John Wilkes Speaks Out

on What the DB Community Needs to Know About Storage, How the DB and Storage Communities Can Join Forces and Change the World, and More

by Marianne Winslett



John Wilkes
http://www.hpl.hp.com/personal/john_wilkes

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today [February 2004] we are at the Department of Computer Science at the University of Illinois at Urbana-Champaign. I have here with me John Wilkes, who is an HP Fellow in the Internet Systems and Storage Laboratory at Hewlett Packard Laboratories in Palo Alto, California, where his research focuses on the design and management of storage systems. John is a member of the editorial board of ACM Transactions on Computer Systems, and until recently he was a member of the Technical Council of the Storage Network Industry Association. John is an ACM Fellow, and his PhD is from the University of Cambridge. So, John, welcome!

Thank you.

John is not a pillar of the database research community; he is a pillar of the ***storage*** *research community, which is next of kin to the database research community. Ironically, though, most database researchers know very little about storage. We tend to treat it as a black box. In fact, John, I can sum up for you everything that most of us database researchers know about disks in six easy terms:* sector, block, seek, rotational latency, transfer rate, *and* RAID*---and that's all we teach about disks in our database courses.*

You forgot *capacity*.

No, we don't do capacity*! We're interested in the number of disk arms, rather than how much you can stuff onto each disk.*

John, you co-authored a wonderful paper last century that described the actual behavior of a particular disk in great detail. That's the March 1994 IEEE Computer *article, "An Introduction to Disk Drive Modeling," that you wrote with with Chris Ruemmler. Can you give us some highlights regarding the actual behavior of real disks today?*

The disk industry has been doing a spectacularly good job of continuing to provide more capacity with lower cost and greater reliability. They have also done a quite good job of increasing the performance of those drives, and they have been slowly making them more sophisticated. Back when we wrote that paper in the early nineties, SCSI disks were becoming the norm, and now they are de facto everywhere. The

complexities, features, and functionalities that the disk drive vendors have put into them slowly increase with time, and the result is that we now get spectacularly good value out of those disks. I am glad I am not in the disk drive business---it is a cut throat environment. But the hard work of the disk industry has left storage system designers and database people in a marvelous position.

I can see how capacity might be important for storage, in the broadest sense. But in the database world, we are more concerned about the number of disk arms and the amount of material under those arms, because performance is so important a factor.

I wish that was more true of the customers who still continue to buy storage based on dollars per gigabyte. Yes, I have agreed with you for a long time that performance is the most important attribute, but not everyone has managed to get to that level of enlightenment.

Are there things that the disk industry could do to really make big improvements there?

They actually are doing it already. You already see a bifurcation in the market between the large capacity, relatively slow drives and the much more expensive, hotter, faster, high speed drives that are optimized for the high end database world. So, the market is driven primarily by what people will pay money for.

When database people think about disks, they don't think about things like the fact that the disk arm first speeds up, then slows down when doing seeks; the big difference between short seeks and long seeks; and all those sorts of details. Should we be thinking about those things, or are those just details that are better ignored?

I've always wondered whether it might be better if the database and storage communities could get together to agree on a slightly more sophisticated model of the storage device behavior.

What features would be in that model?

In the same way that you people may not know about storage devices, we don't know about databases either. But my understanding is that the query models and the optimization algorithms that get used are wonderfully sophisticated algorithms on top of a relatively simplistic base of expectations about how the storage device will behave--

Oh, yes.

--and that seems almost backwards. For example, most people do not put databases on raw disk drives any more; instead, they put them on storage arrays, which have massive amounts of caching and pre-fetching and intelligent algorithms to try and do data layout themselves inside the boxes. And unless you are aware of some of those things, you could get surprised at run time.

I view the creation of a model of storage device behavior as something that neither community could do well by itself. We could probably do much better if we collaborate on trying to find some way of expressing what it is that the storage system is trying to do and what the database would like to have done.

Do storage companies have liaisons with the database companies and--?

Yes, certainly.

So maybe those connections will happen?

Who knows? Part of the problem is that this model and its underlying assumptions need to be brought out in the cold, gray light of day, and I think that is something the research community can contribute to.

So would you say that the research community is not aware of the issues that those liaisons would be looking at?

