Transcript

Speaker 1: [inaudible] welcome to the graduates, our radio show dedicated to graduate student research here at Berkeley. My name is Stephanie [inaudible]. I'm a graduate student myself and I'll be your host, this position here on k a l ex Berkeley.

Speaker 2: So today I'm talking to Andy Konwinski, a phd student in computer science. So welcome Andy. Thank you very much. And we're going to be talking about measuring and [00:00:30] improving the performance of programs that run on supersized computer clusters. So first, I know that you work in a lab of the computer science that's called the Rad labs. So can you first tell us about what that is? Sure,

Speaker 3: yeah. So the Rambam, uh, it's an acronym. It stands for reliable adaptive distributed systems and it's kind of the systems lab, uh, in the computer science department here. And what that means is that we're working on that kind of the core [00:01:00] part of computer programming and now, um, with, and that's kind of traditional systems. So I kind of building your operating system on your computer and now we've kind of taken a step back and started focusing on building the operating system for, uh, a large number of computers. So like data centers or, um, this is where we take a large number of small computers and have them all talk to each other. And, uh, the Rad lab then is focused on allowing a very small number of people [00:01:30] to manage and write software that runs on a large number of computers.

Speaker 2: Right. And that's what you refer to as a super-sized computer cluster. Yup. I like that. Um, okay. So in the Rad lab vision statement, it says that your vision is pretty much what you just described, but to enable one person to invent and run the next revolutionary it service, uh, operationally expressing a new business idea as a multimillion user service over [00:02:00] the course of a long weekend, which you referred to as a internet fortune 1 million. And I, I like that too. So, um, so with the Rad love is doing is enabling the fortune 1 million to manage these supersized computer clusters.

Speaker 3: So kind of the origin of that story. And that tagline for the Rad lab is a lot of these stories about one college student, like yeah, the guy who made Napster or the guy who made Facebook [00:02:30] and, or the guy who made Ebay. And what they do is they sit down for a long weekend and they come up with this new idea that catches on like wildfire. But the first iteration of their program, um, tends to become so popular so fast that they end up rewriting their, their application like seven or eight times. So ebay's on like, it's seven 33 right now. And that's from scratch just because they hit these kinds of plateaus where their servers will crash because the application wasn't written to scale very well. So the more users they [00:03:00] get, the more they have to rethink, well now the way we were doing this before isn't going to grow fast enough.

Speaker 3: So we're trying to do is enable somebody with a new idea to write it once from the very beginning and then it'll scale for them with them as they grow. And where does the ability to scale come from? I guess the difference is that a, it's actually a kind of in the framework that sits underneath the code. They, right. So, um, they were using kind of more traditional tools like traditional programming languages, something like PHP is a lot of people use [00:03:30] it for web programming. And now we've got all these like world famous computer science professor sitting down and putting their brains into the project and maybe we can still write PHP, but the, the kind of the underlying frameworks, like the operating system basically interprets that code is handling it differently. Right. So the same PHB used to run, you know, on one computer. Now without the program or even knowing it, it'll run on tens of thousands of.

Speaker 2: Wow. Okay. So who is your target audience for the research that you do at [00:04:00] the Red Lab?

Speaker 3: Well, I was wanting the starter, right? Yeah, definitely. I don't really, um, we, we meet with Facebook and uh, Yahoo and Google and our founders, like kind of the people who've, who funded us the most, um, are the three big sponsors of our lab. Our, Our sun microsystems and they do a lot of software and Microsoft and Google

Speaker 2: [inaudible]. And you're going to make your work available on your website for the [00:04:30] littler people.

Speaker 3: Yup. Yup. So it will be available to download everything. If I think we'll eventually give the URL for the lab. And if you go there, you can find links to all the projects that are currently being worked.

Speaker 2: Yes. For those of you who want to start a revolution over a long weekend, I'll give you the URL at the end of the conversation.

Speaker 3: Yeah. So if they keep on listening, they can, they can download everything from the individual projects websites.

