BIG DATA
TECHNOLOGY SPECIAL

TECH SPARK, H2 2016

6
Big data in financial services: past, present and future

28
Enterprise Blockchain Accelerator: Join us!

36
Drive fast, flexible VaR aggregation with Spark

excelian
LUXOFT FINANCIAL SERVICES

CONTENTS

TECH SPARK, H2 2016

EDITORIAL

Neil Avery
CTO, Excelian
Luxoft Financial Services

For the last six years the financial services sector has struggled to keep pace with the overwhelming growth in big data, cloud, analytics and data-science technologies. The situation reminds me of the music industry. Once you had pop, rock, R&B, blues and a few other distinct genres and it was simple – you knew what you did and didn't like. Now, a new music genre emerges every three months, and such clear definitions are a thing of the past. Sound familiar?

Big data can't simply be categorised as a branch of enterprise architecture, databases or analytics. It is a completely new IT genre with blending, processing and mutating of data at scale to create the 4Vs – volume, velocity, variety and veracity – and more. Big data forms a baseline platform, which brings us to the realisation that we are building something completely new – and much bigger.

The impact of big data thinking is as profound as the emergence of a new programming language (and we are seeing a lot more of them too). It is giving rise to a new industry – one standing on the shoulders of giants. Within this new landscape we see sub-genres rapidly evolving around graph-databases, streaming, cloud and DevOps, all with at least 17 new ways of solving old problems. The curious thing about this 'new industry' is that it makes the old one obsolete, replacing it with something as disruptive as the IT revolution was in the past. I have no doubt that we are on the cusp of the 'next big thing' and that the future of financial services technology will be transformed by the power of data, cloud, streaming, machine-learning and internet of things. More importantly, it will be fundamentally different, with more giants and more shoulders.

With so much to cover, collating this issue of Tech Spark was both challenging and exciting. In any market, the emergence of new capabilities that enable dramatically different ways of doing things creates huge opportunities for disruption and financial services is no exception (the latest example is of course blockchain). Only a year ago this story would have been very different. Of course, big challenges remain, not least hiring people with the right skills to tap the rich potential of new, existing and unrealised use cases. But with everything to play for, I hope that this publication will give you more insight and inspiration as you venture forward on your big data journey.

EDITORIAL BOARD	Neil Avery | Andre Nedelcoux
EDITORIAL AND MARKETING TEAM	Martyna Drwal | Lucy Carson | Alison Keepe
CONTRIBUTORS	Deenar Toraskar | Mark Perkins | Conrad Mellin | Raphael McLarens | Thomas Ellis | Ivan Cikic | Vasiliy Suvorov | James Bowkett | Darren Voisey | Jamie Drummond | Theresa Prevost | Aleksandr Lukashev | Alexander Dovzhikov
DESIGNED BY	S4

Neil Avery

FINANCIAL SERVICES TECH RADAR

Inspired and encouraged by ThoughtWorks, we have created the first Excelian Tech Radar. It's a snapshot from the last 12 months that captures our key industry observations based on work in capital markets with most of the tier one and tier two banks in London, North America and Asia-Pacific.

As a practice, we're constantly looking to learn, lead and stay abreast of top technology trends – and separate the hype from the fact, while understanding how hype can fuel demand.

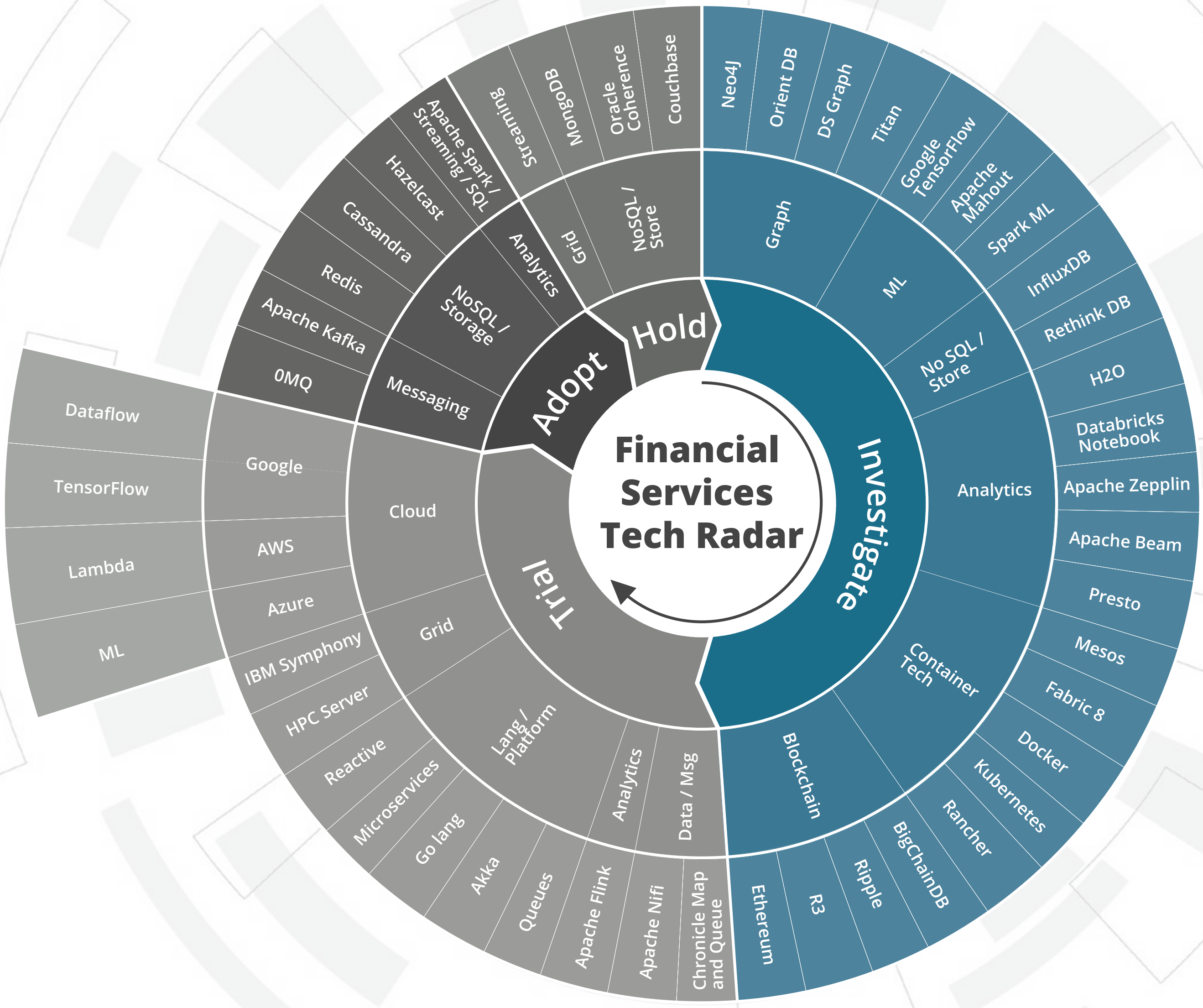
On the Tech Radar, the category we've flagged as 'hold' means that we generally see this space as having matured sufficiently, that it has slowed and that other more creative and unique approaches could be explored. But as expected, the largest single group of technologies falls within the 'to investigate' category: this reflects a growing appetite for R&D investment. Some trends stand out as particularly noteworthy.

THREE TECHS TO TRACK

The first wave of hype is around the universal appeal and uptake of **Spark**, which provides everything that was promised and beyond. What's more, as one of the key innovations in the big data arena – Kafka being the other – it has really helped to drive big data adoption and shape its maturity as part of a viable business strategy. We also see innovation with Kafka K-Streams and Apache Beam pushing the streaming paradigm further forwards.

The second wave of hype is around lightweight virtualisation tool **Docker**. With its shiny application containers, it's witnessed a two-year growth frenzy, mostly attracting hardcore tecchies with little more than three years in tech development. Who'd have thought infrastructure could have such appeal?

Finally, the biggest tsunami is **blockchain**. They say there's a blockchain conference in the US every day. It's therefore no surprise that early innovators are scrambling, Fintechs are all the rage and incumbents are joining the R3 consortium to embrace the threat rather than risk disruption from new players. It's a two-pronged hype cycle that we haven't seen in a very long time, being industry led just as much as it is technology led.



Neil Avery

BIG DATA IN FINANCIAL SERVICES: PAST, PRESENT AND FUTURE

When the genesis of Hadoop emerged from a Google file system paper published in 2003, it was the seed that launched the big data technology revolution. Since then, the financial services sector has weathered countless storms. But throughout the ups and downs, big data has continued to evolve into the all-pervasive force that it is today. So how did big data get so big and where's the smart money on where it will it go from here?

WHERE IT ALL **BEGAN**

Google had been using the precursor technology to Hadoop in production for almost 10 years before publishing its 2003 paper. MapReduce was published in 2004 and finally, in January 2006, Hadoop was formally hatched from Nutch 107. The next two years witnessed a whirl of activity in the technology stack, paving the way to the first Hadoop Summit in March 2008. HBase, a columnar store based on Google's BigTable had also emerged in 2007 and today forms the foundation for many NoSQL stores including Apache Cassandra. Although many anticipated rapid change in the way we develop technology, it wasn't so clear back then where the change would come from and the central role that big data would come to play.

GLOBAL **TURMOIL**

In 2008, the sector was rocked with the collapse of the global markets – triggering what became widely termed the credit crunch. Not since the 1929 Wall Street Crash had the financial community seen 12 months like it. Merrill Lynch, AIG, Freddie Mac, Fannie Mae, HBOS, Royal Bank of Scotland, Bradford & Bingley, Fortis, Hypo and Alliance & Leicester all had to be rescued from the brink of collapse. Lehman Brothers didn't escape so lightly and filed for bankruptcy.

Since then, recovery has seen waves of regulation forced into institutions. Reporting and compliance is now an industry in itself, one which costs billions to run. As a result, for a long time financial services IT providers were so focused on reporting that it diverted their attention from innovation. But while MIFID 2, Frank-DODD, FRTB, C-CAR and other regulatory standards continued to impact the shape and pace of innovation, the mood was starting to change.

FRESH IMPULSE **FOR INNOVATION**

The last three years have seen a renaissance of IT innovation across the banks. The rate of adoption now allows companies to leverage technology not only for compliance and regulation, but to also benefit from additional data insights, data agility and a growing range of valuable use cases.

Traditional innovation incumbents are adopting data-warehouse replacements by using the de facto Hadoop standard. Having at last overcome many of its past challenges around performance, batch-style semantics and complexity, Hadoop is seen as a true data platform. It is the data lake where a plethora of tools are available to build any type of ecosystem and support multi-faceted views for different groups of users.

More recent financial services innovators are making a strong play for Apache Cassandra. It can leverage a less complex data environment and use some of its innovative data centre features to become cloud enabled.

When Apache Spark was launched in 2012, it forcibly upstaged Apache Storm. It has emerged as the SQL for big data platforms and everyone has built a Spark connector including Cassandra, Couchbase, Hadoop, MongoDB, etc.

TODAY'S STATE OF PLAY

So far, 2016 has been an exciting year for big data. In Europe and North America, Hadoop projects continue to run at scale. You'll find Spark and Spark Streaming at the core of all projects. Particularly interesting is the expanding range of use cases – trade surveillance, fraud detection, market surveillance, real-time analytics and more. Banks feel more confident in extracting value from unstructured data using data lakes and there is an appetite to invest in technology that has proven its worth. A compelling example of this is the Wharf bank with more than 122 Hadoop clusters and over 60 staff running it day to day.

Many of the regulatory requirements and associated solutions have a natural implication for data collection, analysis and reporting. Rather than building bespoke database solutions, forward-looking institutions are increasingly leveraging data lakes – not only for compliance, but to drive innovation, which is once again becoming a true staple of financial services IT.

WHAT LIES AHEAD?

Eight years after the global meltdown, we are finally seeing financial services technology efforts coming full circle, accompanied by unprecedented levels of innovation. Spark is being adopted as the core of many systems. Data platforms have no value without analytics and the move to a standard solution has further accelerated Spark development. As a result, Spark Streaming and SparkSQL continue to evolve and lower the barriers to entry. While no-one can forecast with great certainty what tomorrow holds, the following are all likely to play a key part in the foreseeable future.

- **Streaming technologies** like Spark Streaming fit well with the challenges of a Lambda architecture but, more importantly, Apache Kafka is now seen as the glue that enables disparate systems to function and scale. Like Spark, it has a streaming solution, Kafka K-Streams, which, as a result, will become part of many standard technology stacks.
- **Interactive notebooks** that leverage the latest web technology and roll up visualisation with agile big data technology are now being adopted. Offering capabilities that until recently were impossible, they can now complete in seconds typical queries that would have previously taken 24 hours to run.
- **Graph databases** continue to be experimental for most, however, some very exciting proof of concepts are happening in this field. TinkerPop and Titan, along with the pioneer, Neo4J, continue to be applied to more and more use cases, including blockchain.
- **Machine learning (ML)** is probably the biggest technical innovation underway right now. Traditionally a realm of computer science, when combined with big data and analytics, machine learning creates a utopia that many system designers have long strived for, but without the conventional architecture. Indeed, it is the next progression from traditional analytics and, with seemingly infinite use cases, the field is exploding. Spark supports machine learning and has deployed a 1200 node deep learning neural net to manage most of its development activities. But ironically, it's Google with TensorFlow that is pushing the boundaries – which seems to bring us rather serendipitously back to where it all began.

WHERE WILL THE BIG MONEY BE?

To conclude, big data continues to become less about the doing and storing and more about the what if and analytics, in other words, deriving value from data and developing new and exciting use cases. Storing data was solved with the first generation platform, which, albeit fragmented and complex, proved its value. The second generation involved Spark. The maturation of that wave brought agility and simplification as key benefits. The pending wave of Fintech disruption – building around blockchain, mobile, digital, faster networks and cloud-first – are set to make the next ten years much more exciting than the last. So the big question now is where should you focus your efforts? Of course, there's no easy answer, but I'd take a fairly safe guess that the learning machines are coming and the smart money will soon follow.

1996 Data storage became more cost-effective for storing data than paper

1997 The term big data was used for the first time by M. Cox and D. Ellsworth

2000 According to a study by P. Lyman and R. Varian in 1999 the world had produced approx. 1.5 exabytes of unique information

2001 3 Vs – data volume, velocity and variety – mentioned for the first time in D. Laney paper

2003 Hadoop first mentioned in Google file system paper

2004 MapReduce published

January 2006 Hadoop born from Nutch 107

2007 Hbase emerged

2008 Initial release of Apache Cassandra as open source project

March 2008 First Hadoop Summit

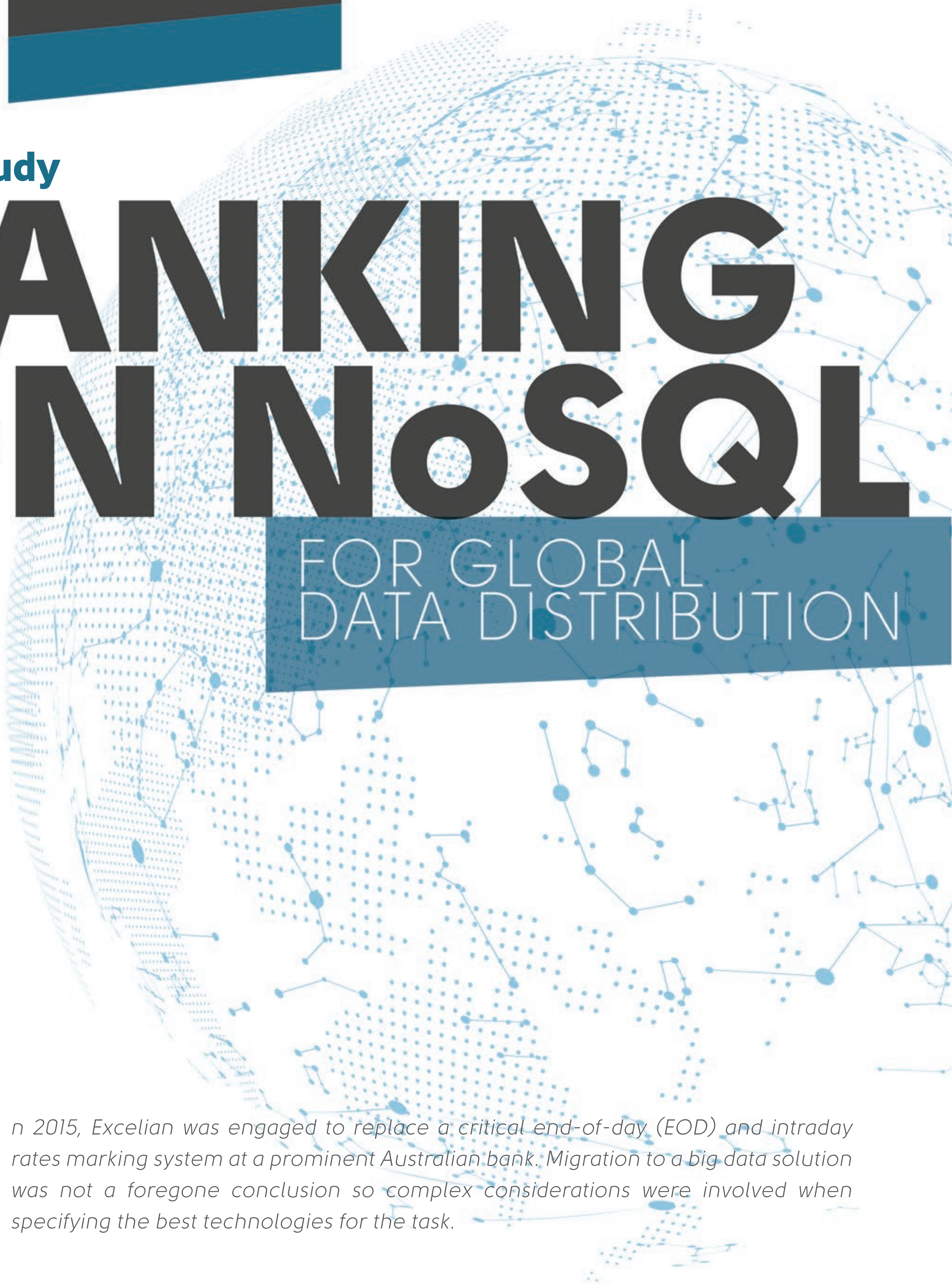
2008 Global Financial Crisis

2011 Initial release of Apache Kafka as open source project

2011 World's information storage capacity grew at a compound annual growth rate of 25% per year between 1986-2007

2012 Apache Spark launched

Mark Perkins
Conrad Mellin
Aleksandr Lukashev
Alexander Dovzhikov



Case study

BANKING ON NoSQL

FOR GLOBAL DATA DISTRIBUTION

In 2015, Excelian was engaged to replace a critical end-of-day (EOD) and intraday rates marking system at a prominent Australian bank. Migration to a big data solution was not a foregone conclusion so complex considerations were involved when specifying the best technologies for the task.

THE NEED FOR SPEED AND SCALE

Almost 20 years old, the existing marking system was written in Sybase, C++, Python and Unix – technologies typically used by investment banks in the late 1990s and early 2000s. Until the 2008/9 global financial crisis, the solution could handle the prevailing low trade volumes. But as business became more flow driven, the framework struggled to scale to the globalised, high-volume trading model that has since evolved.

Complex set-up made it hard for the business to directly maintain the configuration of rates, so it had to rely on additional technology to make changes, which pushed up time and costs. When combined with the additional expense of using Sybase to replicate the volumes of real-time data required worldwide, the total cost of ownership became prohibitive. A scarcity of C++ skills and its limitations for rapid development further compounded the problems. Faced with these challenges, the bank wanted to completely overhaul the system using technologies that could deliver the rapid development speed and scalability to meet its business needs.

KEY REQUIREMENTS

The new marking system had to meet a range of key criteria.

- **User-friendly** – create a flexible user interface so business users could easily define and persist rules about the source of raw data that constitutes a mark (ie whether it is data snapped from market data vendor systems or retrieved from a file/database saved to by traders) and perform automatic calculations on saved marks to produce new marks.
- **Fast** – persist official marks to a lightweight storage layer able to handle high volumes with frequent changes while also distributing data globally, then save to recognised 'sources of truth' for auditors and historical analysis. It was accepted that retrieval from the ultimate official system – ie the Sybase relational database (RDBMS) or even tape – would take significantly longer.
- **Auditable** – enable auditors to play back the trail to see exactly where any given mark came from. This required long term storage capabilities to ensure records are persisted in accordance with regulations.
- **Easy to inspect** – ability to load 'past-date' rate data from the source of truth via the lightweight storage layer for inspection by users.

ASSESSMENT AND ARCHITECTURE PRINCIPLES

The decision to employ a NoSQL solution seemed obvious given the need for a data lake/playground where large quantities of unstructured data can be quickly housed for temporary storage, accessed, modified and replicated globally before finally being written to the RDBMS where it would not be accessed frequently.

Having considered the options, we chose Apache Cassandra for a variety of reasons.

