The Critical Moment:

Getting Operational Intelligence from Logs, Metrics, and Transactions

Table of Contents

The Critical Moment: Getting Operational Intelligence from Logs, Metrics, and Transactions

For Java developers, IT ops professionals and members of the DevOps group, the ultimate advantage would be to know everything as it happens in their business—and everything that could happen. To know all and see all with complete vision is the competitive ideal: operations managers detecting performance problems before delays arise, the company discovering trends the moment they form. If there is a function in modern technology that offers anything close to this ideal of omniscience, it's analyzing and visualizing machine data in real time.

Such awareness is so difficult to attain, and it is often impossible for companies to know in advance what events need to be analyzed and when analysis must happen. IT must store everything and analyze everything or risk missing the most important evidence of operational lags, risks or rising customer trends. For security compliance reasons alone, enterprises are required to maintain good logs, store logs for one year, secure these logs—and review them daily.

An important consideration is whether staff is looking at the data with the right degree of resolution. A low-resolution view lets support staff isolate a problem to, say, a specific infrastructure tier and then pass the problem to a specialist—who will start all over again diagnosing the problem with a different tool. While this common approach works, it is time-consuming, expensive and disruptive. A more productive approach is to use a high-resolution analysis that enables application support to diagnose a problem and immediately begin its resolution.

Real-Time Analytics for Operational Insight

A steadily increasing number of operations staff throughout the IT economy are finding analytics a compelling part of their tactical suites. Nearly half of respondents to a TDWI survey revealed that they are engaged in real-time analytics for operational insight. Roughly 25% of DevOps professionals surveyed by IDC reported a desire for analytics. And the competitive benefits of real-time data are clear. An Aberdeen Group survey revealed that executives who were engaged in real-time streaming analytics averaged nearly 15% improvement in operational cash flow and overall profit. They reported a 6% reduction in operating costs. And 70% of them reported that they are able to hold regular operational reviews.

Indeed, analytic platforms designed to deliver operational insights are rising in prominence. Yet for a company to act on trends and problems with precision and immediacy, a real-time, operational intelligence system must allow users to analyze perishable data while it still matters. This is essential in order to achieve the ever-important business goals of keeping customer services levels high, applications running, developers productive, and costs low. Transaction behavior, delivery delays, consumer behavior: the value of this sort of data is ephemeral, its significance at first so high to users and then diminishing rapidly as time passes.

Figure 1: An example of an operational intelligence dashboard from jKool. The data displayed is streaming into jKool from Log4j and Syslog Collectors.

This type of data has an expiration date. As Forrester Research asserted in a 2014 report, “Business won’t wait. That is truer today than ever before because of the white-water flow of data from innumerable real-time data sources. Market data, clickstream, mobile devices, and sensors may contain valuable but perishable insights. Perishable, because the insights are only valuable if [users] can detect and act on them right now.” For IT Operations, these data sources also include Java logs, Syslog data, various server and application metrics and transactions.

Too many times, developers are pulled from their critical and customary roles as coders of innovative apps, roles that are essential to the growth of their businesses. They are instead

thrust into triage assignments in application support, drafted into diagnosing problems that, as the common perception goes; only they have the skills to address. They lose hours, days, and weeks scanning and annotating logs, tracing transaction and message flows and measuring application performance, their organizations finding no easy way to transfer their skills to less expensive, less critical personnel. Often operational in origin, machine data is scattered across multiple logs, servers and event infrastructure tiers.

