Parallelising Kiwiplan Framework

Prabhjot Singh Jassal
ID: 3666860
Email: pjas002@aucklanduni.ac.nz

ABSTRACT

Nowadays most of the computers are coming with multi cores. However, if the programmer wants to make their program efficient by making use of different cores, then they need to program in a different manner. Most of the computation intensive tasks are being done underlying Kiwiplan Framwork. Each task is separated into filters and each filter algorithm runs sequentially. Kiwiplan interested in investigating how much efficiency could be increased by making the filters algorithm parallel, wherever possible.

I. INTRODUCTION

From the last two decades or so enormous research has been made to solve the computational intensive problems more efficiently. To do this computers have evolved a lot. The clock speed of the computer is the frequency of a processor to execute instructions. Initially, in 1970's computers with a clock speed of 200 MHz started to come. The first sudden jump came in 2000 when Pentium introduced Intel 4 processor computer with an initial clock speed of 1.5GHz [5]. From then on there has been gradual increase in the clock speed, which allows programmers to solve problems more efficiently. However, there is a limit to which the clock speed can be increased as mentioned by Herb Sutter in one of the most famous publications, "The Free Lunch Is Over", in the field of concurrency [3]. Now days most of the computers have more than one core. However, if the programmer writes the code in a sequential manner then multi-core machine takes same amount of time to solve the given task as singe code machine does. To solve important computational intensive problems programmer needs to write their code in a special way which makes use of the different cores of the computer.

For more scientific and computer intensive tasks, parallel computing has traditionally been employed with great success. For example, design of airfoils, high-speed circuits, design of micro-electromechanical [4].

The term concurrency is used where multiple computations are executing simultaneously. Depending on the problem the subtasks might need to interact with each other. The process of solving multiple computations simultaneously can be achieved by using multiple cores in the computer, where each core is doing a subset of whole computation. There exist many problems which cannot be solved efficiently without the use of concurrency. For example, in image processing the color space conversion and quantization are computationally intensive and impractical to solve in sequential manner. There also exists many problems which cannot be solved efficiently even with the help of multiple cores. These problems are known to be in NP. The famous NP-complete problem, decision problem of Hamiltonian cycle, needs to check all permutations of n, where n is order of the graph, in the worst case, to decide whether the graph has a Hamiltonian Cycle or not. Even for a small graph of 20 nodes, the computer with around 10 million cores is needed to solve the problem efficiently and the probability of having a computer with these many cores is very minute yet.

Parallel programming is mainly of three types. This division is mainly termed as multiple cores vs many cores. In multiple cores, the code can be run in parallel by making use of the multiple cores of the computer. In many cores programming, an external device such as Graphics Card is used. Graphics card usually contains many more cores than the standard computer. However, it is just limited for the numerical computations only and cannot do anything for the tasks running in the background. Third type is known as cloud computing. In cloud computing, many computers are joined together to do the computations. Big companies like Amazon, Google etc. use this to keep their servers efficient even when millions of people are using them at the same time.

Due to the increasing requirements of making programs efficient, most of the popular programming languages like C++, Java, C# etc. provided the in-built library. The libraries make it easier for programmer to write the code using multiple cores of the computer. In this report, only the Java concurrency library will be discussed.

The report is structured as follows; in section 2, brief introduction of the company with which this project is collaborated is mentioned. In section 3, the scope of the project is being discussed. In section 4, 5 and 6 discusses the project goal and scope. Section 7 discusses the filters and framework of Kiwiplan. The framework introduction is very brief.

II. ABOUT THE COMPANY

Kiwiplan NZ Limited.

Level 3, BDO House
116 Harris Road
East Tamaki
PO Box 58-456, Botany, East Tamaki
Auckland, New Zealand

Phone: (09) 272 7622
Fax: (09) 272 7621
Web: www.kiwiplan.com

About Kiwiplan

Kiwiplan is a software development company that services the corrugating and packaging industries. Typical customers include firms that produce corrugated cardboard products such as boxes, display stands, and other packaging products.

Kiwiplan started business in 1970's as a small corrugating firm. As their throughput increased they developed computer systems to help them keep up with demand. There was considerable interest from other packaging companies in these computerised systems, and the IT department grew and eventually separated from the box plant division.

With over three decades of expertise in developing innovative software for the corrugated and packaging industry, Kiwiplan delivers the total solution for all of your software and Manufacturing Execution System needs.

Kiwiplan is now one of the world's leading software suppliers to the packaging industry. They have customers in 28 countries. All research and development work is done from the New Zealand offices at East Tamaki, Auckland.

Kiwiplan's product range covers the entire business process for a packaging firm, from order entry to shipping. The core products relate to controlling the plant and scheduling the corrugating machinery.

III. Scope Of The Project

Kiwiplan displays the information to the user in the form of tables. They are being termed as flexible tables as users can customize it according to their requirements. Some of the common operations which user can do are sorting, grouping, validating user expression, add/remove columns etc. When user does some operation, the query goes through the sequence of filters where each filter works independently of others and does one task only.