- The appeal of a cost-effective open source (fix-or-amend-it-yourself) solution.
- NoSQL solutions typically have a flat database structure with no joins or indices, perfectly suited for the high volumes of data and performance level required to support the critical nature of EOD marking.
- With a ring partitioning of the keys and data distributed equally among the cluster, the Cassandra model provides high resilience and maximum throughput. Although an existing cluster of high-end hardware was used – which doesn't exactly fit Cassandra's commodity hardware mould – it was fit for purpose.
- Data filtering and manipulation was executed at Java application level, using a minimalist Cassandra query to retrieve or update the data.
- Creating a service layer above Cassandra to persist/retrieve all data guarantees data quality by ensuring there is no direct data access or modification by users – which had been a regular feature of the old set-up.
- The availability of high-quality monitoring tools, such as DataStax OpsCenter, also increased confidence in the technology choice. As part of the DataStax Enterprise offering, OpsCenter has proven extremely powerful for rapidly understanding the state of the cluster and performing administrative tasks.

NoSQL CHALLENGES

Every system has its pros and cons, and our experience with Cassandra on this project was no exception. By highlighting some of the key issues to anticipate, we hope that our insights will prove useful for anyone looking to switch to a NoSQL solution. Given that most developers have an RDBMS background, a significant shift in thinking is required from the application programming perspective.

- **Data integrity** – it was vital to ensure consistency of data underpinning the cluster in different regions. Most modern NoSQL solutions deliver eventual consistency focused on data availability and partition tolerance. While this is how write operation performance is achieved, regional consistency was equally important. Cassandra extends the concept of eventual consistency by offering tuneable consistency. We allowed a write operation to succeed when 50%+1 nodes in a local data centre reported a successful completion of the operation, with other remote data centres set to be eventually consistent.
- **Query performance** – Cassandra's CQL query language is very similar to SQL. However, by comparison, CQL proved somewhat restrictive. So while Cassandra solves the problem of potentially time-consuming queries by forbidding anything at CQL level that can't be executed in constant time, it transfers responsibility for query performance to the application code. Although this makes any query performance problems explicit, it requires significantly more effort from a developer.
- **Latency** – without indices, queries couldn't be optimised to exploit the known structure of housed data. Secondary indices have worked well in classic relational database systems for years but only when underlying data was confined to a local server. In a distributed environment, secondary indices are not recommended as they may introduce latency. Cassandra indexes should only be used when data has low cardinality.
- **Backwards compatibility** – API changes between major Cassandra versions have broken builds and required some re-engineering of our internal code base. This could be both a development challenge and risk. However, Cassandra recently adopted a 'tick-tock' release model, which may improve things.

BENEFITS OF SWITCHING TO NoSQL

- **Global speed** – while large volumes being moved worldwide still put tremendous pressure on the network, the speed of global data distribution has largely resolved the Sybase global replication issue.
- **Open source** – the benefits of Cassandra's open source status are significant and enable easy pinpointing of bugs or potential enhancements, which has driven the bank to seek further open source solutions.
- **Real-time data** – Cassandra's underlying data model of fast, efficient hashmap storage has delivered the near real-time properties that users wanted.
- **Ease of use** – the ease of administering the system, especially for the initial set-up and addition of new nodes to expand the cluster has greatly enhanced the user experience.

SYBASE VERSUS CASSANDRA

In order to run a comparison of data store performance before (Sybase model) and after (Cassandra model), a test environment was set up with two databases containing the same data, as illustrated in Figure 1:

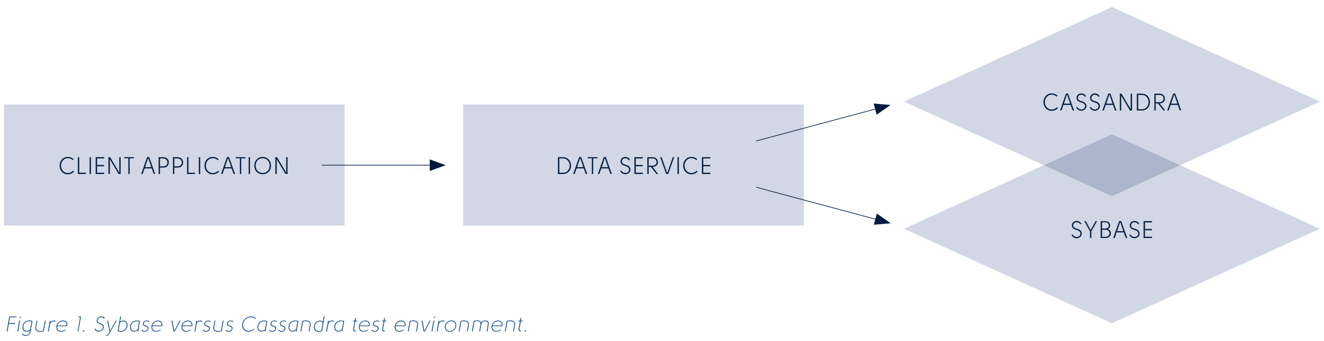


Figure 1. Sybase versus Cassandra test environment.

The data sets were loaded, different volumes of records from both databases were tested and times measured. In Figure 2, a thousand records corresponds to roughly 1Mb of data in the database and on disk. In all cases the results showed that Cassandra significantly out-performed Sybase by a factor of three times or more.

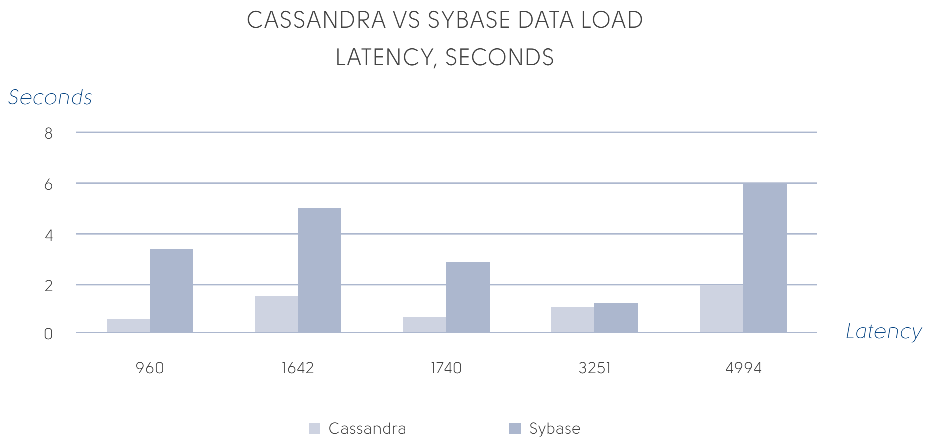


Figure 2. Timings (# of records vs time).

MATCH YOUR TECH CHOICE TO MIGRATION NEEDS

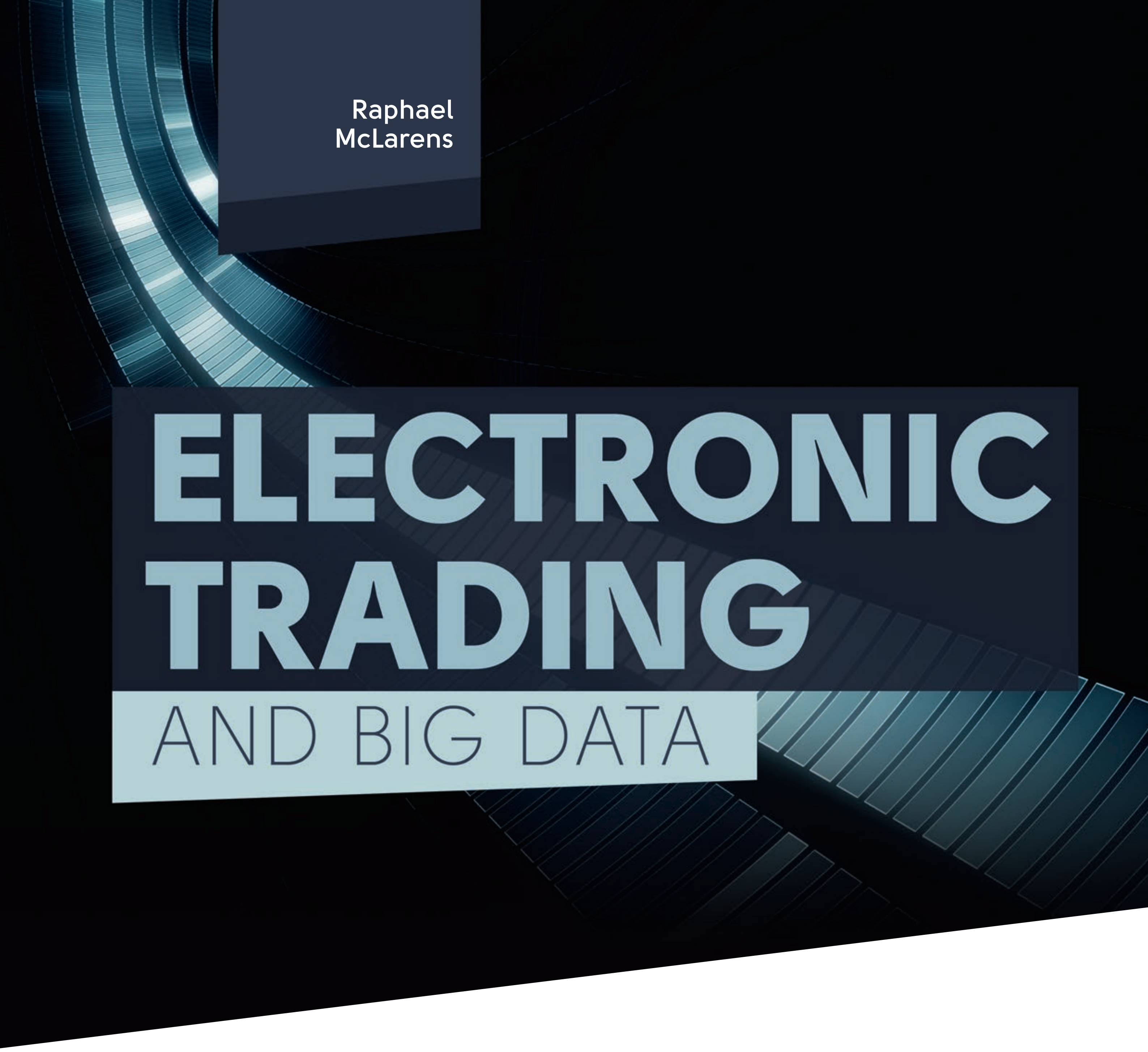
Migration from a relational database to a big data store is a major undertaking and it's not always obvious that NoSQL will offer the best approach. Too often it can seem as if migration to a big data solution can readily resolve all existing RDBMS woes. However, it's not a foregone conclusion and sometimes some fine-tuning of performance, or de-normalisation are all that is required.

For this project, Cassandra was the best fit for the task, however, we highly recommend exploring all the big data technology options to make the best match for your next migration specification. Here are a few suggestions:

- Seriously consider the underlying data types you need to store, for example, column store versus document. Consider too the consistency and partition (CAP) theorem and de facto solutions. For instance, placing more weight on Consistency and Partition Tolerance might point to the suitability of MongoDB or Redis, whereas Availability and Partition Tolerance would suggest Cassandra or CouchDB.
- Bear in mind the benefits of a NoSQL solution that supports a rich query language. It will lessen the burden on developers with an RDBMS background and accelerate time to delivery.
- Finally, for mission-critical systems in an enterprise setting, you could consider a DataStax Enterprise solution. Its many advanced features can greatly improve production deployment, including the enhanced OpsCenter, caching and Spark master resilience, which are essential when running at scale. When combined with increased testing and integration with other key technologies, these powerful tools can greatly enhance a vanilla Cassandra deployment.

Raphael McLarens

ELECTRONIC TRADING AND BIG DATA



Rapid advances in low-cost cloud infrastructure, the falling cost of big data toolkits and the rise of scalable, distributed, parallel computing are transforming the ability to carry out tasks that until recently were unfeasible and creating opportunities to bring disruptive new trading ideas to market.

Capital markets have proved to be early adopters of technology in a range of areas. A good example of this was the rise of electronic equities trading in late 1990s, later extending to other asset classes such as futures and FX.

In the last decade the technology arms race has gathered pace to improve performance in areas such as speed-to-market, asset coverage and algorithms. Away from the high-frequency trading world, with its narrow focus on reducing latency by micro- or nano-seconds, attention is turning to the potential of big data technologies to generate disruptive trade ideas and catch up with retail players in this space.

RESEARCH APPLICATIONS

US-based FinTech company Kensho (**www.kensho.com**) claims to have created the world's first computational knowledge engine for investment professionals. It combines machine learning, unstructured data engineering and powerful analytics that marry masses of financial data, such as stock prices and economic forecasts, with information about world events to produce unique insights and analyses. Instead of using traditional drop-down menus to select particular products, currencies or date ranges, Kensho's highly intuitive interface enables you to pose millions of different questions in plain English via Siri's speech search or IBM's Watson command-line search.

Let's take a scenario like a Fed rate hike, Chinese New Year or a natural disaster such as a hurricane. Kensho can rapidly assess which market sectors have performed better, for example, commodities or stocks, US or Europe and so forth. Within minutes it generates a detailed report of past performance trends – something that would take several analysts days or weeks to research through standard approaches. The benefits of this technology are already being utilised by big market players including Kensho backer and customer Goldman Sachs; JP Morgan; Bank of America; and CNBC which regularly uses it for market commentary.

TRADING APPLICATIONS

Statistical analysis – once the domain of quants and mathematically inclined traders – used to be primarily focused on structured data. But given the rapid growth of unstructured data, big data technologies that use machine learning/artificial intelligence (AI) to filter and rationalise information offer game-changing potential.

Consider, for instance, the benefits of linking social media feeds into black-box style trading systems. In the past, several black-box trades were triggered erroneously when old news was broadcast by accident or a Twitter feed was hacked. For example, in April 2013, the Dow Jones dropped 143 points after a fake Associated Press tweet said the White House had been hit by two explosions[1]. The black-boxes had reacted purely to certain keywords on a few specific Twitter accounts. Fortunately, recent advances in machine learning and statistical analysis, enable scrutiny of a much broader data set to verify the latest breaking news story or at least assign probabilistic indicators around it.

On the buy side, many hedge funds have been turning to more sophisticated mathematical models to drive their trading strategies, with several, such as Bridgewater, Two Sigma, Renaissance and Point72, investigating the benefits of big data. While naturally guarded about their plans, they hope the application of machine learning to masses of structured and unstructured data will give them an edge both over the previous generation of narrow quant models and, more importantly, the competition. The aim is not just to make research and analysis more efficient, but to design computer programs that can detect new patterns in the avalanche of data and generate new trade ideas.

Aidyia, a start-up based in Hong Kong, is mimicking evolutionary theory in its quest to optimise trading strategies. Another start-up, Sentient, is also using evolutionary computing to develop better models. A large set of predictive models are created by analysing various historic big data sets – including multi-language news feeds, exchange data, company accounts and macroeconomic indicators – to find potential correlations in key measures such as stock prices. These are constantly evaluated and the poor ones weeded out, while 'genes' from the successful models are used to seed the next population. Eventually, this produces a strong predictive model. Sentient claims that it now takes trading instructions on what stock to trade and when to enter/exit a position direct from its AI models. Crucially, because these self-learning systems are much more dynamic, they can adapt autonomously as markets evolve – faster than humans can.

[1] *The Telegraph 'Bogus' AP tweet about explosion at the White House wipes billions off US markets, goo.gl/PhLT0s*

SOME CAUTIONARY **CONSIDERATIONS**

While some people say these technologies will give them a probabilistic edge over the previous generation of strategies, it is essential to understand that technology alone cannot fully predict the markets – not even in theory. Other detractors believe there is still too much hype to determine if these approaches have yet uncovered anything truly original and whether spin is playing a part in rebranding some existing approaches as smart or AI-enabled.

There is still a need for human judgment alongside technology-based approaches. One potential drawback of machine learning algorithms, for example, is to mistakenly spot patterns where none exist. This phenomenon, described as overfitting, occurs when a correlation is found between unrelated datasets. The example below on cheese consumption neatly illustrates this point.

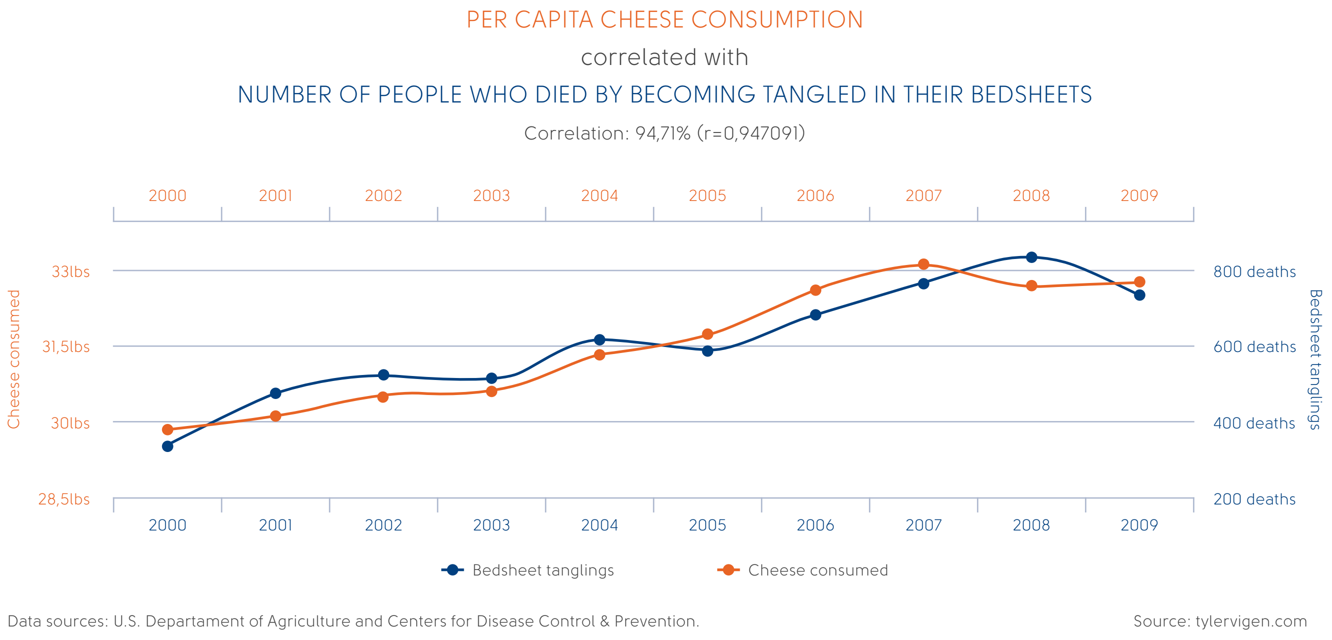


Figure 1. The overfitting phenomenon in action.

AN EVOLVING **TECHNOLOGY LANDSCAPE**

It's important to note that the ability to probe big data has only been possible thanks to the dramatic technology advances and enormous growth in low-cost cloud infrastructure, combined with the rapid rise of scalable, distributed, parallel computing. These have transformed the ability to carry out tasks with an ease and speed that were unfeasible a decade ago.

Take the case of Ufora, a specialist distributed computing start-up that recently made its software open source. Following the 2008 global financial crisis, Ufora's founder had a painful experience while carefully re-factoring large amounts of 'infrastructure-optimised' code to implement new models. This inspired him to set up Ufora and create a solution to produce more efficient code while enabling the code to be easily modified without impacting speed.

