A.M. Turing Award

Interview with James ("Jim") Nicholas Gray

1998 ACM Turing Award Recipient

United States

This interview was initially produced by Microsoft Research for their Channel 9 series of interviews with prominent researchers. The ACM is grateful to Microsoft Research for putting this program into the public domain.

Barbara**:** = Barbara Fox, Interviewer,

Jim: = Jim Gray, ACM Turing Award Recipient,

Mike: = Mike Harrison,

David: = David Vaskevitch,

Tom: = Tom Barclay,

Catharine: = Catharine van Ingen

?? = inaudible (with timestamp) or [ ] for phonetic

Barbara: Hello. I'm Barbara Fox and I'd like to welcome you to *Behind the Code*. In this series, we feature Microsoft employees who have achieved great things. As your host, it's my goal to uncover passion, insight, decision making, wins, losses, and key learning points as they relate to a successful career. This interview will not focus on technology but rather on the person behind the code.

Jim Gray is a Technical Fellow in the Scalable Servers Research Group and Manager of Microsoft's Bay Area Research Center, or BARC. Jim has been called a giant in the fields of database and transaction processing computer systems. He is a member of the National Academy of Engineering, National Academy of Sciences, and the European Academy of Sciences, and is a Fellow of the Association for Computer Machinery. He is also the editor of the Morgan Kaufmann Series in Data Management Systems. In 1998, Jim was awarded the ACM's prestigious A.M. Turing Award.

Before joining Microsoft, Jim worked at Digital Equipment Corporation, Tandem Computers, IBM, and AT&T. He is the editor of The Performance Handbook for Database and Transaction Processing Systems and co-author of Transaction Processing Concepts and Techniques. Jim holds doctorates in computer science from the University of California, Berkeley, the University of Stuttgart, and the University of Paris.

Please join me in welcoming Jim Gray. [applause] Jim. Thanks for coming.

Jim: Hello, Barb.

Barbara: Thanks, Jim. Thanks for coming.

Jim: Great work. [laughs]

Barbara: [laughs] Let's get into some serious questions here. Right now, you're working on a really fascinating project. It's basically the SkyServer. That's part of a larger initiative called eScience. Can you tell us about that?

Jim: Sure. One of the reasons I came to Microsoft is that our so-called strategic intent is "Information at Your Fingertips." What that means for knowledge workers in the future is vast amounts of information that they in fact are struggling to understand. I mean it's not that the problem is that the information isn't at your fingertips. It's just that there's petabytes of it out there. How the heck do you get to it?

One context in which you can explore that is the area of sciences. The scientific community is gathering information at a prodigious rate. Unlike the situation at Walmart or the situation in many commercial enterprises, the science community is pretty public about what they're doing. If we work with say Target or Walmart, we can't talk to the other about what we're doing with them. If we work with the science community, we can talk to people in other… other physicists about what these physicists are doing, or in fact to biologists about what the physicists are doing.

I've taken the task of getting all of the science data online, getting it accessible so that you can easily understand what the information means, getting it cross-indexed to the literature, getting it cross-indexed to the other sciences as being a really good challenge for Microsoft as part of "Information at Your Fingertips."

Barbara: What's the hardest part of SkyServer?

Jim: Well, SkyServer is the astronomers, first. Somebody explained to me once that computing would be really easy if it weren't for tapes and users. The biggest problem with the SkyServer is the astronomers – that it's very, very hard to get people to agree. The fundamental thing we're trying to do with the SkyServer is build a conceptual model for astronomy. That means that you have to agree on what a star is and you have to agree on what a galaxy is, and you have to agree where the galaxy starts and stops, and you have to agree on how you're going to measure things. You have to agree like on the metric system. And you have to appreciate that, well, you think the astronomers all would agree on the metric system. Well, they still actually use this thing they got from the Phoenicians called the sexagesimal system – they measure things in hours, minutes, and seconds. [laughs] So simple things like that you'd think would be… "Boy, if we can't solve that, we're in big trouble." Well, we're in big trouble.

Barbara: A question for you. In your eScience work, a lot of people want to work with you because you know everybody in the industry. Also it seems like, moreover, you seem to pick the right problem. Everyone that we talked to about you says, "This guy has the knack for picking the right problem to work on." We had an opportunity to talk to Mike Harrison. Mike of course was your professor and mentor at Cal. We asked him a question about that. Here's what Mike had to say.

Mike: There's so many kinds of talent in the computing field, but Jim's got the ability to understand it all, to be good at everything, and I think to pick good problems. I think that's perhaps the most important thing, to pick a good project, to pick a good problem. We can invent things which are very hard and beyond us or almost all people in science. And Jim's had a way of picking problems in many areas and advancing those areas. That I think is really the significant thing. You can find many people who are great visionaries but never do anything, or great coders who… There's so much talent out there, but Jim's got breadth and depth, and that's a wonderful thing. He's also a good human being.

Jim: Wow. [laughs]

Barbara: [laughs] Quite a compliment. So Jim, how do you do it? I mean, how do you pick the right problem to work on consistently over your career?

Jim: Well, so Mike Harrison was my thesis advisor. We actually worked on complexity theory and "How complicated is it to do things?" The interesting thing about theory is that there is in fact no "simplicity theory." There's only complexity theory. The goal is to find something that is simple enough that you can actually make progress on. Many people show up at my doorstop, as you say, and say, "Help! We've got a mess on our hands. Can you help us clean up the mess?" The typical answer

is “No, I don’t see a way of cleaning up the mess.” Occasionally you see something which essentially is speculative, but you say, “If we could solve this and this and this problem,” and you can enumerate the problems, “then we could make an advance.” So I think the key thing is to go into a project with an idea that “If you could do this and this and this, then things might work out.”

