An Approach to Managing the Software Engineering Challenges of Big Data

featuring Ian Gorton and John Klein interviewed by Suzanne Miller

----

Suzanne Miller: Welcome to the SEI Podcast Series, a production of Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center headquartered on CMU's campus in Pittsburgh, Pennsylvania. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I'm a principle researcher at the SEI. Today, I am pleased to introduce two of our other researchers to you, Ian Gorton and John Klein. In today's podcast we're going to be talking about what's known as *big data*, and the challenges it presents for software engineers. Ian and John have developed a system for helping software engineers manage the challenges of big data or at least some of them. I don't know if you've got all of them yet. But first, a little bit about our guests.

In his work at the SEI, Ian investigates issues related to software architecture at scale. This includes designing large-scale data management and analytic systems and understanding the inherent connections and tensions among software, data and deployment architectures in cloud-based systems. John's research focuses on developing design and analysis methods for systems-of-systems and helping individuals, teams, and organizations assess and improve their software architecture confidence. John and I had a great time working together on the architecture confidence work a few years back, and I'm sure he's having fun tackling the challenges of big data. So, welcome Ian and John.

Ian Gorton: Hi.

John Klein: Thank you, Suzanne.

Suzanne: So let's start off by having you explain exactly what *you* mean by big data. What types of systems are we talking about that need, and exploit big data.

Ian: It's maybe worth thinking back a decade or so to get some perspective on this. As hard as it is to believe, 10 years ago YouTube didn't exist. Facebook didn't exist. Google was starting to boom but still really in its infancy. In the last 10 years, we've seen this huge growth in data that's become available. The things with YouTube are obviously video sources. Facebook, we're getting people contributing content to the internet. This has just led to an exponential growth in the amount of data that's available on the internet. Of course, this goes into other application domains that are perhaps more pertinent to what we do here at the SEI. So, for example, in a commercial or military jet, the amount of data collected during a flight is at the order of hundreds of gigabytes per hour and this can be...

Suzanne: Per hour? Not per day?

Ian: Per hour, per operational hour of flight. Yes. This can be used to do all sorts of wonderful things in terms of predictive maintenance and analytics and performance of engines. So, these increased resolution sensors that we now have—as well as social media, the internet, and many other sources—are just contributing to this need to collect and analyze huge amounts of data. Where as a decade ago there was the 1 or 2 petabyte scale repositories around, now, they're almost common. Organizations in healthcare and government are now having to build these systems which are incredibly challenging to do.

Suzanne: So, what kinds of challenges do these kinds of systems give to our software engineers when they're architecting them, when they're trying to implement them? One challenge is going to involve designing a system that can be responsive to these exponential increases in size and complexity of the data that's available and then potentially that it needs to perform its functions, right? So, how do you make these systems responsive, and how do you tell the software engineers how to respond differently because many of them were trained 10 years ago before this was something that was prevalent? I know I wouldn't know how to deal with big data if you put me out into one of those areas now.

John: So, Suzanne, the way that people have been dealing with data storage is using relational database management systems. These systems over the past 30, 40 years have grown. They provide a very nice abstraction. As a developer, they scale up to a certain point and you don't need to worry about how the data is distributed. You don't need to worry about availability. The database management system takes care of all of that for you. What has come up in the past few years is a new type of technology, No SQL, which can mean no sequel or not only sequel, depending on either way you interpret that. What these are are systems that are based on horizontally-scaled commodity hardware, so scaling out rather than up.

Suzanne: Adding to the number of servers.

John: Number of servers rather than…

Suzanne: Increasing the size of a single server.

John: …increasing the size of a single server. Right. Because of that, these systems are now inherently distributed, and distributed systems are hard. They're hard to build.

Suzanne: And could I say, is it correct to say that they are massively distributed, and sometimes the boundaries are not known as to what the distribution is?

John: Yes. They scale to perhaps tens of thousands of nodes, servers, working together to solve your problems. And as a developer it is often hard to know how that is physically organized. This intersects with the area of cloud computing where, as a developer, you have no knowledge at all of where your computing resources are coming from. That's managed all by your cloud provider. So, distributed systems are hard to build. They're also hard to test because a good distributed system is going to mask partial failures.