Speaker 2: Good. So if you just tuned in and you do want to start a revolution over a long weekend, you're [00:05:00] listening to the graduates on Cadillacs. I am talking today with Andy Konwinski from computer science about measuring and improving the performance of programs that run on supersized computer clusters. So, uh, in the vision statement, it also says that you try to borrow technology whenever possible. So let's talk a little bit about the technology that you borrow and, uh, specifically when it's called MapReduce. Um, oh, and I should [00:05:30] also mention that you work with Latte. Does that Hurria who couldn't be here today, right. That, um, I did want to mention his name. So first, what is, what is math produce?

Speaker 3: MapReduce became famous when Google wrote a paper about it, um, last year or sometime in the last few years. And um, essentially what it is is it's a new, it's actually an old way that they've re popularized. It's a way of thinking about make about distributed or paralyzed computing. [00:06:00] That's easy. So, um, it's, it's very highly abstract. All the complexities that go into to making one program run on hundreds of thousands of computers. So it's, it's basically a framework that, um, at an abstract level anybody can implement in software. And Google has implemented it in their, in their data cluster, but they keep it a secret and keep everything a secret and a, so some open source, people have been working on a project re implemented, but they are one of your sponsors. [00:06:30] That's right. So they're interested in a Google that's kind of an, an interesting dynamic. So they're very secretive as a company.

Speaker 3: They have a small branch of people that work on open source technology, so they're not against open source, but because they also borrow a lot from open source, especially ideas. But, um, they don't release any of their source code at all. Okay. But they really want us to do as an, as a university is to produce better educated computer scientists. They'll consume on the other end [00:07:00] of the lack of personnel. Right. That's exactly why they're interested in us. So, but back to MapReduce. So how does it, how does it decide how to distribute which tasks to which nodes? In the supersized computer cluster, it takes a large input file, something like a crawl of the Internet, which could be, you know, now you're talking terabytes, petabytes and um, like, you know, thousands and thousands. Yeah. [00:07:30] Um, and so it takes this huge set of data, basically a text file and it breaks it up into really small chunks, all the same size.

Speaker 3: And then it gives each one of those chunks, which has just enough for one computer to handle two different node or computer in your super-sized cluster. And each one will kind of crunch on that for a while. It'll do the computation on it and then it'll kind of set it in a bucket. And then these, the next wave of tasks, which is your reduced tasks, I'll [00:08:00] run on the same set of computers in your, in your supersize cluster and grab the output from the map tasks, um, across the network. So there's the shuffling stage where they all kind of grab from each of the map tasks on all the different computers and then they do their crunching and then they write their output to some really, really big hard drive. Or are these individuals working over the lung? We can get and get access to these supersized clusters. Like you go online and you can rent [00:08:30] from Amazon. Actually, you can rent computers by the minute, by the hour, and you can, so you can rent a thousand computers for one hour for $100 computers that are hanging out in a big warehouse using their processing power all of the time. Yup. Yup. And Amazon's trying to turn this and they call that actually call it, um, hardware as a service or a utility computing because of this turning into utility, like the, like the electric unit.

Speaker 2: Okay. So, so I know that you use, [00:09:00] uh, an implementation of MapReduce called Hadoop. So yeah. What is had been, how do you use it?

Speaker 3: So like I said, Google wrote their own version of MapReduce and then that kind of, and then they wrote a paper about it, which kind of taunted the rest of the world saying, look at how fast we can and how easy it is for our programmers to write distributed programs using our implementation of this old idea called MapReduce. So the rest of the world then retaliated by writing this open source version in Java and open source language [00:09:30] that they, they can, uh, that you can download yourself and run. So this is kind of, it's actually being led by a team at Yahoo who, but the software is all free and open for you to download. So Yahoo was kind of giving, it's combating Google's initial bragging about map produced by their own version, which is not only are we going to write it, but we're going to write it in a transparent fashion so that anybody can both download the source code on their own and look at it, but also contribute back to the project. And since Yahoo kind [00:10:00] of started that with their core team, um, it's grown and people from all over are contributing to it, including us here at UC Berkeley.

Speaker 2: Mm. So I know you've been doing some work for Facebook with Hadoop. What's that been like?