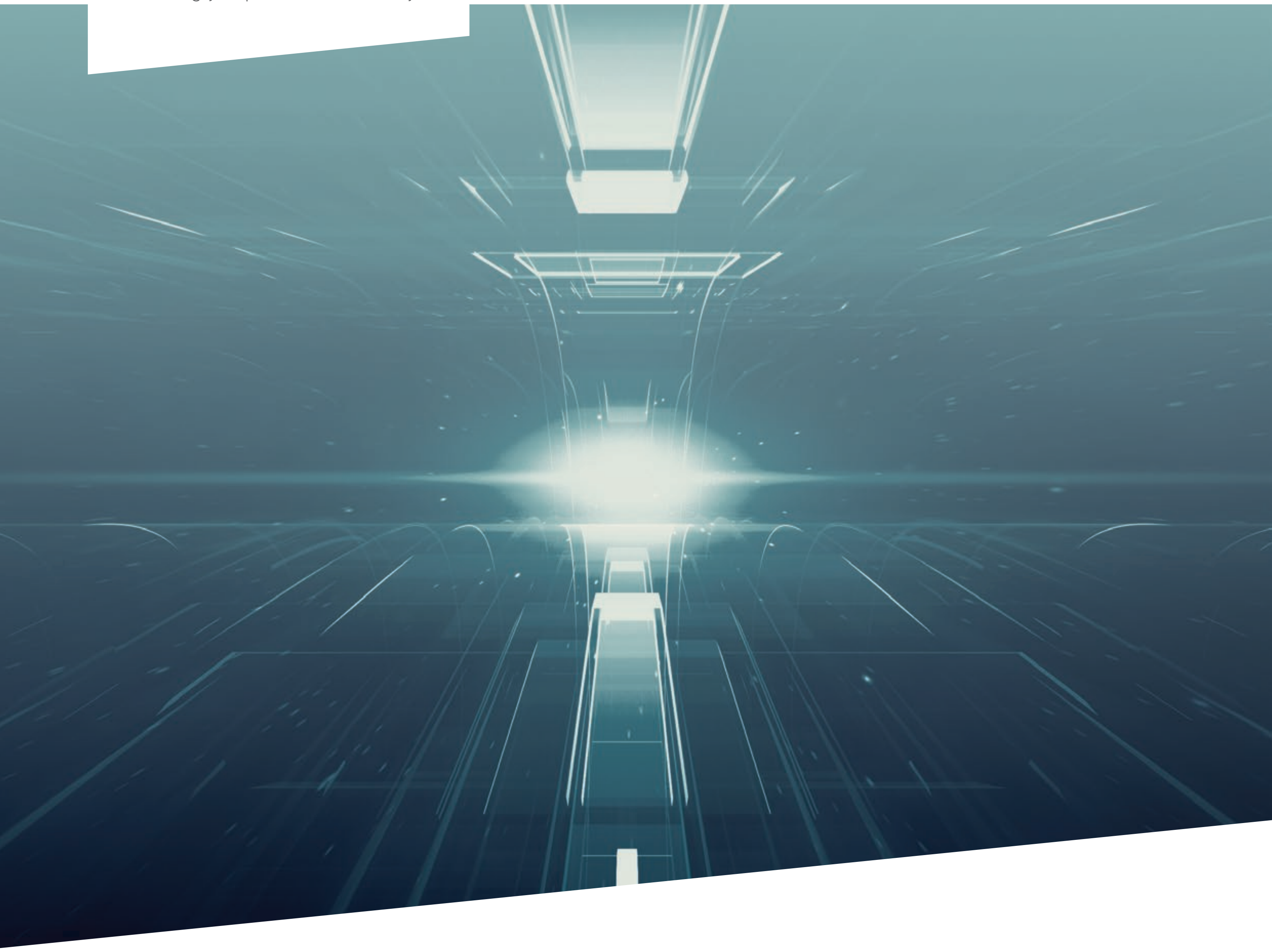
Ufora's platform can take in Python code and automatically parallelise it, then intelligently distribute the threads across a given cluster of local or Amazon Web Services (AWS) machines – effectively multi-threading it. It uses machine learning to understand how best to allocate and manage resources while the program is running, so can handle machine failures gracefully. One headline figure that demonstrates the power of this approach was when a conjugate gradient algorithm (iterative, numerical method) was applied to a matrix with one trillion elements. Within 45 minutes it had delivered a solution by leveraging Amazon Spot instances – essentially splitting the compute job up across 500-1000 cores – for a cost of just $10.

EXPLOIT THE TRUE **POWER OF BIG DATA**

The evolution of technology will continue to transform a range of industries, including financial services. Falling infrastructure costs and growing availability of mature software – including open source – are making the tools to manage big data far more efficient and affordable. Companies that still see IT purely as a support function. to the core business will become left behind. By contrast, firms that invest in big data innovation and develop a clear information strategy aligned to their overall business vision will be in a prime position to exploit the full potential of data as an increasingly important commodity.

KEY TAKE-OUTS **IN BRIEF**

- Many banks and hedge funds are trying to use AI/machine learning as part of their big data strategy.
- The cost of big data toolsets is decreasing; even open source tech provides some useful self-learning algorithms.
- Financial services firms need to develop a clear information strategy to keep pace with the competition.



Neil Avery

INTERACTIVE NOTEBOOKS

FOR RAPID BIG DATA DEVELOPMENT

The latest web-based notebooks are considered by some to be revolutionary and a true sign of maturity in the big data industry. By closing the gap between analytics, visualisation and data agility, they create a powerful platform for rapid application development that can leverage many new business benefits.

Optimising agility has been a long-running technology goal: the ability to react fast and respond instantly to change is a key driver of competitive advantage. So, with the latest generation of notebooks coming on stream, the legacy notion that big data is slow and batch-oriented is being turned on its head. Using interactive notebooks as a platform for rapid application development (RAD) is setting a new benchmark for agile data analytics. And, for the financial services sector they combine the trusted flexibility of Excel with the power of big data.

First generation notebooks offered an interactive computational environment, in which you can combine code execution, rich text, mathematics, visualisation, rich media and collaboration. They proved very useful at the time in fulfilling the needs of reproducible research. However, because they run on a single desktop, with restricted storage and power, their use beyond academic circles has been limited.

By contrast, the new generation notebooks are browser-based and utilise big data technology. As a result, they provide an agile, interactive experience similar to Excel but with the benefits of a visualisation layer that leverages analytics to process terabytes of data. The level of interaction enables real-time, ad hoc analysis which leads to greater business agility.

BLENDING AGILITY, FLEXIBILITY AND POWER

The interactive notebook platform is achieved by providing a glue that combines the different technologies to create a rich, integrated, user-oriented environment that enables seamless collaboration. A layer on top of Spark – called SparkSQL – provides a user-friendly interface that allows expressions to be presented in familiar SQL-type syntax.

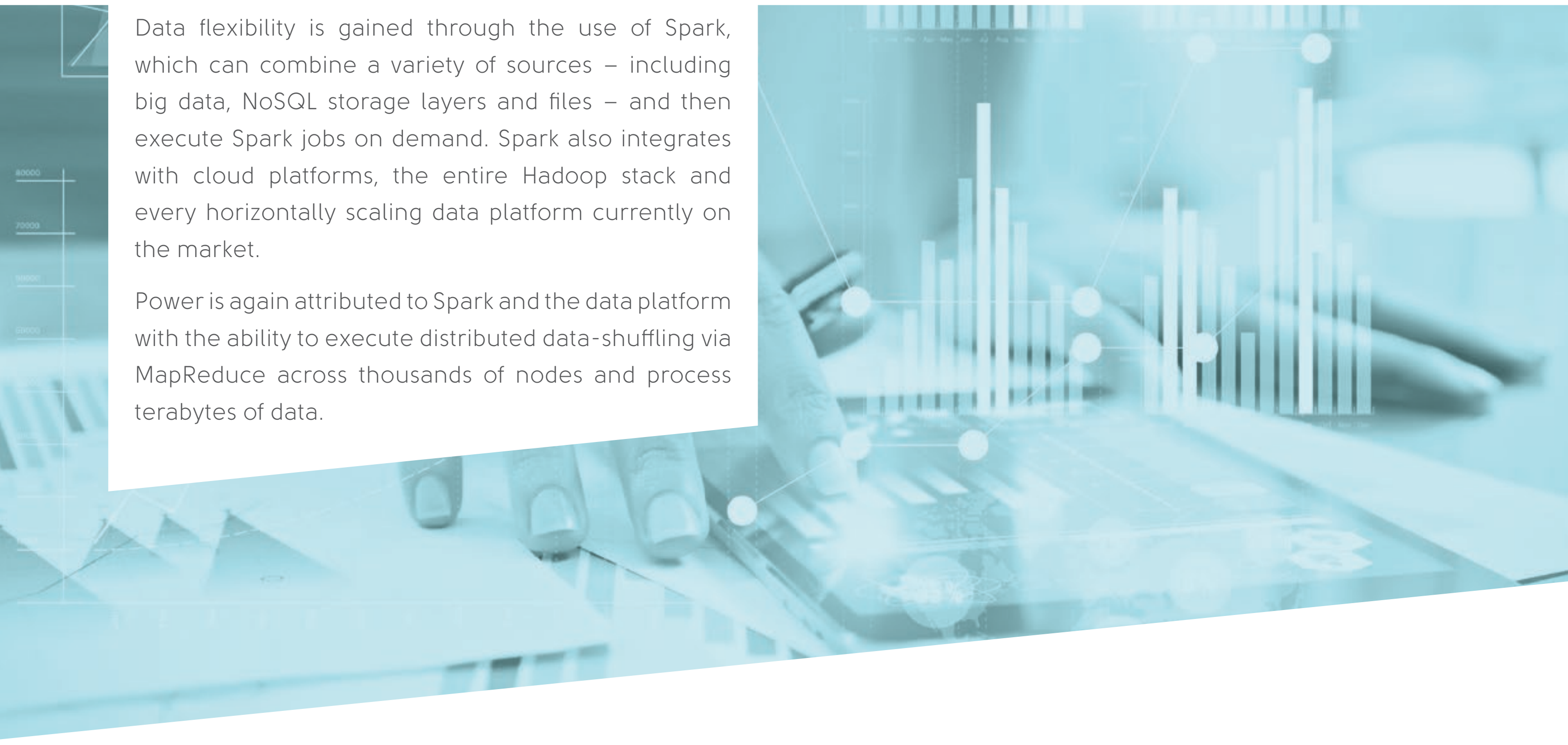
Agility is achieved by leveraging the power of the latest HTML5 runtime found within compatible browser platforms. Rich HTML features support inline script-editing and real-time rendering using SVG graphics and powerful charting libraries such as D3 for dynamic data visualisation.

Data flexibility is gained through the use of Spark, which can combine a variety of sources – including big data, NoSQL storage layers and files – and then execute Spark jobs on demand. Spark also integrates with cloud platforms, the entire Hadoop stack and every horizontally scaling data platform currently on the market.

Power is again attributed to Spark and the data platform with the ability to execute distributed data-shuffling via MapReduce across thousands of nodes and process terabytes of data.

BENEFITS FOR FINANCIAL SERVICES

Traders, quants and other data-heavy roles within financial services can benefit from the notebook platform. For example, what-if analysis can easily be overlaid and executed on-demand to provide immediate feedback. Backtesting, analytic validation and other regular functions that may require a formal release to production can also now be done – as and when needed – and the results viewed simply by visiting a webpage. Other types of post-analytics are also possible. The open nature of the platform makes a plethora of options accessible to anyone with sufficient domain knowledge to derive data insights that were never previously available.



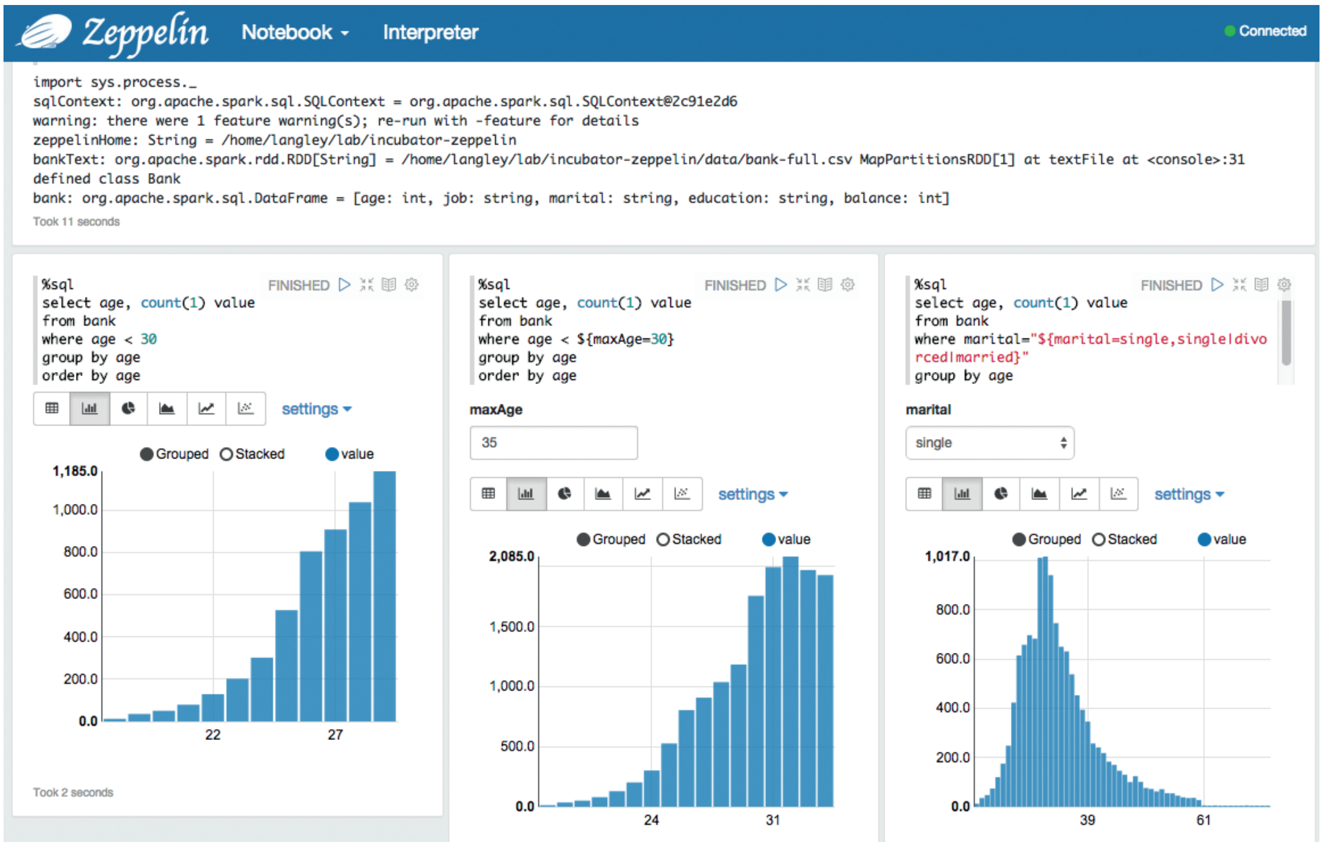
POPULAR INTERACTIVE NOTEBOOKS

Currently, the three most popular notebook platforms are Apache Zeppelin, Databricks (SaaS) and Jupyter (IPython), which all have many potential use cases:

- Facilitate adoption of agile development
- Enable data visualisation
- Analyse what-if and ad hoc scenarios
- Prototype new applications
- Support an ETL pipeline
- De-risk big and fast data project deliveries.

APACHE ZEPPELIN

Apache Zeppelin is an Apache incubation project with Spark functionality at its core. With its multiple language back-end, Zeppelin is language-agnostic, so will likely attract more interpreters because they are easy to add via the API provided. Zeppelin's built-in interpreters include Spark, SQL, Scala, Shell, Markup, Cassandra and many others. While it is very similar to Jupyter/IPython, the user interface is more appealing and it supports a richer range of interpreters as you can observe in Figure 1 below.



```
import sys.process._
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@2c91e2d6
warning: there were 1 feature warning(s); re-run with -feature for details
zeppelinHome: String = /home/langley/lab/incubator-zeppelin
bankText: org.apache.spark.rdd.RDD[String] = /home/langley/lab/incubator-zeppelin/data/bank-full.csv MapPartitionsRDD[1] at textFile at <console>:31
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string, marital: string, education: string, balance: int]
Took 11 seconds
```

```
%sql
select age, count(1) value
from bank
where age < 30
group by age
order by age
```

Took 2 seconds

```
%sql
select age, count(1) value
from bank
where age < ${maxAge=30}
group by age
order by age
```

maxAge: 35

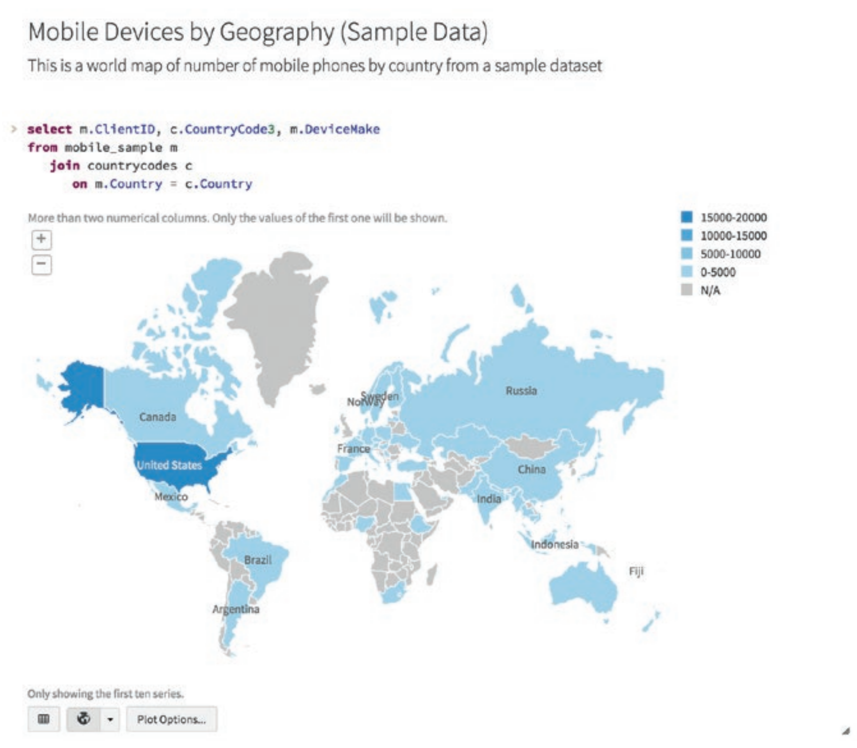
```
%sql
select age, count(1) value
from bank
where marital="${marital=single,single|divorced|married}"
group by age
```

marital: single

Figure 1, Apache Zeppelin provides a rich web environment to enable interactive data analytics.

DATABRICKS NOTEBOOK

The Databricks notebook, illustrated in Figure 2 on the following page, is broadly similar to Zeppelin, however, being available as a cloud-based SaaS pay-per-use model (hosted by Databricks), it delivers all the benefits of lower up-front costs. Databricks, which was founded by the creators of Apache Spark, is continuously pushing the forefront of data analytics, which ensures its platform offers the most up-to-date Spark features.


Mobile Devices by Geography (Sample Data)

This is a world map of number of mobile phones by country from a sample dataset

```
> select m.ClientID, c.CountryCode3, m.DeviceMake
  from mobile_sample m
     join countrycodes c
        on m.Country = c.Country
```

More than two numerical columns. Only the values of the first one will be shown.

Only showing the first ten series.

Figure 2. The cloud-based Databricks notebook offers the latest Spark functionality.

JUPYTER NOTEBOOK

Promoting itself as offering open source, interactive data science and scientific computing across over 40 programming languages, the Jupyter/IPython notebook is well funded and very popular in academic circles. Its growing list of academic backers includes the University of California, Berkeley and California State Polytechnic. In 2015 the Jupyter project received a $6million funding boost to extend its capabilities as a tool for collaborative data science, creation of interactive dashboards and rich documentation workflows.

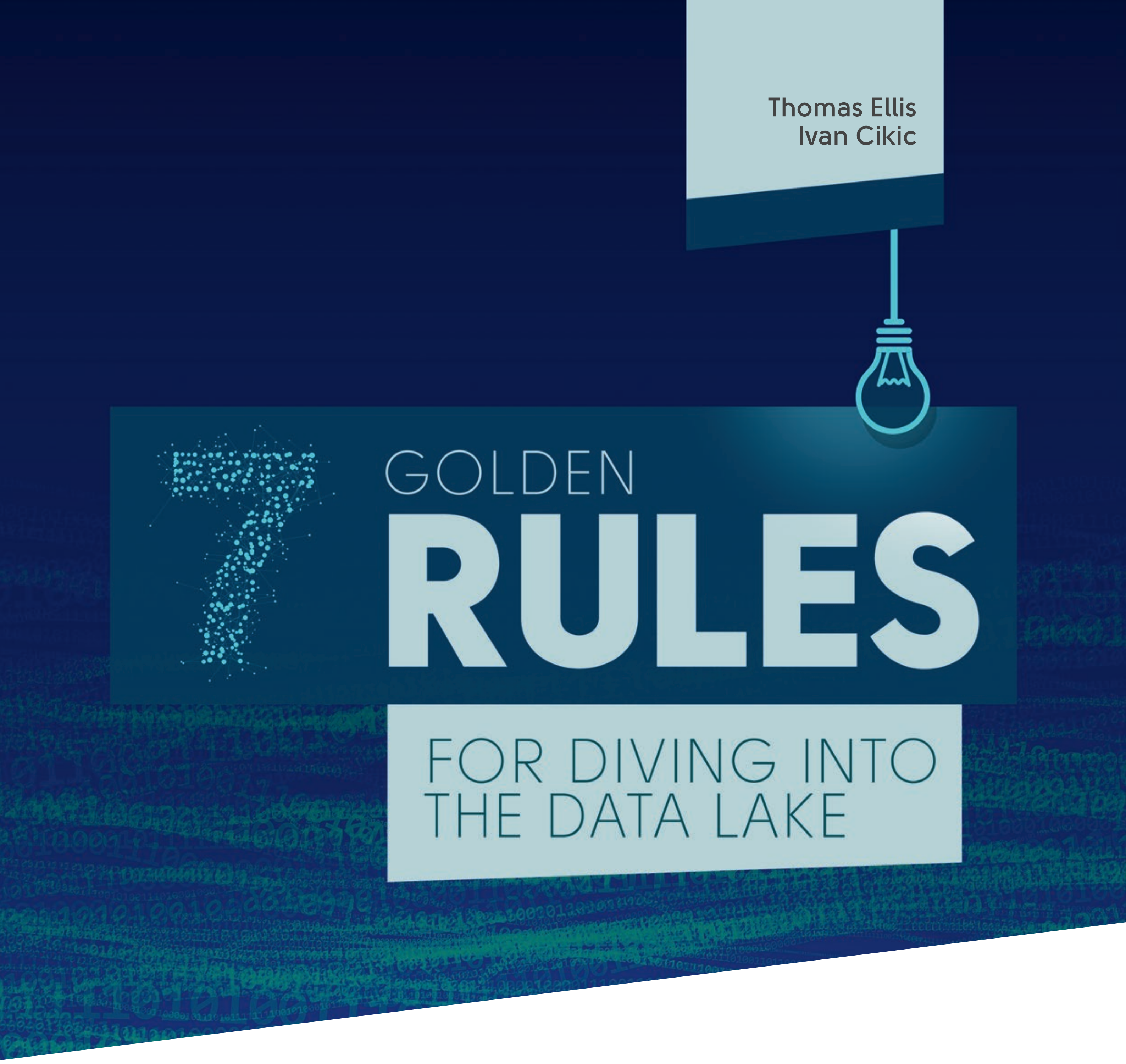
NEXT STEPS FOR NOTEBOOKS

To conclude, the potential of web-based notebooks to support rapid application development is as revolutionary as Apache Spark itself. Many users already regard Zeppelin and similar tools as a data science studio. In practice, we see the power of notebooks as having an exciting future that will give businesses access to novel benefits that have not been available until recently.

