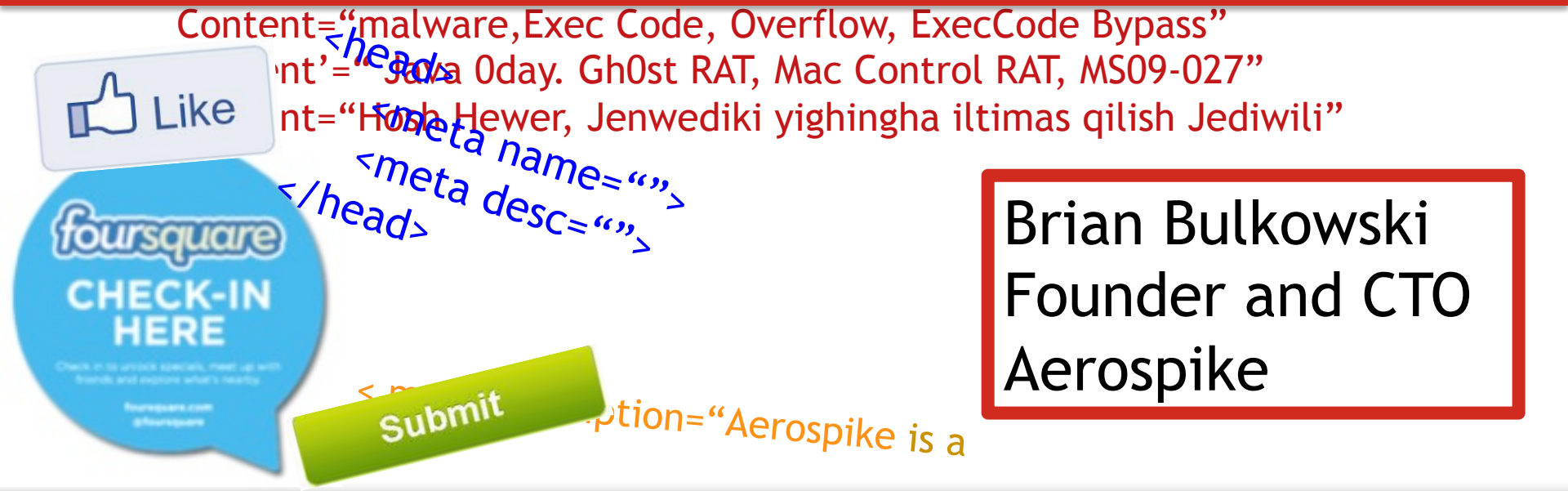
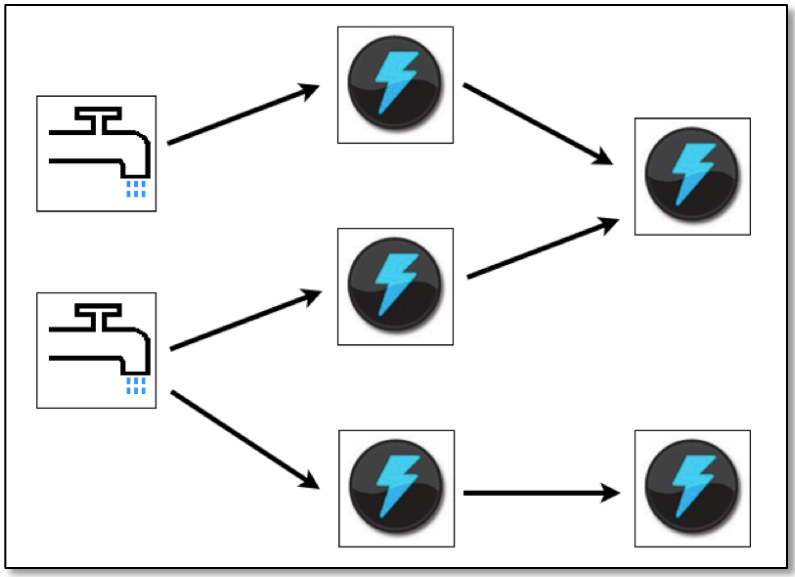
Real-time analytics with Storm, NoSQL, and Hadoop



Brian Bulkowski
Founder and CTO
Aerospike

What is Storm?

- Storm is an Apache open-source project (Hadoop of real time)
  - Created by Twitter
  - No “Cloudera”-like vendor
  - Return of “Message Oriented Middleware”



- Overview
  - **“Spouts”**
    connect with data sources
    (example: web log tail ingest)
  - **“Bolts”**
    analysis and data manipulation of any sort
  - **“Nimbus”**
    control entity re-balances and re-configures without downtime, add servers seamlessly
  - **“Topology”**
    Ordering of bolts and spouts
  - **“Tuple”**
    A Storm message
  - **“Trident”**
    Higher level abstraction supporting real-time joins, reliability

Why Storm?

