




loggly

Apache Lucene/Solr Meetup  
07/28/2010

# Introduction

- I'm Jon, from Loggly...
  - Founded 8 months ago, by Kord, Raffy,  and me to do...
- Log file management in the Cloud, which means...
- High-volume real-time indexing of your log data
  - An elephant-free approach
- My life before loggly...
  - Various search @ LookSmart (since '98; Ferret, Fast, WiseNut)
  - Lucene @ LookSmart (late '04; Furl), @ Technorati, @ Scout

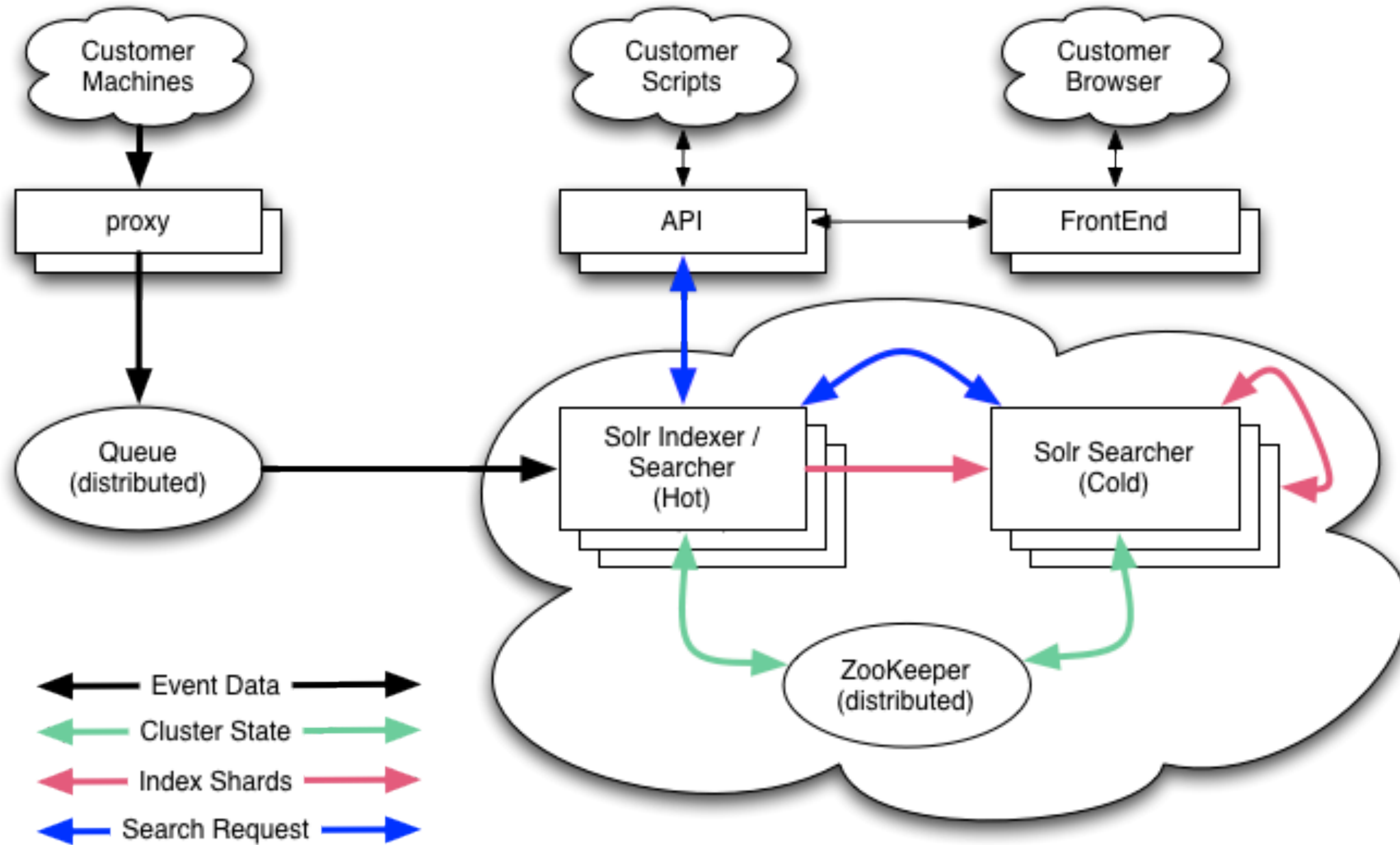
# Context

- Bad: Logs are painful for users
  - Distributed, large, ephemeral
- Bad: Logs are painful for us
  - High volume, high input rates
- Good: Most Log search is highly skewed
  - New data is far more important than old
- So: Distributability is not an option
  - For scale, performance, stability

# How we do it

- Syslog + 0MQ + SolrCloud
  - All of your logs in one place
  - Full-text search and graphs in (near) real-time
- The Cloud
  - Elastic disk, elastic compute
- Details, details
