THE NEW STACK:

Kafka, Spark, and Cassandra

BY STEPHEN SWOYER

PERFORCE

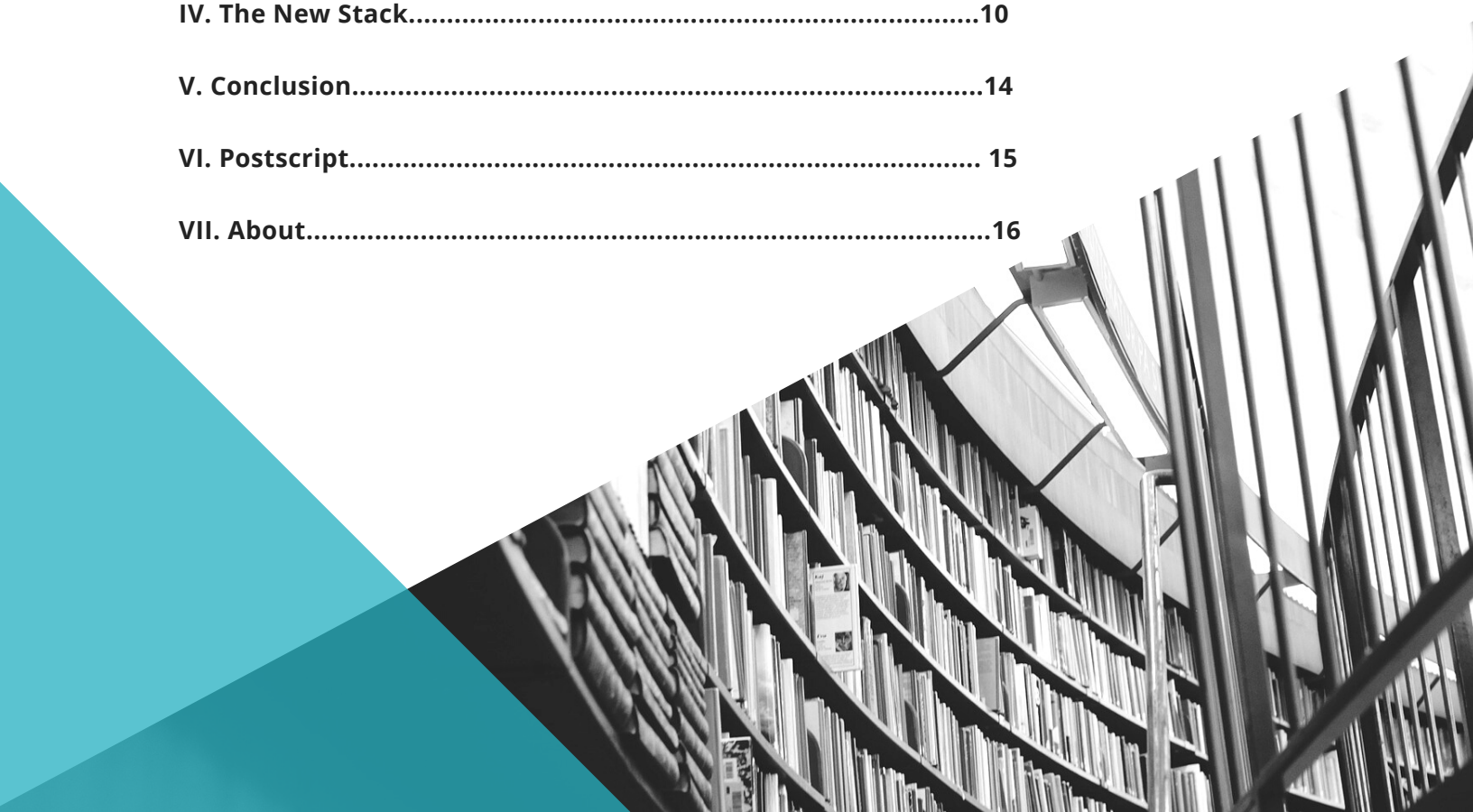
BG

Increasingly, companies of all sizes expect to be able to develop new business use cases, create new products, enter into new markets, and pursue other strategies that are dependent on access to real-time data. This is the remit of stream processing, a new data processing paradigm that gives businesses a scalable, reliable, low-latency solution for accessing and analyzing data in real-time.

This paper explores the transformative uses of stream-processing. It describes how stream processing supports a data-in-motion paradigm that differs radically from the legacy data-at-rest paradigm. It outlines the reference components of the combined open-source technology that underpins stream-processing: a **New Stack**, comprising Apache Kafka, Apache Spark, and Apache Cassandra.