And you have to have some theory about how you might approach those things. We all are optimists. Most of us wouldn’t undertake the projects we’ve undertaken if we knew how hard it was going to be. Just, I mean, “How hard could it be, right? What’s the worst thing that could happen?” Well, you can’t imagine [laughs] the worst things that can happen, how hard it’s going to be. On all these projects that I’ve been through, it’s much harder than I thought when I started, but it looked easy when we started. So I typically work on problems that I think are solvable. I mean that’s about the only way I can describe it.

Beginnings

Barbara: Let’s go back to you as a child here, try to figure out how you got to where you are today. You were born in San Francisco in 1944. What is kind of interesting is your first language was Italian and you spent the first three years of your life in the American embassy. That’s an unusual childhood. You want to tell us how that started?

Jim: Yeah. My dad was in the Army, World War II, and he distinguished himself. When he came out, they gave him a plum job working in the embassy in Rome as an intelligence officer. We had a villa and his job was to entertain people and figure out who were the good guys and who the bad guys were, and gossip and find out what was going on. Italy at the time was considering swinging to the very, very far left and becoming seriously communist, and he was trying to forestall that as were many other people in the US military and the US government.

Barbara: Jim, what did your mother do?

Jim: My mother was a schoolteacher, taught third grade for many, many, many years, and is now retired.

Barbara: Do you have a sister?

Jim: Yeah. My sister Gail was a CPA. She now lives in Mexico.

Barbara: Well, let’s skip to college. You went to the University of California at Berkeley. You went to a school at a time that everybody will remember as “the ’60s.” Basically that was the Vietnam War era. It also I think at Cal

was the center of the free speech movement. What was it like to go to school there at that really turbulent time?

Jim: Well, Berkeley is a great university to this day. It was a great university at the time. Multicultural, multinational, lots and lots of different ideas. There's a quadrant of the campus which is full of nerds. It's all physics and chemistry and engineering and mathematics and so on. But there are three other quadrants that are quite different. I found it to be just a wonderful place to get a good education. The American education system is kind of strange. We screw around till we get to college and then we start learning. It's more or less a socialization process till then as far as I can tell. So I had an awful lot of catching up to do and had to learn a lot of science and a lot of mathematics and to learn to read and write in fact.

Barbara: You started as a philosophy and math major. Then I believe that you really looked at Russell and Whitehead's Principia Mathematica, saw how computers were used in that context, and that might have been the very beginning of your real interest in computers. Was it?

Jim: Yeah, yeah. I mean fundamentally I was interested in how we understand things and thought philosophy was the right department to be in to understand epistemology. The problem is that the philosophers were using the same stuff they got from Lewis Carroll. They were using modus ponens and predicate logic as their approach to representing knowledge, and it just didn't scale. It wasn't going to work. It was clear I think probably to them that it wasn't going to work, and certainly to me. I was looking for something else and Principia Mathematica was recast by Russell and Whitehead into mathematics, and then I think it was Newell and Simon came along and proved most of the axioms… most of the, well, theorems in Principia Mathematica using a computer. And it was clear that they had managed to represent that information in a computer and in fact were manipulating information in ways that was very, very promising. So I basically caught the bug and said, "This looks like the way to represent knowledge."

Barbara: Well, your timing was great. You ended up working on the CAL Timesharing System. That was a very early capabilities-based operating system. But you had a lot of people around at that time, like Charles Simonyi, I think Peter Deutsch were there, Butler Lampson. I mean these people are pretty much right now legends. What was it like sort of growing up with the greats?

Jim: Just the way it is here at Microsoft.

Barbara: [laughs]

Jim: I mean they are just ordinary people, actually. [laughs] We were all equally confused. They're very bright. I mean I remember how quick each of them was. They were a lot quicker then than they are now, actually. But Ken Thompson was just an ordinary guy. He just hung around in the Computer Center at night and he'd spend a lot of energy trying to get the tape drive to march across the room by writing this program that would spin the tapes back and forth, back and forth, and see if the tape drive would… [audience laughs] I mean we were basically kids having fun.

IBM

Barbara: You ended up graduating. As a matter of fact, you were the first computer science graduate, PhD, from Cal. You went to IBM in Yorktown Heights, New York. Your first project I found really fascinating. At that time, there was a book published by a think tank called the Club of Rome. The book was actually 1972, Limits to Growth, in which they extensively used computer modeling. The idea was dire consequences for mankind. IBM assigned you to do that as a computer science guy.

Jim: Well, sort of. I mean that's not exactly how it happened. At Berkeley, we were very socially conscious and we wanted to see if we could use computers for things more than inventory control, and in particular, could we use them for some kinds of social planning? There was a guy at MIT by the name of Jay Forrester who had similar ideas. He'd been using computers for inventory control and he said, "Maybe we could use this for city planning," and then he did some of that. Urban Dynamics was a book that he wrote about that. Then he wrote I think it was called World Dynamics. And something called the called the Club of Rome got formed and there was this very dystopian view of the world, which is that "We're going to run out of resources in the year 2020, and this computer model proves it."

So I had re-implemented Forrester's models at Berkeley, and I was a postdoc at Berkeley for two years, an IBM postdoc, and I needed job. I went and I got a job at IBM in the general sciences group. And indeed, the people who were running IBM at the time, Watson was looking at the Club of Rome and didn't actually believe or like the conclusions that they had come to and was eager for research at IBM to work in this area.

