Stark Industries Case Study

Summary

Phil J. Łaszkowicz

Introduction

When I originally read Ken’s tweet, I was reminded of the continuous product placement in the Iron Man films. From Sun SPARC servers, through to Larry Ellison’s cameo at Stark Expo.

Not found

The idea that Tony Stark would use anything other than Java seemed ludicrous. He’s a bought and paid for Oracle stooge, surely. And then I thought some more, and some more, and… some more.

As a mostly satirical exercise, I considered my own experiences of over 20-years working with and for some of the largest companies on technology projects, and I applied that experience to produce a likely technology stack employed at Stark Industries, assuming (incorrectly) the Iron Man 3 timeline happened in 2017.

None of the following is a recommendation. In fact, if I was asked to produce a stack for what a modern Stark Industries / Iron Man would use today, it would be quite different.

J.A.R.V.I.S.

Starting with J.A.R.V.I.S. we have to assume the neural network which formed Tony's assistive AI was built prior to Berkeley's Caffe, and indeed prior to AlexNet. It takes time for such a sophisticated neural network to be built, through iterative improvements and recurrent learning exercises. So it's likely J.A.R.V.I.S. was built on a custom C or C++ based neural network. I'd suggest C++ the most likely language, as Stark probably utilized Nvidia's CUDA platform heavily, and C++ is often the preferred language on CUDA projects.

Over time J.A.R.V.I.S. would have taken on new elements and functionality, and was likely a very ad-hoc distributed convoluted neural network used internally at Stark Industries. The persona created by Tony for J.A.R.V.I.S. would have been one component more unique to Tony's requirements than the company as a whole. Take note Zuckerberg, Marvel's J.A.R.V.I.S. is not just home automation.

Of course, as time has moved on, so has the technology J.A.R.V.I.S. was based on. Many of the academics hired at Stark Industries were avid MATLAB users, with some dabbling in C/C++. Especially the aeronautical engineers. Although there's a lot of ugly code repositories featuring examples of the C/C++ in use, much of the academic work on the AI has moved to Caffe and TensorFlow with Python. Although Caffe2 is available, experimentally, none of the production models are running in Caffe or TensorFlow anyway, so there's little incentive to move again. The internal Stark Industries AI core technologies are system proprietary C/C++ products.

The Suit

Although the Iron Man suit is meant to also run J.A.R.V.I.S., it's in-fact more likely that a sub-component of J.A.R.V.I.S. is running within the suit, (likely on an Nvidia Tesla GPU-based solution – the suit is probably very toasty inside), and the rest of J.A.R.V.I.S. is running centrally on the (Oracle) cloud. There needs to be an element of fail-over and network redundancy support in-built, just in case, but the core features of the suit could be managed by a lightweight version of J.A.R.V.I.S. – disconnect yourself from current reality; remember, this is fiction folks.

Moving from the artificial intelligence for a moment, the rest of the suit needs to be as performant and energy efficient as possible. Especially if there’s a Tesla GPU running in there. So it’s ARM-based chips all round. The suit likely has a collection of ARM Cortex-M chips for general management and sensors, ARM Cortex-R for the base of J.A.R.V.I.S., and an ARM Cortex-A for the display interface. I would assume the Mali GPU would exist to off-load some of the heavier concurrent tasks too, like computer vision for identifying objects. The chip-sets used are likely to have been licensed and augmented by Stark Industries to have hardware-level support for specific features Stark needs.

As far as the primary operating system in the suit, although I suspect Tony used a custom Assembler & C-based solution, I would hope ARM’s mbed is in there somewhere, utilizing the secure memory capabilities to reduce the chance of threat-vectors being used in-flight.

The display itself is likely to use Qt (still primarily with C++) with OpenGL / Vulkan running on Mali. Although there would be a prototype React VR experiment on-going, I doubt Tony would be so experimental and unnecessarily GPU heavy in the suit.