Speaker 3: Um, well, one big thing is that it's huge. Uh, their, their data sets are really big. So those are always interesting. But another is that they have a unique application. It's a social network. So networks, um, or graphs are fascinating to computer scientists. Facebook [00:10:30] has the, the ultimate in that because not only are we looking at this huge craft, but it's interesting to everybody because everybody's a node in this graph. So you're connected to your friends and you can run just like you can run queries against their dataset that that would make everybody jealous because you can find out such interesting things about how humans work and, and, um, how college students work.

Speaker 2: So what have you went into, what have you learned about college students work? Specifically [00:11:00] Berkeley graduates?

Speaker 3: No, actually, um, Mottainai aren't, aren't interested in that. That's one interesting thing to Facebook. I can't, I can't isolate anything because I'm not a Facebook employee. But that's why Facebook's application is interesting and what [inaudible] and that's why they're using Hadoop because it enables any of their programmers to write these really interesting queries against their, you know, their tons and warehouses [00:11:30] full of data that they collect and find out interesting things to maybe direct ads better at you or something like that. So now I can't speak with any authority about what's happening inside of Facebook, but that's what, why their problems interesting and why they're using Hadoop because it makes it easy for their developers to write applications for such a large Dataset. And so what they're, the reason they're interested in my work [inaudible] is because what we're focusing on is making can do faster and they are one of, you know, the people who are using [00:12:00] it at such a large scale that even if we can improve it by a few percentage points, then Vail will see a huge improvement in the amount of time it takes for these large queries to run.

Speaker 2: Good. So, okay, so let's talk about the, your project specifically, what you're using Hadoop for. So you're working on something called ECS chase, and that's, I love all these words by the way, had Dubin XJS supersized computer classes. So can you first explain what x trays is?

Speaker 3: Sure. That's actually a kind of difficult project [00:12:30] that, uh, to explain it in an easy way to understand. Good. This is your challenge. Um, so x-rays is a type of tracing, uh, tracing traditionally and computer science means we, whenever we're writing source code, we put these little statements in the printout on your computer screen or in a, in a text file somewhere. Um, useful information to the developer. So if there's a bug in your software, like if it stops working or crashes, we all get the little popups that say, do you want to send this report to Microsoft [00:13:00] or apple or whoever your operating system vendor is the thing that it's going to actually send to Microsoft or apple or whoever is going to be a big punk chunk of text that will represent what the state of the program was then what it had been doing on your computer.

Speaker 3: And maybe by looking at that and parceling it out, we can figure out why it crashed. So that's traditional tracing and x-rays is kind of a new approach to the same idea. But now instead of, um, using these print statements inside of our [00:13:30] programs, we, we connect the statements to each other. So before when we get into a distributed system, we actually care about the relationship between these, these printouts or these kind of check points, um, in our application because now we're running on hundreds of thousands of computers. So if I have to collect this information from, from a thousand different computers and figure out when I clicked one button on one computer, what that caused on each of the other computers, I'd have to have some really smart [00:14:00] way of kind of putting this, this disparate set of data back into this, into this one interleaved kind of contiguous section of trace states.

Speaker 3: So what we do now is we kind of, we add a little extra data to each print statement and as your application sends packets across the wire to from one computer in your cluster to another, it carries that information with it. So it makes each checkpoint that that gets reported in the log or in the trace aware [00:14:30] of where it came from, right? So now we can build these directed again, we build graphs out of it, right? So instead of a whole bunch of unconnected little balls in our graph or nodes, we have a whole bunch of connected, um, edgy, a whole bunch of connected points. So now we can kind of, we can look at when you clicked in our golf and we can just like follow the arrows pointing to where that happened and that be on different computers. It can be a halfway across the world and it can be on the same computer. Yeah.

Speaker 2: And how is that displayed to you? Do you visualize it?

