



<http://www.dotnetrocks.com>



Carl Franklin

Carl Franklin and Richard Campbell interview experts to bring you insights into .NET technology and the state of software development. More than just a dry interview show, we have fun! Original Music! Prizes! Check out what you've been missing!



Richard Campbell

Text Transcript of Show #378
(Transcription services provided by [PWOP Productions](#))



Michael Isard on Dryad
September 18, 2008
Our Sponsors





Geoff Maciolek: The opinions and viewpoints expressed in .NET Rocks! are not necessarily those of its sponsors, or of Microsoft Corporation, its partners, or employees. .NET Rocks! is a production of Franklins.NET, which is solely responsible for its content. Franklins.NET - Training Developers to Work Smarter.

[Music]

Lawrence Ryan: Hey, Rock heads! Quit Ego-Surfing and listen up! It's time for another stellar episode of .NET Rocks! the Internet audio talk show for .NET developers, with Carl Franklin and Richard Campbell. This is Lawrence Ryan announcing show #378, with guest Michael Isard, recorded live, Monday, August 25, 2008. .NET Rocks! is brought to you by Franklins.NET - Training Developers to Work Smarter and now offering SharePoint 2007 video training with Sahil Malik on DVD, dnrTV style, order your copy now at www.franklins.net. Support is also provided by Telerik, combining the best in Windows Forms and ASP.NET controls with first class customer service, online at www.telerik.com, and by Data Dynamics, makers of ActiveReports.Net, simple, powerful and cost effective reporting for Windows Forms and ASP.NET web applications, online at www.datadynamics.com. Support is also provided by CoDe Magazine, the leading independent magazine for .NET developers, online at www.code-magazine.com. And now, the man known as the Larry King of .NET, Carl Franklin.

Carl Franklin: Thank you very much and welcome back to .NET Rocks! This is Carl Franklin in New London, Connecticut. Richard is still on vacation, he'll be back here next week and he's just getting back from climbing Mount Everest, if you can believe that. You know, we're in the middle of our contest, the .NET Rocks TechEd Europe Sweepstakes. What we're doing is we have a form on the website. If you go www.dotnetrocks.com/barcelona, answer a question about one of last week's shows, and this week's question is in show 376, what word does Charles Petzold use to describe Alan Turing? Is it a geek, a nerd, or a genius? If you listened to that show, you should probably know that. Just go ahead and enter that in. You might win a Tom Bihn Brain Bag, which is the most indestructible backpack known to man, and the winners of those every week we're going to choose one lucky winner to go to Barcelona for TechEd which is happening -- you know, here's the really cool thing, we're going to pick on October 20, we're going to pick the winner. TechEd Developer is happening November 10 to 14, so we're going to pick on October 20, but here's the cool thing, you can choose to go either this year or next year, that's the deal and we'll pay your airfare, your hotel, and give

you admission to the conference all for just listening to .NET Rocks! So there you go.

We're suspending Better Know a Framework and email until Richard gets back, but I do want to point out a couple of things. dnrTV, if you don't know what that is, dnrtv.com, it's .NET Rocks television, it's an hour long weekly screencast with lots and lots of people that you know. There are 122 of them in the archives. Recently Miguel Castro did a demo where he created a WCF service in a client and he did it using Visual Studio and we took a look at all the goo that Visual Studio wizards puts in the projects, and then he did it from scratch and shows you how easy it is just to do it from scratch; so that was good. What else have we got? Beth Massi did a great show, show 119 on XML Literals in VB.NET, a very popular series. Dan Simmons did two shows on the Entity Framework from Microsoft, Rocky Lhotka has been showing off his CSLA.NET 3.5, Mark Miller did the Science of Great UI and his part 2 will be up this week, and also Billy Hollis did a great demo of regular line of business application that uses WPF, Windows Presentation Foundation. You want to check it out, dnrtv.com. A lot of our listeners like to gather a group of people at lunch time once a week and watch dnrTV together over pizza; this is the *Lunch & Learn* phenomena that's been sweeping the country and the world. dnrtv.com, know it, love it, learn it.

Our guest today is Michael Isard who is a researcher at Microsoft Research in Silicon Valley. He received his *D.Phil* – now what is that, a doctorate of Philosophy?

Michael Isard: Right, it's PhD done around the other way.

Carl Franklin: Okay. In Computer Vision from the Oxford University Engineering Science Department in 1998. In 1999 he started work as a Researcher at the Compaq Systems Research Center in Palo Alto, and he has worked for Microsoft Research in Silicon Valley since 2002. Majority of his early research was in the field of visual tracking and sequential filtering, and he helped to introduce particle filters to the computer vision community with the Condensation algorithm. From the time he joined Compaq, his interests had been broadened to include distributed systems research and this is where he's spending the majority of his time at Microsoft. Welcome, Michael.

Michael Isard: Thanks.

Carl Franklin: Your current project that you're working on that everybody is very excited about is Dryad. Tell us about that.

