Datacenter Simulation Methodologies
Web Search

Tamara Silbergleit Lehman, Qiuyun Wang, Seyed Majid Zahedi
and Benjamin C. Lee

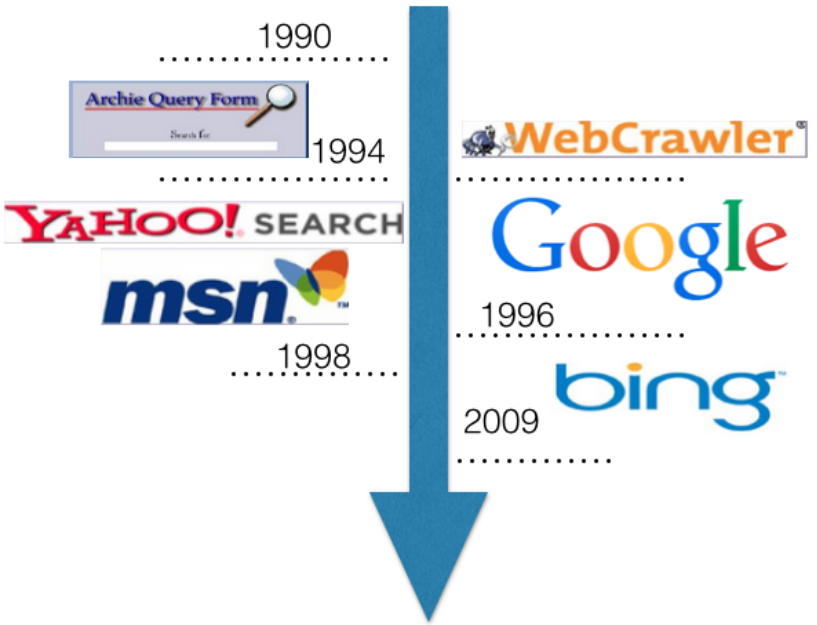
Tutorial Schedule

Time	Topic
09:00 - 10:00	Setting up MARSSx86 and DRAMSim2
10:00 - 10:15	Break
10:15 - 10:45	**Web search simulation**
10:45 - 11:15	GraphLab simulation
11:15 - 12:00	Spark simulation
12:00 - 13:00	Questions, Hands-on Session

Agenda

- Goals:
  - Be able to study real-world search engine that uses a large index, processes diverse queries
  - Be able to simulate search and queries
- Outline:
  - Introduce Apache Solr
  - Set up Apache Solr
  - Prepare Wikipedia search engine
  - Set up search on MARSSx86

Why Study Search?



- Computation and data migrate from client to cloud
- Search is a representative datacenter workload

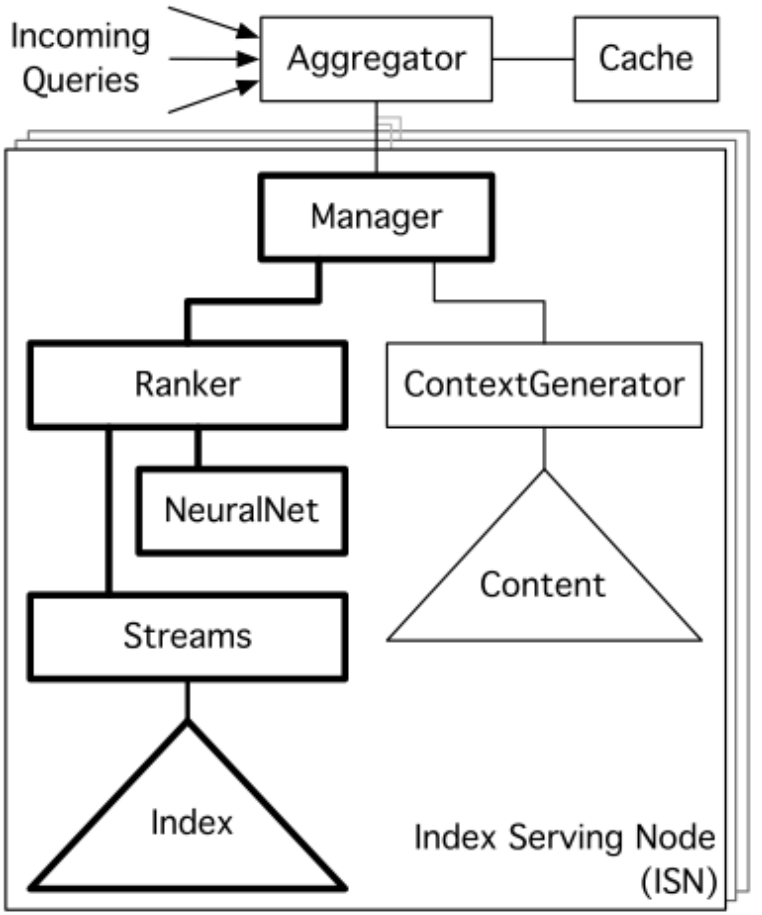
Why Study Search?

