Pravega
Reliable streaming storage

Pravega Release Notes – 0.6.0

Table of Contents

Pravega in a nutshell

Pravega is a software defined streaming storage system that enables applications to write streams of events or bytes in an append-only manner. It offers several features for stream applications, including the ability to auto-scale streams, provide transaction semantics, and even build replicated state machines via a state synchronizer.

Pravega has three core components:

- **Controller**: manages the lifecycle of streams and transactions
- **Segment store**: implements the functionality in the critical path of reads and writes
- **Client**: library for applications that use Pravega

Pravega is an open-source system released under the Apache 2.0 License. The source code is available in a GitHub repository: http://github.com/pravega/pravega

What's new?

These notes are about the changes introduced in the release version 0.6.0 of Pravega. Here we list the highlights of the release, while we give a detailed list of the changes in the release description on GitHub: https://github.com/pravega/pravega/releases/tag/v0.6.0

Watermarking

Watermarking is the highlight of current release where we add a new capability in Pravega to provide a bound on “time” from the writers to the readers. The readers can make use watermarks to identify their position in stream with respect to time.

The notion of time is left to application. The Pravega Controller Service acts as the central coordination service which receives and consolidates times reported by multiple writers to compute a bound on time and position on the stream. It emits this information in a dedicated Pravega stream segment as watermarks. These watermarks are consumed by reader-groups to receive markers on stream positions with corresponding bounds on time.
PDP: https://github.com/pravega/pravega/wiki/PDP-33:-Watermarking
Github: Issue 3344: Watermarking

Performance Improvements

We have made several performance related improvements chiefly in segment store service and Pravega client. Following classification by service highlights some of the more significant improvements introduced in release 0.6.0.

Pravega Segment Store Service

We have made improvements in various areas of Segment Stores ranging from lazily creating certain remote records and data structures to pre caching table segments and configuring for more aggressive throttling.
GitHub:

- Issue 3832: Lazily creating the Attribute Index only on the first write.
- Issue 3918: Table Segment Pre-Caching
- Issue 4011: Making Commit Backlog Throttler more aggressive
- Issue 4034: Throttling and Cache Management Tweaks
- Issue 4206: Add support for multi ledgers and journals

We have also added configuration knobs to tune performance.

- Issue 3645: Allow direct reads in RocksDB to be configurable.
- Issue 4084: Max Rolling Segment Size
- Issue 4151: Add option to make RocksDB only use memory environment

Pravega Controller Service

Controller to segment store communication is updated to make use of Pravega RawClient which has support for connection pooling. This optimizes the number of connections that are established for communication between controller instances and segment stores and multiplexes and reuses connections to send different control instructions to segment stores for different streams.

- Issue 3621: use RawClient to simplify SegmentHelper implementation

Pravega Client:

Connection pooling was introduced as part of 0.5.0 to improve scalability by multiplexing requests to multiple segments over a single connection. However, we identified that enabling connection pooling could incur performance penalties on writes because of the way we batch events on the clients before sending them over the wire.
As part of 0.6.0 we have made several incremental improvements to write throughput and write latencies. This include increasing the client batch size for throughput-oriented workloads and serializing writers to improve small event (up to 32K size events) throughput rates. Now, the client ensures Epoll transport is used on Linux based systems and NIO is used only as a fallback.

Following are list of issues towards this purpose.

- Issue 3969: Performance Improvement with connection pooling by serializing writers
- Issue 4196: Increase batch size in the client side for throughput-oriented workloads
- Issue 4263: Ensure Epoll transport is used by Pravega client.

Security

Several improvements, updates, and fixes were made to improve security in Pravega.

The significant changes related to authentication and authorization are:

- Delegation tokens used by Segment Stores to authorize data plane operations now expire, and the expiry time is determined based on a configurable time-to-live parameter (controller.auth.accessTokenTtlSeconds).
- Clients automatically obtain new delegation tokens when an existing token is nearing expiry or after it has expired. This feature avoids the need for client applications to handle token expiry and reauthorization.

The significant changes related to SSL/TLS are:

- A Pravega deployment may now have mixed-mode TLS configuration where TLS is disabled for Controller and enabled for Segment Store. This feature allows administrators to deploy a Pravega cluster in such a way that SSL/TLS terminates at a reverse proxy or Kubernetes Ingress for Controllers and at the service itself for Segment Stores.
- Segment Store automatically reloads SSL/TLS context soon after the SSL/TLS certificate is refreshed in the file system or Kubernetes Secrets, thereby making the new certificate effective.
  This feature reduces operational complexity for administrators/operators, by eliminating the need for bouncing Segment Stores when SSL/TLS certificates are rotated. Hot-reload of SSL/TLS certificate is not supported for Controllers as of now. Administrators may choose to terminate SSL/TLS at reverse proxies/Kubernetes Ingress instead, which in turn should reduce operational complexity in that respect.
- Sample TLS material supplied with Pravega, which may be used for development/testing purposes only, has been modified.

- If a client application does not specify a truststore path, the client library now uses the Java/system truststore automatically. This can help reduce the operational complexity at the client application side.

Security documentation was updated to reflect most of the items listed above.

Also, several fixes and updates were made to improve the internal security of Pravega. The major items in this area were:

- Several 3rd party libraries used in Pravega were upgraded
- Code changes were made to address source-code level security bugs

The major issues addressed in this release are:

- Issue 2330: Delegation token expiry
- Issue 4107: Automatic delegation token renewal
- Issue 3669: Update default TLS material
- Issue 3652: Segment store TLS reload upon certificate refresh
- Issue 4157: Support for Divergent TLS configuration for Controller and Segment Store
- Issue 4231: Use system truststore for TLS when client truststore is not specified

Tests

We have tested Pravega extensively:

- We have currently over 2,000 Junit test cases that we exercise on a regular basis.
- We have currently 30 system test cases that we exercise on a regular basis. Our system tests currently run on both Kubernetes and Docker Swarm.
- We have executed longevity workloads on different infrastructure: Google Cloud, Amazon Web Services, and on-premises PKS.
- We have performed a number functional and non-functional tests, such as killing pods.

Published artifacts

GitHub:
https://github.com/pravega/pravega/releases

Docker image:
https://hub.docker.com/r/pravega/pravega

Maven Central:
http://central.maven.org/maven2/io/pravega/