So I came along and I could work in this area. It wasn't so much that they assigned me to work on it. It's I wanted to work on it and I had the credentials. And made some progress on it, but frankly the basic problem was the model was so screwy that Forrester had come up with and made such bogus predictions that there really wasn't much to say besides "This model is bogus." Doing a correct model is not something for dilettantes. I

mean it's fundamentally macroeconomics. The economists have been working on this for a good long time. They've made a lot of progress in the last… It's been 40 years. They've made a lot of progress in the last 40 years. But it is a very, very slow process, requires a lot of data gathering and a lot of very careful modeling, which frankly neither Forrester nor I was up to.

Barbara: You ended up leaving IBM. You went for a short stint with UNESCO in Romania, right?

Jim: Mm-hmm.

Barbara: Then you came back to IBM basically in the Silicon Valley. The most stunning part about that part of your career is that that was really the incubator time for relational database. I think Codd was there. What's interesting I think to a lot of people is that it was highly controversial within IBM.

Jim: Well, you have to appreciate that in the beginning, there was COBOL. Maybe they would say, "In the beginning, there was FORTRAN," but okay, for the EDP, electronic data processing people, in the beginning, there was COBOL. And COBOL had a Data Base Task Group and they had defined something called DBTG – "Data Base Task Group," DBTG. It was a database model for how to access data. It competed with IBM's product, which was IMS. It was a network data model and IMS was a hierarchical data model, and there were these wars between network, hierarchical, network, hierarchical.

And off in left field were these relational guys who said, "You guys are completely wrong. It's crazy to have such a procedural way of poking around through data. You don't get very much data independence, you don't get very much leverage, it's hard to write programs. You should be programming in set theory."

Barbara: [laughs]

Jim: Now you laugh. Right. That's what everybody else did. They laughed. But that's fundamentally what Ted was saying, Ted Codd was saying. He said, "It is much, much simpler to express the problems you're trying to solve in set theory than it is in DL/I or DBTG. Both express the information and express the manipulation." And everybody said, "Well, that may be true, but it'll be too inefficient. Computers are expensive, and you can't waste computers. And these problems are huge. I mean we're talking about thousands or tens of thousands of records, and you can't just use…"

Barbara: [laughs]

Jim: I mean literally. You got to appreciate this was the time of whole disks were 10 megabytes.

So it was a perfect research project. The challenge is "Could you make it efficient? Could you make it competitive? What if the trade-offs were different? What if people were expensive and computers were cheap? Then what would…?" Well, now, 34 years later, it's obvious. Codd was right. But he wasn't right at the time, he's just right now.

Barbara: One of the recurring themes that keeps coming up in your career is this was an opportunity, the first of many you've taken, to really get down and dirty in understanding what's going on, not only just on the research side but on the product side. But at that point, you did a lot of foundational work, you wrote a lot of papers that have really made a huge impact. So a couple of those themes, I mean you brought concurrency and transactions to people who were thinking about databases. But two of the ones that were really huge, well, one is your book, the book on Transaction Processing, which I, if you guys can see this…

Jim: [laughs]

Barbara: …pointed to Jim is still $70 used, okay, which said something. [laughs] One thing from the book and from that time was predicate locks, which was a paper, and a concept called ACID. Can you start out and explain some of those and why they were revolutionary then?

Jim: Sure. There are a lot of points in that question.

Barbara: [laughs] Yeah.

Jim: First, people were building database systems. They worked and they had concurrency. Not just that, there were a bunch of people in academe who'd been working on concurrency. The people in academe who'd been working on concurrency were primarily concerned about improving the throughput of the computer by doing things in parallel. The canonical thing that they worked on was matrix multiply. They wanted to do matrix multiply in parallel, and they figured out that if you did matrix multiply in the following order, you got a speed-up. But the goal was always to get the right answer, and when you multiply two matrices together, there is only one answer – it's the product, the determinant of the matrix.

Okay. We come along and we are doing database things where transactions are arriving, people were making requests to the database, and there is no right answer. There are wrong answers, but there's no

single right answer. So we were trying to figure out, “Well, how do you actually say that?” The answer is, well, there are certain invariants, there are certain properties you want to preserve. Like if it’s a theatre and you’re selling seats, you don’t want to sell the same seat twice to different people. You don’t even want to sell it twice to the same person, but okay.

So we tried to come up with a theory that described or a set of rules that described the kind of concurrency that could be allowed. In retrospect, it seems really straightforward. At the time, it wasn’t exactly straightforward. And in fact, there were lots of different approaches that people took. Some of them have fallen by the wayside, some of them have prospered.

We concluded that if you did the following things, then it’s as though you ran one transaction at a time, and running one transaction at a time is not going to have any concurrency anomalies. So if you run things in parallel and you get a behavior that’s identical to some serial schedule, some “running one thing after another after another,” then you don’t have any concurrency anomalies. Okay? Everybody can understand that. It’s pretty straightforward.

Then the question is “How do you get the maximum concurrency and still preserve this appearance of sequential execution?” We developed a bunch of strategies for that that all are generally called “locking.” Just what do you keep hidden until or what do you block people from doing until the previous transaction is completed?