Search requires:

- large computational resources
- strict quality of service
- scalability, flexibility and reliability

Index Serving Node (ISN)

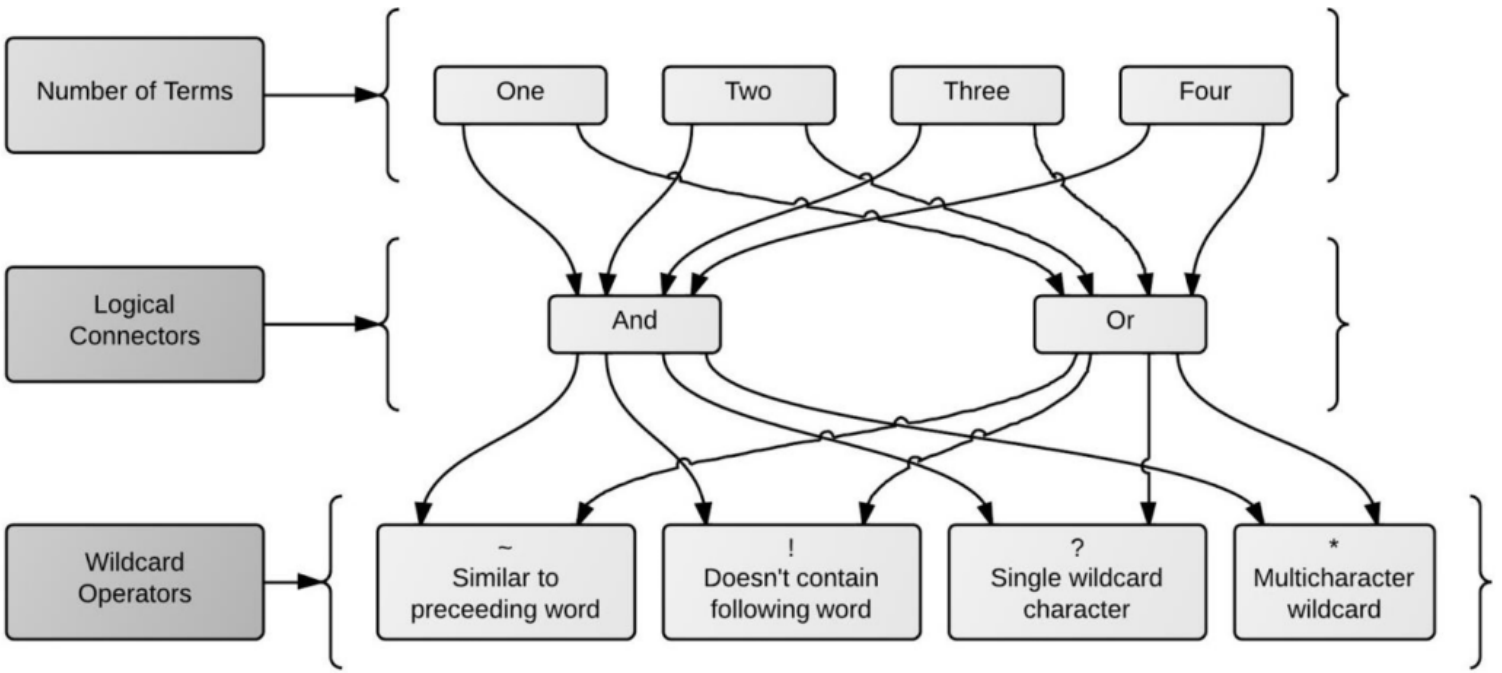
- Queries enter through the aggregator
- The aggregator distributes queries to ISNs
- Each ISN ranks the pages
- The ranker returns captions to the aggregator



"Web search using mobile cores" by V.J.Reddi *et al.*, ISCA, 2010

Search Query

- Search queries are important to the workload.
- Queries exhibit varying complexity and latency.