So, testing these systems in all of the various ways that they may fail can be very difficult. So, it puts more importance on developing an architecture that satisfies good design principles in analyzing that architecture up front.

Suzanne: So, what I understand is that you have developed a risk reduction approach to help engineers develop these kinds of systems with the big data sets. What approach do you use, and how did you develop that? What kinds of principles give us a good architecture for a large, big-data, kind-of-system?

Ian: Just to build on what John said, the issues with distributed systems are known to be difficult. For your average software engineer, they've not had to deal with these at the scale that new systems, new big-data systems, are forcing them to think about. In many ways, a good way of thinking about it is that issues of distribution and data consistency used to be handled at the system-level of the software system. The relational database used to handle these things. Now, these issues are being pushed up the stack, so that the actual application programmers themselves have to deal with these hard issues explicitly in their code. They can't rely on some third-party system to make sure everything stays correct.

That's what's really hard about this as well as the diversity of the software, the new types of software systems for managing data that are emerging. So, what we're trying to do with our approach is to get a systematic framework whereby an organization can slowly navigate through all of these complex issues, understand exactly what sort of problems they have and what types of data they need to manage, and then focus in on choosing a technology and design approach that's amenable to solving their problem. This technological landscape is really huge, and there's so many complex issues lurking in sight here that organizations just aren't familiar with. So, our approach is all about helping them navigate this potential mine field.

Suzanne: So, you're helping them to characterize the really relevant attributes of that landscape?

Ian: Yes, and the technologies that are most appropriate for doing the sorts of things that will solve that problem because all technologies are not created equal despite what the marketing tells you.

Suzanne: Well, and all servers are not created equal, and all database management systems are not created equally. Now that we're interacting with multiples of those across a single system, you've got to deal with all of that. As an engineer, if I don't know what those are, then that really hampers me in my ability to understand how to deal with those kinds of issues.

John: One of the things that we see is the technology has become much more specialized. The relational databases that we've used in the past were, to some extent, general purpose. Through tuning and optimization you could adapt a particular relational database to solve a particular problem. With this NoSQL technology, the products are much more specialized.

Suzanne: So, you would see a different product for healthcare than automotive systems or avionics perhaps?

John: Less by the domain like you just described and more by the low-level characteristics of the problem. *Am I trying to store unstructured documents? Am I trying to store data that fits neatly into a set of key value pairs?*

Suzanne: Am I doing streaming sensor data?

John: Steaming sensor data. *Do I have a graph that I'm trying to represent and navigate?* And, so, that's led to an approach called polyglot persistence or an approach where you, rather than using one single relational database for your entire application, you may store different parts of the data for your application in different databases depending on what's the best fit.

Suzanne: Right. So, you can optimize the way that that data is made available, stored, etc.

John: And, that has the benefits of increasing performance perhaps. But, you've now made your system even more complex by breaking it up into parts that may have very different characteristics in the way they scale, in the way they react to network partitions, and in the way that you deal with recovering from failures.

Suzanne: So, you've named this framework LEAP is that correct?

John: Yes.

Suzanne: And, LEAP stands for?

John: Lightweight Evaluation and Architecture Prototyping.

Suzanne: Okay, so, where are you in terms of your status with this? I mean this is work that I hadn't heard of before, so I'm assuming that this is relatively recent. Where are you in the cycle of this?

Ian: So, what we've done in LEAP is we've focused on a particular quality of attributes that are really pertinent to large data systems. The SEI has a long heritage in developing methods that spread across the quality attribute concerns of a range of applications. We're much more focused on scalability, availability, and consistency.

Other quality attributes get driven by these three factors within a big data system. So, in LEAP, we have an essentially an evaluation framework that enables an organization to characterize their problem space and their needs in terms of their needs to scale data and processing, how available that data has to be, the complexity of the analytics that are required, and how they can achieve the performance that their application is required to deliver. For example, we see in many application domains, architectures are being specialized at different levels depending on the latency of the responses that they need to provide. So, you can provide very fast responses to some data, but you can't do much processing to provide fast response to perhaps, streaming data. So, this requires a completely different architectural approach, a completely different data model, and different technologies potentially to an application that has to grab a large amount of data and run complex analytics on it and deliver a response in 20 or 30 seconds or even longer. So, understanding these tensions and how you can look at the needs of your application and then understand the consistency, availability, and scalability requirements is really what we are focusing upon.

