

# AMDDEVELOPER INSIDE TRACK

---

COMMUNITYONE 2009 WEST: TAKING ADVANTAGE OF MULTI-CORE



**Patrick Leonard**, VP Product Strategy for Rogue Wave Software

**David Maples**, VP of North America at Allinea software

**Jim Falgout**, Chief Technologist of the DataRush Group, Pervasive

**Tracy Carver**, Software Developer and Technical Evangelist AMD

This panel discusses the recommendations on how to take advantage of multi-core systems.

---

Sharon: So we're here at CommunityOne with Patrick. Patrick why don't you introduce yourself?

Patrick: Alright, my name is Patrick Leonard and I'm vice president of strategy for Rogue Wave Software.

Sharon: So we're here talking about multi-core processors and taking advantage of them. What are the top things that you would recommend to customers that are interested in taking advantage of multi-core?

Patrick:

It's a big topic now. A lot of people are talking about it with all the new hardware coming out. It's something that software developers have to think a lot more about than they have in the past. I guess the first thing I would recommend is to really think about what's the nature of the parallelism that you're trying to achieve. There are different levels of granularity and different solutions that go along with those different approaches. So, for example, if you're looking at something lower level like making a loop parallel, there are things that compilers can offer. There are libraries that can help you out a lot. And then if you have larger components like services, there are also frameworks and runtimes that can make that much easier and make the manageability and maintainability of those things a lot easier as well. So, it's really important to really think through what you are trying to achieve. Is this a new application that's being built or are you repurposing existing code? All those things will help guide you toward the right solution.

Sharon:

Great. And I think you know, we talked a lot in the panel about the ideal thing is to design in the threading capability into your overall program algorithm, but if you can't do that, if you're taking old code that you need to migrate, single threaded code that you need to get beefed up to take advantage of all these new cores on die, what would you recommend for that?

Patrick:

Right, right. It would be nice if we could just start over and rewrite things. That would be a lot of fun, but most of the time we have existing applications and then you have to make those work and bring those forward. And really, kind of, the same thing. I think it's really important to look at the architecture your existing application in total, not just at the parallelism. Again, one of the approaches that we like for larger applications especially is to look at if those can be broken up into components or services. Then you can run those in parallel and take advantage of multi-core which can be multithreaded, it can be multi-process, it can be multi-server in a distributed environment and that can be a really effective way of getting the performance gains from multi-core hardware or blade racks in distributed environments without having to rewrite the applications to be multithreaded at the lowest level.

*Next scene:*

David:

Hi I'm Dave Maples, I'm Vice President of North America for Allinea Software. We make profiling and debugging tools for parallel and threaded environments and also we support the scalar developers as well.

Sharon:

Great. So, we're going to ask you the same question that we ask Patrick which is: What's the number one thing developers should do when looking at multi-core adoption?

David:

Well, I think the big challenge that everybody is facing today is multi-core technology in its own right. The challenge is, number 1, we are getting more and more cores available for developers to start writing applications to. The applications have started off as scalar applications in many cases or very little level parallel. Now they need to have tools to allow the regular developers to be able to develop in parallel and threaded environments easily. So what we've done is focus on the scalability which makes things be able to scale to very very large number of processes. That fits the high performance computing market. Then you have the bin market which is probably where most of the industrial work is done. Then, of course, we focus on the small markets which are people that are developing applications which are particularly scalar. Now they need to be able to take advantage of multi-core and are going to need to parallelize these applications and be able to take some sort of threading or parallelization and put that into their code. The issue is that people have been used to one way of doing business. This is the first time in 30 years that they've had to change the way we program. So, our tools are ones for parallel and threaded developers to be able to debug your code and we have another set of tools that are focused on being able to optimize your code and actually make it run fast. And, now with the other people in the panel here, they have tools that are able to focus on people doing the development of the application prior to the compiler which requires the debugging. So we fit a little bit further down the stream, but very, very critical as people start moving into really taking advantage of Multicore technologies.

Sharon:

That's a great point. A lot of times we talk about getting started on multi-core adoption, but not often do we talk about once you have Multicore. There's obviously going to be a debugging process and that kind of thing, and the ease of use of tools is also very important and key as we get more and more cores on die everyone's going to have to be multithreading their code at some point.

