Scaling Solr

Amir Yahyavi

April 17, 2014

McGill University

1 Objective

The aim was to improve scalability and availability of Solr currently deployed on a single node.

1.1 Timing & Results

The project duration was 5 weeks which is a shame because I had just finished setting up the necessary scripts for comprehensive testing and experimentation. The results are not thoroughly repeated to make sure they are very robust and the recommendations require better study of SolrCloud and ZooKeepers so take them with a grain of salt. That being said they seem stable enough to draw some conclusions.

1.2 SolrCloud vs. Custom Solutions

I had a look at SolrCloud and also some of the custom solutions built (e.g., Tumblr). Admittedly I haven't had a chance to study all of them. SolrCloud also seems in very active development addressing many of the requirements that led companies to develop their solutions.
A complete evaluation of different solutions requires a lot more time and a much better study of how SolrCloud works. I had a choice here whether to implement one or two custom solutions and run a test or two and see if

hopefully they work, or write scripts and code that would help with deploying SolrCloud in different settings that would enable experimentation with different number of nodes, indexes, etc on different network settings to help run more comprehensive tests and find the solution. I opted to do the latter for 3 main reasons:

a. All of the solutions would require a time consuming setup and changing them won't be easy. It would require deploying zookeepers, Solr nodes, splitting the files, queries, and different document routing mechanisms for queries and documents, etc. Automated scripts will help with this.

b. Should I, another student, or you want to continue working on this later or look at it as more of a research question having these scripts would be more useful than having the results of one experiment.

c. Developing and maintaining a custom code might prove more costly vs. the gains it provides when the scale od deployment is limited (organization is not too big).

For these reasons, I wrote 16 scripts that would allow to deploy a given number of Solr nodes, with a given number of shards, and a given number of zookeepers on a number of different remote nodes with a number of different data files. A sample logs and some screenshots are included to show how they work/look. It would allow to kill running services on a node, start them up, deploy new nodes, etc. The scripts allow for posting different file to different shards, cleaning the files for faster posting, and splitting the files. They also generate property files that tell a SolrJ Java program which hosts to connect to and query. I have written a small java program for testing Solr using a given number of concurrently working threads.
SolrCloud automatically uses additional nodes for availability in case some nodes fail. I have tested the 16GB document files, split them (to 12 files) and indexed them and overall it seems to work fine. I have not experienced crashes so far when I tested it running 24+ hours with 6 threads for stressing the system.

2 Description of scripts

Here's a list of different scripts for deploying, starting, removing, cleaning, splitting, etc for SolrCloud. The scripts assume that your setup is similar to

the one on the github. However you can change that in the `config.sh` file. More detailed instructions are in Section 4.

Table 1: Table of scripts

clean.sh	Cleans XML data files of non-utf and control characters that cause Post utility to crash. Currently it doesn't enforce ASCII coding but if that's necessary, change the command in the script to the commented ASCII version.
index.sh	Will use Post utility on local node#1, with the XML datafile #2.
newzk.sh	Used to create a new local zookeeper node and to start it.
split.sh	Used to split a very large xml file #1 into #2 files with the same number of lines. It is mostly used for files larger than 1.5GB. The last doc inside the split file must be handled/corrected manually.
config.sh	The main configuration file. Must be adjusted to reflect the parameters in your setup including different folders, number of nodes, shards, etc.
install.sh	Extracts (if cannot find the file, downloads it) the Solr and Zookeeper. If the download link doesn't work replace with another server from Solr/Zookeeper page. Also generates the properties file.
oldrmnodes.sh	Removes local nodes. Deprecated by rmremotenodes.sh.
startnodes.sh	Connects and to existing nodes and starts them with their current configs.
deploynodes.sh	Main script for testing. Kills/removes existing nodes, configs, zookeepers. Creates new zookeepers and Solr nodes and starts them up. It also asks to whether index the default data files indicated in the config file or not. The parameters are configured in config.sh.
killremotenodes.sh	Connects to different hosts and zookeepers and kills their process. It does not remove the nodes.
rmremotenodes.sh	Same as kill except it also removes the nodes.

utils.sh	Includes a couple of common commands used by several scripts.
genprop.sh	Creates a properties files to be used by the SolrJ file.
newnode.sh	Creates a new local node #1, also passes the rest of parameters used to the starting Solr server. Copies the config files from config path.
smaller.sh	This is used for quick testing. Creates smaller files with n lines named U-"filename" which can be used for quick deployment and testing.
zkdeploy.sh	Creates and starts new local zookeeper nodes according to config.sh properties.

