

MIT OpenCourseWare
<http://ocw.mit.edu>

Multicore Programming Primer, IAP 2007

Please use the following citation format:

Rabbah, Rodric and Saman Amarasinghe, *Multicore Programming Primer, IAP 2007*. (Massachusetts Institute of Technology: MIT OpenCourseWare).
<http://ocw.mit.edu> (accessed MM DD, YYYY). License: Creative Commons Attribution-Noncommercial-Share Alike.

Note: Please use the actual date you accessed this material in your citation.

For more information about citing these materials or our Terms of Use, visit:
<http://ocw.mit.edu/terms>

MIT OpenCourseWare
<http://ocw.mit.edu>

Multicore Programming Primer, IAP 2007
Transcript – Lecture 1b

The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

PROFESSOR: So I will go into a little about what the class is, and what you're going to do. I'll just put one slide about requirements and outcomes.

So what are the requirements? We require you to be a good programmer with a lot of programming experience. As I said, we are going down to very bare metal. And there won't be any niceties of just writing Java code. So you're going to run into a lot of these kind of issues. Fluent in C is -- I wouldn't call it a 100% requirement, but it really helps if you're fluent in C. If not, you'll have to probably spend a lot of time trying to understand C, C++. And in this class that means -- just start doing it today, just get a C book and then try to understand. If you run a lot of Java it might be an easy transition. But get to the transition fast, don't wait.

What are we expecting at the end? We expect at the end for you to know --

AUDIENCE: I have to go. Thank you very much [INAUDIBLE]

PROFESSOR: So what do we expect? We expect at the end you'll probably understand a lot of fundamentals and concepts of parallel programming -- both hardware and software. Because the nice thing about -- or the nice thing or the bad thing about Cell -- is a lot of hardware concepts to expose. There's no nice abstraction layers in there.

You'll understand a lot of issues about parallel performance. We started here, you write a program, compile -- it's not just order n you're looking for. You're looking for all these constants. All these things matter, basically.

You should be able to synthesize a fairly complex parallel program by the end, because that's what your group project's going to be.

And you'll get a lot of hands-on experience with the Cell processor, so we'll understand this one instance very well. So that is what we are expecting will happen at the end of this 3.5 weeks.

So the project. You proposed the projects, we did something different. So it's all your ideas that we are going to do. We selected seven groups out of these -- there are seven projects in there -- mainly based on the strength of the project proposals. And I was pretty impressed with the height of things you guys proposed.

So here are the seven projects. Let's actually try to figure out who's who. Who's doing distributed real-time ray tracer? OK, that's not true. OK, how about the global

illumination? OK. Linear algebra pack? OK, good. Molecular dynamics? OK. Speech synthesizer? OK. Soft radio? OK. Backgammon tutor? OK.

And there are some people who are, I think, in this class thinking that they can join a group. So at the end, there are a few groups that only have one member, so probably go ahead and talk to them and see whether they are willing to work with people. So I'll give you a little bit of time, because I'm going to finish early.

OK, let's ask -- who are the people who are hoping to join a group? Is there anybody who came? I got a few emails, I don't know that anybody's here. Oh good. So it's not going to be a problem.

So I did the project characteristics. You can see that they are ambitious projects, but accomplishable. I was worried that people would say, "I'm going to build this entire thing", and it's going to take a year. But I think most of the projects that they proposed are doable.

Another thing is important and relevant things. I think for a lot of these things, if the code is done right other people might even use it. I think this can start something, it's not just going to be this project.

And there's a lot of opportunity to sizzle. I think some people will come up with these things that I think at the end will be so amazing. And that's what got me excited. But get them started as soon as possible, because this is not a semester course, this is going to run so fast. There's only a few all nighters you can pull, basically is what it is.

So there's a note of caution. Cell processor is very new. We heard about that today. I mean this is a really cool thing, but it has its problems in there. It's not an easy architecture to work with. There is though this nice abstraction. So it's just then from this being completely abstracted out, writing Java code, to all the way to metal. So it's just almost a jump from going to a higher language of assembly. So there's a huge, big gap in there, and that's going to be a lot of issues in there.

The tool chain is very thin and brittle. So my pointing out all those things, you wish there's a tool, but there's nothing. And even the existing tools -- we just got the SDK 2.0 into working I think this morning -- and so there might be issues in there.

Most of the staff have limited experience. For example, SDK 2.0. We just got it working, so we haven't used it yet in there. So that can create problems, especially in a fast moving course.