Michael Isard: So the original purpose of the Dryad project is we were interested in trying to make it easier to program large clusters of computers so if you have anything up to a data center with thousands of computers down to a personal cluster of five or six machines, the idea is to try and take out the distributed systems pain and have the system deal with the distribution of the data and figuring out what to run where and the tolerance and let you just get on with programming it.

Carl Franklin: So this is scheduling as well as the basic parallelism?

Michael Isard: That's right.

Carl Franklin: Wow, this is interesting, Richard, and this is something that you've been interested in and I guess...

Richard Campbell: Well, we've done a few shows around this area to one degree or another.

Carl Franklin: Sure.

Richard Campbell: So are we talking about particular problems that this is best for or are you really talking about sort of transparent clustering that you don't have to write any code?

Michael Isard: It is transparent but only for particular problems so it's no magic.

[Laughter From All]

So if you have a program that works well as a batch computation, we concentrate on throughput, not latency. For example, the tolerance may take a minute for time outs to kick in and things. You're not going to be answering real-time queries with this.

Richard Campbell: Right.

Michael Isard: But if you have some problems that can be expressed to the batch computation, and in particular you need to be able to structure the program so that it starts out with some input data which you can think as immutable and then does some series of operations on that and the intermediate data that you produce and then produces a single output dataset, then that's the kind of problem we can address. So it doesn't work for streaming on internet streams either. It starts out with some fine input data and do a bunch of transformations that end up with an output.

Carl Franklin: So is this something that you would take maybe a large set of data and divide it up between a hundred or a thousand or whatever many computers and then have them all working on their

chunk of that data and reporting back. Is this the typical grid computing application?

Michael Isard: Yes until the point where you said reporting back. So yeah, you would start off by partitioning your dataset onto a bunch of different computers, but the way you can think of the computation is if you can draw a graph where you can think of each of the original partitions of the data as one of the vertices in this graph and then you can draw edges throughout the vertices which is where you're computation takes place. So each of those internal vertices can be getting data from several places that's been produced sort of upstream and then at the end it writes out to the final datasets at the end, but it's not kind of reporting back in the middle exactly.

Carl Franklin: Okay.

Michael Isard: You know, they can do aggregations of datasets, you can exchange data between. It's not like these things are acting completely isolated and then each doing their bed and reporting back. They get to communicate the outputs with each other so you can do more complicated processing.

Richard Campbell: Interesting; and ultimately that report back could result in additional processing so you can really get in sort of a parallel recursive process.

Michael Isard: Right, yeah. I mean, at the moment the way that we do recursion is we just unroll the loop but certainly we can, if somebody needs it, we can extend it to make that recursion based on some stopping criteria and, you know, if something is converged for example.

Carl Franklin: Interesting, and now there is also a LINQ aspect to this as well.

Michael Isard: That's right. So when we originally did the Dryad project, we were building the machinery to do this kind of distributed computation where you're taking very large datasets and transforming them but we didn't really know how we're going to program it and it turned out to be a pretty flexible system for doing a bunch of computations that we're interested in and the performance was fine but people weren't jumping up and down to use it based on the programming model. Then actually my colleague, Yuan Yu, came up with the idea of integrating it with LINQ and that's been a lot more successful. People really like that because LINQ, to my mind, is a great technology for programming with datasets and it works for us in a few ways. One is it's just a nice programming model for this kind of problem that we're interested in, and the other is that

the design of LINQ made it very easy for us to integrate with Dryad so that from the programmer's perspective, I can just write a LINQ program and debug it locally using Visual Studio the way you would any regular LINQ program, and then if they just say that actually now the dataset comes across the cluster, then the design of LINQ means that the system can transparently suddenly invoke Dryad and do all the distribution work using all of the right .NET reflection and everything to make it all happen pretty transparently.

Richard Campbell: I guess that is the part that I'm still to get my head around. I dig the idea that you're using LINQ, and I see here looking at that specifically PLINQ, to write the initial expression so my code is pretty straightforward, I'm just writing a LINQ statement...

Michael Isard: Right.

Richard Campbell: But just pushing a collection to a set of servers essentially, how does the data get partitioned out like that?

Michael Isard: So we assumed that you've started out by partitioning the dataset yourself.

Richard Campbell: Right.

Michael Isard: So this maybe because the output of some previous program that you wrote with DryadLINQ or it may just be that you got the data on some distributed storage system that was automatically partitioning or, you know, you could have manually partitioned it and then you tell DryadLINQ "here is the location of my partitioned dataset." Either you pointed out the URI in the storage system or you can manually make a list of the partitioned site, you know, these functions are on these computers with this filenames, that kind of thing. Then that becomes a LINQ data provider and if you don't tell the system anything about it, then it will assume that there's just some set of random records. I mean, you have to tell it the time for the records but you can add an annotation to tell it that it's ordered for example, that it's partitioned by a particular hash key and then the system can make use of that so it doesn't do unnecessary work to repartition it.

Carl Franklin: Getting back to the technology itself, Dryad, by the way, what is a Dryad? What is the significance of that name, D-R-Y-A-D?

