Natassa Ailamaki Speaks Out on How to be a Systems Researcher and How to Manage a Large Research Group

Marianne Winslett and Vanessa Braganholo

Anastasia Ailamaki

https://people.epfl.ch/anastasia.ailamaki?lang=en

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we're at the 2017 SIGMOD and PODS conference in Chicago. I have here with me Anastasia Ailamaki, who's a professor at the Swiss Federal Institute of Technology, better known as EPFL. Before that, Natassa was a professor at Carnegie Mellon. She's an ACM Fellow, a Sloan Fellow, and received the European Young Investigator Award, as well as ten Best Paper awards. After this interview, she received the Edgar F. Codd Innovation Award from the ACM SIGMOD in 2019, and the Nemitsas Prize in Computer Science from the President of the Republic of Cyprus in 2018. Her Ph.D. is from the University of Wisconsin, Madison. So, Natassa, welcome!

You work in two very different areas, new hardware and information management for the life sciences. Why those two?

It started that way. I started as a student. I worked with Yannis Ioannidis initially and then with David DeWitt when Yannis moved back to Greece. The work that we were doing with Yannis involved interaction with the Soil Science group, at the time, in Wisconsin. We were building a data management infrastructure for them. There were very interesting challenges to meet. I liked the topic, but then I ended up doing a Ph.D. on architecture-conscious data management. I also discovered my love for interdisciplinary science through these two projects.

Afterwards, as an assistant professor, I always thought that I should stop one of the two thrusts, but then it got more and more interesting on both sides. On the architecture side, we got a lot of interesting devices. Hardware trends, as you know, go upward and onward in many directions. There were (and still are) many problems to solve, and I had a lot of students interested. And on the other side, I also started working with astronomers, with mechanical engineers, earthquake simulation people. Then, when I moved to EPFL, I started talking with life scientists, neuroscientists in particular, clinical and experimental. The interesting problems just flourished. So, I ended up working on both sides. It's actually been very motivating to me.

In your research on new hardware, how can you be sure that your insights into hardware-related effects aren't an artifact of running student-quality software?

That's a very good question. The quality of student software varies wildly, and one cannot be fully sure that it's thoroughly tested. I get away with less work testing it very closely on the experimental side of my work: the numbers have to be generated and tested and re-generated and corroborated in so many different ways that the software that we write is mostly scripts to test what's going on and figure out whether, for example, the breakdowns of time reveal any bottlenecks and where in the machine the time is attributed to and things like that. It's not production software, it's not something intended for end users. It's meant to reveal how the software uses the underlying hardware, and that we can test in many other ways than just looking at the result and making sure it's correct.

On the other hand, when we're doing work for scientific databases for actual users, there we have more problems. There, a lot of times, we need the involvement of an engineer to make sure that the software is tested correctly, does the right thing for the user, is reliable and so on and so forth.

How have the changes in the hardware world affected relational databases up until now?

The hardware world has been giving us quite a few opportunities, taking away decision-making processes at the hardware level, and giving us, essentially, choice. The big example there is parallelism. We view its different facets as they come up: instruction-level parallelism, pipelining, now multi-core, multi-threading, and so on and so forth. But it's been pretty much the same, explicit or implicit parallelism methods revealing opportunities at the hardware level.

Now, databases, from the very beginning of time, have been called to use this parallelism that's available in the hardware. We are, more or less, trying to go towards the same general direction. However, it's not always the same because the details of how the parallelism is implemented, implicitly or explicitly, are important for the architecture of the system design that we will end up with for the final product.

The other important aspect, however, that's come up very recently is heterogeneity in the hardware, heterogeneity at the compute level, at the interconnect level, and at the memory level, more recently. There, we're really not ready to use what's available. I foresee a very interesting future in hardware/software co-design for databases, just because this is the first time, really, we get something fundamentally new.

Hardware/software co-design means that you design software while keeping in mind what the hardware can do. And you also, at the same time, have the hardware get influenced by the software when you make design decisions at that level.

Tell me more about that co-design.

Well, when you architect a database engine, you write code for it. When someone tells you what's going on with the hardware, the first reaction is "why should I care?" Right? Because there is an operating system in

between. There is other software and lots of stuff happening. You're writing middleware; databases are middleware. They're going to run other applications, but they're not really that close to the hardware. People will go as far as data placement, and then they will stop there. They won't automatically think that they have to really consider what's going on in the compute subsystem or the high-level memory subsystem.

And they should!