In summary, the New Stack comprises a simple and cost-effective solution for stream processing that is designed to integrate non-disruptively with an organization's extant IT infrastructure.

TABLE OF CONTENTS



INTRODUCTION

Our relationship with data has changed radically in the last 20 years. Today, we generate more data than anyone could possibly have anticipated at the turn of the millennium. For most of us, background processes of data creation, data analysis, and data consumption are thoroughly interpenetrated with our day-to-day lives. Increasingly, each and every aspect of human activity is characterized by a basic (essential and unquestioned) relation to and dependence on data – or, more precisely, analytics.

This is broadly true of all human activity – and it is especially true of business. Not surprisingly, the methods and the technologies we use to create, store, process, and distribute data – the totality of which comprise the discipline called "data management" – have changed radically in the last two decades. In just the last ten years, for example, a slew of new data-dependent applications and services, user-focused practices, and business use cases has emerged to complement or displace once-dominant apps and services, practices, and use cases.

In response, data management has incorporated new data storage and processing technologies (e.g., NoSQL platforms such as Hadoop, multi-purpose compute engines such as Spark) into its toolkit. Also, data management has recognized new sources of data – among them, apps and services, along with file data of every conceivable type – as well as new means of accessing this data, including message queuing and stream-processing "buses," API-endpoints, file systems, and so on.

Transformation at this scale is not always easy to see, let alone to accommodate, however. The problem is that data management has not evolved to address new data distribution and data consumption requirements in a rational and consistent manner. Instead, it has opportunistically bolted new technologies "onto" the existing data and application integration infrastructure.

INTRODUCTION

This is analogous to a person who wants to purchase a supercharger in order to improve the performance of their car: a four-cylinder Toyota Camry. In theory, this logic is sound enough; in practice, they will need to modify their car to accommodate the supercharger – e.g., by installing a new hood (or modifying the existing hood), modifying the engine to compensate for increased wear, tuning the exhaust, changing the brakes, etc. Analogically speaking, data management did none of these things.

The predictable consequence was that organizations squandered billions of dollars on platforms such as Hadoop, deploying them alongside – and, in some cases, in place of – relational database management systems (RDBMS), data warehouses, and other established technologies. Another was that some organizations attempted to repurpose existing technologies to support applications, practices, and use cases for which they were not designed. Neither approach was ideal.

Once burned, businesses might be skeptical of streaming, another much-hyped technology paradigm.

However, streaming is not Hadoop: first, unlike the Hadoop platform, core streaming technologies – such as Apache Kafka, a "bus" (or conduit) used to transport messages and other types of streaming traffic – are highly mature: for example, the first public version of Kafka was released in 2011, while Kafka components such as Kafka Streams (2016) and Kafka Connect (2018) are also highly stable.

Second, Hadoop was marketed as an all-purpose data management and data processing platform par excellence: basically, as one platform to rule them all. This was marketing malpractice: at once wildly optimistic and disturbingly cynical. By contrast, Kafka's role is strictly delimited: it functions as a conduit for ingesting and performing operations on data in real-time, as well as for distributing data to downstream consumers. Nothing more, nothing less. Far from occupying the role of an all-in-one, all-purpose platform, Kafka is designed to be complemented by other pieces: for example, by a compute engine – such as Apache Spark – that is able to analyze data in both real-time and batch modes. And because Kafka does not provide a (scalable, cost-effective) means of data persistence, it is usually complemented by a scalable, fault-tolerant data store – i.e., a database, such as Apache Cassandra.

Collectively, these technologies comprise the core components of a combined stack for processing, analyzing, distributing, and storing streaming data: a New Stack, in the lexicon of this whitepaper. The features and capabilities provided by the New Stack will permit organizations to develop and support new types of real-time, event-driven applications; to support new user practices (especially with respect to real-time analytics); to automate business workflows and optimize business processes; and to radically transform sales and marketing, procurement and supply chain management, transportation and logistics; and – not least – to improve notionally all business function areas.

DATA MANAGEMENT

Then and Now...