- Same management framework as Hadoop
- Many Spouts and Bolts available

https://github.com/nathanmarz/storm-contrib

- Simple database integration
- **High performance & reliability (1MM/S/S ++)**
- Great documentation
- Multiple languages

- Excellent framework for event oriented implementation

But ---

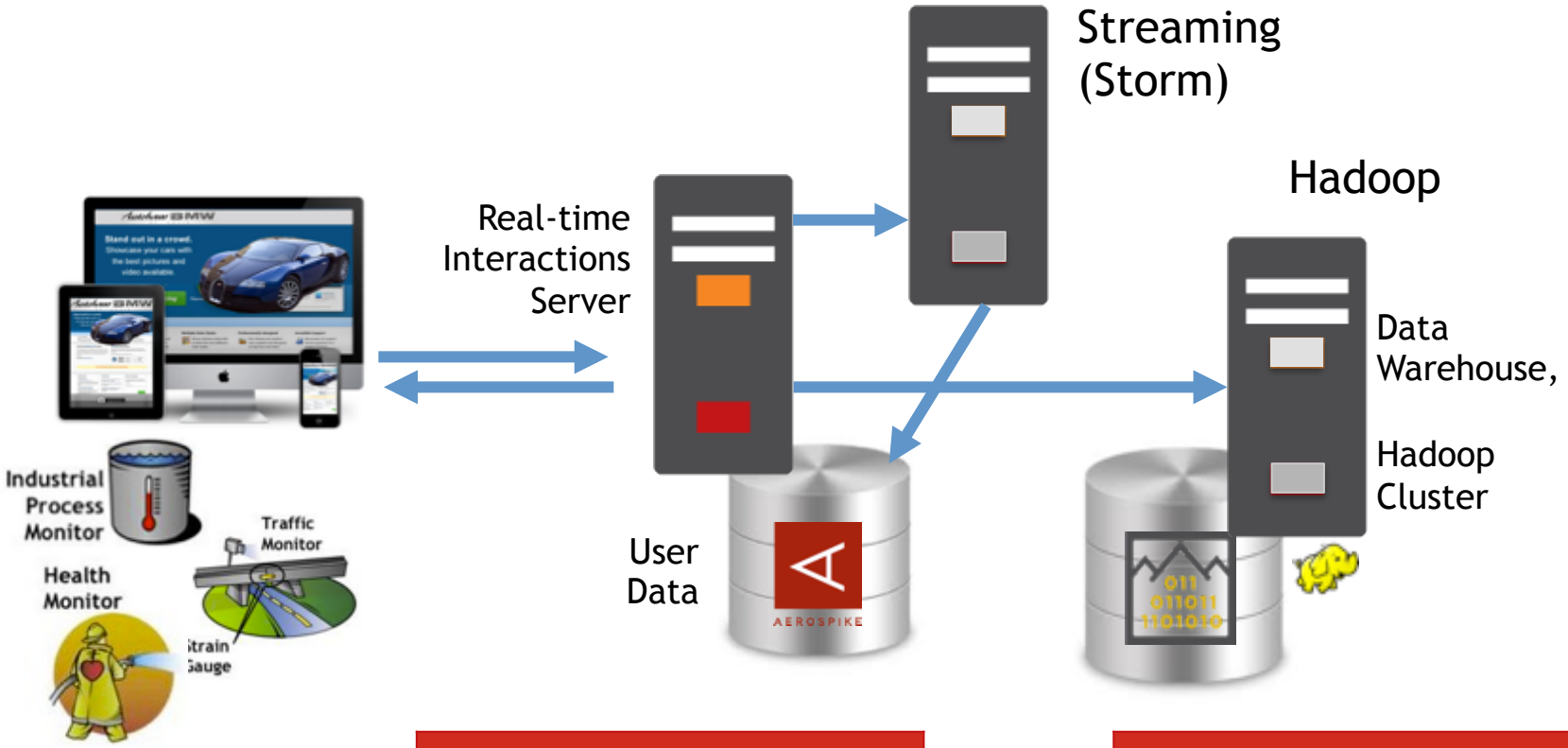
- Storm has problems

- 0mq -> Netty TCP transition
  - So much easier!
- Trident not good enough
  - But closer than most things
- Auto-rebalance framework

- Good news:

