Project Arimaa – CR7 : Friday, October 24th

Presents : Everyone, Christian Raymond with Nikos Parlavantzas

Order of business :

1. C++ VS Multithreading
2. Architecture and solution software
3. Remarks / Questions about the report
4. General remarks for the writing of our futures reports
5. See who is the "rapporteur" of our project
6. Writing rules for next projects

Information :

- Comparison of 2 methods of parallelization : PC VS Thread. With MPI, it is simple to implement, efficient and fully optimized. Both of them are using sockets, the difference is MPI doesn't show the complicated part. MPI is simple to use on a simple implementation. With MPI, we can even keep the last esqueleton code if we change the main program. We will choose MPI. We just need to find some references to justify our arguments.
- Comparison with GPU/CPU. Not all Grid'5000 computers possesses GPU processors (in Lille, Grenoble, but not Rennes). It is very asked, we won't be able to use them on a long period of time. We will begin with the CPU, and then change to GPU. The parallelization methods can be used for the two, but at least, we need to choose one of them. Neither Mr Raymond or Mr Parlavantzas have use GPU (NVIDIA Tesla GPU card) on Grid'5000 beforehand. There is a lack of documentation and it will require a lot of effort to make. Furthermore, we need to reserve Grid'5000 machines a time before using it.
- Boost C++ VS OpenMP. The only problem of C++ is C++ delete threads during the process and not after, so it increases the Software time. OpenMP is better, we can personalize our threads. Performances are the same. C++ is more complicated due to the Boost Library to use (difficult to write recursion tree level). Paper conclusion : << Our basic conclusion is that although we could express the parallelism using either Pthreads or OpenMP, it was easier to express the parallelism at a higher level of abstraction. OpenMP allowed enough control to express the parallelism without exposing the implementation details. Also, due to the higher level specification of parallelism with OpenMP, the tools available to assist in the construction of correct and efficient programs provide more useful information than the equivalent tools available for hand threaded programs. >>
- OpenMP VS OpenACC. OpenMP is specialized for CPU and OpenACC for GPU but both of them can work in both of the processors. The two softwares were created to be merge, but they just merged partially, even though, it works perfectly fine. OpenACC is better with data management (in some examples, it takes 10 more time with OpenMP). An other possibility is the Cuda language. It's a low level solution (between assembly language and C language). The problem is the code is changing to much comparing to others languages that we know about. Cuda is complicated, only for GPU, and we don't see the point of getting it without advantages (created by Apple). Christian Raymond said OpenMP was simply better.
  We just need to find some references to justify our arguments.
- We may test the CPU code on our machines, because it should be able to work on a simple machine. MPI is able to direct tasks on different machines, and OpenMP is able to assign tasks to different core of a machine.
- Mail sent to Ivan Leplumey (23th) and to Eric Anquetil(24th) about the "rapporteur". We assume that is Ivan Leplumey, but to be sure, Dan sent it to both of them.
- We talked about creating a Software able to handle other games than Arimaa. We will use abstrait classes to implement, we will separe the display from the rest, Our code will give us a new move only for instance.
- We have talked about a writing rules file (see in the folder help). We can list all things we need to respect about writing rules. Everyone can add something. The next question to

discuss is about WriteLatex + Git VS Compilateur + Git (use it only to write the report).

- We can begin to write part of our next report during the holidays (The first draft is due the week after the INSALON). I'm talking about Baptiste, Benoit and Mikail, about software they have already chosen. Don't forget to look for good references on the Internet. Next report is due to 11/19 (first draft) and 11/26 (last draft), to give the final document the 11/27. Be careful, we don't have a Project's week to work on it!
- The assignement of work is on the planification. Prateek, Dan and Gabriel have to work together on some points. Pascal Garcia uses a language for coding game rules. Maybe we can use it, or make a C++ library/package. We need to complete the tasks' schedule after Baptiste will send us an email to warn us. Priority is to write or to prepare the next report (Sofware architecture and solution is the heart of this report).
- The next report will be composed of the global Architecture for our Software, diagrams describing it, sofware we will use with reasons. This will be the basic structure of our program. We will go deeper in this Architecture in the December report, so we need to stay general. If we have time, we will be able to compare mutexes, parallelizations, strategies. Dan will think about the plan of the next report

Planification

Task	Responsible	Deadline
Schedule time	Baptiste	10/27
Note schedule	Everyone	11/05
Parallelization method implementation	Mikail	11/05
General Software architecture	Gabriel	11/05
Input/Output of the program	Dan	11/05
Prepare the plan of the report	Dan	11/05
API of the program	Prateek	11/05
MCTS simple implementation	Benoit	11/05
Packages with rules implementation	Baptiste	11/05
OpenMP implementation	Not decided	Begin after 11/05
OpenMP report part	Benoit	11/19
MPI report part	Mikail	11/19
OpenAcc report part	Baptiste	11/19
First draft of the report	Everyone	11/19
Last draft of the report	Everyone	11/26
Due date for the report	Dan	11/27
Finish the game application	Gabriel	12/18

Details of the tasks

Detailed Task	Responsible
Parallelization method (method implement on the PC : tree, leaf, roots, others to implement)	Mikail
Software architecture (work on I/O with Dan and API with Prateek)	
It is about program structures, general UML (containing task rules) , diagrams work	Gabriel
Determine I/O data (use table data format json? , handle generalization of the game, using dll library ?	
How handling events, board screen, clics (the MCTS program will calculate positions)	
Global interactions of the user with the program	Dan
API (Graphic interface to choose, command lines optional ?, move with mouse or keyboard,	
work with Dan/Gabriel SFML	
who takes care of the screen	Prateek
MCTS simple (basic implementation on one computer, one thread for now	
In C++, we need to test it, on Tic Tac Toe for instance	Benoit
OpenMP implementation basic (OpenACC later)	
(require MCTS implementation/parallelization methods)	Not decided
Packages with rules (a dll library ?), what to do with the screen display (Dan) How to write rules	
(Pascal Garcia language can be used), an abstract class implemented by Arimaa methods	
and later other games (examples of possible methods : MakeMove, List of available moves	Baptiste
Go deeper on OpenMP (references) CPU parallelization	Benoit
Go deeper on MPI (references) computers parallelization	Mikail
Go deeper on OpenAcc (references) GPU parallezation	Baptiste

Next meeting : 11/05