MATLAB APPLICATION

Documentation

University of Wisconsin - Madison

April 16, 2013

INDEX

1. PROBLEM DEFINITION

1.1. *Problem abstraction*

The purpose of this application is to help people submit their applications and data to Condor system, then return the output which is processed by Condor to users. It also tries to figure out what Matlab can and can not interact with Condor system.

1.2 *The current application*

There are some applications that allow people to interact with Condor, but people must have the knowledge about Condor. If they don't know how to use Condor, they can't use these applications.

1.3 *The proposed application*

Developing Matlab application which helping people who don't know about Condor use Condor system to process their data.

1.4 *Boundaries of the System*

Submit data to Condor then get output results back from Condor system, in Matlab environment.

1.4.1 What system can do:

- Submits custom functions and input data to Condor, then returns output data back.
- Can resume the application at any time.
- Uses graphs to illustrate data.
- Can submit custom-compiled functions which are written by any languages.
- Can be used from Matlab command windows or operating system command windows (Linux, Windows).
- Friendly and easy to use.

1.4.2 What system can not do: (and its disadvantages)

- The custom-compiled functions must have some following conditions:

  - The input of functions must be an array.

  - The functions must display output to screen.

  For example, this is a C function:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
```

```
{
        FILE* file = fopen(argv[1], "r");
        float i = 0;
        fscanf(file, "%f", &i);
        while(!feof(file))
        {
          float a = i * i;
                printf("%f ", a);
                fscanf(file, "%f",&i);
        }
        fclose(file);
return 0;
}
```

- Can not run without Matlab environment.
- The application runs quite slowly, because it is written by a scripting language (Matlab).

1.5 *Hardware and Software Requirements*

1.5.1 Minimum Requirements

