NoNoSQL@Google

Olaf Bachmann, Google Inc.
olafb@google.com

Background

Ads Traffic Quality Team / Click police

One of biggest user of data analysis/storage systems

Major interest: Data analysis
Less interested: Transactions, Serving

Hypothesis

SQL: best-of-breed data analysis language

Google: A road from MapReduce to SQL

- Protocol Buffers
- MapReduce (MR)
- Sawzall
- Dremel
- SqlMR

==> SQL: Where it shines, where it whines

Protocol Buffers

http://developers.google.com/protocol-buffers/

Think XML, JSON:

language-neutral, platform-neutral, extensible

But:

smaller, faster, simpler

```
message SearchResult {
  required int32 doc_id = 1;
  required string url = 2;
  optional string header = 3;
}
```

```
WebSearch search;
add_res();
search.set_query("flowers");
search.set_ip_v4(str2int("1.2.3.4"));
```

```
message WebSearch {
  required string query = 1;
  required uint32 ip_v4 = 2;
  repeated SearchResult res = 3;
}
```

```
SearchResult *res = search.

res->set_doc_id(12345678);
res->set_url("http://flowers.com");
```

MapReduce: Basics

2003: Jeff Dean, Sanjay Ghemawat

Map: input ==> (key, value)
Reduce: (key, [values]) ==> output

Number of queries:
Map: WebSearch ==> (query, 1)
Reduce: (query, [1,...,1]) ==> query, len([1,..,1])

MapReduce: Implementation

Map, Reduce: user-supplied C++ functions

- concentrate on domain, only

Link with MRLib ==> Binary

- provides primitives of work-flow and parallelism

Launched on shared cluster of +10K cores.

- process data at tune of 50GB/s

MapReduce: Shine and Whine

Shine:

- write a simple program, run on 10k machines
- no need to have experience with parallel systems
- flexibility/low-level control via cmd-line opts

Whine:

- need MR-foo to debug/optimize
- a lot of boilerplate/repetitions (think: sums)
- engineers-only tool (C++, etc)
- no auditability and field access control

MR Memegen

Sawzall: Basics

2005: Rob Pike et. all

szl: scripting language for MR

Map: input + szl ==> Emit to "aggregators"
Reduce: 10+ system-provided aggregators

sum, maximum, quantile, sample, unique, top

Sawzall: Example

```
proto "WebSearch.proto"
search: WebSearch = input;
nperip: table sum[ip: string] of count: int;
if (search.query == "flowers") {
  emit nperip[format_ip(search.ip_v4)] <- 1;
}

$ saw --program example.szl \
      --input_files /gfs/cluster1/websearch/2012/05/28/websearch/*.
recordio
      --destination /gfs/cluster2/$USER/nperip@100

$ dump --source /gfs/cluster2/$USER/nperip@100 --format csv
```

Sawzall: Shine and Whine

Shine:

- sawzall instead of C++ & gcc
- powerful & extensive library and aggregators
- users: engineers, product managers and analysts
- auditability and per-user access control

Whine:

- value-only semantic of szl, no custom reducers
- no built-in/awkward chaining
- Not as efficient and flexible as MR

Sawzall Memegen

Dremel: Basics

2006/2010: Sergey Melnik, Andrey Gubarev

Needs: Protocol Buffers as nested Columnio
Interprets: SQL queries over Protocol Buffers
How: Serving tree of +10K cores

- leaves: linearly scan through column-files
- inner nodes: aggregate results

==> Interactive query results over +10B records

Dremel: Example

```
$ dremel
> SELECT format_ip(ip_v4) as ip,
         count(*) as count
     FROM "/gfs/cluster1/websearch/2012/05/28/websearch/*.columnio"
    WHERE query == "flowers"
GROUP BY ip
```

ip	count
"1.2.3.4"	23
"3.4.5.6"	736
"6.7.8.9"	42
...	

Dremel: Shine and Whine

Shine:

- Interactive data analysis of +10B records
- SQL & CLI instead of szl & saw
- users: all Googlers (including sales folks)
- auditability and access control

Whine:

- Only effective on columnio ==> duplicate data
- Limits: size of output/intermediate data, input size
- See SQL Whines

Dremel Memegen

SqlMR: Basics

Idea: Implement SQL on top of MR

2011: Biswapesh Chattopadhya et. al. **Tenzing**

- cluster of specialized MR machines
- goal: sub-minute response times

2012: Bohdan Vlasyuk, Robert Buessow **SqlMR**
built on top of the general MR framework

- uses FlumeJava (Craig Chambers et. al. 2010) as intermediate API
- goal: close the Dremel gap