Two decades ago, data management was relatively straightforward. At that time, in that context, a single general type of data – namely, tabular data – tended to predominate. Almost all of this tabular data was generated by line-of-business applications, the overwhelming majority of which lived in the on-premises data center. These applications consisted of a mix of custom-built and fit-for-purpose (i.e., use case- or vertical specific) systems, buttressed by the then-new category of enterprise resource planning (ERP) software. Then, as now, a share of business applications and data sources also lived in mainframe or minicomputer systems. Even though the overall mix of systems was highly heterogeneous, the dominant format of data – invariably tabular; increasingly relational – and the techniques by which this data was acquired and made available for access (usually via batch-based loading processes) conformed to a simple model: data got extracted from upstream business applications at predictable intervals (batch "windows"), got transformed in a middle-tier repository – usually a relational database management system (RDBMS) or an extract, transform, and load (ETL) tool – and, from there, got loaded into a destination RDBMS: in most cases, a data warehouse. This data warehouse preserved a (large or small) fraction of the data generated by business applications.

The too-long-did-not-read version of this précis is straightforward enough:

- Source data was overwhelmingly tabular in format;
- A growing share of tabular data originated in or was managed by upstream RDBMSs;
- A RDBMS, the data warehouse, was the privileged locus of data integration and
- analytics;
- Integrated data was batch-loaded into the warehouse, usually at predictable intervals;
- Real-time data integration, with the data warehouse as its locus, was prohibitively costly;
- Only a fraction of business data was loaded into the data warehouse; the rest was discarded.

Only one of these things – the preeminence of tabular data – is still true. Today, in fact, strictly relational data accounts for a steadily shrinking proportion of business data. 1 Tabular data is still preeminent, to be sure, but most of it is now generated by sources – such as connected sensors, data collectors, and telemetry signalers – that, although not unknown 20 years ago, were not as pervasive and were not treated as sources of potentially useful business data. Another big difference is that data management no longer enforces quite so hard and fast a distinction between useful and potentially useful data: rather, its preference is to preserve as much raw (or "potentially useful") data as is possible. The upshot is that a large proportion of the raw data that is generated by core on-premises business applications and middleware services, cloud apps and services, connected sensors, telemetry signalers, and other sources is now preserved. Just a fraction of this raw data ends up in the data warehouse, however. Instead, it is usually stored and managed in a central repository – a data lake – where it is made available to human and machine consumers. It is especially useful to data scientists, data engineers, business analysts, and similar expert users who prefer to work with data in its raw form, and it is essential for advanced analytical practices – including not only data mining, predictive analytics, and conventional machine learning (ML), but also advanced ML techniques like neural networks (e.g., deep learning) – which work best with raw data as input, too.

DATA MANAGEMENT

Then and Now...

What is more, the data-at-rest paradigm to which the legacy batch-centric model corresponds has been superseded by a data-in-motion paradigm. There are several reasons for this, the most obvious of which stems from the success of open-source messaging, stream processing, data analysis, and machine learning (ML) technologies. A related factor has to do with mainstream adoption and deployment of middleware messaging technologies – starting with the venerable enterprise service bus (ESB), which, more than anything else, helped expose the message traffic exchanged by applications as potential grist for analysis. The upshot is that a growing number of applications, use cases, and analytical practices now expect to analyze data as soon as it is created (in real-time) or as soon as possible after it is created (at right-time). This was not possible – or, more precisely, was not cost-effective – two decades ago. It is close to becoming table stakes for large businesses today.

STREAMING
and stream-processing explained

Streaming is several things. One, it is a new way of ingesting, managing, and distributing data. Streaming captures data feeds generated by sensors, telemetry devices, and other signalers at the enterprise edge; by (on- and off-premises) applications and services; and by core databases and file systems. Similarly, streaming distributes data to consumers of all kinds – be they machines (applications, services, etc.) or human beings (data scientists, business analysts, etc.). Simply put, streaming is a new way of distributing – in effect, of delivering – data to downstream consumers.

Two, stream processing is a means of performing automated operations on data in real-time – i.e., as it transits the streaming bus. Some operations are simple enough: a stream-processing engine might simply strip whitespace from messages, or convert upper- to lower-case text. Advanced operations might require converting from one encoding format (ASCII) to another (UTF-8), or "joining" – that is, combining – data from different columns or fields and formatting it into a new message.

Stream processing lends itself to a wide variety of use cases, starting with event-driven automation. The logic behind event-driven automation is that intervention is most valuable if it occurs as a more or less immediate response to an event. Streaming permits instantaneous – real-time – access to data feeds. Even though stream-processing does introduce some latency, this latency is typically measured as a function of milliseconds, as distinct to seconds, minutes, hours, or even days. This right-time latency permits businesses to design more tightly knit processes, as well as to automate and orchestrate time-sensitive, event-driven actions that span internal and external business processes.

