SPM 2011: Final project

M. Danelutto

Version 1.0

The project is assigned to individual students or to groups of max 2 students. The project has to be prepared and sent to the teacher by one of the dates published on the course web site (secretary web site). The project sent to the professor via email *must* consist in:

1. a message with subject "SPM project submission"
2. a PDF document, in attachment, with the project report, of max 10 pages
3. 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.

After receiving all the projects relative to the exam session, the teachers will take about one week to mark them (a little bit more in case of a huge number of projects submitted) and then they 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:

1. a short demo of the project run by the student using one or more text 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, behaviour.
2. two/three questions relative to the topics presented and discussed in the course and covered by the teaching material (project course notes + book chapters covering the last part of the course arguments).

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. 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/implementation for one of the structured parallel programming patterns discussed in the course. The implementation may be written using C, C++ or Java and must run on POSIX (Linux) workstations. 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. the multicore `ottavinareale.di.unipi.it` which will be made available by the teacher to the students targeting multicore only architectures.

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

Stencil skeleton. The student must implement a stencil skeleton. The stencil skeleton takes as an input a matrix (2 dimensions), a stencil (a vector of index offset pairs defining the stencil) and a function (the function to be applied to the stencil) and computes a matrix whose element i, j comes from the computation of the stencil function on the stencil computed on the input matrix centered on item i, j. In this cases, the space of the indexes has to be considered toroidal. If the matrix dimension is $N \times N$, all matrix indexing is to be considered modulo N. The element $A[N][i]$ (with $i < N$) is therefore $A[0][i]$ as $N\%N = 0$.

As an example, the computation of the skeleton

```
stencil f [<-1,0>,<0,-1>,<0,1>,<1,0>] A
```

should return a matrix whose element i, j is the result of the application of f to the parameters $a_{ij}, a_{((i-1)\%N)j}, a_{i((j-1)\%N)}, a_{i((j+1)\%N)}, a_{((i+1)\%N)j}$

Please take into account that `<0,0>` is not present in the stencil, but a_{ij} must be the first parameter passed to function f.

It is up to the student to define whether the new matrix is the matrix A modified *in-place* or a new matrix B completely distinct from the first one.

Parallel prefix skeleton. The student must implement a parallel prefix skeleton. The parallel prefix skeleton takes as an input a vector, a binary associative and commutative function and computes the parallel prefix (scan) of the vector.

As an example

$$\texttt{parallelPrefix} \oplus \texttt{V}$$

should return a vector whose element i is equal to $v_0 \oplus v_1 \oplus \ldots \oplus v_i$

Parameter sweeping skeleton. The student must implement a parameter sweeping skeleton. The parameter sweeping skeleton takes as parameters a function f, a comparison function `comp` (a binary function returning a boolean, `comp`(a,b) = `true` iff a is "better" than b) and a set of input data sets $s_1, \ldots, s_k$, and returns the input data set s_i such that $\forall j \neq i$ `comp`(s_i, s_j) = `true` The student must consider the possibility f is given as an application (executable reading input parameters from standard input). In this case we may assume the "result" of the application is an integer, and the `comp` simply evaluates $a > b$.

Domain specific skeleton . The student may propose a new/different skeleton. The new skeleton has to be discussed with (and approved by) the teacher *before* actually starting the project.

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

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.

A comparison of the expected performance vs. the achieved performance figures must be included in the project report.

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 is:

1. Muesli (C++)
2. SkeTo (C++, data parallel only)
3. FastFlow (C++)
4. Skandium (Java)
5. Muskel (Java)
6. OcamlP3L (Ocaml)

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 `ottavinareale` 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):

Directory comparison The application compares the files in two directories and returns an answer telling whether the directories contain the same files (same names, same contents, same position) or not. This is more or less the `diff -r` Linux/Unix command.

Image stream processor The application receives a stream of images and converts each one of the images applying a filter. The choice of the filter is left to the student. The simpler case is to consider *color image to black and white* filter. In this case the application *must* exploit both stream and data parallelism.

Matrix multiplication The application is actually a library providing a function computing matrix multiplication (integer matrixes).

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 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 and the student(s) may start working.

3 "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 P. Dazzi (via email).

4 Project submission: attachments

4.1 Project report

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.

4.2 Project sources

The whole sources of the project, including

- 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`

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 `ottavinareale.di.unipi.it`.

5 Project validity

The project described in this document is valid for all the exam terms in Academic Year 2010-2011, that is from June 2011 to February 2012. 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 2011-2012 course. The start of next year course "resets" any pending project work.