SqlMR: Whines and Shines

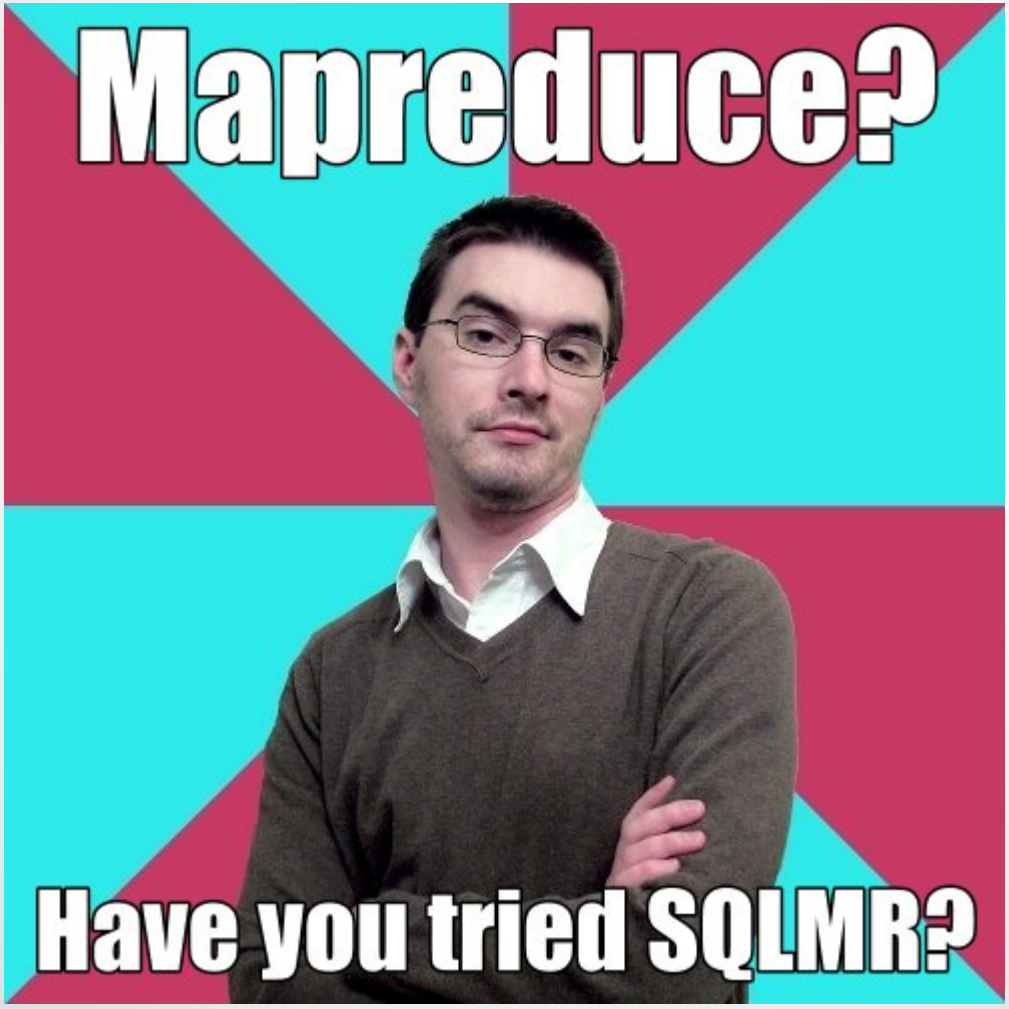
Shines:

- General MR limits, only
- Works on recordio, Bigtable, columnio, etc.
- Rapid prototyping based on Dremel
- Brings MR to the masses

Whines:

- No interactivity
- Dremel SQL compatibility
- Hard to debug/low-level control

SqlMR Memgen



SQL Shines

Simple

- 30K Googlers == 30K SQL users
- Easy to develop rich UIs
- Auditable and low-level access control

Declarative/Structured Query Language

- Same query ==> different exec engines
- Allows advanced optimizations
- Transparent multi-day result recovery

Concise

- Concentrate on business logic
- Sawzall vs SQLMR: +5x less code

SQL Conciseness

```
  SELECT format_ip(ip_v4) as ip,
         count(*) as count
    FROM "/gfs/cluster1/websearch/2012/05/28/websearch/*.columnio"
   WHERE query == "flowers"
GROUP BY ip
ORDER BY count DESC
   LIMIT 10;
```