The projects you are doing are of your own making. Most of the classes -- if you need to [UNINTELLIGIBLE] classes, we actually come up with the projects, TAs go and implement it once, we know all the gotchas. And then when you get the project we know there's an existence proof that it's doable, it works, the tools work. We know at least there's one path through it. There's nothing like that here. Everybody's kind of just jumping at it head on, and hoping that we can read them. And we might find issues that are very hard, and that's why we actually have very close contact with those guys at IBM. So if things happen, we'll call them and they will be available to help us. But it's not something you can say -- oh yeah, I know, just go do that. Or -- here's the way out. So there might be issues like that, so we have to be very careful of that.

And you will face these kinds of problems. That's more given than anything else, so be ready for that. This is not just writing some high level code and getting away with it. And you're all in this together. So the staff, you, me, everybody -- we're just doing it for the first time in some sense. And that's the fun part about it, because as you pointed out, there might be some stuff that you guys come out with that will become more like folklore and the basis for the things people do. Because you might say -- hey, there's a neat way of doing that. You will be the first to do this kind of thing.

So there's that time, but on planning the three week course it's a large chunk to bite. If it is more for a PhD, then you can say -- OK, spend three months figuring it out. There's no other time limit. So we will try to manage this process, but you need to try to -- expectations of what you are trying to get done, and then be able to get it done. So you have to always balance this out. If you're running into issues, get to us very fast. Because we need to figure it out. And we might have to actually also figure it out from somebody else, so we can escalate it up and get the information. Because three weeks are going to go so fast.

How many people are taking the class for a grade? So people who are taking it for a grade, here is what we are going to do. The first thing, what I call mini quizzes, is just a prop so you will come into class. So the way I am going to do that is, at the beginning of the class I am going to give you a piece of paper with two questions from the previous lecture. You can do anything you want to answer it. You can talk to people, look it up, whatever. And then at the end of the class basically we want it back. And the only requirement is, you have to come individually to pick your thing up, and you have to come individually to hand it in, and you can't write somebody else's answers physically, but you can talk to each other. So we'll try to make it a little bit entertaining, and a little bit thought-provoking, but nothing like you have to spend hours doing it. It'll be probably five minutes. Normally we will do a break inbetween, so you probably can talk to each other and come up with some of the answers. So this is just the fact that the people who are actually getting the grades will be coming to class, more than anything else I'll be testing.

And then we have these lab projects at the beginning, that will get some grade. And the final group project.

And then we have a final competition. So since everybody's not doing the same project, we are going to decide on performance -- so how well your code ran, how much MIPS or [? GOPs ?] you were able to get from that. And how raw performance is in there.

And then we will also look at it from a complexity point of view. So for example, as I pointed out, if you do matrix multiplier you get certain performance, so if you do [? FF3, ?] you get certain performance. So just absolute numbers is not what matters, but how far you can get hard algorithm too.

Completeness. So were you able to get your project through? How much do you actually have a complete solution at the end that something works? That's important.

Algorithmic complexity -- how much of a complexity did you bite in, both from more of a pure algorithm point of view, and also implementing some shared complexity in there.

And demo and presentation. So OK, those two.

So we are going to select a winning team. We haven't figured exactly what, but we'll give everybody a gift set depending on -- this is what we are aiming at right now. Also we are going to invite the winning team to spend a day at IBM TJ Watson. You get to give a presentation to a bunch of people there, and probably go see what they are doing. That should be a fun day to spend. We haven't scheduled a day, depending on the class schedule and stuff like that, the winning team will go spend a day out in there. That will be a fun thing to do, there's a lot of cool stuff going on there. They will get to actually interact with some researchers, see what they do, as well as present this work to them. And since this is new everybody's excited. I think hopefully we'll come up with a team that will do things that other people will be interested in and will use.

So these kind of come from backward because I wanted to get Mike Perrone first. So my name is Saman -- Saman Amarasinghe -- I live in CSAIL. I have had a lot of interest in --

[LAUGHTER]

Not as much as you guys probably. I have a lot of interest in languages, compilers, and computer architecture. So if you have been at this game for a long time, in the early 1990s there was a huge surge in parallel performance, work on parallelism, and at that time I did the SUIF parallelizing compiler in there. And then, everybody says -- hey, there's this Moore slope of performance, who cares about parallelism? So there was this decade when this parallelism kind of died down. I kind of hid under the radar and kind of survived through that thing, and then again I think parallelism coming up. So we are doing a bunch of parallel work in here. So I probably have a perspective that goes somewhere old, and then coming up. Probably Aaron can also relate to that.

Aaron: There's one of us who did that.

PROFESSOR: Yeah exactly. Going through that, kind of survived in that.