3 Other resources

The following other resources exist in the project:

Table 2: Table of folders

Solrj	Contains the SolrJ java sources that are designed to read and stress the servers using the q.txt queries. You can create a jar file using the build.xml file to easily deploy and run this. It creates a num_threads threads as defined in the config.sh file.
data	Contains the main XML data files (with current default name Solrn#.xml) and a queries file called q.txt
configs	Contains the schema and solrconfig files.
zk	Only exists so so zookeeper id files can be created.
logs.tar.gz	The logs from my runs

4 Code & Setting up the system

All the codes are available on my github: https://github.com/yahyavi/solrcloud. You can use `git clone git@github.com:yahyavi/solrcloud.git` to get the sources. You can follow the structure of the github repository or you can setup the config files to reflect your paths. In case your NFS/cloud drive has limited space or for performance improvement, you can deploy your large data files separately to the local hard drive of each node and have the solrcloud (from github), configuration files, query files, solrj.jar, etc files on your NFS folder (see `cmd.txt` to see how that would be setup) and create symlinks to these folders in your local hard drive's installation to these folder. This way the code runs on local hard drive and only configuration, and other smaller files that you might want to keep updating is on the NFS folder and is kept sync across.

5 Evaluation

The scripts allow us to test SolrCloud using many different parameters. The ones I decided to test were the number of nodes (N), number of shards (S), number of query threads (T), and number of zookeepers (Z). A 16 GB XML data file containing 752,450 documents was used to populate Solr (See Figure 1). It was first cleaned using the `clean.sh` script to avoid having control characters that cause problems with Solr and post tool. Post tool does not allow files of this size to indexed. As a result I used the scripts I wrote to split it into 12 files of roughly the same size (1.2GB). The file was deployed to all the nodes in the cluster (7 in total) for testing. They have Intel(R) Pentium(R) CPU G2020 @ 2.90GHz with cache size 3072KB cache, and with 8GB of memory and are connected with 1 Gbps connections running Ubuntu Precise Pangolin (12.04.4 LTS). The final tests were all done on Solr 4.7.2 and ZooKeeper 3.4.6.

Unless otherwise indicated the default configuration is: 2 Nodes, 2 Shards, 2 threads. In this case each node is responsible for 1 shard and data files are split equally between shards (i.e., each 6 files). A separate node hosts the query application that uses 2 separate thread loops to repeatedly query Solr nodes (each thread connects to one of the hosts). It uses a query file containing 63486 queries and loads the file and uses random access to choose the consecutive queries (randomized). Queries are simple, for exam-

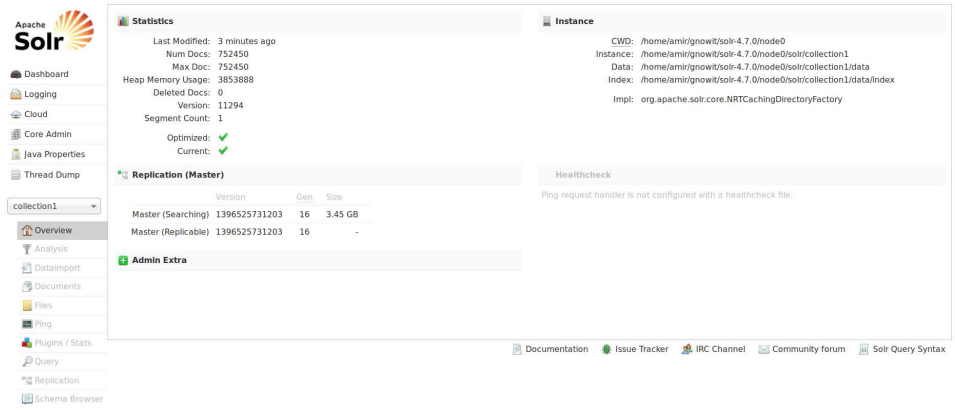

Figure 1: 2 Node SolrCloud setup on localhost with 2 Shards and 12 data files indexed.

ple: "q=Business+AND+Technology+AND+Ottawa" with no condition on other fields. The loop measures the time after 1000 queries have been performed and prints the number of doc results returned as well as time to a file and to the stdout (example: "ThreadID: 1, 1000 Queries, Time passed: 9223, #doc results: 7402"). The numbers reported are for each thread. In this dataset and queryset, an average each query returns 7.2–7.6 docs. If there more threads (T) than number of nodes (N), the threads are assigned to the same nodes in round robin (i.e., $T0 \rightarrow N0, T1 \rightarrow N1, T2 \rightarrow N0, T3 \rightarrow N1$,..). If the number of nodes is larger than shards these nodes the nodes become responsible for the same shards and act as a back up if one fails. See Figure 2.