Closing the gap between analytics, visualisation and data agility provides tight feedback cycles that can be leveraged to create many unique commercial advantages. Removing the layers of business analysis and data modelling means that analysts with complete domain knowledge can now gain rich insights and utilise notebook power that ultimately combines the flexibility of Excel with big data. Whatever comes next in notebook innovation, it seems clear that they have a big future as a dynamic platform for big data application development.

Many thanks to Deenar Toraskar from ThinkReactive for his contribution to this article.

Thomas Ellis
Ivan Cikic



7 GOLDEN RULES FOR DIVING INTO THE DATA LAKE

Here at Excelian we are seeing more and more large financial institutions turn to data lakes to provide the centralised, malleable repository of data needed to react swiftly and cost-effectively to ever-changing regulatory and market demands. However, it's not always plain sailing, so here we've flagged some of the key issues to consider before you dive in.

Data warehouses, with their rigid data storage structures, have tended to be slow to adapt to the new world order. Data lakes and associated big data technologies, on the other hand, offer an extremely powerful and agile tool that can deliver value fast while reducing cost. But although they offer great potential, difficulties in implementation and delivery can easily lead to a data lake becoming a data swamp. To help you avoid getting bogged down when planning your own data lake, we've pooled our experience to create a few top tips designed to keep you afloat.

RULE 1
ENGAGE STAKEHOLDERS EARLY

Paramount to any good agile delivery is engagement with product owners and business stakeholders. At the outset, try to establish buy-in from the individuals within the business directly responsible for essential parts of the project. Making these stakeholders directly available to your implementation team will ensure prompt answers and a more accurate understanding of the requirements. Ideally, actions required to complete the enquiries from the team should be monitored and expedited.

Key stakeholders will likely include:

- **Product owner** – the individual who develops the project vision and can direct it
- **Data governance guardian** – the person who understands the structure of data in the enterprise and how it flows
- **Infrastructure expert** – an individual (preferably embedded with the team) responsible for managing the lake's infrastructure, ie its environments, hosts, installation and customised distributions
- **Security** – a business-facing individual who can develop requirements around security and user access at a global level
- **End users** – representatives from the teams likely to use the lake, whether supplying or extracting data to/from it.

RULE 2
SET CLEAR GUIDELINES

Requirements for data lake implementations are extremely fluid, so, when it comes to project governance, we strongly advise running the project in an agile fashion. That means each iteration planning session will involve defining a minimum set of features to implement, deliver, integrate and test for exposure to end users. A project like this will inevitably require wide integration across the organisation. It's therefore extremely important to establish high-level guidelines early on, along with early demonstrations of functionality and integration capability with end users.

Initially, work through each iteration with 'just enough' design up front, ideally using whiteboarding sessions with the entire development team. In this rapidly growing field, technology is constantly evolving, so it's important to include all the different perspectives to justify why certain decisions were made.

Any elaboration of the high-level requirements made by the development team should be validated with the wider group of stakeholders at the earliest opportunity, so refinements can be made when it is still manageable to do so.

And, while it may be tempting to migrate existing functionality directly onto the lake, caution is advised. Once you have established a known set of requirements, it's worth first taking a fresh perspective to ensure that the design can use the features of the new environment.

RULE 3
ADOPT A PHASED APPROACH

Consider running projects in a phased approach to continually build on the previous stages. Outlined below is a typical five-phase pattern that works well for us.

PHASE I
PROJECT INITIATION

This includes all the tasks/set-up required before writing the first line of production code.

- Appropriate and install your preferred big data distribution technology (eg Cloudera/Hortonworks/MapR) in your test environments and configure as closely as possible to your target production environment.
    - Secure the test environments to the standards required – including configuration of Kerberos and synchronisation with any enterprise user directories eg LDAP/Active Directory.
    - If the infrastructure for your test environments is unavailable, consider provisioning cloud instances and use these as systems integration test environments until your internal infrastructure is ready. Should test data confidentiality be a concern, anonymise or generate it.

    - Develop/configure a sandbox virtual machine (VM) instance that developers can run on their workstations. This should be a like-for-like copy of your test/target environment, including security – scaled down only to a single host.

- Obtain or develop a suite of generic tools to automate deployment of lake artefacts and deliverables to target environments, including developers' sandbox VMs. These tools should use environment-based configuration to distribute and configure their deployments.

- Obtain or develop a suite of integration testing tools that make it easier for developers to build acceptance and integration tests against a target environment. These tools should interact with your targeted big data technologies (eg HDFS/HBase/Hive) and make it easy to set them up and fill them with test data, as well as to tear them down. The integration tools should utilise the developed deployment tools at the earliest opportunity to ensure adequate and continuous testing.

- Install and configure build servers to perform continuous delivery and static code analysis. The build process should:

    - Build and package the application and install it in a centralised artefact repository running all unit tests

    - Raise any static analysis issues

    - Deploy to a development test environment and run integration and acceptance tests

    - On a successful run of the integration and acceptance tests, deploy to other systems integration test environments. This can either be automated or configured to provide push-button deployment to other environments.

- Codify and document development standards and best practices, using IDE plug-ins to enforce them, eg SonarLint, Checkstyle.

PHASE II
INGESTION

In this phase, you will develop the interfaces required to obtain data from external systems and sources. It is likely that the end users' analytics and extraction requirements are still being determined, so look to simply accept and retain raw data in an immutable machine readable format, eg via Apache Avro or Apache Parquet. Storing raw data this way provides options when you need to rerun analytics, present lake content, generate metadata and manage disaster recovery. Ensure that any processing done on ingested data is idempotent, then should the need arise, that data can be re-ingested as necessary.

As data starts to flow into the lake, it is important to generate and retain relevant metadata – including items such as lineage, type, any business-specific tags and validity. This metadata will be used in a variety of ways, including security and data discovery.

PHASE III
DATA DISCOVERY AND SELF SERVICE

Once data is present in the lake, you need to make it available to potential end users. Initially, these will be power users and data scientists with the security clearance to interrogate raw data and derive analytic and extraction use cases from it. There are a number of options to facilitate this.

- Use your distribution's data governance tool, for example, Cloudera Navigator or Apache Atlas. End users can use these to explore the data on offer and establish uses for it.

- Employ a third-party tool such as Waterline or Tableau. Third parties are producing some very interesting tools that interact with big data technologies to provide a rich data governance toolset.

- Provide power users with notebook access to search and analyse data on the lake. Apache Zeppelin, Hue and Spark notebooks enable advanced users to write snippets of code that directly interact with and display data.

- Build a custom solution – for instance, a simple user interface to effectively display your data catalogue and related metadata can be more than enough to interest end users.

Whatever data discovery route you opt for, be vigilant about security by ensuring it is sufficiently and appropriately locked down to enable only authorised access.

PHASE IV
ANALYTICS AND AGGREGATIONS

As end users' analytics or data aggregation requirements become more refined, you can begin to develop them against the lake. Responsibility for developing analytics can be handled by either the data lake project team or end users themselves. If allowing end users to develop their own analytic solution, it's important that they have the same like-for-like sandbox VM environment as the data lake developers and enough test data to develop their solution. Consider having data lake project team members 'consult' with these end users to ensure they're following best practice and their solution will be compatible with the lake. Ensure early integration with a test environment and that their analytics solution is segregated using its own application user, with appropriate authorisation rules in place.

When the data lake project team develops its own solution, constant communication with the end user is vital to validate and test what is being produced.

PHASE V
EXTRACTION

This phase, which can run concurrently with analytics and aggregation, is used to develop storage of any analytics or aggregation outputs in the lake using the most appropriate storage technology. Again, make this available to the end user by utilising an appropriate technology, for example, for a large batch file output, you could push to HDFS. Alternatively, for key value outputs with fairly low latency access patterns, store in HBase – and have clients pull using REST endpoints – or push data out onto a message bus.

As this data will probably be valuable to other end users, ensure that it is also published to your data discovery utilities, is suitably structured and that relevant security policies are in place.

RULE 4
SAFEGUARD SECURITY

Never underestimate the importance of security – and keep it front of mind at all times. It is imperative to secure your cluster using Kerberos. Due to its fiddly nature, all developers and support staff should become acquainted with it because it will be used to secure all environments, including the developers' sandbox VMs.

You can create a security choke point by surrounding your cluster with a firewall and providing REST endpoint access via Knox. This will greatly reduce your attack surface while simplifying identity assertion and external auditing of communication with the lake.

Each big data technology will come with its own authorisation tooling. Tools like Apache Ranger and Apache Sentry can manage this centrally for you. Of the two, Ranger has more features, including plug-ins for centralised authorisation and audit for most of the big data technologies. Sentry has similar features with a much reduced scope, but is under active development. If your data is especially sensitive, consider using SSL communication between services and additional encryption for data at rest.

RULE 5
PUT MONITORING IN PLACE

At the same time as developing your components, you'll also need to build corresponding metrics. Developers are best placed to incorporate metrics while the components are being built, for example, providing JMX mbeans and publishing them using Jolokia, and logging to an appropriate log facility, for example SLF4j and Log4j 2. Distributors also offer impressive tooling functionality which should be utilised effectively. In conjunction with this, consider employing log/metric aggregation tools like Logscape, the ELK stack or Splunk.

RULE 6
PLAN FOR DISASTER RECOVERY

It's vital to plan for business continuity, should your data lake be compromised. The data lake will largely become a repository of data with analytics running across it, so consider running another hot active site in parallel. All data is channelled to and ingested at both sites, and likewise all analytics and extractions also run synchronously. In addition, consider developing the capability to clean up state and replay. The process of ingesting raw data in the lake is idempotent, so in the event of a problem at a site, you can run a job to clean up the state of the problem confident that once it is resolved, the data can be replayed to generate the correct state.

RULE 7
COMMIT TO BEST PRACTICE

Big data technologies are in a constant state of flux and innovation. There's a fierce competition among distributors and vendors to see their tools adopted as the industry standard, so you'll often see a high rate of change in preferred tooling, methodologies and best practice. With this in mind, it's imperative that your developers stay at the top of their game. Consider creating opportunities for them to learn and share their knowledge and experience in ways that make their work more fulfilling and enjoyable.

These could include, for example:

- Lightning talks, lunch-and-learn sessions and technology roundtables with walk-throughs of developed components
- Team members running training labs for colleagues and users within the organisation to introduce them to new technologies
- Sending members to conferences and them reporting back with their findings and views on hot trends in the sector
- Encouraging developers to contribute to open source solutions.

GET SET
TO DIVE IN

As you'll see, there's a lot to think about when planning a data lake implementation. It's a long article but we make no apology. In fact, it barely gets below the surface of the many crucial elements required for a successful data lake implementation. In summary, these are:

- **Engage stakeholders early** and identify key contacts in relevant teams to radically reduce decision-making time
- **Adopt agile methodology** and 'just enough' high-level design to ensure sound project governance
- **Take a five-phase approach** choosing from commonly used and emerging tools to best suit your needs
- **Put security first from the outset** – it is a cross-cutting concern that is often addressed too late
- **Factor non-functional needs** into your planning, eg disaster recovery, monitoring and best developer practices.

Of course, much of this may seem obvious, but it is often the obvious that gets overlooked and that's when tech projects can start to flounder – especially in such a fast-moving, dynamic and exciting area where things are changing rapidly. However, guided by these key principles, you'll be much better equipped to keep your head above water and ensure that your next data lake implementation goes swimmingly.

Vasiliy Suvorov

JOIN US

ENTERPRISE BLOCKCHAIN ACCELERATOR

Over the last 18 months we have been avidly following developments in the blockchain space. For us and for many others, it began with an understanding of Bitcoin, the original blockchain model. With blockchain now firmly established as one of the most dynamic and disruptive forces in the tech arena, this month we are launching our Enterprise Blockchain Accelerator program. We hope you'll join us on what promises to be an exciting journey…

Although the elusive Satoshi Nakomoto, Bitcoin's anonymous creator, did not use the term blockchain in his paper, his underlying distributed, shared and decentralised ledger technology was truly revolutionary and formed the basis for the original blockchain. As an add-only database shared and simultaneously maintained by the so-called 'miners', it successfully enabled the world's first fully digital, decentralised currency that is not administrated by any bank or government.

A BRIEF HISTORY OF BLOCKCHAIN

Since the first 'genesis' block was placed on the Bitcoin network in 2009, the world has seen roughly three waves of interest and investments in Bitcoin and, increasingly in blockchain, its enabling fabric.

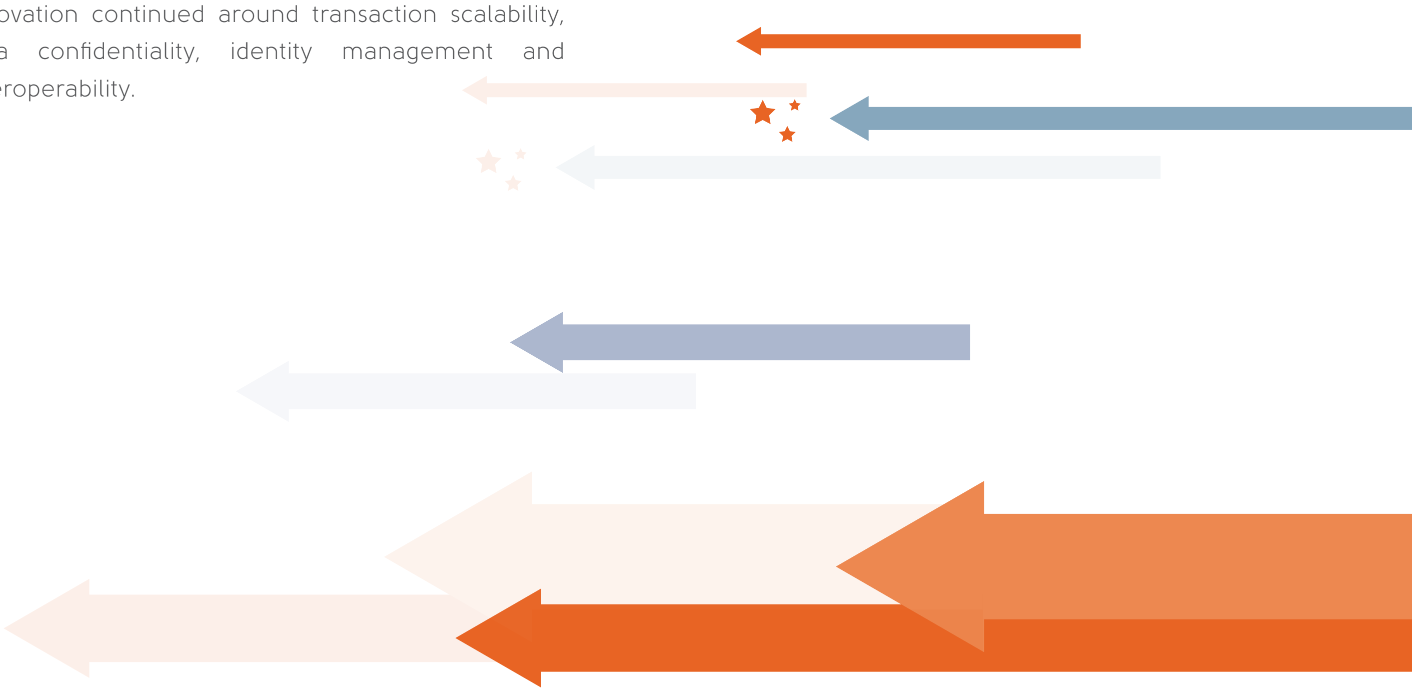
- **Wave 1:** The first wave was all about understanding the Bitcoin itself.
- **Wave 2:** The second was about discovering the blockchain and building alternative 'coins' – various asset tracking and exchange mechanisms that leveraged the unique characteristics of the Bitcoin protocol and its mining network.
- **Wave 3:** Once the limitations of the original design were understood, the arrival of the third wave brought next-generation blockchain designs based on 'smart contracts' – code snippets that are tightly integrated with a blockchain so that participants can add arbitrary business logic to each transaction. These next-generation designs also introduced newer network-wide synchronisation mechanisms also known as 'consensus'.

The later designs ensured that data synchronisation across the whole network was not based on energy-inefficient mining or, more formally, Bitcoin's Proof-of-Work based process. In the meantime, innovation continued around transaction scalability, data confidentiality, identity management and interoperability.

THE FOURTH WAVE

Fast forward to the present and we are witnessing a fourth wave of blockchain investment. With it the unique characteristics of blockchains or, more broadly, distributed ledger technology (DLT), are being applied to real-world business cases. Today, business and technology opportunities to deploy blockchain/DLT solutions are being actively pursued by organisations from the open source community, industry and government-led alliances and associations, including industry incumbents like the Depository Trust & Clearing Corporation (DTCC).

It's quite remarkable that technology which was invented to disrupt banking and money management as we know it, has been embraced by the financial services sector and, to a certain degree, the financial regulators – the very same people it was supposed to displace. In fact, the successive waves of interest described above were actually driven by the business leaders, a rare phenomenon for such a complex emerging technology.



A WORLD OF **OPPORTUNITIES**

Along with many smart individuals and visionary companies, we see amazing potential for this technology. When applied correctly, it offers enormous scope to reduce costs, time and risk in financial markets. It could also enable truly innovative internet of things (IoT) applications, open up markets to new opportunities and drive new efficiencies by minimising or, in some cases, completely eliminating the need for intermediaries.

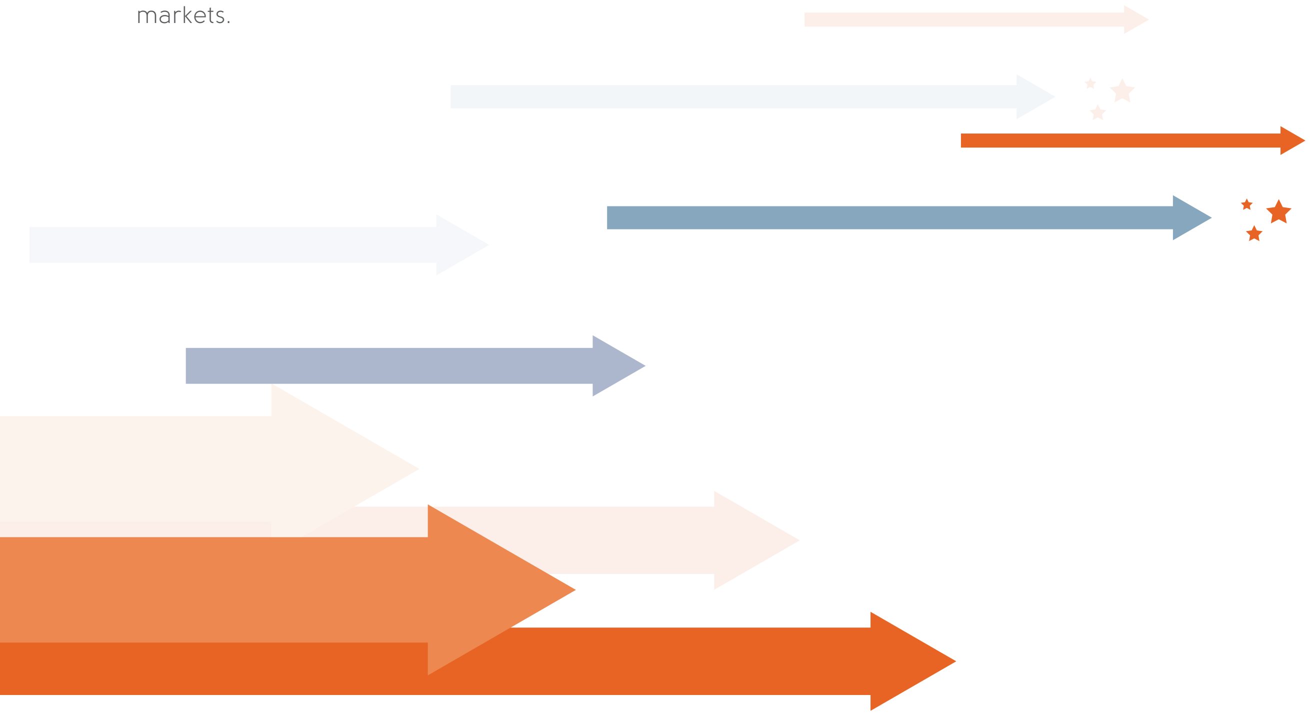
It's important, however, to stress that a blockchain/DLT approach is not a universal solution. Indeed, many applications will be better served by other well-established technologies, such as a NoSQL or relational database, micro-services architecture, business process management (BPM) solution or enterprise service bus (ESB).

