GobblinProposal

Apache Gobblin Proposal

Abstract

Gobblin is a distributed data integration framework that simplifies common aspects of big data integration such as data ingestion, replication, organization and lifecycle management for both streaming and batch data ecosystems.

Proposal

Gobblin is a universal data integration framework. The framework has been used to build a variety of big data applications such as ingestion, replication, and data retention. The fundamental constructs provided by the Gobblin framework are:

1. An expandable set of connectors that allow data to be integrated from a variety of sources and sinks. The range of connectors already available in Gobblin are quite diverse and are an ever expanding set. To highlight just a few examples, connectors exist for databases (e.g., MySQL, Oracle Teradata, Couchbase etc.), web based technologies (REST APIs, FTP/SFTP servers, Filers), scalable storage (HDFS, S3, Ambry etc,), streaming data (Kafka, EventHubs etc.), and a variety of proprietary data sources and sinks (e.g.Salesforce, Google Analytics, Google Webmaster etc.). Similarly, Gobblin has a rich library of converters that allow for conversion of data from one format to another as data moves across system boundaries (e.g. AVRO in HDFS to JSON in another system).

   2. Gobblin has a well defined and customizable state management layer that allows writing stateful applications. These are particularly useful when solving problems like bulk incremental ingest and keeping several clusters replicated in sync. The ability to record work that has been completed and what remains in a scalable manner is critical to writing such diverse applications successfully.

   3. Gobblin is agnostic to the underlying execution engine. It can be tailored to run ontop of a variety of execution frameworks ranging from multiple processes on a single node, to open source execution engines like MapReduce, Spark or Samza, natively on top of raw containers like Yarn or Mesos, and the public cloud like Amazon AWS or Microsoft Azure. We are extending Gobblin to run on top of a self managed cluster when security is vital. This allows different applications that require different degrees of scalability, latency or security to be customized to for their specific needs. For example, highly latency sensitive applications can be executed in a streaming environment while batch based execution might benefit applications where the priority might be geared towards optimal container utilization.
   4.Gobblin comes out of the box with several diagnosability features like Gobblin metrics and error handling. Collectively, these features allow Gobblin to operate at the scale of petabytes of data. To give just one example, the ability to quarantine a few bad records from an isolated Kafka topic without stopping the entire flow from continued execution is vital when the number of Kafka topics range in the thousands and the collective data handled is in the petabytes.

Gobblin thus provides crisply defined software constructs that can be used to build a vast array of data integration applications customizable for varied user needs. It has become a preferred technology for data integration use-cases by many organizations worldwide (see a partial list here).

Background

Over the last decade, data integration has evolved use case by use case in most companies. For example, at LinkedIn, when Kafka became a significant part of the data ecosystem, a system called Camus was built to ingest this data for analytics processing on Hadoop. Similarly, we had custom pipelines to ingest data from Salesforce, Oracle and myriad other sources. This pattern became the norm rather than the exception and one point, LinkedIn was running at least fifteen different types of ingestion pipelines. This fragmentation has several unfortunate implications. Operational costs scale with the number of pipelines even if the myriad pipelines share a vasty array of common features. Bug fixes and performance optimizations cannot be shared across the pipelines. A common set of practices around debugging and deployment does not emerge. Each pipeline operator will continue to invest in his little silo of the data integration world completely oblivious to the challenges of his fellow operator sitting five tables down.

These experiences were the genesis behind the design and implementation of Gobblin. Gobblin thus started out as a universal data ingestion framework focussed on extracting, transforming, and synchronizing large volumes of data between different data sources and sinks. Not surprisingly, given its origins, the initial design of Gobblin placed great emphasis on abstractions that can be leveraged repeatedly. These abstractions have stood the test of time at LinkedIn and we have been able to leverage the constructs well beyond ingest. Gobblin's architecture has allowed us at LinkedIn to use it for a variety of applications ranging from from optimal format conversion to adhering to compliance policies set by European standards. Finally, as noted earlier, Gobblin can be deployed in a variety of execution environments: it can be deployed as a library embedded in another application or can be used to execute jobs on a public cloud. A fluid architectural and execution design story has allowed Gobblin to become a truly successful data integration platform.

Gobblin has continued to evolve with a variety of utility packages like Gobblin metrics and Gobblin config management. Collectively, these allow organizations utilizing Gobblin to use a system that has been battle tested at LinkedIn scale. This is something that its consumers have to come to appreciate greatly.