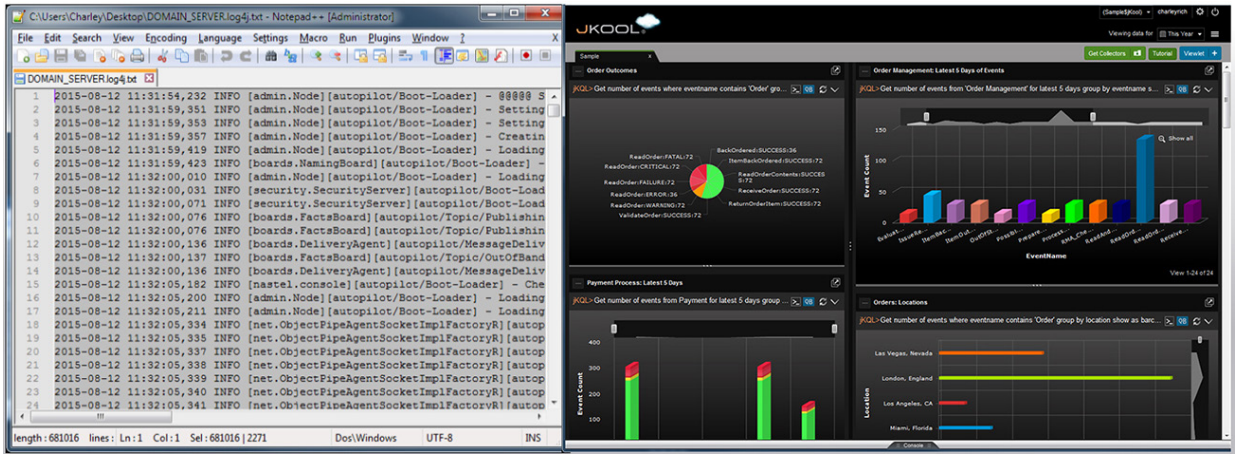

Figure 2: Traditionally, Java developers are called in to diagnose application behavior by interpreting log data as presented above on the left. Using jKool's analytics and visualization the upper right represents a presentation that Application Support can use.

In such a workflow, it can be very challenging for staff to piece together the story the data has to tell. Yet the right form of real-time, operational intelligence—intelligence that brings machine data into a coherent, consumable and configurable visualization—is a superb way to scrutinize a broad range of performance metrics. From Java logs to Syslog, metrics and transaction records, such intelligence can deliver instant insight users need to

keep the operation running, liberating developers to do their true jobs creating apps that drive the business forward.

Professionals at every level in development and operations can take advantage of real-time, operational intelligence to boost immediate insight into their machine data. Seeing performance issues like troublesome trends, patterns and urgent problems as they unfold is crucial—yet awareness is only half of the game. Action and decisiveness are important capabilities as well. With real-time analysis highlighting perishable data at its most useful heights, employees are empowered to respond to issues rapidly, to anticipate difficulties, and to make fully informed decisions.

A real-time operational intelligence product, jKool offers immediate insight, simplified, swift analytics and visualization to Java developers and DevOps running production, test, and development environments. Users can observe Java transactions as they occur; teams can consolidate all logs and metrics onto a single configurable web dashboard. With this single pane of glass, staff no longer has to piece together stories from disparate logs or tools. Users get a detailed record of their transactions stitched together, along with metrics and logs, and are then able to see their flow of execution across applications, servers and infrastructure tiers, as well as the environment context. This creates a high-resolution image of application behavior.

Available as a SaaS offering, jKool allows IT departments to reduce time fixing problems and spend more energy on building business resources, focusing on innovation, as they are freed from providing endless support. Using SaaS, there are no servers, schema or database or storage to manage. In either on-premises or using SaaS versions, jKool can

greatly accelerate the cadence of new releases, enabling the business to increase its agility. As an open-source tool, jKool embraces the most compelling technologies in the world today and makes trends and patterns reassuringly clear to users.