Speaker 3: Yeah, we actually have some really awesome visualizations. [00:15:00] If you check out the project page, that will give a URL to it and then you'll be able to see, um, if you want to play with it, you can see that what MapReduce gives you, it gives you a very long initial thread, which is just a bunch of setup happening on one computer. And then there's this huge burst and like out of one circle on the graph comes like 10,000 arrows because it'll, it'll, it'll spawn multiple operations on each of the computers in your cluster. So now it's, it's some factor [00:15:30] of the number of computers in your cluster that comes out of this one graph that says, okay, now go. And then you can have all these parallel iges in your graph running next to each other because everybody doing their little part of the job and it's all going to return. So it's typical and I guess MapReduce is one of the kind of, um, standard examples of this sort of [inaudible]

Speaker 2: you can start printing these things on tee shirts. [inaudible]

Speaker 3: I'm sure a lot of computer scientists would love to. Yeah. Actually I saw like last week that I'm [00:16:00] one of the guy who guys who's working on, on Hadoop set up as a Hadoop store. So you can actually buy in there. Their logo was an elephant and you can buy the logo emblazoned tee shirts and hats and stuff. Now

Speaker 2: the uh, the,

Speaker 3: no, no, we should, we should propose that it would be a good logo for them to use as a secondary logo to the ELA.

Speaker 2: Ah, so that's, uh, so combining, um, the juicy stuff, combining Hadoop and x trays together. Uh, so what happens when you do [00:16:30] that?

Speaker 3: Yeah, so, and now we take this really neat idea about path based tracing I just talked about with x-rays and we take this really popular framework and we put them together and we offer new insights into how Hadoop works. So the problem with Hadoop is that being open source and not being made by Google, basically it's, it's, it's slower than whatever Google can make. Google, uh, hires so many, um, PhDs and computer scientists right out of school that they have more [00:17:00] engineers than any other company by far. Um, computer science engineers, that is, they're software engineers. And so the softer they write is, is coveted by the open source community. And we want to emulate that. So, um, by kind of revealing and shedding some light on the inner workings of Hadoop, are able to tease out the details about where time was spent inside the job and then we can know where to focus our efforts so that, um, we can improve at Dubin. Make it faster.

Speaker 2: Um, okay, so we will be right [00:17:30] back

Speaker 1: on next week's show. I'll be talking to Stella oftener phd student from the Astronomy Department about simulating star formation. So please join me for the graduates every Monday from 12 to 1230 on calyx and visit us on Facebook search

Speaker 2: for the graduates, k a l x in quotation marks on Facebook. Welcome back. Today I'm talking to Andy Konwinski from [00:18:00] computer science. So let's talk about the broader implications of your work. Just the fact that any of our personal computers can now be part of a supersized computer cluster for an it service or something else. Can you talk about the implications of that for the client server distinction?

Speaker 3: People have discovered the value of distributed computing or, and this is kind of a different type of distributed competing. [00:18:30] The I we're talking about in the show so far, which a, which is peer to peer sharing or peer to peer computing. And it represents a break from the traditional [inaudible] model of client server where now everybody's a client and everybody's a server. So the way that we, um, the way that we kind of capitalize upon that in, in the realm of, you know, challenging the traditional client server is writing applications that do the same thing as these traditional client server. And instead of now making that server really [00:19:00] robust and really high powered, we use a whole bunch of computers that have our friends or maybe there's a whole bunch sitting in a warehouse and now we have a, the ability to kind of go out and talk over the Internet to all different types of computers and give them a little piece of, of data and a little tasks to do on it.

Speaker 3: And maybe it's, it's a, it's something like Seti at home where they give you some information that they've collected from by pointing antennas at the sky and they want to know if we can, if [00:19:30] that little piece of information they gave you contains any noticeable patterns because it should be all random if there's no aliens out there. But if there are any aliens that are trying to talk to us, then they're gonna send us like some recognizable sequence of bytes of, of ones and Zeros or numbers or something we can recognize. So if you want your computer to download a little chunk of, of data than and process it and check and see if any aliens have spoken in it and then send that back up to the server or to somebody else, the process on for awhile, then you can use study at [00:20:00] home and then we can do this with protein folding, which is another kind of biological application that's folding at home. And there are even more applications. Yes, yes.

Speaker 2: So does peer-to-peer computing end up consuming less energy? Yep.