Due to the increasing size of the tables, some of the operations takes too long to complete and hence freezes the GUI. This whole application is written in Java and Kiwiplan interested in knowing how the efficiency of the filters can be increased by making the filter (in general framework) run in parallel. Particularly, this needs to be done by using the features introduced in the most recent version of Java ie. Java 7. Hence, the primary goal of the project is to identify the bottleneck in the application and make it run in parallel. Similarly, other filters needs to be run in parallel, if possible. Most of the code which can be converted to parallel code requires some of its part to run in sequential order. So, careful investigation of the code is needed to see whether a particular filter can be parallelized or not and if yes, then what parts need to be run in parallel and sequential order. So, the objective is to develop a system and integrate it with Kiwiplan GUI application which produces the same output for same input, but with increased performance. Second goal of the project is to investigate how other applications (products) of Kiwiplan can get benefit from it.

There exist many open source libraries made by 3rd party which makes the parallel programming easier. However, company requirement is to use standard java libraries only.

IV. Project Goal

The aim of the project is to develop an application that is

- **Scalable** – For example, the code should be written in such a manner which works in a same way for four cores computers and eight cores computers. The code should be able to figure out itself how many cores there are in the computer and use them as necessary. The fork-join framework, introduced in Java 7, takes care of this issue.

- **Workable with Business Object** – This is one of the most important requirements of the project. Business object is any object that represents something of importance from the real world. Formally, a business object is a "reification of some abstraction that is important in the problem domain".

- **Easy to integrate with the Kiwiplan Products** – Initially, the code will be written as a standalone application i.e. does not depend on the Kiwiplan GUI framework. However, it should be written in such a way that makes it easy to integrate with the existing Kiwiplan Products.

- **Works with the maps** – This is also one of the most important goals. The GUI object does not hold the actual records but the mapping to the records. When certain operation is being done, the computations are being done not on the actual data but on the mappings to the data.

- **Explore Java 7 capabilities** – Though the whole code is written in Java, but in Java 5 and 6, programmer explicitly needs to construct the threads and allocate work to them. However, this is not required to be done in Java 7. Sun's has released the beta version of Java 7 and will be releasing the final version in September this year. However, all the updates they wanted to introduce for the concurrency has already been included in the beta version. Java 7 made the parallel programming even more easier because programmers need not to construct threads and allocate work to them explicitly. The fork-join framework introduced in Java 7, has predefined functions which does the thread work automatically for the programmer.

V. What I Hope to Gain

- **Skills in analysis and design** – To finish the project well, the existing code needs to be examined carefully to see which filters can be parallelized and then what steps of those algorithms can be run in parallel. Of course, each task can be decomposed in many ways hence, the benefits of each decomposition method needs to the evaluated.

- **Practical experience with the software design** – Most of the projects I worked on are small projects. But this project is big and will be developed through each of the stages, from requirements to implementation.

- **Increase programming skills** – Writing parallel code requires more insight to the problem than solving the same problem in a sequential order. This project involves the conversion of sequential code to parallel code and hence, will improve my programming skills.

- **Learning new technologies** – Java 7 is one of the latest technologies and this project requires using their feature in detail. There will be many more features going to be introduced in Java 8 concurrency library e.g. use of closures. Learning new technology is always important and enables programmer to concentrate more on design issue and less emphasis on the implementation.

VI. Filters

Filters are bit low level details of the framework. The main framework is divided into 3 layers.

1) Data Management

This layer retrieves the data from the back-end database and populates the Java business objects.

2) Report Processing

This is where the main computation happens. This layer will be the main focus since the goal is to parallelize the computational intensive tasks.

3) Presentation

Once all the required computations are done, output needs to be displayed on GUI. Making this layer more abstract, output can be sent to GUI, to a web based application, or to some other format such as printing, emailing etc.

The following diagram shows the 3 layers in a systematic manner.

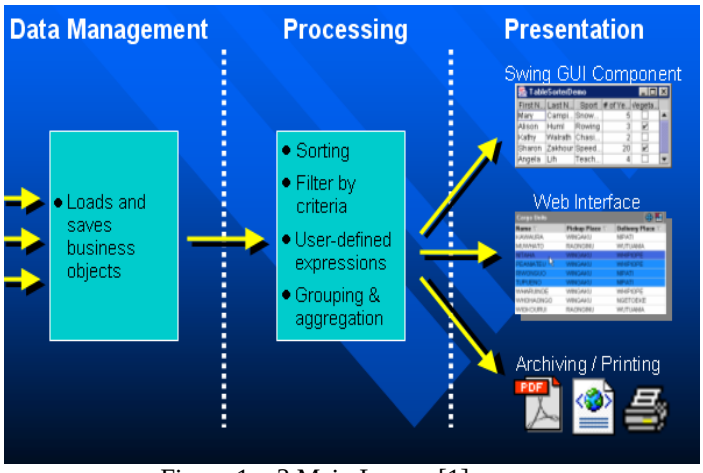

Figure 1 – 3 Main Layers [1]

Kiwiplan uses lots of important design patterns to manage their code. One of the important one is Model-View-Controller (MVC) pattern. In this pattern, the implementation is divided into 3 main parts. In Model part, only the data is kept. View part tells how to use the whole/portion of data. Controller part takes care of any event got fired by the user for example, by clicking button etc.