"Understanding Query Complexity and its Implications for Energy-Efficient Web Search", E. Bragg *et al.*, ISPLED, 2013

Search Engine

Possible ISN studies:

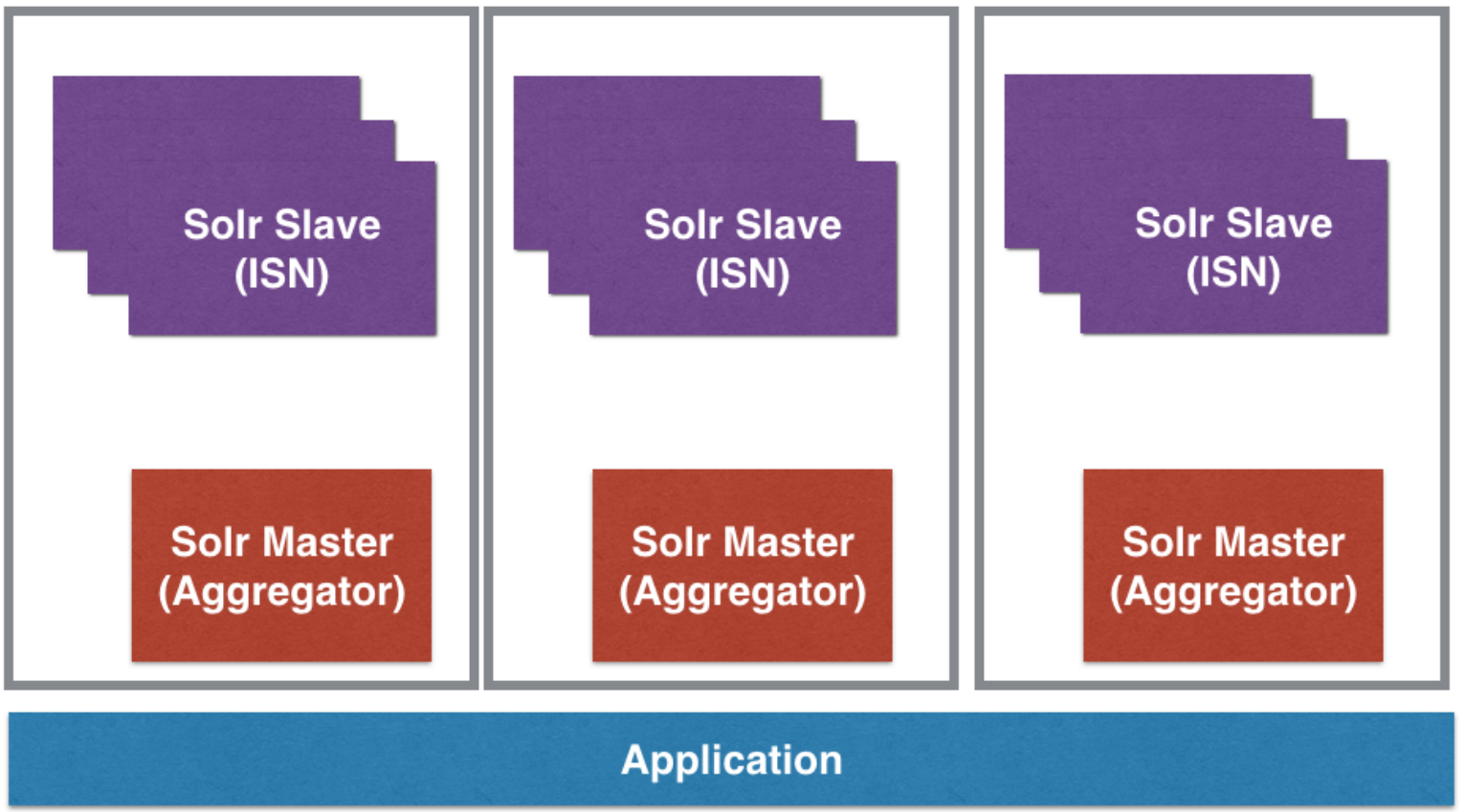
- Designing processor microarchitecture, memory systems
- Deploying machine learning algorithms
- Understanding query complexity and end-to-end behavior
- Managing resources and scheduling tasks

Apache Solr Engine

We set up Apache Solr on one Index Serving Node.