And Rodric is sitting there. So Rodric is right now at IBM PJ Watson, but he's probably going to live at CSAIL next month, so he's going to be here more than me in fact. He was actually in the lab until a few months ago, as a research scientist. He's also interested in a lot of these kinds of things -- compilers, architecture, parallel architecture, and FPGA type work.

And then we have two TAs. Where's David and Phil? OK, get up. So they actually have some experience. Probably they are about a week ahead of you on understanding Cell, not too much. They are going to lead the recitations and be available to help you in there. So really use those resources as much as possible.

And then we are going to have a huge amount of guest lecturers, and I was pretty happy with the kind of list we got. Mike Perrone just came and gave you the lecture, and Alan is going to talk in a few weeks. And Professor Arvind, Brad Kuszmaul, Mike Acton -- that Mike Perrone talk about is going to come at the end. And Bill Thies, who was one of the PhD students in the Streamit group, is going to talk about the Streamit language.

So the way we have kind of structured the class is, how do you go about extracting parallelism. There's implicit parallelism. And sometimes the current way of everything done by hardware. In the next lecture, I will just touch upon superscalar hardware, what it is and what its problems are.

Implicit parallelism means you can do by compiler. So you write a C program, and let the compiler deal with the parallelism. For as long as you are a programmer, you don't have to deal with that thing. In lectures 11 and 12 I will talk about parallelizing compilers. So we'll cover how you can do that [? in C. ?] So that means kind of abstract it out, and leaving it out of the end programmer. So the hardware exposes it, but we put a software layer.

And then the other end is explicit parallelism. That's where, I think, we are going to spend most of the time in this class. So in the first part of explicit parallelism is the library approach. What that means is you are using MPI type things, so you are basically at the metal dealing with this parallelism.

I am going to do lecture 4 on concurrency. How many people took 6.170 last year when I was teaching? OK. I did a lecture on concurrency there, so something very similar I'm going to do in here -- to get the basics of parallelism. In order to write a parallel program, the program has to run concurrently. There's a lot of issues for writing concurrent programs. It is necessary to get parallel, it's not sufficient. There are a lot more other things to worry about. But let's deal with those kind of issues -- deal with things like synchronization, deadlocks, and issues like that.

So I'm going to give that, and then Rodric is going to have a few lecture on design patterns. So if you want to write a parallel program, how do you look at it? Different algorithms will have different ways of parallelizing. So he's going to go about looking at these kind of parallel patterns. If you have this kind of program, here is a way to look at it. Here's another way to look at it. So kind of go through that in there.

And in concurrency we work with a lot more abstract parallel model, and get slowly down to all the realism needed in dealing with Cell.

And then we are going to have a bunch of guest lectures, mainly dealing with other ways of doing parallelism. So this is one way. We are going to have a lecture on Streamit, because you might be able to use that also as a way on Cell. So a little bit of high abstraction, we are trying to make that available on Cell. So if that comes to, then you might be able to even have that as a part of your toolbox in there.

The other three talks are about very different approaches for parallelism. I think they will give you a big picture view, it's not that you're just going through one parallelling down in there. I think at the end of to the day you'll understand not only how to deal with Cell, but kind of a general way of looking at parallelism, and maybe from that approach [UNINTELLIGIBLE] parallelism in there.

So the lectures are organized as follows. We are now at the end of the day here. Tomorrow we'll have a recitation on getting to know Cell. We are trying to figure out a good room, so just check your email to figure out exactly where we are going to have it, because we are trying to figure it out. We want to find a room with machines so everybody can actually get hands-on experience in kind of a lecture format.

And then we will have the architecture, just general parallel architecture, and an introduction to concurrent programming. And then Rodric is going to have individual meets with individual groups, trying to get the groups' projects going. So at that point kind of figure out how we are doing it, have some early ideas about it so you can start thinking and start the process of doing that. You can start working on Cell probably from tomorrow, but really get a day to kind of get familiarized, and then here start the project.

And then we will have some lectures on design patterns in here. And as we go we'll have more hands-on experience also.

Then we'll have a lecture on Streamit. So if you are using Streamit for a part of the run program, you'll be able to understand what's going on. And we'll get to some debugging tools, hopefully we will have to deal with these things before that, but here's something that if you're really stuck, to get a breadth of tools available in here.

And then we'll talk in general about debugging and performance monitoring and performance optimization. So at this point hopefully you have done the first part, you have figured out what problems, and start optimizing things in here.

And then we'll going into a little bit more breadth, things like traffic parallelizing compilers that I talk about, getting implicit parallelism in here.

