Editorial

Open Source Software (OSS) used to be taboo within Financial Services (FS), attracting the same frequent comments: *Who are we going to get support from? OSS is not serious: it’s just a bunch of hackers playing with code at night, we can’t possibly use that for mission critical systems!*

Fast forward to 2015 and OSS is everywhere. It has been adopted by FS organisations, some of which actually contribute to the community. OSS used to be acceptable only for simple low level utilities such as logging frameworks, but has now gone ‘up the stack,’ with freely available middleware and even business platforms e.g. OpenGamma, an Open Source risk management package.

In our second edition of Tech Spark, we focus on why OSS adoption is increasing and, more importantly, what this means for the way we build software in our industry. At Excelian, we strongly believe OSS is where innovation has been happening the last few years, and that it leaves traditional technology companies in the dust in terms of new ideas.

At Excelian, we are increasing our OSS activity. We have started multiple internal initiatives to contribute to the OSS community, from enterprise data fabric frameworks to modern user interfaces. A few weeks ago, the leads from our User Interface and Usability stream published a dashboard framework built in AngularJS on our GitHub. Within a few days, it had already generated a buzz. Dailyjs.com, one of the reference websites for JavaScript development, picked it up and community members have already started using and contributing to the solution... So stay tuned on our GitHub (https://github.com/Excelian) for further Open Source software made by Excelian.

*André Nedelcoux – Partner, Head of Technical Consulting*
Software has undergone radical change over the last 10 years. From the time when virtualisation was in full swing, Cloud has since accelerated to become the core of IT, now running 80% of the world’s computer estate.

OSS has now become fundamental to ensuring a competitive advantage. Commercial firms are now built around the notion that providing software for free is a good thing – providing that the adoption rate accelerates and positions Open-Source as a business fundamental.

Venture Capital (VC) firms are now investing millions into this model, and it is working. Today, billion dollar organisations such as Hortonworks and Cloudera are prime examples of this.

These OSS projects embrace the web-scale advantage of technology whitepapers published by giants such as Google, LinkedIn etc. and turn these into Apache projects or publish them via GitHub.

The epicentre of technology is not as it used to be ten years ago. It is now based around OSS rather than commercial vendor solutions.

Our technical experts have put together the following infographic to depict these changes. It shows how the standard FS technology stack grew from silos to span thousands of compute cores globally and that the software that supports the global economy now runs within a completely different ecosystem.
EVOLUTION OF SOFTWARE TECHNOLOGY

FRONT END
- Java Swing
- WinForms
- WPF
- Silverlight
- GWT
- jQuery
- AngularJS
- Bootstrap

ANALYTICS
- Bespoke Compute
- Vendor Compute (compute bound)
- Hadoop (data bound)
- Spark

APPLICATION FRAMEWORKS
- J2EE
- Spring
- AJAX / REST
- WebSockets
- Microservices

LANGUAGES
- Java
- .NET
- Scala
- Clojure

MESSAGING
- TIBCO Rendezvous
- MuleSoft
- TIBCO EMS
- ActiveMQ
- Kafka
- Rabbit

STORAGE LAYER (DB)
- SQL Relational Database
- In-memory data grid
- Cassandra
- Hadoop HDFS
- MongoDB

DATACENTER
- On-Premise
- Software as a service
- Public cloud
- Hybrid cloud

Excelian Tech Spark, H2 2015 edition: Open Source Technology special
Open Source in Financial Services: Why now?

What does Open Source look like in 2015?

Open Source has evolved significantly since the early Linux days, having transitioned to a more professional approach. Gone are the days when the model consisted of amateur software enthusiasts spending their evenings coding.

Engineers working for technology companies now develop the majority of the popular, large scale OSS solutions available. The explanation behind the shift in approach embraced by most vendors is the universally adopted Freemium model, where the OSS version is provided free of charge and premium features incur costs.

The .com companies (Facebook, Google, and Yahoo) were the first generation to change the face of OSS by open sourcing large-scale frameworks such as Cassandra or Hadoop. These OSS products were developed and tested at an unprecedented scale - often greater than that required within FS. Other players in the internet industry were quick to follow suit, often in conjunction with cloud technologies, leveraging the appealing combination of large scale on-demand infrastructure and middleware’s running at scale.

As a direct result of this increase in adoption, an ecosystem of wealthy companies funded by Venture Capital (VC) developed around these technologies. Docker, one of the success stories of 2014, raised $40M in funding, while Cloudera and DataStax managed £165M and £160M respectively. The amount of money invested in OSS is now colossal, with cumulative VC funding for Hadoop and NoSQL vendors breaking through the $1B barrier in 2013. These figures outweigh the average bank’s ability to invest in pure technology!

With organised contributions to OSS, overall quality has vastly improved, and there are now virtually no differences between an OSS solution and a commercially packaged one. However, OSS enjoys one key advantage: it is more transparent since everything is visible to the user. There are open, online communities involved with the products, making it easy to understand common issues and judge the level of activity around a product without having to take a vendor’s word for it.

Traditional brick and mortar technology companies are struggling to respond. The most forward thinking ones (e.g. VMWare) have announced new projects and contributions to Open Source, such as open sourcing of solutions previously licensed under a commercial model.

Cost efficiency is becoming the priority in FS, and OSS offers an alternative model to “consuming” software. Although not free as the name suggests, it is generally cheaper, and the option of using a free version before purchasing offers the chance to “try before you buy.”

It’s also possible to develop skills within an organisation, leveraging all the publically available material. From a skills perspective, there is now greater liquidity across industries for solutions which were once more vertically aligned - e.g. Big Data has now matured in the start-up, telco and consumer space.

WHY IS BANKING OPENING UP TO OSS NOW?

The latest key trends in technology are all fuelled by Open Source technologies. This is where innovation happens and it’s impossible to ignore it now

- **Big Data**: First Hadoop, and now all other NoSQL solutions.
- **Cloud**: Docker is about to revolutionise the cloud world and offer a standard cross-vendor container technology which Microsoft and Amazon have already embraced
- **New Front End technologies**: the two emblematic frameworks, AngularJS & Node.JS, are Open Source
Finally, most of these OSS solutions have been built with public cloud in mind. They are designed to be deployed across hybrid on-premise/public cloud environments, guard against failure and support dynamic scaling. As public cloud continues its expansion within FS, it will become increasingly critical to leverage solutions designed for it, rather than legacy vendor middleware.

WHAT’S NEXT?

The FS community has tiptoed around these solutions for the last few years, adopting some as direct alternatives to vendor solutions. Clients are now leveraging the new possibilities of these Open Source platforms, going beyond the original use case and re-visiting their architectures. Traditional database-based data warehouses are being replaced by Big Data solutions and are moving towards real-time; NoSQL database are being used to scale-out Risk Systems through hybrid on premise/public cloud deployments. This is the beginning of an era of innovation in FS… fuelled by Open Source technologies!

KEY POINTS:

• The advent of large-scale frameworks led to massive investment in OSS.

• OSS solutions became widely used thanks to improvements in quality.

• The expansion of public cloud continues to increase demand for OSS.

Andre Nedelcoux – Partner, Head of Technical Consulting
Noble Group is one of the world’s largest physical commodities trading houses. It has experienced tremendous growth over past 10 years, with revenue increasing from $7.95B in 2004 to $97.6B in 2014. It has also employed an aggressive growth strategy in terms of its geography, range of commodity trades and the type of business it undertakes.

This phenomenal growth has placed unprecedented pressure on Noble Group’s IT systems, leading to a variety of individual business systems, which in some cases only cover one particular asset or type of trading.

Over time it has become apparent that Noble Group’s IT needs an approach, methodology and tools to accommodate an ever-increasing hunger for data and structure. It should also generalise and commoditise business processes into generic functions across all business units. Most importantly, it should make data travel faster within the organisation, from day-to-day activities to consolidated group reporting and controls. Clearly, the business has grown dissatisfied with the constraints imposed by fragmented IT systems which do not allow cross functional business.