  - Logs are streamed to indexer/searchers, sliced into time-based shards, then merged and distributed automatically in the background (one “index” per customer)

# Pics or...



# Why SolrCloud?

- Solr is, well, you already know why its good
- Zookeeper maintains state about every node, every collection, and every shard, so
  - Any node can service any request, given only a Collection
  - Shard migrations are instantly visible to all nodes
- Each Customer has their own collection
  - Collection “slice” is a single shard, possibly on multiple nodes
  - Time based shards mean we can be smart about which shards to search, and where those shards live

# Why 0MQ?

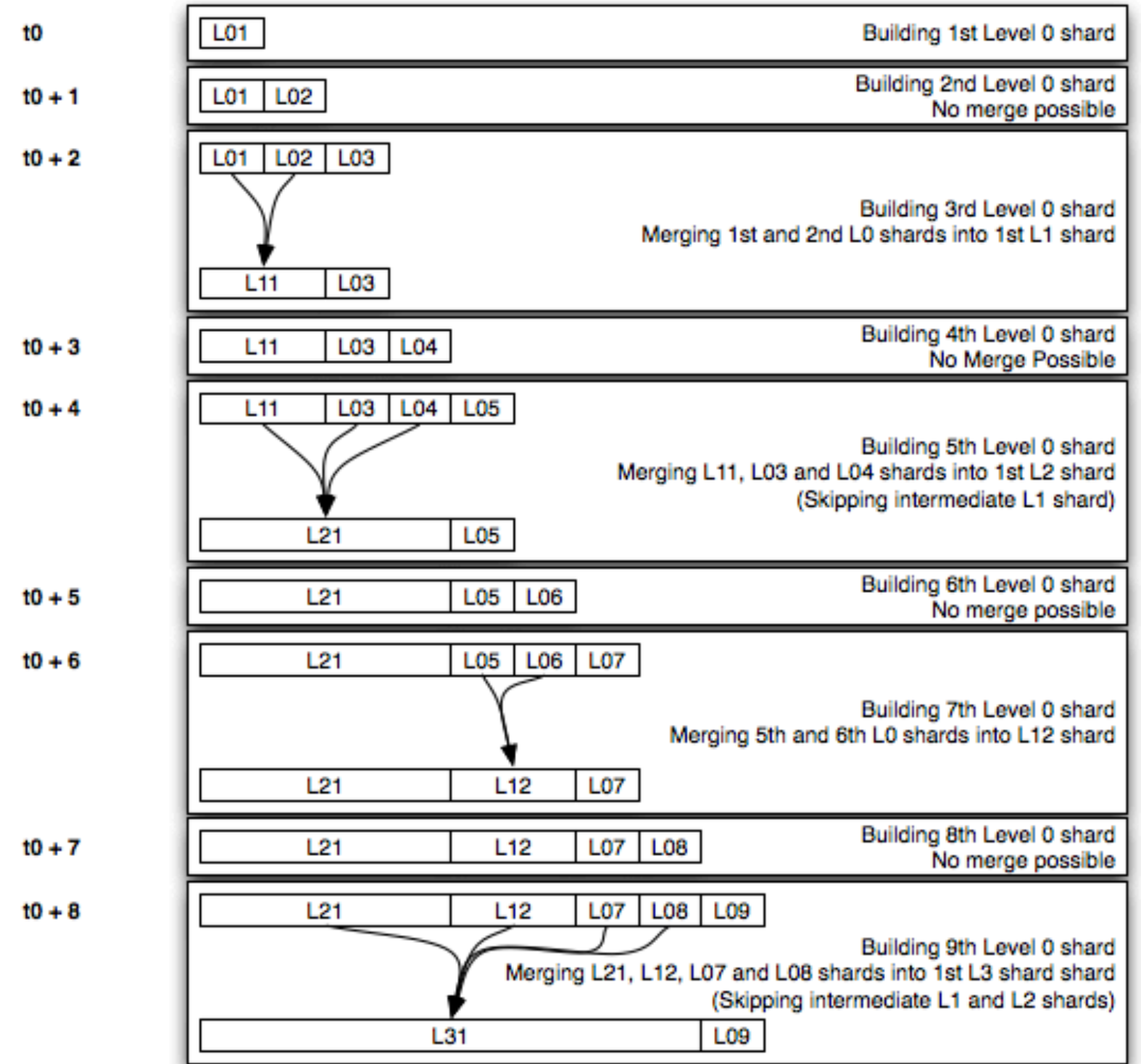
- 0MQ gives us node-specific input queues for Solr
  - Only contain data being indexed on that node
- Potentially multiple input queues for each Solr node
  - Input could come from any proxy
- Solr internal reader routes each event to an indexing core
  - Routing is cheap (one header field in the event)
- 0MQ is faster than a one-armed paper hanger
  - Easily able to cope with 100k's of events/second