Rationale

Gobblin's entry to the Apache foundation is beneficial to both the Gobblin and the Apache communities. Gobblin has greatly benefited from its open source roots. Its community and adoption has grown greatly as a result. More importantly, the feedback from the community whether through interactions at meetups or through the mailing list have allowed for a rich exchange of ideas. In order to grow up the Gobblin community and improve the project, we would like to propose Gobblin to the Apache incubator. The Gobblin community will greatly benefit from the established development and consensus processes that have worked well for other projects. The Apache process has served many other open source projects well and we believe that the Gobblin community will greatly benefit from these practices as well.

Initial Goals

Migrate the existing codebase to Apache Study and Integrate with the Apache development process Ensure all dependencies are compliant with Apache License version 2.0 Incremental development and releases per Apache guidelines Improve the relationship between Gobblin and other Apache projects

Current Status

Gobblin has undergone five major releases (0.5, 0.6, 0.7, 0.8, 0.9) and many minor ones. The latest version, Gobblin 0.9 has just been released in December, 2016. Gobblin is being used in production by over 20 organizations. Gobblin codebase is currently hosted at github.com, which will seed the Apache git repository.

Meritocracy

We plan to invest in supporting a meritocracy. We will discuss the requirements in an open forum. Several companies have already expressed interest in this project, and we intend to invite additional developers to participate. We will encourage and monitor community participation so that privileges can be extended to those that contribute.

Community

The need for a extensible and flexible data integration platform in the open source is tremendous. Gobblin is currently being used by at least 20 organizations worldwide (some examples are listed here). By bringing Gobblin into Apache, we believe that the community will grow even bigger.

Core Developers

Gobblin was started by engineers at LinkedIn, and now has developers from Google, Facebook, LinkedIn, Cloudera, Nerdwallet, Swisscom, and many other companies.

Alignment

Gobblin aligns exceedingly well with the Apache ecosystem. Gobblin is built leveraging several existing Apache projects (Apache Helix, Yarn, Zookeeper etc.). As Gobblin matures, we expect to leverage several other Apache projects further. This leverage invariably results in contributions back to these projects (e.g., a contribution to Helix was made during the Gobblin Yarn development). Finally, being an integration platform, it serves as a bridge between several Apache projects like Apache Hadoop and Apache Kafka. This integration is highly desired and their interaction through Gobblin will lead to a virtuous cycle of greater adoption and newer features in these projects. Thus, we believe that it will be a nice addition to the current set of big data projects under the auspices of the Apache foundation.

Known Risks

Orphaned Products

The risk of the Gobblin project being abandoned is minimal. As noted earlier, there are many organizations that have already invested in Gobblin significantly and are thus incentivized to continue development. Many of these organizations operate critical data ingest, compliance and retention pipelines built with Gobblin and are thus heavily invested in the continued success of Gobblin.

Inexperience with Open Source

Gobblin has existed as a healthy open source project for several years. During that time, we have curated an open-source community successfully. Any risks that we foresee are ones associated with scaling our open source communication and operation process rather than with inherent inexperience in operating an open source project.

Homogenous Developers

Gobblin's committers are employed by companies of varying sizes and industry. Committers come from well heeled internet companies like Google, LinkedI and Facebook. We also have developers from traditional enterprise companies like SwissCom. Well funded startups like Nerdwallet are active in the community of developers. We plan to double our efforts in cultivating a diverse set of committers for Gobblin.

Reliance on Salaried Developers

It is expected that Gobblin development will occur on both salaried time and on volunteer time, after hours. The majority of initial committers are paid by their employer to contribute to this project. However, they are all passionate about the project, and we are confident that the project will continue even if no salaried developers contribute to the project. We are committed to recruiting additional committers including non-salaried developers.

Relationships with Other Apache Products

As noted earlier, Gobblin leverages several open source projects and contributes back to them. There is also overlap with aspects of other Apache projects that we will discuss briefly here. Apache Nifi, like Gobblin aspires to reduce the operational overhead arising from data heterogeneity. Apache Nifi is structured as a visual flow based approach and provides built-in constructs for buffering data, prioritizing data, and understanding data lineage as data flows across systems. Apache Nifi has its own dataflow based execution engine with buffering, scheduling and streaming capabilities. Apache Falcon is a Hadoop centric data governance engine for defining, scheduling, and monitoring data management policies through flow definition typically for data that has been ingested into Hadoop already. Apache Falcon generally delegates data management jobs to tools that already exist in the Hadoop ecosystem (e. g. Distcp, Sqoop, Hive etc). Apache Sqoop is primarily geared for bulk ingest especially from databases which is one part of Gobblin's feature set. Apache Flume focuses primarily on streaming data movement. Finally, general purpose data processing engines like Apache Flink, Apache Samza, and Apache Spark focus on generic computation.