Blockchain is undergoing a frenzy of innovation. In some ways, it's a bit like the Cambrian Explosion – the short but highly significant period in the history of evolution when most of the major animal groups appeared. New blockchain/DLT variants are constantly emerging to fill every potential market niche, ranging from libertarian crypto-currencies to highly secure permission-based applications for private financial markets.

WHAT'S AROUND **THE BLOCK?**

It's very likely that all of these applications and related blockchains will start to appear on the market as soon as 2017. Major corporate players are already announcing plans. It is also likely that between now and 2020, business model pivots, lessons learned and government regulation will drive standardisation and convergence. The early adopters are likely to be in the finance and logistics markets, followed by insurance and pharma. We also expect the internet of things to drive demand across all sectors, which will further accelerate the adoption of blockchain.

Looking further ahead to the 2020s, blockchain/DLT will start to become a mainstream IT component. Consequently, we can expect that many future systems will be built around integration of a handful of interoperable blockchains. Some of these will be public and decentralised, such as Ethereum, and some will be run by international consortia. Of course, there will also be completely private blockchain/DLTs as standard, but these will mostly augment or even replace existing database technologies to meet specific intra-company needs.



MAPPING THE ROUTE AHEAD

Our friends at Consult Hyperion have proposed a layered, modular approach to blockchain/DLT design study. This approach makes it easy to understand, or make, design trade-offs and perfectly match the resulting architecture to a specific set of use cases:

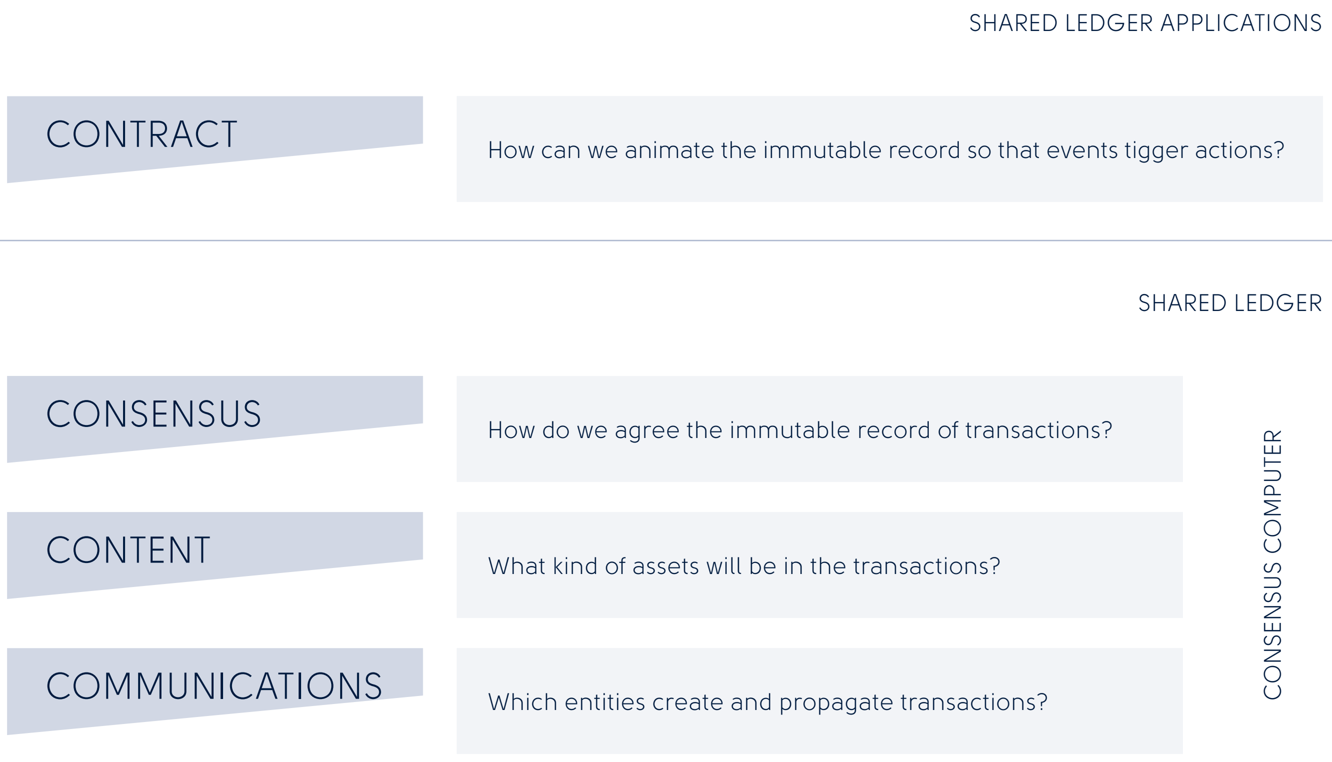

SHARED LEDGER APPLICATIONS
CONTRACT
How can we animate the immutable record so that events tigger actions?
SHARED LEDGER
CONSENSUS
How do we agree the immutable record of transactions?
CONTENT
What kind of assets will be in the transactions?
COMMUNICATIONS
Which entities create and propagate transactions?
CONSENSUS COMPUTER

Figure 1. Consult Hyperion 4x4 model of the shared ledger technology.

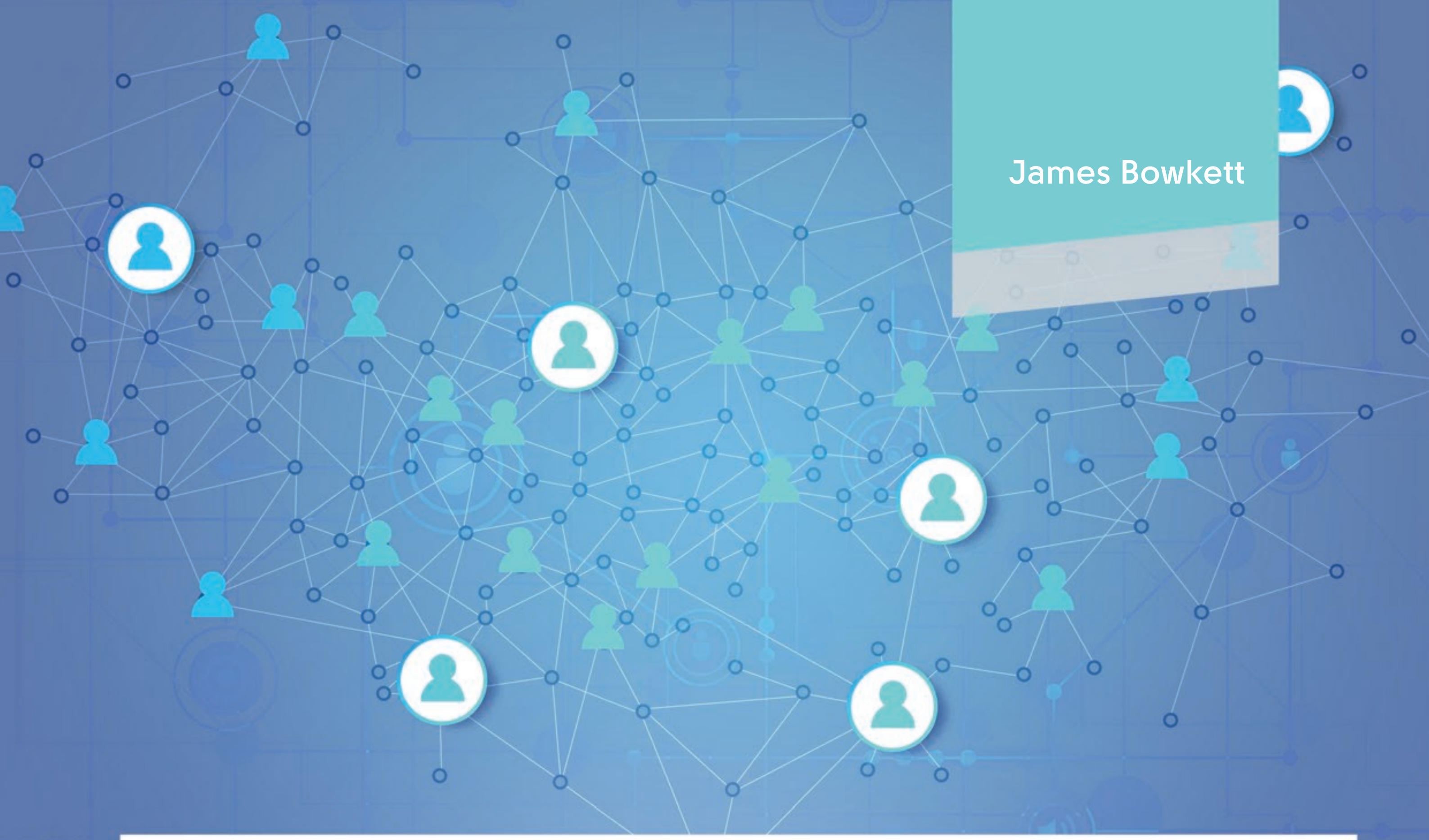
Navigating this new landscape and the road ahead is not easy. We strongly believe that our customers will benefit from hype-free, independent advice based on a solid understanding of their business needs and processes combined with the expertise to select the best technology.

To meet this requirement, we are delighted to announce the launch of our **Enterprise Blockchain Accelerator program**. We'll share a range of resources, including technical articles in this and future issues of TechSpark, blog posts and invitations to join an upcoming series of in-depth webinars on blockchain/DLTs.

With so much more to say on the subject, we'll keep you informed with a range of insights into both the technical and business aspects of blockchains/DLTs. In the next issue, for example, I'll take a closer look at what the future looks like for industry adoption and related challenges.

On page 32 of this issue, you can read the first technical feature in our new series, written by my colleague James Bowkett: *'Blockchain and graph, greater than the sum of their hype?'* In the meantime, we look forward to your company as we move forward on the blockchain journey.

For more information on our Enterprise Blockchain Accelerator program, please contact James Bowkett (james.bowkett@excelian.com) or Neil Avery (neil.avery@excelian.com).



James Bowkett

BLOCKCHAIN AND GRAPH:

MORE THAN THE SUM OF THEIR HYPE?

The buzz about blockchain technology is hard to ignore: it is mentioned in every other finance feature and is a standard item at any Fintech conference. In effect, its data structure is an encrypted temporal linked list of transactions, however, linked lists aren't appropriate for random access. So we decided to integrate one much-hyped technology with one enjoying something of a renaissance – graph databases – to see if the whole adds up to more than the sum of its parts.

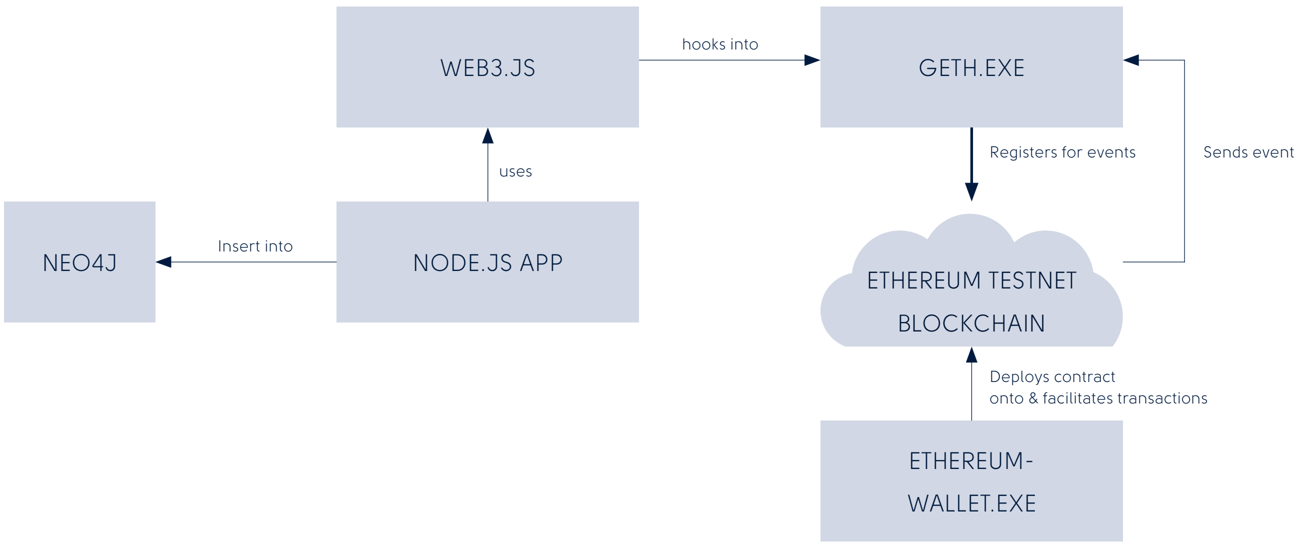
Integrating the two approaches creates a platform for immutable, confirmed transactions, with an appropriate fast index that allows you to report on the data within. In a standalone blockchain application you can only search sequentially for an asset or transaction. But leveraging graph enables you to easily track the source of assets held on the blockchain. This has many use cases, notably around checking authenticity, fraud prevention and detecting money laundering.

Graph is a good fit for transactional data because it allows you to model, query and visualise relationships between entities, in addition to the entities themselves. Such relationships may also contain attributes which can be used for later querying. Contrast this with a relational or document store where the relationships are often modelled as entities in a similar way to the entities themselves. Although this latter approach can and does work, graph allows for a more natural way to think about and model transactional data.

This article is an abridged version of a series of Excelian blog posts where you can find details about the code and more on the approach we used.

BRINGING TO LIFE A BLOCKCHAIN-GRAPH MODEL

To explore in more detail how blockchain and graph work together in practice, we devised a prototype equity share issue using blockchain events to persist sell trades to graph database Neo4j. The architecture of this example is as follows:



Having installed the necessary platforms – node.js, Neo4j, geth, Ethereum (and web3) – it is possible to write and deploy smart contracts written in Solidity, Ethereum's smart contract language. Solidity is reminiscent of Go (Golang), but without the rich choice of APIs – for good reasons, as we will see. The contract is then deployed to the Ethereum network and run on every miner node within the Ethereum virtual machine – hence the restricted APIs – and the immutable record of ownership is created.

Using the Ethereum wallet, the following smart contract is deployed:

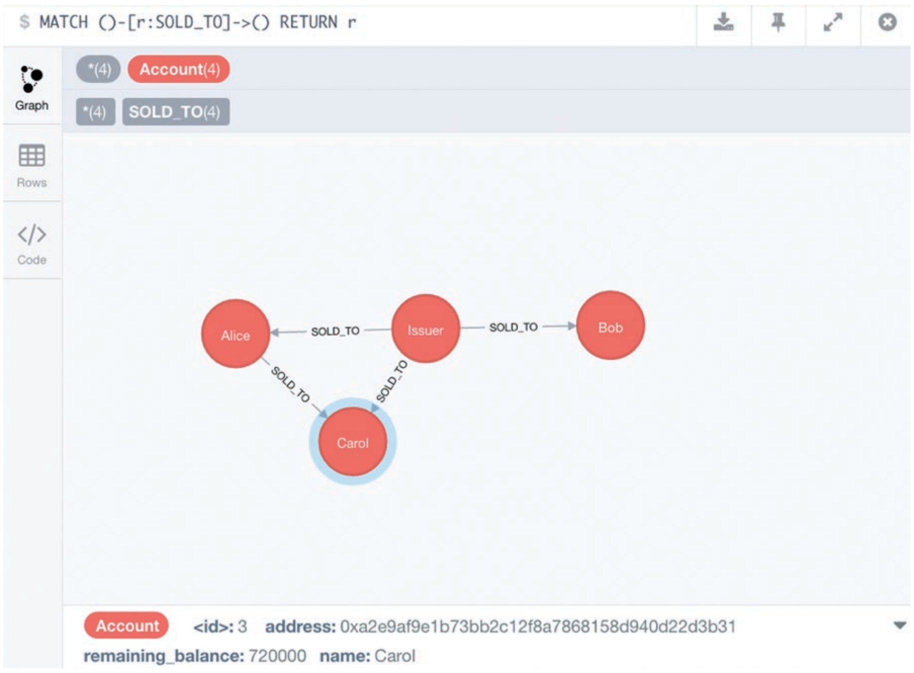
```
contract ShareClass {
  string public name;
  uint8 public decimals;
  string public symbol;
  string public isin;
  string public description;

  /* the register of how many shares are owned by which account */
  mapping (address => uint256) public balanceOf;

 /* Generates an event on the blockchain to notify clients */
 event Transfer(address indexed from, address indexed to, uint256 value);

 /* Initializes the shareclass for the instrument with initial
    supply of all equity assigned to the issuer */
   function ShareClass(uint256 initialSupply,
                     string tokenName,
                     string isinId,
                     string desc,
                     uint8 decimalUnits,
                     string tokenSymbol) {
     // Give the creator all equities
     balanceOf[msg.sender] = initialSupply;
     // Set the name for display purposes
     name = tokenName;
     // Amount of decimals for display purposes
     decimals = decimalUnits;
     // Set the symbol for display purposes
     symbol = tokenSymbol;
 isin = isinId;
 description = desc;
 }

 function transfer(address recipient, uint256 quantity) {
 ensureSenderHasEnough(quantity);
 balanceOf[msg.sender] -= quantity;
 balanceOf[recipient] += quantity;
 // Notify of transfer event:
 Transfer(msg.sender, recipient, quantity);
 }

 function ensureSenderHasEnough(uint256 quantity) private {
 if (balanceOf[msg.sender] < quantity) throw;
 }
 }
```

Analogous to a new share issue, the contract holds the register of how many shares in this issue are contained in each Ethereum wallet, plus information about the instrument itself, such as its ISIN. Each time the contract is traded, it will emit an event notification to all its listeners, that can be listened to using the web3 API in a node application. In this example, the transaction details will be stored in Neo4j, using the following Cypher code.

```
 MATCH (owner:Account),(buyer:Account)
 WHERE owner.address = '${ownerAddress}'
 AND buyer.address = '${buyerAddress}'
  CREATE (owner)-[
:SOLD_TO { amount:${amount}, tstamp:timestamp()}
]->(buyer)
```

This code finds the owner and buyer accounts (nodes) in Neo4j, then creates a new 'sold_to' relationship in the database.

Without any additional code or configuration, Neo4j then enables the following visualisation:


```
$ MATCH ()-[r:SOLD_TO]->() RETURN r
```

Account <id>: 3 address: 0xa2e9af9e1b73bb2c12f8a7868158d940d22d3b31
remaining_balance: 720000 name: Carol

FUTURE POSSIBILITIES AND USE CASES

Our blockchain-graph equity share issue prototype, as described above, could be further extended and integrated with an identity or know-your-customer (KYC) service. This would make it easier to look up and identify each account holder for sending out annual shareholder reports or dividend payments. Because shareholders could manage their own identity and details for their different holdings in one place, the share issuer would be relieved of the burden of maintaining its own shareholder register.

In another scenario, we have identified a strong use case for fraud visualisation. With a few lines of Cypher, you can visualise centres of trading activity for targeted investigation – a functionality that is available out of the box. This makes graph, and Neo4j in particular, a compelling choice for data visualisation.

While blockchain offers undeniable benefits, legislation has a way to go to catch up with the pace of technical innovation in this area. For instance, the laws around property rights still need to be addressed, as does the issue of how to reverse an irreversible, immutable transaction, if it was in dispute. However, we can certainly expect the appropriate law changes to come in time.

In the meantime, it's clear that blockchain and its system of trust is enabling existing applications to work faster and more securely. It is also facilitating new distributed applications. For instance, Honduras has worked with Factom to build a blockchain-based land registry. This will deliver a secure, reliable and verifiable land registry. Although the project is still in its prototype stage, it highlights the value of the trust inherent in blockchain applications, especially in use cases where there is no pre-existing application.

SUMMING UP

Looking beyond the hype, while there are compelling use cases for blockchain (or an immutable ledger) on its own, we see its real value being realised when used as a complementary technology with other storage techniques such as big data. As illustrated in our prototype, using blockchain in combination with graph can provide visualisation by modelling transactions in a form that best fits its transactional data model – so in turn creating additional value. In other words, the output could indeed add up to more than the sum of the parts.

Deenar Toraskar
Neil Avery

DRIVE FAST

FLEXIBLE VaR AGGREGATION WITH SPARK

With banks investing ever more resources to meet increasing regulatory reporting demands, Spark's big data processing methodology offers compelling evidence of a dramatically different approach that could reduce the reporting burden while driving new operational efficiencies.

Since 2008, FRTB, CCAR and BASEL requirements have inundated teams with untenable workloads and up to two-year backlogs. At the same time banks also need to prepare for more reporting changes and the anticipated switch from the long--established Value-at-Risk (VaR) reporting measure to Expected Shortfall (ES). Despite this, many teams still rely on traditional ways to manage and warehouse their data. Big data, and more recently Spark, have come to prominence with analytic use cases that map onto traditional problems, but solve them in a very different way. Spark's methodology provides a flexible and powerful processing stack that could transform the standard approach that many institutions are currently using. When applied to VaR it becomes a simple workflow of Spark tasks selecting, joining and filtering data, as we will explain.

WHAT IS VALUE AT RISK (VAR)?

