the experimental setup and performance analysis results have been described in detail. Section 4 presents the conclusions and future works.

II. USING SIMULATION MODELS TO HELP DESIGN DATA INTENSIVE APPLICATIONS

In this paper we present a use case of data mining and integration, in which the architecture has been proposed to use a pipeline schema. This schema is flexible enough to make several different implementations. Considering factors such as computing power, data access and storage, and network that influence the performance of systems, it is necessary to make a decision to choose the best implementation. Generally, there are two ways for this decision-making. One common way is through a trial and error process with multiple possible combinations. However, executing these trials with real hardware and software is hard and costly and cannot cover all the possibilities. The other way is to use simulations for supporting the decision-making.

In this work, we adopt a flexible and configurable simulation framework SIMCAN to help the decision-making on the architecture design of the data mining application.
The SIMCAN framework is composed of a set of modules. All those modules were developed using OMNeT++. A SIMCAN module (like in OMNeT++) is a building block. The highest level in the SIMCAN architecture consists of nodes and network components to communicate them (like routers or switches). A node in SIMCAN can act like a computing node or a storage node. Computing nodes contain applications that perform I/O requests (these kind of nodes can contain, or not, local storage devices).

All the logic of the modules that simulate the computing hardware and software (CPU, disks, caches, schedulers, etc) have been programmed from scratch (using OMNeT++ facilities), or are obtained from others external simulations modules (like DISKSIM).

Storage nodes contain a complete I/O storage subsystem, with its corresponding storage devices and the corresponding server application, like a NFS server.

SIMCAN follows a modular and hierarchical architecture. The most important features of the proposed framework are its flexibility and scalability.

A real storage network contains several components that must be simulated, like Disk Drives, File Systems, Volume Managers, Schedulers, Caches, Communication Networks, etc. Due to the effect of File Systems on the behavior of real systems, a modelling of a File System was built up [9].

Figure 1 shows the basic node schema of a generic SIMCAN framework. A computing node contains applications that generate I/O requests. Those requests can be local or remote. Another node type is the storage node. This kind of node must contain an I/O subsystem. Generally, this node serves I/O requests from computing nodes. This node contains server applications, like NFS servers, FTP servers, etc.

The case of data intensive applications used here [11] is a typical image pattern recognition task. The image data is come from [12] and have 4 Terabytes in total. The recognition goal is to automatically identify an anatomical component (e.g., ear) in the image stored in file systems and label the component using a term from developmental ontology of mouse embryos stored in databases.

This image pattern recognition task mainly involves in three stages: training, test and evaluation, and deployment stages, as shown in Figure 2. The training stage is to train the annotated data and build up classifiers. The processes include image integration, image processing, feature generation, feature selection and extraction, and classifier design. The specific processes are described as follows:

- **Image integration**: before starting the data mining, we need to integrate data from different sources: the manual annotations have been stored in the database and the images are located in the file system. The outputs of this process are images with annotations.
- **The size of the images is variable and there are noises in the images. We use image scaling and image filtering method to rescale and denoise the images. The outputs of this process are standardized and denoised images, which can be represented as two-dimensional arrays.**
- **After image pre-processing, we generate those features that represent different gene expression patterns in images. The resulting features of wavelet transform are 2 dimensional arrays.**
- **Due to the large number of features, the features need to be reduced and selected for building a classifier. Either feature selection or feature extraction or both can do this. Feature selection selects a subset of the most significant features for constructing classifiers. Feature extraction performs the transformation on the original features for the dimensionality reduction to obtain a representative feature vectors for building up classifiers.**
• The main task in this case is to classify images into the right gene terminologies. The classifier needs to take an image's features as an input and for each of anatomical features outputs a ‘yes’ or ‘no’.

The test and evaluation stage will use the result from the training stage to test images to measure the performance of classifiers. The processes in the test stage have the same processes except for the classifier design, which only appears in the training stage.

The deployment stage is to deploy the classifiers to perform the classification of un-annotated image and deliver results to the users.

We specify this image pattern recognition task as a directed graph of process elements and pipeline the processes of training stage and test stage and execute these processes in parallel for obtaining high performance.

III. SIMULATION SCENARIOS AND DESIGN LEADS FROM THE RESULTS

The use case of this image pattern recognition task need take into account factors on computing capability, data access and storage as well as network communication, for example, the images to be processed are big and the method used is computational. Some of data are stored and shared on the disk, while some of others are streamed between each process element. Thus, we have developed and built up a set of scenarios with different configurations for each one of the main possible influential factors in order to measure the performance using the proposed pipeline model for this data intensive application and provide useful information for the decision-making.

A. Experimental setup

By using the SIMCAN, we have built up a hardware platform that includes computing nodes, storage nodes and network bandwidth. The configuration for the hardware is as follows:

a) Computing nodes: In this case, the computing time for each process element is always calculated on a 2.2 GHz Intel core 2. The number of CPUs is increased by incrementing the number of computing nodes (1 node, 5 nodes and 10 nodes).

b) Storage nodes: Each storage node is a complete NAS (Network-Attached-Storage) system with a simulated version of the Seagate ST3400620A disk (400 GB). The numbers of storage nodes vary from 1, 5 to 10 storage nodes. The files are distributed along the nodes using a round robin distribution.

c) Network bandwidth: We use a switched Ethernet network with different bandwidths (10 Mb/s and 1000 Mb/s).

