Faster, Larger, Easier: Reining Real-Time Big Data Processing in Cloud

Chengwei Wang
Georgia Institute of Technology

Infantdani Abel Rayan
Riot Games, Apache Flume

Karsten Schwan
Georgia Institute of Technology

ABSTRACT

Flume [1] is a widely used open-source real-time data processing framework. We have proposed, in a Middleware'12 research track paper [2], a middleware named VScope for troubleshooting complex big data systems like Flume. This poster introduces the recent evolution from Flume Old Generation (OG) to Flume New Generation (NG) [4], and research improvements in VScope to address the system management challenges.

General Terms

Management, Performance, Design, Reliability.

Keywords

Cloud, Big Data, Streaming, Data Center, Troubleshooting.

1. PROBLEM STATEMENT

1.1 System Design Challenges from Flume OG

Flume is a widely used open source framework in industry for real-time big data processing. Table 1 is a list of companies who have publicly mentioned its usage [3]. The Flume's code base reached a point, where the original design had to be changed fundamentally to achieve further optimizations and avoid further extraneous code. The work begun last year and is in progress. The major goals are to (1) reduce code complexity, (2) core component lifecycle standardization, (3) drastic simplification of common data paths, heartbeat and (4) master re-architecture [4].

1.2 System Management Challenges

Flume's highly adaptable software architecture that can be seamlessly integrated with Hadoop HBase, HDFS and MapReduce, along with the ongoing code base evolution, makes deploying, operating and scaling real-time big data processing in data center environment much easier than ever before. However it also brings following challenges to system management:

Scale: It is not unusual for a data center to have thousands of servers. Large web companies' data centers may have over 1 million servers. With large volumes of monitoring/tracing data, troubleshooting, then, is much like finding a needle in a haystack.

Complexity: Applications commonly consist of distributed software components deployed on different machines or even different data center sites. Components may come from different software vendors or open source developers. Component interactions are complex, not only due to scale, but also because they use built-in resilience methods like those based on replication, quorums, automatic restart, and many others. Different 'teams' of developers may be responsible for the different services or 'tiers' of SOA-based applications, implying the involvement of multiple such teams when something goes wrong with any service or the machine on which it runs. In public clouds like Amazon EC2 or Google App engine, the level of complexity is increased further [5], because applications come from different customers with countless implementation logics, and typically, data center operators have little knowledge about them, one reason being to protect customer privacy.

Company	Description
IPinYou	A RTB platform for 450 million people.
Lijit Networks	Advertising services, audience analytics & reader engagement to 125K+ sites
Meebo	A multi-platform instant messaging engine
Path	2 million+ people share life through Path.
Sematext Intl.	Big Data Analytics products and services.
Sharethrough	Works with Fortune 1000 brands to drive views and maximize social engagement
SimpleGeo	Push Notifications, Rich Media Messaging,
SimpleReach	Tracks every social action to deliver insights and clear metrics around social behavior.
Syoncloud	Discovery in large unstructured data.
ViaSat	fast, secure communications anywhere.

Table 1: Companies Adopting Flume Technology

Dynamism: A data center is a shared infrastructure, with frequently changing users, and with applications frequently installed/deployed and removed. Workloads vary in time, temporally, and across data center nodes, spatially, exacerbated by their aforementioned resilience methods.

2. PROPOSED SOLUTIONS

2.1 Flume NG

Flume NG's high level architecture hardens few concepts from the OG version and simplifies others. It retains Flume OG's general approach to data transfer and handling which is a *N:M* push model where *N* is big and *M* is significantly smaller. *Sources* and *sinks* still exist but are now connected by channels which are pluggable and dictate durability. It also ships with an in-memory channel for fast, but non-durable event delivery and a JDBC-based channel for durable event delivery. The concept of 'logical node' from Flume OG is dropped in Flume NG. Every physical node is now referred as 'agent' which can run zero or more *sources* and *sinks.* Also, the concept of central 'Flume master' is dropped in Flume NG. Instead it runs with a simple file-based configuration system and hence there is no more dependency on Zookeeper (which was a separate external co-ordination service). There are good interfaces for developing plug-ins useful for end-user facing, tools and system developers.

2.2 VScope Improvements

Statistical Guidance: We are leveraging statistical tools [6,7,8,9] and Principle Component Analysis (PCA) to diagnose high dimensional monitoring data. PCA can help narrow down the set of metrics which mostly contribute to the abnormal behavior of the application and provide cues to root causes.

DPG Optimization: Based on our previous work [10], we are designing a deployment engine which is able to deploy Distributed Processing Graphs (DPG) optimal in cost and performance. For instance, the engine can reuse existing *VNodes* to construct new DPGs or it can use the cost-effectiveness model proposed in [10] to decide the best DPG topology.