Michael Isard: Dryad is the animating spirit of forced entries so you know, you have this Tree-like computation. It's actually a graph, but anyway, the Dryad is the thing that animated the Tree. That's why we pick it.

Carl Franklin: That's interesting. Using Dryad by itself, it does have a .NET interface. Is it easy to sort of hook this up to any particular class? Do you use attributes or what's the interface like?

Michael Isard: Well, for the Dryad system itself, it's actually all C++ interfaces. So DryadLINQ is a .NET interface but you go internally through the LINQ interfaces there.

Carl Franklin: Okay. So if you want to access these objects remotely directly, you have to use C++?

Michael Isard: I am not sure, I'm not sure...

Richard Campbell: I don't know why you'd ever do that.

Carl Franklin: You said you're using C++ interfaces to access objects.

Michael Isard: No, sorry. So the data elements just look like LINQ in collections and so there is no explicit Dryad interface for getting at those data elements. You just write LINQ Expressions on the collections...

Carl Franklin: Okay.

Michael Isard: And then the functions that you put in the LINQ Expressions get executed remotely.

Carl Franklin: So with Dryad, without LINQ, there isn't a way -- is there or isn't there, a way just to day I want this process to go to these hundred or so machines and is there a .NET interface for that?

Michael Isard: We do have refers around the C++ interface for that, but most people are not using that essentially because it is easier to program with LINQ.

Carl Franklin: I see.

Michael Isard: But yes, we can drop it down to the more loadable interfaces if people need that.

Carl Franklin: I see.

Richard Campbell: Yeah, I guess I'm still just trying to get my head around how all that data gets farmed out to all the servers. What's running on those machines that makes them part of this?

Michael Isard: Okay, so on each of those machines, there is a little daemon that is just in-charge of forking processes on Dryad's behalf and so when you run a DryadLINQ program, what happens is it starts out with the LINQ Expression so while the program is running, as the LINQ Expressions are

being executed, they're not really doing any processing. They are just building up an Expression Tree inside LINQ and then when you try to evaluate any of the outputs of that Expression Tree, at that point DryadLINQ kicks in and it takes the whole Expression Tree and it treats it a little bit like a database query plan so it does some static optimizations to either push select up and down and figure out what the whole computation is and then it writes out an XML description of that and then it pass it over to a Dryad executable that takes that XML description and forms it out to the cluster. So then you have this process running which we call a Job Manager which is a single process that's in-charge of doing all of the scheduling and figuring out what to run where and so that will communicate with these demons that live on the cluster and say "please run this process here with these inputs," and then if there it fails for example, it will rerun it somewhere else where it is running slowly, it will run to get it somewhere else, that kind of thing.

Carl Franklin: I see.

You're listening to .NET Rocks! from dotnetrocks.com. This is Carl. I have a message from our sponsor Telerik who wants you to know about the best way to learn using new dev tools and technologies. Well, is it reading manuals, watching videos, playing with sample code? How about all the above? So Telerik recently launched their new interactive trainer tool to help you effectively learn all the Telerik products in your own pace. The Telerik trainer is a sleek WPF app that combines a video player with synchronized highlights, a table of content for topical navigation, and a context sensitive code module. While playing the narrative video, you'll see a code button lied up at a relevant section, click the button and you'll open the respective file from the provided project directly in the Visual Studio. No more searching for code while watching a training video. This is indeed innovation in training. They're always releasing new tutorials for the Telerik products so don't waste any more time and download this amazing training tool now at www.telerik.com and as you know when it comes to developer tools, it's not just about great products, but also about reliable support and effective training materials and that's exactly what our friends at Telerik have done. Check it out. Now let's get back to the show.

Richard Campbell: The workload schedule here has got to be the real complicated part that you're dealing with, that disassembly and distribution than all of those executions, what's running well and what isn't, and what failed and what didn't, what needs to be retried and then trying to recompose that into something coherent.

Michael Isard: Right, so we do a few simplifications to make it manageable. So the DryadLINQ front-end decides on the granularity of what's going to run at particular vertices. So it breaks the LINQ Query down into subqueries and it uses some heuristics on it that you can override those with annotations. For example, if it can't figure out whether or not a particular subquery is memory intensive, you can tell it and so you can pipeline them together. So then it decides what the sub-Expressions, sub-LINQ Expressions are going to be in each of these vertices and then it passes those on to the Job Manager. So then the Job Manager just to have to decide how many copies of these things to run based on the size of the data and they have decide where to run them and when to reschedule if they're finally using things, but because of the way that we've restricted the computation, it's then a cyclic graph so all the edges in that graph host, talking about at the beginning, they're all going in the same direction where you can sort the graph from top to bottom. That simplifies the scheduling problem because, you know, even if you're only down to one working computer in the cluster, there is some sort of order for the graph and so if you just execute the vertices one after another in the right order, they will eventually complete. So really then the scheduler's job is just to pick a greedy algorithm to try and do better than that and try to use the available computers as well as possible trading off between putting the computations close to its input data and then also using machines that are idle...