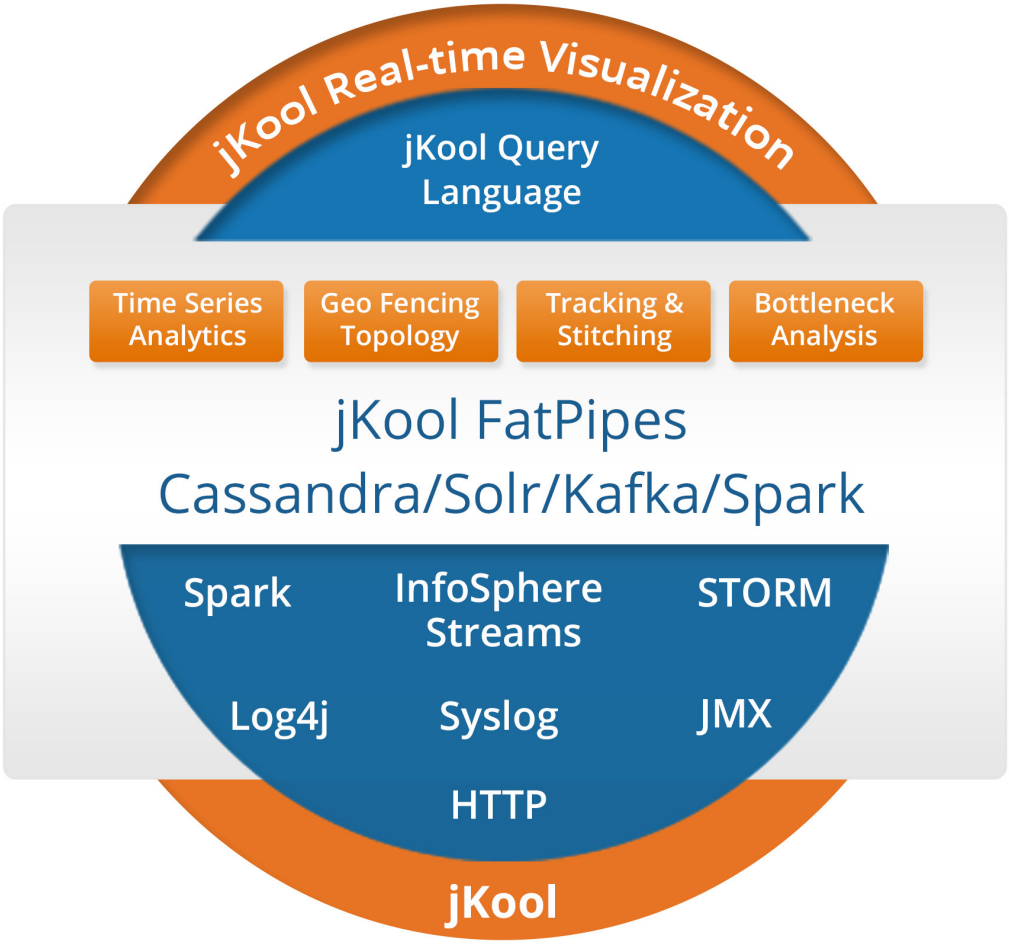

Figure 3: jKool Architecture is built on open-source technology orchestrated via FatPipes micro-services.

Addressing the Real Problem: Too Much Data, Too Little Time

With operational realities generating immense, unorganized levels of data at a relentless pace, many enterprises are unable to keep up with the quantity of data they are accumulating. Institutions collect more data than they can manage, the sheer amount and organization of the data making it difficult to achieve the instant insights staffs need for the decision-making that keeps costs at suitable levels. The competitive advantage that rests in such data is buried in its huge volume and its inconsistent organization. Relational databases are taxed, scalability and performance lagging under the weight. Professionals have no time or means to figure out what the data means quickly enough—losing the opportunity to act on trends and patterns before the data's potency wanes. If organizations can move quickly enough, there is opportunity hidden in the data that can provide value.

Support and development of mission-critical Java applications can be an area of tremendous drain on IT departments. Manual analysis of logs and metrics is costly, slow, and results in errors. Through jKool, users can streamline this responsibility, delivering fast identification of problem areas and performance. Users can address the data with a mere click or two. Machine data can be presented in a way that enables actionable insight into misbehaving applications and business processes without requiring constant developer involvement. Diagnostics are clear.

jKool permits organizations to leverage fewer expensive personnel in support, who will not need to do heavy lifting to keep the business running. For development organizations,

this freedom means higher quality releases, delivered in less time. This can be a big boost in overall productivity for all concerned: development, IT ops and DevOps.

Analytics in the moment are useful for IT to achieve aims like loss prevention, projecting when users will run out of resources and rapid diagnosis of problems. These capabilities are valuable to any number of industries in a diversity of contexts. The ability to simulate (provided by jKool) an application is of great help to enterprise architects, who can use this to improve the effectiveness of pending deployments, as they will be able to visualize transaction dependencies and interactions prior to the actual deployment. They can then adjust their plans based on what they learn from the simulation. jKool's simulator enables this important benefit.