  Twitter doubling down

Streaming architecture



Real-time Interactions

- Frequency caps
- Recent ads served
- Recent search terms

Batch Analytics

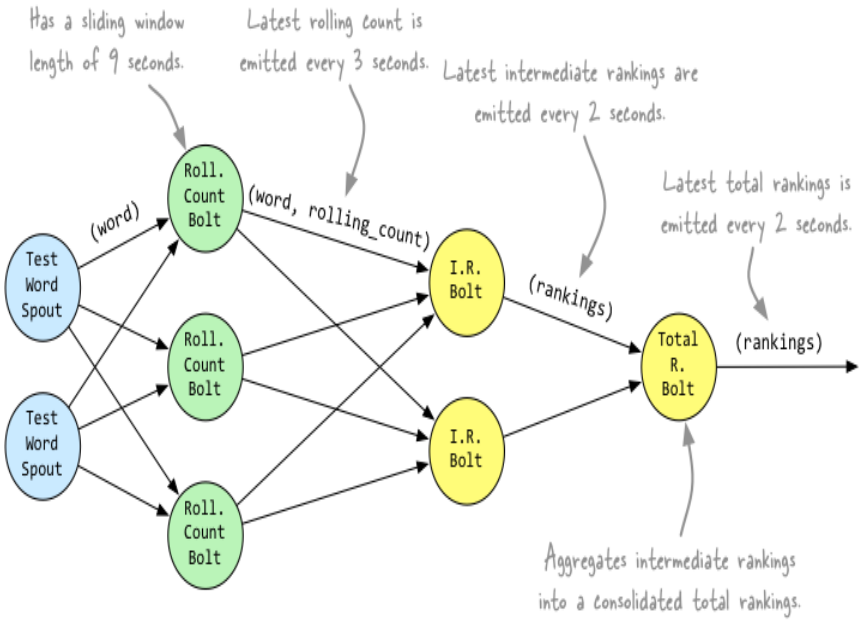
- User segmentation
- Location patterns
- Similar audience

Integration with Hadoop

- Example Bolt connectors for
  - Hdfs
  - Hadoop
- Example pattern
  - Bolt writes to HDFS
  - Bolt reads from HDFS
- Problem:
  - Storm is AT LEAST ONCE
- Solution:
  - write from your app to Hadoop
  - Emit from Hadoop to an edge database
  - Or read from Hdfs directly

Examples

- TrendingWords
  - Slice time into “slots” (1 minute / 5 minute) for each word
  - On seeing a word, increment bucket
  - Use Storm Ticks (0.8) for emitting top events
  - Determine most popular words in a category
  - Clean up old data on Ticks or on reads



Medium easy !
(Google
‘trending topics storm’)

http://www.michael-noll.com/blog/2013/01/18/implementing-real-time-trending-topics-in-storm/

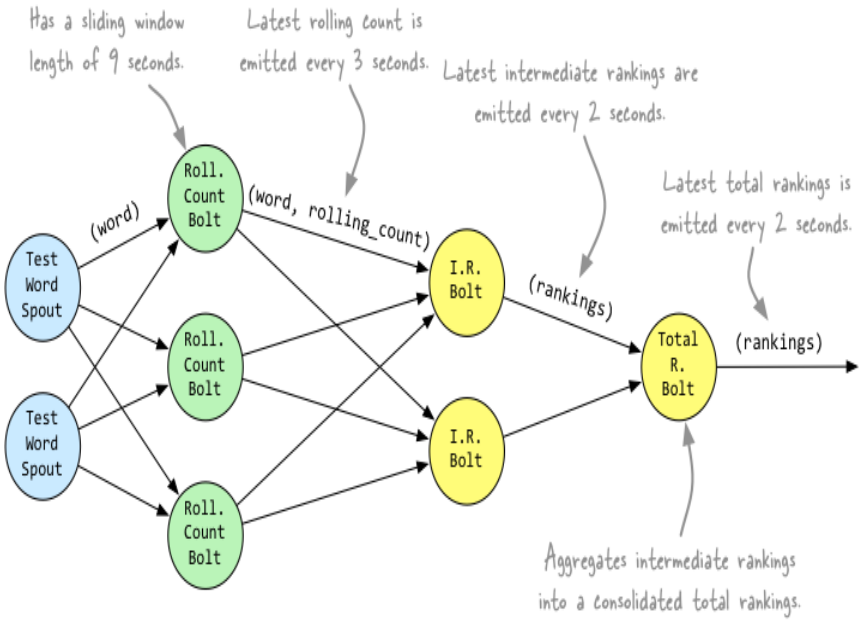
Examples

- TrendingWords – Database analysis – single words
  - Twitter at … 5k tps?
  - 20 indexable words per tweet
  - 10k writes per second
  - 50,000 words in English
  - A reasonable RAM problem

  Requires queries & kvs

- Tuples
  - Perhaps 30 tuples per tweet
  - 600K tps writes
  - 10M tuples ? Flash might be better

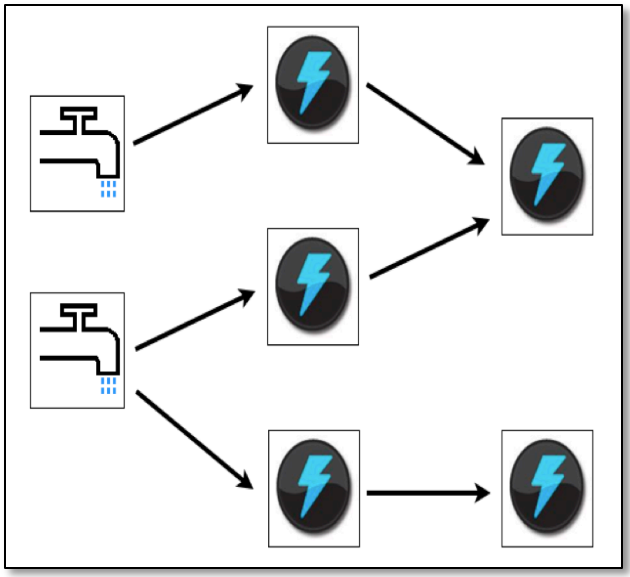
  Problem is much harder now!