David:

Absolutely, and you know, the reason is that people, our clients' clients, already have multi-core and the clients are demanding to be able to use the technologies that they already have. That they already own. Which means distributing applications in one way or another, whether it's a threaded environment, whether it's parallel, or whether it's SOA's or whether its some sort of VM world. So, those are really the ways we are going to take advantage of the technology. And that's the big challenge, because now the, as much as we're very pleased to have all this hardware and this capability, we've really thrown the problem of how to use that hardware, use those chips, has been thrown over to the developers. So, there's a whole new group of tools that are now necessary to be able to do this and there's a migration in thinking. And to do this quickly, that's where the ease of use comes in, because we don't have 10 years to re-learn this. We've got a year or two to be able to be successful and that means things are going to have to be quickly, easily, with as many automated features as we can give them.

*Next scene:*

Jim:

Hi, my name's Jim Falgout. I work with Pervasive Software. I'm the Chief Technologist of the DataRush Group.

Sharon:

So, what's the number one thing developers should do when looking at multi-core adoption?

Jim:

Well, really, they should think about what domain they live in. That will have a big affect on what type of solution they look at. There is no one answer for multi-core in the software arena. So, they have to look at what it is that they are trying to do and they may have to take a mixed mode approach, right. They may have to use tools from several vendors to mix and match to get to what it is, the solution that they want to get to. And one thing they should really consider, which isn't really thought about a lot... a lot of times when we think about multi-core we think about scalability. We think about software, how's it going to take advantage of it, and we think performance – runtime performance. How can we make it run faster. We should also think about design time performance. At Pervasive, we think design time performance is as equally important as run time because if you have a tool or you have a language, or whatever it is that you are using, to help you take advantage of multicore, if it's too hard to use, or too hard to learn you won't get adoption within your organization and then it won't really work. You have to think about design time. It's extremely important. It gets adoptability it gets people using it familiar with it, without people having to spend months and to learn new technology.

Sharon:

I think that's a great point. Especially when, during these times, we are so resource constrained around just about every corporation these days. You know, ease of use of a tool is huge. How would you suggest a developer go out and gauge the ease of use of the different tools that are available?

Jim:

It's really going to depend on kind of their expertise level, what things they are familiar with. For DataRush, we pick Java as our implementation language because we felt like with the wide adoption of Java throughout the business world that it would be easy for developers to pick up DataRush and utilize its API's which are all developed in Java and so part of it is going to be, again, looking at what it is that they are trying to do and looking at what domain they are trying to live in. and then picking a tool that will fit that seamlessly for them. IF they are Java developers, something like DataRush, they can walk in and within a week or so, be up to speed and be writing code that's scalable, that's high performance, that can crunch through massive amounts of data. So, it's very important to find that tool that's going to fit their domain and fit their problem that they're trying to solve, and help them just seamlessly walk in and start building parallel code

Sharon:

Great.

*Next scene:*

Tracy:

I'm Tracy Carver with AMD. I'm a software engineer in our software performance organization. I function often as a technical evangelist for AMD.

Sharon:

What's the number one thing that developers should do when looking at a multi-core platform?

Tracy:

They should begin first by looking at their data and understanding what dependencies their data has. They should also begin by looking at their data and understand where their data resides. Understanding where their data resides, for example, is it on multiple disks, is it on a single disk? Or understanding your data dependencies for example, does this one row of data depend on another row of data? When you look at those two things - that will give you a much better understanding of the inherent parallelism that might be possible when processing your data. Once you've done that, you'll get a much better feel for whether or not your ultimate solution is going to be one of typically two types. There might be a scale out, scale up approach meaning scaling up a solution to take advantage of Multicore on a single system. Other problems lend themselves more naturally to a scale out. Meaning, you're taking advantage of multiple cores, but you're doing it across multiple different systems that don't share a memory address space. Finally, the other thing you should do is understand your problem domain and take advantage of your community of experts. Talk to your colleagues and find out what they are doing to solve their parallelization problems. And finally, just remember that chip makers like AMD are continuing to add and increase end user value by increasing core counts over time and continuing to deliver additional cores to the market and we want software developers and end users to take advantage of them.

Sharon:

Great. I think that's a very succinct answer and developers have a lot to do to take advantage of multi-core programming but a lot of advantages to gain as well. Thanks so much Tracy. The slides will be available from CommunityOne, online, at [developer.amd.com](http://developer.amd.com).