Built to consume time-series machine data, jKool ingests, correlates and visualizes this operational data in memory, in real time. As jKool consumes data, it automatically stores, sequences, correlates, groups, enriches, synchronizes and computes ingested data in-memory and in persistent storage, delivering the instant insight that positions an enterprise to act with speed and confidence.

jKool Features

From 2012 to 2014, Forrester tracked a 66% increase in firms using streaming analytics. Yet adoption has its challenges, as many of these solutions do not provide out-of-the-box operational intelligence. In essence, such solutions are toolkits from which users must build their own applications. Real-time visualization of streaming data is usually not provided. Other offerings do not yet provide the breadth of functionality—including a useful query language—that is needed to derive instant insight from the data ingested.

Figure 4: The Viewlets show two different views of a "stitched" transaction flow. The upper Viewlet shows the flow on a geographical map, while the bottom Viewlet presents this as application topology.

Easy to use and built to facilitate an array of queries and illustrate metrics in bold, versatile graphics, jKool offers visualization capabilities to address all elements of Java and other logging and metric sources. The system connects to several logging utilities: Log4j, SLF4J, Logback and Syslog. It also ingests data from collectors for various extensions, including JMX, Java EE, HTTP and Spark. jKool consolidates and summarizes all logs and metrics, permitting users to detect anomalies and view trends in error rates over time. Self-service capabilities enable users to add metrics via an open-source API, allowing an automatic

track and trace of Java transactions. Offering a wide and comprehensive view of the landscape, jKool automatically stitches together transactions and scans problems in operational flow and performance in order to determine problem causality.

jKool's quick-to-use language for analyzing behavior, jKQL, communicates queries with a direct, precise, and logical simplicity. Through jKQL, users initiate ad-hoc analysis of a range of metrics. Using the jKQL language, users can talk directly with their data and get answers. jKool's process of querying is designed to let the user learn things they didn't already know. For example, a Java developer might ask, "Get worst 10 events fields applname order by elapsedtime desc show as barchart" and receive a barchart representation of the 10 most chatty classes over the last hour. The user can then ask more questions for better insight into the causes and realities of relevant operations. Users can drill down into the data returned. They then select "analyze" and see a multi-panel chart that shows event elapsed time, the volume of all activities and user-selected snapshot data over time.

Stitching is a process of connecting events from seemingly unrelated sources, much like putting together a jigsaw puzzle from many different pieces. These events may occur at different times, come from different applications and have different attributes – yet they are related to a specific activity and tell a story about your application: such as payments, claims, searches, orders, etc. The process of stitching combines these events to describe a lifecycle of a composite activity spread across multiple applications, compute analytics and discover topology and flow. The result is high-resolution view of application activities, improved root cause analysis, dramatic reduction in MTTR (mean time to problem resolution) and increase in MTBF (mean time between failures).