http://www.michael-noll.com/blog/2013/01/18/implementing-real-time-trending-topics-in-storm/

Examples

- Recommendations
  - Multiple recommendation systems
  - Multi-arm bandit
  - https://github.com/tdunning/storm-counts/wiki/Bayesian-Bandit



- Simple fraud counts
  - Store recent requests for payment
  - Store recent users
  - Calculate fraud scores, drop events if past threshold

Storm overview

Development installs (easy)

- Everything on one server
  (no ZeroMQ)

- Zookeeper, Nimbus, Supervisors (slaves)
  all on one machine

- Write your Spouts and Bolts,
  but what next?

- Resources:
- https://github.com/nathanmarz/storm/wiki/Setting-up-development-environment

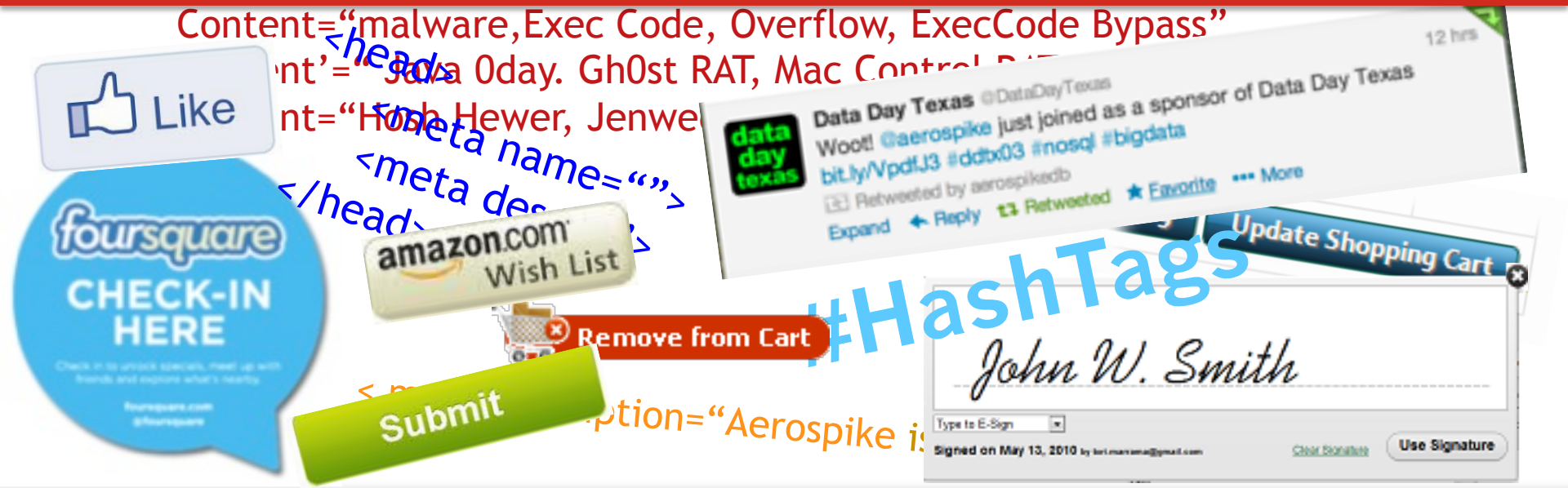
Real-world install

(distributed or go home)

- Zookeeper
  - A small (1 to 5) set of nodes to hold configuration. It's like DNS. Hopefully you have this running for Hadoop.
- Nimbus
  - Single server where "Topologies" are managed
  - Communicates via Zookeeper, "storm" namespace
- Supervisor (slave)
  - Needs the addresses of zookeeper nodes
  - Needs the Nimbus address
  - Then simply waits for work
- Best guide – this is really good !

http://www.michael-noll.com/tutorials/running-multi-node-storm-cluster/

Let's Begin



Real-world install

- Ubuntu 12.04 LTS

- In Amazon, there's a variety of scripts and helpers
  - (now you can build your own)
  - You can scale back to 1 machine (zoo & nimbus)

- In a laptop VM environment, use NAT
  - Share your `/etc/hosts` file

Prerequisites

- Packages
  - `sudo apt-get install git-core libtool autoconf g++ build-essential uuid-dev maven`

- Java
  - OpenJDK 6 “is unfortunate”, OracleJDK 6 ok
  - 7 can be OpenJDK or Oracle
  - Can’t mix and match 5, 6, 7
  - Will be building & submitting “fat jars”
  - JAVA_HOME everywhere