- Open source, well-documented, configurable search engine.
- Features:
  - Support full-text search
  - Near real time index
  - User-extensible caching
  - Distributed search for high-volume traffic
  - Server statistics logging
  - Scalability, flexibility and extensibility
- Rich API support: HTTP, XML, JSON, Python, Ruby, etc.

SolrCloud


Solr Slave (ISN)
Solr Slave (ISN)
Solr Slave (ISN)
Solr Master (Aggregator)
Solr Master (Aggregator)
Solr Master (Aggregator)
Application

Apache Solr Engine Users

Other Notable Users

AT&T	Instagram	Buy.com
Ticketmaster	Netflix	The Echo Nest
Chegg	Disney	Adobe
eBay	Internet Archive	SAP Hybris
Magento	IBM Websphere Commerce	Bloomberg
Comcast	MTV Networks	Travelocity

‘http://lucene.apache.org/solr/‘

Datacenter Simulation Methodologies
Web Search

Tamara Silbergleit Lehman, Qiuyun Wang, Seyed Majid Zahedi
and Benjamin C. Lee

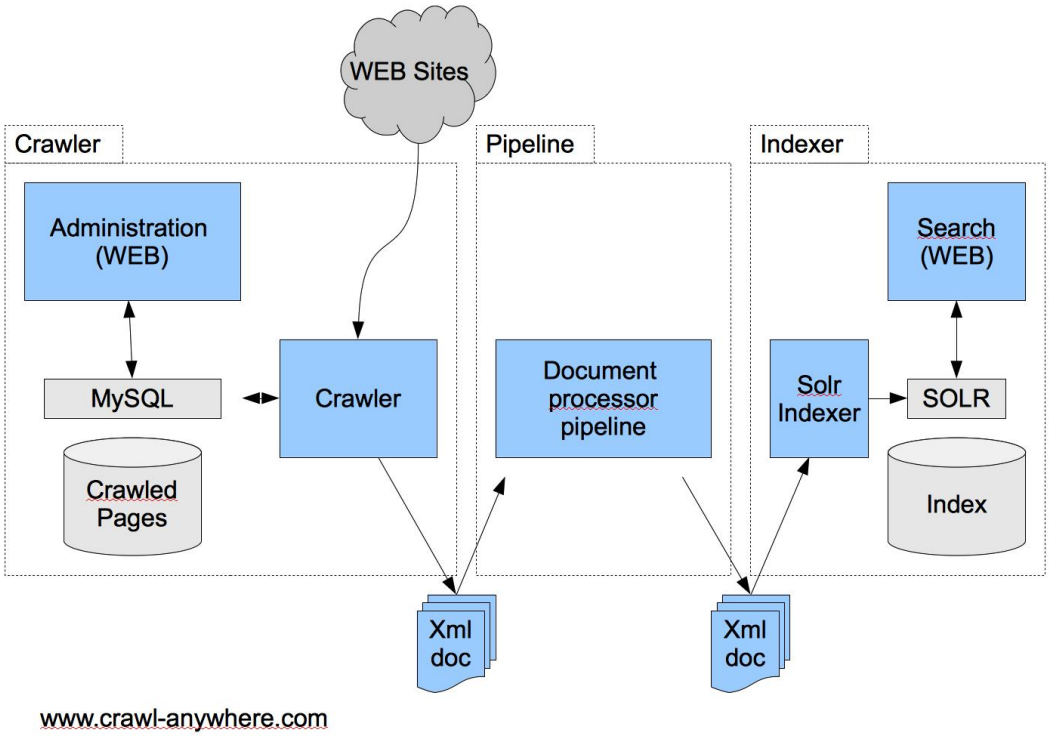
Agenda

- Goals:
  - Be able to study real-world search engine that uses a large index, processes diverse queries
  - Be able to simulate search and queries
- Outline:
  - Introduce Apache Solr
  - Set up Apache Solr
  - Prepare Wikipedia search engine
  - Set up search on MARSSx86

Duke UNIVERSITY

Introduce Apache Solr

- A fast, open-source Java search server.
- Easily create search engines for websites, files, databases.



