First results of the feasibility study of the Cassandra DB application in Panda Job Monitoring

ATLAS S&C Week at CERN
April 5th, 2011

Maxim Potekhin for :

BNL Physics Applications Software Group
BNL RHIC and ATLAS Computing Facility (M.Ernst, S.Misawa, H.Ito et al)

potekhin@bnl.gov

Looking at noSQL solutions from the Panda perspective

noSQL is not a replacement for the Oracle DB in the core part of the Panda server functionality. In addition to RDBMS vs noSQL comparison presented in Vincent's talk, we observe that:

- There are classes of complex queries which include time and date ranges where Oracle typically performs rather poorly
- Storing lots of historical data on expensive transaction-oriented RDBMS does not seem optimal
- An option to unload significant amounts of archival-type reference data from Oracle to a highly performing, scalable system based on commodity hardware appears attractive

Objectives of the project

Ultimately, we aim to create of a noSQL-based data store for archive-type reference Panda data. In the validation stage, it would continuously function in parallel with the Oracle DB in order to ascertain the performance and characteristics of such system under real life data volume and query loads.

To this end, we are pursuing multiple objectives, in the context of Cassandra DB:

- to create an optimal data design for the data generated by the Panda server as well as indexes
- to perform a broad performance comparison of the recently built cluster at BNL (March 2011) to the existing Oracle facility at CERN
- To determine the optimal hardware configuration of that system and its tuning parameters

Stages of the project

In recent months, we accomplished the following:

- In Jan-Feb 2011, using the 4-VM Cassandra R&D cluster at CERN (generously shared by the DDM team), gained experience in data design and indexing of Panda data, as well as operational experience with Cassandra
- Demonstrated usefulness of map-reduce approach to generation of date range indexes into Cassandra data, which results in marked improvement over Oracle
- In March 2011, with support from RACF, commissioned a 3-node real hardware cluster at BNL and loaded a year worth of real Panda data in a modified format (based on initial CERN experience). Ran a series of performance tests aimed at the “worst-case performance” estimation, to set the baseline for further optimization.

The Cassandra Cluster at BNL

Currently we have 3 nodes, with possible expansion to 5.

A node:
- two Xeon X5650 processors, 6 cores each
- due to hyper-threading – effectively 24 cores
- total of eight 500GB 2.5in, 7200RM SATA disks
- out of 8 spindles, 6 are RAID0 for the Cassandra data
- 48GB RAM on a 1333MHz bus

Such configuration will be highly optimal for write performance, which in Cassandra is CPU and memory bound. What we needed to do was to gauge the read performance under variety of queries and loads.

Loading Data/Write Performance

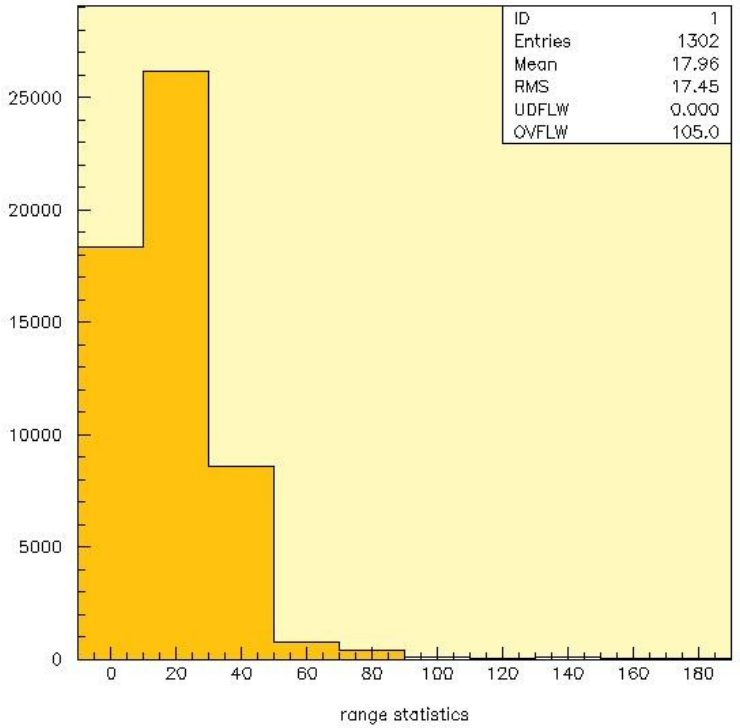
To stress a decent Cassandra installation in write mode, it is necessary to have multiple clients operating simultaneously, and ideally have data pre-staged on local drives on the client nodes.

We created a multithreaded Python application that is capable of loading data either directly from the Amazon S3 data store created by T.Wenaus, or from local data files previously downloaded from the network. The data comes in CSV format and is parsed using a standard Python module.

We observe rates of 500 rows/s on the CERN VM cluster and more than 10000 rows/s on the BNL cluster. *Therefore, network bandwidth of the connection to the external data source will be the limiting factor, not Cassandra performance.*

Patterns in the Data

In Panda, jobs are often submitted not individually, but in batches which can measure anywhere from 2 to hundreds of jobs. This is demonstrated by the following plot which shows the distribution of lengths of continuous ID ranges as they are found in the database



Patterns in the Data

Practically, this means that a lot of queries will be wasteful – when looking at activity of a user on a particular cloud, or his/her job sets, or tasks, we will typically make 20 trips to the database, each extracting a very small amount of data, instead of one read.