Prerequisites

- 0mq – only required in 0.8
  - Native, high performance messaging
  - Storm 0.9.0 avoids 0mq, uses HTTP (!)
  - 2.1.x (2.1.6 -> 2.1.11) ONLY
  - `sudo apt-get install libzmq-dev`

- Storm's Java 0mq client (ugly

http://stackoverflow.com/questions/12115160/compiling-jzmq-on-ubuntu

```
git clone https://github.com/nathanmarz/jzmq.git
cd jzmq
./autogen.sh
./configure
# overcoming flaw in marz's repo
cd src
touch classdist_noinst.stamp
CLASSPATH=.:./:$CLASSPATH javac -d . org/zeromq/ZMQ.java org/
zeromq/ZMQException.java org/zeromq/ZMQQueue.java org/zeromq/
ZMQForwarder.java org/zeromq/ZMQStreamer.java
cd ..
make
sudo make install
```

Zookeeper server

- Packages
  - `sudo apt-get install zookeeperd`
    'd' is for servers, without is for clients
  - `export PATH=$PATH:/user/share/zookeeper/bin`
  - `sudo update-rc.d zookeeper defaults`

- Config
  - All zookeepers know about all other zookeepers
  - `/etc/zookeeper/conf/myid`
    -> 1 through 255
  - `/etc/zookeeper/conf/zoo.cfg`
    list IP or DNS for all other zookeepers
  - Service is up only if a majority is up

Zookeeper server

- Validate
  - zkCli.sh -→ (whines a bit, ends up [CONNECTED])
  - ```
    ls /
    create /foo mydata
    get /foo
    set /foo otherdata
    get /foo
    ```

- Notes

  Data directory is /var/lib/zookeeper

  Config file is /etc/zookeeper/conf/zoo.cfg

  “client port” is 2181
    ```

Let's Install Storm

Storm

- No packages in Ubuntu
  - Download http://storm-project.net/downloads
  - Into /opt/storm, unzip, add to path
  - export PATH=$PATH:/opt/storm/storm-0.8.2/bin
- Create user “storm” ; remember Path + Java

```
sudo chown –R storm:storm /opt/storm
sudo chown –R storm:storm /var/data/storm
```

- Update config file

```
# update the storm configuration file - create a data directory, point storm
at it
sudo mkdir -p /var/data/storm
sudo chmod a+w /var/data/storm
sudo vi /opt/storm/storm-0.8.2/conf/storm.yaml
storm.local.dir: "/var/data/storm"
:
storm.zookeeper.servers:
   - “192.168.62.149”
```

- YAML is touchy. Make sure to have a space after the ‘:’

Storm

- Try running components stand alone

```
sudo su - storm
cd /opt/storm/storm-0.8.2/
bin/storm nimbus
bin/storm supervisor
bin/storm ui
```

- Reliability through low state, quick restart – supervisor !

```
sudo apt-get install supervisord
cp storm.conf /etc/supervisor/conf.d
```

- Extra hint

```
sudo unlink /var/run/supervisor.sock
```

- Storm.conf

```
[program:storm-supervisor]
command=/opt/storm/storm-0.8.2/bin/storm supervisor
user=storm
autostart=true
autorestart=true
startsecs=10
startretries=999
log_stdout=true
log_stderr=true
logfile=/var/log/storm/supervisor.out
logfile_maxbytes=20MB
logfile_backups=10
```

Storm

- storm-master.conf

```
[program:storm-nimbus]
command=/opt/storm/storm-0.8.2/bin/storm nimbus
user=storm
autostart=true
autorestart=true
startsecs=10
startretries=999
log_stdout=true
log_stderr=true
logfile=/var/log/storm/nimbus.out
logfile_maxbytes=20MB
logfile_backups=10

[program:storm-ui]
command=/opt/storm/storm-0.8.2/bin/storm ui
user=storm
autostart=true
autorestart=true
startsecs=10
startretries=999
log_stdout=true
log_stderr=true
logfile=/var/log/storm/ui.out
logfile_maxbytes=20MB
logfile_backups=10
```

Let's Run Storm

Run some storm !

- Copy storm.yaml to a local user's directory

```
mkdir ~/.storm
sudo cp  /opt/storm/storm-0.8.2/conf/storm.yaml ~/.storm/storm.yaml
sudo chown brian:brian ~/.storm/storm.yaml
```

- Install 'leiningen' (clojure tools)

  Package version too old
  http://leiningen.org/ download 'lein'
  into /usr/local/bin - chmod +x /usr/local/bin/lein

- storm-starter

```
git clone http://github.com/nathanmarz/storm-starter.git
cd storm-starter
lein deps ; lein compile ; lein uberjar
```

Run some storm !

- Submit the topology

```
storm jar target/storm-starter-0.0.1-SNAPSHOT-standalone.jar
storm.starter.ExclamationTopology exclamation-topology
```

- Watch the output

