Ovum
TMT intelligence | informa

The Real Costs and Benefits of Open Source Data Platforms

An Ovum white paper for Striim

Publication Date: 01 August 2017

Tony Baer

Summary

Catalyst

Today, when we come across new platforms and companies offering big data solutions, one of the first questions that we ask is whether the underlying technology is open source. As a business model for the software industry, open source has come a long way since Red Hat pioneered this model to become a successful billion-dollar company. Many enterprises have embraced "open source first" strategies with the requirement that there be robust commercial support behind the technology. But how far can enterprises take this strategy? The strongest proponents and pioneers of open source software, including Google, Facebook, and Netflix, continue to also use vendor-specific technology solutions that deliver unique value. So where should enterprises draw the line?

Ovum view

Open source *projects* work best when the underlying technology is or should be commodity, and the technology is mature enough that it can be organized for outside eyes to look at and work with. Open source *products* work best when components are integrated and supported by a viable technology provider. Survey data from Black Duck Software – which conducts annual surveys of open source development activity – reveal that operating systems, database platforms, and development tools are currently the hot spots.

There is a myth that open source software is "free" because customers do not pay for software licenses. With open source, there are costs associated with maintaining, updating, and integrating the software; those tasks are performed internally by the IT team or externally through annual subscriptions paid to commercial providers of open source technology "distributions."

Depending on the maturity or breadth of the open source project, there may additionally be hidden costs when enterprises assume the tasks of evaluating the scalability and touchpoints with other open source or legacy enterprise software to ensure that the open source project meets security, reliability, ease-of-use, and scalability requirements. This is especially critical for open source projects that are still maturing, where core components such as APIs may change.

Ovum believes that a hybrid approach that blends open source with proprietary software is becoming the norm for enterprises and technology providers alike because it is ultimately the most cost-effective. It avoids reinventing the wheel when it comes to commodity, foundational technology, and focuses the value on the unique IP that the commercial vendor brings to the last mile. For most scenarios, enterprises will rely on technology providers that support their own mix of unique and open source IP, delivering the level of support that enterprises have come to expect from enterprise solution providers.

Key messages

- In the big data sphere, open source has become almost a default option in several product categories: big data stores, advanced data science, and machine learning-related languages and frameworks.
- Hybrid models are becoming popular for packaged BI tools and analytical applications, leveraging open source technologies such as Spark, programming

languages such as R or Python, and/or machine-learning libraries to extend their core products.

- Open source technologies typically leave off at the application tier, where enterprises rely on unique business logic to differentiate, and at "the last mile," where technology providers integrate components and ensure reliability through guarantees of delivering consistent, predictable service levels.
- Open source *software* may be free, but there are both monetary and time-based costs associated with supporting, integrating, and updating it.

The draw of open source

The rising barometer

Open source has had an enormous impact on the big data technology ecosystem. Although not every data platform or component is available as open source, the vast majority are. For example, Hadoop, along with a number of NoSQL databases, are available as open source, as are a significant proportion of compute engines and algorithms.

A good barometer of the progress of open source can be found by comparing notes from Black Duck Software's annual "Future of Open Source" surveys dating back over a decade. The 2016 survey indicated a doubling of use of open source to run business IT environments since 2010. This year's survey pointed out that the preponderance of open source was in operating systems, database platforms, and development tools, and that in the next 2–3 years, the hot spots for open source will be cloud, database, and big data platforms.

Community is key to success

Open source works best when the appeal of the software is broad enough that it can draw a critical mass of contributors to the community. When the community hits critical mass in terms of breadth, the open source project provides access to resources and intellect that could surpass those provided by a single vendor, such as what has happened with the Apache Spark project. Successful open source *communities* foster rapid innovation. Not surprisingly, this model works best with general-purpose platforms, compute engines, or modeling approaches that draw large communities of developers; for machine learning, open source is viewed as a means for generating interest and expanding the technology portfolio.

Degrees of "open"

All open source projects are not created equal. The difference between community- and vendor-led projects impacts how decisions are made regarding direction of the project, and the priority lists for new features and fixes. Community-led projects will deliver more of the openness and visibility that is associated with open source compared to vendor-led projects.

Open source: The real costs and savings

The true cost of "free"

Open source software is often thought of as "free" software because the source code is readily available, and can be downloaded for scenarios ranging from test and evaluation to production at the customer's risk. Open source has played a large role in changing expectations for how much enterprises are willing to pay for software.

However, there is a cost to this "free" software. Like all enterprise software, there will be costs relating to support. And, depending on the completeness or complexity of the project, there are costs associated with integrating, upgrading, extending, and maintaining the software solution.