John: And, then, going from there and developing some very focused architecture prototyping work. What we've found is engineers are curious. It's very easy to spend a lot of time and effort playing in the sandbox. You get these technologies. You install them. You start running some tests, and you discover things that you want to tweak; and see *If we change this, what happens?*

As an example, we were working with one organization where they varied the number of client sessions on the database. What we've found is there is an inflection point where the performance would start to flatten out. There is this natural tendency to say, *Well, how can we make it better?* There are so many knobs and levers to start tweaking that it's very easy to get hung up and try to optimize and you're only at the prototyping stage.

It's optimization and understanding. So, one of the benefits of the approach that we've taken, as Ian said, is going from understanding your requirements to developing some decision criteria and then doing prototyping to address those decision criteria.

Suzanne: So, it helps you to sort of bound the sandbox a little bit, so that you're more effective in that exploration? So have you piloted this? Have you worked with organizations using this?

John: Yes. We have been working with one organization, a federal agency that was looking at this NoSQL technology for a particular application. So, we went in and began with a discussion with their stakeholders, understanding—as Ian was just saying—what were the important qualities that they had in terms of availability, latency, and scalability. From that we developed a set of prototypes that we're in the process of executing right now. We're about halfway through that, and then we're able to feed back the data into their development stream.

Suzanne: Cool. So, I know here at the SEI, one is not enough. So, I'm assuming that you're looking for more collaborators in this area to help understand what the boundaries are of where LEAP can help where it shouldn't be used all those kinds of things that we typically do in our technology development here. Can you say a little bit about the kinds of collaborators you're looking for at this point?

Ian: Absolutely. Yes. We'd love to prototype and extend this approach with people who have real problems with big data. The collaborators could be in any domains really where they are capturing and managing ever-increasing volumes of data, and this is becoming challenging on their current IT infrastructures. So, healthcare would be an obvious one where this is becoming a really dominant problem. Many areas of the military—logistics, planning, and intelligence—all of these have very large big-data-problems which are really starting to stress on how people build applications.

Suzanne: And even things like commercial avionics and transportation would be another area that I can see being very useful for this.

Ian: And, we really like the diversity as well because our approach goes down to actual technologies. So, we don't just don't deal with big dataat a conceptual level. We actually look at real technology. So in our current work, we've worked extensively with databases such as, MongoDB , [Apache] Cassandra, and Riak. And, each time we go into a project, we can leverage this information if it's pertinent. If it's not pertinent, then we can still use the framework to gather the same information about other database technologies, which is then comparable within this framework that we have.

Suzanne: So, you're also developing an understanding that we don't have today about what some of the boundaries and limitations and criteria are for these technologies. So, that's going to be a wonderful thing in the future as you build up your own data set.

Ian: Yes. It's information that's informally known in the development community, but it's never been any rigor or systematic approach applied.

John: Also, we've been developing a test bed so that we have a version private cloud capability that we can use to host the testing, and some reusable processes and work flows so that we can get that jump started. We're also interested in—not necessarily doing this for an organization in terms of running the prototypes—,but working with organizations that have development staff and helping to guide their development staff, keep them on track, interpret results, that type of work. One of the benefits here is, if we do the evaluation for your organization, that's great for us. We develop the knowledge, but you just get a report.

If we collaborate with you in more of a coaching role, where you're doing the development for the prototyping, then you're building up that knowledge.

Suzanne: And, we're learning also about different strategies for teaching about this and coaching that works. This is what we do at the SEI. So, this is a very exciting, new thing our horizon, and I'm really really pleased that we got to talk about this today.

If you want to get more information on LEAP, the approach that Ian and John have been talking about, please check out their recent blog post at blog.sei.cmu.edu, then click on the Big Data category. Thank you so much for joining us today. This is a wonderful conversation, and I think we have some more in the future as you get some more information about some of your results.

This podcast is available on the SEI website at sei.cmu.edu/podcasts and on Carnegie Mellon University's iTunes U site. As always if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you for listening.

Ian: Thank you.

John: Thank you, Suzanne.