SPM 2012-2013: Final project

Version 1.0

May 13, 2013

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 deadlines (exam dates) published on the course web site[1] (and on secretary web site). One deadline per exam session will be given. At the moment being, the first three deadlines (relative to the Summer session exams) are fixed on June 4th and 24th and on July 17th, respectively. The last deadline of the term (last deadline in June-July, September deadline and the last deadline in January-February) may be extended (one week) provided the student sends an email to the professor with the work made so far by the official exam deadline. 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]http://didawiki.cli.di.unipi.it/doku.php/magistraleinformaticanetworking/spm/start

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

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 implementation. 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. a symmetric multicore [3] 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):

Orbit skeleton. The student must implement an "orbit" skeleton. The orbit skeleton accepts an input data set S whose elements have type α, a set of *generator functions* $g_1, \ldots, g_k$ such that $\forall i \; g_i : \alpha \rightarrow \alpha^\star$ and a function $check : \alpha \times \alpha \rightarrow Bool$ returning true in case the two elements may be considered "equal", and computes the transitive closure of the generators over the initial set S. In other words the skeleton considers the elements in S, generates new elements of S using the genenerators $g_1, \ldots, g_k$ and adds to set S the elements which are not already present. The computation stops when no new element may be added to the set S.

The skeleton implements therefore the following algorith:

```
do {
  for all gi
    for all s in S
      Si = {elements generated by gi(s)}
        for all x in Si
          for all y in S
            if(!check(x,y)) then add x to S
} while (new elements added to S);
```

The possibility to compute the skeleton on a stream of different input sets $S_1, \ldots, S_m$ with a unique set of generators $g_1, \ldots, g_k$ should be taken into account.

Knapsack skeleton. The student must implement a skeleton computing the solution of a knapsack problem, that is, given a set of items $S = \{item_i = \langle cost_i, weigth_i \rangle \mid i \in [1, k]\}$ compute the number and type of items such that the total cost is maxmized subject to the constrain that the total weight is smaller that W.

The skeleton should work on a standalone problem instance $\langle S, W \rangle$ as well as on a stream of instances $\langle S_1, W_1 \rangle, \ldots \langle S_m, W_m \rangle$.

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.

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

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

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 include[4]:

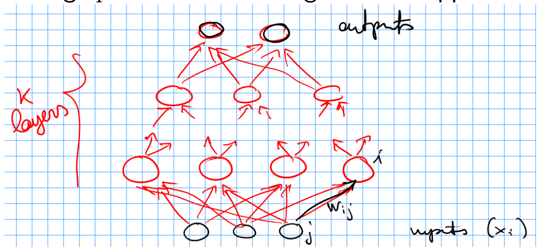
Env.	**Host lang.**	**Target hw**
Muesli	C++	multicore + COW/NOW
FastFlow	C++	multicore + COW/NOW
Skandium	Java	multicore
SkePU	C++	multicore (data parallel only)

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 `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):

Invasion percolation Invasion percolation models the displacement of one fluid (such as oil) by another (such as water) in fractured rock. In two dimensions, this can be simulated by generating an N×N grid of random numbers in the range [1..R], and then marking the center cell of the grid as filled. In each iteration, one examines the four orthogonal neighbors of all filled cells, chooses the one with the lowest value (i.e., the one with the least resistance to filling), and fills it in. Filling begins at the central cell of the matrix (rounding down for even-sized axes). The simulation continues until some fixed percentage of cells have been filled, or until some other condition (such as the presence of trapped regions) is achieved. The fractal structure of the filled and unfilled regions is then examined to determine how much oil could be recovered.

Histogram Thresholding This module performs histogram thresholding on an image. Given an integer image I and a target percentage p, it constructs a binary image B such that $B_{i,j}$ is set if no more than p percent of the pixels in I are brighter than $I_{i,j}$. The input data are therefore the matrix I and the percentage p. The thresholding must be applied to a stream of input images.



[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 P. Dazzi (via email).

Neural network Given a *multilayer perceptron* (see Figure) and a set of input data sets, compute in parallel the output of the newtwork. The input sets are vectors $X = \langle x_1, \ldots, x_n \rangle$ where n is the number of elements in the neuron layer. The output of each neuron i is computed as

$$\begin{array}{rcl} net_i(X) & = & \Sigma_j (w_{ij} x_j + w_{i0}) \\ out_i(X) & = & \frac{1}{1+e^{-a \times net_i(X)}} \end{array}$$

where the weights w_{ij} represent the weight (floating point, $w_{ij} \in [-1, 1]$) of the arc leading to the neuron i from neuron j in the previous layer (the values w_{i0} are further constants), and a is a constant (assume $a = 1$).

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

[5]`http://didawiki.cli.di.unipi.it/doku.php/magistraleinformaticanetworking/spm/spm1213proj`

- 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.

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 `ottavinareale.di.unipi.it`.

Project validity

The project described in this document is valid for all the exam terms in Academic Year 2012-2013, that is from June 2013 to February 2014. 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 2013-2014 course. The start of next year course "resets" any pending project work.