The logic of event-driven processing is not new; today, in fact, a wide range of automated operations depend on event triggers to kick off. Absent real-time (or right-time) data processing, however, event-driven actions tend to trail events in the real-world, in some cases by significant periods. This is because the event data 2 that would trigger an action is not refreshed in real-time; rather, it is batch-loaded at predictable intervals. If a task is not especially time-sensitive, it could take anywhere from a few minutes, to an hour, to (notionally) even an entire day for data to refresh and for the designated action to trigger. For tasks that are especially time-sensitive, accelerated, or micro-batch, loading can be used. In practice, micro-batch can shrink the batch window from several minutes to several seconds – still not quite real-time. Even today, some organizations task human resources with the responsibility of monitoring tasks in order to intervene if delays occurred. This latency is a function of the traditional, batch-oriented data-at-rest paradigm that is still entrenched in most organizations.

Stream-processing permits businesses to automate actions in real-time in response to pre-determined events. Actions can be relatively simple: for example, if x, then y. An online merchant might instantiate an event-driven rule to the effect of if a customer applies for credit (x) then trigger an automated credit check (y). In fact, rule-based, event-driven automation of this type is common today. But what if the customer does not apply for credit? Should the business offer credit to the customer anyway? And, if so, under what circumstances? At what rate?

STREAMING
and stream-processing explained

In this scheme, the automated credit check process still takes place, but – assuming a positive result – triggers several related actions, too. From the merchant's perspective, an unprompted offer of credit might tip the balance and close a potential sale. It might help improve profitability, either via the interest the merchant generates in servicing the customer's debt or by encouraging the customer to purchase additional items.

The scenario described above involves not only automated, rule-driven actions, but automated, rule-driven decisions – e.g., whether to extend credit to a customer, at what rate, and so on. Both kinds of automation presuppose an ability to process and analyze data in real-time.

This is the remit of streaming analytics. The ability to process and analyze data in real- or right-time permits a new way of modeling – of representing – the business and its reality. The ability to integrate and analyze data at (or close to) real-time speeds has the potential to change the way the business makes point-in-time and day-to-day decisions. In the same way, for the same reasons, it has the potential to revolutionize business strategy. By integrating up-to-the-second data from more and varied sources, businesses can construct increasingly realistic models that capture and represent distinct aspects of their operations. They can arrange these models into combinations (called "ensembles") that more accurately depict the reality of their worlds. This is the raison d'etre of streaming analytics.

At a minimum, streaming technology makes it possible for the business to create a high-definition representation of itself – warts and all. Analogically, the business is able to "see" itself in a higher resolution, in more colors, via the creation of analytic models that not only incorporate data from more and diverse sources, but preserve more details and, as a result, present a richer context. But the time dimension contracts, too. Analytics produced in the data-at-rest model capture a reality that is several hours or minutes old; in the streaming (data-in-motion) paradigm, analytics have the potential to model a business reality that is – at most – seconds old. It's the difference between snapping a picture on a film camera, dropping it off at a same-day processing facility, and viewing it several hours later and … snapping a digital photo and viewing it more or less instantly on one's phone or computer. The takeaway is that streaming permits a more complete, and near-instantaneous, view of business reality.

Traditional analytic models achieve the equivalent of an aliased – jagged, gappy – view of the business and its operations. People (and software) rely on mathematical interpolation (sampling, error functions, etc.) to close gaps or to smooth out the jaggedness in the resulting picture. This is not unlike how the digital-to-analog converter (DAC) in a Bluetooth speaker generates smooth curves as it reconstructs an analog waveform. The digital waveform consists of a slew of point-in-time snapshots (samples) – much like a continuous sequence of plots on a scroll of graph paper. The DAC uses math to create curves that bridge the gaps between these plots, reconstructing a smoothed-out representation of the original analog waveform. So far, so good. The thing is, if you can capture more samples – say, 192,000 snapshots per second instead of just 48,000 – the DAC has to guess less as it reconstructs the signal. And the encoded signal is more faithful to the reality to which it is supposed to correspond.

STREAMING
and stream-processing explained

The same is true if you can record more (or more granular) details each and every time you capture a sample. So, instead of capturing, say, 65,536 possible details 192,000 times each second, the audio stream now captures (say) 16.7 million. Or, it might capture just a few more details while encoding a lot more information about these details. The result is that the Bluetooth speaker relies less on guesswork and more on empirical data. In the analogy, this is the difference between 16- and 24-bit audio. In business, this is the difference between a fragmentary as distinct to a holistic view of reality. What does this mean in concrete terms? Consider the business use cases that streaming data – and the ability to process streaming data in real time – can either enable (as the condition of their possibility) or augment (as a means of delivering fresher data faster and more reliably):

