etcd metrics
etcd 度量指标概览

Jingyi Hu 胡景懿 (Google)
Wenjia Zhang 张文嘉 (Google)
Speakers

• Jingyi Hu 胡景懿 (Google)
  • etcd maintainer, kubernetes member
  • github/jingyih
  • jingyih@google.com

• Wenjia Zhang 张文嘉 (Google)
  • etcd contributor, kubernetes member
  • github/wenjiaswe
  • wenjiazhang@google.com
Agenda

• etcd metrics port
• Documented metrics
• New metrics
• How to analyze etcd metrics
作为一个分布式键值存储，etcd 是 Kubernetes 控制平面中最关键的组件，为集群元数据提供了强大的一致性和持久性。etcd 实施了 Raft 共识算法，以跨多个节点分发数据。所有数据复制都由 Raft 完成。您是否知道，etcd Raft 软件包也被用于许多其他项目？CockroachDB 为其组成部分协议分享 etcd Raft 实施。TiKV 将 etcd Raft 接入 Rust（最初在 Go 中编写），并将其用于实施分布式事务数据库。本演讲将介绍 Raft 共识算法的基础知识、其实施细节以及未来的 Raft 软件包路线图。

Speakers

Jingyi Hu
Software Engineer, Google

Jingyi Hu is a Software Engineer for Google Cloud. He is a maintainer of etcd and an active contributor to Kubernetes.
etcd metrics port

etcd 监控指标接口
Each etcd server exports metrics under the /metrics path on its client port and optionally on locations given by --listen-metrics-urls.

- $ curl -L http://localhost:2379/metrics
- --listen-metrics-url http://localhost:9379
  - $ curl -L http://localhost:9379/metrics
Documented metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>has_leader</td>
<td>Whether or not a leader exists. 1 is existence, 0 is not.</td>
<td>Gauge</td>
</tr>
<tr>
<td>leader_changes_seen_total</td>
<td>The number of leader changes seen.</td>
<td>Counter</td>
</tr>
<tr>
<td>proposals_committed_total</td>
<td>The total number of consensus proposals committed.</td>
<td>Gauge</td>
</tr>
<tr>
<td>proposals_applied_total</td>
<td>The total number of consensus proposals applied.</td>
<td>Gauge</td>
</tr>
<tr>
<td>proposals_pending</td>
<td>The current number of pending proposals.</td>
<td>Gauge</td>
</tr>
<tr>
<td>proposals_failed_total</td>
<td>The total number of failed proposals seen.</td>
<td>Counter</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>wal_fsync_duration_seconds</td>
<td>The latency distributions of fsync called by wal</td>
<td>Histogram</td>
</tr>
<tr>
<td>backend_commit_duration_seconds</td>
<td>The latency distributions of commit called by backend.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>peer_sent_bytes_total</td>
<td>The total number of bytes sent to the peer with ID $\text{To}$.</td>
<td>Counter($\text{To}$)</td>
</tr>
<tr>
<td>peer_received_bytes_total</td>
<td>The total number of bytes received from the peer with ID $\text{From}$.</td>
<td>Counter($\text{From}$)</td>
</tr>
<tr>
<td>peer_sent_failures_total</td>
<td>The total number of send failures from the peer with ID $\text{To}$.</td>
<td>Counter($\text{To}$)</td>
</tr>
<tr>
<td>peer_received_failures_total</td>
<td>The total number of receive failures from the peer with ID $\text{From}$.</td>
<td>Counter($\text{From}$)</td>
</tr>
<tr>
<td>peer_round_trip_time_seconds</td>
<td>Round-Trip-Time histogram between peers.</td>
<td>Histogram($\text{To}$)</td>
</tr>
<tr>
<td>client_grpc_sent_bytes_total</td>
<td>The total number of bytes sent to grpc clients.</td>
<td>Counter</td>
</tr>
<tr>
<td>client_grpc_received_bytes_total</td>
<td>The total number of bytes received to grpc clients.</td>
<td>Counter</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>peer_sent_bytes_total</td>
<td>The total number of bytes sent to the peer with ID <strong>To</strong>.</td>
<td>Counter(To)</td>
</tr>
<tr>
<td>peer_received_bytes_total</td>
<td>The total number of bytes received from the peer with ID <strong>From</strong>.</td>
<td>Counter(From)</td>
</tr>
<tr>
<td>peer_sent_failures_total</td>
<td>The total number of send failures from the peer with ID <strong>To</strong>.</td>
<td>Counter(To)</td>
</tr>
<tr>
<td>peer_received_failures_total</td>
<td>The total number of receive failures from the peer with ID <strong>From</strong>.</td>
<td>Counter(From)</td>
</tr>
<tr>
<td>peer_round_trip_time_seconds</td>
<td>Round-Trip-Time histogram between peers.</td>
<td>Histogram(To)</td>
</tr>
<tr>
<td>client_grpc_sent_bytes_total</td>
<td>The total number of bytes sent to grpc clients.</td>
<td>Counter</td>
</tr>
<tr>
<td>client_grpc_received_bytes_total</td>
<td>The total number of bytes received to grpc clients.</td>
<td>Counter</td>
</tr>
</tbody>
</table>
New metrics
Version related

etcd_cluster_version
etcd_server_version (To replace Kubernetes etcd-version-monitor)
etcd_server_go_version
Snapshot metrics