(a) SolrCloud with 2 nodes and 2 shards (b) SolrCloud with 5 nodes and 3 shards

Figure 2: Nodes and shards are setup in round robin fashion. If there are more nodes than shards they become a backup node.

In these experiments I report the time (in ms) taken to answer 1000 queries to each thread. In case the number of nodes or shards are different they are

highlighted in the graph.

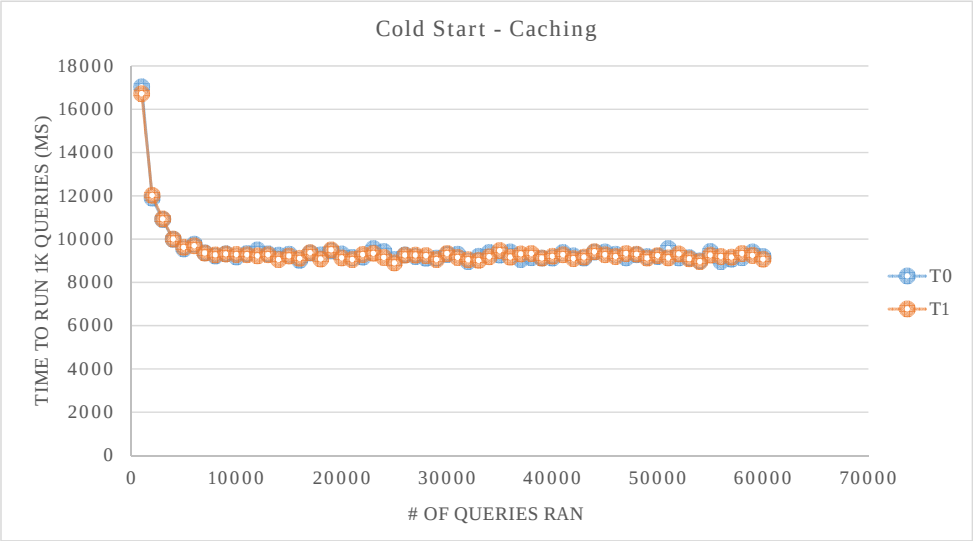

Figure 3: Cold start performance of SolrCloud

Figure 3 represents a cold start scenario where the nodes have just been deployed and cache has not been built yet. As the figure shows after 10000 queries (per thread) the system becomes stable. The initial delay experienced can sometimes be as high as 70–80 seconds.

Figure 4 shows the performance of SolrCloud in case of increasing the node number with the same shard number, and stressing the system by increasing the number of threads querying. In this scenario the nodes become backup for each other (similar to Figure 2(b)) in which case reliability is increased. One of the main interesting things seen in the results is that while the increase in the number of threads increases the delay, the increase is not linear meaning more threads in the end will be able to perform more queries. This suggest the bottleneck is not necessarily just on Solr search core but transport and other delays are also to blame. Increasing the number of nodes as expected improves the performance, as the first node that can answer the query answers it and the load is more evenly distributed by zookeepers (at least that's what I expect but I haven't that carefully studied the documentation). The yield is not very high but the added reliability is definitely a bonus.