Gobblin design choices intersect with specific features in all of these systems, however in aggregate, it is a different point in the design space. It is designed to handle both streaming and batch data. It supports execution through a standalone cluster mode as well as through existing frameworks such as MR, Yarn, Hive, Samza etc allowing users to choose the deployment model that is optimal for the specific data integration challenge. It provides native optimized implementations for critical integrations such as Kafka, Hadoop - Hadoop copies etc. Gobblin also supports both Hadoop and non-Hadoop data, being able to ingest data into Kafka as well as other key-value stores like Couchbase. Gobblin is also not just a generic computation framework, it has specific constructs for data integration patterns such as data quality metrics and policies. Gobblin's configuration management system allows it to be fully multi-tenant and take advantage of grouped policies when required. For batch workloads, Gobblin has a planning phase that provides for better resource utilization.

In summary, there is healthy diversity in the number of systems approaching the interesting and pressing problem of big data integration. We believe that Gobblin will provide another compelling choice in that design space.

An Excessive Fascination with the Apache Brand

Gobblin is already a healthy and well known open source project. This proposal is not for the purpose of generating publicity. Rather, the primary benefits to joining Apache are already outlined in the Rationale section.

Documentation

The reader will find these websites highly relevant:

- Website: http://linkedin.github.io/gobblin/
- Documentation: https://gobblin.readthedocs.io/en/latest/
- Codebase: https://github.com/linkedin/gobblin/
- User group: https://groups.google.com/forum/#!forum/gobblin-users

Source and Intellectual Property Submission Plan

The Gobblin codebase is currently hosted on Github. This is the exact codebase that we would migrate to the Apache foundation.The Gobblin source code is already licensed under Apache License Version 2.0. Going forward, we will continue to have all the contributions licensed directly to the Apache foundation through our signed Individual Contributor License Agreements for all the committers on the project.

External Dependencies

To the best of our knowledge, all of Gobblin dependencies are distributed under Apache compatible licenses. Upon acceptance to the incubator, we would begin a thorough analysis of all transitive dependencies to verify this fact and introduce license checking into the build and release process (for instance integrating Apache Rat).

Cryptography

We do not expect Gobblin to be a controlled export item due to the use of encryption.

Required Resources

Mailing lists

- gobblin-user
- gobblin-dev
- gobblin-commits
- gobblin-private for private PMC discussions (with moderated subscriptions)

Subversion Directory

Git is the preferred source control system: git://git.apache.org/gobblin

Issue Tracking

JIRA Gobblin (GOBBLIN)

Other Resources

The existing code already has unit and integration tests, so we would like a Jenkins instance to run them whenever a new patch is submitted. This can be added after project creation.

Initial Committers

- Abhishek Tiwari <abhishektiwari dot btech at gmail dot com>
- Shirshanka Das <shirshanka at apache dot org>
- Chavdar Botev <cbotev at gmail dot com>
- Sahil Takiar <takiar.sahil at gmail dot com>
- Yinan Li <liyinan926 at gmail dot com>
- Ziyang Liu <aat z dot flights>
- Lorand Bendig <lbendig at gmail dot com>
- Issac Buenrostro <ibuenros at linkedin dot com>
- Hung Tran <hutran at linkedin dot com>
- Olivier Lamy <olamy at apache dot org>
- Jean-Baptiste Onofré <jbonofre at apache dot org>

Affiliations

- Abhishek Tiwari - LinkedIn
- Shirshanka Das - LinkedIn
- Chavdar Botev - Stealth Startup
- Sahil Takiar - Cloudera
- Yinan Li - Google
- Ziyang Liu - Facebook
- Lorand Bendig - Swisscom
- Issac Buenrostro - LinkedIn
- Hung Tran - LinkedIn
- Olivier Lamy - Webtide
- Jean-Baptiste Onofre - Talend

Sponsors

Champion

Olivier Lamy < olamy at apache dot org>

Nominated Mentors

- Olivier Lamy <olamy at apache dot org>
- Jean-Baptiste Onofre <jbonofre at apache dot org>
- Jim Jagielski <jim at apache dot org>

Sponsoring Entity

The Apache Incubator