# Nash Tech Homework

# Document Information

|  |  |
| --- | --- |
| Author(S) | Shashwat Anand |
| Version | 1.0 |
| Date | 23nd Apr 2017 |

# 1. Business Summary

## 

## 1.1 Objective

The objective is to search patterns and store the search results into filesystem. This can be used later for reporting.

## 1.2 Design Abstract

As part of this requirement, following things has to be implemented: -

1. Retrieve a pattern call with a specified identifier, e.g. 42
2. List all pattern calls with a specified name, e.g. “myPattern”
3. List all pattern calls with a specified path, e.g. “src/patterns/Functional.pat”
4. List all pattern calls which are skipped, i.e. when the “called” flag is false
5. List all pattern calls which are not skipped, i.e. when the “called” flag is true
6. Dump the queries into an output file

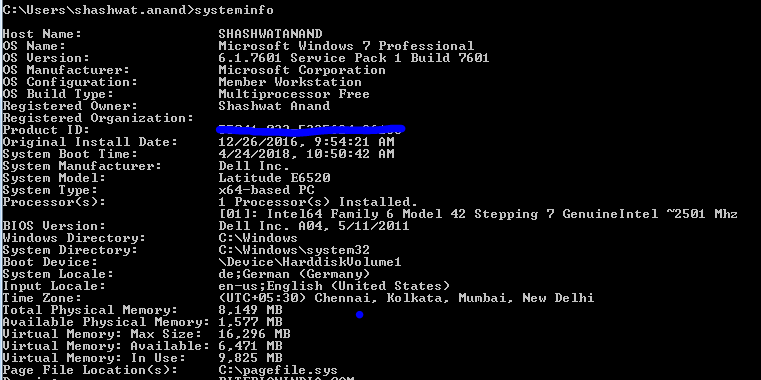
1.3 Volumetric

Not known

# 2. System

# 2.1 System Configuration

Below is my system configuration. It is around 4 year old machine.



# 2.2 Design

Input Console or File

(Java NIO for File)

(UI Eclipse Plugin)

Controller/Search Module

Output in Console/File

Testing

(Junit)

UI testing

(Manually)

# 3. Benefits

1. UI application
2. Colour Coding used for showing error or success in search results
3. UFX

# 4. Trade Off/Assumption

Assumption

1. Application will be tested in windows 64 bit machine
2. JDK 8 is used for developing the application so application should only tested on machines where JRE 8 is installed.
   1. Will also provide bundled jre in executables (but can’t test this fully as lack of environments)
3. Eclipse Neon is used as runtime for the application, which can be changed with simple configuration in POM file.
4. Tuples format is - ID, “Name”, Pattern, flag
   1. 1, “MyPattern”, src/patterns/Functional1.pat, false
   2. 2, “myPattern”, src/patterns/Functional2.pat, true
5. All search queries are case-sensitive.
6. In dumping queries into file functionality, I am writing the whole file (not appending it). Also I added \r\n for line break, so user of the system will have proper output of file in windows notepad.

Trade Off

As requirement document say tuples need to be in memory. I consider my development machine as the minimum-maximum sustainable system.

# 5. Improvement

1. Giving support to search using wild cards.
2. UI testing can be automated.
3. Adding support and testing for different platforms

# 6. Time Complexity

As far is time complexity is concerned for queries is Big O (n)

As you can see in class SearchResultProvider, I used stream and parallel stream. Stream filter always traverses the complete and applies the filter criteria to each element.

I used parallel stream only when if number of available processor is more than 1 and number of tuples in memory as more than 1000. Parallel stream give better performance if it has to act on more data to process.

# 7. Remark

Design took maximum time, had to take lot decision

1. UI or Console Application
2. Should I use functional paradigm of Java

8. Load Analysis

With all my day today applications, I found the following results. When I loaded the application with 300 MB text file with 129 processes running on system (Fig 8.1). Following are the observations.

* System was using 5.75 GB of RAM (Fig 8.2)
* After loading the 300 MB file RAM utilization went up by 700 MB (approx.) (Fig 8.3).
* It took 5 seconds to load the file “Time take to load file : 4820” ms

To summarize if we want to load 1 GB file then we may require 2.5 GB RAM for loading the file with you day to day application.