Procesor(CPU)	CPU Intel 800, RAM 265 MB
Disk space	100MB
Sofware	- Windows, Linux or Unix - Matlab - Condor

1.5.2 Recommended Requirements

Procesor(CPU)	CPU Intel(R) Core 2 Quad , RAM 4 GB
Disk space	1 GB
Sofware	- Windows, Linux or Unix - Matlab 7.13 - Condor 7.8.1

1.6 *FAQ*

- **What is the application?**

The MATLAB application has 2 modules:

1. Module DrawGraph: can submit millions of jobs and a custom function to Condor system, then get the output results back and display them. It can also display the histogram of running times of the jobs.

2. Module ResumeGraph: can resume current task.

❖ **Who are the users?**

The customers are the people who know MATLAB or other languages (C, C++, Java, Python...), but they don't know about Condor. However, they still want to get benefits from Condor system to solve their problems. So they can use this application to submit their jobs and their custom function.

❖ **What is the custom function?**

Custom function is an arbitrary function that user writes to calculate the output of the jobs. For example: a custom function A can be: Y = X * X. So when user submit their jobs with values X = 1, 2, 3 and the custom function A to Condor system. The Condor system will use A to calculate output values: Y = 1, 4, 9, then return these output values to MATLAB application. The MATLAB application will display the graph of X and Y: (1, 1) (2, 4) (3, 9) to the screen.
The custom function can be written by any language. It can be the source code (ex: func.cpp or compiled code (ex: a.out). However, Condor system must support this language because this custom function will be executed (in case compiled code) or compiled then executed (in case source code) in Condor system.

❖ **How can the MATLAB application submit few million jobs to Condor system?**

It divides million jobs into smaller chunks, then submit these chunks sequentially. For example: when user try to submit 1M jobs, the MATLAB application will submit first 10k jobs to Condor system. When 3k jobs in the total 10k submitted jobs done, MATLAB application will submit next 3k jobs, and so on...

