SPM 2013-2014: Final project

Version 0.1

December 9, 2013

The project is assigned to individual students or, exceptionally, to groups of max 2 students. The project has to be prepared and sent to the teacher by one of the deadlines (exam dates) published on the course web site[1] (and on secretary web site). One deadline per exam session will be given. The project sent to the professor via email *must* consist in:

a message with subject "SPM project submission"
a PDF document, in attachment, with the project report, of max 10 pages
a `tar.gz` document, in attachment, with the project code, the examples, the makefiles, and all what's necessary to recompile and run it

Each one of the two attachments is described in detail later on in this document (see Sec. 3). After receiving all the projects relative to the exam session, the teacher will take about one week to mark them (a little bit more in case of a huge number of projects submitted) and then he will publish a calendar of oral exams for the students that submitted a project eventually ranked sufficient or higher. The oral exam is made of two parts:

- a short demo of the project run by the student using one or more text (only) terminals connected via SSH to the parallel machines where the project has been developed. During the demo the student will be asked to answer questions relative to the project structure, code and execution behaviour. Possibly, the student(s) may be asked to implement small changes in the project code[2].
- two/three questions relative to the topics presented and discussed in the course and covered by the teaching material.

At the end of the oral exam, the student will get the final exam mark registered. In case, the student may submit the project at exam session i and have the exam at session $i + k$ provided it is in the same period (e.g. submit the project in June and have the oral exam in July, but not in September).

1 Project subjects

The student can choose one of the two projects listed below. Both projects are somehow "underspecified": part of the project has to be defined by the student. As an example, the "skeleton" projects do not specify any use case to be used to test the implementation. It is up to the student to figure out proper tests/simple applications, in this case. The complete definition of the project out of the project schema presented here is part of the project itself and it will be evaluated during the exam.

1.1 "Skeleton" projects

The final goal is the implementation of a simple run time/library for one of the structured parallel programming patterns discussed in the course. In other words, the student must provide some code that can be used to implement a generic application exploiting the parallel pattern subject of his/her

[1] `http://didawiki.cli.di.unipi.it/doku.php/magistraleinformaticanetworking/spm/start`

[2] usual Linux text only editors will be available: vi, emacs, pico, etc.

implementation. The implementation may be written using C, C++ or Java and must run on POSIX (Linux) workstations. Students may consider to use the "core" FastFlow components to implement the skeletons in addition to the POSIX/C++ framework. Depending on the skeleton, COW/NOW, multi cores or even network of multi core architectures should be targeted. The reference target architectures, that is the ones where the project will be run during the final demo, are

1. the Linux PC in Aula H (or aula M or aula I)
2. a symmetric multicore [3] which will be made available by the teacher to the students targeting multicore only architectures. This year, students making their projects using C++ programming frameworks will be given an account on a machine equipped with a Xeon PHI board. Students using Java based frameworks will be given an account on a Nehalem multicore.

This year we will consider the following skeletons (that is the project consists in implementing one of the following skeletons):

Divide&Conquer skeleton. The student must implement a classical divide and conquer skeleton taking as arguments the function computing the solution for a base case (b), the boolean function stating whether a task is a base case (isB), the function splitting a tasks in subtasks (d) and the function "conquering" the subresults from subtasks into a result for the task (c) and executing, in parallel, the algorithm:

```
1: function DC(isB,b,d,c,X)
2:     if isB(X) then
3:         return b(X)
4:     else
5:         return c(map (divide&conquer(d, isB, b, c)) (d(X)))
6:     end if
7: end function
```

Pool skeleton. The student must implement a skeleton computing the evolution of a pool of individuals, subject to "mutations". The pool skeleton must implement, in parallel, the following iterative algorithm:

```
1: function POOL(termination, select, evolve, filter, Pop)
2:     while not(termination(Pop)) do
3:         N = evolve(select(Pop))
4:         Pop = Pop ∪ filter(N,Pop)
5:     end while
6: end function
```

where $termination$ is a boolean function checking if the current population is the "good" (final) one, $select$ selects (part of the) individuals to be mutated from the current population, $evolve$ mutates and individual and $filter$ selects which one of the mutated individuals must be inserted in the current population. At least the $evolve$ function must be implemented in parallel. Students must consider the possibility to evaluate in parallel also the other functions parameter of the pool skeleton.

Distributed farm with load balancing The student must implement a two tier task farm skeleton where i) the inner workers are farms running each on a separate multicore processing element and ii) the outer farm distributes tasks to the inner farms and achieves load balancing through proper task scheduling *and* job stealing policies.

Domain specific skeleton The student may propose a new/different skeleton, that is a new/different parallelism exploitation pattern which is efficient, resusable and parametric, as other skeletons are. The new skeleton has to be discussed with (and approved by) the teacher *before* actually starting the project.

[3] most likely `andromeda.di.unipi.it` or other multicores possibily available

In all cases, the project report *must include*:

a description of the concurrent activity graph and the implementation graph
the proper performance models related to the skeleton (abstract model) and to the implementation (concrete model)
the code implementing the run time support
the comparison among the predicted and measures performances (completion time and/or speedup/scalability)

and the skeleton should be implemented

- targeting either a single multi core or a COW/NOW, in case the implementation/project is developed by a single student, or
- targeting a network of multi core workstations, in case the implementation/project is developed by a group of two students.

