

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

Hand in deadline: Friday 20th March 2015

Report
By Giovanni Lenguito

Contents

Introduction.....	3
Technology/System Design.....	3
Number of Cores (Speed Test)	4
Performance Evaluation	4
Extra Features	5
Usage	6
Testing	6
Attachments	8

Introduction

This document will explain and include the system design, performance evaluation and diagrams. The software developed is for a finance team at a company called spur (Staffordshire University Assignment). 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

The application will be developed 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 from dealing with a large amount of data. Using tasks prevents the user interface from freezing up and hanging. Linq is used to group the graph data.

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), Graph 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. The testing machine running the tests 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 there was less overhead and the data can be quickly accessed.

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 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 found that takes the most amount of time is the printing the data to the screen using the GUI thread. To overcome this issue everything gets generated 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 the application was using a lot of threads. This was because the application was using too many parallel for each's and tasks, to speed up the application the unnecessary parallel for each's and tasks were removed. Also there was a lot of background tasks running on the testing machine.

Improvements could be made on the speed. The application could run dummy tasks so the computer allocates the application threads on load of the program. This could potentially improve the speed on the first start up. This is because the computer will allocate threads to the application ready for large tasks.

Extra Features

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

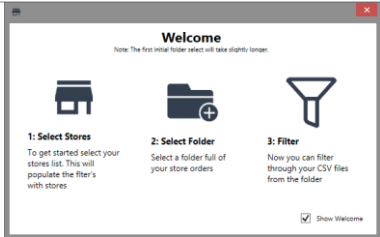
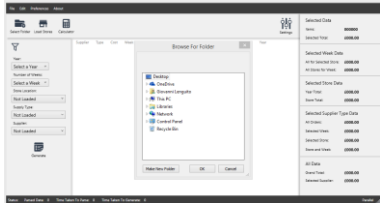
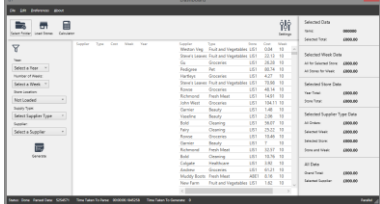
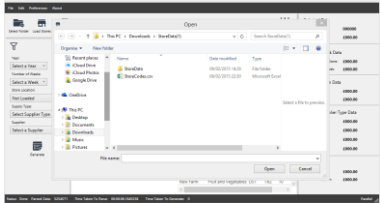

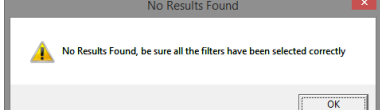
- Settings has been added which allows the user to save paths to the folder and stores file. This allows the user to click a button to load the folder and auto load the stores. Also there are options that can be turned off or on such as:
 - Graphs will load automatically after a report has been generated.
 - 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, this uses linq the group data.
- Users can access the calculator easily is needed.
- About window to give important information on the application
- The user is able to cancel a task from the bottom right on the status bar at any time when a task is running. The button will appear automatically.
- Added extra statistics that may be relevant to the selected data.



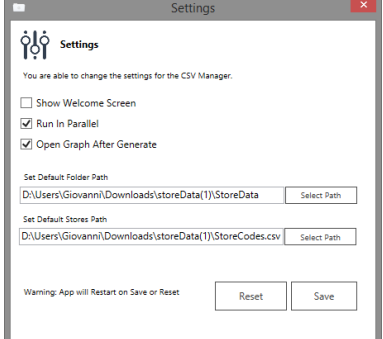
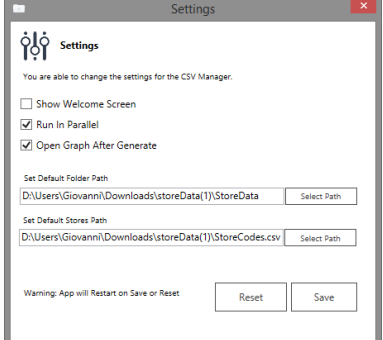
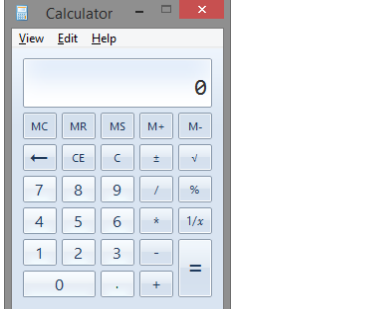
Usage

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 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

	<p>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.</p>	<p>True</p>	<p>N/A</p>
	<p>Graphs loads the historical supplier order data from selected data. Correct data is displayed.</p>	<p>True</p>	<p>N/A</p>
	<p>If 'Settings' button is clicked the settings windows will pop up and display saved data from previous settings made.</p>	<p>True</p>	<p>N/A</p>
	<p>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.</p>	<p>True</p>	<p>N/A</p>
	<p>If 'Calculator' button clicked the windows calculator application will pop up.</p>	<p>True</p>	<p>N/A</p>

Attachments

- L010516C_Activity_Diagram.vsd
- L010516C_Class_Diagram.vsd
- L010516C_User_Case.vsd