Pravega Release Notes – 0.7.0

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 desirable 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?

The major goal of this release was performance and stability improvements. Core to these changes is a new Cache used by the Segment Store, designed from the ground up to support the type of streaming workloads that Pravega uses. It also contains full support for DellEMC ECS as Tier 2 binding.

For a full set of changes, please refer to our GitHub release.

Streaming Cache

We have replaced RocksDB with a home-grown, block-based memory cache, which improved stability in containerized environments and eliminated a major bottleneck on the ingestion and read paths. A few highlights about the Streaming Cache:

- Index-less, block-based storage for arbitrary-sized cache entries. Metadata overhead is a constant 0.2%.
- Uses direct memory to store all data.
- Can be provided with an explicit upper bound for memory usage and it will never exceed this.
- All cache entries are broken down into equal-sized 4KB blocks which are memory-aligned to the OS page file boundaries.
- Enables copy-free appends to existing cache entries, which significantly minimizes the amount of time required to process small Events.
- Enables copy-free cache entry slicing (reading a sub-range of a cache entry without copying the whole entry into a heap buffer).

Github Issue 4044.

Dell EMC ECS Support

We are happy to announce that Pravega now fully supports DellEMC ECS object storage (ECS for short) as Tier 2 using the open-source Extended S3 Client. This is in addition to other Tier 2 options Pravega supports today (HDFS and Filesystem (NFS)).

In this release we focused on testing ECS support more thoroughly. We found and fixed a few issues. We also changed the way we configure ECS to make it more user friendly and extensible. Pravega now uses the term "prefix" instead of "root" to be more consistent with the Object Storage community and accepts REST style URIs for ECS client side configurations.

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 0.7.0.

Pravega Segment Store Service

- **Streaming Cache.** Please refer to the section above, describing this in detail.
- **Low single segment write performance**. We have solved an issue with calculating our Bookkeeper checksum which improved ingestion performance for a single segment.
- **Optimization of String splits for Metrics**. We are now using a streamlined string tokenizer instead of a regular expression to parse metric tags, which resulted in a significant reduction in tail latencies.

Pravega Controller Service

- Improved handling of transactions.

Pravega Client:

- **Avoid Netty Flush from the Client**. This change minimizes the amount of Netty flush calls issued by the client and avoids blocking on the futures returned from write() unless it is needed
- **Early wake reader in response to server reply**. This change helps unblock reader threads waiting on idle segments in the event that data arrives from the server on another segment.
- **Write event padding more efficiently**. We are now bulk-writing zeroes instead of adding them one by one.

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/tag/v0.7.0

Docker image:

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

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