That is probably too broad an allegation. I have an association with Carnegie Mellon University, where there is research on how to do joint optimizations between the database and storage system. We can probably benefit from more of this kind of cross-community research.

Let's move up a level of abstraction, from the disk to the storage system. Most database graduate students know nothing about storage systems. They do not learn about them in their database classes, and there aren't any classes that focus on storage systems. In fact, there are hardly any professors *who focus on storage systems, yet storage systems are extremely important in industry. Big businesses spend a lot of money buying them. A range of companies build and sell storage systems, from behemoths like IBM through midsize companies like EMC and on down to startups like Network Appliances. In fact, IBM Almaden has about as many researchers working on storage as on databases. Yet there was not even a research conference that focused on storage, until a few years ago when the FAST conference was started. How could an area of such immense economic importance be so invisible in academia, both in research and education?*

I wish I knew. It is absurd. People spend as much money on their storage system as they do on their processors these days, but the amount of attention paid to the two is wildly disparate, at least in academia.

Would you say that the CS curriculum should include coverage of the storage area?

I think it would be a wonderful thing to do. As we mentioned earlier, the performance of many systems is dictated by the behavior of the storage subsystem---and by performance I mean not just an aggregate read and write rate, but reliability and generic quality of service. The storage subsystem is becoming one of the more complicated parts of many systems. Unless we can persuade people that spending time on this area is interesting and worthwhile, I think we will find that we have to rely on what the storage industry will do. It is actually doing a pretty good job, but I am sure we could do better, as we have done in other areas.

What would researchers do that is not already being done in industry?

One of the things that the academic community can accomplish is that to try things that would not appear to make immediate business sense. The industry is dedicated to doing what is going to be delivered next year, and that limits very much the scope of things that can be tried. One of the great things about the academic community is that they can go far enough ahead to say, “Hey, this is worth looking at, and maybe we don't have to have a business justification for investing in it.” Think of them as scouts, looking out in the world and saying, “What if we tried this? What if we removed that restriction?” The academic community has that wonderful degree of freedom that can benefit everybody.

What hot topics do you think people would work on if we had academics looking at storage issues?

I'd like to see more emphasis put on the large scale systems that people have in the real world. All too much energy is spent on the kinds of things you can do with a desktop system, with a single disk inside it running Linux. That is not the way most real databases and large data stores behave.

So what are those large things like? I think our audience doesn't know. What do they consist of?

Think terabytes on the way to petabytes, rather than a few gigabytes. I was having a conversation this morning with one of your colleagues, and he was talking about having large quantities of data. I asked, "So what does large mean to you?" And he said, "Oh, a few gigabytes, at least."

Oh, come on! Obviously not a database colleague.

This was a database colleague.

You're kidding!

He was interested in the algorithms that you get to apply to the data, and the algorithms were large and complicated rather than the data. For a real-world large-scale scenario, just add a few zeros on the end of all the calculations, move to a world where instead of single disk spindles we are now talking thousands of spindles, move to a world where your multiple data centers have to collaborate 24 by 7---they never go down---, move to a world where battery backup and uninterruptible power supplies are the norm rather than the exception. Those are assumptions that change many things. One of the things that has been neglected for many years---and I made one stab at it but there is plenty of work still to be done---is how to guarantee quality of service in that space. I think we need to try to move to a world where predictability is the most important parameter: if it works this way now and I like it, I want to work it that way tomorrow, and I will still like it.

Predictability is a great goal, but I know you also work a lot on making self-tuning storage systems, and in my experience, self-tuning systems tend not to be so predictable because they change their behavior. So how do you reconcile those two goals?

The point of being self-tuning is to achieve some end result which ought to be "the system is doing what I want," i.e., behaving the way I like, and that is a kind of predictability. So I view self-tuning as a way of responding to the vagaries of the environment and workload in order to be able to achieve a target business goal or, in my case, a quality of service goal.

So in that continuum between predictability and performance, you would rather skip the peaks of really excellent performance if it meant you could deliver things at a steady rate?

No, that depends on what you, the user, want. If meeting those peaks is more important to you than meeting the steady rate, you should say so and the system should adjust accordingly.