The savings

The licensing model for open source software is different from traditional proprietary software. The core open source software code or functionality may be free. Because there is no sticker price, it is also seemingly more cost-effective to scale out with open source. For example, a 100-node Hadoop cluster is theoretically the same price as a two-node Hadoop cluster. Commercially supported open source projects have largely adopted the subscription pricing models that were first made popular by cloud software-as-a-service (SaaS) providers such as Salesforce. In practice, annual support subscriptions of commercial open source software providers are akin to the maintenance fees charged by traditional proprietary solution providers.

The emergence of open source is one of the factors that have helped lower pricing expectations for software. With IT budgets of most organizations being tight, enterprises looking for new data platforms or analytic tools will not be willing to pay the price associated with legacy software, unless the perceived unique value is obvious.

Vendors that deliver packaged, integrated, and supported products incorporating open source software should deliver the reliability, security, and quality that enterprises expect from open source software – as long as the open source software projects are sufficiently mature. As noted below, the savings equation changes when enterprises adopt do-it-yourself approaches to supporting and integrating disparate open source projects.

The real costs

When it comes to the real costs of "free software," enterprises should examine the development, support, and maintenance costs of homegrown development, as well as the risks and costs associated with ensuring security, the need for extended functionality, and obsolescence.

Development, support, and maintenance costs

Enterprises implementing open source software projects on their own are taking on the same tasks as custom developing homegrown applications and/or assuming the burdens of integrating them. That entails deploying, configuring or customizing, patching, maintaining, and integrating the open source components with other software.

Part of the issue is the maturity of the technology; admittedly, this is a factor that is also relevant to software that is not open source. But with the rapidity by which open source projects are emerging, this is a factor that must be considered.

The other part of the issue is endemic to choosing a best-of-breed strategy where the customer selects multiple products and then either assumes the burden of integration or maintenance, or retains integrators or consultants to handle it. While this cost is not necessarily unique to open source projects, the fact that many projects lack commercial vendor support compounds the challenge for customers, especially when they need to scale and the complexity increases significantly. Even where commercial support is available, if the open source product or capability is not broad enough, paying for multiple support contracts while ensuring that there is support for the integrations may be cost-prohibitive compared to obtaining comprehensive support from a single provider.

Security

Integrating an open source stack brings with it the need to harmonize security; while open source data platforms such as Hadoop may have their own frameworks for managing security across most (or hopefully all) components, related components that operate on Hadoop such as Spark, or streaming engines such as Storm or Flink, may have different security models.

Obsolescence

An ongoing concern for IT organizations is getting left behind when it comes to software support. This can be an issue at both ends of the lifecycle: at the beginning, when the software is evolving rapidly, and at the end of life, when the software is ready for retirement. And as many open source projects enter their second generation, APIs are often changed; this occurred in the transition from Spark 1.x to 2.x for features such as structured streaming, and also for Kafka's Consumer API when it transitioned from the 0.8 version to the 0.9 version.

Admittedly, the issue of changing APIs/interfaces is not exclusively a problem with open source; however, when an open source project is either vendor-led, or the vendor predominates the contributors in a community-driven project, customers can be especially vulnerable.

Extensibility

Organizations assuming the task of deploying and integrating open source technologies may also have to extend the software with additional functionality outside the bounds of the open source project. Several real-life examples for these cost and risk areas are described in the next section.

Case studies

Homegrown integration

A major US bank built a cybersecurity solution, relying on several open source projects to develop a prototype, including NiFi (for managing and routing data flow), Storm (for streaming data), Metron (security analytics framework), Kibana (for visualizing log data analytics), AlertUI (a GUI designed for alerts), ElastAlert (for alerting), with the solution currently requiring a staff of 45 engineers to manage, integrate, and maintain it. This recurring staff cost amounted to $20–$30m over the course of 5–6 years.

Although (or because) the solution involved multiple open source projects, there were gaps in delivering end-to-end security that the development team had to fill. It should not be surprising that delivering support to multiple open source projects that were integrated through internal development requires significant overhead. When multiple software components are integrated, who takes responsibility? Admittedly, this is an age-old problem whenever solutions are built through custom integrations; with open source, the issue is compounded if the projects were implemented *without* subscribing to vendor commercial support, or if the integrations go beyond what the vendor explicitly supports. This is frequently true for projects backed by specific vendors (e.g., NiFi, Storm, Elasticsearch) where vendor support leaves off *before* the integration abstraction layer is built.

Some of the costs and challenges are related to the need for highly skilled practitioners with popular new technologies or open source projects. In this case, the bank struggled with staff training and retention because of the nature of the skills required, and the high demand for them.