Well, you and I may think that they should. And I certainly didn't think so until I realized that they should, after reading and experimenting and realizing that the operating system actually does very, very little and for a very small part of the time. There's really nothing between the database system and the hardware that impedes very good communication between the two. But this communication can only be good, which means the database system really taking advantage of the hardware resources, if the database system actually is aware of the hardware resources. So, co-design means that you design both of the things at the same time. Hardware/software co-design means that you design software while keeping in mind what the hardware can do. And you also, at the same time, have the hardware get influenced by the software when you make design decisions at that level.

But from our point of view, what we really need to do is to, first of all, evaluate the current situation because otherwise, you will run into a chicken and egg problem, a vicious cycle: which is designed for what… We want database software to be sort of general-purpose. We don't want to completely and deeply vertically-integrate everything. But at the same time, we also want the hardware to be able to run a vast breadth of applications and not just database systems. So, there are interesting tradeoffs to be solved.

I find very rewarding to work with micro-benchmarks: look at really simple fundamental database operations, and figure out, when they are executed (and they're executed very often), how they use the hardware resources. I've been doing this with many generations of databases and many generations of several different brands of computers.

I need an example of a couple of those fundamental database operations.

Like, for example, an index probe or even sorting for larger scale or a join. These are the kinds of operations that you can look at. Now, why are these important at the micro-architecture level?

Well, you may be executing a very short transaction where an index probe is the main action, or you may be executing two-megabyte-long code for one query where the index probe is a millionth of what it does. But in both cases, the processor grabs a bunch of instructions, gets them through the pipeline, graduates them, executes them, gives out the results, then grabs the next bunch of instructions. So the index probe code is as much as the window "seen" by the pipeline of execution at the processor level.

What happens is there's this window of instructions that moves; that window is very, very small. By basically measuring what's going on with those micro-benchmarks, I found it pretty educating to understand how the interaction between the software and the hardware works, how smoothly it works, whether the processor was sitting there idle or whether it was actually doing useful work.

So, co-design, to go back to your previous question, means that we work out details in the architecture of our engine keeping in mind how the hardware receives both the instruction stream of the code that we write, but also how it will be able to feed the high-level caches with data in the memory subsystem, without having to thrash them, and how we can also, ourselves, sort and schedule those accesses today, the instructions, so that we can achieve a better result.

The phrase co-design sounded exciting to me because it sounds like we were gonna get some control over what the hardware was gonna be like.

I had the same impression and the same hope. It actually isn't completely hopeless. There are a lot of times where we come up with an intuition for an algorithm, and we test the algorithm. That algorithm works really well, but then there is a software part that could be done a lot more efficiently if the hardware could just implement a very inexpensive hint, for example, that could alert the software that an event has happened: the cache is full or something like that.

I have had meetings with people at hardware companies, at Intel, for example, where we had these discussions for a long time. There were actually very reliable results that we were presenting, that they were convinced that this would be a great thing. I don't know that any of these ever went to production because the time from when one person, no matter how high up the management hierarchy they are, realized that this could be a great thing and the time that this might make it to production overlap with different generations in the company roadmap. So, priorities change.

I have good news for you. Andrew Chien, just a few minutes ago, told us that we're gonna get our own SoCs (System on a Chip).

(jokingly) So, your own personal SoCs!

No. I mean personal to the database world. But I guess it's a long, slow process. You've probably been having those discussions a long time.

So, getting database-specific systems on-chip, I think it's a great idea. But this is just one of the things that will happen. I really believe we're heading to more–different futures that will all happen concurrently.

What are the other futures?

Database engines, transactional engines, will be different, depending on the hardware that they're executing on. This is a realization that comes from the necessity to move the line of abstraction, of independence, if you will, from the hardware, from the very low level that it's at right now to a bit higher, to obtain better utilization of the resources. It's sometimes simply impossible to do that, just because the resources at this point don't only give us more parallelism, but hardware also gives us a lot of heterogeneity. So, we get different kinds of cores on the same chip. And in order to use them, we have to really alter the architecture of the system.

Virtual and dynamic solutions only work up to a point. Then they impose so much housekeeping overhead, so long of a "case" statement, that you will not be able to execute them efficiently.

It sounds like it could be very painful for the database companies to handle all these different types of architectures unless they have a very clean way of…

It could be painful, but I view it more as an opportunity. First of all, the research community is going to be given a lot of new questions to answer. Second, as I said, it's a matter of raising the abstraction line that divides the hardware-oblivious from the hardware-conscious part of the system. So, it really means that there will be multiple products coming out, which means growing the market and getting more people to really like databases. I see it really as an opportunity.