So, you can tune the relative importance of those two parameters, the consistency and the peak performance. Are there other tunable variables besides those two?

For example, is the system reliable? Does it store data in a way that is not going to get corrupted? Maybe the data is transient and I can make optimizations to get better performance at the cost of things going wrong once in a while. The system should not pre-decide the level of reliability available to the user.

Performance is the easy tunable parameter, the one most people have focused on so far. In some sense, we have made pretty good progress there. Let's work on some of the other areas, such as reliability, availability, usability, functionality, manageability. Very few systems are static and unchanging over their lifetime; instead, they are always being tinkered with, added to. Their requirements are being changed. And the easier it is to make those changes and accommodate those things, the better the resulting system is going to be.

The database community is very interested in having self-tuning database systems. What kind of techniques do you use to make storage systems self-tuning?

Probably much the same techniques, actually. I have long advocated the notion of a control feedback loop, where you take measurements of the system; compare those measurements against what you would like to be happening (which by the way, requires ways of describing what you like); design some kind of response, perhaps diagnosing some kind of problem; determine what is currently the most appropriate solution to apply; and then find some way to apply that solution, preferably in a non-disruptive fashion; get the system back to a state where it is operating again; and repeat the whole process. The only question is, how sophisticated can you be in each of those stages? We started off with really simple measurements of average transfer rates we were trying to achieve. The design considered how far across multiple spindles to spread the data. Now we are moving towards taking into account more things, like reliability. You can imagine the system responding, for example, if something within it breaks. If one thing breaks, that's not such a big deal, but if three things start breaking, maybe you should be rather more aggressive about making sure the reliability goals are being met.

When you see that the system isn't behaving quite the way you wanted, and you want to move it closer, do you rely on a cost model to determine whether the changes you are likely to make are going to move it in the right direction?

There are two different philosophical approaches. One is prediction-based: you put a lot of emphasis on being able to predict the result of a potential change, and you work quite hard to make sure those models are accurate and reliable, so that you can use them to explore potential alternative designs. If your model is right, then when tuning, you will probably get to where you want to be very quickly. My group has done a lot of work in that space, and put a lot of effort into trying to produce good cost models. So that is philosophy number one.

Philosophy number two, which I am now moving towards myself, says that we will never know the system well enough to model it extremely accurately, so let's do a good job of modeling but not bend over too far backwards to try for a perfect prediction. Instead, let’s do a good enough job that when tuning, we can say, “*That* direction is the one I want to go in.” And then perhaps lift some ideas from traditional control theory to determine how far should we go in that direction, and put in a responsive feedback loop.

The issue I have found in my own research is that if I require a very accurate cost model, then every system needs an installation specific cost model, and who is going to create that?

The system should create the cost model itself by learning. Observations are wonderful ways of deducing how the system is behaving. My own group started with our own very well-crafted, hand-tuned analytical models of the likely behavior of the system, and after a while we decided that this is crazy: it is too much work and it requires too well-trained people. So we instead moved to an approach where we don’t measure the heck of out something; instead we build a little table and use that to extrapolate, predict, interpolate. That is the approach we now use. It is more robust, the information is easier to gather, and as the system runs you can just add more entries into the table and make it more accurate for your particular situation. So I believe in this approach----models that get better on their own as you use them.

So you can predict the performance of an expected workload very accurately with a little table?

At least as well as the clever analytical models.

I see. What accuracy can you could attain with that kind of simple table?

We tend to be comfortable once we get to prediction errors of 20-30%, which for one of these complicated storage systems is pretty good. Remember again, with a feedback system, the cost model does not have to be exactly right, it just has to be good enough to send you in the right direction.

So when the storage system is tuning itself, what kind of parameters does it have control over?

The easiest parameter is how many resources get used: disk spindles, memory available in the caches, potential access paths (if there is more than one of them, which there often are for reliability reasons). The second parameter is the way those resources get used. There are lots of policies embedded inside the storage systems: what order requests get serviced in, how long are data left in a write buffer before they get flushed out to the backend storage device, what kind of pre-fetching algorithms should you use, when should you declare that a request pattern is sequential, and how should you place the data on the storage devices. The behavior of the outside of the spindle is different than the inside of the spindle, so the placement of two things on the same spindle can give widely different performance. You could start out with what appeared to be two sequential workloads touching two objects on the same spindle, that when merged are essentially random access with terrible performance for both of them, just because they continue to seek between the two objects. There are actually quite a lot of knobs if you look inside a modern storage device; they are incredibly complicated.