Carl Franklin: So you're taking eventually resources that are available.

Michael Isard: Right.

Carl Franklin: So you're looking at CPU, you search and things like that as well and...

Michael Isard: Right. I mean, again, at the moment it's mostly fairly simplistic so unless the programmer is confident as to put on an annotation, then the system just assumes that it should only run on one of these vertices at a time on a given computer. It's not doing fancy sorts of things.

Richard Campbell: And then all you really do is measure how long each vertices take to give you a sense of what's fast and what's slow.

Michael Isard: Exactly, that's right.

Richard Campbell: Yeah, so you're not really having to dive deep into the machine to understand it's performance. You're just looking at I send you this piece of work, how long did it take.

Michael Isard: Right. So we do actually have other research Dryad on the side to monitor the whole

cluster and look at the machines and find anomalies. We had one actually the other week where one of the computers had a bad network card and so it was working, things were completing, all of the network Dryad was running, but tends to slow.

Richard Campbell: Right.

Michael Isard: So you know, by looking at the whole cluster across multiple jobs, you start to see those things popping out. That's kind of a separate monitoring tool on the side of Dryad. Dryad just tries to do the right thing locally for that job and it assumes that some cluster monitoring thing on the side is then going to retire those machines sooner or later and so it works around it in the short term but it doesn't go and do anything extreme to that computer.

Richard Campbell: Doesn't actually figure that out per se. It's very cool, I'm still getting my head around and I've actually dug into the website and found the sample applications that were there, some sort of basic ones just to get folks started. Can you talk to some of these?

Michael Isard: Let me just go and load that as well.

Richard Campbell: Yeah, I'm looking at the PDF.

Michael Isard: Right. So this is actually a companion to a paper that's going to appear in the MSDI conference in December.

Richard Campbell: Cool.

Michael Isard: We don't have the text of that paper out yet.

Richard Campbell: I had this far as 1.4, the F graph example. The nice thing about this is that there is not that much code

Michael Isard: Right.

Richard Campbell: You don't feel like you're parallelizing anything.

Michael Isard: That's right and really that LINQ that we're leveraging there, I mean I'm a big fan of LINQ and I can play all sorts of nice things about it because I didn't do any of it, but you know I think LINQ is just a great sort of interface and environment for doing parallel programming and you see people sometimes who look at it and they think that it's really a front-end for SQL and I think it's a lot more than that.

Carl Franklin: Sure.

Richard Campbell: Yeah, we definitely had conversations on the show where it seems to me that LINQ to SQL is the least of LINQ's capabilities, that it is the more obscure or more complex data types like XML and so forth where LINQ really shows its chops for making life easier for developers, but the fact that we can abstract the way the fact that this is being clustered is amazing to me like what a great proof of the potential of LINQ.

Michael Isard: Absolutely. Okay, so I have the 1.4 there.

Richard Campbell: And so for our listeners, if you get -- well, the links for the show will include a link to the Dryads site and you can get to this research paper that is a PDF file. We're on page 7, it's the F graph example. It's only a handful of lines but it really, to me, encapsulates the great abstraction we've got here where you'd never know we were running in a cluster here.

Michael Isard: Right. So the first thing there is a static F graph function which is just saying it takes a string from a collection and returns the evidence of that collection where the string can find the particular subscript. So it's a very simplified graph example.

Richard Campbell: Right.

Michael Isard: And then there's a main program which starts off by essentially telling the system where it can find the input files. So it says that the partitions of the original input can be found in this part and then there's a input file, .txt, which describes the number of partitions in which computers they live on and that kind of thing. Then you just make a Dryad data context which is a LINQ description of what, where the data can be found and then you ask it to get a table from that data context based on this filename, this input file.txt which is what the actual partitions are and that's parameterized by saying that is a line record and essentially what that is saying is that these filenames are all made up of regular new line terminated strings.

Richard Campbell: Right.

Michael Isard: So here we have a standard deserializer, deserializer for new line terminated strings because lots of log problems particularly turn off as that. If you have some more complicated data format of your own, you can write custom deserializers of course, and if you just use a regular C# struck then DryadLINQ will generate all the serialization code for you, but in this case the file is assume to come out with this new line terminated strings that's using a built-in DryadLINQ cluster to deal with those and there's a line of LINQ that basically turns those line requisites into strings, it just

does a select on that initial table where each line record gets mapped on to the string field of that record and then we do the match just by applying this F graph function to that table and when this main programmer executes each of those lines, as I've said earlier, they're just building up the Query Expression in the local machine and then it close this show in console function for that output of that F graph and it's at that point that it tries to materialize some of the results of the query and at that point Dryad LINQ will kick in and it will construct the query time and ship it off to the customer and actually run it and then go fish the results back.

Richard Campbell: So the Dryad data context statement was really the point where we said here is the block of data that you're supposed to distribute, and Dryad table contract set the how to decompose that data across those machines.

Michael Isard: Well, no, so the data has already been decomposed by somebody across those machines. So what happens is somebody has placed all these on the custom machines.