2. CUSTOMER REQUIREMENT SPECIFICATION

2.1 Users of the System

All users who want to use Condor to process their data.

2.2 Application functions

The application has some versions for various purposes:

2.2.1 2D Console pure-MATLAB version

This version gets input points then draw graph in MATLAB. It doesn't submit jobs to Condor (for demo purpose).
For more detailed information, please see the documentation of each version.

2.2.2 2D Console version 1

This version submits a chunk of 10,000 jobs to Condor, then wait for all the submitted jobs complete. After that, it will submit next chunk to Condor.
This application accepts params in console mode. Input value is an array of X. The results that are displayed on graph are points of (X, Y).

2.2.3 3D Console version 1

This version submits a chunk of 10,000 jobs to Condor, then wait for all the submitted jobs complete. After that, it will submit next chunk to Condor.
This application accepts params in console mode. Input value is an array of (X, Y). The results that are displayed on graph are points of (X, Y, Z).

2.2.4 2D Console version 2

This is the latest version and is implemented with new algorithms. It submits first chunk that has 10,000 jobs, then it will submit next 3,000 jobs if the total jobs in Condor queue <= 7,000 jobs.
This application accepts params in console mode. Input value is an array of X. The results that are displayed on graph are points of (X, Y).

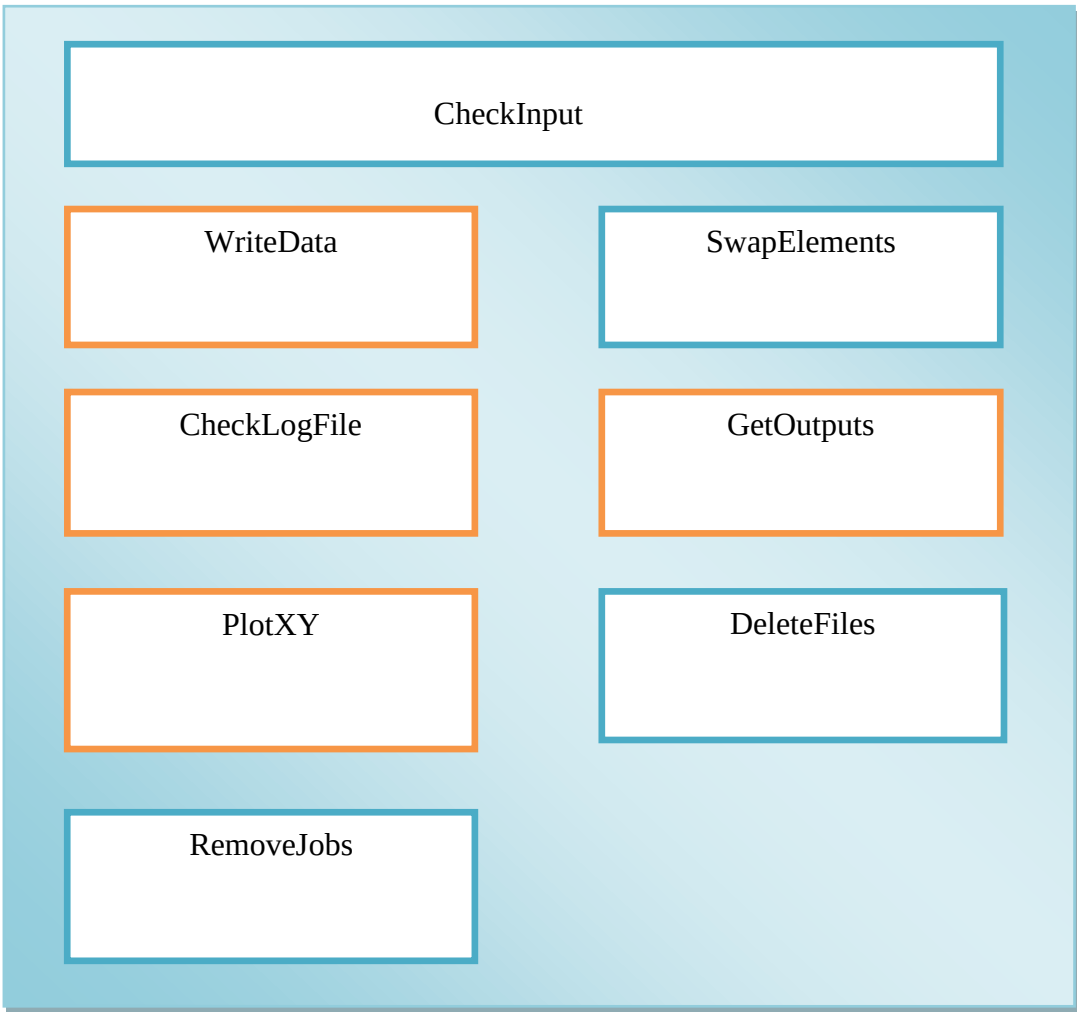
2.2.5 2D GUI version 2

Similar to 2D Console version 2 in section 2.2.4, but it accepts params and display information in GUI mode.

3. APPLICATION DESIGN

3.1 Architecture Design

- **DrawGraph**

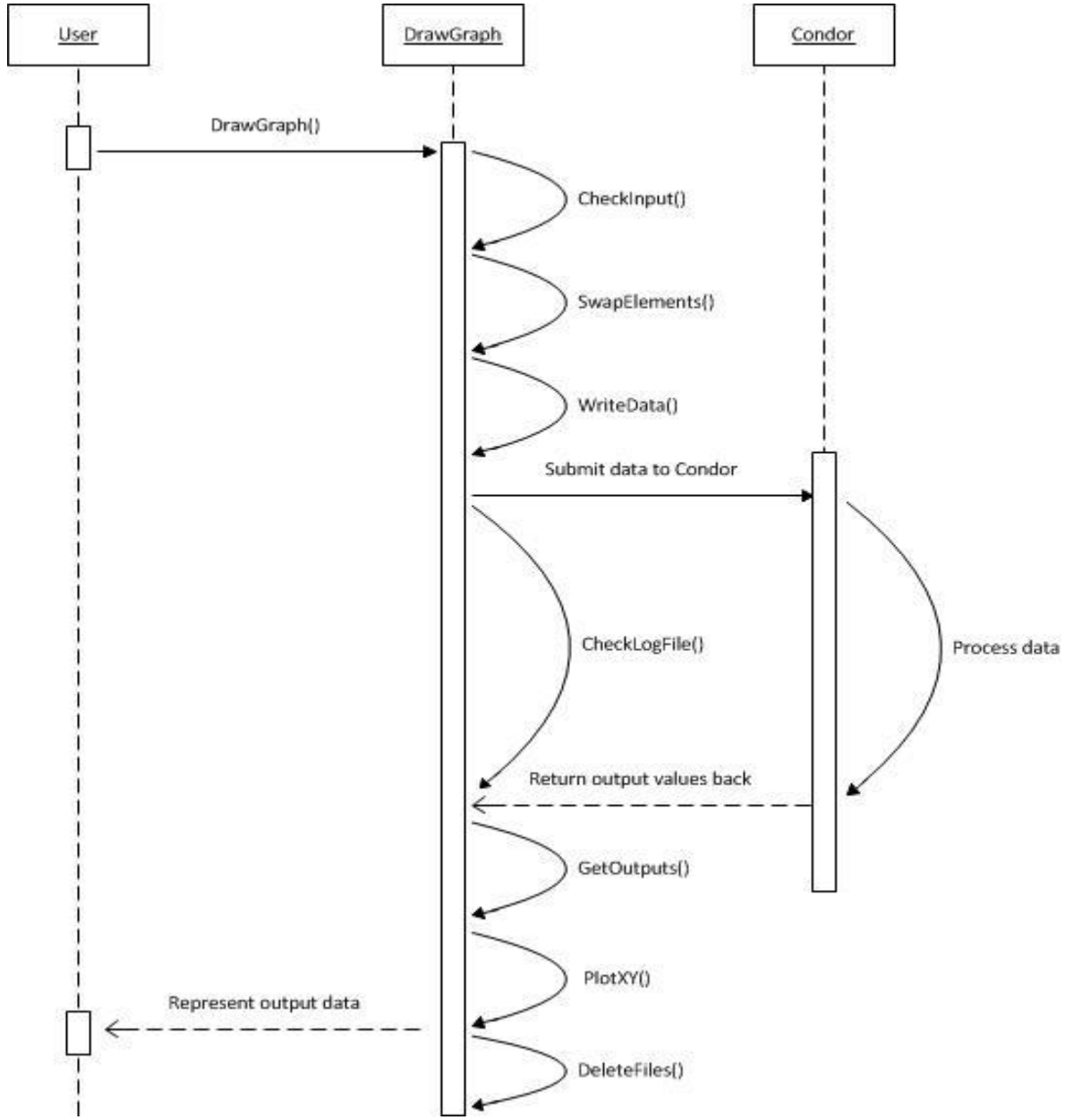


For more detailed information, please see documentation.

3.2 Diagrams:

- DrawGraph

- Sequence diagram:


User
DrawGraph
Condor
DrawGraph()
CheckInput()
SwapElements()
WriteData()
Submit data to Condor
CheckLogFile()
Process data
Return output values back
GetOutputs()
PlotXY()
Represent output data
DeleteFiles()