For me it's a fundamental change of scene because heterogeneity has never been this accessible from us. I have been doing work with CPUs and GPUs or CPUs and NPUs for research long time ago, but the other processors that weren't the main CPU were treated like the poor cousin that we would enlist just to see what happens, get a result, answer an academic question we had posed ourselves but nobody really cared about in the industrial world. Now, this changes a lot. I'm very excited.

It's exciting times!

What changes do you see on the horizon in the kinds of applications that use relational databases? I mean, new applications.

Changes in the form of an application being one way now and changing in the future? I'm not sure I see any changes whatsoever. What I see, however, is new applications that come up, given that we are evolving, in so many ways, the infrastructure that we build that more and more things are possible. Now, for example, more and more systems involve just-in-time access to data. So, NoETL is a big wave. My startup is in that domain. We access data that just arrived in multiple formats, from multiple sources. We don't access it unless it's asked for, unless it's queried. That's a big change.

Applications are not ready for something like that. But from the moment that this sinks in, for example, there will be a lot more interesting opportunities at the application level, to take advantage of this stuff. Apps are too bound, now, to the ingest-then-query model.

What do you think of Spark Streaming and Storm as ways to handle that data?

They're great enablers of handling data on a distributed infrastructure. They free the user from having to worry about all of these things, like Hadoop configuration and all that nitpicking that one has to go through in order to make things work today at scale.

But since you have your startup, you must not feel that they solve all the problems that matter for these new applications.

As I said, they're enablers. They're mechanisms. It's up to the application or the middleware infrastructure to solve the problem. It's a tool. You have to use the tool correctly to solve the problem. It's not a policymaker in the software.

How are we doing in the database world with respect to energy efficiency?

We have been doing pretty well in investigating what we can do. I don't think we've hit home run yet in understanding the relationship between database management systems and energy efficiency because, in

a lot of ways, it's a moving target. We have industry that is very qualified to actually do whatever they can to drop the numbers in the power bills. We have research people at all levels, at the architectural level, even lower, and then on top of that, at the system level, systems researchers, and even database researchers that do work in the area. But all of this work is coming out at the same time. We're taking each other's work and trying to measure what's happening in our world.

I don't think we've hit home run yet in understanding the relationship between database management systems and energy efficiency [...]

It's an interesting and very dynamic research environment right now. We haven't really hit a home run yet, I don't think, even in understanding the problem rather than solving it. Because the problem is very close to numbers and quantifying such a problem requires a very big breadth of numbers that are constantly changing today.

But I think that we will have to solve this problem. Eventually, the rest of the factors will converge. We will know that unless we do something in the software that's fundamentally energy-conscious, these numbers are not going to go down, the costs are not going to go down. So, we will have to solve a more stable problem, hopefully, in the near future. It's certainly a very interesting area.

You are already viewed as a rock star researcher in Europe. I've been told, this week, that you are much better than you think you are in every way, which suggests that your fame is only going to increase. How do you deal with the demands that come with this?

I'm very happy to be here. I take demands as they come. Whenever I receive a request to do some work or give a talk, I evaluate it honestly. If I can do it, I do it. If I can't, I just can't. The biggest problem that all of us face at some point, very early on, even, is to be able to say "no". Right? I am very, very bad at that. I don't say as many "no"'s as I should and a lot of times it takes a toll on my other work, that I have accepted too much to do on the outside. But at the same time, I find it very rewarding. Even for things that I regret immediately having said "yes" to, then I realize that there was a very positive side to this after I've done the work. So, I've never really regretted doing something even if it meant that it was a lot of work. But I am trying to train myself to say a bit more "no"'s. We should all try to do that, I think.

Your research group is fairly large and you're known as being hands-on with your students. How do you scale up being hands-on to a group of a dozen people or so?

I have twelve students, ten to twelve. The number varies in that neighborhood. Then I have a few engineers because now I'm in Europe and European Union projects demand engineering work. I have post-docs and scientific collaborators, who come after their Ph.D. Then, I have interns and masters students. It's a healthy group of 20 to 25 people. So, I organize meetings with everybody on a weekly or bi-weekly basis.

Individual?

Individually, depending on demands and depending on my travel as well. So I end up seeing everyone at least every two weeks. As is normal, I need to see mostly the graduate students. They take priority. Then I see the post-docs and the engineers as needed.

I am proud to say that this is a great group. The people who apply and are admitted to EPFL, and the people who I receive and end up in my group, they are top people. My students are fantastic. Everybody, post-docs, engineers, my administrative staff, is great. EPFL is a very supportive place. They have a great infrastructure, lots of resources. So, I find that my day-to-day job there, at least from that point of view, is very, very easy. I don't see any trouble.

