

Solr Power FTW

Presented by



Alex Pinkin
@apinkin
#solrnosql

Produced by



What Will I Cover?

- Who I am
- What Bazaarvoice does
- SOLR and NoSQL
- Can SOLR handle 20K queries per second?
- Lessons learned: large scale multi data center deployment
- Conclusion



Alex Pinkin

- Software Engineering Lead, Data Infrastructure team, Bazaarvoice
- Loves to play with SQL and NoSQL





Bazaarvoice

 Bazaarvoice is a software as a service company powering user generated content such as ratings and reviews on thousands of web sites



- 5 billion page views per month
- 230 billion impressions
- 75 million UGC



NoSQL?

HOW TO WRITE A CV







Leverage the NoSQL boom

SQL vs NoSQL





PostgreSQL















NoSQL is Not Only SQL

- Departs from relational model
- No fixed schema
- No joins
- Eventual consistency is OK
- Scale horizontally



Types of NoSQL

- Key-value (Redis, Riak, Voldemort)
- Document (MongoDB, CouchDB)
- Graph (Neo4J, FlockDB)
- Column family (Cassandra, HBase)



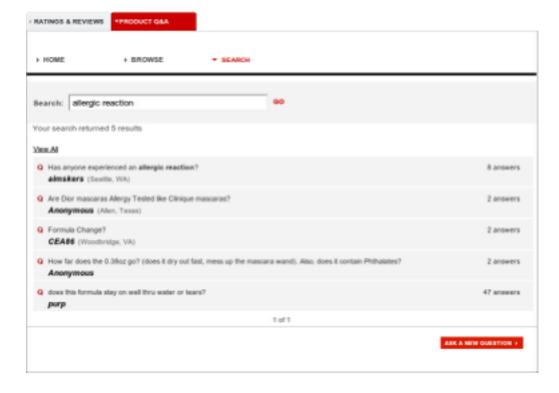
SOLR as NoSQL

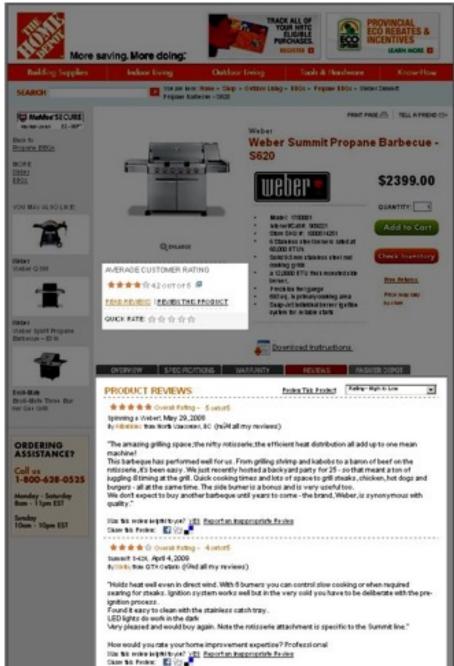
- Non-relational model Check
- No fixed schema Check (dynamic fields)
- No joins Check (denormalization)
- Horizontal scaling Check (with work)