In 2012, Noble Group’s IT department commissioned a global data programme, leading to the introduction of a “Big Data” storage platform to act as a consolidated data repository for all business systems. This platform was tasked with providing unified access to data for all downstream integration, internally built applications and end user access. There was a choice between vendor solutions or seeking the help of OSS. The team settled on embracing the Open Source route and this article summarises our experiences on that road.

The following concerns are just a few of many that were raised. However, unlike with well-established vendors who have large presales and marketing workforces, there were not many people we could ask to respond to these questions. Nevertheless, the architectural team had to in the end – but more about this later.

On the other hand, the advanced capabilities of Open Source offered many advantages such as the ability to handle vast amounts of data in NoSQL databases, greater agility and speed of application development with AngularJS and Bootstrap. Amongst other things, overall project delivery risk reduction was particularly attractive:

- The technology team was able to “try before they buy”, a concept only available with limitations from commercial vendors.

- The Open Source community fulfils an important role of vetting and optimising software. It strives to reduce the rate of support without financial gain, allowing less efficient implementations to die off or be replaced.

- Some of the software has been deployed on massive scale, surpassing any requirements a financial institution might have – think Facebook and Twitter volumes.

- A relatively small development team was able to focus more on the functional side rather than building a reusable core component.

- Less code to maintain also leads to lesser dependency on talent with internal proprietary knowledge.
However, embracing Open Source concepts is not an easy ride. A capable engineering team is a pre-requisite for implementing such a strategy successfully. There are few Open Source end-user applications; instead, one might think of Open Source as a collection of foundation core components which reduce the total engineering work required. These components must be brought together into a coherent platform, and one must take additional care not to leave the platform dependent on individual Open Source components, in case any of the components are discontinued.

In other words, embracing Open Source is an investment, and by no means guaranteed to pay off immediately. However, provided there is sufficient scope of functional work, such an investment is attractive.

Our approach was to split overall development into functional user facing applications and a core platform which provides common capabilities. The functional side was inherently tied to the business, and thus proprietary by nature. However, the core platform was a collection of carefully selected and integrated Open Source components, tied into the microservices architecture.

Integration deserves a separate note here. It is no secret that embedding new technology leads to costly and sometimes complex integration projects. For the new technology platform, we adopted a uniform set of development languages, tools, release management and build. For external interoperability we implemented a unified messaging layer with message protocol supported in multiple languages including Java, Python, .Net, and JavaScript. In addition, we provided an SQL access layer to the raw data held within our storage platform using Cloudera's Impala. We paid equal attention to the ability to extract data as to acquiring original data across the firm.

While this may sound abstract and perhaps academic, the results surpassed any expectations:

- The development team was able to acquire all historical trade data from 5 trading systems with incomparably different data models. It also provided an online report, including hierarchical and aggregation queries, which was able to process over 5 million trades in under 300ms!
- A similar application to enable businesses to query costs held in the trading systems took only a couple of days to develop. This was mostly a configuration and data mapping exercise rather than an actual development effort.
- We introduced a uniform “Big Data,” NoSQL storage platform. This was capable of handling entire company data volumes with intraday incremental data update, as well as serving as a central data repository for all downstream systems. While data is stored in non-SQL databases, there remains an SQL based access layer allowing interoperability with vendor based systems which do not support any other means of integration.

CAUTIONS AGAINST USING OPEN SOURCE:

- Who will be on the hook if things go wrong? What is the support model and SLA?
- If no-one has ownership, does this mean the platform can disappear without prior warning?
- Is it safe and secure to use? Are we exposing commercial organisations to extra risks by using Open Source?
- What if we need to modify it? How do we make sure such changes are included into future evolutions of the software? Are we giving away proprietary knowledge and competitive advantages by contributing to Open Source community?
- What if it simply does not work? Does it mean we are stuck with deficient software and facing lengthy and expensive rework?
We defined and implemented a number of key principles throughout the entire architecture:

1. No component, whether Open Source, vendor or internally built, is irreplaceable. We assumed upfront that each will be replaced or upgraded at some point. We did this by wrapping each component into core APIs, isolating the rest of technology stack from direct dependencies between components.

2. Test early, fail fast, and move on if fail – we threw production volumes of data into databases and measured responsiveness.

3. We considered only components with significant implementation or usage base and community following. While we can take ownership of the source code maintenance should any particular piece of software cease to exist in the open source community, we would rather avoid such increases in support and maintenance.

So, to revert back to the concerns frequently raised over using Open Source software:

Q: Who will be on the hook if things will go wrong?

A: There are many options available if things go wrong, from submitting changes to the Open Source community to engaging third party companies which specialise in providing such support on a commercial basis. The worst case scenario would involve taking ownership of the code maintenance internally, which is equivalent to activating an escrow agreement on a failed vendor.

Q: Is it safe and secure to use?

A: Again, there are companies which will provide a security assessment of the OSS. Having access to the source code actually enables a better and more thorough assessment than relying on the assurances of a software vendor with a closed source code. Furthermore, the vast majority of software license agreements indemnify vendors of any consequential loss.

Q: What if we need to modify it?

A: As with support, there are multiple options, and one should not be scared to contribute back to Open Source. As long as the functional application development is cleanly separated from the core platform, there is little risk of giving away much proprietary knowledge. If you don’t contribute back, there are chances someone else will.

Q: What if it does not work?

A: Fail fast, and wrap components into APIs, so that replacement is easy and natural. Embrace the ever changing nature of the software (unless, of course, it is all fixed from hardware to software to protocols, and never going to change). Even spacecraft sent to the moon receive software updates.

In conclusion, Open Source strategy requires strategic planning and additional investment into a talented team. The financial and functional rewards make such an investment well worthwhile.

KEY POINTS:

- After a period of rapid growth, Noble Group embraced Open Source when building a “Big Data” storage platform.
- The switch to Open Source was no easy task, and required significant engineering work to bring components into a coherent platform.
- The results surpassed all expectations, proving Open Source to be a fantastic investment.

Anton Bakharevski- Head of Energy IT
David Haines- Head of Front Office Development
Shankar Vasudevan- Global Data Warehouse Architect

The open source software we use includes:

Cassandra, Neo4J, Elasticsearch, Impala, AngularJS, Apache Mesos, ZeroMQ, RabbitMQ
Recently, there has been a lot of interest, and perhaps hyperbole, around microservices. Together with 8 of our consultants, we dug deeper into the main Open Source platforms and toolkits behind the hype.

Martin Fowler author and chief scientist for the software delivery company ThoughtWorks, defines microservices as small, decoupled services that serve one purpose. His other more colloquial definition is of “smart endpoints separated by dumb pipes”. Many see microservices as the antidote to the organic growth of server applications into the large, monolithic ‘big balls of mud’ (software systems that lack a perceivable architecture) which occur after lengthy periods of time, especially within FS. These networked components follow the UNIX principle to foster simplicity: do one thing, and do it well.

For anyone who has spent significant time within the industry, this paradigm is nothing new. It is effectively the basic design principle of componentisation applied to architecture. However, the principles of decoupled components offer many advantages, namely:

- Ease of unit and integration testing
- Re-use
- Ease of deployment
- Faster delivery of business value

Our panel investigated the main Open Source options available on the JVM for this approach.

POPPULAR PLATFORMS

Spring Boot - Spring is widely used in the industry, but over the years both Spring itself and its deployments have grown unwieldy. Spring Boot favours code layout convention over configuration. This means simple classes can be written quickly without the XML files normally required in building Spring applications. For example, a simple hello world app can fit in a tweet.

Since it sits on top of Spring, many pre-existing and useful Maven dependencies can be dropped into an application. For example, the addition of monitoring and health checks was achieved by adding in a single dependency which created a set of rest endpoints for monitoring the status of the application.