SQL Conciseness vs Sawzall

OR: One Steps vs Five Steps

```
# Step 1: saw szl program
proto "WebSearch.proto"
search: WebSearch = input;
nperip: table sum[ip: string] of count: int;
if (search.query == "flowers") {
  emit nperip[format_ip(search.ip_v4)] <- 1;
}
# Step 2: run saw MR
$ saw --program examples_saw.szl \
      --input_files /gfs/cluster1/websearch/2012/05/28/websearch/*.recordio \
      --destination /gfs/cluster2/$USER/nperip@100
# Step 3: resaw szl program
type NPerIp = { ip: string @ 1, count: int @ 2};
topips: table maximum(10)[ip: string] of count: int;
nperip: NPerIp = input;
emit topips[nperip.ip] <- nperip.count;
# Step 4: run resaw MR
$ saw --program example_resaw.szl --resaw /gfs/cluster2/$USER/nperip@100 \
      --destination /gfs/cluster2/$USER/topips@1
# Step 5: dump results
$ dump --source /gfs/cluster2/$USER/topips@1
```

SQL Whines

SQL **Yay**: Family Van, Station Wagon or Sedan
SQL **Nay**: Porsche 911 or 4x4 Truck
==> 90% Yay, 10% Nay

Tree structured data:

```
message SearchResult {
  required int32 doc_id = 1;
  required string url = 2;
  optional string header = 3;
}
```

```
message WebSearch {
  required string query = 1;
  required uint32 ip_v4 = 2;
  repeated SearchResult res = 3;
}
```

```
SELECT count(*) from WebSearch;
SELECT count(*) from WebSearch OVER SearchResult;
```

SQL Whines

Metadata storage and discovery

Different dialects and extensions

SQL Whine Memgen

Conclusions

Google: MR ==> Sawzall ==> SQL

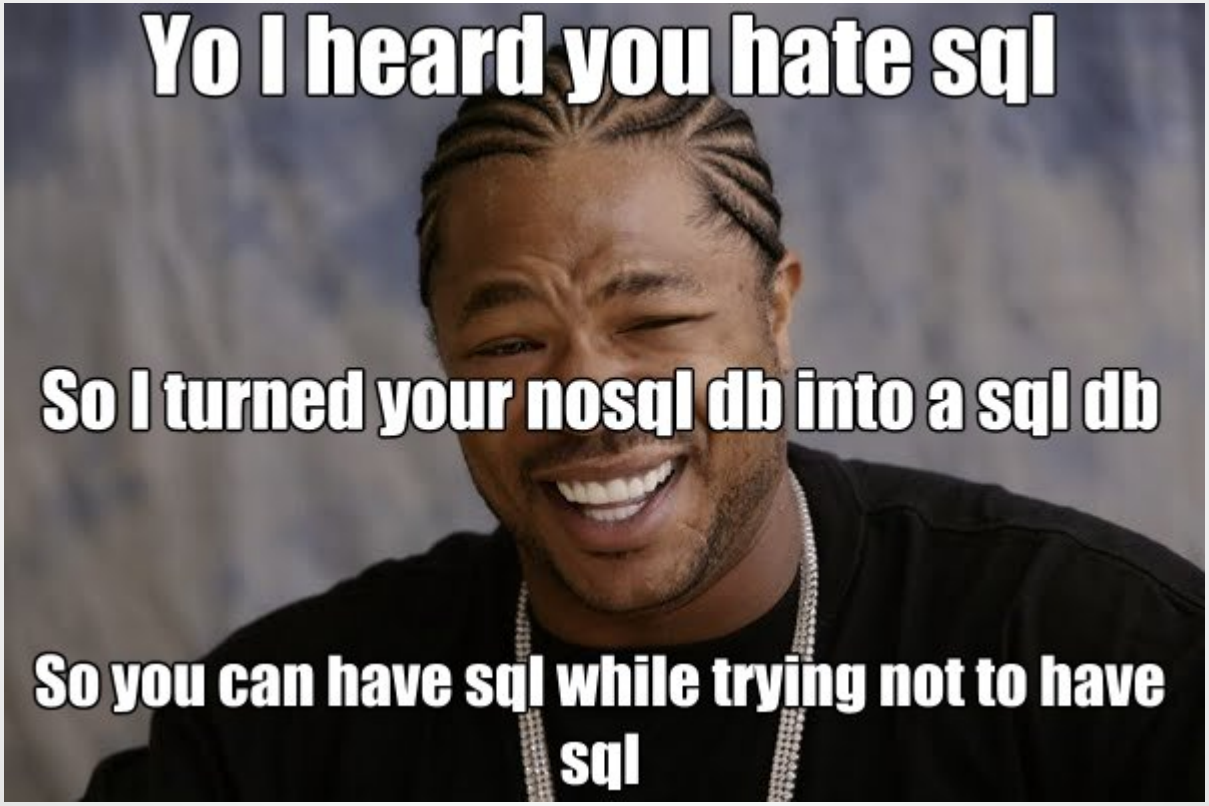
SQL@Google:

data analysis language of choice, read-only

NoSQL@Google:

RDBMS, read-write transactions

Final Words


Yo I heard you hate sql
So I turned your nosql db into a sql db
So you can have sql while trying not to have sql