I sometimes saw erratic behavior in the timing, e.g., seeing an extraordinarily

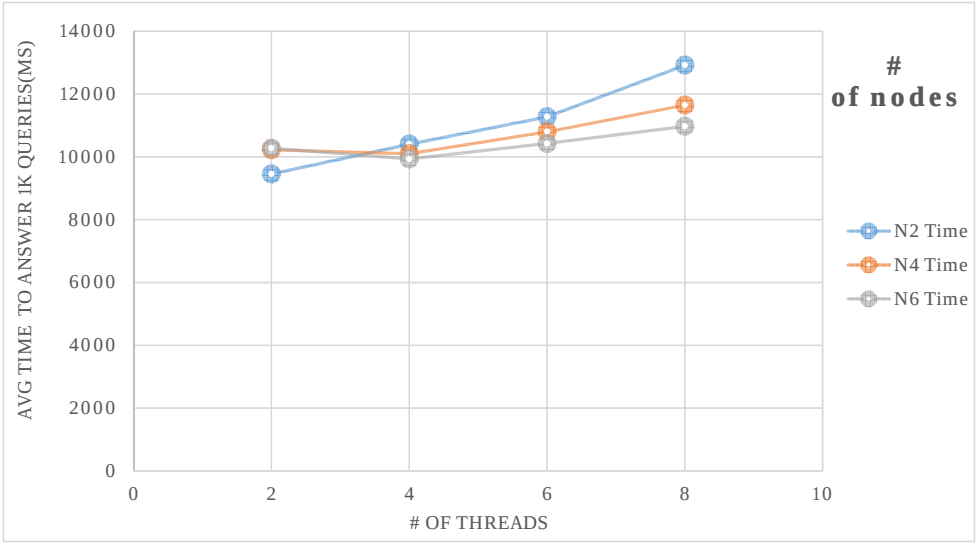

Figure 4: Increasing the node number with the same shard number, and stressing the system by increasing the number of threads querying

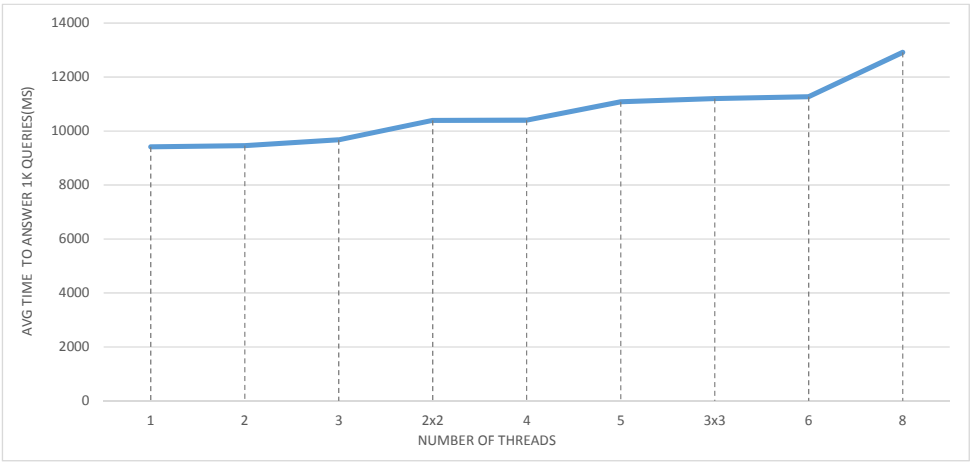

Figure 5: Thread Number vs. Node x Thread number

long processing time (e.g., 700s) when the normal processing time is 9-10s, suggesting that SolrCloud was performing a particularly time consuming task but I have not gotten to study the exact reason yet.
I suspected that running too many query threads on a single node might