For Spring, this is a step in the right direction in terms of helping developers produce services quickly and efficiently. But its impact on the wider area of microservices remains to be seen.

Dropwizard - Dropwizard allows developers to quickly build and deploy RESTful applications. As Dropwizard ships with an embedded Jetty server, little configuration is required when setting up an application. All dependencies are bundled into one Jar with a single main class for deployment.

Overall, Dropwizard removes the need to spend time configuring and setting up applications, allowing the developer to focus on business logic and completing a RESTful application production as quickly as possible.

Akka - Akka is an actor model system written in Scala, which steers development towards a more functional approach. It can be thought of as an in-memory messaging system where messages are sent across modules in a highly concurrent fashion. Although this separation of concerns fits well in a micro-service approach, it does not provide the uplift of totally independent and deployable components in the same way as other APIs on offer. Nevertheless, Akka has many benefits: it is a powerful framework for highly concurrent systems, and its integration with Reactive Streams has much to offer, so Excelian will re-visit Akka in a future Tech Spark.

Ratpack - Ratpack is a Java port of Ruby's Sinatra framework. It provides a quick way to create an HTTP endpoint which can respond to HTTP calls. Ratpack's API is somewhat immature and there remains a large amount of boilerplate code required to create a new endpoint.
It has also been slow to accommodate Java 8 features such as Lambdas (which would be a natural port from Ruby). Ratpack has some integration with reactive streams, which may help certain use cases, but due to its current immaturity, developers would be better off looking elsewhere for a reactive streams implementation.

**Vert.x** - Vert.x is a polyglot, non-blocking, event-driven application platform that runs on the JVM. It has been strongly favoured at hackathons and is very good for agile coding as well as mashing up microservices in Java, JavaScript, Ruby and other languages.

**MICROSERVICES: NOT JUST FOR DEVELOPMENT TEAMS**

Microservices are the first architecture to sympathetically support DevOps and Continuous Delivery. While the above frameworks are great starting points for adoption, key success factors are:

- **DevOps**: The culture of both development and support teams, ideally amalgamating the functions.
- **Continuous Delivery**: It is imperative that services are constantly integrated and undergo heavy automated testing to ensure quality releases.
- **Testing suite/plan: Design by Contract**: By mocking the endpoints, different scenarios can be tested within code. If this new architecture is to bring all its perceived benefits, all teams must have faith in the level of automated testing.
- **Rapid deployment**: The ability to automate deploy and rollback functions is of great importance to operations, since the complexity of deployment increases with the number of new services and applications.
- **Extensive telemetry and monitoring tools**: Sufficient monitoring of an application or ecosystem of mini-applications is an important factor for an operations team, helping to manage failures and recovery.

**MICROSERVICES WITHIN FINANCIAL SERVICES**

This form of architecture has been slow to find a foothold within FS, and Excelian has not found too many clients employing them just yet. Within several projects we have either delivered, recommended or are in the planning stages of building out microservices as a strategic architectural solution.

Outside of the industry, microservices are certainly gaining momentum within a number of established enterprises such as Netflix, Gilt and Soundcloud. Excelian recently attended QCon where we heard from key technologists at these companies, and were impressed by their progress.

Netflix certainly helped to blaze a trail, producing a large number of extremely successful Open Source libraries which power microservices. Gilt and Soundcloud successfully broke up their monolithic Ruby on Rails architecture into hundreds of services, and both continue to grow, with Gilt professing to be running over 450 services!

**WHAT’S NEXT?**

At Excelian, we are optimistic about the microservices ethos. However, as there is no clear leader in terms of platform choice, we suggest a pragmatic approach, giving developers the scope to choose from the different APIs, toolkits, or languages on offer. This allows them to select the best tool for the service, avoiding wholesale adoption of one API or vendor (including Open Source vendors). Tying implementation to a specific API creates lock-in, and, if it covers 80% of use cases, the remaining 20% will often be difficult to implement. This will almost certainly lead to technical debt within the codebase, which is clearly at odds with one of the central tenets of this approach: agility.

Ultimately, teams will face failure with microservices unless they institute a DevOps culture and apply Continuous Delivery practices. Therefore, we see this architecture as a catalyst for DevOps and Continuous Delivery adoption at financial institutions.
There is much hype surrounding microservices, which have been successfully used by enterprises such as Netflix, Gilt and Soundcloud.

With no clear leading platform, it is best to adopt a pragmatic approach, and avoid tying implementation to a specific API.

Microservices are a great introduction into fostering a successful DevOps culture and Continuous Delivery practices.
DataStax recently announced the acquisition of Aurelius, the company which created Titan. There have been a few implementations of Graph, the new kid on the block, in the Open Source space, such as Titan, Apache, Spark and GraphX. However, Neo4J has clearly been the most successful so far.

Graph analysis is perhaps the single most effective competitive differentiator for organisations pursuing data-driven operations. It enables users to describe dynamic and complex domains more naturally and efficiently than with a relational database management system.

The community may question DataStax’s motivation in becoming a multi-model platform. Announcing Native JSON/Document schema-oriented support with Graph provides the broadest offering of any vendor.
THE DATASTAX BLOG ANNOUNCEMENT ON MULTI-MODEL:

Why Multi-Model?

It’s not uncommon to see NoSQL databases characterized by their underlying data model (e.g. wide-row/column store, document, graph, etc.). However, the reality is that since NoSQL has gone mainstream, our customers are building modern systems where the underlying applications require more than one NoSQL data model format.

Because of this need, architects oftentimes have to shard an application and use different NoSQL providers to meet the multi-model requirements of the underlying system. This increases complexity, cost, the IT staff’s learning curve, and slows the application’s time to market. That’s something we aim to remedy.

In the same way we’ve solved the mixed workload problem with DataStax Enterprise that allows you to run transactional, analytic, and enterprise search workloads in one database (thus eliminating the need to shard an application along those traditional lines), we now want to provide the ability for you to have support for multiple data models in the same database platform. Doing so removes the need for multiple NoSQL databases, and supplies you with simplicity, reduced costs and one software vendor with which to work.

Why Graph?

One of the most enjoyable parts of my job is to talk to our customers about their inventive new applications and hear about their current and future needs. About 2.5 years ago, no one talked to me about graph database support.

Fast forward to today and, my, how things have changed!

Since that time, the request for graph database support in DSE has been the single biggest customer ask, outpacing every other product/feature request. The customer demand for graph in DSE perfectly mirrors the dramatic increase in popularity of graph databases seen on db-engines.com (notice that #2 in popularity is the wide column store model, which is Cassandra’s category).

Why the huge spike in graph interest?

It’s really not hard to figure out. A very large part of today’s Web and mobile world is comprised of systems of engagement and systems of inquiry that deal with highly connected data.

Take your pick – fraud detection, social communication, contacts management, recommendation and personalization, financial analysis, buyer behaviour analysis – all these and more must manage a seemingly infinite series of connections between data.

Just as modern businesses have found the 40-year old relational database model inadequate to handle the data distribution and performance needs of today’s Web and mobile applications, they’ve watched the RDBMS fail to deliver the speed, uptime and agility needed to service these highly connected data applications.

Enter the graph database: an engine that can model these types of engagement and inquiry systems in a way where connecting data is easy and where performance doesn’t suffer from the antiquated join methodology that slows down an RDBMS.

Now, add to that engine the power and benefits businesses get from Cassandra and DSE. You have continuous uptime, horizontal scalability and linear performance capabilities that handle any sized workload. In addition, you’re supplied with easy data distribution across multiple data centres and cloud availability zones, seamless integration with powerful analytics and search functionality, and operational simplicity coupled with a great TCO.

That, in short describes our vision for DataStax Enterprise Graph.
While the blog mentions why graph is a huge asset to DSE, it is also worth noting the level of investment. DataStax are writing a new graph engine with the Titan team’s expertise. The Titan team will continue to contribute to the Apache TinkerPop 3 project.