1.2 "Application" projects

The final goal is the implementation of an application using a skeleton based structured programming environment. The candidate programming environments are those introduced and discussed during the course, that include[4]:

Env.	Host lang.	Target hw
FastFlow	C++	multicore + COW/NOW
Skandium	Java	multicore
SkePU	C++	multicore (data parallel only)

Depending on the framework chosen, three kind of architectures may be targeted: multi cores, COW/NOW or network of multi core workstations. The reference architectures will be `andromeda` in the first case and the machines in Aula H, M and I, in the second and third case.

This year we consider the following applications (that is the project consist in implementing one of these applications, using a structured parallel programming framework):

Graph search Search a given item in a (large) graph in parallel. The graph to be searched should be produced using the SNAP library from Stanford (available at `snap.stanford.edu/snap/`). The application should search the large graph for a single node or for a set of nodes.

Matrix multiplication An application should be implemented performing (dense) matrix multiplication on a stream of matrix pairs with at least two different algorithms. The application should eventually give a comparison of the relative performance of the algorithms. Systolic algorithms should be considered as an option (see as an examples the slides at `http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf`).

Hello world "genetic" The application considers an initial population of random genes representing strings. At each iteration, a subset of the genes is selected and mutated to obtain a new set of genes that is eventually inserted in the original population. The "fitness function" to be used to evaluate genes is the closeness of the gene to the generic "Hello world" string, expressed as number of characters different from the ones appearing in the target string. Different mutation mechanisms should be considered (including random mutation of a character and crossover of two genes) and different parallelisation strategies should be evaluated.

[4] but are not limited to. Students may consider other structured parallel programming environments (to be approved by the teacher).

Help Desk

Any question relative to the project (design, coding, debugging) can be posed during the lesson breaks, during the question time (question time will be active even in periods without lessons) or by email (simple questions only). Questions relative to the programming environments may also be sent to M. Torquati and T. De Matteis (via email).

Free application The student can pick up an application in his/her favorite domain and implement the application using a skeleton framework. The application has to be discussed with (and approved by) the teacher *before* actually starting the project.

In all cases, the project submission should eventually include:

The design of the implementation with the available skeletons, including an estimate of the performances
The application implementation
A comparison of the performances achieved with the ones expected

2 Project assignment

When a student (group of two students) decides to start working on the project he/she should first "negotiate" the project with the professor. He/she communicates to the professor the project chosen sending an email (subject "SPM project choice"). The email text should give an idea of i) which project subject has been chosen and ii) of the unspecified parameters of the project. In case of "free application", as an example, the application chosen should be introduced; in case of any application, the framework chosen for the implementation is to be specified; etc. The professor, in a short amount of time, returns an acknowledge message possibly including more detailed requirements and/or modifications of the choices made by the student/s. In particular cases, the professor may ask the student(s) to discuss the project assignment during question time. After the acknowledge, the assignment is registered on the project web page[5] and the student(s) may start working. The acknowledgment may be implicitly substituted by the pubblication of the assignment on the project web page.

3 Submission details

3.1 Project report attachment

The project report is a short report (no more than 10 pages, excluding code). It must include:

- the specification of the project chosen, as agreed with the professor
- the general design of the work done
- the peculiarities tackled when implementing the project
- the comparison among performance predicted with the performance models and the one observed during the experiments
- a one page "user manual" explaining how the code may be compiled and run

Please do not copy&paste parts of the project text, of the SPM notes book, etc. Conciseness of the project report is appreciated. The ability to report only important facts is also evaluated.

[5] `http://didawiki.cli.di.unipi.it/doku.php/magistraleinformaticanetworking/spm/spm1314pro`

3.2 Project sources

The whole sources of the project, including

(full) code
sample code/data used to exercise the run time support or to run the application
makefiles or ant files used to compile the project
scripts used to run the project (if any)

must be sent as a tar, gzipped file. Following the instruction in the project report the teacher should be able to run the code with proper inputs and to observe the results described in the project report.

The source code file should be named as follows: `NameFamilyname.enrollmentNumber.tar.gz` As an example, I would submit a file with name `MarcoDanelutto12345.tar.gz` In case the project is prepared by two students, the project file should be named using both names and enrollment numbers.

The code should be decently commented. In case of usage of tools such as `javadoc` or `doxygen`, the commands necessary to generate the documentation should be included in the proper `makefile` or `ant` files.

The code may be developed on any machine, including student notebooks or other machines they have access to, but as far as the evaluation process is concerned, the code must run either on the Aula H/I/M machines or on `andromeda.di.unipi.it`.

Project validity

The project described in this document is valid for all the exam terms in Academic Year 2013-2014, that is from January 2014 to September 2014 exam sessions. The assignment may be required at any time since project publication on. The assignment is valid up to moment a new, different assignment is required by the student(s) or up to start of the 2014-2015 course. The start of next year course "resets" any pending project work.