And those knobs will be interacting in nonlinear ways?

Absolutely, just delightful. [Laughs.]

If you have distilled your whole performance model down to a simple table, how can that table capture these nonlinear interactions?

You said ***simple table***. I just said ***table.***

Oh, what kind of table is it?

It is a multidimensional table. The space that you are operating in has one dimension for each of the knob settings you could have.

So you aren't just measuring*, you are* measuring the heck out of *the system. With all these different combinations of parameter settings, everything is being measured all the time.*

That is our ideal. I mean, why throw this data away? You might as well use it. As time goes by and you get more comfortable with the prediction accuracy, then you can perhaps tone down the enthusiasm with which you gather data and put it into the table. But at first, you may as well take advantage of as much of the data as you can. We have learned that people are not very good at predicting which axes matter.

We had a little piece of work that we never got to publish because two or three people and a summer student worked away for several months trying to built a predictive model for caching behavior on a couple of disk arrays, and we could never get it right. We were off by factors of 2 to 10 in performance behavior. We obviously didn't capture enough of the important magic parameters. The approach of trying to predict everything analytically works well for some things, but for others it has limitations.

Were you trying to model a preexisting storage system, or were you also building the system?

We had two systems we had just bought off the shelf. Standard--

*Shouldn't these models be created by the people who put the devices together and **know** what policies they are using for caching, and so on?*

You would think so, but this is in the business world. Those people haven't that motivation. Vendors provided models in certain places, but the people who had designed the systems thought those models were remarkably simplistic. Vendors sell a lot of disk arrays based on benchmarks, which push the system into extreme behavior on very simplistic loads. We were much more interested in realistic loads. We do a lot of work with trace replay from systems where we have measured the actual behavior, as opposed to synthetic loads. We have discovered that the synthetic loads that people use are not very close to real loads, except for the very special case of benchmarks. But real workloads are nothing like benchmarks.

Another open problem is the workload merging problem. You have one set of work, index accesses or something like that, and another set of table accesses. Now put them together. How they interleave is a very complicated description problem. And the prediction problem for that is even worse.

The sad thing about that is that if you were up at a higher level, maybe at the application level, you would know that information. But at the storage system level, it has been lost, and all you are getting is this worthless little low level trace, from which you are trying to deduce the high level behavior. Why can't you have the higher level tell you this up front, so you don't have to try to guess it?

I think we should ask the database people that. Why wouldn't they pass the information down? Intentions are incredibly valuable piece of information.

How would database people pass that down to you? (I think they should *pass it down, for the record.)*

Let's start really simply: just tell me that this access path that I'm about to do will be random, this one will be sequential. Just actually telling us about sequential accesses before they happen is probably going to be the single most useful thing to do.

One thing that database people have on hand already is the query plans for things that are going to be executed, and they could certainly share that information. I don't know if it would help you or hinder you.

It seems tragic to have all of that useful information gathered and then not passed down.

Thrown away.

To be fair, both communities are guilty, because we have not managed to sit down and negotiate an interface for passing this kind of information down to the storage system. The academic community can help in figuring out what is the right interface. How simple can the interface be, to get the maximum return value?

Also, I think there isn't exactly a normal form for representing query plans---it is more vendor specific. But from a query plan, you could tell which things might be interleaving randomly and which things were actually related, and could probably handle things a lot better.

The other interesting question is, when do you pass this information to the storage system? Clearly at run time when the query is issued, you have a quite a lot of information about what the query is about to do, but when you are doing the system design and provisioning and placing data on the system---that is well before you have actually run the system. Perhaps during the tuning process this information could be made available and could be used.

That is a nice segue into something we were talking about earlier, this notion of how if you are tuning at two different levels, what happens when you put the two together, because they could--

A mess--

--have been fighting each other. The only self-tuning OS I have run on was Windows NT. We were trying to do self-tuning, but we had a hard a time tuning anything because the OS kept changing its behavior.

[Perkily.] It was just trying to help.