Speaker 3: That's a problem we really care about in the Rab lab is power. Um, and that's a problem that any of these very large players in the kind of Internet sized data center or distributed computing world care about because it's getting to the point where they're building their data centers or these [00:20:30] big warehouses of computers, which they have so many have along rivers that have hydro-power and they're looking to build them in like Siberia where authority cold so they don't have to cool them down because the major cost factors in power are in a data center, in a, in a warehouse full of computers. That is our, uh, paying for the electricity it takes to cool it. Um, so paying for cooling because it takes a really big air conditioner and also paying for the power it takes to run these computers. So you're right. Yes, [00:21:00] it is one of the biggest consumers and we've done a lot of research in the Rad lab and it's going to be a successor, one of the successors to the Rad lab, an idea that came out of it.

Speaker 3: And we're looking at ways of hopefully turning computers off when we're not using them in our cluster of computers. In our data centers so to speak. And then looking at ways of making them go into sleep modes when they're not being used if we need to. So the problem with turning computers off is that it takes a long time to come for them to come back on our boot up. If we can just [00:21:30] put them to sleep and cut the power consumption by like 60%, then they can come back on, you know, and one 10th of the time of the cold boot. That's a good, that's a good thing. If we can distribute the computing across people who are just volunteering their, there's computer cycles cause they leave their computer running on at home anyway, which is a really bad idea. Everybody's turned their computer up at night, then we save even more money.

Speaker 3: So yeah, that's a great thing that you brought up. And, and there's a lot of research going on this here at the Lawrence Berkeley Laboratory. Um, we've had talks [00:22:00] and visiting speakers come in and talk to us about the major culprit is not actually in the Google and Yahoo size data centers for the total number of watts consumed in useless psychos because these guys use their cycles. They care because they have to pay for them because it hits their bottom line. So they're interested in turning things off. But where the majority, you know, the vast majority of these power cycles go is to people's computers sitting at home, leaving their monitor on, leaving their computer on and not using it. So it, and it's something, some ridiculously small [00:22:30] percentage, like five or 10% of the time on average, that of all of the added cycles on people's computers are used by somebody sitting there. And the only reason we leave it on is because we hate waiting for it to boot up. But really, even, yeah, if we turned it off, we would, we would be saving countless numbers of trees, save

Speaker 2: trees. You can save energy and you can dedicate your unused, a computer processing power to some, you know, virtuous project. Either your friends it service that they're starting [00:23:00] or looking for aliens. And if you're interested in this, you can look up distributed computing on Wikipedia. You'll find a list of projects there in your, in your vision save. I clearly read this mission statement closely. It also says that innovation is fastest when it can leverage well encapsulated prior building blocks as well as lessons, which is essentially, this is the story of Code [00:23:30] and arguably the story of the evolution of life. Um, but, and you're seeing a lot of this now, right, where the building blocks are just getting bigger and bigger and it's letting people that, that don't, didn't, don't necessarily have computer programming skills or even web design skills. And all of a sudden you can create a blog, you can develop software. It's like you're almost using the web as a software development platform. You can manage a data center. Um, so can you talk a little bit [00:24:00] about what this will mean for people who are not necessarily computer programmers?

Speaker 3: Yeah. Um, so you're right that in systems, which is what the Rambam is all about. Traditionally we care a lot about building blocks that have really well-defined interfaces for other blocks so that we don't, and that's what encapsulation really means is that I can do whatever I want inside my black box and you can't tell what's going on inside of it. But you know that have you [00:24:30] asked her the question and in a well formatted way that I've told you how to ask, I will give you an answer and another well formatted way that you know how to interpret. So and, and we've done this at operating system levels and lower files system levels so that I can be given an assignment in my, in my programming class, in my systems class might like operating systems class that says, you know, develop a file system.

Speaker 3: And that means that it's, this is what runs in the hard drive of your computer and the rest of the computer doesn't need to know how it works. But [00:25:00] it needs to know that when it asks for a file from your, from your system, from your block, it's going gonna get one back or it's, it's going to get something that says, you know, sorry, I don't have that file. And so like you said, this is at a small level and we all use this everyday on our computers. People have all of these blocks talking to each other underneath every mouse click that they don't know about. And that's kind of the, the beauty of this. But what the rat lab is doing is kind of continuing that vision. And as you said, taking it to the next higher level when the, when the Internet came along, it [00:25:30] revolutionize everything in computer science because now instead of caring about one computer, we care about millions of computers and we can have them talk to each other really easily.