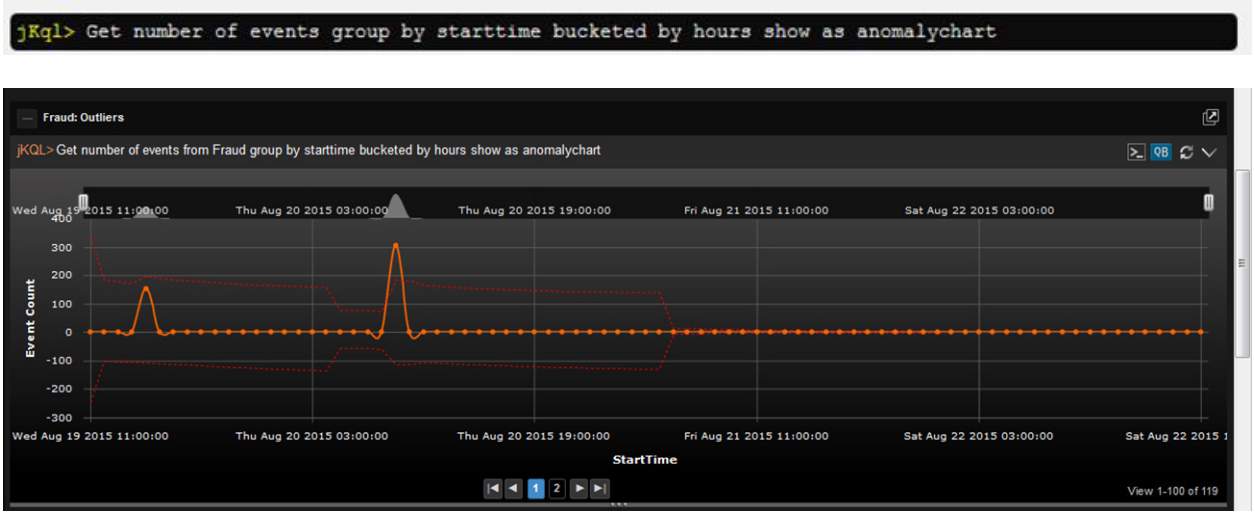

Figure 5: The above image shows a jKQL query that gets events, groups them by when they start and puts the results into hourly buckets. In addition, the query shows how jKool automatically detects anomalies. The Anomaly Chart uses Bollinger Bands and Exponential Moving Averages (EMAs) to detect anomalies when an event crosses either the lower or upper bands.

Figure 6: This is an example of a real-time query. Notice the verb "Subscribe," which tells jKQL to send matching events as soon as they arrive, even before they are saved to disk.

Alternatively, a business user can enter, say, “get worst 50 orders for the last hour.” jKool’s visualization graphics can display data in a number of different formats, including candlesticks and trend lines. The graphics can then simultaneously show the elapsed time for each order, the value of orders and the overall volume of events during that same time period. The user can view the flow of execution of the order transactions and also isolate them geographically.

jKQL lets users perform analysis using streaming operators. They can compare and highlight differences between activities. Compatible with both browser and tablet formats, jKQL displays results as tables, charts, scorecards, heat maps, candlesticks, or comparison and topology. The visualization capabilities span:

- **Real-time:** Using in-memory analytics before events are written to disk, jKool can display events and metrics as they arrive.
- **Historical:** jKool provides full analysis of historical data and enables comparison between different time periods all using the same dashboard as real-time.
- **Graphing:** Visualization options include tables, scorecards, line charts, bar, column charts, pie charts, area charts, summary views and more.
- **Streaming operators:** Comparison of metrics can involve EMAs, Bollinger Bands, count, sum, min, max, avg, first, last, top, bottom latest, earliest, best, worst, largest, smallest, shortest and more.
- **GeoFencing:** jKool’s GeoFencing capability can segment events and activities by geolocation, plotting relationships and displays results on a map. jKool automatically converts latitude and longitude to text locales.

- **Comparisons:** jKQL can designate metrics as master data and display other metrics in comparison to main data.
- **Automatic detection:** jKool automatically detects anomalies and displays those using statistical functions.

With a strong offering of features, jKool grants companies an easy set-up and a level of insight and usability that gets staff acting on trends and problems in short order. For developers, the benefits soothe a constant stress in modern IT departments: the endless pulling of developers into the application support role, a reoccurring situation that drains so much energy and time from their main mission.