That’s the concurrency stuff. We implemented that, and there was a lot of interplay between our implementation and other people who’d done implementations, and us learning from them and them learning from us. Then somebody came along and said, “Well, what are the properties that you really want of transactions?” Andreas Reuter in fact, the co-author on this book, is the guy who coined the term “ACID.” It’s a pun on the fact that his wife hates sweet things and loves vinegar. It’s basically that the transactions should be Atomic, they should be all or nothing; they should be Consistent, they should transform the database from a correct state to another correct state; that once the transaction completes, it should be Durable, and that’s where the “D” comes from, that its effects should persist forever; and that the transaction should run as though there are no other transactions executing, so it should run in Isolation, and that’s what “I” stands for in “ACID.” So two ways of thinking of it. It’s a pun on the fact that Christiana doesn’t like sugar. Another way of thinking of it is that it is this Atomicity, Durability, Isolation, and Consistency property.

Barbara: It’s become quite famous.

Jim: Yeah, it has. I mean people talk about the "ACID properties." It's also a pun on the acid test for the goodness and badness of things.

Barbara: Were these a series of aha moments? Is that how you work? Or did it come to you all at once, or…?

Jim: Especially for theoretical things, there are moments where you don't understand and then you understand. You finally get the proof to go through or you finally get the crisp statement of the problem. So there were some aha moments there. And when you write codes, there are aha moments when you find a bug. [laughs] You've been chasing a bug for… I mean concurrency bugs could elude you for weeks and months, actually. When you finally find it, usually it's something fairly subtle.

Tandem

Barbara: Let's go back in your career again. Let's jump to 1980. You went to Tandem. Tandem was a distributed-system, fault-tolerant operating system environment, and called NonStop. That was quite a change coming from IBM.

Jim: Yeah, it was.

Barbara: What was the challenge going there?

Jim: Well, it's an interesting thing. People at Microsoft think they work for a big company. When I left IBM, it was a third of a million people. It was – what? – six times bigger than Microsoft in round numbers. It was also a much older company, so it was very stodgy. So I show up at this company that's got a thousand employees. I described it as "a computer company on a chip." I mean you could go downstairs and see them making the computers, you could go over there and see them writing the software, you could go over there and see them selling the computers to the customers. The president's office is over there, the manufacturing floor is there, the… You know? You were able to know people from all over the company. That was very, very educational. Learned a lot.

Also, the ship time was a lot shorter. I shipped the first code I wrote out of IBM about two years after I left IBM. I'd been there for… So it was 12 years and the first line of code ships. And about three months after I was at Tandem, I shipped some code. Now…

Barbara: What was it?

Jim: A text processing system.

Barbara: [laughs]

Jim: It did right justification and a few other things. You know, we have terrible problems. Slavery is illegal in America. But if you're working for a company and you're working on relational databases and you know a lot about relational databases and transaction processing and you go to work for another company, how exactly do you work on databases and transaction processing without violating all of the intellectual property that you know? It's just in your blood. It would be hard to write a program that doesn't have that stuff built in. So I personally have this sort of statute of limitations, which is in round numbers about two or three years, and I try not to work on… So for about two or three years at Tandem, I didn't work on databases or anything like that. I just worked on other things. I worked on a system dictionary, I worked on this text processing thing. And the reason for doing the text processing was just to see, "Well, how do we ship code here? What's the process? What's the programming language? [laughs] How does QA work? How does…" And it taught me all those things.

Barbara: Actually, this not working on… like you say, carrying forward, respecting intellectual property, that has given you a lot of diversity in your career.

Jim: Yeah.

Barbara: It's a huge asset to you, don't you think, over time?

Jim: Well, yeah. You can teach liability… I mean the reason they hired you is because you know all this stuff. [laughs] And yet you're not supposed to know it. So yeah, it cuts both ways.

I mean after the two or three years, I went back and started working on a very, very nice SQL system, which was fault-tolerant, distributed, and so on. I'm still very proud of what we did. It was a very nice system. We built a great team of people and did very cool stuff. But yes, it definitely encourages diversity. That is to say, I mean if you've got to take two years off and work on something else, there's plenty of things to work on.

DEC

Barbara: Let's jump to 1990. In 1990, you went to Digital Equipment, DEC. DEC at that time was starting to lose its pedestal as the premier provider of midrange systems and software. You went in as a lab manager, but you were also a manager, very much so in that role in your life. What was it like to go into that environment?

Jim: First, another IQ test I failed. DEC was in a power dive at that point. But I didn't know it and in fact most of the people at DEC didn't know it. There certainly were some people who understood it. I thought the Alpha was a great instruction set, a great chip. DEC was the second-largest computer company on the planet at that point. There were people like Wang who were having problems, but DEC actually seemed to be doing okay. Yeah, they were losing a little bit of market share to this company called Sun and there was this workstation stuff they weren't doing so well on, and these "PCs" were coming along and that was kind of problematic. But they had this minicomputer market that was really great. They had ALL-IN-1 and they had this "DEC gets it." They were selling a lot of IT software. Actually on the outside it looked pretty healthy to me.

And DEC was an interesting company. They had what's called a dual ladder. They had a technical ladder and they had a management ladder, and they more or less treated the technical people with respect, which is not true of most companies, most technical companies. Usually the managers are in charge and the techies are considered staff. For better or for worse, DEC actually let the techies steer to some extent. And frankly, the techies drove this company off the cliff, but that's… [audience laughs] I was sitting there at DEC wishing, "Gee, wouldn't it be great if this company had some marketing?" [laughs] "Somebody who understood that when you build it, you have to have a customer to pay for it."

Barbara: Jim, wasn't it at DEC that you first started performing your now trademark "stunts," benchmarks that really show real products and how they work? What drives you to do that?

