Present Tense

Challenges and Trade-offs in Building a Web-scale Real-time Analytics System

Benjamin Black, b@fastip.com
@b6n
Problem

Collect, index, and query trillions of high-dimensionality records with seconds of latency for ingestion and response.
A (Partial) Record

Dimensions

- Src MAC
- Dst MAC
- Src IPv4 Addr
- Dst IPv4 Addr
- Protocol
- Src Transport Port
- Dst Transport Port
- Src IPv4 Lat
- Src IPv4 Long
- Dst IPv4 Lat
- Dst IPv4 Long

Metrics

- Start usecs
- End usecs
- Packets
- Octets
- UID

Dimensions

Metrics
Why Is This Hard?

- All our data is multi-dimensional – and we retain that dimensionality throughout.
- 2-D systems (1 dimension + time) are simpler, but there is significant information loss with the dimensionality loss.
- Aggregation is loss of dimensionality.
First Attempt

› MAXIMIZE SIMPLICITY.
› Insert records into HBase.
› Retrieve all records in time range, then filter, aggregate, sort.

› Similar to OpenTSDB approach.
Apply Load

PREPARE YOUR DATA
First Failure

- All resources consumed, database corrupted, hate gizzards swollen.
- Architecture scales poorly to queries touching billions of records – linear on number of records.
- HBase in late 2009 extremely fragile, especially on low-end hardware.
Second Attempt

- Insert records into Cassandra.
- Random partitioner and indexing to evenly distribute load.
- Index every dimension independently.
- Multi-step retrieval & set operation process to determine all records of interest.
- Finally, aggregate and sort.
Apply Load

PREPARE YOUR DATA
Second Failure

- Write load enormous, query latency agonizing, hate gizzards swollen.

- Repeated, multi-million key fetches per query makes things slow.

- Treating the application as independent of the database just won’t scale.
Third Attempt

- Almost identical to previous...
- **BIG CHANGE**: Perform statistical sampling of records in the database.
Apply Load

PREPARE YOUR DATA
Third Failure

- Write load enormous, query latency only marginally better, results unreliable, hate gizzards swollen.

- Sampling is unstable without sophisticated algorithms.

- Both too slow and surprising. In a bad way.
Pause. Regroup.

Our mental model is obviously broken. What is a better one?

Hit the library: Citeseer.
Epiphany

We have an OLAP problem!

Not just any OLAP problem, but a real-time, high-dimensionality OLAP problem.

Not impossible, just really hard.
What is OLAP?

A business intelligence (BI) approach to "swiftly answer multi-dimensional analytics queries" by structuring the data specifically eliminate expensive processing at query time, even at a cost of enormous storage consumption.
The Cube

OLAP systems rely on pre-computing results, then filtering, sorting, and aggregating to produce results. The artifact of the pre-computation is a hypercube with size proportional to the dimensionality and cardinality of the data.
Dimensionality

We have dozens of dimensions; Materializing the entire cube would require absurd memory – out of the question.
Cardinality

The number of unique values possible for each dimension. We have a lot of these, too – thousands to billions.

Materializing the entire cube would require absurd memory – out of the question.
Strategy

Starting points:

High-Dimensional OLAP: A Minimal Cubing Approach by Li, Han, and Gonzalez

Sorting improves word-aligned bitmap indexes by Lemire, Kaser, and Aouiche
Research

Academic research can point you in a direction, but it is rarely a complete solution. Even more rarely is it a complete solution for your problem.

A useful, even essential, tool, but one to be used with great care.
Fourth Attempt

- Insert records into Cassandra.
- Materialize lower-dimensional cuboids using bitsets, join as needed. *This is a lot harder than it sounds.*
- Perform all query steps directly in the database. *So is this.*
Apply Load

PREPARE YOUR DATA

SUBMIT
Success!

- Low write load.
- Low latency – compact indices mean more of them in memory.
- Distributed cuboids mean distributed load.
- Hate gizzards shrink.
More To Do

- Data-specific indices – network prefix queries, etc.
- On-disk structures specific to our application.
- MORE MAGIC.
Lessons

› READ THE LITERATURE!
› Generic software is 90% wrong at scale, you just don’t know which 90%.
› ITERATE TO DISCOVER.
› BE PREPARED TO START OVER. Repeatedly.
› DON’T GIVE UP!
We’re Hiring

https://fastip.com/jobs