Just trying to help, yes. And in the end, just like you said earlier, we would have been happier for the OS to be predictable and slower, rather than faster and inconsistent.

A recurring theme in the computer science community is that the people on top always want the thing underneath to not do anything clever, because they think they are in control and understand what needs to happen. When things start out, I think that's probably right. We made a huge leap forward in the disk community when the disk device driver for Berkeley UNIX came along, because it actually had a model of how the disk drive was going to behave and got hugely better performance. But then the disk drive guys moved on, and people need to let go of assuming they have complete control of the innards of the system. We need to move to a world where the user declares, "I want you to behave this way; I don't need to know how you do it, but the behavior I want is like this"---and then the storage system can optimize around that. We haven't made that transition yet.

In the case of the interface between database and storage systems, how would we specify that behavior? Would it be in terms of quality of service? For what kind of things?

We should specify the behavior that is most important for the database level: reliability, performance, availability, those kinds of things. They all matter, so they all have to be specified somehow. I am being a little vague here about precisely how that is done. We have done some work in that space, but we have not done the validation of putting it into a database and a storage system and having the two collaborate. That is something I would like to see done, as there are a lot of open-ended questions regarding how much better could we make the system if we are willing to exchange information across that boundary.

I don't have a model in my head for what database people would tell the storage system that they want their behavior to be. At the highest level, I could imagine handing a workload to the storage system and saying, "Here, run this and be sure you finish it all in a certain amount of time, with a certain response time, and don't lose the data." I can imagine that, but surely we are talking at a lower level.

I am going to let you know that I am making this up on the fly.

That is okay.

Suppose we pass the query plan down to the storage system and say, "I am going to do all these things in this order." Surely we could do better with that information. Now maybe that is too much information to pass down. The question is how much could we back off from passing down that entire query plan.

But how would you specify your non-functional requirements? Would you say, "Here is my query plan and I want it done in the next five minutes"? Would it be more like a real time system, then?

I think you might do that. I could imagine saying, "Here is the sequence of requests I'm going to emit. Here are the dependencies I have: I cannot issue that request until this one is completed, or this one is completed

plus so many seconds have gone by because I need to process the content. Execute them in any order that makes sense as long as it conforms to this set of dependencies." So that provides degrees of freedom: "You can do anything you like as long as it doesn't violate this constraint." Those degrees of freedom are what give you opportunities for optimization down below.

So the constraints would be perhaps total time spent and consistency constraints for correctness and maybe some constraints having to do with reliability and those other dimensions.

And probably things like the amount of buffer space available. The thing I have at the back of my mind is the experience that we had with disk directed I/O in the parallel systems community, where once upon a time we used to have the people using the data issue low-level requests to the backend storage nodes. After a while we realized that the backend nodes were the bottleneck, and the better thing to do was to tell them what we were trying to accomplish and let them do the sequence on their own.

And let them figure out the sequence on their own, yes.

Same idea now, just applied to the storage system of the database.

I think you may be the first person I have interviewed whose PhD is from a European university. If you were graduating right now, would you still move to the US?

That is a good question. I don't know. When I came over 20 years ago, the opportunities in the United States were much more interesting and attractive than in the UK; I was an operating systems person and the west coast of the US was where things were happening. These days the European computing environment is much more attractive. Cambridge, where I came from, now has little vibrant start-ups, with all sorts of activities around the university; "Silicon Fen" they call it. That provides much more attractive opportunities than existed back then. I now work for HP, and they now have a big research lab in Bristol. That is where I grew up, so it is another potential attractive opportunity. And the climate has changed a lot. It used to be the case that all of the interesting research in hard-core systems was done in the US, and that is less true now.

Do you see a globalization, at least in hard-core systems area research, or is that specific to England? You were mentioning places in England.

The globalization of talent: some people have now chosen to put buildings around where some of the talent is, as opposed to moving the talent to the buildings.

Recent years have seen some hard times for industrial research labs. What do you think the future holds for industrial labs? Will they become extinct?

I certainly hope not. And I doubt it. This is one of those business questions of how to choose to invest in your future. There are many different models, all of which appear to be viable. At HP, we have chosen a model where we have a core center research lab whose funding is protected from the day to day throes of getting products out the door. Other companies have chosen different ways, and they have been more or less successful depending on many different factors. But insofar as HP Labs --and other research labs-- continue to provide value to their investors, they will continue to exist.