Jim: Well, I think it really started at Tandem with trying to show off SQL systems running lots of transactions per second. But at DEC, we did sorting benchmarks and we continued the transaction processing kinds of benchmarks. More recently, the TerraServer is an example of a stunt. The work we've been doing with the people at CERN, moving data at a very high speed from CERN to Pasadena over the network is an example of a stunt.

All of these thing go through the product from front to back and find things that are broken, what's called the "guru gap" – that the gurus can get great performance, but you have to set this knob and this knob and this knob and this knob. We just try and figure out, "Well, exactly what do you have to do to get the great performance?" then we go back to the product guys and say, "You know, we should make that the default behavior. You shouldn't have to do all of that to get good performance." I've seen that again and again and again have high payoffs. Benchmarking work for transaction processing really dramatically improved the performance of

everybody's system, our systems and the competitors' systems. And a similar story with sorting.

Barbara: Also I think during that same period, your lab did the foundational work on what's now called the whole field of data mining. Can you show us in our very expensive props I thought you might have brought along with you to show us how that actually works?

Jim: [laughs] Yeah. Well, I'm not sure we did the foundational work for data mining, but the challenge that the world faces these days is "How are we going to use lots of processors and lots of disks in parallel?" The answer I believe is dataflow programming.

The props I use to explain that is imagine that you have lots and lots of data sources. So here's a data source. You can take the data and you can process it in various ways. One style of processing is what's called pipeline parallelism where you take data from here and you run it through some program and out comes the resulting data. So you can get parallel processing by pipelining data from one to another. The key thing here is that the data that's coming out here is uniform. It's like a relational database. The records are coming out in a very uniform way. And you can take and build fairly elaborate dataflows this way and get natural parallelism where this program is executing in parallel with this program, is executing in parallel with this program, is consuming data from a disk. Here, we have a program that's taking data from two data sources and producing some results. We can take those results and feed them into a larger web.

Here is an example of a parallel program that you could build fairly simply. The key thing about this is that there's actually no parallelism inside your programs. This program is sequential, this program is sequential, this program is sequential. You can debug these as though you'd be debugging a sequential program, but this whole thing is running in parallel. This is the kind of pipeline parallelism you see in a production line where everybody along the line is doing something slightly different, but in fact things are flowing along the line and being processed in a highly parallel way.

This is pipeline parallelism. There's another kind of parallelism which is partitioned parallelism, where you take this whole line and you replicate it. You can take this whole process, and if you have twice as much data, you can process twice as much by giving this stream half the data and that stream half the data. That's partitioned parallelism.

This very simple model of programming I think is going to revolutionize the way we do parallel programming. It's the core technology inside of

relational database systems and it's now beginning to appear as a core technology in many other places. If you look at SQL Server Integration Services is the name of it, it gives you a programming model for dataflow like this. If you look at what people are doing at websites, like Google talks about Bigtable and Sawzall as two processing systems. They're a dataflow programming model very similar to this where you're doing parallelism and yet your programs are completely sequential. And this is a really very good way of mining very, very large quantities of data.

Barbara: Although you've spent a lot of your career in the commercial side of research, I think people in academia credit you with contributing a great deal to the understanding of and creating a field actually of how algorithms work in transactions. You legitimized by writing a number of papers your own research and explained a lot to that community. I've heard a lot of people say that was the basis of the Turing Award. Do you think that's true?

Jim: Yeah, I do. Fundamentally, Mike Harrison, who we heard from earlier, taught me to write things down. An awful lot of the work I did was joint with other people – Franco Putzolu, Irv Traiger, Mike Blasgen. I work with a lot of very, very bright people, and on most of these papers, they were co-authors who were in my opinion equal contributors to the articles. But I wrote lots and lots of stuff, and they didn't write very much. So the fact is I got credit for a lot of work that was really the work of our group. I think when they decided to recognize somebody for the Turing Award for the contributions to transactions, I was the obvious choice because I'd done most of the writing and I was the front man. But there are a lot of other people who contributed to that work. Similar to the TerraServer – people think I did the TerraServer, and the simple fact is Tom did the TerraServer. I was the manager.

Microsoft Research

Barbara: Let's get you to Microsoft. In 1995, you went to a conference – you spent some time in academia in the middle – went to a conference and ran into David Vaskevitch, who is now Chief Technical Officer.

Jim: High Performance Transaction Processing Workshop, right.

Barbara: That's right. Since David starting recruiting you right away, we went and we talked to him. Here's what David had to say.

David: He has a great sense of humor and he's a very engaging person. I think that's a big component of it. I think one of the biggest things – and this is a rare quality, that people who have this quality all tend to be viewed as great – he has an ability to go back to first principles. A lot of people…

One of the things that I'm working on in general is converting Microsoft as a whole to be more intentional. When you think about what it means to be intentional, part of the definition of intentional is saying what you mean. Another part of it is meaning what you say. But there's a big part of it which is about knowing the reasons for the things you do. A lot of people don't, most people don't know the reasons for the things they do. You know, "I'm doing it because I'm doing it" or "I'm doing it because that's the way people always have done it" or "I'm doing it because it's part of the plan" or "because somebody told me to do it." Whereas Jim is able to take things back to kind of bedrock and "Why is a database interesting? Why is a transaction interesting? Why would you write code a particular way? Why would a customer want this versus that?" Jim's always able to explain those things in terms of the things that are really real in our lives.