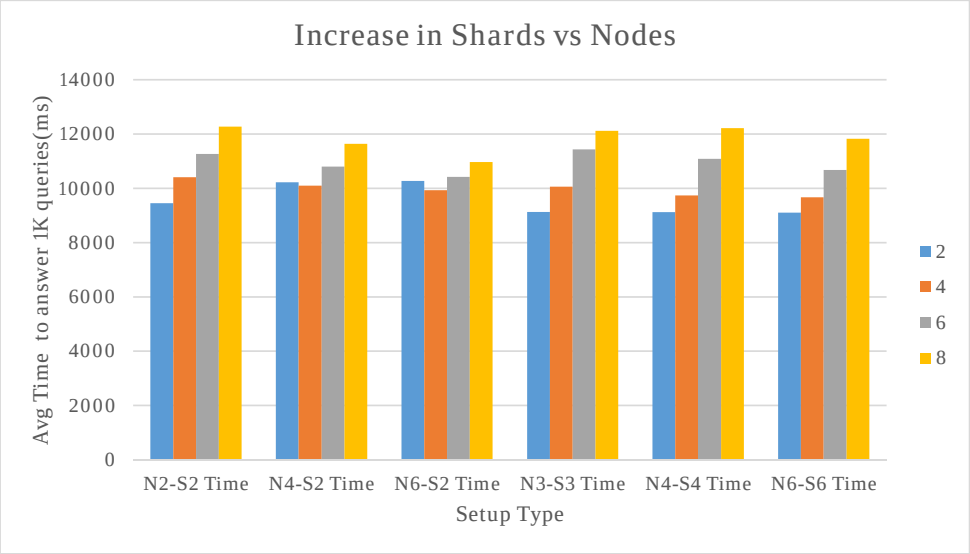

Figure 6: Increasing the number of Shards and Nodes

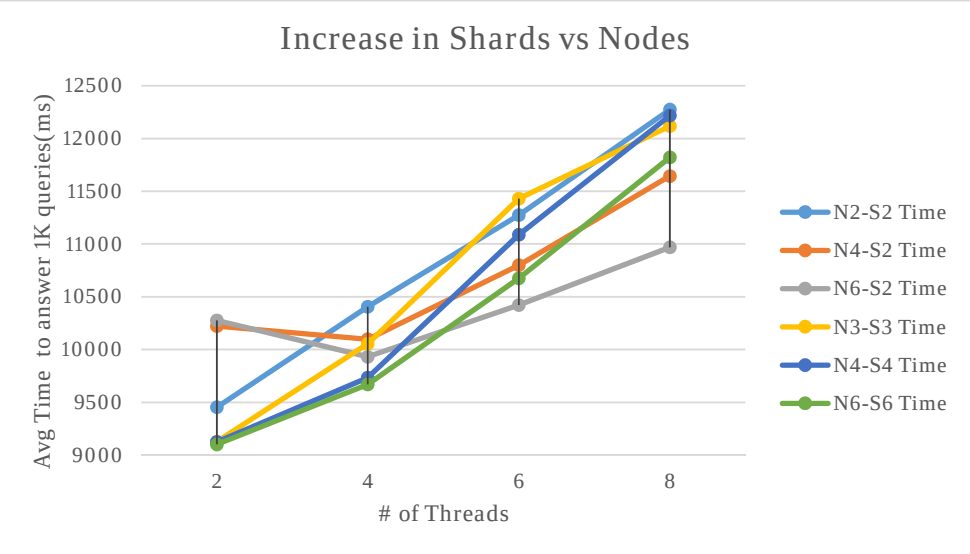

Figure 7: Increasing the number of Shards and Nodes - Detailed

result in slow downs due to local processing or network use, meaning there was a bottleneck on the query node. As a result, I decide to see whether

having the same number of overall threads spread on nodes will make a difference. This is the case shown in Figure 5. 2x2 shows a scenario where 2 nodes each running 2 threads which is compared against a single node running 4 threads. As you can see the difference is minimal and single node scenario is slightly better. This might be due to maintenance and reuse of connection pools on Solr nodes and clients (i.e., very small overhead of maintaining connections to 2 nodes vs. 1 node).
Figure 6 shows the effects of increasing the number of shards (which also requires increase in the number of nodes). This means each node has a smaller index file to search. The effects are not very big and are harder to translate. In some cases it results in better performance and sometimes in lower performance. Figure 7 shows the same graph with a smaller range to show the differences.

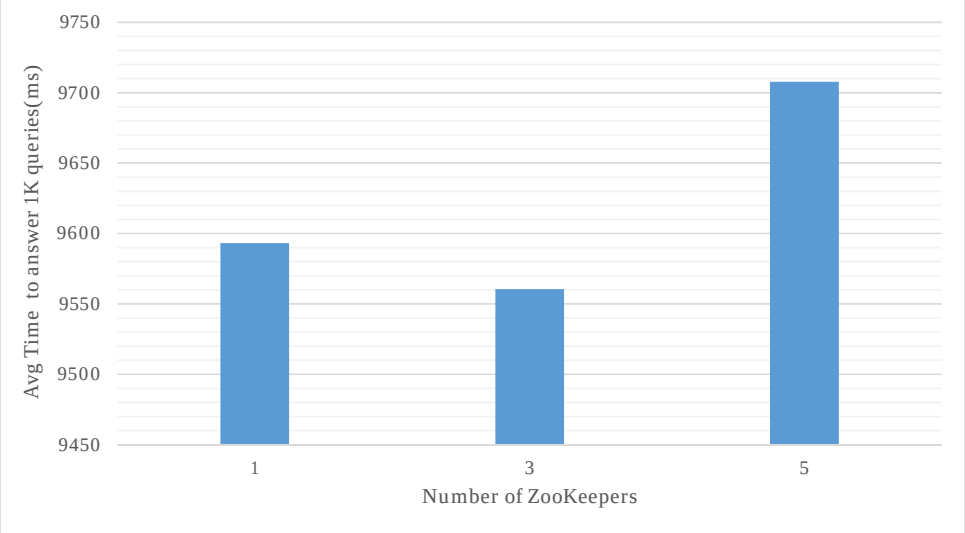

Figure 8: 2 Node SolrCloud setup on localhost with 2 Shards and 12 data files indexed

