My SolrCloud Experience

- Currently, working on scaling up to a 200+ node deployment at LucidWorks

- Operated 36 node cluster in AWS for Dachis Group (1.5 years ago, 18 shards ~900M docs)

- Contributed several tests and patches to the code base

- Built a Fabric/boto framework for deploying and managing a cluster in EC2

- Co-author of Solr In Action; wrote CH 13 which covers SolrCloud
Solr Scaling Toolkit

- Requirements
- High-level overview
- Nuts and Bolts (live demo)
- Roadmap
- Q&A
• Provisioning N machine instances in EC2
• Configuring / starting ZooKeeper (1 to n servers)
• Configuring / starting N Solr instances in cloud mode (M x N nodes)
• Integrating with Logstash4Solr and other supporting services, e.g. collectd
• Day-to-day operations on an existing cluster
boto - Python API for AWS (EC2, S3, etc)
Fabric - Python-based tool for automating system admin tasks over SSH
pysolr - Python library for Solr (sending commits, queries, ...)
kazoo - Python client tools for ZooKeeper

Supporting Cast:
JMeter - run tests, generate reports
collectd - system monitoring
Logstash4Solr - log aggregation
JConsole/VisualVM - monitor JVM during indexing / queries
Fabric helps you do common system administration tasks on multiple hosts over SSH ...

- Just Python
- Easy to install / learn; good documentation
- [http://docs.fabfile.org/en/1.8/](http://docs.fabfile.org/en/1.8/)

```python
def kill(cluster):
    ec2 = _connect_ec2()
    taggedInstances = _find_instances_in_cluster(ec2, cluster)
    instance_ids = taggedInstances.keys()
    if confirm('Found %d instances to terminate, continue? ' % len(instance_ids)):
        ec2.terminate_instances(instance_ids)
    ec2.close()
```
Fabric in 3 minutes or Less, cont. ...

- Define all commands in a file named: **fabfile.py**
- Get a list of supported commands with short description

```
$ fab -l
Available commands:
  backup_to_s3                         Backup an existing collection to S3
  check_zk                             Performs health check against all ...
  commit                              Sends a hard commit to the ...
  ...
```

- Get extended documentation for a command

```
$ fab -d new_solr_cloud
Displaying detailed information for task 'new_solrcloud':
  Provisions n EC2 instances and then deploys SolrCloud; uses the new_ec2_instances and setup_solrcloud commands ...
```
**Solr Scale Toolkit: Architecture**

- **Meta Node**
  - Solr-Scale-Toolkit
  - SiLK

- **ZooKeeper Ensemble**
  - ZK Host 1
  - ZK Host N

- **SolrCloud Nodes (NxM nodes)**
  - Solr Node 1: Custom AMI
    - Solr Node 1: 8983
    - core
    - ..
    - core
  - ..
  - Solr Node N: 898x
    - core
    - ..
    - core

- Deploy and manage SolrCloud cluster
- System monitoring of M machines w/ collectd and JMX

---

**Confidential and Proprietary © Copyright 2013**
Provisioning machines

```
fab new_ec2_instances:test1,n=3,instance_type=m3.xlarge
```

- Custom built AMI?
- Block device mapping
  - dedicated disk per Solr node
- Launch and then poll status until they are live
  - verify SSH connectivity
- Tag each instance with a cluster ID and username
Two options:
- provision 1 to N nodes when you launch Solr cluster
- use existing named ensemble

Fabric command simply creates the myid files and zoo.cfg file for the ensemble
- and some cron scripts for managing snapshots

Basic health checking of ZooKeeper status:
- `echo srvr | nc localhost 2181`
SolrCloud Cluster: NxM nodes

EC2 Instance: RedHat Enterprise Linux, 64-bit

Solr 4.7.1 Node 1

collection1 shard1 / replica1 (Solr core)

... collection2 shard2 / replica1 (Solr core)

Limit to 50-100M docs across all cores per node

... Must design to give bulk of the memory to OS cache

Solr 4.7.1 Node N

collection3 shard1 / replica1 (Solr core)

... collection2 shard2 / replica1 (Solr core)

OS cache
memory mapped I/O

OS cache
memory mapped I/O

dedicated disk 1

dedicated disk N

... x M instances
• Upload a BASH script that starts/stops Solr
• Set system props: jetty.port, host, zkHost, JVM opts
• One or more Solr nodes per machine
• JVM mem opts dependent on instance type and # of Solr nodes per instance
• Optionally configure log4j.properties to append messages to Rabbitmq for Logstash4Solr integration
• BASH script that implements:
  – start/stop Solr nodes on each EC2 instance
  – sets JVM memory options, system properties (jetty.port), enable remote JMX, etc
  – backup log files before restarting nodes
  – ensure JVM is killed correctly before restarting

• Environment variables in:
  solr-ctl-env.sh
• Deploy a configuration directory to ZooKeeper
• Create a new collection
• Attach a local JConsole/VisualVM to a remote JVM
• Rolling restart (with Overseer awareness)
• Build Solr locally and patch remote
  – Use a relay server to scp the JARs to Amazon network once and then scp them to other nodes from within the network
• Put/get files
• Grep over all log files (across the cluster)
• **fab mine**: See clusters I’m running (or for other users too)
• **fab kill_mine**: Terminate all instances I’m running
  – Use termination protection in production
• **fab ssh_to**: Quick way to SSH to one of the nodes in a cluster
• **fab stop/recover/kill**: Basic commands for controlling specific Solr nodes in the cluster
• **fab jmeter**: Execute a JMeter test plan against your cluster
  – Example test plan and Java sampler is included with the source
SolrCloud Tools (SolrJ client app)

./tools.sh -tool healthcheck

- Java-based command-line application that uses SolrJ’s CloudSolrServer to perform advanced cluster management operations:
  - healthcheck: collect metadata and health information from all replicas for a collection from ZooKeeper
  - backup: create a snapshot of each shard in a collection for backing up to remote storage (S3)
- Framework for building complex tools that benefit from having access to cluster state information in ZooKeeper
SiLK Integration

• SiLK: Solr integrated with Logstash and Kibana
  – Index time-series data, such as log data (collectd, Solr logs, ...)
  – Build cool dashboards with Banana (fork of Kibana)

• Easily aggregate all WARN and more severe log messages from all Solr servers into logstash4solr

• Send collectd metrics to logstash4solr
Log Records Include:
- host:port
- collection
- shard
- test label
+ standard Log4J message fields
What’s Next?

• Migrate to using Apache libcloud instead of using boto directly
• Use this framework to perform large-scale performance testing
  – Report results back to community
• Ability to request spot instances
  – Good for testing only
• Chaos monkey tests
  – integrate jepsen?
• Open source so please kick the tires!
Wrap-up

- LucidWorks: [http://www.lucidworks.com](http://www.lucidworks.com)
- Connect: @thelabdude / tim.potter@lucidworks.com

Questions?