Supercharged sales. Businesses have always been sensitive to the importance of real-time data – especially in connection with sales and marketing. One of the earliest real-/right-time use cases involved analysis of clickstream data – i.e., data associated with a shopper's history and behavior on a merchant's website and, if available, on other websites, too – to promote cross-selling or upselling opportunities. In the past, too, many retailers – supermarkets, for example – used loyalty cards to build profiles of customers and their shopping habits. Today, however, companies have the opportunity to collect detailed data about the behaviors, interests, and backgrounds of their customers. The combination of ubiquitous connectivity (via connected devices) and affordable access to real-time processing via streaming data makes it possible for companies to supercharge their sales efforts. To cite just one example, the ability to analyze customer data in real time – ideally, while a salesperson is interacting with a customer on the phone, in a meeting room, or on the sales floor – provides new tools businesses can use to improve the customer experience and to maximize sales opportunities.

Transportation and logistics. Transportation is another vertical that was far out in front of the shift to stream processing. Sensor-equipped trucks, railway cars, planes, ships, etc. have revolutionized modern logistics. A staggering variety of sensors monitor equipment wear and detect imminent equipment failure. Data scientists, analysts, and other expert users have identified a myriad of new applications for sensor data, too. For example, trucks and railway cars are now outfitted with sensors that capture not only temperature and exhaust but audio telemetry data, too; this data is analyzed for certain characteristics (such as anomalously high temperatures in wheel bearings or brakes, or sounds in specific frequency ranges) that correlate with specific phenomena (brake or rotor wear, air compressor failure, incipient engine maintenance on trucks; hotboxing – i.e., incipient bearing failure – on railcars). In the legacy data-at-rest paradigm, transportation companies would collect, process, and analyze this data asynchronously, in large batches, usually once a truck or railcar reached its destination point; in the streaming paradigm, companies ingest, process, and analyze this data at close to real-time, proactively identifying and averting failure – saving lives (and improving services levels) in the process.

STREAMING
and stream-processing explained

Procurement and supply chain optimization. In a context in which supply chains are stretched thin, as in the still-ongoing COVID-19 epidemic, companies that have real-time insight into their own supply chains, as well as those of their suppliers, are at an advantage. Today, for example, large retailers use event-driven workflows – triggered by real-time event data – to automatically replenish materials as they are sold, or to dispatch excess inventory to store locations in which it is in demand. Real-time replenishment, in particular, can make the difference between having product on hand to use or to sell and having to wait weeks for upstream suppliers to replenish their own inventories. In companies that have developed real-time, event-driven transportation and logistics capabilities, this kind of automation confers other notional benefits, too: shipments can be redirected while still in route, for example.

Manufacturing optimization. Streaming technology has already transformed manufacturing. Most of the equipment in a modern factory – from conveyor belts to assembly-line robots, to lathes, shrink-wrap machines, light curtains, and so on – bristles with sensors. Manufacturers use this sensor-enabled equipment to proactively diagnose equipment failure, identify poor or inconsistent yields, optimize maintenance schedules, and to support other revenue- or productivity-enhancing use cases. The most revolutionary benefits are typically achieved by combining sensor data from manufacturing tooling with data from other sources – such as temperature and atmospheric sensors, audio sensors, etc. Data scientists and ML engineers construct analytic models that incorporate previously unknown variables – such as atmospheric humidity – to optimize manufacturing conditions in real-time.

Business process optimization. Event-driven automation is common in business today and is frequently used as a means to automate steps in business workflows, to partially automate business processes, and to simplify interactions between processes that cross business function areas. In most cases, however, event-driven automation, especially between heterogeneous platforms, applications, and services, does not occur in real time: the event data that triggers an action in a workflow or process is refreshed on the basis of predefined batch loading intervals. These intervals can be larger (as much as a day or more) or smaller (several seconds) depending on the technology used. In shifting applications and services to a streaming substrate, organizations can design rule-based, event-driven workflows that take place in, or close to real time. In practice, real-time access to event data has the potential to accelerate workflows and improve the performance of business processes. Other common business use cases include fraud detection – one of the earliest and most widely supported real-time use cases – and enterprise security, especially with respect to intrusion detection.

THE NEW STACK

The New Stack is not a platform monolith – à la Hadoop – but a combination of complementary technologies. Working in concert, these technologies address several different requirements:

- They provide a high-performance bus for all enterprise data (Kafka)
- They support real-time processing and analytics on streaming data (Kafka, Spark)
- They provide a scalable, fault-tolerant repository for persisting data (Cassandra)
- They support different kinds of batch-based data processing and analytics (Spark).