The male professors want me to ask you how they can find more female Ph.D. students.

I don't particularly try to get female Ph.D. students. I always hire on a merit basis. One realization that I had when I went to Europe is that intelligence is uniformly distributed. I was worried that I wasn't going to have as good students as I had at CMU, just because I was at EPFL. It turns out that I am getting just as good students. But maybe male professors should realize that intelligence is also uniformly distributed not only geographically but also gender-wise. Giving equal opportunities doesn't come automatically. Perhaps we need to work toward that direction.

That's interesting because I did not find that to be the case at Illinois. I was the head of the awards committee, and the awards winners were very disproportionately female (considering how few

women we had). So, I realized that, in fact, our average female was smarter, significantly.

I believe that, but I would believe the same if you told me that it was male. I think you are describing a snapshot. We should just forget about gender. That has been – if I have one big change that I wish it could just magically happen is that, upon entrance into my building of work, I could be a genderless person. My life would be so much easier. I think that a lot of people would agree with me, from both genders, that that would be a very, very good idea. However, human nature doesn't allow that. So, we have to work with what we have.

But as far as students are concerned, the one thing to remember – talking about working with what we have and getting a tight grasp of reality here – is that female graduate students, on average, are way less confident than male graduate students. That's something that comes from nature (it is the big word). I don't know how to specialize it more. But I want to work with the result. I've always learned to look at the result of my experiment and reverse engineer, not try to find the source. Unless the problem is solvable, I'm not interested, and I think that this problem is not solvable at the source. We need to treat the symptoms.

So, I receive this graduate student, who is way less confident than she should be. And I take it upon myself to actually work with her: channel her motivation the right way; make her understand what she's doing well, what she doesn't do well; and make sure that all her capabilities, at the end of her tenure of five years, whatever, in my group, reach the level that will allow her to have a good job and a good life.

But wait a minute. That's exactly what I'm trying to do with the male students, as well. It's just that some people, male or female, are better at some things and they're not as good at some other things. As professors, basically, our job is to have this person across from us once a week (or more often), and to try to help them surface their best qualities and apply them to whatever it is they are passionate about.

I like that summary. How do you get so much energy?

(jokingly) You mean like the energy efficiency we were talking about?

No! No! No! No!

I understand. I think I'm naturally pretty energetic as a person. I don't know how I am like that. But I have a very supportive environment. Not just at work. I described that already, and I couldn't have wished for a better environment. The first half of my career at CMU, the second half of my career at EPFL; both environments have been the most nurturing I could have ever hoped for. I cannot emphasize this enough.

But also, my personal life. I have my ups and downs, like everybody. I've had my troubles, like everybody. But every time in my life there's been a lot of good stuff happening. So, I try to be appreciative, even when things are not the best of what's going on, and start every day as a new opportunity. I know that sounds as a cliché, but we do have to look around and see that there's great, great things going on. And not just at work. Our families, our parents, our children, where we live, what we can do during one single day… There's a lot of good stuff.

That's a lovely answer!

[…] the beauty of computer science is that the size and time change the solutions to the same problem. We can actually innovate in different ways solving similar problems as our predecessors, but really pushing the envelope of science, and of technology of course, much further.

Do you have any other words of advice for fledgling or mid-career database researchers?

I don't know that I have so much advice, because advice is something that you're very sure is going to work for the other person and that's a very big crowd. But actually, what worked for me was to try to look back to the research that people did when things were relatively simple at the infrastructure level. I can only know how to be a systems researcher.

With systems, history is a very important education factor. I found a lot of inspiration in the work that the first database groups did. And I still do. I find myself going back and looking at it very often; more often than I read blogs. That's one thing: drawing inspiration from the mental ideas because the beauty of computer science is that the size and time change the solutions to the same problem. We can actually innovate in different ways solving similar problems as our

predecessors, but really pushing the envelope of science, and of technology of course, much further.

If you magically had enough extra time to do one additional thing at work that you're not doing now, what it would be?

Two things. One would be I would give a lot more talks to junior students, both undergraduate and graduate. Not necessarily computer science talks but general talks about what I feel a good technical talk should be structured like. Stuff that colleagues of mine do, as well, and I'm very grateful that they do, but I would like to also do some more service in that direction.

Then at large, not within what I do right now, I would like to get some education and perhaps even a degree in a different science. I've always been fascinated by architecture and material sciences, so I would really go toward that direction to get, maybe, a degree or definitely some classes there.

Do you mean computer architecture or building architecture?

No, building architecture.

Civil engineering. Cool.

Thank you very much for talking with us today.

Thank you very much. It was a pleasure.