Richard Campbell: Oh, I see. So the file that you referencing to in the Dryad data context actually represents files that exist on every one of the machines that cluster?

Michael Isard: That is right.

Richard Campbell: So we've got to go take this -- imagine we're starting off with a huge file, we're going to break it up into chunks and ship it to all of the machines in the cluster...

Michael Isard: Exactly right and...

Richard Campbell: And better make it the same filename or you're going to have trouble.

Carl Franklin: Right.

Michael Isard: So this input file.txt is actually the description of the names of the partitions so you can call it whatever you want.

Richard Campbell: Okay.

Michael Isard: That's kind of a bootstrapping issue, I mean the question is whether you get this big file and for a lot of applications, say this is some log files, well, maybe it's already been partitioned because you're collecting the logs on a big cluster already or you know, it was writing out, it was truncating each file at 10 megs of something and so you reload this very large data application, you're already starting off with something that's already

partitioned and if you aren't, then you have to partition it manually.

Richard Campbell: Yes but I think it's a salient thing to understand that the actual partitioning process is done separately. That's not Dryad responsibility.

Michael Isard: Right, although when it writes out the output of a computation, that will be written out as set of partitions into the cluster which you can then use as input for the next one, but yes if you're starting out with...

Richard Campbell: Right.

Michael Isard: And something to note there is that the way that you have partitioned your original data is going to have a big effect on the program that Dryad then runs because, you know, its natural incarnation unless it can see a better thing to do. You know, if you have 50 partitions, it will run 50 processors on one or each of those partitions.

Richard Campbell: Right.

Michael Isard: Then if you have a much larger cluster, it may try and repartition that, but given that local disk is a lot more efficient than pushing everything across the network at once, it tries to keep things just waiting from the local disk as much as possible.

Richard Campbell: But it does have the ability to move workload to other machines?

Michael Isard: That's right.

Richard Campbell: So if one of these machines, this cluster is having problems for whatever reason and starts to kick out exceptions, there's a mechanism by which Dryad is able to pull that data from that machine and place it elsewhere.

Michael Isard: Well, not exactly. So it can pull that data from the remote machine just using a remote file transfer protocol. If the machine is really having problems, you may not be able to pull the data...

Richard Campbell: Yes.

Michael Isard: So if you want to guard it against that, you can have multiple replicas of each of these partitions and so in reality most large clusters that you would run this on, you would have a real distributed file system that was managing all of that for you and that was moving this partition around and replicating the machines file and that kind of thing.

Richard Campbell: Yeah and I've just recently configured the distributed file system set up as exactly that. That would be simply harness it across multiple machines anyway so that would just happen for you.

Michael Isard: Exactly and so there's a fairly simple enterprise lair that we can plug Dryad into existing distribute file systems. You know, a lot of these examples assume that you just have the five machines that you had spare in your storage closet and so you don't necessary have a distributed file system and if you just want to take and play with it, then you can just manually partition the files, but in practice if you're using this seriously for any long running application, you would want a real distributed process.

Carl Franklin: Hey, I just want to give a shot out real quick to our friends at Data Dynamics who make ActiveReports.NET among other awesome things. ActiveReports.NET is great because it allows you to just build your reports with the Easy Editor, embed them right in your application, provide PDF and HTML output, give your end-users a Report Editor, royalty free of course, a great Access report upsizing Wizard and all this for a price that isn't going to break the bank. ActiveReports.NET from Data Dynamics, go check it out now at datadynamics.com.

Richard Campbell: I'm just thinking of other possibilities on how I might utilize this. You know, I like the distribution mechanism as a better way to harness even multi-core machines rather than trying to farm out a whole bunch of threads to just kick off a whole lot of processes that will individually use processors anyway.

Michael Isard: Well, so that's actually an interesting -- I think the research is still not fully conclusive on that so we do actually have multi-core machines on our clusters, but if you remember what Dryad LINQ does is it takes its original LINQ Expression and it decomposes it into sub-expressions, but each of those is still a LINQ Expression.

Richard Campbell: Right.

Michael Isard: So the way we actually execute it is at the remote machine we use PLINQ to use the multiple cores on that machine so that each of those sub-expressions, it also execute in parallel in the cores of these machines. You know, the PLINQ team is doing a great job of solving that problem for multi-tour and there are slightly different trade-offs. When you have the shared memory, there are different things which are more efficient and so for the moment, yeah, that seems to be a good division of work that we use PLINQ within a single shared

memory machine and Dryad to distribute among machines.

Richard Campbell: Multiple machines. It really comes down to what resources the limiting factor in this process. Again, it's not CPU. We've got lots of that. It's probably not even memory. I guess it's really disk. If you're reading serially through one big file it's going to take longer and if that file's chop up into 50 pieces on 50 machines and they're all reading simultaneously.

Michael Isard: Right. I mean different people run different workloads on this and some of them are CPU bound actually.

Richard Campbell: Okay.