We have also simulated the data mining application. Due to the limitation of space, we only simulate the test stage of this data mining case. The pipeline model for the test stage consists of 5 process elements with different functionalities. Each process is modularized and the computing time for each process is listed in Table 1, which is obtained from the execution of the prototype on real machines:

![Table 1. Computing time for the pipeline processes](image)

Since the pipeline model supports task execution in parallel, considering computing, storage and network, we have divided image data into subsets and investigated performance under different scenarios as follows:

• Run all of processes of a task (5 processes) with subsets image data on each single computing node in a pipeline way. We call it as a complete pipeline. An example experiment setup is shown in Figure 3.

• Run each process of a task with the whole image dataset on different computing nodes in a pipeline manner. We call it as a one-distributed pipeline. An example experiment setup is shown in Figure 4.

• Run each process of a task with a subset of the image data on the different computing nodes and with several same processes deployed on the same node in a pipeline way. We call it as a several-distributed pipeline. An example experiment setup is shown in Figure 5.

Each scenario above has been performed under two situations below:

• All the data transferred between processes is stored in disks when possible, which is referred to as disk transference that means to write the data into disks and send the reference only.

• All the data transferred between processes is streamed when possible.

B. Performance analysis and results

We have used the speedup as a performance indicator. The speedup is considered as a ratio between the execution time of the image pattern recognition task (Ts) (In this case, a task means either one process of a pipeline or the whole pipeline) on one single processor and the execution time of the task on multiple processors (Tm), represented as

$$S = \frac{T_m}{T_s}$$

The execution on one single processor means all of processes, data, and storage on one computing node. The execution on multiprocessors includes any of these situations: distributed data, distributed processes and distributed storage.
Figure 6, 7, 8 and 9 present the results (speedup) for a complete pipeline, one distributed pipeline and several distributed pipelines under various scenarios: 1) one computing node runs all processes of a task, data and storage at one computing node (no parallelization; 2) 5 computing nodes, with 1, 5 and 10 storage nodes respectively; 3) 10 computing nodes, with 1, 5 and 10 storage nodes respectively.

There are three observations from the result diagrams. The first observation is the parallel execution is obviously better than execution of all processes on one single computing node. The second observation is that the performance increases with the increase of computing nodes. The numbers of storage nodes have a considerable influence when using the disk transfer (in Figure 6 and 8) instead of using the net-based transfer (in Figure 7 and 9). Finally, the performance of the complete pipeline model always has the best performance than other pipeline models, such as several distributed and one distributed models.

Since the complete pipeline model in all of four cases always outperforms and the number of storage nodes has no significant influence on the net-based transference, therefore, to investigate this reason behind, we have set up new experiments by running each process of a task on different computing nodes with several different processes of parallel tasks deployed on the same computing node as shown in Figure 10. We call it a mixed pipeline. The purpose is to investigate whether CPU time is an influential factor when having the same or different processes in one node. We have performed our experiments by changing CPU time and the deployment of processes under the following conditions:

- We use the real value of each process from the table 1.
- We set up all of processes with the equal CPU time (we use 0.04 seconds because the total time of the pipeline gets similar to the previous case).
- We increase the CPU time by 100 on both two cases above (to check what happens when the bottleneck is in the CPU).

The result in Figure 11 shows the mixed pipeline has almost the same performance as the complete pipeline. Moreover, when using the same computing times for all of processes (similar to the real ones) the performance doesn’t change. This suggests the CPU time is not a bottleneck but the way to arrange processes of the pipeline does matter the performance.

To further investigate this, we select a big value for computing time (to ensure that the CPU is the bottleneck) equivalent to the total time of all of processes, the performance is the same for all models. This also suggests that the way processes are arranged is related to the performance (because each process of a task has different functionalities with different processing time). Furthermore, the experimental result shows the best performance can be obtained by arranging processes of a task on the same computing node.
Figure 6. Speedup for the scenario with network of 10 Mb and disk-based transferences

Figure 7. Speedup for scenarios with network of 10 Mb and net-based transferences

Figure 8. Speedup for scenarios with network of 1000 Mb and disk-based transferences

Figure 9. Speedup for scenarios with network of 1000 Mb and net-based transferences

Figure 10. Each process distributed on different computing nodes with several different processes on the same node (a mixed pipeline)

Figure 11. Speedup for scenarios with 5 computing nodes, 1 storage node and net-based transferences

IV. CONCLUSION AND FUTURE WORK

In this paper, by simulating hardware and software platforms of a data intensive application using a generic simulation framework - SIMCAN, we have studied the influential performance factors, which can provide useful suggestions for the decision-making on the architecture.
design of this use case (a typical image pattern recognition task that has been proposed using a pipeline model).

The simulation results suggest that the number of computing nodes is more important than that of storage nodes when using the net-based transfers. If the system uses the disk-based transfers, then both, the numbers of computing and storage nodes must be increased accordingly. The simulation results also show that the arrangement of processes of a task has an influence on the performance. In future works we will test different and more complex uses of case also with different architectures and also focus on obtaining a general strategy to employ simulation models to aid the design of data intensive applications.

ACKNOWLEDGMENT

This work was carried out under the HPC-EUROPA2 project (project number: 228398), with the support of the European Community - Research Infrastructure Action of the FP7.

REFERENCES