Jim: [laughs]

Barbara: Jim, when you came to Microsoft, David had just written a piece in Datamation basically saying that Microsoft's challenge was running SQL on "big iron," meaning mainframes, [and "steam irons." 39:00] What was the biggest challenge you saw Microsoft face when you joined?

Jim: Well, again, just as when I went to DEC, I was clueless about what it was like on the inside. When I showed up at Microsoft, I'd heard about this duopoly, the Wintel duopoly. I sort of assumed that Microsoft and Intel had this plan and they were going to go forward. The first thing I learned was that the Intel guys didn't care about servers at all, that they weren't actually planning to build very big servers, and that the computers that we were working on were pretty modest. It's been quite a while for us to get bus bandwidth and other properties that kind of match our brethren who have… Well, I'm thinking in particular these days of the IBM PowerPC. So one of the challenges we faced is we had really modest hardware.

The other challenge we faced is that it was a desktop company and David was trying to get it to be server-centric. I remember talking to somebody from NetWare and asking how on Earth they could have essentially all of the fileserver market when Microsoft controlled the interface. He said, "They don't get servers. They don't understand that instructions on the server are precious, speed on the server is precious. Simplicity is the key to speed and they're a functionality company." So one of the challenges was to form, and I think David and Dave Cutler as well managed to form a group of people who are server-centric as opposed to desktop-centric and are very worried about the kinds of issues that come up in a server environment.

My goal, and I think the goal that David sketched in the *Datamation* article, was to do scale-out. For one reason or another, until very recently we've been doing scale-up, which is to say get our products to run on bigger and bigger and bigger, more mainframe-like systems. We are I think now starting to do the scale-out agenda seriously. But one of the challenges is I've constantly been saying, "You know, there's a lot more mileage in doing scale-out than scale-up, because you can go a lot further. There's always a biggest machine you can buy. There isn't really a biggest cluster you can buy."

Barbara: I'm going to ask you about machines here right away. You did an invited ACM paper and you used an analogy that I think we want to demonstrate here called "smoking hairy golf balls." We have a smoking hairy golf ball.

Jim: Yeah, absolutely. I've brought mine along.

Barbara: [laughs]

Jim: Here it is. The concept is that the speed of light is finite and a nanosecond is a foot. So if you buy a gigahertz processor, it's doing something every nanosecond. That's the event horizon. But that's in a vacuum, the processor is not a vacuum, and signals don't go in a straight line and the processor is running at 3 gigahertz. So you don't have a foot. You've got four inches. And the speed of light in a solid is less than that. So this is the event horizon. If something happens on one side of this thing, the clock is going to tick before to the signal gets to the other side. That's why processors of the future have to be small and in fact golf ball–size.

Why are they smoking? Well, because they have to run on a lot of electricity. The way you get things to go fast is you put a lot of power into them. So heat dissipation is a big problem. Now it's astonishing to me that Intel has decided that this is a big problem only recently, because people knew that we were headed towards this heat cliff a long time ago.

And why is it hairy? Because you've got to get signals in and out of it, so this thing is going to be wrapped in pins.

Now another interesting thing about this is that we actually haven't gone 3D with our processor architectures. Processor architectures are some integer number of layers, like 10 or 20. But we could actually make 3D things which would give us much better space density if we could deal with the heat problem. Probably in the next decade, the processors will be sort of on this scale and cooling is going to be the big problem for them.

Barbara: Well, before we move on, I'd like to ask you a question about… your role is really at the intersection of research and product. I believe you called getting ideas from research into product "tin-cupping." So you wander around and you ask the product guys what they want or what they can use. What is the most challenging part of that process?

Jim: Well, actually the challenge for a researcher is getting product guys to embrace your ideas. Frankly, a product guy has schedules and they have of course a product that they are doing. When you come through the door with a new idea, you represent risk. The managers are trying to minimize their risks. That's one of their key things. And also minimize dependencies. Dependencies is not something you want, and here's somebody coming through with potentially a dependency.

So, quote, "selling research ideas" is a full-time job for researchers. The "tin-cupping" aspect of it is that oftentimes a research project needs collaborators, needs help. Take the example of the TerraServer. We needed to have people help us with hardware, we needed to have people help us with support for the hosting. So we would go to various parts of the company and say, "This would show off SQL" or "This would show off clustering" or "This would show off HomeAdvisor" or "This would show off…" Finally, MSN decided that Virtual Earth was one of their main strategic objectives, and then there wasn't tin-cupping anymore. They said, "We want it," and that was great. But for almost eight years of the TerraServer's life, we were supporting it year-by-year by going around with a tin cup and saying, "Will you be part of this research project?"

Barbara: In your experience, how long does it take for an idea from research to really show up? And this is in product long-term.

Jim: It varies enormously. When we did the data cube paper, I went and talked to the SQL guys and about two months later somebody called up and said, "Hey, why don't you download this thing and see whether you like it?" I downloaded it and there SQL had implemented data cubes and it shipped about, oh, six or nine months later. So that's as good as it gets. [laughs] More typically, the TerraServer is 10 years.

Barbara: Interesting.

Jim: And there's everything in between. We did system mirroring for databases and that's in SQL Server 2005. The snapshot isolation paper that we wrote in 1995 ships in SQL 2005. So 10 years is pretty typical.

Barbara: Let's move on to another project you're working on now, and that is the TerraServer. I know every time we talk to you about TerraServer, you always do great attribution of Tom Barclay.

Jim: I do.

Barbara: Tom is a researcher who works with you on this project and you've said he's done all the heavy lifting, really. So we brought Tom.

Jim: Oh, great.

Barbara: So Tom is here, and…

Jim: No kidding. Hey, Tom!

