**Efficient Task-Based Solution for a Given Software Engineering Problem That Uses Parallelism.**

**Hand in deadline: Friday 20th March 2015 before 3:30pm**

**Report**

**By Giovanni Lenguito**

**Contents**

[Introduction 3](#_Toc414476912)

[Technology/System Design 3](#_Toc414476913)

[Number of Cores (Speed Test) 4](#_Toc414476914)

[Performance Evaluation 4](#_Toc414476915)

[Extra Features 5](#_Toc414476916)

[Usage 5](#_Toc414476917)

[Testing 6](#_Toc414476918)

[Attachments 7](#_Toc414476919)

# Introduction

In this document I will explain and include the system design, diagrams and performance evaluation. The software developed is for Spur. At the end of this documentation you will find the testing that was produced. List of extra features will be mentioned in this document also. The application fulfils the user’s requirements which were:

* A Graphical User Interface using WinForms or WPF
* All features should be calculated when the option is selected in the UI, not at initial load. This will allow the data sets to be updated and the calculations rerun without reopening the application. Spur's technical team would like the UI to continue being responsive during these calculations.
* Have the flexibility to point the application to a certain folder on a PC to find the .csv files.
* List all stores, suppliers and supplier types.
* Allow the Finances team to find the following data:-
  + The total cost of all orders available in the supplied data
  + The total cost of all orders for a single store
  + The total cost of orders in a week for all stores
  + The total cost of orders in a week for a single store
  + The total cost of all orders to a supplier
  + The cost of all orders from a supplier type
  + The cost of orders in a week for a supplier type
  + The cost of orders for a supplier type for a store
  + The cost of orders in a week for a supplier type for a store
  + The ability to plot the historical supplier order data on a graph

# Technology/System Design

I will develop the application in C# using WPF not WinForms. This is because WPF is more powerful. The application will use TPL (Task Parallel Library) in order to get the best performance form dealing with a large amount of data. Using tasks prevents the user interface from freezing up and hanging. I will be using Microsoft Visual Studio Professional 2013 as my choice of IDE to develop the application in.

WPF will allow for a more attractive, effective and responsive user interface. The application will include a number of screens which are Settings, About, Welcome (Help), Graphs and of course the main GUI. Everything will be object orientated and will include comments for easy readability for future development such as updates.

# Number of Cores (Speed Test)

Below is a list of speed tests on a set number of cores through 1 – 8 cores. This will test how well the application will run different number of cores using TPL. My machine I am testing on only has 4 cores.

|  |  |
| --- | --- |
| Cores | Parallel |
| 1 | 00:00:13.0856546 |
| 2 | 00:00:09.2489290 |
| 3 | 00:00:09.1924041 |
| 4 | 00:00:08.7855767 |
| 5 | 00:00:08.8368624 |
| 6 | 00:00:08.8983588 |
| 7 | 00:00:08.8372315 |
| 8 | 00:00:08.8619523 |

The code used to test and set the number of cores is as follows:

ParallelOptions po = new ParallelOptions();

po.MaxDegreeOfParallelism = 2; // uses only set core

Parallel.ForEach(files, po, file =>

The computer system knows the machine only has 4 cores. So 5 – 8 cores will be limited automatically to 4 cores. This is why there is no change in speed after the 4th core in the test.

# Performance Evaluation

The time it takes to load all files within a folder for the very first times takes around 30 seconds in parallel, in sequential it takes 1 minute 3 seconds. This is a dramatic difference. Over 50% of time was saved. After this every other attempt takes around 7 seconds. The cause for the long wait on the first load is because the threads need to be allocated by the computer. After the first load of files the application has been given threads for future use. This is a theory that might be possible. The option to switch between parallel and sequential is built in the settings of the application. This will allow you to see the speed difference between the two. Below is a table showing the times taken to load and display the folder selected.

|  |  |  |
| --- | --- | --- |
| Attempt | Parallel | Sequential |
| 1 | 00:00:05.1423689 | 00:00:07.6220202 |
| 2 | 00:00:06.0378696 | 00:00:10.9925419 |
| 3 | 00:00:05.9658013 | 00:00:10.3916339 |

As you can see from the results there isn’t too much of a difference however it is a performance increase. There is around a 4 second increase. Collections used are Concurrent Queues and List. Using Concurrent Queues I found there was less overhead and I could quickly grab data out of it.