And then we will have a bunch of guest lecturers coming in.

And we'll talk about some hardware and the future, because we are just starting this very new phase in programming that might change almost everybody. Five years, ten years down the line if you go through computer science, you might have a very different flavor, if what people think of parallelism takes hold. So let's look at some future, and talk about that. And it's a nice way to finish the lectures.

A few more days and then we'll have group presentations in here. So each group will get about 15 minutes time to present what they have done and give a little bit of a demo. And then finally on Friday we will have awards and reception.

So this is what we got. Any questions?

AUDIENCE: For recitations, if we're not in a room with machines should we bring laptops?

PROFESSOR: So that's a good question. Who doesn't have a laptop they can bring? OK. If that is the case, we can have it in any room we want, and have the network. And we might have one or two additional machines for people who don't have a laptop. And that actually will work much better than everybody marching into some kind of a computer room that's not structured properly for doing a class. OK, that actually helps a lot. That's a good point, I think that helps a lot. OK, we'll do that that way then.

AUDIENCE: How do we get access to these machines?

PROFESSOR: OK. Rodric, you --

RODRIC RABBAH: Every room has a PS3 dedicated to them. And every group has accounts on their dedicated PS3. I have your usernames and passwords. I will pass them out tomorrow, because I'm going to propagate the new SDK to all the PS3's -- to sort of test it with one and make sure everything's working. So you'll get them at the first recitation tomorrow. Unless you really want them today, then come talk to me or send me email.

PROFESSOR: You'll get to take home --

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yes.

RODRIC RABBAH: Yes.

PROFESSOR: We couldn't even put it in the machine room, because they were scared somebody was going to take them home. So we actually have it in a different room -- closed, nicely locked -- or on the network. So I have been a very popular person lately, a lot of people have tried to bribe me to get access to one of these masters.

AUDIENCE: [UNINTELLIGIBLE]

AUDIENCE: How much does the PS3 cost?

PROFESSOR: Officially \$600, but then eBay's going up to like \$2,500 these days.

AUDIENCE: No.

PROFESSOR: That's gone down?

AUDIENCE: They're down between \$400 and \$500.

PROFESSOR: Oh, OK.

AUDIENCE: They tried [? to rope me in. ?]

[LAUGHTER]

PROFESSOR: OK. But during Christmas it was down for -- a few desperate payers who were trying to bribe me.

[LAUGHTER]

So I guess their volume has caught up to --

AUDIENCE: No, the demand has dropped.

PROFESSOR: OK.

AUDIENCE: [INAUDIBLE]

AUDIENCE: Will we get to see the Playstation?

PROFESSOR: Tomorrow we can bring one --

AUDIENCE: I can bring one [INAUDIBLE]

PROFESSOR: If you put a game in it's fun. You can put a game in and play it, but this is going to be Linux so it's a little bit boring.

[LAUGHTER]

AUDIENCE: So these are things that actually cannot play the games.

PROFESSOR: It can. Interestingly, mostly these game machines are sold for less than production value. So Microsoft, Nintendo, and Sony before made sure that nobody can run anything other than the published games that they've charged \$25 per pop to recover the cost.

Huh? No, \$60 for the game. I think they are charging \$25 from the game maker just for the privilege of running the game. That's how they make their money. But, [UNINTELLIGIBLE] when they made PS3 bootable by Linux. So in fact, it's much cheaper, you get a Blu-Ray player, and this Cell processor for a lot less than the production cost.

AUDIENCE: Does Linux on the PS3 run on top of the PS3 OS though?

PROFESSOR: No. It just boots up PS3. What it does is in fact -- right now, you don't get access to the graphics processor properly. So you can run open G11. You have framebuffer access, but you don't get the full graphics access out of that.

AUDIENCE: Which distribution are we using?

PROFESSOR: Which one?

AUDIENCE: We're using Yellow Dog.

AUDIENCE: So in other words, Mario and Luigi are subsidizing what we're going to do.

[LAUGHTER]

PROFESSOR: Yeah, exactly. All those shoot them up games are basically what subsidizes. Actually he said, in one sense, you get the processor down to \$200 because they are selling all the other things. And the other thing, actually if you want to build a high performance Cell farm, it's probably cheaper because I think that the Cell blend would be \$2,000. It should be cheaper to buy a bunch of PS3's, connect them together. But there might be issues about things like network access, file access, because as you pointed out, the board access is pretty primitive. Because for games you don't need this high bandwidth connection, you just need to download it once. So there might be issues using it for cluster. OK. Anything else? OK, good.