Kiwiplan termed the second layer (figure 1), Processing, as "Table Model Filter" (I will use filters from now on). Each filter works independently of others and does one specific task. Most of the operations performed by the user on GUI go through certain sequence of filters known as pipeline. Each filter receives the data from the previous filter in the pipeline, do some processing and pass the resulting data to the next filter.

Sorting Filter

There exist many sorting algorithms. The sorting filter implements a shuttle sort algorithm. Shuttle sort is based on divide and conquer strategy and is a variation of merge sort and stable as well [2]. However, there are two important changes.

- **List Cloned** – Instead of creating a temporary list for every merge operation and copying the values back to the original list, the list gets cloned at the beginning. Values are then shuttled between the original and cloned list. At each recursive step, the values get shuttle between the two copies and at the end the final values are in the original list.

- **Performance** – Shuttle sort increases the performance of the merge operation. Before merging one by one, algorithm checks whether the maximum value of the first list is less than the minimum value of the second list or not. If it is, both the lists concatenated together otherwise they merged in a same way as the merge sort.

Shuttle sort algorithm is as follows:

The ShuttleSort algorithm is as follows [1]:

```
shuttleSort(array1, array2, start, end) {
      if (end – start < 2) return

      // divide and conquer ... note that
      // arrays are swapped around
      middle = (start + end) / 2
      shuttleSort(array2,array1,start,
                  middle)
      shuttleSort(array2,array1,middle,
                  end)

      if(max(array1[start…middle]) <
         min(array1[middle…high])) {
            copy array1[start...high] to
            array2
      else {
            merge array1[start...middle]
            and array1[middle...end] into
            array2
      }
}
```

Shuttle sort has a time complexity of *O*(nln(n)). Besides having little bit inefficient than famous Quicksort in practice, shuttle sort has been chosen because it is in-stable in nature. However, with slight modification in Quicksort algorithm, it can become in-stable as well. This is one of scenario where efficiency could be increased even though the algorithm still runs sequentially.

These types of algorithms can easily be parallelized. Most of the algorithms usually decomposed by four techniques, namely, recursive decomposition, data decomposition, exploratory decomposition and speculative decomposition [4]. Recursive decomposition is used for problems than can be solved using divide-and-conquer strategy. Since shuttle sort is based on this strategy, recursive decomposition technique can be used to solve it.

Grouping Filter

The previous filter was an example where pipeline contains one filter only. In most of the cases this is not true. Operation such as "group the records based on some criteria" requires two filters. Initially, the sorting filter sorts the list according to the column we are interested in. Then the grouping filter scans the desired column and starts a new group each time the column value changes. A grouping filter itself groups according to a single column; however multiple levels of grouping can be obtained by adding more grouping filters in the pipeline.

Again, this is one of the scenarios where parallel programming. This task can be decomposed by data decomposition technique.

VII. Conclusion

Some of the common filters have been explained in the report. In particular, sorting algorithms have been discussed. As mentioned, sorting algorithms are the place where actual bottleneck does not likely to exists. However, due to its simplicity it is a good place to start working. The Kiwiplan GUI framework is also been discussed in some detailed. To write/modify any part, programmer should have a sound knowledge of framework structure. Some of the features introduced in Java 7 have also been discussed. These features helps programmer to focus more on the high level design part and leaves the low level details to the operating system or predefined functions. In particular, the parallel array data structure is very simple to use and saves lot of programmer's effort.

VIII. Future Work

Lot of work and effort is required to complete the project well. Concurrent programming is still new to me and I don't have much experience with the predefined libraries. Firstly, I need to get a good knowledge of the framework and see how everything relates together. Secondly, I need to investigate which filters algorithm can be parallelized and find the bottleneck of the program. Thirdly, new features introduced by Java 7 and the fork-join framework needs to be learned as well.

IX. References

[1] Timothy Paul Walker, B.Tech final report, "Kiwiplan Reporting Tool".

[2] Dinneen, M., Gimel'farb G., Wilson M., "Introduction to Algorithms, Data Structures and Formal Languages".

[3] Sutter H., "The Free Lunch Is Over", Dr. Dobb's Journal, 30(3), March 2005.

[4] Grama A., Gupta A., Karypis G., Kumar V., "Introduction to Parallel Computing" second edition, 2003.

[5] Clock speed - http://www.intel.com/technology/timeline.pdf, retrieved on 5th June, 2010.