When the user selects a folder and clicks ‘Okay’ the application will go through all the files inside that folder only if the file has the extension .csv. Each file will be parsed. The individual data gets turned into an object which is stored in a collection. In this instance the collection I have found to be the fastest to access is the concurrent queue. Two hash sets are set up to hold the data for the supplier, supply type and years. This is then used to populate the filters for the user to use to generate the report. Finally all the data prints to the data grid for the user to review.

The time it takes to generate the report is very small. On average it takes <2 seconds. In the application the thing I found that takes the most amount of time is the printing the data to the screen using the GUI thread. To overcome this issue I generated everything in a separate thread ready for the dispatcher to call and display to the user once the operation is complete.

The speed of the application mainly depends on the end users computer hardware. Running on 4 core processor is no problem.

Using Concurrency Visualiser I could see the application was using a lot of threads. This was due to using parallel and tasks. I also noticed there was a lot of background tasks running on my machine which meant I couldn’t fairly test my application’s time.

# Extra Features

Some of the extra features added were designed to help speed up productivity within the application. Below is a list of features:

* Settings windows has been added which allows the user to save paths to the folder and stores file. This allows the user to click a button to load the folder and auto load the stores. Also there are options that can be turned off or on such as:
  + Graphs will load automatically after a report has been generated.
  + Welcome screen pops up on start up
* There is an option to toggle between Parallel and Sequential code.
* To help the user understand the application there is a welcome screen which can be toggled on or off on start up. The welcome screens tells the user what to do and how to use the application.
* The status bar at the bottom will display all the times needed to know for the assignment on the left. On the right there is a progress bar and a cancel button that will appear when an operation is in progress such as generating the data.
* A graph has been added which shows the historical supplier order data.
* User can access the calculator easily is needed.
* About window to give important information on the application

# Usage

The application uses around 630.7MB when 1040 files have been parsed and stored in collections. Also the report being generated. The application uses only 16.2MB on first load and with no interaction. There are no known memory leaks or Out Of Memory Exceptions.

# Testing

The data the finance team needs works and functions correctly. Below is a test of the application working.

|  |  |  |  |
| --- | --- | --- | --- |
| Screenshot | Supposed to happen | Happens? | Fix |
|  | Welcome windows pops up if show welcome is true, else don’t show | True | N/A |
|  | ‘Select Folder’ button opens explorer window and allows user to select a folder | True | N/A |
|  | When folder is selected data will load, populating combo-boxes and data grid. As well as update status bar | True | N/A |
|  | ‘Load Stores’ button opens explorer window and allows user to select a file named StoreCodes.csv | True | N/A |
|  | When file selected and ok is clicked, the combo-box will populate with the data in the csv file | True | N/A |
|  | If ‘Generate’ button clicked and no filters have been selected message box will appear | True | N/A |
|  | If filters have been selected and ‘Generate’ button has been clicked the data grid will populate selected items and the statistics will update on the right, as well as the status bar. | True | N/A |
|  | Graphs load the historical supplier order data from selected data | True | N/A |
|  |  |  |  |
|  | If ‘Settings’ button is clicked the settings windows will pop up and display saved data from previous settings made. | True | N/A |
|  | Selected path on both textboxes will pop up explorer to allow the user to select their paths. On save the data is saved even when the application is closed. All save functionality works. | True | N/A |
|  | If ‘Calculator’ button clicked the windows calculator application will pop up. | True | N/A |

# Attachments

* L010516C\_Activity\_Diagram.vsdx
* L010516C\_Class\_Diagram.vsdx
* L010516C\_User\_Case.vsdx