The Enterprise

So we’ve covered the cool kit. Let’s talk about the enterprise. Stark Industries is undoubtedly Java-based, as discussed. I assume the company has a set of guidelines of what technologies are acceptable for production, what are experimental, and what are simply not acceptable.

I would assume the majority of the business uses Oracle Cloud-based services. In fact, we know some of these details because Oracle produced their own case study for us.

> *"Jones is reusing some Stark Industries Java application design patterns and frameworks and collaborating with the company's development and QA teams to deliver secure, device-independent Java-based apps to manage the custom business processes for this mission."*

In this, Oracle mentions use of Oracle Cloud HCM for managing resourcing. It's likely they'd be using Oracle CRM On Demand (High Technology Edition) for their sales management. Remember, they're Oracle bought and paid. Stark Industries isn't going to re-invent every component of their business. The majority of

Stark Industries data is going to be housed on Oracle Database, of course.

They'll have a lot of varying Java apps they've collected over the years running core elements of the company. They'll include a mixture of Scala and Java code, with Tony preferring Clojure personally. He seems like a functional kind of guy or gets personal pleasure from writing code most object-oriented developers wouldn't be able to grasp. Kotlin hasn't landed at Stark yet as these systems don't see re-writes. They're enterprise-level and have had years of maturity.

Client Solutions

For the client-facing web applications, Stark Industries has moved most of its stack to NodeJS. They found it quicker to work with and easier to hire for. I'd like to think they use TypeScript, but my gut tells me they're a Flow type of company. Naturally they’d also be using React, with a few half-built Vue projects sitting around. Again, React was easier to hire for as a more popular tool, and once React VR was announced it cemented the technology within the company.

I assume there’s a lot of intelligence that goes into the solutions Stark Industries produce. Not all of it as powerful as J.A.R.V.I.S. Apache Cassandra and Hadoop are utilized for managing big data, with Spark and Mesos used for scaling the computing and data clustering needs.

On the Horizon

So, Stark Industries have abandoned weapons and are more focused on impact solutions. As they should be. What does that mean for their stack? Well, it’s unlikely they’ll move from being

Java-focused anytime soon. Sure, their solutions are likely now being migrated to Docker containers, and Kubernetes has had some success within Stark Industries, specifically on Oracle Cloud and internal OpenStack solutions. However Stark Industries has a wealth of technical competency in Java and a huge code-base that works well for the industry with little incentive to move away. Perhaps a decrease in revenue from abandoning the defense industry may mean a reduction in licensing costs is required, but they can just as easily move most of their stack to Amazon Web Services, once they work out how to re-engineer their in-house Oracle data solutions.

Rust has seen increasing use for core file IO tasks, and a select few embedded projects, and there are some minor Go components now being utilized to manage infrastructure, but there’s little incentive for Stark to move from the technology stack they’re familiar with, so those components are purely considered part of the experimental side to the internal development efforts.

As for the suggestion on Twitter that they would be using .NET – as much of a fit as that is for McLaren Applied Technologies, a likely competitor to Stark Industries, they’re already knee-deep in the Oracle’s JVM, and Microsoft’s .NET is the anti-thesis of that. The closest thing to .NET I can imagine existing at Stark is perhaps the odd case of MonoDevelop being used on experimental Unity VR and AR projects. Even then, I expect Unreal Engine is a better fit with their considerable C++ experience.

Thanks

Perhaps I should have produced something more time-relevant, like an analysis on the convoluted neural networks that makes Star

Wars robots empathic, yet provides them with no status greater in their society than slaves amongst the republic, empire, or rebellion. Perhaps that's for another post. Hopefully you enjoyed this little diversion to end 2017, so thank you for reading!

Thanks to Daniel Cheung for the feature photo provided (link below).

Daniel Cheung (@danielkcheung) | Unsplash Photo Community

See the best free to download photos, images, and wallpapers by Daniel Cheung on Unsplash.