The New Stack is designed to be non-disruptive to an organization's existing IT systems, as well as to existing IT and business processes. The New Stack is deployed alongside – in coexistence with – an organization's existing data management, data integration, messaging, and data storage infrastructure. It expects to consume data from these systems and – via its high-speed, low-latency streaming bus – to make this data available to different types of (human and machine) downstream consumers.

This section explores the individual components of the New Stack.

THE NEW STACK

Kafka is several things. First, it is a high-throughput, low-latency "bus," or transport conduit, for ingesting, processing, distributing, and, persisting data – especially data from real- or right-time feeds.

Second, Kafka provides a built-in means of managing and organizing data feeds. This goes to its roots: Kafka started out as a publish-subscribe (pub-sub) messaging system. In the pub-sub model, data consumers "subscribe" to "topics" that correspond to data feeds. Whenever a data feed "publishes" to a pub-sub system, this data is distributed to downstream subscribers. Kafka retains at least some of this nomenclature – e.g., it manages data feeds as "topics" – even as it eschews the terms "publisher" and "subscriber" in favor of "producer" and "consumer." Virtually any data feed can function as a Kafka producer, including not just sensors and embedded devices but web servers, databases, ESBs, different kinds of distributed services (SOA, RESTful, GraphQL, etc.), file systems, and so on. Kafka also exposes a so-called "connect source API," which basically describes any Producer that an organization does not control: e.g., social media services such as Twitter, LinkedIn, Facebook, etc.

An open-source component, called Kafka Connect, permits Kafka to ingest and process large volumes of data from RDBMSs and other types of databases. Lastly, human or machine subscribers – "consumers," in Kafka's argot – can subscribe (or be assigned) to topics. In theory, Kafka is simple enough to scale: under its covers, discrete topics map to "partitions," hosted by Kafka "brokers" – basically, server clusters – which ingest data from producers. A partition is conceived as a scalable building block: add partitions to a topic and you increase Kafka's processing capacity for that topic.

Third, Kafka is able to process and transform data in real-time. This is via an open-source library, Kafka Streams, which Kafka consumers invoke to perform operations on data as it transits the Kafka bus. Similarly, Kafka consumers can invoke a separate component, KSQLdb, to perform SQL-like manipulations on in-flight data. So, for example, developers can design applications and services, or create reusable data pipelines, that invoke Kafka Streams or KSQLdb in order to join, validate, and/or deduplicate data; similarly, data scientists and ML engineers can use Kafka Streams or KSQLdb to create ad hoc data pipelines that perform the same tasks. Again, both technologies are useful for performing operations on data as it transits the Kafka bus; in practice, this means that consumers function as producers, feeding data back into Kafka for consumption by other consumers.

Fourth, Kafka also supports data persistence. Kafka itself can be configured to log and persist all message traffic, but – in the vast majority of cases – organizations opt to persist Kafka message traffic into a "sink:" for example, a cloud-based object storage service such as Amazon S3 or Azure Blob Storage, or a cloud/on-premises data store, such as an RDBMS, Cassandra, Hadoop, etc.

THE NEW STACK

Spark. The New Stack requires a scalable multi-purpose compute engine capable of performing operations on data of different types, not only at (or close to) real-time, but in batch-based operation, too. Spark is tailor-made for these requirements. In the first case, it offers built-in integration with Kafka and with other streaming sources via its Spark Streaming engine, which permits close to real-time analytics on streaming data. Second, Spark is a multi-purpose compute engine; in addition to Spark Streaming, it exposes SQL (Spark SQL), ML (MLib), and graph-processing (GraphX) engines.

The upshot is that Spark is adept at multitasking: in practice, multiple Spark jobs, involving a mix of different Spark engines, some running as multiple (parallel) instances, can execute concurrently. So, for example, a developer could build an application or service that tasks Spark Streaming with ingesting data at micro-batch intervals from Kafka and performing a sequence of analytic operations on it. If the application or service needs to process a large volume of data very quickly, the Spark cluster manager (e.g., Kubernetes) can spawn multiple (parallel) instances of Spark streaming. Optionally, the data engineer or developer could invoke Spark SQL to perform additional operations on this data and persist the results to a downstream repository, such as Cassandra. Depending on the characteristics of this workload, Kubernetes might spawn parallel instances of Spark SQL, too.