Set up Apache Solr: Download and Install

- Open the image with QEMU:

```
$ qemu-system-x86_64 -m 4G -drive file=demo.
    qcow2,cache=unsafe -nographic
```

- Getting started!
  Download a version of Solr from
  http://lucene.apache.org/solr/ into the image.

```
# mkdir solr-small
# cd solr-small
# wget http://mirrors.advancedhosters.com/
    apache/lucene/solr/4.10.2/solr-4.10.2.zip
# unzip solr-4.10.2.zip
```

Set up Apache Solr: Install Required Libraries

- Setup Java 1.7 to default Java.

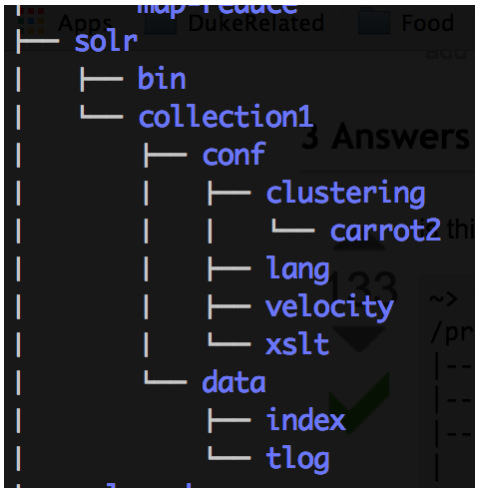
```
# sudo apt-get update
# sudo apt-get install openjdk-7-jdk
```

- Install curl command to submit HTTP requests.

```
# sudo apt-get install curl
```

Set up Apache Solr: Directory Overview

- Solr directory (an example for kernel - collection 1):

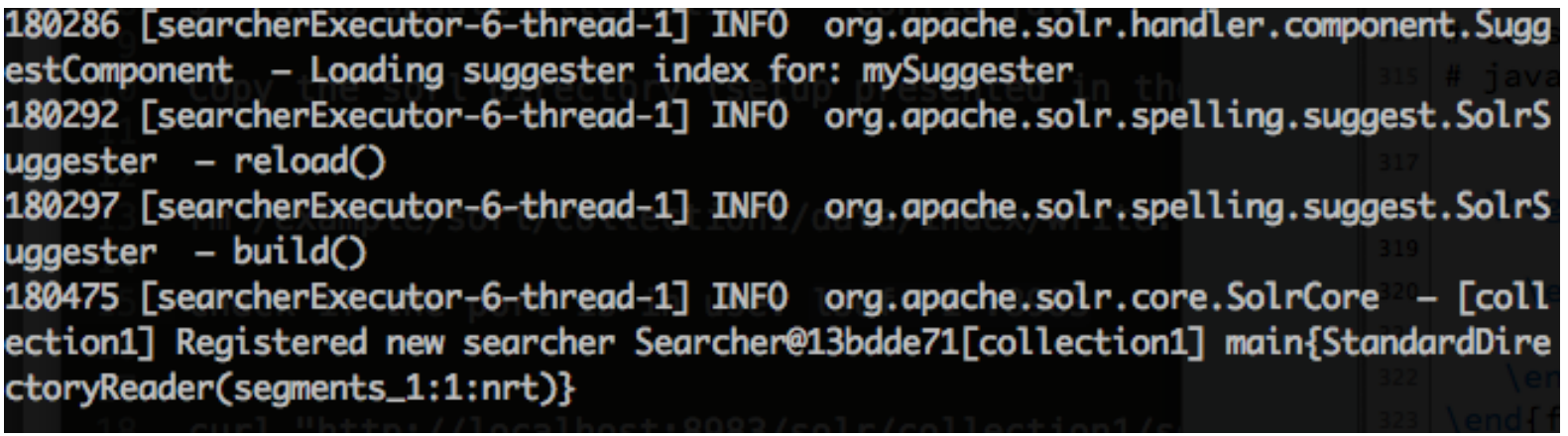


  - binary files:
    start.jar: start the search engine
    post.jar: index data
  - configuration files:
    solrconfig.xml, data-config.xml, schema.xml, etc.
  - data index

Set up Apache Solr: Start Engine

- Launch Solr Engine with the example configuration, run

```
# cd solr-4.10.2/example
# java -jar start.jar &
```