SOLR stats - Bazaarvoice

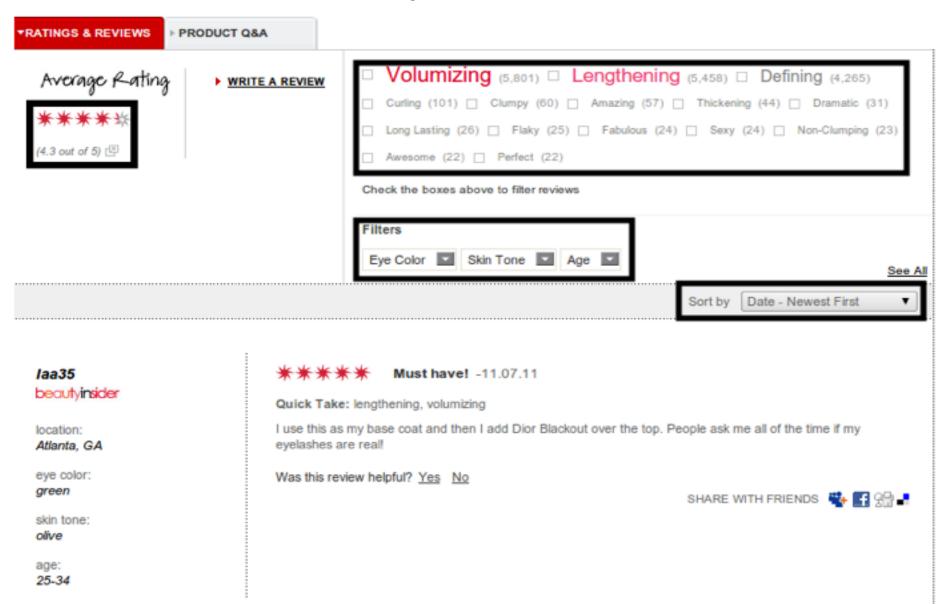
SOLR Case Study







SOLR Case Study





Life Before SOLR

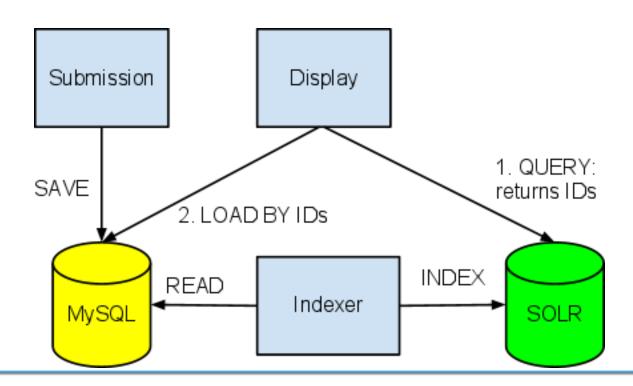
- Indexes for sorting and filtering
- Aggregate tables for stats
- Nightly jobs
- Bugs...





Enter SOLR

- Index content and product catalog
- De-normalization
- Filtering and sorting
- Index every 15 minutes (20 seconds NRT)





SOLR - Statistics

- COUNT, SUM, AVG, MIN, MAX (StatsComponent)
- Stored fields
- Whenever content changes, re-calc stats for all affected subjects

PRODUCT

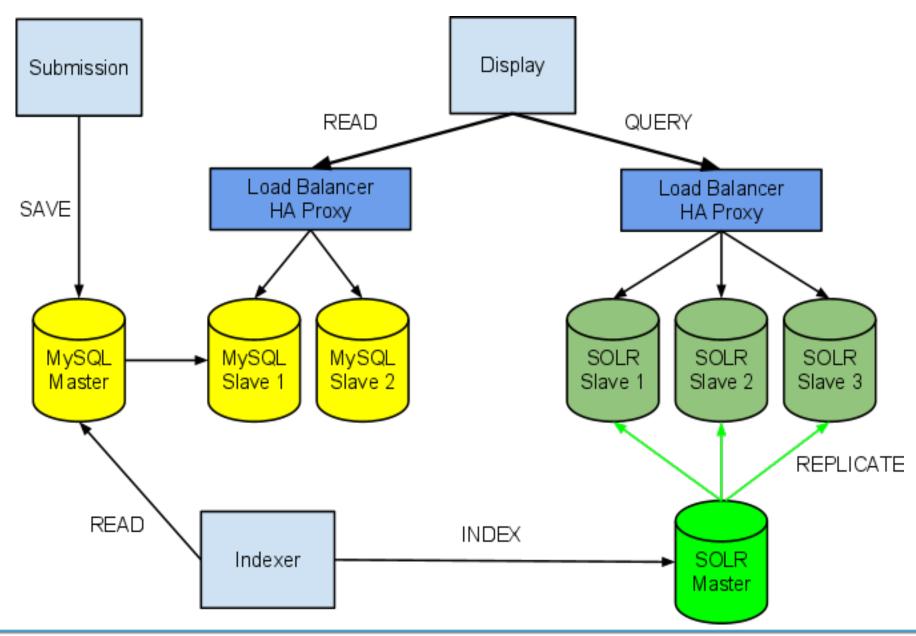
id name description categoryID

categoryHierarchylD categoryHierarchylD_0 categoryHierarchylD_1 categoryHierarchylD_2