```
tail -f /opt/storm/storm-0.8.2/logs/worker-xxxx.log
```

- View the job

```
$ storm list
Topology_name        Status  Num_tasks Num_workers Uptime_secs
-------------------------------------------------------------
exclamation-topology ACTIVE 16         3           798
```

Coding to Storm

Creating topologies

- Say it with code

```
public static void main(String[] args) throws Exception {
  TopologyBuilder builder = new TopologyBuilder();

  builder.setSpout("word", new TestWordSpout(), 10);
  builder.setBolt("exclaim1", new ExclamationBolt(), 3).shuffleGrouping("word");
  builder.setBolt("exclaim2", new ExclamationBolt(), 2).shuffleGrouping("exclaim1");

  Config conf = new Config();
  conf.setDebug(true);

  if (args != null && args.length > 0) {
    conf.setNumWorkers(3);

    StormSubmitter.submitTopology(args[0], conf, builder.createTopology());
  }
  else {

    LocalCluster cluster = new LocalCluster();
    cluster.submitTopology("test", conf, builder.createTopology());
    Utils.sleep(10000);
    cluster.killTopology("test");
    cluster.shutdown();
  }
}
```

From the ExclaimationTopology

Spouts

- RandomSentenceSpout

```
public void nextTuple() {
  Utils.sleep(100);
  String[] sentences = new String[]{
    "the cow jumped over the moon",
    "an apple a day keeps the doctor away",
    "four score and seven years ago",
    "snow white and the seven dwarfs",
    "i am at two with nature" };
  String sentence = sentences[_rand.nextInt(sentences.length)];
  _collector.emit(new Values(sentence));
}
```

Spouts

- Easy interaction with queues
  - https://github.com/nathanmarz/storm-contrib
- storm-kafka
  - Popular persistent, topic-based queue
- storm-jms
  - Java message service
- storm-sqs
  - EC2's queue service
- more!

Bolts

- Simple exclamation bolt

```
public static class ExclamationBolt extends BaseRichBolt {
  OutputCollector _collector;


  @Override
  public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
    _collector = collector;
  }


  @Override
  public void execute(Tuple tuple) {
    _collector.emit(tuple, new Values(tuple.getString(0) + "!!!"));
    _collector.ack(tuple);
  }


