MultiTenant

Multi-tenancy with Cassandra

Work is currently being done to support multi-tenant capabilities in Cassandra. This page is meant to list tickets associated with that effort as well as provide some documentation on how multi-tenant clusters can be configured.

Open tickets having to do with multi-tenancy

- Concurrent schema migrations (multiple users changing schema at once) - CASSANDRA-1391
- Some kind of client transport encryption option. thrift+ssl for clients requires some minimal code for each language - THRIFT-106 is for Java, potential avro ticket - AVRO-341
- For multi-tenant order preserved partitioning, auto/online load balancing is pretty important CASSANDRA-1418
- Track per-user/per-keyspace statistics CASSANDRA-841
- Improve load balancing to take into account load in terms of operations - probably add no. of operations performed on top of disk space used CASSANDRA-1037
- Set query/resultset limits per CF to help prevent OOMing nodes CASSANDRA-2068

Completed tickets having to do with multi-tenancy

- Provide relative memory usage settings for Memtables (and eventually caches) out of configurable totals CASSANDRA-2006
- Ability to lock down schemas for Column Families - CASSANDRA-1554
- Better authorization system - CASSANDRA-1271
- Dynamic schema changes - CASSANDRA-44
- Basic scheduling based on keyspace - CASSANDRA-1035
- CASSANDRA-1237 - Store AccessLevels externally to IAuthenticator (better authorization system)
- Implement a weighted scheduler - e.g. for dev/prod accounts on the same cluster CASSANDRA-1485

Ideas under consideration

- Compaction fairness: currently, compaction makes no attempt to balance attention across multiple CFs
- Include memory usage in "load" considerations for load balancing: ColumnFamilies with higher cache requirements have memory to consider.
- Namespaces - in a multi-tenant use case, each user might like to have a keyspace XYZ for whatever reason. So it might be nice to have namespaces so that keyspace XYZ could be specific to their user. Ideally this would be an option that would not affect those that don't use namespaces.
    - The distinction from keyspaces is that a namespace would be completely transparent to the user: the existence of namespaces would not be exposed. It might be returned by the authentication backend on login, and prefixed to keyspaces transparently.
- JMX authentication without breaking nodetool - currently if you try to use JMX authentication, some of the Cassandra tools won't connect properly.
- Perhaps provide a way to throttle certain operations, maybe useful outside the scope of multi-tenancy. Voldemort, for example, uses a throttler for operations that quickly iterate over all the data of a node, such as a rebalance.
- The Hector java client for Cassandra has experimental support for a “Shared Keyspace, Shared Column Families” multi-tenancy model that allows for application-level virtual keyspaces Virtual keyspaces in Hector.

`https://c.statcounter.com/9397521/0/fe557aad/1/|stats`