Note that the best use of shards is by doing smarter document and query routing which I did not get to experiment with. Solr offers the ability to specify the router implementation used by a collection by specifying the router.name parameter when creating your collection. If you use the "compositeId" router, you can send documents with a prefix in the document ID which will be used to calculate the hash Solr uses to determine the shard a

document is sent to for indexing (see Solr documentation).
Figure 8 shows the impact of having more or less ZooKeepers. 3 seems to be the best configuration. This might be because 1 ZooKeeper can get too many requests by processing while 5 nodes would require too much coordination among themselves. This requires a better study of how ZooKeepers work which I did not have the time to do.

6 Recommendation

This is not a complete analysis so take these with a grain of salt.

- Deploying more threads can improve the overall performance of the system. As a result have a number threads that do the querying instead of a single one.
- Having query threads on multiple nodes doesn't offer a benefit, other than redundancy.
- Even if you have 1 server, split the data and have 2 or more Solr nodes on that server. It improves the performance and even if one crashes you keep getting partial results. Plus you can have redundancy. Disk space is not very expensive.
- Having redundant nodes improves the performance and reliability. Start redundant nodes on servers.
- Have 3 ZooKeepers. It seems to offer good performance and redundancy but the performance gains are small. REdundancy is the main reason you would want to do this.

7 Future Work

- Experiments with the cache size can prove very important.
- Experiment with document routing.
- Isolate the reason for sometimes erratic behavior (very rare).
- Build a custom solution for intelligent query and document routing.

- More exhaustive test to ensure results are correct, and tests with many other possible parameters.

8 Appendix

8.1 Deploy Log Sample

This deployment has 6 Nodes, 6 Shards, and 3 ZooKeepers with 12 files equally distributed among the nodes.