USE CASES OF GRAPH RELATED TO FINANCIAL SERVICES

Graph can provide substantial value in a space like FS which revolves around data. It can model and exploit the relationships between trades, customers, transactions, assets, staff, phone calls, security cards, network traffic, and user log-on/off activity to an almost endless set of analysis, known as “network science”.

The following two use-cases are completely diverse, however both apply to FS and both can be modelled in graph.

Fraud: First-party fraud involves fraudsters who apply for credit cards, loans, overdrafts and unsecured banking credit lines, with no intention of paying them back. It is a serious problem for banking institutions, and Fraud Detection is one of the more commonly identified use cases for graph. Conventional methods cannot handle the high degree of indirect relations when modelling the data. With Fraud, the challenge is the exponential nature of the relationship between the number of participants in the fraud ring and the overall dollar value controlled by the operation. This connected explosion is a feature often exploited by organised crime.

Asset Management: Financial Instruments are formed of multi-relational data. For example a Trade is a type of Instrument which has legs, counterparties and assets. From this a model can be built linking together bank wide relationships to which analytics are applied.

“Network Science” is an emerging space where we are beginning to see a pioneering opportunity of graph as applied to FS. Below is an example of a “Network Science Dashboard,” which provides “Financial Network Analytics”:
In this example, the primary visualisation represents correlations among assets across different asset classes (the accompanying charts are used to provide detailed information for individual nodes). As graph adoption grows, an explosion of use-cases is widely expected, providing an opportunity for pioneering analysis techniques to emerge within FS.

WHAT’S NEXT?

The uptake of Graph is likely to be a big driver for DataStax Enterprise (DSE), and it will be interesting to see how the wider community responds to the DSE roadmap. The overlap in capability with Apache Spark and GraphX may confuse adoption, unless clear differentiators such as performance, capability or perhaps a VC-backed entity are drawn into the equation.

KEY POINTS:

- Graph Analysis enables users to describe dynamic and complex domains more naturally and efficiently than with a relational database management system, making it perfect for dealing with highly connected data.
- Graph can provide substantial value in a space like FS which revolves around data.
- The growth of network science provides an opportunity for pioneering analysis techniques to emerge within FS.

Neil Avery – CTO, Technical Consulting

References:

10 years ago, Oracle Coherence G, GigaSpaces XAP and Gemstone Gemfire began to set the standard for In-Memory Data-Grid solutions (IMDG). IMDG solutions have since become commoditised, and are now at the heart of many financial applications. Their generic appeal gave way to a wide and varied set of use cases, from simple caching to driving large, complex data driven risk-analytic workflows across large scale compute grids.

More recently, NoSQL solutions have also evolved from simplistic ‘sharded’ stores with restrictive functionality. They are now part of a rich, global datacentre analytic empowered engine, incorporating real time analytics, graph storage, machine learning and *QL at scale.

The difference between commercial, ‘off the shelf products’ versus Open Source offerings is not only that Open Source is ‘free,’ but that it is provided by the likes of Facebook, Twitter etc., which brings credibility in a ‘hot technology area.’

Corporations chasing a competitive edge are now embracing Big Data and NoSQL as a means of securing their future. Open Source technology is the staple of most FS technology stacks.

Does this mean that the time is ‘nigh’ for traditional IMDG? Can it be replaced with a large scale Open-Source NoSQL solution? With higher licensing costs than an OSS alternative, even factoring in vendor support, many financial institutions are looking to cut costs and consolidate.

**IMDG USE-CASES WITHIN FINANCIAL SERVICES**

We tend to see IMDGs in a wide variety of scenarios within FS. Most commonly, they are used as a caching scale-out layer for analytic compute grids of between 20,000 to 100,000 cores. Otherwise, they are generally used within trade repositories, market data services, or even data warehousing solutions.

**NoSQL Candidates:**
- Cassandra / DataStax
- MongoDB
- Couchbase

**GAP ANALYSIS & RISK**

Data Grids are extremely flexible. Eventing, data triggering, reactive eventing style scenarios, distributed workflows, complex event processing (CEP), aggregations, complex object oriented data querying and filtering are all possible. However, most implementations still use data grids in the simplest manner, as a cache.

**COUPLING: THE DATA GRID MODEL**

- Depending on the degree of coupling, it is possible to adopt NoSQL, provided the key functional requirements are met.
- Near-Caching: Scaling out the data layer to support ‘near caching’. This is mostly used on compute grids where data is shipped and cached locally. Local caches are driven by data access patterns and negate shipping.
- Querying capability: Most systems accessing data in an OO style will use standard primary key patterns (trade id etc.).
- Secondary Indexes: Querying data by non-primary attributes (i.e. trades by book).

- Cross data centre resilience: Coherence introduced the Push-Replication pattern on their Incubator, and it’s now being baked into the product.

- Eventing: Many IMDG systems rely on events to drive workflow using data-driven events (i.e. Risk Analytics for CDS’s are now complete > Start generating downstream reports).

- In-place-processing: A common function to aggregate or process data with IMDG using map-reduce style processing. The benefit is that data is processed in-place, rather than lifted. Similar functionality can be achieved utilising Apache Spark, where collocated spark nodes access local data nodes.

THE BIG DATA CRUX?

The problem with IMDG is that it doesn’t align itself with a Big Data Strategy. It may be considered a bolt on, however it doesn’t form part of a company’s data future. Data retention, data lakes, collection, audit, analytics, BI Integration, compliance etc. are now being driven as essential business deliverables, and Big Data solutions facilitate key elements of these.

OSS COHERENCE ALTERNATIVE?

Hazelcast now has a degree of maturity and acceptance: it delivers on distributed/partitioned caching, and provides many of the same benefits as NoSQL platforms, including Spark and other processing capabilities. It provides ‘like for like’ functionality with mature Data Grid implementations (Oracle Coherence, backing stores, aggregations etc.). Its selling point is that it provides a drop-in replacement for Coherence, great for ‘point solutions’.

It is not a NoSQL solution like Cassandra or MongoDB, nor a ‘system of record,’ nor a platform for a future proof Big Data Strategy. Industrial NoSQL solutions provide benefits well beyond the capabilities of Data Grids. Running across 20 000 servers spread globally will never happen in an IMDG cluster (without replication). The scale is different and so is the business opportunity.

A QUICK HISTORY

**2004**  
DB’s were not scalable, IMDG Caching introduced and is perceived as a specialist technology. Skills are in short supply, and Commercial vendors jostle for positioning

**2006**  
Analytics Compute Grids start exploding in size, hence driving demand for IMDG

**2007**  
Global Financial Crisis brings an upheaval in regulator reporting. Compute Grids continue to explode in size

**2008**  
Cost becomes a driver for everything

**2011**  
NoSQL gives birth to a data explosion and “Big Data Analytics” become the Next-Big-Thing

**2012**  
IMDG systems are now commoditised  
Cloud starts seeing traction

**2013**  
Hadoop shows its maturity as version 2.0 is released.  
NoSQL is no longer a bleeding edge risk, but provides a more fathomable, less specialised storage layer

**2015**  
NoSQL at scale is being touted, deployments of over 100,000 nodes are published  
Businesses start to strategise around Big Data and cloud. The barriers to entry are eroding
OUR STORY

By providing a functional interface over a NoSQL implementation, Excelian believes it is possible to migrate away from data grid platforms. Of course, this statement should be taken with a pinch of salt. A key benefit of migrating to a NoSQL solution is that the platform becomes accessible for formulating business strategy. Data analytics like Spark become a reality, and large scale persistence on several thousands of machines becomes reliable, as does global deployment models. This also allows user platforms to adopt the ‘industry standard’ in the technology of choice.

Throughout 2015, we shall work closely with several tier 1 banks to help them decouple from IMDGs, and migrate to industry standard NoSQL platforms. Such technologies are already in use in these institutions, so the barrier to entry is low or virtually non-existent.