Barbara: [laughs]

Tom: Hey, buddy. How you doing?

Jim: Nice shirt. [laughs]

Tom: [laughs] Yeah, where's yours?

Jim: They gave me a dress code.

Tom: "No TerraServer shirts"?

Jim: "No TerraServer shirts."

Tom: Thank you.

Barbara: Hi, Tom. Thanks for coming.

Tom: You bet, Barbara.

Barbara: Can you do a quick little on a huge project like this, a brief explanation of what the TerraServer…? I think everybody pretty much knows what it is. But can you say something about its impact over the time you've been working on it on Microsoft and on the industry?

Tom: Well, I guess the joke we always say about it, it's the project that keeps on giving and taking at the same time. When we first started, the problem was scale-up. Jim mentioned the Intel and Windows community really wasn't focused on large scale. So that's what got us all started on it, was to show off first the problems we had, then when we succeeded with each release of either SQL Server or Windows, keep going. And as luck would have it, every time we get time to turn it off and shut it down, it would be the next great jihad at Microsoft. Then it became four-node clusters and what was going to be something that demonstrated it at scale? It turns out

having something that was real and had real data behind it was a convenient thing. And as time has gone on, we've now moved on to scale-out as well. So it's an interesting thing, as Jim pointed out earlier on in his career, is that actually going off and doing the stunts and trying to actually show how you could do scalability simply is an ever-recurring and important theme in the company.

Barbara: So Tom, I understand it was quite an adventure getting some of the data for the TerraServer. In particular, your venture in getting the data in Russia. You want to tell us about that?

Tom: Well, it sure was, Barbara. There we were, two Californians and an ex-Berkeley hippie stomping around Red Square. The first day we get there, Jim's on TV in the Russian space agency on live broadcast, then in the afternoon, we're met with an AK-47 as we were escorted into the production facility. And true, if you've ever heard some of the stories about how business is done in Russia, we were at Danilov Monastery and we had dinner with our hosts. Here we are, there's 27 people in the room, very elegant table setting, and sure enough we're invited to give a toast, and the next person, and the next person.

Of course, Jim and I are down about number 13 or 14 into this whole thing, and I come to find out that vodka is a truth serum for Jim. He had done really great being a politician, and he started out his toast with "Well, when we first arrived here, we didn't trust each other," and you could see all the people with guns get excited. I kind of look at Jim, "Not now," and picked right up and moved right on into the great trust we had formed and now we have this wonderful relationship. And another 14 toasts later, we staggered out and into cab and left the country.

Jim: [laughs]

Barbara: Is it true?

Jim: Yes, absolutely. Tom never lies. [laughs]

Barbara: One thing I'm curious about. You've been with Jim since DEC, the DEC days.

Tom: Yes.

Barbara: Rather than just the technology, what sort of non-technological insight did you really get from Jim?

Tom: Well, it's hard to pick just one, Barbara. The big thing, and it's been a reoccurring thing, is Jim's ability to be able to really take really, really

deep, hard concepts and distill them down to something all of us mere mortals can understand and also act on in the whole thing. I can remember when Bob Supnik and Jim was very fascinated by the Alpha. We just, "How can we help? In what ways could we actually help?" And that's where things like a number of the benchmarks came out later on came, to just basically be able to demonstrate the value of the Alpha. So that is a key thing, is being able to take a really hard idea, and not only that, boil it down but also give clear-cut examples of how that does move the whole industry forward.

Barbara: Yeah, he's tremendous at attribution. He's incredible at attributing.

Tom: Well, as you know, "No good deed goes unpunished" is his motto, and…

Jim: [laughs]

Barbara: [laughs] Okay. Well, I thank you very much for giving us a little explanation of the TerraServer and your work with Jim. I know that you completely admire Jim.

Tom: I do, absolutely.

Barbara: And you followed him all over the Earth. [laughs]

Jim: I don't know about that. But hey, thanks for the TerraServer.

Tom: Yeah, you bet. My pleasure.

Jim: It's working actually.

Barbara: Thanks, Tom.

Tom: Can I come back to San Francisco? It's cold up there. [laughs]

Jim: Please. Please! [laughs]

Barbara: Thanks, Tom.

Jim: Thanks, Tom.

Barbara: Jim, actually we…

Jim: Wow. That was good.

Barbara: Yeah. [laughs] Surprise. Actually we talked to someone else in your group. We actually had a chance to talk to Catharine van Ingen.

Jim: Oh, super.

Barbara: I want to share with you what Catharine said about working with you.

Catharine: One of the things that I stole from Jim is “Never let the best get in the way of the better.” Every team that’s worked with me has heard me say it. Often as engineers, we try to build the best piece of technology, the coolest, the cleanest, the fastest, the best. And a lot of the time, that can be really good, but making some step pragmatically forward is a much better thing for everybody, because you learn by making that step. So it’s that maybe the best thing wasn’t really the best. So yeah, I think that’s definitely one of Jim’s…

Barbara: Jim, when we talked to you, you said, in preparation for this, that you had three goals and you measure them in years. One is papers, projects or programs, and people. How do you weight those and what are the metrics?

Jim: Well, people are most important. When people ask, “What are you proudest of?” you always say your family. And when you think about your academic career or your professional career, it’s your professional family. The way you weight that is if it’s your professional family, well, it’s how well they’ve done professionally and how well they’ve done as people. So I very much… I hope Mike is proud of me and I’m very proud of some of the people that I’ve mentored. That’s the people one.