  @Override
  public void declareOutputFields(OutputFieldsDeclarer declarer) {
    declarer.declare(new Fields("word"));
  }
}
```

Persisting Bolts

- Persistence is good ! Scale the data layer
  - Shared required for counters like TrendingWords
  - Stateless Bolts are better
- Easy persistence through examples
  - https://github.com/nathanmarz/storm-contrib
- storm-rdbms
  - Stores to a database table
- storm-cassandra
  - Stores to a cassandra cluster
- storm-redis
- storm-memcache

Aerospike Bolts

- Bolts available on github
  - https://github.com/aerospike/storm-aerospike

- EnrichBolt
  - Add fields from column after looking up a key
- PersistBolt
  - Store fields based on a key

Consider Aerospike

- If you need the speed of Storm, you need the speed of Aerospike
  - That's why we talk about Storm!
- Free version at http://aerospike.com/
- Internap – free high performance SSD servers for trial
- Benefits
  - In memory with FLASH
  - Clustered for high performance
  - HA state matches Storm's stateless model

Interesting Bolts

- MovingAverageWithSpikeDetection
  - https://github.com/stormprocessor/storm-examples
- TrendingTopics
  - http://www.michael-noll.com/blog/2013/01/18/implementing-real-time-trending-topics-in-storm/
- Baysian Bandit
  - https://github.com/tdunning/storm-counts

Why Aerospike ?

Why not other databases?

- Low latency - database requests in bolts
- Proven in production with large-scale deployments
- Flash optimized
  - Do you need more than 30G ?
- Read / write optimized
- Faster & more reliable than than Kafka (Cassandra based)
- Faster than Mongo
- More scale than Redis

Aerospike: the gold standard for high throughput, low latency, high reliability transactions

Performance

- Over ten trillion transactions per month
- 99% of transactions faster than 2 ms
- 150K TPS per server

Scalability

- Billions of Internet users
- Clustered Software
- Automatic Data Rebalancing

Reliability

- 50 customers; zero service down-time
- Immediate Consistency
- Rapid Failover; Data Center Replication

Price/Performance

- Makes impossible projects affordable
- Flash-optimized
- 1/10 the servers required

AppNexus Scales Up Performance with Aerospike* NoSQL Database

- AppNexus, largest independent Ad Platform deployed for almost 4 years, 4.35M TPS (steady state), 67 Billion records, 122 TB, only 120 servers across 8 data centers

AppNexus	data center	cluster	Nodes per cluster	TB per node	TB per cluster	utilization % per node	Read and Write TPS (K) per node	Read and Write TPS (K) per cluster
Baremetal, Equinix Data Centers	NY	1	12	2.2	26.4	50	85	1020