It all comes down to that definition of value. So what is the value that you are providing at HP and that people can't live without?

Ink jet printing is the one we always point to.

And what year did you do that?

It is a while back. We need to do a few of those big contributions each decade. It is hard to do that, I mean, we are there to take the big risks. I used to have this conversation with the head of HP Labs, saying that if we are not failing in two-thirds of the things we try, then we are not taking enough risks.

So what have you done since inventing ink jet printing?

PA-RISC, IA64 Itanium, and a bunch of other things that are smaller activities and that do not show up so much. But there are 600 or so people now at HP Labs leading the way---

I see. But you did not mention anything in storage.

[Laughs.] I was trying to paint the broader picture.

Okay. Have any special things at HP come out of your storage group?

I like to think that we helped AutoRAID get out the door, which was one of HP's first smart disk arrays. HP has been quite big in the storage management arena for quite a few years, and we have had influence on the storage management plan. Have we had as much impact as I would have liked? Absolutely not. We can always aim for more. I am currently moving into the adaptive utility computing space, and I would argue that the research we have been doing in my group for the past five or six years has been right on target. We just happened to look at the storage domain first. As a proof of concept and a demonstration of what can be done with kinds of control feedback loops we were talking about earlier, the storage domain has been great. HP is now shipping products that provide these kinds of facilities, partly I think because we helped lay the groundwork showing that these things are possible and can be done.

Do you have any words of advice for fledging or mid-career database researchers or practitioners?

If you are not in this for fun or profit, what the heck are you doing here? Think about what you are trying to accomplish! The failing that I see is the people who get enamored of what I would call "science projects": little simple things that they would like to try to take to fruition and get across that published paper barrier. Yes, that is fine. We all have to do that, and occasionally it is wonderfully rewarding to take a simple idea and move it. But I would encourage people to take a step back and say, "Well that is great, but who cares? What else do they care about? How can I solve their *other* problems, rather than just assuming that this is a good solution?" Go back and see if you can find a broader set of questions.

If you magically had enough time to do one more thing at work you are not doing now, what would it be?

[Laughs.] Sleep.

Oh, you are going to sleep at your desk, are you?

[Laughs.]

That *will look good.*

Magically. You said magically. Okay: create more hours in the day.

But weren't you saying last night that the life of us academics is way too hard and we work so *hard, and here you are the next morning saying that* you *need more sleep!*

I like to think of it this way: you get to do all the things I get to do, plus you have to teach, plus you have to go through grant proposal writing. The reason I work hard is that I enjoy it, so that is not an issue.

If you could change one thing about yourself as a computer scientist, what would it be?

I would get better at persuading other people that the things I think are interesting are interesting to them, because what we accomplish---

Marketing.

No, that's unkind!

Unkind? I think it-- [Laughs.]

I think marketing is trying to persuade people to do things that they don't want to do.

Not necessarily. When you are showing the importance of what you are doing, such as when you are writing a grant proposal, I consider it to be marketing, but it is not an evil thing (if you think marketing is evil).

Okay. In that sense, the positive sense, it is not evil.

[Teasing,] [To camera man.] We have got that on the audio tape, don't we?

What you really accomplish, you cannot accomplish alone; you have to do it through and with other people.

True, absolutely.

And getting them as excited as you are---and it feeds on itself, right? If other people get excited, you do too.

Do you know how you could really build that skill and give back to the community? You are not going to like this; are you ready?

I can see something coming. I am being set up.

What you (and anyone out there who is like you) need to do is go to a funding agency and start a program in an area you think is important. In your case, that area might be storage, because the storage research funding out there now is tucked inside other programs--- they don't stand on their own. That is how you could both build that skill (because program managers have to convince the upper levels that their topic is important, and you know that storage is incredibly important, so it should be an easy sell) and then get that funding program in place, which would encourage more academic types to start looking at storage issues.

That is an excellent idea, but I may not be the right person for it.

Well, I hope someone reading this article is the right person.

That would be wonderful.

Thank you, John.

My pleasure. Thank you.