To Monitor Snapshot Save Operations on local node
etcd_snap_db_fsync_duration_seconds_count
etcd_snap_db_save_total_duration_seconds_bucket
etcd_snap_fsync_duration_seconds

To Monitor Snapshot Operations between remote peers
etcd_network_snapshot_send_success
etcd_network_snapshot_send_failures
etcd_network_snapshot_send_total_duration_seconds
etcd_network_snapshot_receive_success
etcd_network_snapshot_receive_failures
etcd_network_snapshot_receive_total_duration_seconds
etcd_network_active_peers
etcd_network_disconnected_peers_total

/metrics:
etcd_network_active_peers{Local="1",Remote="2"} 1
etcd_network_active_peers{Local="1",Remote="3"} 1

/metrics:
etcd_network_active_peers{Local="1",Remote="2"} 1
etcd_network_active_peers{Local="1",Remote="3"} 0
etcd_network_disconnected_peers_total{Local="1",Remote="3"} 1
etcd_server_quota_backend_bytes
etcd_mvcc_db_total_size_in_bytes
etcd_mvcc_db_total_size_in_use_in_bytes

Storage size limit
(\texttt{--quota-backend-bytes})

DB size in Use

Physically allocated DB size
Database size metrics

etcd_server_quota_backend_bytes
etcd_mvcc_db_total_size_in_bytes
etcd_mvcc_db_total_size_in_use_in_bytes

Can be saved from defragmentation!
Storage layer metrics

etcd_server_heartbeat_send_failures_total
etcd_server_slow_apply_total
etcd_disk_backend_defrag_duration_seconds
etcd_mvcc_hash_duration_seconds
etcd_mvcc_hash_rev_duration_seconds

Indication of possible overloading of slow disk
etcd_server_is_leader
etcd_server_id
etcd_server_health_success
etcd_server_health_failures
etcd_server_read_indexes_failed_total
etcd_server_slow_read_indexes_total
etcd learner metrics

etcd_server_is_learner
etcd_server_learner_promote_failures
etcd_server_learner_promote_successes

Ref:
etcd learner implementation: https://github.com/etcd-io/etcd/pull/10645
gRPC proxy expose endpoint metrics

Metrics and Health

The gRPC proxy exposes `/health` and Prometheus `/metrics` endpoints for the etcd members defined by `--endpoints`. An alternative define an additional URL that will respond to both the `/metrics` and `/health` endpoints with the `--metrics-addr` flag.

```
$ etcd grpc-proxy start \
    --endpoints https://localhost:2379 \ 
    --metrics-addr https://0.0.0.0:4443 \ 
    --listen-addr 127.0.0.1:23790 \ 
    --key client.key \ 
    --key-file proxy-server.key \ 
    --cert client.crt \ 
    --cert-file proxy-server.crt \ 
    --cacert ca.pem \ 
    --trusted-ca-file proxy-ca.pem
```
etcd_debugging_disk_backend_commit_rebalance_duration_seconds
etcd_debugging_disk_backend_commit_spill_duration_seconds
etcd_debugging_disk_backend_commit_write_duration_seconds

Note that any etcd_debugging_* metrics are experimental and subject to change.
etcd leases debugging

etcd_debugging_lease_granted_total
etcd_debugging_lease_revoked_total
etcd_debugging_lease_renewed_total
etcd_debugging_lease_ttl_total

Note that any etcd_debugging_* metrics are experimental and subject to change.
Example of how to use etcd metrics
Warning “Apply entry took too long”

- Request too large
- Slow disk: backend_commit_duration_seconds
- CPU starvation, memory swapping
• Client request timeout

$ ETCDCTL_API=3 etcdctl put foo bar --endpoints “XXX”

Error: context deadline exceeded

• Can cluster make progress:

  etcd_server_has_leader, proposals_failed_total

• Networking: peer_sent_failures_total, peer_round_trip_time_seconds

• Slow apply: etcd_server_slow_apply_total
Thanks!

- Jingyi Hu 胡景懿 (Google)
  - github/jingyiZh, jingyih@google.com
- Wenjia Zhang 张文嘉 (Google)
  - github/wenjiaswe, wenjiazhang@google.com
Speakers

• Jingyi Hu 胡景懿 (Google)
  • etcd maintainer, kubernetes member
  • github/jingyiZh
  • jingyih@google.com

• Wenjia Zhang 张文嘉 (Google)
  • etcd contributor, kubernetes member
  • github/wenjiaswe
  • wenjiazhang@google.com
Figure 1: Replicated state machine architecture. The consensus algorithm manages a replicated log containing state machine commands from clients. The state machines process identical sequences of commands from the logs, so they produce the same outputs.

etcd_network_server_stream_failures_total

The total number of stream failures from the local server.

Example output:

etcd_network_server_stream_failures_total{API="lease-keepalive",Type="receive"} 1
etcd_network_server_stream_failures_total{API="watch",Type="receive"} 1
Fault Tolerance
Consensus latency

Member @Seattle
Member @Barcelona
Member @Shanghai

Tuning heartbeat interval and election timeout setting
- CPU
- Disk
- Networking