Of all the regulatory reporting measures that need to be managed, for almost 20 years VaR has been the most widely adopted risk measure. It is used for risk management, margin calculations, regulatory reporting, capital charges and pre-trade decision making. If you have a trading account or a betting account, your broker or bookmaker is likely to use VaR to calculate margin calls. VaR is used for hedge optimisation, portfolio construction and to optimise the tracking error of a portfolio against a recognised benchmark. You can also use VaR to help make risk versus return trade-off decisions when managing the portfolio of assets in your pension fund.

VaR is equal to predicted worst loss over a target horizon within a given confidence interval. For example, if you hold $1000 worth of GOOG shares, one day VaR with a 95% confidence level is $22.

HIGHER THE CONFIDENCE LEVEL	greater the VaR	1 Day var for $1000 in goog at a 99% confidence level is $31
LONGER THE TIME HORIZON	greater the VaR	10 day 95% VaR for $1000 in GOOG is $60
MORE VOLATILE THE ASSET	greater the VaR	1 day 95% VaR for $1000 of TWTR shares is $41

VAR REPORTING CHALLENGES

VaR is used in many contexts, with many different VaR users in a typical enterprise, each with varied reporting needs, as this summary of a typical set of VaR users illustrates.

VAR VIEWS	USE
VaR by counterparty	Margining
VaR by trader	Trader P&L
VaR by desk	Desk head
VaR by legal entity	Finance/regulator
Contributory VaR/VaR drivers	All/what drove VaR?

It is important to note that VaR is not simply a linear measure. For instance, the VaR of a portfolio containing assets A and B does not equal the sum of the VaR of asset A plus the VaR of asset B, as seen in the example below.

VaR ($1000 GOOG + $1000 TWTR) != VaR($1000 GOOG) + VaR($1000 TWTR)

Therefore, SQL and traditional warehouses have limited utility when the information that is being reported cannot be aggregated in a linear way. As well as VaR, many other important risk measures such as counterparty credit risk fall into this category. Typically, risk reporting warehouses pre-aggregate VaR by all frequently used dimensions and use the pre-aggregated values to report VaR. This helps to an extent, but views are limited and fixed. To calculate VaR along any other dimension, or for a custom set of assets, users have to run a new VaR aggregation job on the analytics engine and wait for the results to load in the data warehouse again.

LIMITATIONS OF TRADITIONAL DATA WAREHOUSES

While traditional data warehouses can be great for reporting simple information that can be aggregated in a linear way, they have their limitations:

- Reports are shallow schemas with limited analytical capabilities
- Reporting is based on standard slice and dice operations and simple aggregation functions
- Limited support is available for non-linear or semi-structured data such as vectors, matrices, maps and nested structures
- Schemas are fixed, so new analytics and aggregations require new views and schema changes
- There is limited support to run user-defined functions or to call out to external analytical libraries, leading to a limited set of analytics being pre-aggregated.

THE DAWN OF HIGH-DEFINITION **DATA WAREHOUSING**

Big data technologies such as Spark SQL, Impala and Hive can be combined with serialisation formats such as Avro, Thrift, Protocol Buffers, Hive and HDFS, to build a smart, high-definition, adaptive, high-performance data warehouse.

- **Smart** thanks to embedded custom analytics for aggregation and reporting that use Spark SQL and Hive user defined functions (UDFs). UDFs allow implementation of a domain specific language (DSL) on the data warehouse by extending SQL. UDFs can also call up external analytical libraries
- **High definition** due to the ability of persistence formats such as Avro, Thrift, Protocol Buffers and Hive Serdes to model complex domain objects via rich type support
- **Adaptive** via the ability of Avro, Thrift, Protocol Buffers and Hive Serdes to support evolvable schemas
- **High performance** through Spark's capability for fast, large-scale data processing using an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

With this approach, you can ask any question or run any reports with no need to request a custom analytical job, giving you easy access to deeper insights on demand. What's more, Spark SQL allows you to store a high-definition view of your data in the data warehouse. So instead of calculating a single VaR number based on limited assumptions, you can store a complete view of the trade in the warehouse, using a Hive Array data type, including all historic PNL.

FOCUS ON USER-FRIENDLY FEATURES

In addition to the benefits outlined above, the high-definition data warehouse approach offers a wide range of features designed to optimise efficiency and ease of use:

- The warehouse DSL is built on top of SQL, the most popular data analysis language
- Spark SQL's industry standard JDBC and ODBC interfaces enable use of standard visualisation tools like Tableau, ZoomDate, Qlik, Microstrategy
- Hundreds of standard Hive and community-contributed UDFs from the likes of Facebook and Brickhouse can be used out of the box and combined with custom UDFs
- Spark UDFs are concise, quick to implement and can be unit-tested
- Spark UDFs offer 'write-once, use-everywhere' versatility for streaming jobs, batching jobs, REST services, ad hoc queries and machine learning
- Polyglot persistence ensures that Spark SQL supports a variety of data sources like HBase, Cassandra and relational databases. This allows for joins across data sources, so positions can come from HDFS, time series from HBase or Cassandra and business hierarchies and reference data from a relational database.

FLEXIBLE, FUTURE PROOF AND READY FOR CHANGE

The Basel Committee on Banking Supervision is currently carrying out a fundamental review of the trading book capital requirements (FRTB). A number of weaknesses have been identified with using VaR for determining regulatory capital requirements, including its inability to capture tail risk.

Consequently, a move from VaR to Expected Shortfall (ES) is being proposed. It will be the biggest and most significant market risk regulatory change since the introduction of VaR almost two decades ago and will necessitate significant changes in the ways that banks manage data.

A high-definition warehouse as described above would allow you to calculate the new metrics required without changing the warehouse. Also known as Conditional Value at Risk (CVaR), ES is more sensitive to the shape of the loss distribution in the tail, which means you can make calculations on the fly by leveraging SparkSQL.

BENEFITS BEYOND COMPLIANCE

Generally, when comparing the big data stack to traditional warehouses, vendors have focused on the cost savings. But in addition to reducing the burden of regulatory reporting, the new generation big data technologies offer the potential to build high-definition data warehouses that give your users self-service access to faster, deeper insights and deliver significant competitive advantage.

A high-definition view of data maximises your ability to obtain new insights. The ability to arbitrarily mine a dataset creates opportunities for business optimisation and new applications. Custom analytics made available via user-defined functions make the warehouse smarter, and transforms SQL into a DSL form that can be viewed as a Spark script. Furthermore, data engineering teams could be freed up to focus on higher value functions, so saving additional time and money.

The technology stack to support this approach is available now. However, the rate of uptake in the financial services sector to date has been slow. By embracing new technologies such as Spark, you could more readily ensure compliance, while reducing development costs and saving on reporting resources that could be reinvested into the business – so helping to drive measurable commercial value.

Darren Voisey

DATA ISLANDS IN THE STREAM

Thanks to its ability to decouple systems and enable interoperability, application message broker software is prevalent in the finance sector, forming the backbone of trading platforms that distribute financial events and information worldwide. Here we take a look at how the recent beta phase arrival of Apache Kafka K-Streams is making waves and what the ripple effect might be.

WHAT IS A STREAM?

A stream is an ordered, and potentially infinite, sequence of data elements made available over time. Examples include website clicks and market stock price changes.

THE CURRENT SITUATION

Most developers and architects will be familiar with application messaging, especially the vanilla functionality of durable[1] topics, point-to-point channels[2] and competing consumers[3]. Many will also understand the extended routing and complex logic that can be constructed via the advanced message queuing protocol (AMQP) implementation supported by ActiveMQ and RabbitMQ.

While these technologies can be used to build asynchronous systems that can scale horizontally, they also have limitations (architecture compromises) that need to be considered. Notably:

- Each topic queue has to fit on a single node within a cluster
- Message ordering is not guaranteed
- Messages are delivered at least once, so duplicates can occur.

Most applications need to allow for these limitations. For example, if an application keeps a running total of trades per minute, it will also have to maintain a separate state to handle failure and restart, because all read messages will have been removed from the queue.

This technology stack is typically seen in event-based remote procedure call (RPC) types of systems, but the limitations listed above do muddy the waters of what we would otherwise consider a perfect stream. However, if you don't need the complex routing, there is a viable alternative.

ENTER APACHE KAFKA

Financial institutions are increasingly adopting Apache Kafka to integrate various big data platform components and using it as a de facto messaging standard. We are aware of at least one bank where uptake has reached such a level of maturity that it is now offered as a shared service. However, unlike RabbitMQ and ActiveMQ, Kafka supports only basic messaging patterns. So the question now is whether Apache Kafka has the potential to do more and if so, what next?

With Kafka often described as a 'distributed log', key differences from the other messaging technologies referenced here are as follows.

- **Scalability** – topics can be sharded or partitioned according to custom rules that permit data to be distributed across nodes in a cluster, enabling horizontal scaling at topic level.
- **Durability** – instead of removing messages once read, removal is based either on age or a custom compaction rule; so if your compaction rule is 'keep latest', your queue can be effectively a table.
- **Accuracy** – with message order guaranteed within each shard, you can model a stream more accurately.
- **Routing** – bear in mind, however, that the framework provides constrained routing logic, with no AMQP-type routing.

What makes queues with history so exciting is how much easier it becomes for applications to restart and initialise themselves with no need for separate journaling. Distributed queues can hold a lot of state and they then act as storage: if we then shard according to business needs, our ordering can be preserved. In other words, we have ordered streams of data that scale.

EVOLVING DESIGN

The ability of Kafka queues to operate as a persistence layer opens up interesting new approaches to system design. You can, for example, rely on Kafka to act as a journal and, through custom compaction, reduce the amount of processing required when initialising data from historical records.

However, what about creating projections on top of queues, in effect deriving new streams of data from underlying queues? This is where Kafka K-Streams currently in pre-release, fits in well. It provides a Java library for building distributed stream processing apps using Apache Kafka. Your application components can therefore derive and publish new streams of data using one or more Kafka queues for the source and output. The durability – or persistence – of the queue data then underpins the components' resilience and fault tolerance.

KAFKA K-STREAMS IN PRACTICE

The following example consumes a stream of stock prices and derives the maximum price per stock per 500ms. Unlike some other stream processing frameworks, Kafka K-Streams doesn't use micro-batching, so you can achieve processing in near real time.

The code below performs the following steps:

- Subscribes to a topic of stock prices, together with the classes, to de-serialise the data
- Creates a new key value pair based on the stock code within the message
- Aggregates by the stock code – in this example we keep a running max
- Windows the results every 500ms
- Writes the results to a new topic called 'streams-output'.

This code is based on the early preview of Kafka K-Streams, with more comprehensive documentation provided by its developers at Confluent at http://docs.confluent.io/2.1.0-alpha1/streams/index.html

```
KStream<String, StockTick> source = builder.stream(stringDeserializer,
payloadDeserializer, sourceTopic);
KTable<Windowed<String>, Long> ktable = source
            map((s, payload) -> {
               //generate a key based on the stock
               return new KeyValue<String, StockTick> (payload.stockCode, payload);
            })
            .aggregateByKey (() -> 0L
                   (aggKey, payload, aggregate) -> {
                      //the aggregation tool
                      return Math.max(aggregate, payload.price);
                   },TumblingWindows.of("stock-maxprices").with (500)
                   ,stringSerializer, longSerializer, stringDeserializer, longDeserializer
            );
ktable.to("streams-output", winowedStringSerializer, longSerializer);
```

OUR ASSESSMENT

The examples and tests included within the Kafka source code proved invaluable during our research for this article. We were impressed at how succinct and easy to understand the API was to use. In fact, our main frustration was that RocksDB, the default local state store used by Kafka K-Streams, isn't supported on MS Windows, our work development environment.

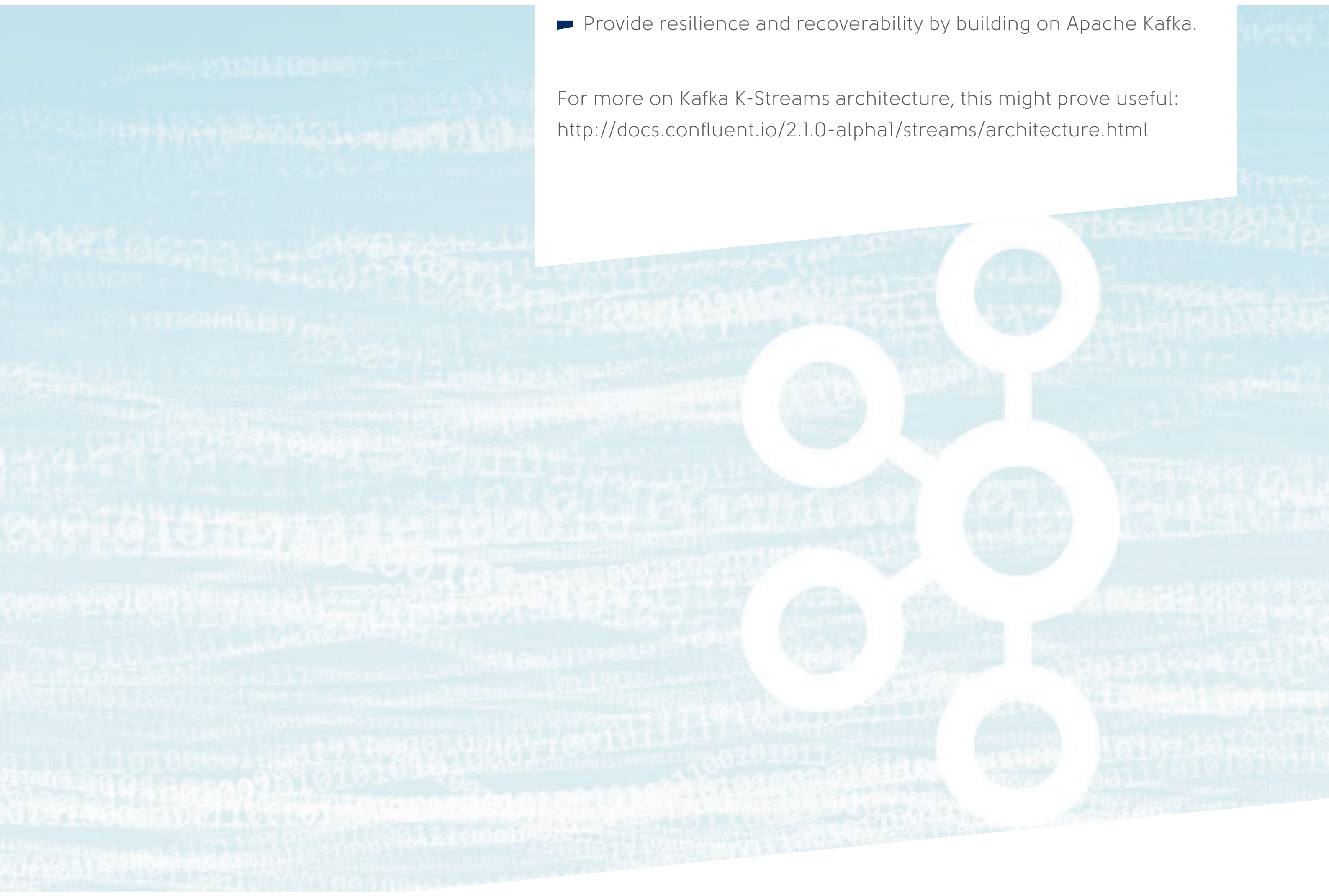
SHOULD YOU GO WITH THE FLOW?

So what are the likely benefits if you decide to go with Kafka K-Streams? If your organisation is already using Kafka, then it's worth starting to look at it as a way to reduce your application logic, especially as it doesn't need additional infrastructure. And, if you are performing this type of logic with AMQP, Kafka and Kafka K-Streams could simplify your architecture and provide better scale and resilience.

To sum up, therefore, Kafka K-Streams enables you to simplify your application messaging and create new data islands in the stream as follows:

- Derive new streams of data by joining, aggregating and filtering existing topics
- Reduce your application logic by using the Kafka K-Streams DSL
- Provide resilience and recoverability by building on Apache Kafka.

For more on Kafka K-Streams architecture, this might prove useful: http://docs.confluent.io/2.1.0-alpha1/streams/architecture.html



1 http://www.enterpriseintegrationpatterns.com/patterns/messaging/DurableSubscription.html
2 http://www.enterpriseintegrationpatterns.com/patterns/messaging/PointToPointChannel.html
3 http://www.enterpriseintegrationpatterns.com/patterns/messaging/CompetingConsumers.html

Neil Avery

BATCH, STREAM AND DATAFLOW

WHAT NEXT FOR RISK ANALYTICS?

In recent times banks have relied on compute grids to run risk analytics. However, this long-established batch paradigm is being challenged by streaming, which is on the cusp of replacing compute grids as we know them. Here we explore the evolution of streaming and evaluate some of the leading approaches, with a particular focus on Google's Dataflow.

SITUATION SNAPSHOT

Historically, risk analytics using a compute grid comprised of a job referencing trade IDs and a set of measures (greeks) sent to each compute core: there could be several millions of these tasks. Compute grids typically range in size from 500 compute cores; many have 10,000; and some up to 80,000 compute cores. Most systems process a business line's analytics as a batch for end-of-day processing, which might take 12 hours to complete. Some systems perform bank-wide aggregations to calculate exposure or analyse other factors that span all business lines. These systems represent a large portion of data centre estate and are classically bound to the server/task compute model.

In recent years many banks have invested in data lake platforms, which has triggered a shift from traditional MapReduce jobs to real-time streaming. Providing that it can scale and support sufficiently rich semantics, the streaming approach brings many benefits. Currently, we see Spark Streaming as the industry standard, and Kafka, a horizontally scalable messaging platform, is generally used as the datapipe that feeds into Spark Streaming for the execution phase. The streaming model has further evolved with the 2015 launch of Google's Dataflow, which presents an opportunity for fast and agile data processing while also replacing compute grids with streaming, as we will explain.

PHASE I
LIMITATIONS OF COMPUTE-BOUND RISK ENGINES

End-of-day batch jobs on a compute grid will submit millions of tasks; execution will involve long running analytics with multiple workflows, forking, joining and handling transient data for reuse in the series of domain/analytic derived workflow patterns. They are a fundamental operational requirement for calculating all kinds of insights about trade positions, risk, etc. Increasingly they are used for regulatory reporting requirements or more generally for calculating close-of-business prices/positioning/reports and setting up start-of-day processes. However, as well as taking hours to complete, they are not real-time. They are part of the T-3 process. Instrument and risk complexity often drives the quantity and duration of individual tasks comprising the overall job.

You could think of it like this:

Trade -> for each risk measure/greek –> Curve data -> for each Tenor-blip -> calculate.

Combine this with credit valuation adjustment (CVA) or C-CAR and we move up the complexity curve.

Depending on the business sector and instrument, the complexity required to drive ticking prices, real-time streams and ad hoc analysis make batch orientation unsuitable or even impossible. The process is too slow and cumbersome. Risk engines can sometimes be written to execute periodic micro-batches, however, they don't operate in real-time and only execute every five minutes.

PHASE II
HOW LAMBDA LINKS COMPUTE-BOUND AND REAL-TIME

In 2014 the Lambda architecture was introduced. It works on the idea that combining an approximation of streaming results with pre-generated batch results makes it possible to create a real-time view. Unfortunately, most streaming algorithms are based on sketching semantics which provide approximation rather than absolute correctness – consider, for example, Top-N, HyperLogLog and others[1]. As a result of this approximation, Lambda can't deliver sufficiently accurate results (batch parity). By the time streamed data has been batch-(re)processed, the (re)derived output will have changed, creating a system of eventual consistency[2]. The alternative is to retrigger a partial evaluation, but although this may serve a purpose, there is also scope to lose the real-time elements for which the Lambda approach was introduced.

PHASE III
DATA LAKES, LAMBDA AND THE DUALITY DILEMMA

The other major problem with the Lambda architecture is that you have to maintain two systems, adding more complexity to an already complicated situation. Data lakes can be viewed as part of a Lambda system: they represent a duality, but can also serve tasks similar to the compute analytics grid. Data lakes generally use Spark Streaming to consume, enrich and process data in real-time. However, these systems offer many of the business benefits associated with data agility and speed, for example, enabling traders and analysts to execute 'what-if' scenario analysis using an interactive notebook.

PHASE IV
STREAMING GOES SOLO WITH KAPPA

Kappa throws the Lambda two-system notion out of the door – duality has too many challenges. Kappa's approach is simplistic, in that the idea is to store all the source data within Kafka, then when you need a new version of the data, you replay the stream. Parallelism is required to scale out, but the need to deal with complex workflows may prove too much and make this approach impossible. While Kafka's Hadoop integration provides powerful leverage over HDFS and Hive, etc, the problem with the Kappa approach is the complexity of the analytics and their execution speed, for example, some analytics can take 10 hours to complete. This is a point we'll come back to.

THE FIRST WAVE OF STREAMING PLATFORMS

In the first wave of streaming platforms, Spark Streaming, Storm, Samza, Flink and Kafka K-Streams have all been dominant. All have contributed to our current value proposition of streaming use cases. However, the typical workload is only to stream a single parallel set of tasks. Storm and Samza support more elaborate workflows but they don't enable rich coordination in data-driven functional style. A classic workload would be massively parallel streaming analytics that process large volumes of data such as click-streams or ticking prices.