Deployment Log

```
[amir][node-02][/local/amir/solrcloud/scripts] ./deploynodes.sh

================================================================================

Solr Path:  /local/amir/solrcloud/solr-4.7.2
Number of nodes:  6
Number of shards:  6
Standalone zookeepers?yes=1:  1

Make sure config.sh reflects your setup. This script will remove any existing
nodes named: nodeThey are named node0-...-node5 in solr path:
/local/amir/solrcloud/solr-4.7.2, with /local/amir/solrcloud/data/solrn#.xml
data files inside the /local/amir/solrcloud/data folder.
This script also assumes that you have the same setup on every machines
listed in hosts parameter. Whether there are standalone zookeepers should be
configured in the config.sh file. Make sure you setup your private/public key
on the servers, otherwise you end up inputing your password a lot!

================================================================================

Continue? y/ny

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-03
/local/amir/solrcloud/solr-4.7.2
```

```
"Backing up node0 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 0

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-04
/local/amir/solrcloud/solr-4.7.2
"Backing up node1 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 1

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-05
/local/amir/solrcloud/solr-4.7.2
"Backing up node2 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 2

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-06
/local/amir/solrcloud/solr-4.7.2
"Backing up node3 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 3

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-07
/local/amir/solrcloud/solr-4.7.2
"Backing up node4 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 4

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
kill: No such process
Existing solrnode services KILLED on node-01
/local/amir/solrcloud/solr-4.7.2
```

```
"Backing up node5 xml data files with name
"/local/amir/solrcloud/data/solrn"Removed Solr node 5

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-03
Removed zk node 1

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-04
Removed zk node 2

.............................................
script rmremotenodes.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-05
Removed zk node 3

.............................................
script newzk.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-03
DEPLOYED

.............................................
script newzk.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-04
DEPLOYED

.............................................
script newzk.sh is being sourced ...
kill: No such process
Existing zookeeper services KILLED on node-05
DEPLOYED

.............................................
script newzk.sh is being sourced ...
/local/amir/solrcloud/zookeeper-3.4.6
JMX enabled by default
Using config: /local/amir/solrcloud/zookeeper-3.4.6/bin/../conf/zoo1.cfg
Starting zookeeper ... STARTED
```

```
.................................................
script newzk.sh is being sourced ...
/local/amir/solrcloud/zookeeper-3.4.6
JMX enabled by default
Using config: /local/amir/solrcloud/zookeeper-3.4.6/bin/../conf/zoo2.cfg
Starting zookeeper ... STARTED

.................................................
script newzk.sh is being sourced ...
/local/amir/solrcloud/zookeeper-3.4.6
JMX enabled by default
Using config: /local/amir/solrcloud/zookeeper-3.4.6/bin/../conf/zoo3.cfg
Starting zookeeper ... STARTED

Sleep to allow for Quorom to be setup
-DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182

------------------------------------------------
node-03.cs.mcgill.ca is a normal node.
Creating a new node: node0
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node0
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8983 -DzkHost=node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2182 -DnumShards=6
-Dbootstrap_confdir=./solr/collection1/conf -Dcollection.configName=amirsolrconfig
Started simple solr check the log if you want to make sure an exception didn't occur
Sleep to allow time for bootup

------------------------------------------------
node-04.cs.mcgill.ca is a normal node
Creating a new node: node1
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node1
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8984 -DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182
Started simple solr check the log if you want to make sure an exception didn't occur
Sleep to allow time for bootup

------------------------------------------------
```

```
node-05.cs.mcgill.ca is a normal node
Creating a new node: node2
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node2
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8985 -DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182
Started simple solr check the log if you want to make sure an exception didn't occur
Sleep to allow time for bootup

------------------------------------------------
node-06.cs.mcgill.ca is a normal node
Creating a new node: node3
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node3
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8986 -DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182
Started simple solr check the log if you want to make sure an exception didn't occur
Sleep to allow time for bootup

------------------------------------------------
node-07.cs.mcgill.ca is a normal node
Creating a new node: node4
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node4
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8987 -DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182
Started simple solr check the log if you want to make sure an exception didn't occur
Sleep to allow time for bootup

------------------------------------------------
node-01.cs.mcgill.ca is a normal node
Creating a new node: node5
solr path is /local/amir/solrcloud/solr-4.7.2
/local/amir/solrcloud/scripts
DONE deploying new node: node5
Starting Simple Solr, with paramsnot mandatory:
-Djetty.port=8988 -DzkHost=node-03.cs.mcgill.ca:2181,
node-03.cs.mcgill.ca:2181,node-03.cs.mcgill.ca:2182
Started simple solr check the log if you want to make sure an exception didn't occur
```

```
Sleep to allow time for bootup


Index the default data in config file? y/ny
INDEXING data.

.................................................
/local/amir/solrcloud/data/solrn0.xml /local/amir/solrcloud/data/solrn1.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8983/solr/update using content-type text/xml..
POSTing file solrn0.xml
POSTing file solrn1.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/update..
Time spent: 0:02:33.711

.................................................
/local/amir/solrcloud/data/solrn2.xml /local/amir/solrcloud/data/solrn3.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8984/solr/update using content-type text/xml..
POSTing file solrn2.xml
POSTing file solrn3.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8984/solr/update..
Time spent: 0:02:28.556

.................................................
/local/amir/solrcloud/data/solrn4.xml /local/amir/solrcloud/data/solrn5.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8985/solr/update using content-type text/xml..
POSTing file solrn4.xml
POSTing file solrn5.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8985/solr/update..
Time spent: 0:02:35.220

.................................................
/local/amir/solrcloud/data/solrn6.xml /local/amir/solrcloud/data/solrn7.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8986/solr/update using content-type text/xml..
POSTing file solrn6.xml
POSTing file solrn7.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8986/solr/update..
Time spent: 0:02:37.014
```

```
................................................
/local/amir/solrcloud/data/solrn8.xml /local/amir/solrcloud/data/solrn9.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8987/solr/update using content-type text/xml..
POSTing file solrn8.xml
POSTing file solrn9.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8987/solr/update..
Time spent: 0:02:33.083

................................................
/local/amir/solrcloud/data/solrn10.xml /local/amir/solrcloud/data/solrn11.xml
SimplePostTool version 1.5
Posting files to base url http://localhost:8988/solr/update using content-type text/xml..
POSTing file solrn10.xml
POSTing file solrn11.xml
2 files indexed.
COMMITting Solr index changes to http://localhost:8988/solr/update..
Time spent: 0:02:41.364
```