We are also working with vendors, through workshops and strategic architectural design, to evolve the concept of modern ‘Enterprise Data Fabric’ Principles. The vendor list includes DataStax, Couchbase and MongoDB.

KEY POINTS:

• In-Memory Data Grid solutions (IMDGs) are at the heart of many financial applications. They are mainly used for caching, but are flexible and have a number of uses.

• IMDGs are a point solution and do not address a full Big Data strategy.

• As long as the key functional requirements are met, it is possible to decouple from IMDGs, and migrate to NoSQL platforms.

Neil Avery - CTO, Technical Consulting

References:

https://github.com/Excelian/Enterprise-Data-Fabric
The second generation of Big Data platforms have begun to emerge throughout 2015. MongoDB, Cassandra and Couchbase, all on version 3 or 4, hook in with the new industry standard for analytics and Apache Spark. They tackle JSON head-on and deliver *QL at scale. The internet is littered with stories about which platform suits which particular use case. Elasticsearch is jostling for attention, but fits more of an edge case around indexing than that of a Data Storage Layer. Furthermore, a lot of historical coverage of ‘Data Lakes’ is being built around Hadoop, however the next wave of NoSQL, as well as Pivotal’s ‘Open Data Platform’, are usurping Hadoop and Data Lakes.

CHOOSING THE NoSQL PLATFORM

Our customers are either MongoDB, Cassandra or Couchbase shops, and there exists a clear polarisation of views concerning platform choice. So it’s common to find all three considered early in the decision making process, regardless of origin. In many cases, the final decision is non-technical, for example: due to personal experience with customer support, bad internet-press or simply because friends are using it. Once this decision is made, there can only be one platform. The longer term strategic significance of this is extreme, and paves the way for future products. As such, the enterprise becomes coupled by data-gravity, and migration may be a long and costly exercise. It’s no surprise vendors want your business.

WHAT IS AN ENTERPRISE AS A SERVICE?

Many Investment banks design systems to be ‘strategic’. For years, an IB technology stack consisted of multiple siloed systems, with duplicate functionality and a high degree of complexity. To drive down costs, these are now systematically consolidated into strategic offerings which service bank-wide infrastructure. So, when considering a storage layer like Cassandra for the ‘UBS Neo Platform,’ this is done at scale and used in anger. The primary consideration is how to offer Cassandra using “Data-As-A-Service” to the bank.

Serving data in the enterprise has its own set of challenges, and doing so at scale to some of the largest grid systems in the world pushes the envelope even further. How does one facilitate the network bandwidth required to concurrently transfer gigabytes of data to several hundred thousand compute engines? This is why Oracle Coherence is the staple system for many such use-cases.

We have compiled a set of generic storage requirements which represents the core of an Enterprise Data Fabric.

Core Non-Functional Requirements:

- Globally consistent view & access
- Real-time access
- Data lineage (data lifecycle, audit)
- Geo-locale data gravity/locality: Making data sticky (legal reasons, access patterns)
- Security: Use/role based access
- Multi-tenancy: support multiple schema sets
- Workload isolation: separate different types of processing, querying
- Multi-site/datacentre data awareness: making data flow to regions based on attributes
• Extensive monitoring

• Analytics: native Spark integration (collocated workers) in place of data processing via collocated spark workers and data nodes.

• Agility: adding/removing nodes, altering schemas

• Backup/Restore

• Apache Spark analytics (collocated with stored data for in-place processing)

• *Caching: a highly scalable client-side data caching layer - compute engines cannot afford the chatter or repeated requests for the data required to build pricing environments.

• *Eventing: data eventing supports the notion that consumers receive notifications when data changes, or even the data itself.

• *APIs: (Java/.NET/Restful APIs) requesting the data needed to facilitate standardised technology stacks, including language or API convention.

Non-Core Functional Requirements:

• *Cloud burst-out – moving data between sites / Hybrid Cloud

• Data-gravity – large scale deployments

• Multi-site awareness – conflict resolution

* These are the requirements that Excelian’s Enterprise Data Fabric is aiming to fulfil. The GitHub ‘Enterprise Data Fabric’ is our public landing page for the project.

MAPPING TECHNOLOGY ONTO THE ENTERPRISE

Excelian is working on building out this story with DataStax (Cassandra) and Couchbase. Both vendors support the above requirements, yet mapping a technology onto the Enterprise is a challenging exercise. This exciting story will unfold over the next few months as the blueprint is developed, mapping real-world FS scenarios of emerging technologies such as Hybrid-Cloud, Apache Spark analytics, Grid-Bursting, and cloud elasticity.

The client-side caching layer provides a uniform, scalable solution through standard API’s, while also allowing developers to leverage native connectivity. The goal is to provide enterprise grade, low-latency, scalable access to data which doesn’t crush the storage layer.

KEY POINTS:

• The second generation of Big Data platforms are emerging throughout 2015.

• Systems are now consolidated into strategic offerings which service bank-wide infrastructure, meaning an Enterprise Data Fabric has demanding storage requirements.

• Excelian is working on mapping a technology onto the enterprise, providing enterprise grade, low-latency, scalable access to data which doesn’t crush the storage layer.

Neil Avery – CTO, Technology Consulting

References

http://excelian.github.com/Enterprise-Data-Fabric
Docker burst onto the scene a few years ago, originating from DotCloud, a little known PaaS cloud provider. Solomon Hykes, relatively unknown at the time, created a container format which made it easier for developers to run applications in an isolated virtual sandbox known as a container.

Solomon’s new container technology provided developers with an easy to read format, resource isolation and a light weight alternative to full hardware virtualisation. Docker added a developer-friendly interface to a 10 year-old technology by providing familiar push, pull and commit semantics to containers. This new-found developer agility spawned an ecosystem of tools, best practices and methodologies to develop resilient, highly available applications without the overhead of running a VMWare estate. Community efforts were tasked with answering the storage and network provisioning problems. The excitement at conferences was palpable as the community rose up and produced tools to enable Docker to be more than just a container format.

**ALL EYES ON DOCKER**

This revolution in the DevOps and open-source community did not go unnoticed. Docker found itself at the front of a revolutionary change in how applications were deployed, configured and executed. Its capabilities and potential put it in a position to disrupt the cloud and virtualisation industry. Even VMWare, which has the largest presence in the virtualisation market, provided tools to integrate with Docker. Docker’s unique position attracted the attention of investors and industry giants like Red Hat, Google and a surprising corporate partner, Microsoft.

![Figure 3 Timeline of Docker Evolution](image-url)
After its first round of investment, Docker acquired high profile hires from VMWare and Puppet, the software configuration management company, and has raised more than $150M in funding to-date. Set for rapid growth in 2015, Docker now faces challenges which could derail its meteoric rise.

CONTAINER TECHNOLOGY

Container technology has been around since 2006 when two Google engineers Paul Menage and Rohit Seth added namespaces and control groups to the Linux Kernel. At the time, Google was developing its orchestration platform for improving server utilisation in its cluster as was spinning up over 2 billion containers a week. Container scheduling and orchestration technologies are central to Google’s competitive advantage, and the techniques used to manage containers across tens of thousands of servers per data centre are a closely guarded trade secret.

Containers became the enabling technology for Platform-As-A-Service (PaaS) cloud providers, which allowed developers to build their applications by describing how they connected to each other. The installation and configuration details were completely hidden, improving developer productivity. Despite containers being used extensively by cloud PAAS providers to provide database, messaging and web server components to developers, there was no standard format to describe, build and deploy them. Docker filled this gap nicely, becoming the universal container format that enabled developers to have similar capabilities on privately-owned hardware.

At that stage Docker, remained solely a container format. CoreOS then launched a separate effort to develop a container operating system which would orchestrate, schedule and scale containers. There was a symbiotic relationship between the two technologies and the rest of the Open Source community.