Third, Spark is an in-memory compute engine, which makes it ideal for processing data in real-time, as well as for analytic workloads in general. An in-memory engine such as Spark reads data from and writes data to physical memory as it performs a sequence of operations on data. Spark is also optimized to make efficient use of different types of processor caches (including registers) in order to “pin” instructions and data into memory. For the same reasons, Spark’s in-memory engine is better able to exploit the on-chip parallelism 3 built into most modern CPUs. In-memory matters because physical memory is several orders of magnitude faster than physical disk. (This is true in comparison to SSD and NVMe flash storage, too.) Instead of accessing data via a local bus – PCI Express on most x86 systems – processor cores access data via a memory controller integrated into the CPU package itself. System memory has much lower latency (measured in nanoseconds) and higher bandwidth – 200 GB/s or greater in eight-channel memory configurations – than disks, SSDs, or NVMe storage.

Fourth, Spark is also useful as a scalable engine for processing data-at-rest. Real-time processing is essential for supporting time-sensitive and/or event-driven workloads and use cases. The irony is that many if not most of these workloads and use cases are first identified and refined by analyzing data-at-rest: terabytes and (notionally) petabytes of data generated by both streaming and conventional data sources and stored in different on- and off-premises repositories. This is the remit of data science, business analysis, ML engineering, and other experimental practices. Data scientists identify patterns, signatures, correlations, etc. by analyzing data derived from diverse sources. Their analysis might link (for example) elevated humidity levels with poor manufacturing process yields. Once they establish a correlation, data scientists and ML engineers work to design and test different kinds of remediations: e.g., alerts, rule-driven, automated actions, and so on. Spark is a useful engine for processing the data engineering and data analysis workloads associated with this experimental work.

THE NEW STACK

Cassandra. The New Stack requires a scalable, fault-tolerant means of persistence: a database, but not just any database – a distributed database. Apache Cassandra fits this bill. It can be deployed in clusters – that is, networks of servers working in parallel and acting as a single system – that are specific to a single location or which span geographically distributed locations. This permits an organization to deploy Cassandra clusters across multiple, geographically distributed data centers.

Cassandra is able to store and manage semi-structured data (such as JSON objects, TXT and CSV files, etc.) and multi-structured data, including audio, video, and image files. (Like a relational database, Cassandra is not ideal for storing large audio or video files, which it ingests as binary large objects, or BLOBs.) Cassandra provides support for RDBMS-like data consistency guarantees and enforces RDBMS-like transaction safeguards. It achieves ACID 4 -like transactional guarantees.

One other thing: the New Stack is easily scaled. Each of its stream processing and general-purpose data processing components – namely, Kafka and Spark – can be deployed on a single node (i.e., server), on multiple nodes, or managed via an orchestration manager such as Kubernetes). Each of its components can also be scaled independently: that is, an organization can improve Kafka's performance by configuring additional parallel or concurrent instances.

Spark scales in the same way. The New Stack's persistent repository, Cassandra, is a distributed database that can be deployed on a single node or can scale to support hundreds of clustered nodes. This makes the New Stack suitable for deployment in different contexts or locations: from the enterprise core to the enterprise edge.

CONCLUSION

Many organizations are integrating streaming into their core IT infrastructures, but the most forward-thinking of organizations understand that stream processing is less a complementary technology – i.e., something that merely augments one's existing data and software architectures – than a superseding or sublating technology. In other words, stream processing is a new paradigm – predicated on novel methods of ingesting, moving, integrating, and using data – that subsumes or "takes up" extant data management and application integration paradigms. So, for example, streaming architecture is usually underpinned by a stream-processing stack (Kafka, Spark, and Cassandra) that provides a transport for data distribution, flexible engines for processing and analyzing streaming data, and a means of persisting this data. In practice, the stream-processing stack can act as a conduit for data traffic of all kinds–including the batch ETL processes that populate the data warehouse, the routine messages that applications exchange with one another, and the data pipelines built by ML engineers and other expert users. In this scheme, an organization's existing data and application integration processes do not go away; rather, over time, they are subsumed as part of the streaming stack. In this way, data from RDBMSs, data warehouses, enterprise applications and message queues, and other sources is made available – via the streaming stack – to new consumers. Streaming likewise gives existing consumers a new – possibly more convenient – option for acquiring data. Developers, data scientists, ML engineers, and business analysts, especially, stand to benefit from streaming and stream processing.