GOOGLE DATAFLOW: THE FUTURE OF STREAMING

The ground-breaking publication of Google's Dataflow model in 2015[3] established the first complete working framework for streaming. Leveraging its experience with MillWheel and FlumeJava, Google provided a comprehensive working model to execute streaming in its Google cloud. A batch is unified within the model and executed using a stream. In terms of resource management, the Dataflow model is geared towards hands-free DevOps. Because it runs in the cloud, Dataflow also makes it easy to optimise efficiency and control costs without the need to worry about resource tuning. For instance, the number of processing nodes you need is provisioned elastically to match your precise requirements, so you don't need to worry about tuning the number of processing nodes. What's more, Dataflow's programming model is functionally biased when compared to the topological or classic MapReduce model. This allows you to shape execution paths and workflow according to the underlying data at execution time.

Nonetheless, there remains a point of contention: is it more expensive to ship data or to scale compute? For example: a Monte Carlo simulation requires you to scale compute – in other words to process thousands of calculations on the same data with differing parameters. Conversely, building a curve model that can be used for subsequent calculations requires shipping yield, credit curves and volatility surfaces for a complex instrument to a single node.

CAN YOU ADAPT A COMPUTE GRID JOB TO RUN AS A STREAM?

In many financial services use cases, gathering the data required for analytics can be expensive. However, organising data into channels for scale-out via fast 10G Ethernet combined with scalable workflows can potentially overcome repeated chattiness.

Key features of the white paper[4] include a complete set of core principles, including:

- Windowing – various models to support unaligned event-time windows
- Triggering – reaction and response to data triggers
- Incremental processing – to support retractions and updates to the window.

It should also be noted that although the framework provides a model that enables a simple expression of parallel computation in a way that is independent of the underlying execution engine, tasks will remain computationally expensive and may take several hours to evaluate a risk measure.

We should also clarify that when we refer to a batch, we simply mean a stream of data with a known start and end – it is bounded data. Streaming without a start point or end point is called unbounded data.

DATAFLOW IMPLEMENTATION: BATCH TO STREAM

Dataflow is a native product on the Google cloud platform, which has evolved to a point where you can build almost anything using three key services:

- Dataflow DF (routing and processing)
- BigTable BT (NoSQL storage)
- BigQuery BQ (query)

To convert an existing compute batch workflow to Dataflow you will need to build a Dataflow pipeline model as follows:

BigQuery-input -> transform -> PCollection (BigQuery rows) -> Transform (ParDo) -> output

The key steps involved are to:

- Source data from BigQuery and/or cloud storage
- Iterate the PCollection for each item and apply one or more transformations (ParDo) and apply an analytic or enrichment
- Merge PCollections together to flatten or join (CoGroupByKey) data.

The above scenario gives you a classic source -> fork -> join pattern. However, Dataflow is so powerful that you can model any kind of pipeline/workflow in code, no matter how elaborate, as illustrated on the diagrams below.

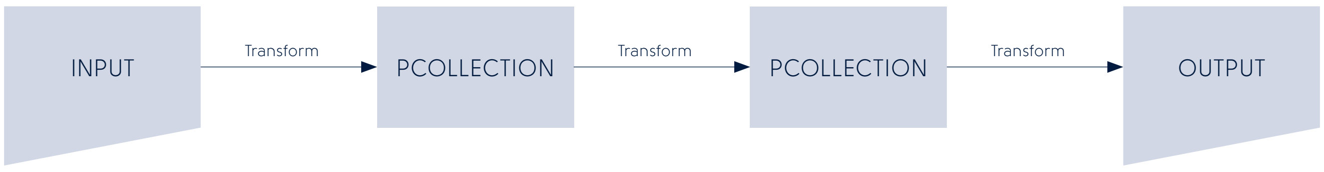


Figure 1. A sequential pipeline showing Dataflow components in relation to each other.

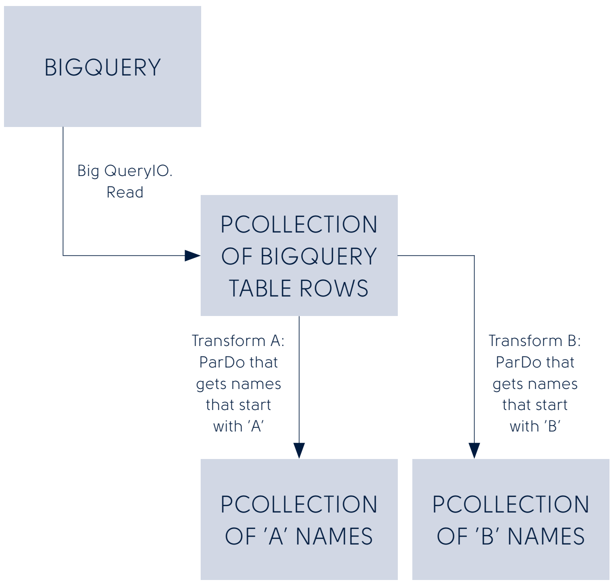

BIGQUERY
Big QueryIO. Read
PCOLLECTION OF BIGQUERY TABLE ROWS
Transform A: ParDo that gets names that start with 'A'
Transform B: ParDo that gets names that start with 'B'
PCOLLECTION OF 'A' NAMES
PCOLLECTION OF 'B' NAMES

Figure 2. Forking the stream.

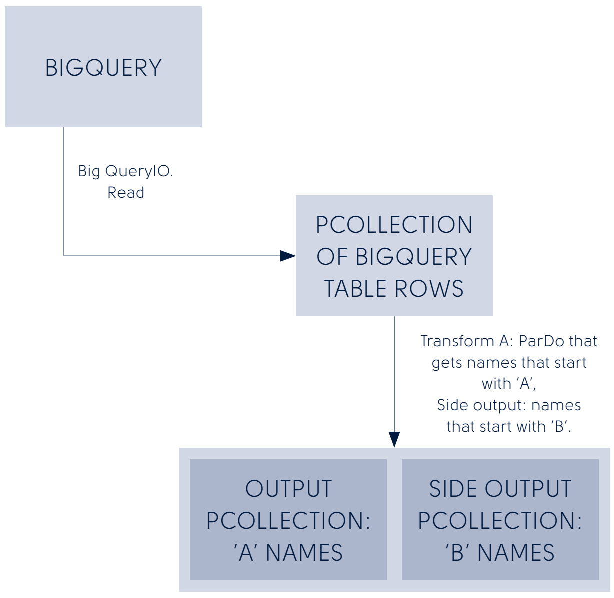

BIGQUERY
Big QueryIO. Read
PCOLLECTION OF BIGQUERY TABLE ROWS
Transform A: ParDo that gets names that start with 'A', Side output: names that start with 'B'.
OUTPUT PCOLLECTION: 'A' NAMES
SIDE OUTPUT PCOLLECTION: 'B' NAMES

Figure 3. Joining data from different streams.

Once the pipeline is built into an artefact (including the QuantLib), it is deployed to the cloud using Dataflow managed services to convert the pipeline into an execution graph.

THE DATAFLOW MODEL IN CODE

The Dataflow model provides windowing as part of the formal notation. As a result, comparing the two side by side shows Dataflow's clean readability versus Spark's somewhat complex code block.

In the code section below, blue represents where the data is being processed (windowing), while yellow represents what is being processed.

On the other hand, because the Spark model lacks a formal notion of event-time windowing, we have to intermingle the what and where portions of the code[5].

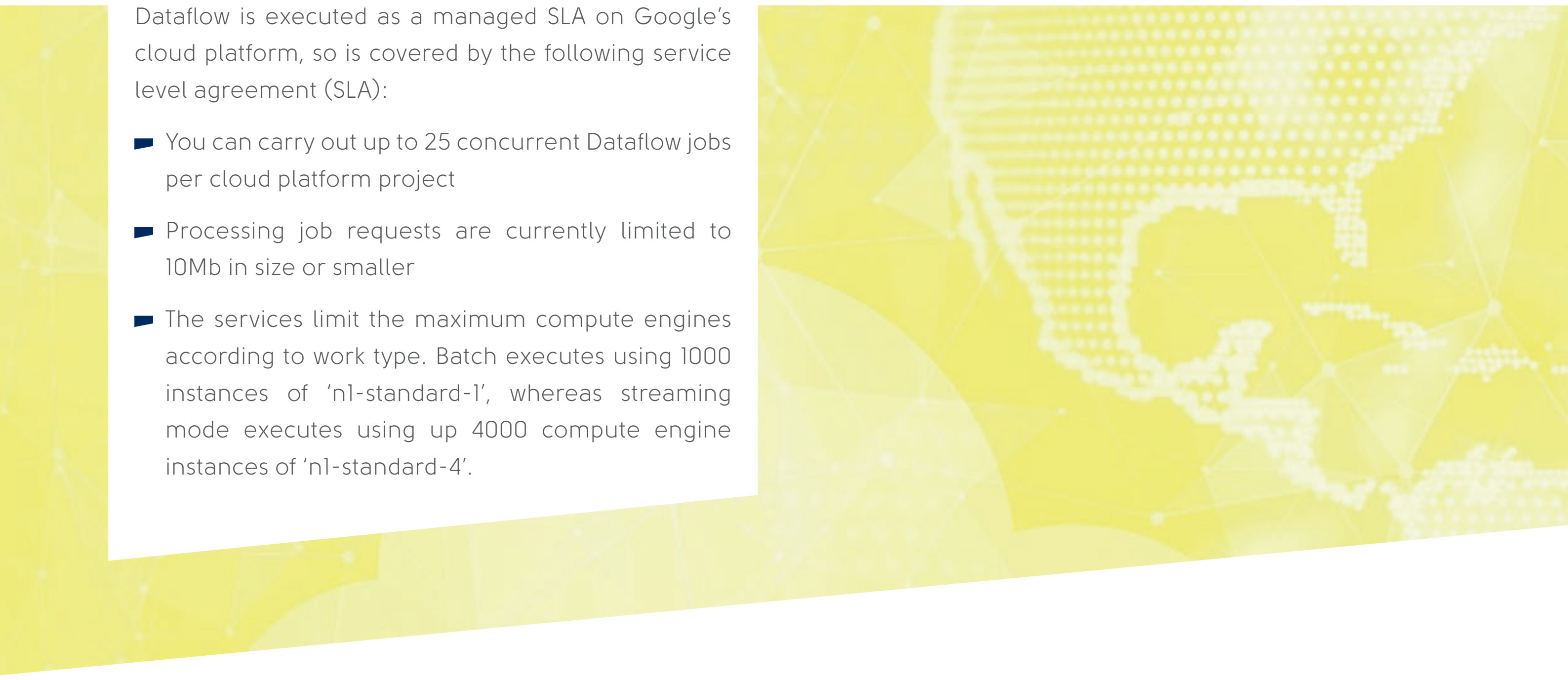
Dataflow

```
gameEvents
  [... input ...]
  [... filter ...]
  .apply(„AddEventTimestamps",
WithTimestamps.of((GameActionInfo i)
    -> new Instant(i.getTimestamp())))
  .apply(„FixedWindowsTeam",
Window.<GameActionInfo>into(

FixedWindows.of(Duration.standardMinu-
tes(windowDuration))))
  .apply(„ExtractTeamScore", new
ExtractAndSumScore(„team"))
  [... output ...];
```

Spark

```
gameEvents
  [... input ...]
  [... filter ...]
  .mapToPair(event -> new
Tuple2<WithTimestamp<String>, Integer>(

WithTimestamp.create(event.getTeam(),
      (event.getTimestamp() /
windowDuration) *
windowDuration),
    event.getScore()))
  .reduceByKey(new SumScore());
  [... output ...];
```

CLOUD EXECUTION SERVICE LEVEL AGREEMENT



Dataflow is executed as a managed SLA on Google's cloud platform, so is covered by the following service level agreement (SLA):

- You can carry out up to 25 concurrent Dataflow jobs per cloud platform project
- Processing job requests are currently limited to 10Mb in size or smaller
- The services limit the maximum compute engines according to work type. Batch executes using 1000 instances of 'n1-standard-1', whereas streaming mode executes using up 4000 compute engine instances of 'n1-standard-4'.

APACHE BEAM: DATAFLOW ON-PREMISE

Google open-sourced Apache Beam as its contribution to the community to provide an on-premise Dataflow alternative. It is essentially the Dataflow API with runners implemented using Flink, Spark and Google cloud Dataflow. More interestingly, it is Google's first open-source contribution of this kind – by contrast, previous innovations like HDFS used the more hands--off 'throw a white paper over the wall' approach on the basis that someone else would pick up the idea and run with it.

Apache Beam enables you to operate the same expressive runtime-defined stream processing model. However, you are limited to on-premise, statically sized deployments commandeered by Zookeeper. You therefore won't have access to the compute resources available when leveraging cloud elasticity.

The vision for Apache Beam is to support a unified programming model for:

- End users who want to write pipelines in a familiar language
- SDK writers who want to make Beam concepts available in new languages
- Runner writers who have a distributed processing environment and want to support Beam pipelines.

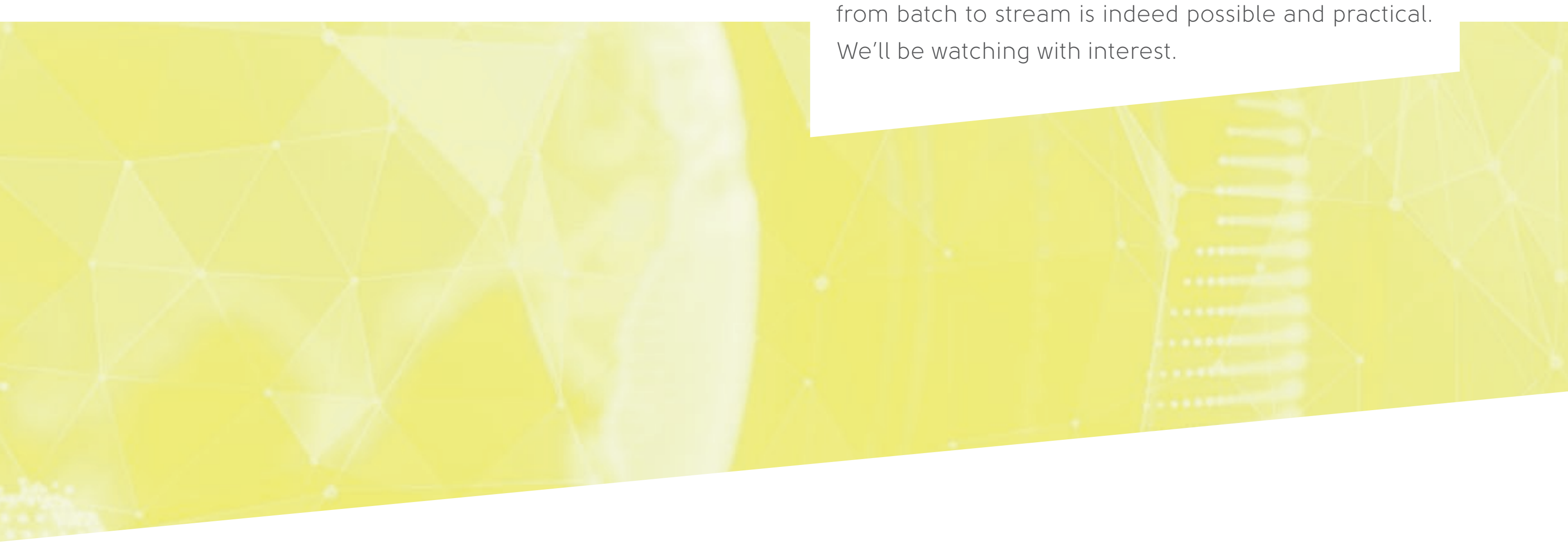
FUTURE RESEARCH

We are planning to convert an open source batch--oriented workload to Dataflow. Part of the exercise will be to identify performance challenges seen in this space, for example, relating to data locality and intermediate/transient data, and to establish performance metrics.

WHAT WILL DATAFLOW DELIVER AND WHEN?

As we have seen, the dominance of compute grids in financial services is being challenged by a major transition to streaming. Yet again, the leading innovators are providing direction on how we will move from a compute or server-oriented paradigm to one that is stream-oriented and server-free. While Lambda and Kappa have produced solutions to the challenge of providing real-time information, the inherent duality and constraints in their streaming models limit practical application.

Only now, with the advent of the Dataflow model, are we seeing a viable streaming solution with the potential to replace the largest financial services workloads and move them to a new paradigm built around real-time analytics, where batch is a subset. The challenge now is to prove to the incumbents that this migration path from batch to stream is indeed possible and practical. We'll be watching with interest.



1 https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
2 https://www.oreilly.com/ideas/questioning-the-lambda-architecture
3 http://www.vldb.org/pvldb/vol8/p1792-Akidau.pdf
4 http://www.vldb.org/pvldb/vol8/p1792-Akidau.pdf
5 https://cloud.google.com/dataflow/blog/dataflow-beam-and-spark-comparison#hourly-team-scores-windowing

James Bowkett
Jamie Drummond

MINIMISE DATA GRIDLOCKS WITH MACHE

In recent months we have been adding features to one of our main use cases for Mache – our open source cache solution for taming bottlenecks in grid applications. Mache is opening up exciting opportunities to dramatically improve network efficiency, clearing the way to more valuable big data applications.

First, let's briefly recap on why we developed Mache. Financial services – particularly investment banks – have always relied heavily on large-scale compute grids. But recently, organisations have been switching from traditional in-memory data grids to NoSQL products with the scalability to process vast volumes of data quickly.

Mache combines a scale-out NoSQL layer with a near cache to create a scalable platform that can be relied on to handle the most intensive data applications. In other words, Mache decouples applications from messaging and storage. And, with the addition of a new REST API, Mache makes storage perform less like a native database and more like a service.

By separating the data layer, the local cache allows data sharing between processes. This in turn enables applications to adopt a more micro-services style architecture. Consequently, by reducing database contention from competing applications, decoupling enables you to service more requests, thus reducing network load and congestion. However, it leaves the storage layer free for tuning by your application suite as required.

Among the key benefits we see for Mache are its wide range of integrations. These include:

- **Messaging and invalidation** – it integrates equally well with Kafka, ActiveMQ/JMS and RabbitMQ
- **Big data and storage** – it has been tested with Cassandra, Mongo DB and Couchbase
- **Client platforms** – a new REST interface allows Mache to serve value objects via JSON to virtually any client application stack.

So let's take a look at the issues around network latency and saturation in a little more detail.

NETWORK LATENCY

With a conventional grid architecture, the number of requests can overwhelm the network and increase latency.

Latency can be difficult to comprehend because network workload and complexity can vary greatly. It is well understood that as the network load increases, the variation of data can also increase greatly and, as a result, the mean response time will also rise.

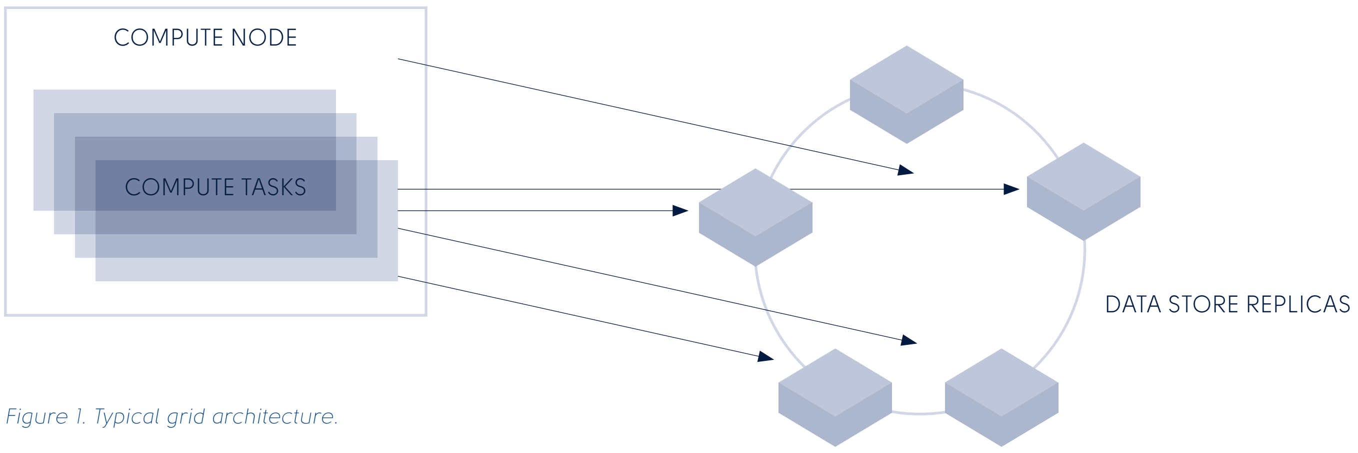


Figure 1. Typical grid architecture.

Figure 2 below shows the impact on cache latency of increasing transactions in a local test environment. This data, often seen under stress test conditions, will not surprise anyone who has measured loads in larger networks.

Common data access patterns have a large number of nodes. Therefore, by reducing the number of network requests by a factor of the number of tasks per node, caching can significantly reduce latency.