DOCKER VS. THE COMMUNITY

Docker owes its success to the open-source and developer community. The open-source community is driven, vocal and agile, and responds quickly to technological changes. While Docker was in its infancy, the community started to solve problems it wasn’t ready to tackle due to the size of its engineering workforce. These problems were typically concerned with storage and networking provisioning, and Docker was happy to accept contributions in its pre-investment era. However, with corporate backing, its roadmap is in direct conflict with the ecosystem that helped build it.

The community has been trying to extend Docker’s capability to provide storage and network management, but interoperability with Docker itself has caused problems. To date, Docker has not released an API or plugin architecture to enable enhancements by third-party vendors. This forces other tools or vendors to wrap around the Docker service, inadvertently preventing cooperation between the tools themselves. For example, the storage technology Flocker cannot be run alongside Weave, a networking component, without Docker releasing an interoperability layer.

Docker is fully aware interoperability is an integral part of their on-going success, and made an early promise to the community that extensibility was a priority. Yet Docker cannot monetise its platform without cannibalising the ecosystem that contributed to its success. It’s reasonable to expect Docker to release its own tooling around orchestration, monitoring, storage and networking before releasing an extension API to the community. This approach would ensure a reference implementation exists which vendors can use to guide their own implementations.

As a container format, Docker cannot stall indefinitely, and has boosted its engineering workforce with an acquire-and-hire strategy. It acquired Fig, a London-based application composition tool, and recently SocketPlane, a software defined networking layer. Despite this, the community remains larger than Docker’s engineering workforce, and is able to bring answers to problems to market faster.

THE MONSTER LURKING IN THE SHADOWS

Docker’s march to an initial public offering (IPO) will not be easy. There are clear challenges to becoming enterprise-ready, however the recent investment and good-will it is currently enjoying will make this easier. Yet Docker’s greatest problem is the company glaringly omitted from their press releases – Google, which has been fighting it by stealth. Instead of tackling Docker head-on, Google recently funded CoreOS through its investment arm Google Ventures, and appears to be using an embrace and extend strategy, as popularised by MS. Google released Kubernetes in July 2014 as a next-generation container management and orchestration platform built on its decade-long experience with running containers at scale.

The capabilities of the project are extensive, and the community around it is extremely active. Kubernetes aims to be the standard API for container orchestration across all cloud environments, not just Google Compute Engine. It was initially adopted under the assumption Docker would
be the container technology underpinning it, but recently Google invested in Tectonic, a CoreOS offshoot. Tectonic is a commercial enterprise edition of Kubernetes which supports Rocket, a different container technology. Rocket is a community driven alternative intended to replace the Docker container format.

Using this vehicle, Google can now drive the development of enterprise features without becoming involved in Docker’s community politics. As with Kubernetes, the Rocket container format is being developed in the open to address some of Docker’s perceived architectural and security flaws. Ultimately, the Docker container format is replaceable since the technology has existed in different forms for almost a decade.

CONCLUSION

The tidal wave of hype around Docker is the same as that which followed cloud and Big Data a few years ago. All major industry players such as Google, IBM and Microsoft want a piece of it, and are clambering over each other in what can be described as a digital gold-rush. Whoever owns the mind-share of developers is uniquely positioned to influence and direct corporate cloud strategy. Google and Microsoft have a unique opportunity to revolutionise the next generation of cloud infrastructure while positioning themselves at the front of it. The container revolution is ushering in a new era in computing, where data-centres are no longer seen as remote servers containing isolated units, but as a warehouse of resources working in cohort as a single machine. On the Excelian blog, we have looked at Hybrid cloud strategies in a container world.

KEY POINTS:

- Docker developed a container format which made it easier for developers to run applications.
- With the help of the Open Source community, Docker’s technology revolutionised how applications were deployed, configured and executed. However, interoperability between Docker and the community has led to problems.
- By investing in Rocket, a community driven alternative to Docker which is being developed in the open, Google hopes to drive the development of enterprise features in container technology.

Zvenyika Gomo- Consultant
Server virtualisation provides many IT infrastructure management benefits, since it can run and manage multiple applications and operating systems on the same physical server. In theory, this allows increased utilisation and improved business continuity by moving applications across physical servers easily and providing a faster server provisioning process via ‘images.’

However, additional performance overheads mean High Performance Computing (HPC) and virtualisation do not go hand-in-hand, restricting access to the benefits of virtualisation. Recent advancements have narrowed the gap in terms of CPU performance when compared to bare-metal, but large scale grid workloads which require high I/O and inter-node communication must still contend with performance degradation when running virtualised.

The recent surge in interest in Linux containers (LXCs), especially Docker, have brought about a renewed interest into ‘virtualising’ HPC applications. A Docker-based HPC platform allows near bare-metal performance while reaping the benefits of a virtualised application stack, including better application isolation and fine tuning of resource allocation within a host.

In this article, we investigate the performance overheads of Docker and VirtualBox, which we use to run compute nodes in the Excelian HPC Lab cluster, and explore the potential benefits of a ‘Dockerised’ HPC compute environment.

**EXCELIAN HPC COMPUTE CLUSTER**

The Excelian HPC compute cluster consists of 5 servers with a total of 56 cores, and its main usage is grid software development and testing. We use VirtualBox to create virtual servers for a number of use cases:

1. Simulating multiple compute hosts connecting to a Grid.
2. Creating and automating HPC build environments for our clients.
3. Simulating hosts with different configuration and operating systems.
4. Application isolation – preventing a running grid process from interfering with another process on the same host.

In our experience, running virtualised clusters meets our development and testing requirements. Yet there are a few drawbacks:

1. Slow provisioning of virtual hosts - especially if provisioning a few instances simultaneously on the same host.
2. Overhead can be high in terms of memory usage and CPU usage on certain HPC workloads.
3. Need to maintain a set of post-provisioning scripts (we use Ansible) for certain Grid middleware which does not work well with ‘imaging’.

Most recently, we experimented with Docker to evaluate its performance and its suitability as a possible replacement for VirtualBox.
BENCHMARKS

We used the Java Grande Benchmark Suite as a basis for our benchmarks. The 3 main benchmarks are:

- **Euler**: Benchmarks the timesteps per second while solving Euler equations for flow in a channel.

- **Monte Carlo**: Uses the Monte Carlo technique to price products derived from the price of an underlying asset.

- **Ray Tracer**: Benchmarks the number of pixels per second rendered on scene using a 3D raytracer.

Figure 4 and Table 1 show Docker achieving Bare-Metal performance in most of the benchmarks while Virtualbox lags behind, achieving only 25% of bare-metal performance in some cases.

Also worth noting is a small performance increase observed on Docker when compared to Bare-Metal. One possible explanation could be the effects of some small loads (e.g. slocate, metric collection, etc.) running on the Bare-Metal host during the benchmarks.