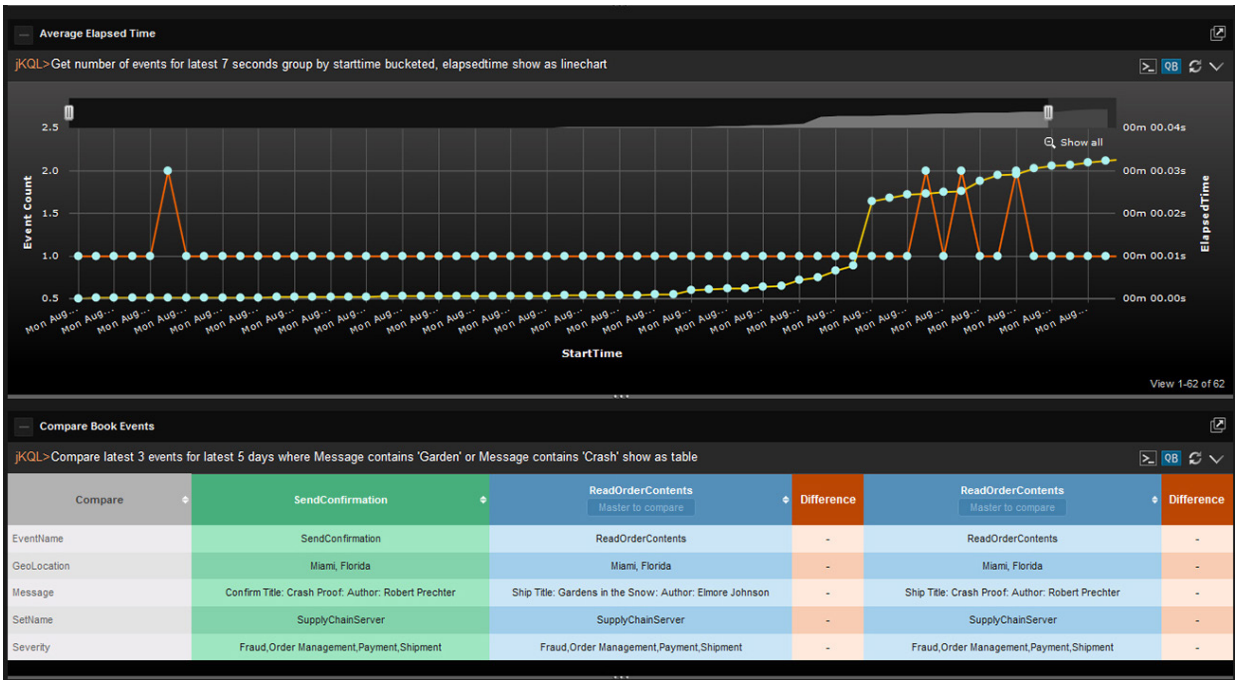

Figure 7: The Viewlet at the top shows an analysis of time-series events using Bollinger Bands and Exponential Moving Averages (EMAs). The Viewlet on the bottom is a comparison showing the differences between events.

Operational Intelligence and the Developer

When an application misbehaves, a developer often has to sift through multiple log files in order to trace the path of each application and find out what went wrong. With multi-threaded applications, these logs can appear chaotic, each thread writing a log entry that may have nothing whatsoever to do with the prior entry. Such a lack of coherence makes diagnostics quite challenging and often requires staff using pencils and paper, chalk and string to discover what happened. This is a tedious manual process that quickly becomes quite difficult, with each application written to different log files, sometimes on different servers and many times even on differing infrastructure tiers. Multiply this effort by a large number of servers and the situation rapidly becomes unmanageable.

IT Ops can find it difficult to follow the flow of execution, requiring significant involvement by developers, the most critical and expensive employees in the organization. Though the purpose of this diagnostics effort is to understand performance issues, the flow of execution as logged may not appear linear. In the jumble of distributed data, causality is very difficult to discern.

Unfortunately, this type of diagnostics occurs all the time in modern IT departments. jKool offers an alternative to this taxing app diagnostic model that is so commonly a default response by companies. By feeding jKool with a range of data collectors, developers will find the set up and preparation of the system easy to do.

jKool uses open-source collectors to stream machine data such as logs and metrics into both on-premises or cloud-based SaaS platforms. These collectors are downloadable from jKoolCloud.com with registration or, alternatively, from GitHub. Each collector enables data gathering and delivery to jKool.

Java developers can use the Log4j Collector, an appender that allows developers to add an additional target to the Log4j configuration without any edits to application code. SLF4j, Log4J, and Logback users can consolidate all Java logs in a single pane of glass; these log visualizations improve diagnostics and root cause identification. A truly comprehensive view of the landscape emerges, as users can search log entries across applications and servers, while annotation of log entries bolsters diagnostics.

Other application developers can make use of the simple RESTful API to send their data to jKool.

DevOps or IT Ops can also deploy the StreamJMX Collector to send Java performance metrics to jKool from any relevant Java servers in the enterprise's infrastructure. Downloadable on the jKool site, StreamJMX is a lightweight framework to stream and monitor JMX-borne metrics. IT staffs traditionally monitor JVMs through remote JMX monitoring tools, which requires the team to deploy and manage JMX configurations, ports, and SSL. Such remote connectivity is notoriously weak and raises cyber vulnerabilities—further burdening IT with security obligations.