Speaker 3: So it made it made computer systems available and able to talk to each other, um, from anywhere in the world. So what we care about now is making those building blocks much larger so that, um, the guy in, so we're not really targeting the blogger or someone who, um, you know, uses technology at the very highest of consumer levels, but we're targeting the guy who [00:26:00] serves that person. So we're kind of one level of, of disconnection or abstraction away from them. And um, that really isn't that much farther from kind of the level of ignorance or not knowing if you will, kind of how the system works because that person doesn't have to have a computer science degree. Definitely doesn't have to be a phd to know how to t type up some PHP code or some, you know, Ruby code or some simple programming language that they can write this big idea of there. So then they can spend all their time to [00:26:30] talking to their friends and convincing them to use their application and not worrying about what our big black building blocks are doing underneath their every click.

Speaker 2: Uh Huh. So, okay, so finally, so which revolutionary it service? Are you going to lunch over the course of a weekend?

Speaker 3: Ooh, that's a really tough question. Um hmm. I think that when the idea strikes me, uh, I'll hopefully write it down and stick it away in my pocket for the next four [00:27:00] years because [inaudible] yeah. And hope that nobody does it because one of the, like the leading killers of PhD students in computer science at Berkeley is startups and, uh, so it kills you or it gets you out. Well, no, it doesn't get you a faster, it gets you to drop out faster. Right. Because

Speaker 2: at this point that's, that's a, that's almost like something you should, you know, it's like being a Harvard dropout. Yeah, exactly. [inaudible] Harvard though I should say I didn't even go to high. Right, okay. That's

Speaker 3: [00:27:30] if you, if you want to be a millionaire then that's the way to do it. But, but um, anybody who is in computer science for a phd at UC Berkeley doesn't want to be a millionaire yet. Ah, we're, we care more about the research are we, so we keep telling ourselves and then we do about the million dollars or so

Speaker 2: you can tell them from the, from the Berkeley Network on the, on the social graph. No, then maybe you could personally never clicked so much different from the Harvard one. Oh, right. Cause and then that's kind of familiar names. Yeah.

Speaker 3: Well known for, for a [00:28:00] startups, students leaving for startups. So in research we've kind of put that on hold for awhile until we get our PhDs out of the way where, you know, we, we're able to focus on research and not on making a lot of money. But that being said, if I had to, uh, if I had to go and do a stu a web startup, then, uh, I think that it would definitely be something that I would design in ruby on rails and it would probably have something to do with, uh, with cooking. Oh really? Yeah. I think, I think, I think it [00:28:30] would be neat to empower. Um, and this is one of like tens or hundreds of ideas that I've learned on with friends. Yeah. Yeah. Um, it'd be cool to empower people to see what other people in their cooking network in their kitchens right now and kind of, um, you know, know that if I get together with this guy and we want to cook for the afternoon, cause I like cooking that if I brought these five ingredients, yeah, I know he has these ingredients already. We could make this dinner. Yeah. And I can kind of select which, uh, [00:29:00] which dinners I want to invite my neighbors over.

Speaker 2: I thought of that visualizing that you're talking about social graphs, but take all the food in your, in your fridge and plug it into the thing and have it just visualized for you. I'll be in amazing things you can make with it.

Speaker 3: Right, exactly. And also tell you that you're only one ingredient away from,

Speaker 2: so therefore you should invite that friend over because they had it sitting in there. Huh. Okay. Good. Well it's been great talking to you, Andy. Thank you. Yeah, I had a great time. And if you'd like to keep up with the word [00:29:30] [inaudible]

Speaker 1: that Andy and [inaudible] and the Rad lab are doing, I'll finally give you that URL. It is Rad lab.cs.berkeley.edu you've been listening to the graduate on k a l ex Berkeley. My name is Bethany Gerson. Please visit us on Facebook and join me next Monday from 12 to 1230.