![Figure 4 Performance comparison to Bare-Metal](image)

**Table 1 Java Grande Benchmarks (higher is better)**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Bare-metal</th>
<th>Docker</th>
<th>Virtualbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler:Init:SizeB (Gridpoints/s)</td>
<td>244084.97</td>
<td>198643.61</td>
<td>90423.73</td>
</tr>
<tr>
<td>Euler:Run:SizeB (Timesteps/s)</td>
<td>33.29</td>
<td>33.58</td>
<td>20.16</td>
</tr>
<tr>
<td>Euler:Total:SizeB (Solutions/s)</td>
<td>0.31</td>
<td>0.31</td>
<td>0.19</td>
</tr>
<tr>
<td>MonteCarlo:Run:SizeB (Samples/s)</td>
<td>4939.90</td>
<td>5101.61</td>
<td>5010.02</td>
</tr>
<tr>
<td>MonteCarlo:Total:SizeB (Solutions/s)</td>
<td>0.08</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>RayTracer:Init:SizeB (Objects/s)</td>
<td>64000.00</td>
<td>32000.00</td>
<td>32000.00</td>
</tr>
<tr>
<td>RayTracer:Run:SizeB (Pixels/s)</td>
<td>29429.08</td>
<td>30484.09</td>
<td>6979.54</td>
</tr>
<tr>
<td>RayTracer:Total:SizeB (Solutions/s)</td>
<td>0.11</td>
<td>0.12</td>
<td>0.03</td>
</tr>
</tbody>
</table>
The system configuration was set up as in table 1.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Lenovo ThinkStation P700</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Dual Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB RAM</td>
</tr>
<tr>
<td>OS</td>
<td>CentOS Linux release 7.1.1503</td>
</tr>
<tr>
<td>Kernel</td>
<td>3.10.0-123.20.1.el7.x86_64</td>
</tr>
<tr>
<td>VirtualBox Version</td>
<td>4.3.4</td>
</tr>
<tr>
<td>Docker Version</td>
<td>1.5.0</td>
</tr>
</tbody>
</table>

**EVALUATION**

Setting up a Docker test environment for our cluster was an easy process with documentation and resources available online. During the benchmarks, we also found containers were created almost instantly, while virtual instances took longer (several minutes) to instantiate. We were also impressed with Docker’s ability to enable fine-grained guarantees of resources and performance isolation for each of our compute containers.

With regards to HPC Grid computing, we foresee the following benefits of Docker:

1. Provision of application insulation on the same host i.e. Docker enforces limited access to resources (CPU, Memory, etc.) for each running application. In traditional Grid/HPC, a running task may potentially take up all RAM/CPU on a single host.

2. Ability to share hosts between Grid and non-grid environments e.g. during weekends, application servers may be used to host Grid containers, and perform long-running weekend/month-end batch jobs.

3. Allows applications to be easily packaged and deployed across different infrastructures (e.g. external or internal cloud platforms, developer desktop etc.)

4. Ability to simultaneously run varied workloads, both Grid and non-Grid, on the same server, thus increasing server utilisation while guaranteeing resource and performance to each application container.

**SUMMARY AND FURTHER WORK**

We shall continue with our efforts to further Docker for HPC, exploring how higher-level orchestration tools such as Kubernetes and Mesos may be used to enable HPC and non-HPC sharing of resources, increase server utilisation and simplify the management of containerised applications. We will also look at LXD, a Docker competitor, and how it stacks up in terms of features and performance.

Our benchmarks have shown Docker HPC compute containers to have negligible performance overheads. And we are now looking forward to testing vendor-based Grid middleware products, which we will do once they start following IBM’s lead and provide container support.

**KEY POINTS:**

- Although server virtualisation provides many IT infrastructure management benefits, additional performance overheads have limited its success with high performance computing (HPC).

- Excelian tests have shown Docker HPC compute containers to have negligible performance overheads, offering many benefits to HPC Grid computing.

- Excelian now intends to evaluate the impact of higher-level orchestration tools, such as Kubernetes and Mesos, and test LXD, one of Docker’s competitors.

*Jay Chin – Principal Consultant*
HTML5 brought promises of near native performance in the browser. It heralded the creation of a move away from native applications on mobile and desktop to the web. Unfortunately for applications with animations or heavy DOM manipulation, HTML5 has not delivered. This is particularly apparent in the data visualisation and animation space. The issue is inherent in the framework, as it was principally designed as an engine to render text.

**Famous** represents an effort to fill this gap. It’s the first JavaScript 3D layout engine with a custom 3D physics engine, and can render to DOM, Canvas or WebGL. It achieves a blistering and consistent rendering speed of 60 frames per second in the browser, providing seamless natural movement. To top it all off, it is Open Source on GitHub with over 7,000 stars, indicating real community interest.

**Famous** applications define a scene and component trees which are added into the 3D space. It then allows you to define transitions and apply them to the components and sub nodes. Whereas these declarations previously occurred in CSS and HTML, or by manipulating the DOM in JavaScript, they are now performed declaratively in JavaScript.

As with any project to change the browser engine, the most pressing concern is compatibility. While Famous certainly works in Chrome (like most things), IE and Firefox experience downgrading and rendering challenges.

**Famous** has interesting demos demonstrating the power of its rendering engine, which is highlighted further in several introductory tutorials. It’s easy to envisage 3D models or complex animations as a result of interactions with an immersive UI.

**FAMOUS FRAMEWORK INTEGRATIONS**  D3

The most benefit is derived from **Famous** when integrated with other frameworks. D3 revolutionised the way data is used to drive browser rendering or data driven code flows, and is lauded for its simplicity. While generic, its power lies in its expressiveness and complete extensibility. It is mature, and, while the hype may have subsided, its value continues to grow. As a result of its processing logic, D3 is generally used to drive SVG rendering, html or other elements, including **Famous**. **Famous** and D3 seem a natural fit: D3 provides the flexible data driven interaction needed to drive such a rendering engine. In fact, the demo on GitHub by one of the Famous team shows D3 rendering bar charts. Although it is not the most practical of applications, it shows that such uses are possible.

**FAMOUS FRAMEWORK INTEGRATIONS**  ANGULARJS

The Famous team have worked hard to make their framework accessible in AngularJS applications, creating Famous-Angular. Integrating these two technologies provides something very special: an application which has a great MV* structure, and is testable and decoupled through dependency injection. It also boasts superb data binding combined with a fluid layout management system and cool animations/transitions within apps. To achieve this, they created a set of directives which allow for easy creation of surfaces and binding to content. Custom Famous directives may also be created by implementing a few handles, allowing them to transition. Effectively, this enables the creation of reusable components with cool interactions.
Recently, there has been a move away from native applications to the web. However, applications with 3D visualisations in finance are still largely native. Previously, this type of visualisation in a web page was almost impossible. However, Famous will soon release its mixed mode, allowing animation of DOM, SVG and WebGL components. This combination is highly performant as the modelling may be offloaded to the GPU, allowing the visualisation of complex financial models such as option volatility surfaces.

We believe Famous is ready for early adopters wishing to produce innovative applications for Evergreen browsers. With super-fast 3D transitions and the combination of DOM and WebGL components in Famous’ mixed mode, we can create a much richer experience for users. We also believe Famous provides an enablement platform: companies striving for a unique edge or USP can explore visualisation techniques never previously available in a browser. To date, FS have never really pushed this boundary. Investors believe in Famous with $40 million invested so far, on top of the $20 million received late last year. Interesting space to watch further.

In finance, the term “Big Data” is commonplace, but with large amounts of data comes the challenge of visualising and exploring trends. Cleaner animations help the process of gaining insight and discovering these trends. One such example is with power production across the US through time: although a complex data set, 3D visualisation trends can be easily picked out.
KEY POINTS:

- Since it was designed principally to render text, HTML5 has not delivered in terms of data visualisation and animation.

- Famous is a powerful JavaScript 3D layout engine which renders to DOM, Canvas and WebGL. Particularly when integrated with other frameworks, it can produce 3D models and complex animations at a blistering and consistent speed.

- The introduction of Famous’ mixed mode will allow the visualisation of complex financial models, enabling finance companies to analyse data further and discover new trends.

Tushara Fernando- Senior Consultant
Tim Hughes- Principal Consultant

References:

https://famous.org
Developing custom software to support business processes can be a costly affair. Businesses often use inefficient approaches to communicate requirements to stakeholders and developers.

Even for more agile methodologies, it is still possible to see the cycle where developers are handed ambiguous requirements, resulting in gaps being filled in by assumptions, leading to defects in code, which then need business input to resolve, who then provide more ambiguous requirements.

Behaviour Driven Development (BDD) is a collaborative approach to software development that bridges the communication gap between business and IT. BDD helps teams communicate requirements with more precision, discover defects early and produce software that remains maintainable over time, whilst still supporting agile methodologies such as Scrum or Kanban.

