



Announcer You are listening to the Sun Microsystems Podcast Network. Welcome to another edition of *Innovating@Sun* with your host, Hal Stern. Today's topic, *Project Darkstar*. And now, here's Hal Stern.

Hal:

Hello, and welcome to another episode of Innovating@Sun. I'm your host, Hal Stern, Vice President of Global Systems Engineering, and today, to talk about the fate of online gaming, and to make obscure Grateful Dead references, I'm pleased to welcome Jim Waldo, distinguished engineer and principal investigator in Sun Labs. And one of Jim's jobs right now is to be the chief architect of Project Darkstar. So, Jim, welcome to the show, and tell us a little about what Darkstar is.

Jim:

Well, thank you for putting up with me. Darkstar is a project currently going on in Sun Labs, and the goal of Darkstar is to provide an infrastructure that lets people write massive multiplayer online games - very large games - that can scale dynamically, while not having to change the programming model that they are used to in writing single-user, single-person, single-machine games. We're trying to hide the multi-threading and the distribution that we're using to scale the games up behind a fairly simple, single-threaded programming model, and do some magic in the background and just have it all work so that it will scale up to the hundreds of thousands or millions of users that are currently playing these kinds of games.

Hal:

I think you hit on, possibly, the really clever point here is that parallel programming is hard. We've been talking about it for decades now. It's a hard thing, so the more you make it look like single-threaded programming, the more you make it, I would say, generally appealing.

Jim:

Well, I think, actually, parallel-programming is one of the real scandals of our industry. We haven't advanced the art in multi-threaded programming much beyond what it was 30, 35 years ago, when Tony Hoare first sort of taught us how to do some of the basics on this stuff. And now, as we're moving to chip multi-threading and parallel processing all over the place, we've got to figure out how to actually let this be done by people who aren't, say, Guy Steele, or James Gosling, and let the rest of us learn how to do some of this stuff. This is especially true in the games space, because game programmers have grown up in a world where they own an entire PC for the game, and they get to do whatever they like with it. If something goes wrong, well, that's okay, the user learns not to do that. But, when you start having shared servers that have to scale up, that kind of programming doesn't translate very well.

Hal:

So really, what we're bringing together here is a model of abstraction, combining what people are used to in the game development space, with really, how we see massively parallel - or massively scalable games being deployed, which is on large amounts of infrastructure.

Jim:

That's right. And, one of the more interesting problems that we've got is not just on large amounts of infrastructure, but on constantly varying amounts of infrastructure. When a game goes online, they don't know how many people are actually gonna sign up for it. They can try to figure it out, but they haven't been very good at figuring this out in the past, so, for example, when World of Warcraft went online, they quickly used up all of the capacity that they thought was gonna take 'em through the first year in about a month, and had to shut down subscriptions. That's a kind of a nasty thing to have to do, because the subscriptions are where you make the money. And just as bad are games that think they're going to be reasonably successful that turn out to be duds, and now you've allocated all this expensive server space and it's not doing anything. So, finding some way to expand and contract the number of machines that are being used for a particular game, without having to reprogram the game, is one of the other things we're trying to push on in the Darkstar infrastructure, the ability to let them do that.

Hal:

Great. So, I want to back up for a second, here. What's the secret sauce inside of Darkstar? As a gaming engine, where do you think the major innovations are?

Jim:

Well, we took the problem of trying to hide the parallel and distributed nature of the game and started off with the assumption that, in general, that can't be done. And Sun has been talking about that for a long time. Peter Deutsch's Eight Fallacies of Distributed Computing, the work I did with a bunch of lab people years ago that led to A Note on Distributed Computing - all of those argue that you really can't hide those things in the general case. So, what we had to figure out was what in this particular case would let us hide these aspects. What we've done is to build a system that is specifically tailored to event-based programming. So, the way games are generally structured is, you've got a very rich client - either a PC or a game console - and it's running lots of graphics and lots of presentations and stuff. And as you play the game, as you take an action in the game, a message is sent down to the server that generates a small sequence of actions which are the result of what you did. And then, a very abstracted model of the world is sent back to your machine where it is then rendered and shown to you on the game. What you have is a sequence of very short tasks that have to be done, generated from the actions of the player on a very hefty machine that can take care of most of the graphics and interaction. So, the servers are very interesting beasts, to begin with. What we've done is to build a mechanism for keeping the information around as the game progresses; for doing communications back-and-forth, both between the clients and the server, and between the clients themselves, because turns out that a lot of the interaction that goes on in these massive games is between the various players - they're talking to each other all the time. And then, we've combined those things with a scheduler and task system that will take these events and the tasks that are generated by them, and run those things reliably. We, under the covers, have a modified transactional system to make sure that either everything happens or nothing happens on that turn, and if nothing happens, it gets rescheduled to be run again - and then, very quickly, return back to the player of the game, so that we're optimized for very low latencies. The secret sauce that we've put in is that every one of these tasks is set up in a way that if it uses the infrastructure correctly, it can be run on any machine running the Darkstar stack and the game logic equally well. So, we can do dynamic load-balancing from machine to machine on these various tasks on the fly, as we see the load go up or down in various places.

Hal:

So, some very slick things there, in terms of paralleling execution and being able to dynamically allocate infrastructure for it, but the first thing that struck me as you were discussing this is, we're talking about latency...

Jim:
[overlap] That's right.

Hal:
...scalability, network IO, realtime and I assume now you're gonna tell me that the whole thing's written in Java.

Jim:
Well, yeah, it is... [overlap]

Hal:
[laughs] So, those are not the attributes that people, I would say, historically associate with an ideal candidate for Java. So, what'd you do differently?

Jim:
Well, first of all, the connection between the client and the server. The game player at home and the servers that they're working on - that's a very optimized protocol. It doesn't matter what's being run on that, that's gonna be small packets being sent very quickly. The Java side itself - we think Java is actually an advantage here between the dynamic compilation that can go on with Java and the very short tasks that we have running, we're getting very good performance out of this stuff - probably better than we can get from straight C programming. People who think that Java is slow haven't been using Java recently.

Hal:
Okay. And you don't run into problems with garbage collection or anything else that would impair your realtime capabilities?

Jim:
No, we don't, because we're not a hard realtime, we're a soft realtime. And we take advantage of the latest Java garbage collection technology in that much of this stuff that we generate as garbage is first generation garbage, so it's the stuff that gets collected very, very quickly without causing any pauses in the garbage collection. Now that Java has a generational garbage collector, it's really sort of optimized for this kind of access pattern.

Hal:
So, let's talk about what the state of the state is right now. Anybody who's interested can actually get access to the binary form of Project Darkstar at Projectdarkstar.com, but I understand that we're also gonna be open sourcing it, and I want to make the obscure Grateful Dead reference - you're our Grateful Dead fan, you know that Darkstar, as a Grateful Dead song, is one of those examples of a tremendous vehicle for endless improvisation. So, to me, it seems to fit the model we're trying to get to of people who want to go write games.

Jim:
Well, that's absolutely right. And so, we've been working on trying to build a community around the Darkstar stuff for over a year. We've had early access versions of the binary out for, now, about 15 months; we put out a new generation of Darkstar APIs about three months ago at the Games Developers Conference. And all of this is available at Projectdarkstar.com. What we have available now is the 0.9 version of the code. This is stuff that will run on a single node, so it doesn't give you all the scalability properties, although we're finding out that it scales remarkably well. We can get 5,000 - 6,000 players on a machine, depending on what the game is like - at least, we've had customers claiming that they can get that - with the 0.9 release. And all of that's freely available on the Projectdarkstar.com site.

Hal:

Give me a sense of where you see the market going. I mean, is this unique? Are there other players here who are publishing gaming platforms?

