Peter Bailis Speaks Out on building tools users want to use

Marianne Winslett and Vanessa Braganholo

Peter Bailis

http://www.bailis.org/

Welcome to this installment of ACM Sigmod Records 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 Peter Bailis who's a professor at Stanford University. Peter won the 2017 ACM SIGMOD Jim Gray Dissertation award for his thesis entitled "Coordination Avoidance in Distributed Databases." Peter's advisors were Joseph Hellerstein, Ion Stoica, and Ali Ghodsi at Berkeley.

So, Peter, welcome!

Thanks.

What is your thesis about?

My thesis looks at distributed databases – if and when it's possible to build databases that execute concurrent operations without incurring communication across replicas. As we saw the rise of geo-distributed cloud computing, it became possible to run databases in multiple data centers, a setting where the speed of communication is fundamentally limited by the speed of light. And so the question we asked was: can I run transactions and other kinds of operations in my database without actually having these different replicas communicate?

Now, we knew from a bunch of research dating back to the 70s and 80s that you have to pay the price of this coordination via synchronous communication when you use conventional serializable transactions. But with the rise of many new applications, like those we saw in the online services (e.g., the Facebook social graph, maintaining distributed secondary indices), we wanted to know: could we satisfy these new types of application demands without coordination, and make them faster?

We are the database community, but I think more broadly we're the data-intensive systems and tools community. So finding users that will actually give you feedback on what you're working on, that can potentially adopt the algorithms or even the software that you're producing is incredibly valuable.

Was it a point paying for them?

There was a bit of a culture war between the database old guard, the David DeWitt's and Mike Stonebraker's of the world and this new class of developers, who threw a lot of the conventional wisdom from database management systems out the window and built their own class of data stores. These "NoSQL" developers started rebuilding databases from scratch and saying: "We don't need transactions. We don't want to run with the overhead of transactions for a number of reasons, one of which is scalability." And so, in our research, we found that we can actually provide many of the guarantees these developers wanted for their applications, but without the overhead of the conventional protocols that they had given up on.

There was a really interesting interplay between these evolving application demands and the core ideas behind conventional protocols, which in many cases were very close to what we'd like in the coordination-free setting, but not exactly. That is, we'd still use protocols like two phase commit in the design of these coordination-free algorithms. But we didn't use them with conventional synchronization mechanisms like locks. We also used a lot of multi-versioning but modified conventional versions of these protocols to scale while still providing guarantees that application developers wanted.

So, would you say you're working on a NoSQL killer? Or relational database killer?

The goal of my work and in particular this thesis is to provide useful tools that help people work more productively with their data. I think that as a community, we tell our users a lot of things that they should do. What I'm personally interested is in helping build tools that our users want to use in the first place. In the case of my thesis, developers had an application specification. They didn't have protocols to implement the specification. However, it turned out there was a lot of interesting theory and practical algorithms that came out of listening to what they wanted to use. We weren't throwing away the old theory, but adapting it to these new use cases and actually bringing it to practice. And so, the "relational versus NoSQL" debate is a bit of a red herring.

What I'd like to see more of our community doing and one of the things I'm proud of in this work is starting to bridge this divide between classical protocols, like consensus and two phase commit, and the demand of modern applications today. And I think that if you look at how programmers actually interact with transactional databases today, they need dramatically different interfaces, abstractions, and semantics than what we've built in the systems we provide them from the last 40 years.

So, can we find those features going into commercial systems now?

The work has seen various degrees of uptake. Some of the work we did early on in consistency prediction with the Apache Cassandra database, we just learned recently it's just now on Azure's CosmosDB. And some of the protocols we did for the secondary index maintenance, we have an ongoing dialogue with NoSQL developers about putting these into their systems as well.

That's great. Do you have any words of advice for graduate students or recent graduates?

The No. 1 piece of advice I'd give for grad students or recent graduates is find people who have real problems working with data. We are the database community, but I think more broadly we're the data-intensive systems and tools community. So finding users that will actually give you feedback on what you're working on, that can potentially adopt the algorithms or even the software that you're producing is incredibly valuable.

And I agree with you completely but in your specific case working on a classic database topic, most of our readers don't have that kind of shoulders rubbing with Facebook and other big companies that are facing this problem because most of them aren't located in Silicon Valley and other hotbeds. So, what does that advice mean for them?

That's a great question. There are many types of users and Internet services are only one type of user. I imagine most listeners are at or near a university and there are a large number of people at universities that are dealing with these sorts of data-intensive problems of crippling scale. Maybe not multi-data center databases, but, for instance, some of our work at Stanford right now is working with folks in Earth Sciences. They have all this seismic data coming in, with literal decades of archives that they'd like to process with more sophisticated methods. But they don't have the computational resources or the algorithms to scale them up.

So, I think that, at almost any university, if you go out and you spend some time doing some needs finding with domain scientists, with large amounts of data or even small amounts of data that could be dirty or not correctly labeled, there are interesting problems there. In a sense, your prerogative as a researcher is to actually step away from classic database systems. Don't work on faster serializable transaction processing. Don't work on query optimization. Don't work on relational analytics. Figure out what people in the wild who aren't necessarily Facebook and Google need to build.

It could be your roommate who is doing her Ph.D. in Biochemistry or in Earth Science. Go talk to them and ask them, "Hey, what do you do with data?" If you think about it, this is really the golden era of data. Everyone has recognized the value of data and yet the tools we have for dealing with data are geared towards a very particular, conventional, buttoned-up world of relational data management are really not in many cases adequate or serving the needs of the people who need it the most.

Working with this class of users that's beyond just the Facebooks and the Googles of the world, the folks who can't afford to hire the teams of data scientists to build these models to maintain their data and so on – that's where a lot of the new action is.

Great advice. Thank you very much for talking with us today.

Thanks.