Orient Data Vertically for Faster Analysis

The Combination ViseGrip, Adjustable Spanner, 5lb Ball Peen Hammer

Oct 12 2014

This is close

**Techzonics.com

Agenda

- Column Store Technology
- Available Technologies
  - Infinidb, Infobright, Vertica
- Data Wrangling
  - Name splitting, ip to integer conversions
- Performance
  - Name splitting, ip to integer conversions

Column Store Technology (Thanks Wikipedia*)

Row Orientation

EmpID	Lastname	Firstname	Salary
10	Horton	Tim	100000
12	Lightfoot	Gordon	100001

Typically stored as

001:10,Horton,Tim,10000; 002:12, Lightfoot, Gordon,100001

Column Orientation

10:001,12:002;Horton:001,Lightfoot:002;Tim:001,Gordon:002;100000:001,100001:002

*Based on http://en.wikipedia.org/wiki/Column-oriented_DBMS

Column Store Technology

Drawbacks:

a) Updates (don't bother)
   (update table set columnX=x where columnY=y)

Benefits:

a) Aggregate queries are faster
b) Loading complete (all values for all columns) is faster
c) Compression
d) No indexing*

Available Technologies

- Infinidb
  - GNU License
  - no documented data limits
  - Multi Threaded
  - larger aggregations challenging
- Infobright
  - Commercial license and Community Edition
  - 50TB data limit (CE)
  - Single Threaded (CE)
- Vertica
  - Commercial license and Community Edition
  - 1TB data limit (CE)
  - Multi Threaded
- Others – SAP Hana, IBM DB2 BLU

Data Wrangling

For larger datasets (~3+ B rows)

Split qnames into components

top level domain, second level domain

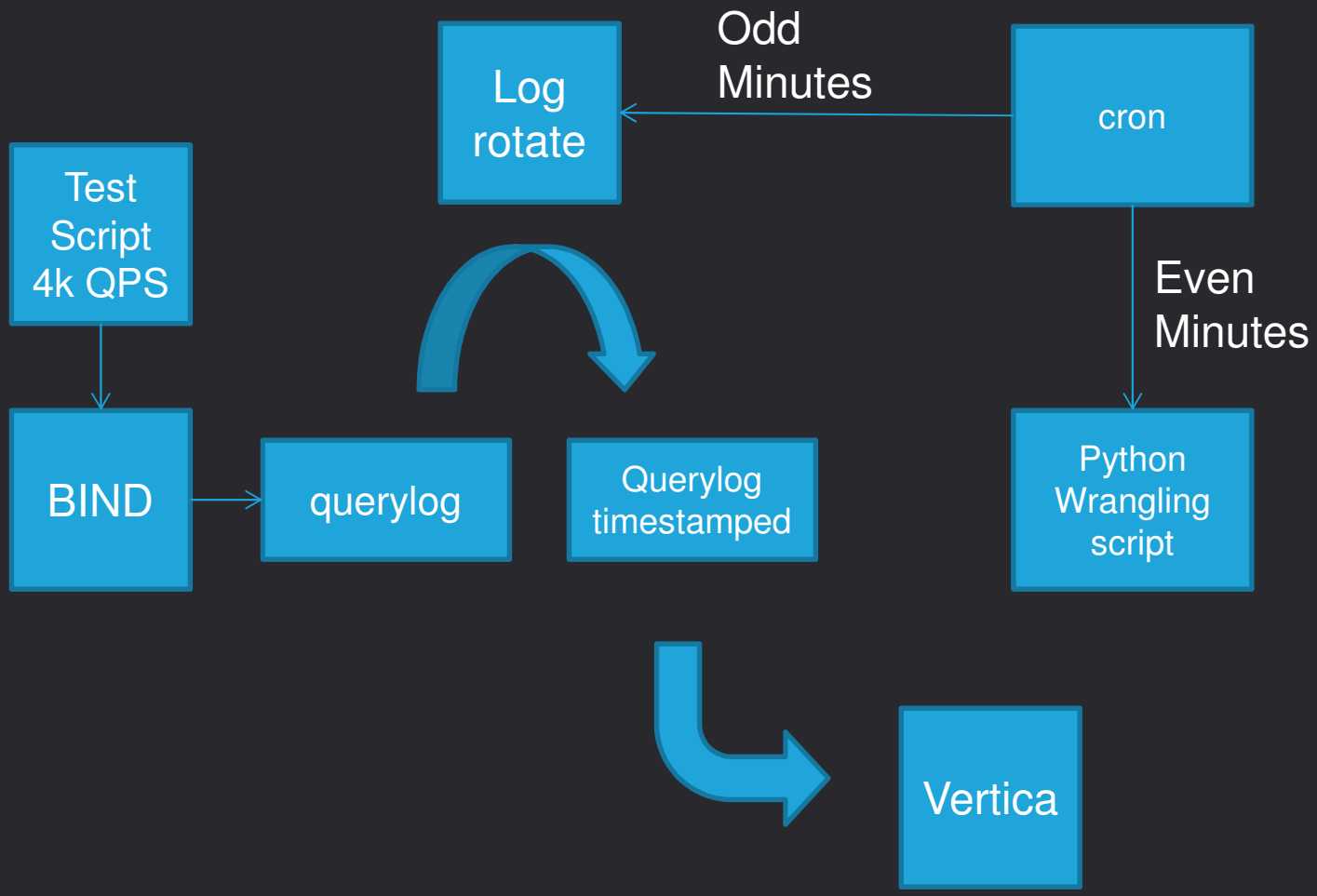
speeds up partial matches use qname_sld='nominum.com.' instead of qname=%.nominum.com.'

Convert IP Addresses to numbers

Convert Flags, etc to numbers

gnu parallel is your friend.

Demo Architecture



Based on http://deadbeefsec.wordpress.com/2013/05/12/bind-dns-query-log-shipping-into-a-mysql-database/

Data Set

24GB, 389M rows

Format

- unix timestamp
- client ip(dotted quad)
- client port
- class
- rtype
- flags
- rcode
- qname

Sample Tasks

Data Loading

Rollups by Time

Query1: Aggregate queries per hour

```
select to_timestamp(ts), sum(count) from dns_data_rollup group by
to_timestamp(ts) order by to_timestamp(ts) asc;
```

Query2: Queries per hour (singletons)

```
select to_timestamp(ts), count(*) from dns_data_rollup where count=1
group by to_timestamp(ts) order by to_timestamp(ts) asc;
```

Query3: Queries per hour by rcode

```
select to_timestamp(ts), rcode, sum(count) from dns_data_rollup group
by to_timestamp(ts), rcode order by to_timestamp(ts) asc;
```

Query4: Queries per hour by rtype

```
select to_timestamp(ts), type, sum(count) from dns_data_rollup group
by to_timestamp(ts), type order by to_timestamp(ts) asc;
```

Performance

Task	Postgres	Vertica
Data Loading	490	366
Coarsing	14958	336
Query1	676.784	4.995
Query2	459.790	3.370
Query3	627.396	5.052
Query4	612.322	5.227

How I Use It

- Fast research tool for caching resolver data
  - DOS/PRSD attacks
  - looking back over old data for patterns
  - csv data generation for reports
- Ask your data questions easily
  - Queries are not data science though.
  - PRSD + OpenResolverScan Data + Botnet
- Single box solution
  - Can be clustered
- Once what you're looking for is documented, pass to data engineering to automate.

Harness Your Internet Activity