3. PRELIMINARY RESULTS

3.1 Flume NG Performance

Various performance tests were carried out on Flume NG by the community [11]. As shown in Figure 1, Flume achieves ~70000 events per second on a single machine with no data loss. The optimal number of parallel flows is nearly achieved by creating one flow per CPU core. Additional flows may be added with marginal benefit, likely up to twice the number of physical cores available in the system, if hyper-threading is available. [12]

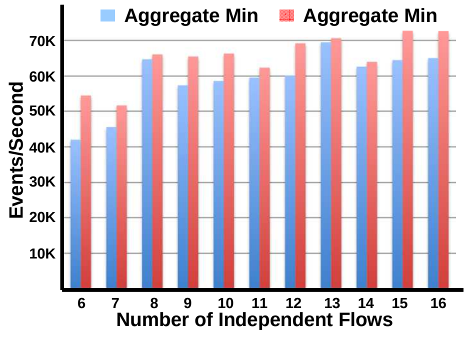

Figure 1: Flume NG Performance [12]

3.2 VScope in Real World Application

We are conducting experiments on real data center logs in Amazon WebStore [13]. We implements statistical guidance in VScope with MapReduce, R and machine learning techniques with reduced metric number and improved interpretability.

4. RELATED WORK

Stream/Event processing is currently a very important paradigm in big data analysis. There are open-source frameworks similar to Flume: Apache Kafka [14], Scribe [15], Honu [16], Chukwa [17], Storm [18], and Apache S4 [19]. Some qualitative comparison discussions can be found at Quora[20], as well.

Apache S4 is primarily developed by Yahoo!. It has been deployed in production systems at Yahoo! to process thousands of search queries per second. LinkedIn is the primary contributor for Apache Kafka. It is used at LinkedIn for activity stream data and operational metrics which power various products like LinkedIn Newsfeed, LinkedIn Today, in addition to their offline analytics systems like Hadoop. Performance results by Kafka community show that one can push about 50MB/sec to the system and can consume about 100M/sec from a broker. Scribe is primarily developed by Facebook. The installation at Facebook runs on thousands of machines and reliably delivers tens of billions of messages a day. Honu is developed at Netflix and is currently processing over 14 billion events/day at Netflix. Chukwa was developed by UC Berkeley and Yahoo!. It was used at Netflix parsing 0.6 TB of logs per day. Storm was developed by BackType which was later acquired by Twitter. It powers Twitter's publisher analytics product, processing every tweet and click that happens on Twitter to provide analytics for Twitter's publisher partners. Storm integrates with Twitter's infrastructure including Cassandra, Kestrel, and Mesos. Many other projects are underway using Storm, including projects in the areas of revenue optimization, anti-spam, and content discovery.

5. REFERENCES

[1] Apache Flume: http://flume.apache.org/

[2] C. Wang, I. A. Rayan, G. Eisenhauer, K. Schwan, V. Talwar, M. Wolf and C. Huneycutt. VScope: Middleware for Troubleshooting Time-Sensitive Data Center Applications. In Middleware 2012, 2012.

[3] Apache. Flume. Public List of Companies Using Flume. https://cwiki.apache.org/FLUME/powered-by.html

[4] Flume NG. https://cwiki.apache.org/FLUME/flume-ng.html.

[5] L. Hu, K. Schwan, A. Gulati, J. Zhang, and C. Wang. Net-cohort: Detecting and Managing VM Ensembles in Virtualized Data Centers. In ICAC '12, 2012.

[6] K. Viswanathan, L. Choudur, V. Talwar, C. Wang, G. MacDonald, and W. Satterfield. Ranking Anomalies in Data Centers. In NOMS'12, 2012.

[7] C.Wang. EbAT: Online Methods for Detecting Utility Cloud Anomalies. In MDS '09, 2009.

[8] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan. Online Detection of Utility Cloud Anomalies Using Metric Distributions. In NOMS'10, 2010.

[9] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical Techniques for Online Anomaly Detection in Data Centers. In IM'11, 2011

[10] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf. A Flexible Architecture Integrating Monitoring and Analytics for Managing Large-Scale Data Centers. In ICAC '11, 2011.

[11] Flume Team https://people.apache.org/~mpercy/flume/flume-1.2.0-incubating-SNAPSHOT/docs/team-list.html.

[12] Flume Performance. https://cwiki.apache.org/FLUME/flume-ng-performance-measurements.html.

[13] Amazon WebStore. http://webstore.amazon.com/

[14] Apache Kafka. http://incubator.apache.org/kafka/design.html.

[15] Facebook. Scribe. https://github.com/facebook/scribe/wiki.

[16] Netflix. Honu. https://github.com/jboulon/Honu/wiki.

[17] Apache Chukwa. http://incubator.apache.org/chukwa/.

[18] Twitter. Storm. http://engineering.twitter.com/2011/08/storm-is-coming-more-details-and-plans.html.

[19] Apache. S4. http://incubator.apache.org/s4/..

[20] http://www.quora.com/What-would-you-choose-between-Flume-Yahoo-S4-and-Backtype-Twitter-Storm-and-why