Extended functionality

A large communications service provider built a call center application that involves collecting call detail records (CDRs). It combined three open source projects, including Flume (for routing large flows of data in motion), Logstash (for collecting and transforming log data), and Elasticsearch (for search). It required a team of five people to maintain this solution. Even with the investment of labor, the solution fell short when it came to integrating change-data-capture (CDC) capabilities from customer databases, requiring the team to perform additional development and maintenance work.

Unplanned obsolescence

A credit card processor that sought to upgrade its business intelligence application for transactions to a real-time solution used a combination of open source and proprietary technologies. Although the project was rapidly implemented, involving a team of 3–4 people over two months, the solution quickly grew obsolete. A key component – building data pipelines – was based on Spring XD, a project that entered end of life last year when it was superseded by the microservices-based Spring Cloud Data Flow project. The vendor that led the Spring XD project subsequently deprecated support to bug fixes.

Striim's blended approach

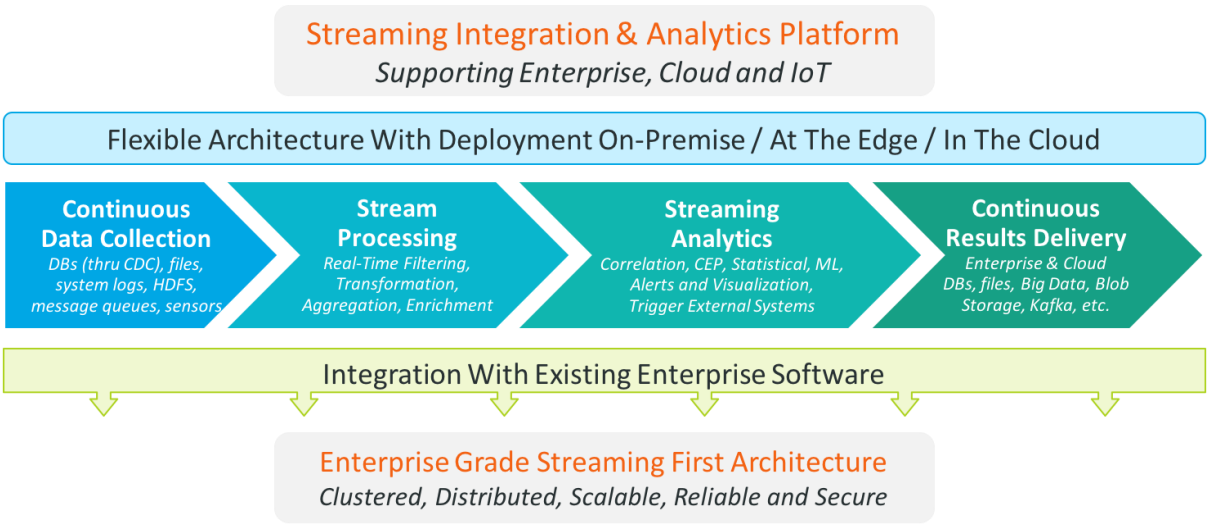
Leveraging and extending open source

Seeking to deliver a broad solution that helps customers avoid the problems of integrating separate best-of-breed projects, Striim embraces the hybrid approach, which blends open source with unique and patented IP. Striim's solution is an end-to-end software platform for building real-time data pipelines and delivering streaming analytics based on SQL queries, with real-time data visualization included. Founded by veterans who helped design Oracle's GoldenGate real-time data replication platform, Striim captures, enriches, transforms, analyzes, and visualizes data in motion before delivering it to various targets in subseconds.

Striim is used for various business-critical, real-time data integration projects; IoT edge processing; and analytical applications such as real-time information security event monitoring, fraud detection, and SLA monitoring. Leveraging open source, Striim embeds the Java version of ZeroMQ as the

distributed messaging backbone, and Kafka for use cases requiring persistent messaging. It natively integrates with Hadoop and popular NoSQL platforms such as HBase, HDFS, Hive, MongoDB, MapR Database, and MapR Streams. By embracing standards and core technologies – including SQL, Kafka, and Elasticsearch – Striim is in line with companies that want to scale with open source rather than fully proprietary software.

Figure 1: Striim's end-to-end solution



Source: Striim

Addressing costs and risks of open source

An end-to-end solution for real-time analytics of data pipelines, like Striim, would require up to 15 separate open source projects (each of them being of varying maturity) and new code to fill in critical gaps. A few examples include data collection, with different projects for different sources such as logs and file systems; distributed caching; indexed results stores; cluster management; and others.