Michael Isard: But we can get PLINQ to, you know, you might be doing image processing on a video for example, and so each of the machines might have a few thousand frames of video and you can process those frames of parallel using PLINQ at those local machines though.

Richard Campbell: And this is not about the size of video issue. This is about how much work it takes to do the rendering on each one of those frames.

Michael Isard: Right, yes or the analysis and so it's just a way of farming out a lot of work to a lot of computers transparently. There is a mixture of things which are IO bound and compute bound and I think we haven't got a great idea which -- we have successful programs running in both of those areas.

Carl Franklin: You worked on the MSN search project.

Michael Isard: That's right.

Carl Franklin: I got to think that some of these came out of some of the things that you did there.

Michael Isard: Yes, both indirectly and directly. As you mentioned in the beginning, my PhD was in computer vision and so shortly after I got to Microsoft, I got the opportunity to work kind of as a virtual developer on the V1 search engine that they had just started out and so for me that was a lot of education in building a real system and in real software development practices outside of research in a product group. You know, I learned a lot of supporting knowledge, I learned a lot from experienced people how large systems work and how you build them, and also I saw some of the needs that groups like search we're going to have for processing large amounts of data and so that was also an inspiration for wanting to work on this.

Carl Franklin: Quite a lot of grid computing going on in the search industry, that's for sure.

Michael Isard: Yup.

Carl Franklin: Yeah. Has Microsoft been utilizing any of these for their search or did they have their own sort of...?

Michael Isard: Yes, this has been used in the back-end processing of file logs for search for a couple of years now.

Richard Campbell: Wow.

Carl Franklin: Excellent.

Richard Campbell: Well, and the big thing here is the integration into Studio, integration to LINQ so you're sort of minimizing the amount of work I have to do as a developer to learn how to use this, a stuff I should already know.

Michael Isard: Right. That is the intention certainly.

Richard Campbell: So where are we going to get surprised as .NET developers in getting our heads around actually doing Dryad and I'm just guessing that one of the challenges got to be partitioning.

Michael Isard: Right. So certainly you have to think a little bit differently, you have to think about partitioning a little bit although the system is fairly good at letting you ignore that. It will usually pick the number of partitions, the intermediate data for example.

Richard Campbell: Right.

Michael Isard: If you really have a large dataset, then the best thing to do is to have a partition in it. So if you have a few hundred machines, if you make tens of thousands of partitions so each machine have multiple partitions on it, that gives the system the opportunity to really do as much work as it wants to do in each machine and pick the granularity so the programmer doesn't have to understand that as he moves to smaller, you know, your initial dataset is going to be in 10 or 20 partitions, then you might have to understand something about the workload to pick the number of partitions.

Richard Campbell: Right. By partitioning in many, many small chunks, you're giving Dryad enough granularity to do what it thinks is best.

Michael Isard: That's right and you know it's obviously not perfect as you can imagine, but it does a pretty good job. Once you've converted your

program to LINQ, then a lot of the hard work has been done.

Richard Campbell: Right.

Michael Isard: There's a certain mindset that you have to get into which is, I mean LINQ is really functional programming if you want to look at it that way and so once you take your old sequential programming or your old sequential mindset and start to think of composing dataset functionally instead, then having done that transformation often gets used to the point where you have a reasonable distributed DryadLINQ execution. So we just had several interns in our lab for the summer that were using DryadLINQ, and so I think by and large they were up and running pretty quickly. It wasn't a big learning curve. I think the main issue is around performance debugging at this point.

Richard Campbell: When it isn't fast, what do you do, I guess, is kind of the hard part.

Michael Isard: Right and when there's some strange bug, how do you find out what it is. So for example, we had somebody who is trying to do a page rank like algorithm and so there are a bunch of URLs that we're partitioned out and one of the machines was just running about slow and all the rest went to look at it and it turned out that the hash partition function wasn't very good. There was some demand that was very large and they were all being put in the same baseline. You know, you don't look at that, you use a different partitioning, but then it was still running much slower than other machines and eventually it turned out that the issue was that in some particular group on some host, the URLs were a lot longer. So even though each of the partitions had roughly the same number of URL at that point, this one machine was starting to page because the memory footprint of that that was just much larger and it's that kind of debugging, you know, you look at all of the sizes of the input and the number of records there, they all look fine and still something is going wrong, then we need better tools to dig down on that level of bug, but they are relatively unusual, I mean, it's certainly compared to what we expected. It's surprisingly easy to get something that works most of the time.

Richard Campbell: Well, and there in lies sort of the opportunities for Dryad optimizer to say this one machine is really struggling, let's shift some of its workload off, and then if you do enough granularity you ought to be able to do that.

Michael Isard: That's right and at the moment we don't have that happening automatically. it's not going to repartition on the fly although we certainly talking about ways to do that and also I think there's a

lot of start for sort of profile guide ay optimization in the large that it's pretty unusual that you only run a program once especially if you're debugging it. So each time it executes, it can collect a lot of statistics and hopefully do a better job the next time.

Richard Campbell: This reminds me of SQL server in that the more you query SQL server, the more it understands about the queries you're going to run.