reviewAverageRating reviewCount reviewLastSubmission questionCount

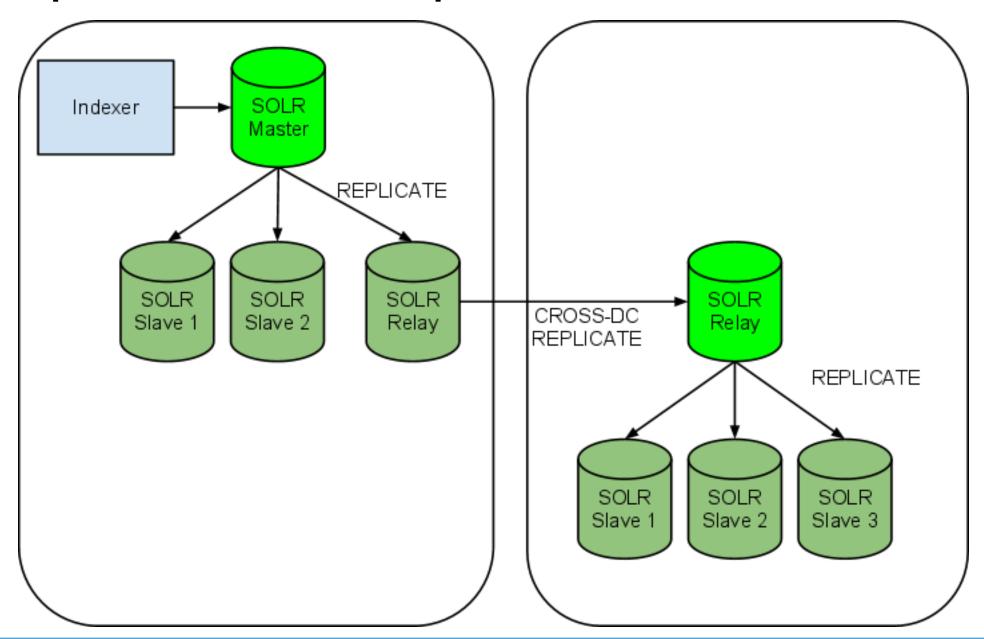


Scaling reads - Replication





Replication - Multiple Data Centers





Replication - Multiple Data Centers

Chatty if using multiple cores

Relay

- Core auto-warming disabled
- Connection wait and read timeouts increased
- Replication poll interval increased (15 min)
- Compression enabled

```
...
<str name="httpConnTimeout">20000</str>
<str name="httpReadTimeout">65000</str>
<str name="pollInterval">00:15:00</str>
<str name="compression">internal</str>
...
```



SOLR Cloud - Bazaarvoice version

- Multiple cores (100+ per server)
- Re-balance indexes across cores and servers
 - Automatic
 - Manual
- Deployment map stored in MySQL
 - Host Core Partition
 - Statistics
- Partition lifecycle



Schema Changes

Re-indexing is time consuming for large indexes

Process

- 1.Full re-index off-line prior to the release
- Incremental indexing after the release

Bottleneck: reading from MySQL

Goal: Transparent re-indexing



Performance Tuning

- Heap size
- Cache sizing
- Auto-warming
- Stored fields
- Merge factor
- Commit frequency
- Optimize frequency

Process: Simulate and measure

- Replay logs
- Analyze metrics
- Monitor GC



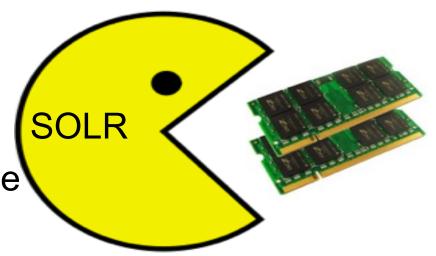
Performance Tuning - GC

```
# Java memory usage settings
# Force the NewSize to be larger than the JVM typically allocates.
# In practice, the JVM has been allocating an extremely small Young generation which
objects to be prematurely promoted to the Tenured generation
JAVA MEM OPTS="-Xms27g -Xmx27g -XX:NewRatio=8"
# -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps --> Turn on GC Logging
                                 --> Use the concurrent collector
# -XX:+UseConcMarkSweepGC
# -XX:+CMSIncrementalMode
                                --> Incremental mode for the concurrent collector
                                --> Let the JVM adjust the amount of incremental
# -XX:+CMSIncrementalPacing
collection
JAVA GC OPTS="-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:
+UseConcMarkSweepGC -XX:+UseParNewGC -XX:CMSInitiatingOccupancyFraction=55 -
XX:ParallelGCThreads=8 -XX:SurvivorRatio=4"
```



SOLR Performance - Summary

- SOLR loves RAM!
- Log replay
- Same config, same hardware
- Get the most out of one instance





Conclusion - SOLR Strengths

- Lightning fast given enough RAM
- Good scale out support including multi-data center
- Great community



Conclusion - SOLR's Gaps

- Not fully elastic
- Real time takes work
- Secondary data store = sync overhead
- Schema changes



Questions