Admittedly, some of this is attributable to any best-of-breed strategy where the IT organization assumes the burden of integrating disparate products or technology components. But the narrow nature of many (although not all) open source projects exacerbates this problem, especially if commercial support is not available (and even then, vendors may not fully support all the integrations). By baking in and fully supporting the integrations among all components of its hybrid solution, Striim enables organizations to save on development, support, and maintenance costs.

Beyond integrating the core building blocks, Striim adds value with the last mile of connective tissue that encompasses distributed SQL-based query processing supporting correlation and complex event processing; maintenance of a common security and fault-tolerance model; simple UI for development; and support for extensibility that allows teams to write their own Java programs (i.e., enhancing Striim analytics with machine-learning capabilities).

For example, a security software solutions provider for the telecommunications and media industry chose Striim for these reasons after spending almost a year evaluating open source. The company relied on Striim's log-based CDC feature for Oracle, a capability lacking from open source. And, thanks to Striim's ease of use, the company required only two developers to build streaming analytics and visualizations, and delivered the solution within two months.

Among the capabilities that Striim delivers that would be difficult, if not impossible, from open source include:

- *CDC from databases, including support for schema evolution* – This feature goes beyond the reading of database logs by addressing the reliability, performance, and security requirements of change data from ingestion to processing to delivery. It can detect changes in the underlying schema, a feature that has not been replicated in the open source world. Other CDC-related challenges that are addressed include failure handling, maintaining transactional boundaries, batch optimization, and data delivery validation.
- *Distributed stream processing, cache integration, and CEP (complex event processing)* – This typically requires multiple tools including a query engine, complex event processing library, distributed cache, and additional code to support windowing and correlation. While there are numerous third-party open source distributed caching technologies, relying on them would require request/response actions that would add latency. We estimate that integrating all the pieces together would require 5–10 staff-years of development time alone.
- *Fault-tolerant, exactly-once processing* – While several open source streaming and message-queuing engines support exactly-once processing, they typically lack the ability to rebuild transaction state that is essential for rollback and replay capabilities.
- *Flow Designer and Dashboard Builder* – Open source rarely, if ever, focuses on ease of use. Since it is developer-driven, open source (almost by definition) results in frameworks, libraries, and platforms that have the developer, not the end user, in mind. Enterprises that are building general-purpose frameworks to combine and abstract multiple pieces of open source into a platform similarly will not spend the time and effort necessary to build out UIs to make building data flows and visualizations easy. Instead, they often use developers or third-party solutions that provide the visual front ends lacking from the original open source project.

Takeaways

Open source is best suited for commodity technologies as opposed to differentiated enterprise solutions. Open source has become a fact of life with big data platforms. The community-led model has had great success delivering general-purpose, commodity technologies that scale, as well as serving large markets and drawing interest from wide communities of developers. For most enterprises, the appeal of open source is not necessarily the opportunity to actively contribute to the code, but instead to gain visibility to the technology roadmap.

But for enterprise solutions, the picture is different. Open source is not well-suited for enterprise applications that support differentiated business processes. It is also not suited for "the last mile," ensuring reliability, integrating with related tools or systems, presenting polished user interfaces, and delivering consistent levels of service. For enterprises, the software might be free, but if the open source projects are narrow, lack commercial vendor support, and require integration with other best-of-breed components, the costs of integration and long-term maintenance will be significant. When selecting enterprise software, enterprises should factor the true costs of free software.

Figure 2. Hybrid Open Source

Not surprisingly, the most practical strategy for enterprises and vendors is a hybrid approach that blends open source foundational technologies with differentiated vendor IP that provides the value-add that would otherwise be too narrow or specific for open source communities to support.

Striim provides an example of the hybrid approach. With a solution spanning from upstream integration and connectivity to delivering a visual, SQL-like query environment under a common model for security and fault tolerance, Striim helps enterprises avoid the time and costs of having to select and integrate the individual pieces themselves. By blending open source with unique IP, Striim enables customers to focus their time and effort on building applications that deliver business value, rather than having to put the pieces together.

Appendix

Author

Tony Baer, Principal Analyst, Information Management

tony.baer@ovum.com

Ovum Consulting

We hope that this analysis will help you make informed and imaginative business decisions. If you have further requirements, Ovum's consulting team may be able to help you. For more information about Ovum's consulting capabilities, please contact us directly at consulting@ovum.com.

Copyright notice and disclaimer

CONTACT US

www.ovum.com
analystsupport@ovum.com

INTERNATIONAL OFFICES

Beijing
Dubai
Hong Kong
Hyderabad
Johannesburg
London
Melbourne
New York
San Francisco
Sao Paulo
Tokyo