The two main practices in the BDD approach are discovery workshops (short meeting to produce concrete examples of business rules and acceptance criteria) and executable specifications (the examples expressed as executable software specifications in plain language that everyone in the team can understand).

- Cucumber is a free, open source product that can run the executable specifications against the software being built.

- Cucumber produces reports that indicate what parts of the specification are implemented correctly, and what parts are incomplete or defective.

Excelian recently invited Aslak Hellesøy to talk about BDD and how it can best be leveraged in software development projects. He also explained the Cucumber features that help developers and other project stakeholders to better utilise BDD.

The full presentation is available on Excelian’s YouTube channel.

These executable specifications also work as automated regression tests which reduce - and in some cases eliminate - costly and time-consuming manual testing.

The full write-up is available to read on the Excelian Blog.

Aslak Hellesøy: Creator of the Cucumber project
Financial institutions face growing pressure to compete with each other, while at the same time complying with regulators. In order to achieve these goals, businesses strive to cut the costs of systems and infrastructure. They service multiple asset classes while attempting to keep them flexible and adaptable to new and changing regulatory requirements. Excelian recently attended QCon London, where we heard about new and exciting practices in Open Source technologies which may help institutions succeed in these endeavours.

**QCon**

A common cost cutting practice at Financial Institutions is to employ reusable components and share them between asset class systems. Microservice architectures were a common theme at QCon, and we believe businesses should employ them to produce such components. Dr Rebecca Parsons, the CTO of ThoughtWorks delivered a defining talk on Microservices and the enabling of Evolutionary Architecture by Continuous Delivery.

**MORE ON MICROSERVICES**

Dr Parsons discussed how Service Oriented Architecture paved the way for Microservices. The problem was that that inherent way of producing SOA software was the monolithic-like Enterprise integration nature of development prevailing at the time. Now, with the enabling principles of agile development and continuous delivery, modern Microservices are built around business capabilities and are independently deployable, allowing businesses to rapidly deliver value.

- Using microservices has a range of implications, which businesses and developers must carefully solve and manage: The granularity of services and maintaining the balance between coupling and cohesion.
- Complex monitoring requirements and telemetry volumes.
- Handling and recovering from service failure.
- Data storage and transactional consistency paradigms.

Bringing products to market quickly is the key to employing microservices, as well as beating competition. At QCon, Excelian heard how DevOps and Continuous Delivery can help achieve this. Dave Farley presented on the rationale for Continuous Delivery and Pipelines, advising that inefficient processes can produce poor quality output. Dave suggested businesses can deliver high quality code by shortening feedback loops and delivering deployable builds frequently.

**DEVOPS**

Stephen Thair, co-founder of the DevOpsGuys outlined how DevOps can greatly enhance the speed of software delivery, and outlined the core steps to transition to a DevOps culture: implementing trust and collaboration between teams. Irrespective of well-established ITIL processes, it still remains a long road to the employment of DevOps within a Financial Enterprise. Nevertheless, we believe some of these practices could complement such processes, and Institutions should look at ways this could be possible.

A number of Open Source technologies were on show at QCon. Mechanical Sympathy stalwart Martin Thompson of LMAX Disruptor fame presented Aeron, his latest contribution to the industry - a ‘next generation’ high performance messaging library. It was built wait and lock free with consistent and predictable latency, and Excelian is currently evaluating it for client use. C24’s John Davies presented its Simple Data Object’s library, which can reduce the size of an enterprise’s data, thus saving on such infrastructure costs as bandwidth and storage, and perhaps reducing licencing costs of certain in memory grids.
Overall, QCon was a great experience which I strongly encourage others to attend. Videos of presentations from QCon will be released as per the schedule at http://qconlondon.com/video-schedule, with a wider round up to be distributed at http://infoq.com/minibooks/emag-qcon-london2015-report.

Tom Ellis- Principal Consultant
You may remember our last Tech Spark which focused on cloud computing and its consequential impact on FS. It was really well received and we enjoyed hearing from many of you directly with your thoughts.

As we anticipated, we saw continued acceleration in cloud adoption in 2015, particularly in FS where tier-1 organisations are currently strategising and planning their move to hybrid cloud.

**ADOPTING CLOUD IN FINANCIAL SERVICES**

In terms of adopting cloud, FS is distinct from many other industries. This is largely due to the size of existing datacentre investment, security and regulatory compliance. Furthermore, bank operations include hundreds of applications spanning thousands of servers. This document summarises the stages of cloud adoption in FS.

**STAGE 1 CLOUD PLATFORM ARCHITECTURE**

When choosing a cloud vendor, one should analyse pricing characteristics to control or understand cost. This stage involves building two costing models and establishing the cloud technology stack.

Instance Right-Sizing Modelling: A reproducible and baseline task is the prerequisite with which to start. A baseline task is a well-known task executed on standard hardware: it represents the base unit of cost as well as the majority of the compute workload for an existing compute grid. Exercising the ‘task’ on cloud instances allows an A|B cost comparison model to be built, and it also facilitates the building of a costing model for each vendor and instance type.

Scale-up/Scale-down Modelling: In order to effectively utilise cloud services, elasticity is key. Scaling instances up facilitates compute job SLA’s and allows for different execution profiles once existing datacentre constraints have been removed. To ascertain the benefits of different cloud vendor pricing models and whether the aggressive scales sought after are practical, compute elasticity must be cost-modelled.

Cloud Technology Stack, Cost Model and Roadmap: Building the full technology stack assumes the stage 2 has commenced. Inputs include application profiles, disaster recovery testing, security and compliance, virtualisation strategy, infrastructure governance, multi-tenancy of cloud services etc. The aim is to build out the full technology stack and develop a costing model and cloud-roadmap.

Multi-Vendor: With a multi-vendor roadmap, workload orchestration and brokerage systems form part of the solution. A multi-vendor solution may also affect the adoption of PAAS.

**STAGE 2 APPLICATION PORTFOLIO ANALYSIS**

This stage determines the application inventory, and drives a series of discovery workshops to ascertain important, yet common characteristics. Once the application profiles are understood, they are mapped on parts of the cloud-technology stack. Non-functional characteristics include complexity analysis, business drivers such as ‘invest’ status and refactoring elements to provide common-services. Each application profile is mapped onto a template, and a portfolio roadmap is built which includes operational elements for on-boarding (teams, security, operations and processes). An application profile includes maturity, business status, investment profile, platform readiness for cloud, security and operational and team attributes.

**STAGE 3 CLOUD ON-BOARDING**

Each application undergoes a different workflow based on its application profile. This process requires a close working relationship and alignment between the development and the cloud-infrastructure teams.
Thank you for reading the latest edition of Tech Spark, put together by industry partners and our technological consultants here at Excelian. I would like to thank all of our contributors for their interesting perspectives and insight into OSS in FS.

My key takeaway from this edition is that we are on the cusp of seeing a full technology stack revolution. We have seen the way in which Docker usurps virtualisation, the way in which NoSQL is becoming the ‘norm’, the general acceptance of Apache Spark for analytics and the wide spread acceptance of microservices. I believe the next several years will allow us to build unique systems in ways we never imagined. Machine-learning and predictive analytics will be less about pioneering and more about leveraging massive business opportunity within FS. It’s all very exciting as it seems that every day there is something new to learn.

We are hoping that this edition has been interesting and perhaps provoked some discussion amongst you. We would be happy to hear from you, techspark@excelian.com

NEXT EDITION

Work is already underway into our next Tech Spark which will focus on Big Data in FS. If you have any particular topics that you would like us to research, or any research of your own which you would like shared, please do let us know.

INTERESTED IN COMING AND JOINING US?

It is due to our forward thinking consultants that we are able to put publications like this together. If you would like to come and join Excelian, we are always on the lookout for Technical Consultants so why not take a look at our careers page or contact us to see the latest opportunities.

Neil Avery - CTO, Technical Consulting

neil.avery@excelian.com