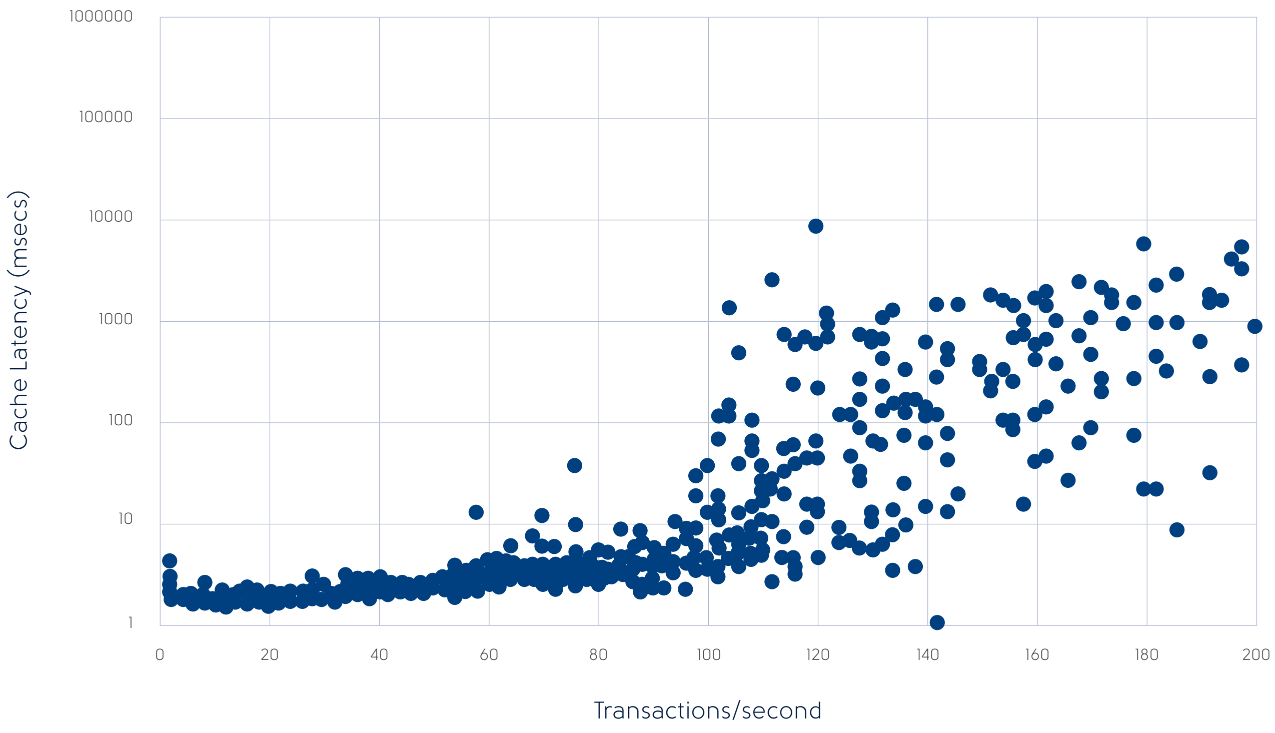
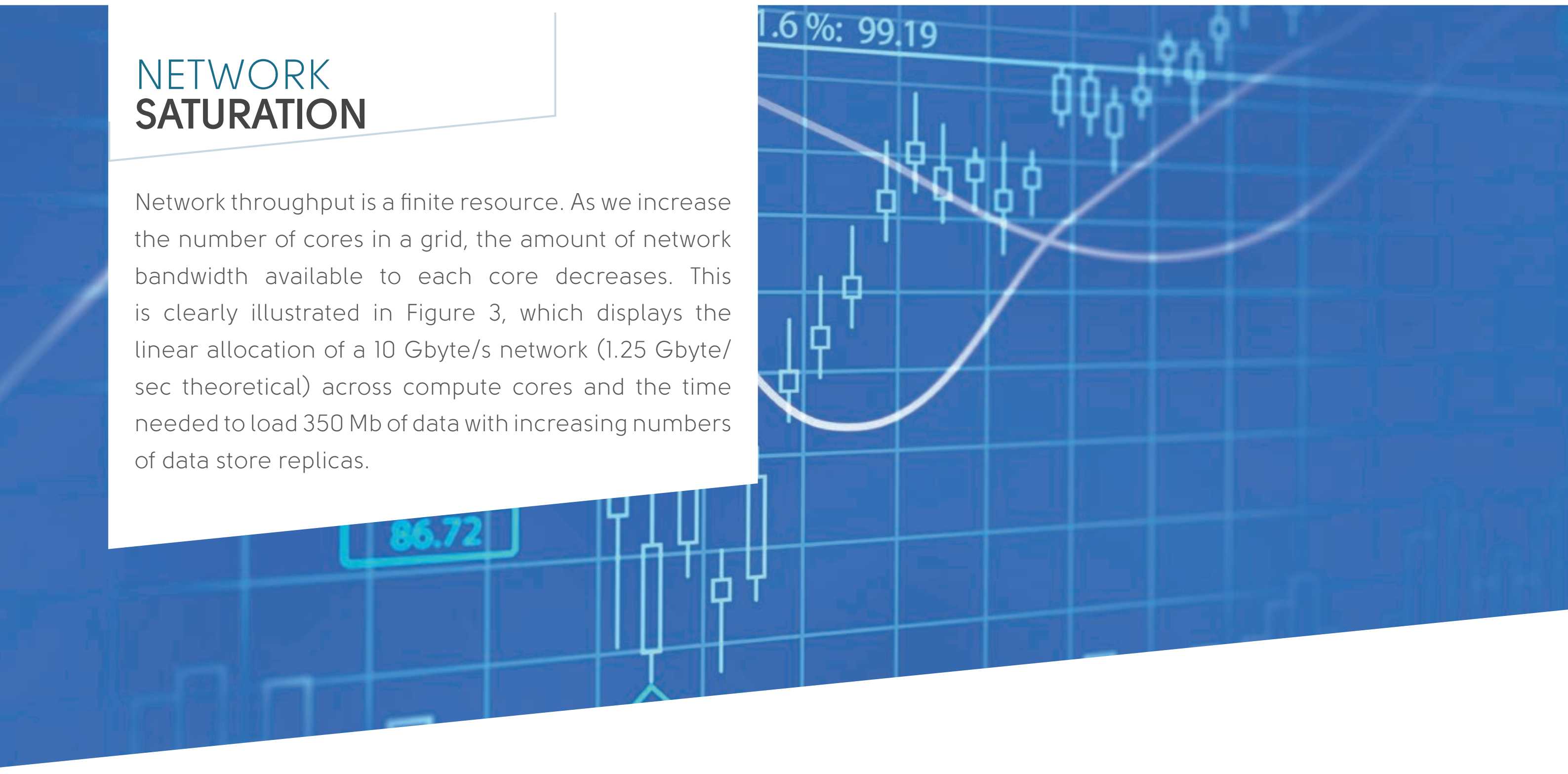


Figure 2. Cache latency and invalidation performance rises as the number of connections grow.



NETWORK SATURATION

Network throughput is a finite resource. As we increase the number of cores in a grid, the amount of network bandwidth available to each core decreases. This is clearly illustrated in Figure 3, which displays the linear allocation of a 10 Gbyte/s network (1.25 Gbyte/sec theoretical) across compute cores and the time needed to load 350 Mb of data with increasing numbers of data store replicas.

DATA 10GIGE WITH 10,000 COMPUTE NODES AND A 350 MB PAYLOAD

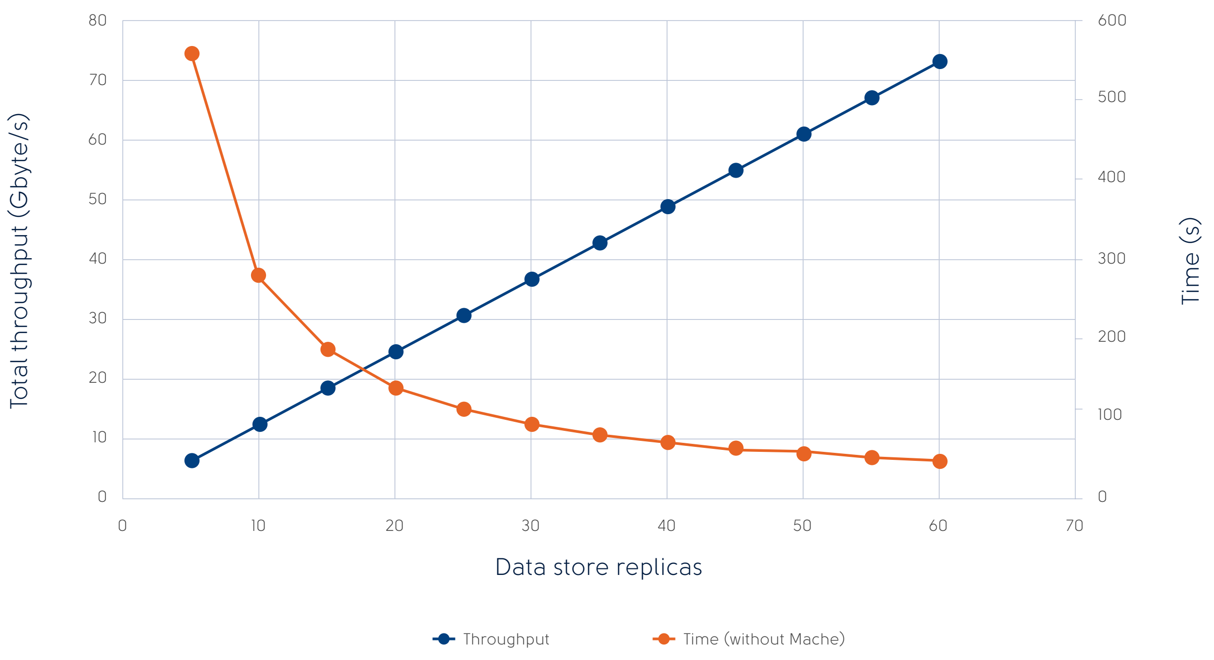


Figure 3. Data store replica shows that as data throughput rises, network efficiency falls.

HOW MACHE IMPROVES EFFICIENCY

By reducing contention for bandwidth, Mache delivers multiple benefits.

- **Reduce data duplication:** tasks within a grid node can query the local Mache service in any access pattern required, helping to avoid database hot spots and the repeated need to send duplicate data across an overstretched network.
- **Enable data sharing:** multiple processes within the same grid node can share the same data, negating the need to send the same data across the network more than once.
- **Enhance data affinity:** because Mache can stay active between multiple job executions, with careful grid partitions, nodes can process the same data throughout the batch pipeline.
- **Reduce queries:** because a larger subset of the dataset can be cached, tasks no longer use individual queries or require query optimisation.
- **Improve scheduling:** Mache provides existent caching across tasks, so demands on grid bandwidth will be less ad hoc and can be better planned, for example, according to a refresh schedule.
- **Eliminate binary bundling:** the language-agnostic REST interface decouples storage from the process, as well as from the client-side language. With many applications written in Java, .NET and more recently Go, REST also decouples code from the NoSQL vendor platform. So when it comes to deployment and upgrades, eliminating the need to bundle client-side binaries is a blessing.

A SMARTER WAY TO SCALE REQUESTS

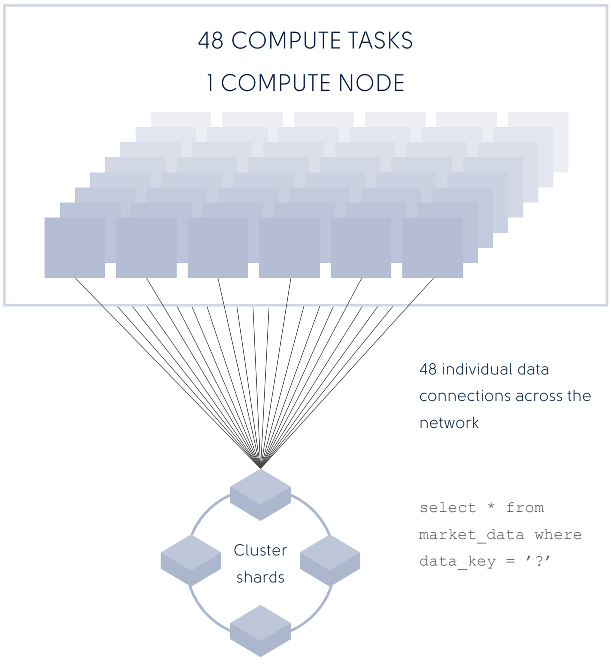
Consider a scenario where a compute grid has to process market data and calculate the PNL for a given portfolio of trades. This is typically achieved by splitting the trades into smaller groups and creating a task per group. Tasks are then queued on a compute grid to execute as quickly as possible, with the results collated later in post-processing.

Historically, each node had to connect and download the required data from a data store. The approach can be simple to implement and the power of the compute grid is harnessed effectively. However, with ever--increasing volumes of data, task run time rises as the network soon reaches capacity. Most applications will try and cache the data locally within the requesting process.

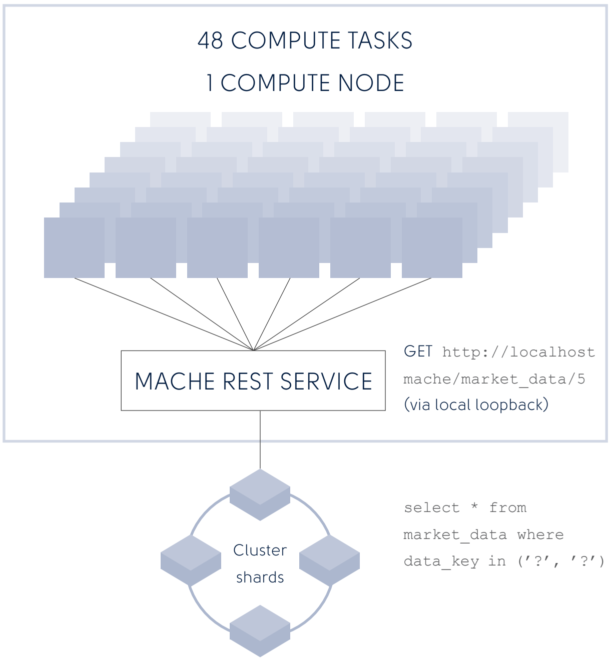
Without Mache every compute task needs direct access to the data store. This greatly increases network traffic, which reduces throughput and increases latency. Typically, many tasks are scheduled at once, further compounding the problems. For instance, if 100 tasks start concurrently and all require the latest copy of the data, the network can suddenly get saturated. Akin to a denial of service attack, the event can cause grid nodes to appear offline. The issue then escalates as the grid attempts to heal, while network infrastructure struggles to manage the load.

Even with a 10 Gbytet/s network, the maximum throughput is 1.25 Gbyte/s. With growing regulatory reporting requirements, compute grids need to scale to tens of thousands of cores, while at the same time coping with the explosive growth of market data over recent years[1]. With this in mind, it is easy to see how having all processes directly accessing the database does not scale.

In the same way that application developers try to increase performance by designing 'machine--sympathetic' applications that attempt to increase cache locality, what they actually need to design are 'network-sympathetic' architectures that can scale across thousands of compute nodes.


48 COMPUTE TASKS
1 COMPUTE NODE
48 individual data connections across the network
Cluster shards
select * from market_data where data_key = '?'

48 individual data connections across the network: No data sharing, other than accidental temporal query locality at the DB.


48 COMPUTE TASKS
1 COMPUTE NODE
MACHE REST SERVICE
GET http://localhost mache/market_data/5 (via local loopback)
Cluster shards
select * from market_data where data_key in ('?', '?')

48 individual data connections over the local loopback connection: Data shared and connections concentrated from one application.

Figure 4. Comparison of a conventional compute approach with 48 task connections to the data cluster versus a single Mache-enabled one.

MACHE IN PRACTICE

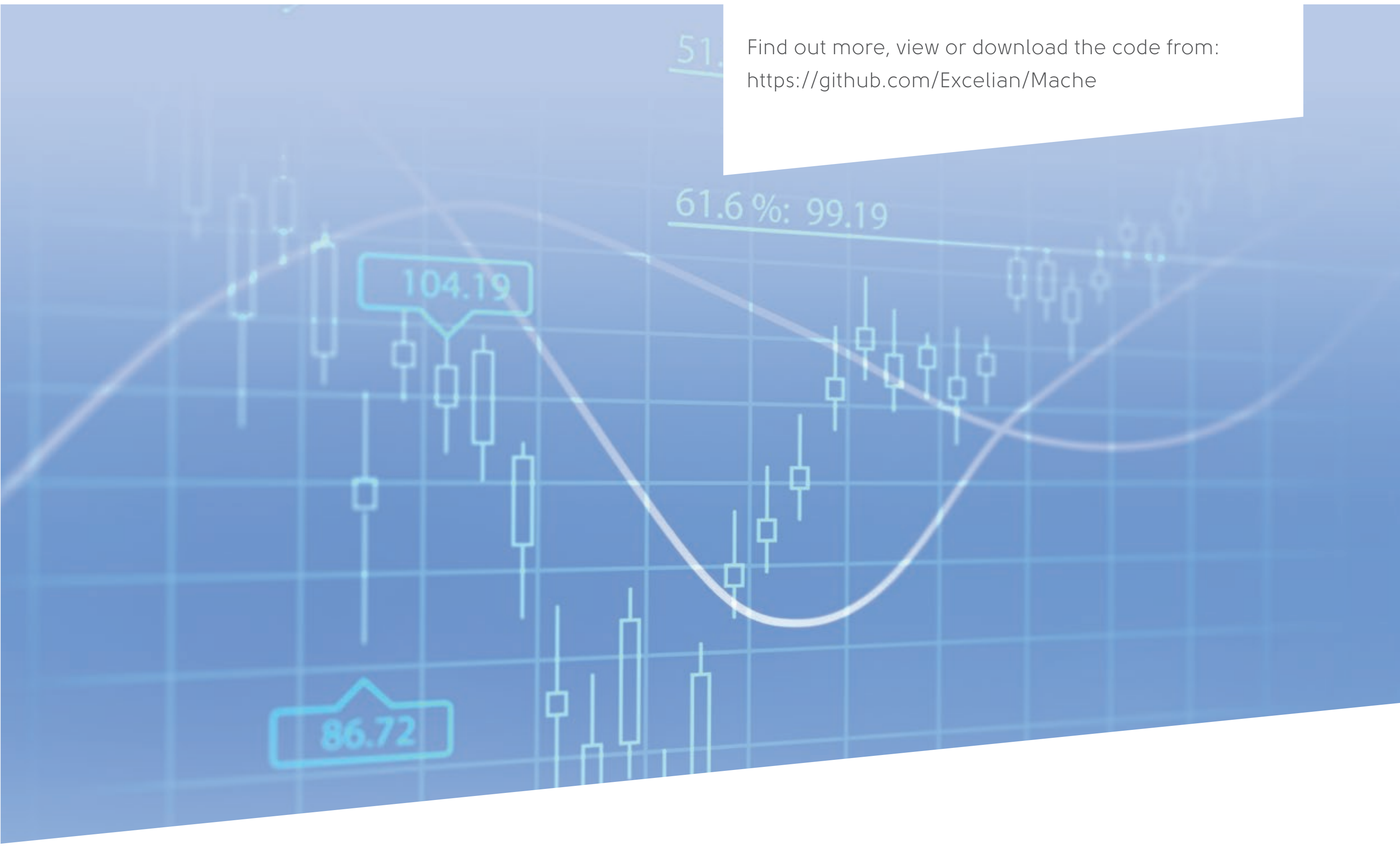
Mache can be accessed via a Java virtual machine (JVM) process or via the REST language-agnostic protocol. As can be seen in Figure 4, within a Mache model only a single database call is required versus a call for each separate task in the conventional approach. Mache retains the latest copy of the data in its memory so that it can be accessed by all tasks. Furthermore, the Mache API abstracts the connection to the data store. This means that any updates to the underlying data store are automatically synchronised, freeing up the developer to focus on task execution.

FOCUS ON THE FUTURE

To date, the project has been a great success and is generating a lot of interest from the open source software (OSS) community. It's great to see the broad appeal of many of the key Mache features for many application architectures. Notable among these are the ability to dedupe data into a localised service; binary decoupling from storage layer binaries and the potential for NoSQL and near-caching. The coming months we aim to roll out a webinar series that will include a presentation of Mache with DataStax and its roadmap.

Building on the encouraging results so far, our team here at Excelian will continue to promote Mache as a key contribution to the open source community. It neatly complements our industry expertise and will enable the adaptation of NoSQL to many use cases within the financial services sector.

Find out more, view or download the code from:
https://github.com/Excelian/Mache



1 *http://www.nanex.net/aqck2/4625.html*

BIG DATA

IN FINANCIAL SERVICES SURVEY

- How do you see big data impacting on your business?
- Are you forward thinking with your big data strategy?
- What does your technology stack look like now? What will it look like in two years' time?
- Can big data work with blockchain?
- Are you ready for streaming and machine learning?

You may know your company strategy, but are you as sure about your competitors?

We would like to invite you to take part in a completely anonymous and closed survey which will bring together multiple views and experiences of the big data landscape.

Once all surveys have been completed and analysed, we will share all learnings with those involved and provide a current status of the industry – which provides valuable insights for you to leverage in your strategy for the years to come.

Why take part?

1. Contribute to the industry knowledge share
2. Benchmark your company against your competition
3. Receive a copy of the full research findings along with your final position

If you would like to get involved, please email techspark@excelian.com and we will be in touch.