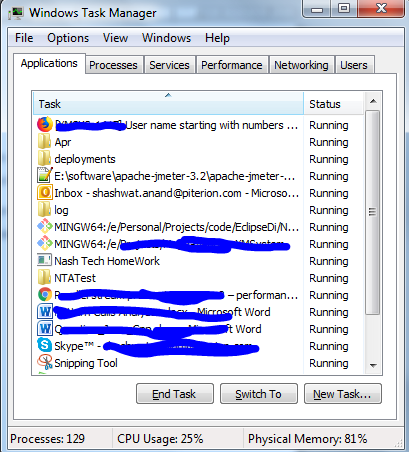


Fig 8. 1

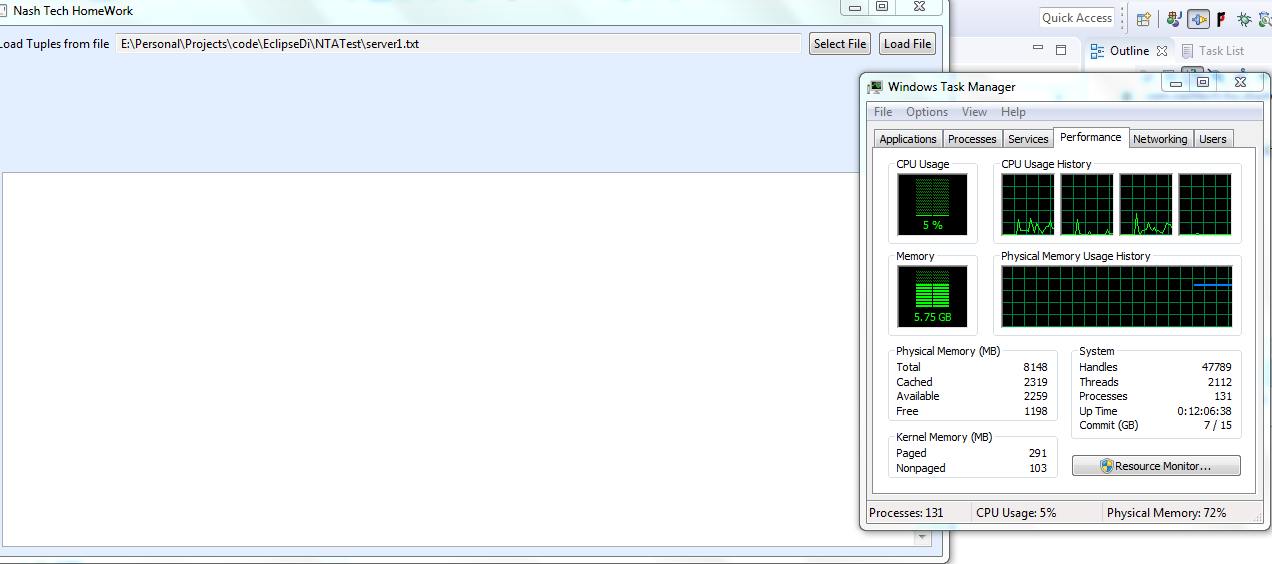


Fig 8. 2

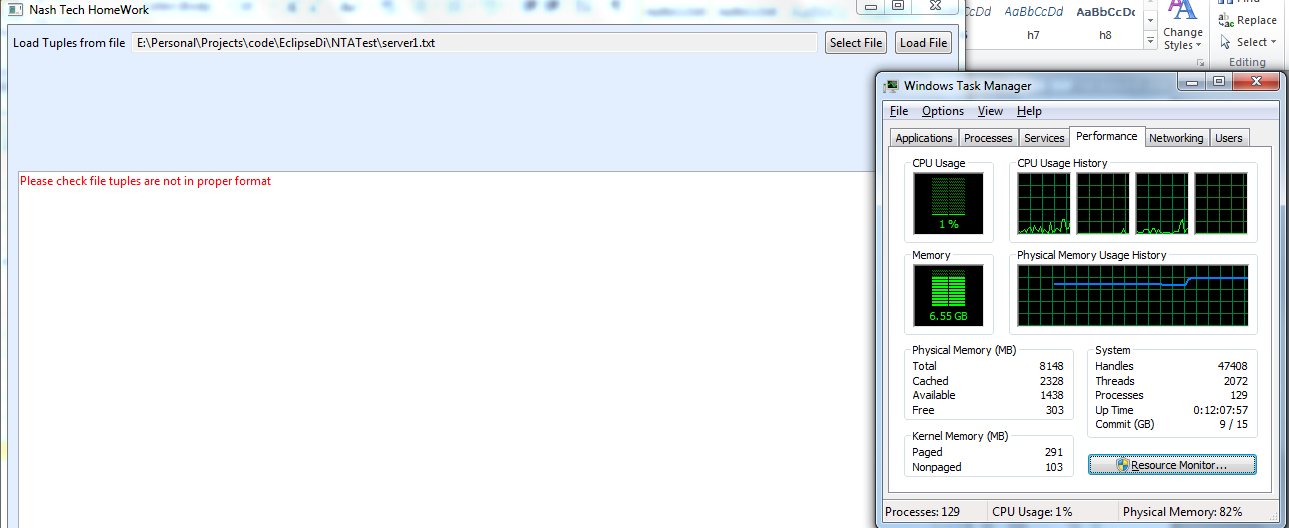


Fig 8. 3