UseCases

Cassandra Use Cases

This summary of a mailing-list survey briefly describes how several organizations (Rackspace, Cisco, OneSpot, more) are using Cassandra: more detail in this mailing-list thread.

The below gives simple use patterns and example implementations in high-level code.

If you've got more simple examples along the lines of those below, please add them.

Twissandra, a Twitter clone using Cassandra

Available at twissandra.com.

A Simple Capped Log

Please help complete

- Adapt e.g. this redis implementation to Cassandra
- This mailing list thread gives an overview for building a production-grade windowed time-series store in Cassandra.

Inverted Index for Document Search

Please help complete

A distributed Priority Job Queue

Please help complete

Use Cassandra to enqueue jobs with a priority and optional delay. At each request, the broker assigns the ready job with highest priority.

Consistent Vote Counting

From a conversation on the #cassandra IRC channel, here's a way to implement Consistent Vote Counting using Cassandra that doesn't depend on vector clocks or an atomic increment operation.

Uniq a large dataset using simple key-value columns

We have to batch-process a massive dataset with frequent duplicates that we'd like to skip.

Here is ruby code using Cassandra as a simple key-value store to skip duplicates. You can find a real working version in the Wukong example code – it's used to batch process terabyte-scale data on a 30 machine cluster using Hadoop and Cassandra.

```
class CassandraConditionalOutputter
  CASSANDRA_KEYSPACE = 'Foo'

  # Batch parse a raw stream into parsed objects. The parsed objects may have
  # many duplicates which we'd like to reject
  #
  # records respond to #key (only one record for the given key will be output)
  # and #timestamp (which can be say '0' if record has no meaningful timestamp)
  def process raw_records
    raw_records.parse do |record|
      if should_emit?(record)
        track! record
        puts   record
      end
    end
  end

  # Emit if record's key isn't already in the key column
  def should_emit? record
    key_cache.exists?(key_column, record.key)
  end

  # register key in the key_cache
  def track! record
    key_cache.insert(key_column, record.key, 't' => record.timestamp)
  end

  # nuke key from the key_cache
  def remove record
    key_cache.remove(key_column, record.key)
  end

  # The Cassandra keyspace for key lookup
  def key_cache
    @key_cache ||= Cassandra.new(CASSANDRA_KEYSPACE)
  end

  # Name the key column after class
  def key_column
    self.class.to_s+'Keys'
  end
end
```

Simple time-series with roll-ups

Cloudkick implements time-series down at the second-level with roll-ups.

An implementation of some DBMS rules written in python using pycassa

We have created a DBMS layer that handles references to other columnfamilys (foreign keys), Automatic reverse linking. required fields in columnfamilys and datatypes (long and datetime). It wraps the get, get_range, insert, remove functions of pycassas columnfamilys. At this time it is limited to: on delete cascade and positive long numbers but this could change if there is enough interest. It suits our project.

ThomasBoose dbms implementation

Based on this article

ThomasBoose EERD model components to Cassandra Column family's

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