StreamJMX allows developers to stream JMX metrics from JVM out to destinations throughout the network. This functionality allows users to maintain a history of JMX metrics across multiple JVMs. Streaming JMX from within JMV is often a more desirable alternative to remote sampling when monitoring large number of JVMs, offering improved security (no open JMX ports) and less administrative overhead associated with opening and securing remote JMX ports and connections.

jKool allows for monitoring of JEE Web applications through HTTP Servlet Filter. Users can measure performance, attenuations, and signals; dashboards can display variable histories

of response times. With entire organizations engaging these applications, IT can track end-user behavior and location.

IT ops professionals using Syslog for system management and security can use jKool to bring together logs from multiple servers on one screen. Viewlets can illustrate the consolidated logs to help with assessing diagnostics and root cause. Users can search Syslog entries across applications, which can be annotated.

With these collectors engaged, jKool automatically sequences, correlates, groups, enriches, synchronizes and computes everything it can about the data as it arrives—immediately providing a rich visualization to the user.

The collectors come with documentation describing deployment methods and details. Before deploying a collector, users can test their visualization with jKool's built-in Simulator. This feature generates and displays a randomized version—at selected intervals and volumes—of what users can anticipate in terms of viewing data.

Because jKool's design prioritizes simple usage, once the data is flowing to it—through the user-friendly jKQL query language and the detail and flexibility of the Viewlet visualizations—less expensive IT staff can serve in support roles for the system.

Developers will find jKQL compatible with a host of data collectors. Equally critical to the system's design is the priority of perishable data, especially machine data that must be captured and understood in the moment. Only then can professionals take the decisive steps that lead to reduced costs and increased profits.

jKool is built on open-source software including the Apache Cassandra NoSQL DB from DataStax, Apache Storm, Apache Spark and Apache Kafka. jKool's FatPipes micro-

services architecture is designed to orchestrate these open-source technologies into a solution offering extreme scalability and ultimate configurability. Complex event processing delivers faster analytics from in-memory resources. This processing capability goes right to the heart of the challenge—perishable data must be captured, analyzed and visualized in real time. With open-source tools for data ingestion, developers can easily use jKool for their needs for real-time operational intelligence. Developers and DevOps and IT ops personnel engaged with the SaaS version also benefit from the luxury of having no servers, databases, or schemas to maintain.

With scalability so valuable as operational data levels continue to rise, jKool's Lambda Architecture aims to handle vast volumes of complex, concurrent data streams. The system's compute grid automatically runs queries in parallel to enhance performance and elasticity.

Figure 8: jKool's multi-panel "stock" chart for analyzing multiple data types over the same time intervals to detect hidden relationships

Conclusion

Business is about anticipation and awareness—knowing what is happening, what could happen, and how the company should act to resolve problems and outperform competitors. The pace of operations today can offer tremendous amounts of data to businesses—amounts that can intimidate and overwhelm as much as enlighten. The common approach to data analysis is unable to match this pace. Developers lose too much of their valuable time serving as ad hoc data managers, piecing together logs from disparate applications. By the time they are able to discern patterns in the data, the opportunity to respond to underperforming operations has vanished.

jKool provides awareness to such businesses by delivering clear, consumable operational analytics to IT and professionals. The product allows users to query data and dive deep into granular details to compare, summarize, and ultimately make swift business decisions.

Through jKool's simplified user design, less expensive IT staff can serve in support roles for the system. Prioritizing extreme scalability and ultimate configurability, jKool's complex event processing delivers faster analytics from in-memory resources. This processing strength goes right to the heart of the challenge—perishable data must be captured in real time. With open-source tools for data ingestion, developers can easily use jKool for their needs for real-time operational intelligence. Developers, DevOps and IT Ops personnel engaged with the SaaS version also benefit from the luxury of having no servers, databases, or schemas to maintain.

With scalability, swift visualization, and easy implementation, jKool aims to empower companies to act with confidence in fast-moving times. With jKool delivering streaming, high-resolution analytics, insights are more readily realized, staff can take more focused action faster, time to remediation is shorter, and a crucial, ultimate aim is achieved: more customers are satisfied with much better frequency.