The papers is pretty easy. It’s citations. And the programs, it’s an art form. You know when you’ve written a good program and you know when you’ve written a bad program.

Barbara: Well, Jim, you’ve got about a decade behind each of your great innovations. You’ve been at Microsoft about a decade and you say that you always move on to some crazy fringe idea. Have you got one that’s baking someplace?

Jim: Well, this eScience stuff is actually fairly new to me.

Barbara: Too new.

Jim: I’m really in the middle of it. And there’s no end in sight for it. If anything, it’s gathering steam. It may be time to step back and let the smart people do it now. [laughs] I’ve still got my hands full with that.

Barbara: Work–life balance, a question for you. You have a daughter, you have a grandchild at this point, you're married. What are your hobbies? What do you do when you're not coming up with an aha moment?

Jim: Well, I love the out-of-doors. I sail. I love to go hiking. I read a lot. I have friends. I spend time with my friends. But frankly I am very engaged in what I'm doing. I try not to add up the number of hours per week. It's a lot.

Barbara: Well, I understand you're never going to retire.

Jim: My plan is not to retire, but I hope they'll kick me out when I stop being useful.

Q & A

Barbara: Jim, now I'm going to ask you some questions that we ask everybody and see what you have to say. The first is what kind of advice would you give to people in the field?

Jim: Well, computer science is at the center of almost all the intellectual disciplines. There's a lot ferment in biology. If you drill down into it, it's genomics and it's in fact computer science. It's possible to be at the center of almost any intellectual discipline by being in computer science. So the first thing is be excited about the fact that you are in the center of things. But also the advice I'd give is that it's important to find something that you are excited about and to focus on that. Don't waste your life working on stuff that doesn't interest you. Life is too short.

Barbara: How would you explain your work to someone who is totally not technical?

Jim: I work at Microsoft and I try to come up with ideas and products that will make the company successful and let them continue to pay me to work at Microsoft.

Barbara: [laughs] And also, what in life would you compare to producing software?

Jim: We're craftsmen. We make products and it's amazing how hard it is to make a product. It's a craft.

Barbara: Another one is "You know you're a computer nerd when…"

Jim: Well, my problem is that I'll occasionally look up and realize that it's midnight and I forgot to have dinner. If you can get so engrossed in things

that you sort of forget to eat, it's maybe a bad sign. It certainly means that you're excited about what you're doing and pretty involved in it.

Barbara: Now I'd like to ask you to draw your favorite data structure. You have to draw it so we can all see it. And sign it when you're finished. You want me to hold it for you there?

Jim: Yeah, I do. That would be wonderful.

Barbara: Okay. Just don't write on my arm here. [laughs]

Jim: So I puzzled about this. I knew about this question in advance. I thought that my favorite data structure is the free pool. As you know, the free pool has a head, which has a next pointer. And it has things in the pool, let's call it "A," and A goes off and points to other things, and A has some payload. And the free pool is currently pointing off to A.

Now the interesting thing about this free pool is you don't own it. It's a pool. It's shared between you and a lot of other people. So first question is how do you put something in the free pool? Well, you go off and you new a "B," and you make B point to A. Now you want to make the head, which was pointed at A, you want to make it point to B. Well, if you just store B in here, all sorts of things could happen in the meantime, because somebody could have come off and for example taken A away or they could have added C in here. So you actually have to do what's called "compare-exchange," and atomically do this – "NG" I think. And this is an 8-byte pointer, a 64-bit pointer, so you have to do the 8 version of that, and you have to say "head.Next", which is really head, "ref head.Next". And you want to make it B, and it better be A. So you have to do something like that. You with me so far? That works fine and everybody knows that. And now we have B here. Fantastic.

What about DQ? Well, DQ is a damn nuisance. If you want to take B away, you want to make sure that not only is the head pointing at B but B is pointing at A. So you can't just do a compare-exchange A B. That won't work. I mean it will work, but occasionally it won't work.

So what you actually have to do is introduce in the head the next pointer and something called a Kilroy. And Kilroy is like the sign on the pyramid that says, "Kilroy was here," or a Sphinx or whatever it is. The Kilroy says, "Somebody's been here lately." Every time somebody does a DQ… NQs don't have to worry about the Kilroy, but everybody who does a DQ is supposed to advance the Kilroy by one. So the Kilroy starts out at zero and every time somebody does a DQ, it gets incremented by one. So you have to use the compare and exchange 16b, the head, and (A,1),(B,0). And we're going to have the… and the Kilroy's going to start out at zero.

The thing that's amazing about this is that I MS-Searched on the web and I found a lot of stuff about exatomic instructions. Lots of people have never heard of the Kilroy. I actually didn't find anybody who did this right. And if you look in the .NET runtime, there is no 16-byte compare and exchange because it's not on the optirun. So this is an interesting story. [laughs] And I learned a lot doing it, so…

Barbara: Sign it. [laughs] Can you see it? Thank you, Jim.

Jim: You're welcome.

Barbara: Really appreciate it. [applause]

Jim: And parenthetically, when people tell you that they're going to make multithreaded programming easy, you got to tell them about the Kilroy and ask them… I mean an interview question is to ask somebody what the problem with this is. This is one of the kinds of bugs that you find the hard way, the "hard way" being "Think about it very carefully and write the assertions" or "Debug it again and again and again and again," because actually getting this to happen, it's not going to happen very often.

Barbara: Thanks, Jim, from the Technical Community Network for being our guest, and thanks to all of you in the audience for coming today. Thanks again. [applause]

[end of recording]