It is essential to consider the implications of this. Streaming entails a reimagining of software and data architecture. It opens up new opportunities and poses a unique set of technical and socio-technical challenges. Section III explored some of the business use cases that either are made possible by the streaming paradigm or which (with regard to a large number of established business use cases, practices, etc.) stand to benefit from real-/right-time access to time-critical data. Organizations must determine how to design, deliver, and support these use cases, most of which also incorporate data from traditional sources – for example, a data warehouse, an operational data store, an enterprise service bus – that must be connected to and exposed via the streaming bus. This underscores a critical point: streaming is not a rip-and-replace proposition. To implement streaming capabilities in the context of an existing IT infrastructure is to integrate streaming capabilities with, as part of, that infrastructure. Reconciling this statement with the claim that streaming is "less a complementary … than a superseding technology" is simple enough. In the overwhelming majority of cases, the existing data and application integration processes that (for example) populate a data warehouse or facilitate the exchange / interchange of data between applications and services continue to operate as normal. However, the resources they populate (a data warehouse, a data lake, an operational data store, etc.) or the messaging backbones they depend on (an ESB) are likewise connected to and exposed as potential data feeds via the streaming bus.

The streaming paradigm is complementary in the sense that it does not displace, but coexists with, existing data management and application integration assets. So far as DBAs, middleware administrators, developers, and other stakeholders are concerned, the day-to-day operation of these systems is essentially unaltered: the strictly governed, reusable flows of transformed (or "integrated") data that populate the data warehouse – and the ESB message traffic that is the primary means of exchanging or interchanging data between apps and services – continue to function as normal.

POSTCRIPT

As we go about our daily activities, we are almost always connected: if we create and consume vast amounts of data, this is because we depend on different kinds of connected devices that (with certain common exceptions) never leave our sides. In a sense, human existence is already augmented – by connectivity. Undergirding the irresistible logic of streaming is the ubiquitousness of connectivity.

On the one hand, ubiquitous connectivity makes it possible for companies to market products to customers at little to no cost. On the other hand, and paradoxically, ubiquitous connectivity increases the importance of physical proximity to, and quality interactions with, customers. There is a scene in the 2007 film No Country for Old Men in which the lead character – having been wounded and hospitalized in Mexico – once again crosses the border into the United States. Clad only in a hospital gown and a pair of Larry Mahan cowboy boots, he walks into the same store from which (a few days previously) he had purchased said boots. "How them Larry's holding up?" the stone-faced salesman asks him. Unfortunately, not everyone in sales has as good a memory, or is as unflappable with customers, as this fictional salesperson. In the era of streaming data, they no longer have to be. Today, for example, a retailer can use streaming data to identify a specific customer while she is parking her car. A salesperson can greet a customer – by name – as soon as she walks into a store.

More important, retailers can now push out detailed information about each customer's sales history to tablet-toting salespeople on the store floor. Better still, they can develop event-driven analytics to equip salespeople with sales prompts. Assume, for example, that a customer purchases running shoes every 7-8 months and that it has been almost 7 months since her last purchase. "How are those Brooks holding up?" the salesperson might ask. The upshot is that each and every time a connected customer walks into a franchise location, drives into a shopping center complex, etc., a business has a better-than-even opportunity to sell them something; moreover, each and every in-person interaction with a customer gives a business an excellent chance to improve its relationship with that customer.

Imagine this same situation playing out, mutatis mutandis, across all verticals. Imagine it playing out in different types of relationships: between companies and suppliers, business partners, and so on.

Key to this is the ability to process, analyze, and deliver data in real-time. Companies of all sizes expect to be able to identify and to access detailed information about their customers, suppliers, partners, even their competitors – and, more important, to collect data about them. They expect to develop new business use cases, create new products, enter into new markets, and perform other actions that are critically dependent on real-time data. Streaming – and stream processing – provide a scalable, reliable, low-latency data distribution and data processing substrate for these and other applications.

ABOUT THE BLOORGROUP

The Bloor Group is an independent research firm that produces objective, high-quality analysis of enterprise technology products, services and markets via new media outlets and traditional research methods.

As a hybrid New Media/Analyst firm, The Bloor Group seeks to educate business and IT professionals on the array of technologies and methods available for managing information, and provide innovative vendors with the resources to promote enterprise software and services.

ABOUT PERFORCE

Founded in 1995, Perforce Software is a leading provider of highly scalable development and DevOps solutions designed to deliver dynamic development, intelligent testing, risk management, and boundary-less collaboration. We partner with organizations that must accelerate time to market and reduce risk in environments where the cost of failure is high.

Our global experts bring insights, experience, and best practices to enterprises across many verticals. We are trusted advisors for leading companies in automotive, semiconductor, financial services, game development, virtual production, medical devices, embedded systems, retail/consumer packaged goods (CPG), travel and entertainment, and industrials.

Today, Perforce solutions are used by 75% of Fortune 100 companies to innovate at scale.