Michael Isard: Absolutely and...

Richard Campbell: And the data that's running against.

Michael Isard: And again this is the nice things about LINQ, it's that although the LINQ Expression plans are more general than the relational query language, you know we have decades of research in databases that we can learn from so it's not like we're starting from scratch here.

Richard Campbell: Yeah and now I'm starting to think in terms of rather than in the file systems, if I had databases on each one of these machines that I could spread data around to, would this still make sense?

Michael Isard: Well, yes. Actually again, you know, let me talk of LINQ. So we can do that if you stick a database on each of the computers, then we can use LINQ to SQL locally and then that will push down as much of the computation as possible into the database engine and then, you know, the rest is the functions that are implemented down there, it then gets materialized out as C# operates and can get shift around by Dryad so it's actually a very nice integration story there. We're looking into maybe after being clustered with each machine having a SQL database on some of the disks and just some flat files on the other ones and see if we can get a really nice different datasets that will be in different parts of the disk depending on their characteristics.

Richard Campbell: You can get a feel for what things are going to behave better in one model or the others, it just got a lot of possibilities.

Michael Isard: Right and we can make use of all the SQL indexes, the things where that's going to be a big win without trying and trying to redo all ourselves.

Richard Campbell: Yeah. Why reinvent that where file systems are found when the data doesn't have an inherent order that you care about.

Michael Isard: Right.

Richard Campbell: Or has already ordered that...

Michael Isard: Or is already ordered, exactly. So for a lot of these intermediate data, we're writing it out sequentially and reading it back in sequentially and you don't want the overhead of stick it into a database which doesn't know any better and is going to do extra work for you. You know, we can just slam it out in the raw disk as fast as possible when that's appropriate, but when it's not it's great to be able to use SQL.

Richard Campbell: So how big a cluster have you played with for Dryad so far?

Michael Isard: So the research cluster that we have in the building is about 250 computers that we've done a little of the DryadLINQ research on. The search team has larger clusters than that. Yeah, most of the research happens on a few hundred machines.

Richard Campbell: And knowing that it runs in a few hundreds really shouldn't make any difference at, I guess, to a few thousand. What elements are these aren't going to scale? Is it the integration that becomes difficult?

Michael Isard: In my experience, every time we've gone up an order of magnitude computers we find something that pulls over and needs to be tidied up but there hasn't been any show stopper. It's been more, you know, we try not to aggressively optimize things that aren't needed to be optimize. So the things that need to be optimized, you usually find one or two extra ones and I think that's fine, that's to be expected.

Richard Campbell: Well, the fun will be, you know, every order of magnitude so get 10,000 servers together, you'll get some new problems. I can't wait to see a hundred thousand servers together. That app would be fun.

Michael Isard: Right. I mean, at some point there maybe diminishing returns on -- I think there's a real question of when you have a hundred thousand servers, the extent to which it's good to have a single cluster compared to 10,000 computer clusters...

Richard Campbell: Sure.

Michael Isard: And you know, because the data is being shipped around across the network, I don't think we really know how to have perfect resource isolation between two jobs running at the same time and so I think it we're still a couple of years out from being able to say with a hundred thousand machine cluster you can have these 10,000 machines and we can give you a perfect SLA regardless of what else is running on the cluster.

Richard Campbell: Yeah, the network contention when you get yet into that kind of size gets very complicated.

Michael Isard: It gets very complicated, exactly.

Richard Campbell: And that's where you'd stumble.

Michael Isard: We do our best with that but it's certainly nowhere near perfect yet. So there are still reasons why and just practical reasons for deployment...

Richard Campbell: On something you said right off the bat that this is not about latency, it's about throughput. There is an overhead to organizing all of this so unless the problem is big enough, the overhead can overwhelm the savings.

Michael Isard: That's right.

Richard Campbell: And the same problem as you try and scale this up, it doesn't scale uniformly because eventually -- I'm an internet geek so I know when I hit 80% saturation in a switch, I start getting a collision rate that suddenly eats more and more of my bandwidth and it gets worse and not better as you put more pressure on it.

Michael Isard: Right and so then historically a lot of these large data centers have been built with hierarchical tree structure networks and maybe there are different apologies that will scale better for example.

Richard Campbell: We still get back to the problem of what are doing that we think we need a hundred thousand computers.

Michael Isard: Right. Do you need a hundred thousand computers for a single job? I mean, I'm sure that there are some that could benefit from that but for most people if you get up to that level what you're really going to be doing is running a lot more jobs at once and then the question is whether it wouldn't be easier to just have more clusters and you know there are arguments in both directions. It depends partly on how much shared data you have. If you have really vast collection of data that you won't be able to keep one copy of, then that can be an argument for having larger and larger clusters but you're always going to need your replications of this. It's that important in any ways.

Richard Campbell: And you see this as sort of a specialized task kind of thing that an organization would put together this cluster of machines purely to

work this one problem. It just doesn't seem to me like a general use cluster with whatever exists.