In the second generation of the Cassandra schema, we opted to store data in bunched of 100 Panda entries, in order to take advantage of that pattern in the data. We also decided to store entries in csv form, as opposed to pre-parsed columns, in order to conserve space and resources spent on serialization/deserialization of the data in the client and the server.

Main results of testing at CERN

Note on indexing: native ability to index individual columns is a relatively new feature in Cassandra, introduced in mid-2010. Since our data is mostly write-once, read-many, we can do a better job by bucketing data in map-reduce fashion, and we don't need to create extraneous columns to implement an equivalent of composite indexes (so we save space)

Example:
Query for 20110102, borut.kersevan@ijs.si:FR:failed
(extracting all data pertaining to these jobs, not just count)

Takes 5.8 seconds (or 2.7 with hot key cache) on the CERN Cassandra cluster vs 1 min on Oracle RDBMS

Main results of testing at CERN

Using asynchronously built indexes allows to do statistical analysis of the workflow that is not currently done due to poor performance of date range queries. We used “natural” time binning of the data into 1 day bins as they exist in our S3 backup facility as basis for building indexes (i.e. all indexes in this exercise were built “on daily basis”).

Find the number of failed jobs in the NL cloud, for user “borut” over a period of 5 days in December 2010:

- 0.005s on Cassandra
- ~1 min on Oracle

A few indexes, such as user::cloud::jobstatus, cloud::jobstatus etc, only take 1 to 3% of disk space, and that means we still have headroom to compute multiple such indexes to accommodate users’ needs.

Contents of the BNL test

We based our selection of queries to be run against the Cassandra cluster on actual Oracle usage statistics. One caveat in using this is a benchmark that is there are not too many date range queries. We believe there is a bias in that sample due to the fact that users are discouraged from doing such queries both by their own experience and also by policies which aim to reduce the load on the Oracle server (e.g. by limiting the number of days covered by a query done from the Monitor).

We were aiming for getting lower limits on the Cassandra performance, i.e. the worst-case scenario. In most testing, key and row caches were disabled, which puts Cassandra at a disadvantage vs Oracle (whose caching capabilities we were not in the position to control)

Contents of the BNL test

Disclaimer:

this work started recently and is an effort in progress! Tuning Cassandra for a particular application and doing proper data design according to query pattern takes some time.

Main results of testing at BNL

Data load: 1 year up till now (which is practically a majority of data generated so far due to explosive growth in 2010)

Case #1:
Large query load, primary key queries:

Looking at throughput in a random query of 10k items over the past year, vs the number of concurrent clients. Numbers are seconds to query completion. Note fluctuation in Oracle's response from day to day (typical).

	1	2	10	30
Cassandra	361	401	682	1490
Oracle	531	512	400	500

Main results of testing at BNL

Apparently our Cassandra cluster did better at 20Hz rate, was equal at roughly 100Hz, and fell behind at 500.

Comments:

iostat monitoring shows that the disks in the Cassandra setup are saturated with 100% load at 10 clients or so. We have 18 spindles vs 108 at the ADCR Oracle (and these are presumably better drives). Also, we utilized NO caching (on purpose) in Cassabdra vs working cache in Oracle.

Key question, what is the realistic load? Answering this question is work in progress, working with Gancho.

Main results of testing at BNL

Case #2: 52 randomly chosen jobsets for user Johannes Elmsheuser (actual popular query)

	1	30
Cassandra	0.5	2.8
Oracle	1.45	7.05

Cassandra came ahead this time, despite obvious stress conditions for both Oracle and Cassandra. Original hints and bind variables were used in the optimized Oracle queries.

Main results of testing at BNL

Case #3: 1 random job picked from past year, 30 clients

	30
Cassandra	0.051
Oracle	0.071

Case #4: 100 tasks are being queried with 1 client

	1
Cassandra	27
Oracle	50

Summary (1)

We conducted preliminary studies of the feasibility of Cassandra application as the archival database for Panda job status and other data. In addition to VM-based platform at CERN, we built an initial version of a 3-machine Cassandra cluster at BNL. We demonstrated the following:

- Good write performance of the cluster (of the order of 10k entries per second written to the DB) which makes it a good data sink for Oracle or any other data source
- By using techniques similar to map-reduce we can index the data in a way that enables a variety of queries over date ranges, a piece of functionality now under-utilized on Oracle due to performance issues

Summary (2)

- On a 3 node cluster, we observe superior performance in important classes of queries even with caching in Cassandra disabled. In another case, under significant load this particular configuration has a worse overall throughput compared to the Oracle RDBMS deployed at CERN, likely due to disparity in number and quality of spindles between the two systems.
- Based on what we know, *we could scale horizontally*, but in order to make such decision we need *to better understand current and projected query load requirements* (i.e. whether we really need to do this to provide an adequate service). Work in progress.
- So far indications are that Cassandra can indeed serve as archival DB for Panda, which will reduce the load on central Oracle service and provide better performance in a variety of cases.
- We need to define the policy: which clients will access Cassandra storage and how. Will the data be spliced in apps like Panda Monitor?

Acknowledgements

- Thanks to Vincent, Donal and others for nice teamwork and help with the CERN VM-based cluster
- Thanks to RACF team, M.Ernst, S.Misawa, H.Ito and others for installation of a new cluster and continued support of this effort.