# Sharding

- Time-sliced, based on when we receive the event
- Hot shards are small, frequently updated
  - NRT + SolrCloud = Our Nirvana
- Hot shards are “chilled” when we stop writing to them
- Cold shards are created by merging
  - chilled + cold = cold, cold + cold = colder, ...
  - multiple configurable “levels”, we’re still playing with values
- Indexing can be migrated with minimal (~10s) cost

# The Life of a Shard

- Events read from 0MQ, routed to an indexer and added to the hot shard
- Hot shard is Chilled (ie. index stops updating)
- Repeat
  - merge into higher level shard?
  - migrate to another node?
- until at highest level
- Expire into bit-bucket (or S3?)



# Anarchy?

- Every Solr (Hot or Cold) node can:
  - ▶ Create new merged shards from any of the shards it owns
  - ▶ Migrate any shard it owns
    - ▶ Usually triggered by creation, but can be manually controlled
    - ▶ New owner chosen using ZooKeeper data
    - ▶ When fully migrated, the new owner can merge the new shard with others it already owns
    - ▶ Migration uses the guts of the replication handler
- Completely distributed, automatic shard management

# So, does it work?

- So far, so good...
  - ▶ We consume our own logs, and use the system to find our own bugs and check system status
    - ▶ A bad push to our dev cluster generated ~3 million events in ~30 minutes. We cruised through it (~1700 events/second)
  - ▶ We have working versions of everything I've talked about:
    - ▶ OMQ, Time sharding, Automatic shard management
  - ▶ We have a handful of private beta customers

# What's next?

- Upgrading to the latest version of Solr/Lucene
- We need a distributed data store
  - We're still using lucene as our data store. There are so many things wrong with this (but hey, it got us going)
- We're moving to Hadoop for large-volume analytics
  - Solr is awesome at what it does, but not so good for mining
- Syslog is the only way in, for now
  - We'll be adding others (http, 0MQ? scribe?, flume? lots of options)

**loggly**

**ACK/FIN**