Jim:

Well, this is remarkable, I have only been working on the Darkstar project now for a little less than a year, and I'll tell you, I feel young again. It feels the way operating systems and systems research felt 25, 30 years ago, where people do stuff because it's fun and because it's cool and not because of legacy requirements and stuff like that. So, this is a very fun community, and we're already finding that we're building quite a community around Darkstar. There are attempts to try and tease out of the game development some common libraries that can be used for game development. Right now, when a game is developed, generally, it's developed from scratch, and they do everything. And there are some engines for things like graphics of physics or some of the AI creatures. But in terms of the massive multiplayer online gaming, most of the companies that are doing that have rolled their own, and what they're finding is that they don't scale particularly well. World of Warcraft has done a very good job, but they've done it by what's called geographic localization, where any particular region of the game is associated with a particular server. And then, they may have two or three copies of that region - things that are called shards - that can take overflow, but if you're in a different shard, you can't interact with the people that are in some other shard. Linden Labs is having many publicized problems with scaling the Second Life platform, because once again, they're game people, they're not scaling people. In terms of the server platform itself, IBM has made some noises about having something like this, but as far as we know, Darkstar is pretty much the only general-purpose game in town. Now, we're currently talking with a number of other gaming platform companies that do other parts of the platform. Darkstar is really a platform for the execution of the game, but there are other things like billing and statistical management of the economies of these games which are not things that we currently have in Darkstar, but we're talking with partners about integrating them in, because one of the things we did in Darkstar was to really build it around a service architecture that allows other modules to be placed in by other people, and we expect a community, then an ecosystem, to grow up around that, as well.

Hal:

Great. So, in terms of our infrastructure, where we see this going, what's going on in terms of making Darkstar, I would say, grid-enabled, or better optimized to run on large, homogeneous gridded machines?

Jim:

Well, right now, the research team is working entirely on the multinode version, getting it running on multinode. And we think that we are, oh, a month or two away from having that running in such a way that we can start showing it to other people. Once that is fleshed out a little more and we've done a little testing, we will put out an open source version of that, because we're not holding these things back, at all. Once we've done the multinode version, we are already in discussion with people in the Sun Grid and people who are wanting to build game centers outside of the Sun Grid, on how we can deploy this infrastructure in such a way that people could easily build up games. And we're beginning to do that a little bit ourselves; we have what we call a Playground, which is a set of machines that are running Darkstar that we're offering on an experimental basis to game developers, so that they can take their Darkstar-developed game and put it on our Playground machines and offer access over the Internet. And we're now in the process of informing those who we've selected to be on the Playground, and we'll get them up and running quite soon now. One of the advantages for us of having the Playground, is we will implement the infrastructure there and we can find out whether we've actually characterized the kind of load games put on the system accurately and if so, continue doing what we're doing, and if not, make changes in it.

Hal:

Great. So, what's your call to action for developers? What do you want developers to pick up, think about and go do, as a result of what we're making available through Darkstar.

Jim:

Well, right now, if they're interested in game development of these network games, go to Projectdarkstar.com, pick up the documentation - we have tutorials, we have Javadoc, we have example programs - and pick up the binary of the 0.9 release, because you can start building stuff with that. And it doesn't matter what your platform is - this is all in Java, that's another reason we did it this way. We have customers currently running on all of the operating systems you can think of. Take a look at it, start playing with it, get involved on the forums. The source code's gonna show up then start combing through there and seeing how we do this stuff. There are some things there that I think are reasonably straightforward, some others that I think are pretty clever. People can learn a lot about doing multi-threaded programming by reading this code. And that's going to help the game developers, as well. If you don't think we're doing the things that need to be done, that will make it easier for you to implement and deploy your game, get in touch with us, because we're listening on all the forums and talking to everybody there. The community is actually quite active and a lot of fun.

Hal:

Great. Well, Jim, I'd like to thank you for being our guest here on Innovating@Sun, and you've been listening to Jim Waldo, distinguish engineer in Sun Labs, talking about Sun's Project Darkstar gaming platform. And this has been another Innovating@Sun Podcast, with your host, Hal Stern.

Announcer You've been listening to Innovating@sun. Join us next time for the latest in innovation from Sun Microsystems - only on the Sun Microsystems Podcast Network.