Michael Isard: Well, we have a lot of people who want to use hard cluster for their problems so I don't know, I think we're still learning what the full range of problems that work well in this architecture but I think we haven't run out yet certainly.

Richard Campbell: Yeah and I wonder if that's more of a proof thing. I have this particular problem and if I can run it on your cluster and it does a great job, then I want my own to keep solving this problem.

Carl Franklin: I should think NASA would be interested in this actually because don't they get tons and tons of telemetry from their various spacecraft that have to be poured over in various ways?

Richard Campbell: And also the whole super computing model in general, right? Like just reading about the new Petaflop machines that the nuclear physics guys need, this is a different way of solving that problem although I wouldn't know if it will actually would solve that problem.

Michael Isard: Well, I think there are some of those problems where they're really pouring out a large amount of data that it might be good for, although I believe they have to do a lot of the work very much in real time because they can't afford to store the data even once.

Carl Franklin: Wow.

Michael Isard: But yes, I don't know beyond that.

Carl Franklin: That's a bad problem to have.

[Laughter]

Michael Isard: Yeah. A lot of the -- historically supercomputers have been used for finer element simulations, so weather and bomb simulations and that kind of thing, and you know that's the sort of problems that MPI was really designed for and where the algorithms...

Richard Campbell: Sorry, MPI?

Michael Isard: MPI is -- actually I don't know what know what the acronym expands to but it's the standard distribution computing interface that people use in the typical computer world.

Richard Campbell: Okay, so sort of standard way of using super computers versus something along these lines.

Michael Isard: Right, and a lot of those algorithms do frequent synchronization so they have all of the computers do a relatively small amount of work and then communicate with each other and then do some more work and then communicate and that's how these element simulations work and so those systems have been optimized for doing that, have no communication very well, whereas Dryad is really optimized for doing more work between stages and having fewer sort of many to many communication barriers. You know, I think we're addressing a slightly different cluster of problems that you can do with either architecture but we're optimize for slightly different problems.

Richard Campbell: The interesting thing about there and really feel like this is one of Microsoft's specialty is bringing this sort of what was once the white lab coat-type technologies into more easy access use to a point where we can really experiment with does my problem work in this scenario and if it does then this is not about buying a supercomputer. It's about getting a bunch of older machines together to get the same results.

Michael Isard: Right. That's really what we're aiming at with DryadLINQ certainly.

Richard Campbell: A message passing interface...

Michael Isard: Okay.

Richard Campbell: That's the computer communications protocol for parallel computation, MPI. That's really an anti-climactic...

Carl Franklin: Yeah, sorry I asked, yeah.

Richard Campbell: Duh, jeez I've ruined it for me, but I guess the magic of really serious parallelization is how do I distribute the work and communicate about the work and integrate the work.

Michael Isard: Right.

Richard Campbell: So what's next for Dryad?

Michael Isard: So we are continuing to do research into the system level aspect of Dryad. I mentioned the way we're doing research, we have interns, some are working on the monitoring problem with trying to build tools to help you learn more about why your job runs slow and we also, I mean I alluded to briefly this resource isolation problem. When you have a lot of jobs running on the same cluster, the scheduling problem obviously gets even more interesting. You know, the original Dryad design for simplicity, so try to make it do the best thing when there was just one job on the cluster and of course, you know it's hard to use an entire cluster for the

entire jobs so really you want multiple jobs running at the same time to get good utilization.

Richard Campbell: Sure.

Michael Isard: So there's other research in the academia that we can build on there and, you know, we'll locate that and we're still looking but the things that we can do, optimization we can make because of the way the Dryad works, the Dryad improves that. You know, we're getting more and more people inside the company using DryadLINQ and Yuan is still going forward finding things that it doesn't do the way people would like it and making improvements and just broadening the user-based and learning more about how it behaves.

Richard Campbell: So this is something we can go download right now and take out for a spin?

Michael Isard: Unfortunately not. It's a research project still and at the moment it's just internal only. Of course we talk to product groups all the time and we would like at some point to have some kind of external release, but for now it's internal only I'm afraid. You can read about how it works and see the example programs that we're talking about on the website, but for now it's just internal.

Richard Campbell: There aren't actually bites we can lay out hands on yet.

Michael Isard: Correct.

Richard Campbell: One of these days.

Michael Isard: Hopefully.

Carl Franklin: That's a fascinating look at a fascinating project. All right, well, I guess that brings us to the end of the show. Michael, thank you very much.

Michael Isard: Thank you.

Carl Franklin: And keep up the great work.

Michael Isard: Thanks very much.

Carl Franklin: And we'll see you next time on .NET Rocks!
[Music]

Carl Franklin: .NET Rocks! is recorded and produced by PWOP Productions, providing professional audio, audio mastering, video, post production, and podcasting services, online at www.pwop.com. .NET Rocks! is a production of Franklins.NET, training developers to work smarter and offering custom onsite classes in Microsoft



**Michael Isard on Dryad
September 18, 2008**

development technology with expert developers, online at www.franklins.net. For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.