	NY	1	24	0.5	12	33	10	240
	LAX	1	12	2.2	26.4	50	70	840
	LAX	1	27	0.5	13.5	30	10	270
	AMS	1	12	1.4	16.8	30	50	600
	AMS	1	12	0.5	6	33	10	120
	FRA	1	10	1.6	16	40	60	600
	FRA	1	11	0.5	5.5	34	60	660
		8	120		122.6	37.5		4350
TOTAL		Total number of Clusters replicating	Nodes across clusters		TB across clusters			TPS (K) across clusters

2 Trillion Transactions per month

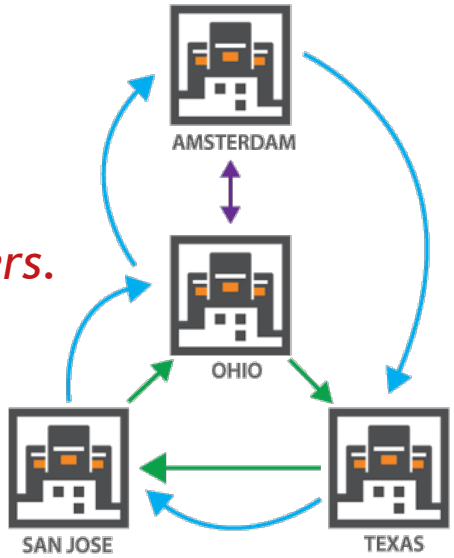
- 100,000 attributes each for 160 Million users
  - Data used by recommendation engines, e-commerce, RTB, mobile, video and display ad platforms

- 16 TB of data on 400 Million users
  - 60 Billion Transactions per month
  - 4 data centers for geographic proximity & high availability

- *"Scale.*
  Real-time performance.
  Real-time replication at 4 datacenters.

  Aerospike delivered."

  - Elad Efraim, CTO



AEROSPIKE

Better than the Competition

mongoDB	Cassandra	Couchbase	redis
➤ Hard to Maintain ➤ Performance	➤ Latency ➤ Number of Servers	➤ Stability ➤ Cost of RAM	➤ Cost of RAM ➤ Scalability

adMarketplace.com
Performance-Driven Pay Per Click Advertising

ACXIOM

ebay

MediaV

CS.
choicestream.

AppLovin
Your Mobile Audience Delivered

Chango

秒针系统
Miaozhen Systems

sitescout

bluekai

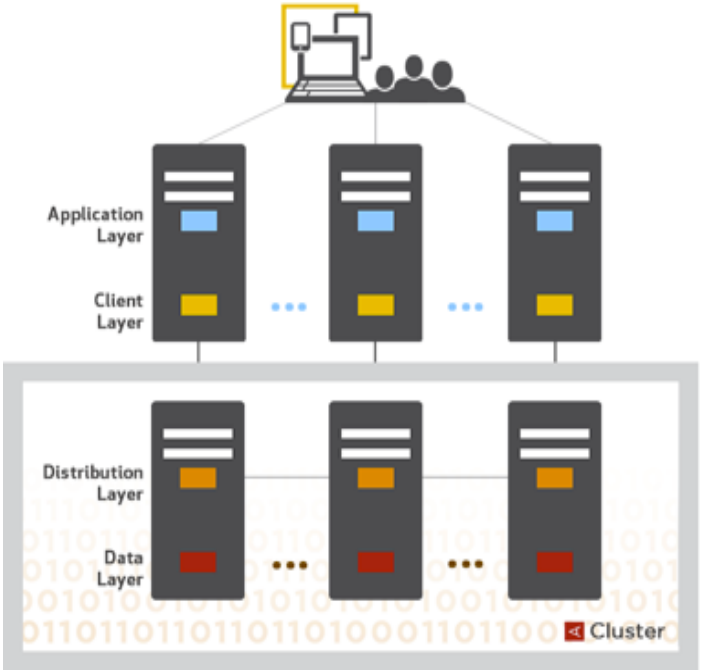
[x+1]
Make Every Interaction Count

madvertise

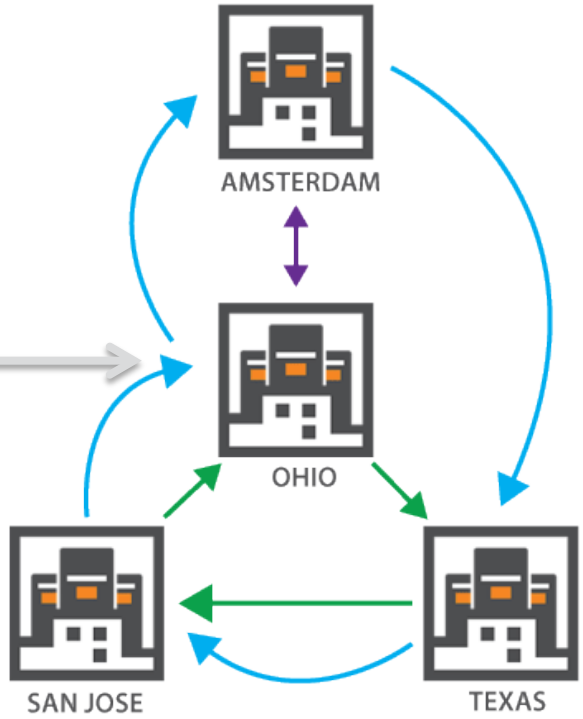
So-net

ADMETA

Simpler Scaling: Fewer Servers, ACID, Zero Touch



OHIO



1) **No Hotspots**
– DHT with RIPEMD160 simplifies data partitioning

2) Smart Client – **1 hop** to data, no load balancers

3) **Shared Nothing** Architecture, every node identical

4) Single row **ACID**
– synch replication in cluster

5) Smart Cluster, **Zero Touch**
– auto-failover, rebalancing, rolling upgrades..

6) Transactions and long running tasks prioritized real-time

7) **XDR** – asynch replication across data centers ensures **Zero Downtime**

Real-time Analytics on Operational Data

DISTRIBUTED QUERIES

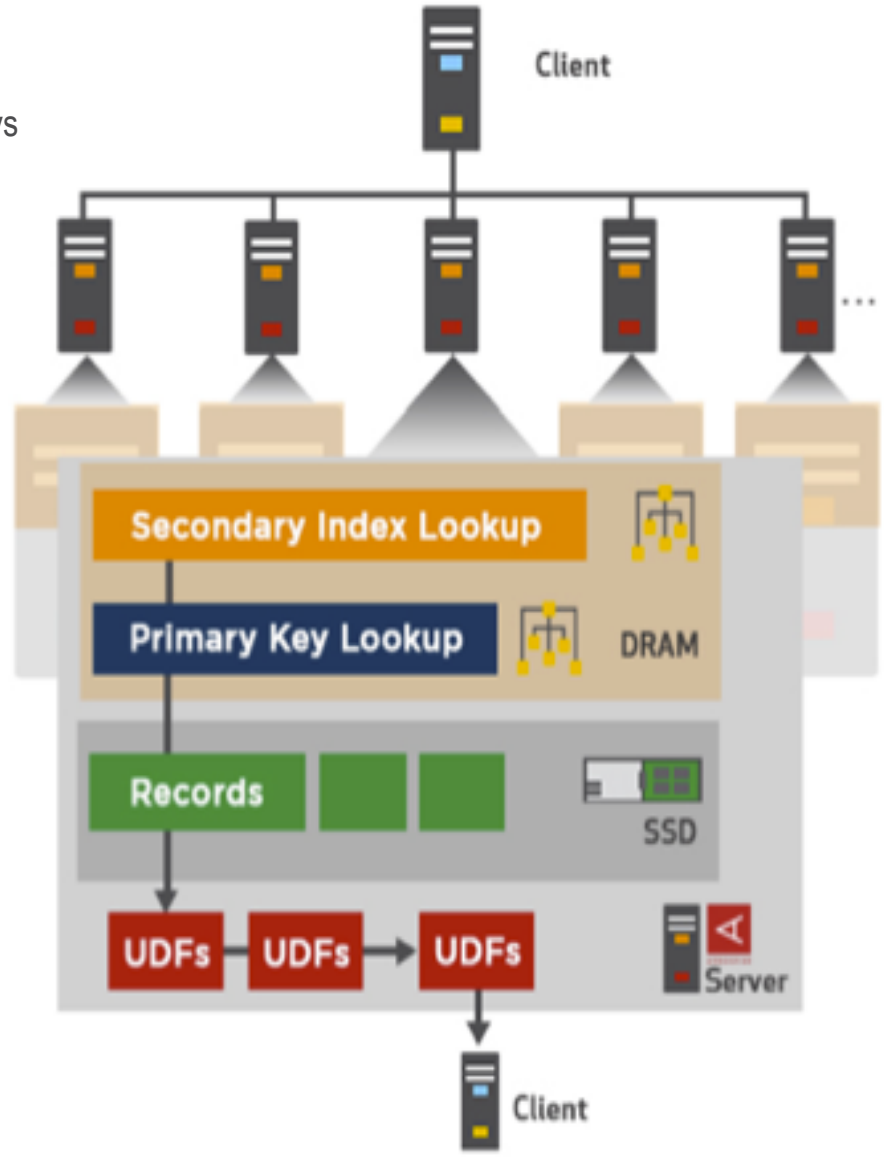
1. "Scatter" requests to all nodes
2. Indexes in DRAM for fast map of secondary → primary keys
3. Indexes co-located with data to guarantee ACID, manage migrations
4. Records read in parallel from all SSDs using lock free concurrency control
5. Aggregate results on each node
6. "Gather" results from all nodes on client

STREAM AGGREGATIONS

1. Push Code/ Security Policies/ Rules to Data with UDFs
2. Pipe Query results through UDFs to Filter, Transform, Aggregate.. Map, Reduce

REAL-TIME ANALYTICS on OPERATIONAL DATA (No ETL)

- **In Database, within the same Cluster**
- **On the same Data, on XDR Replicated Clusters**



Questions ?