Set up Apache Solr: Check if Solr is Running

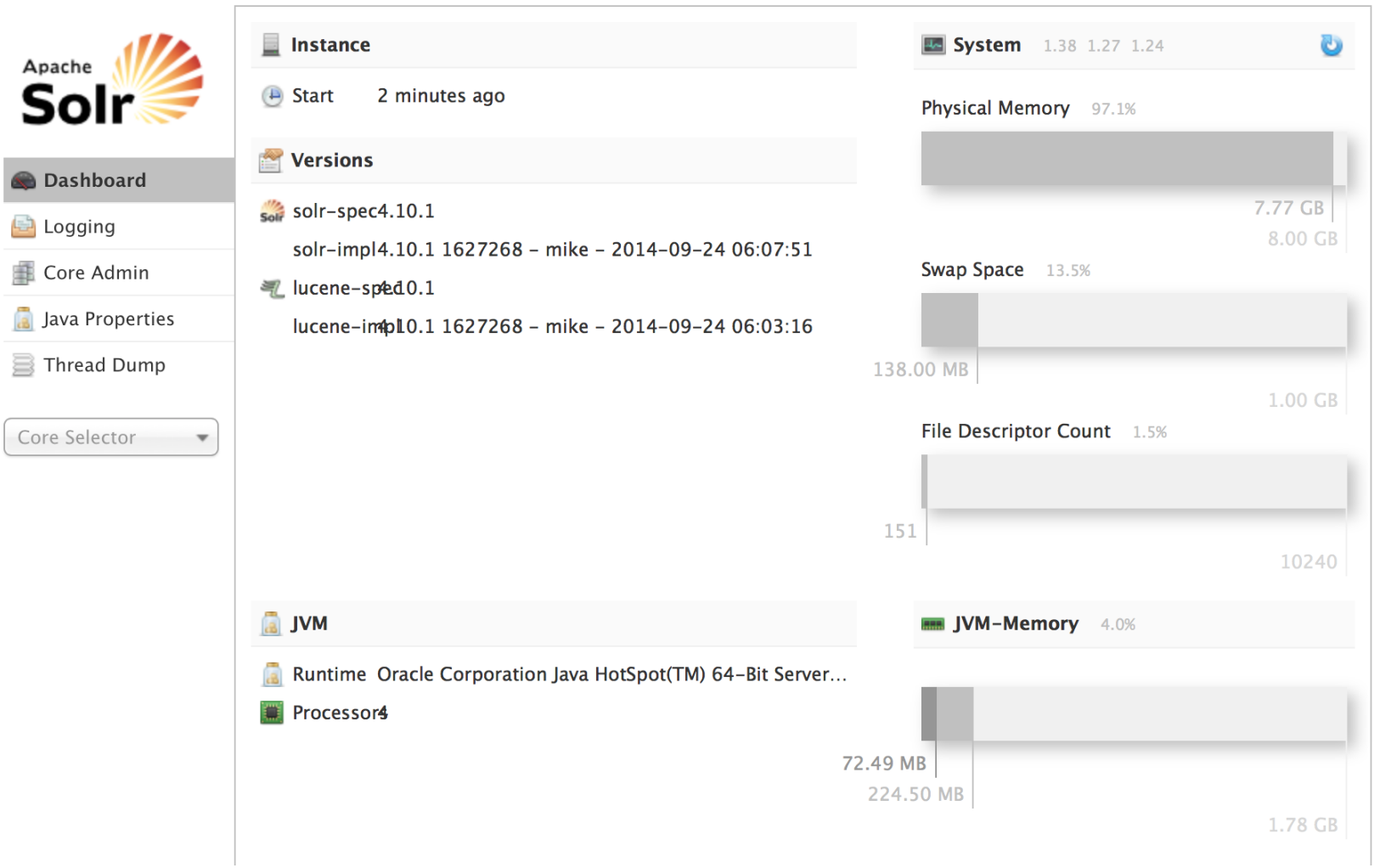
- No Java error message. If everything is setup correctly, a search engine will be running on port 8983. We could use a command to check the port:

```
# lsof -i :8983
```

```
root@ubuntu:~/solr-small/solr-4.10.2/example# lsof -i :8983
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java    1018 root  132u  IPv6   8044      0t0  TCP *:8983 (LISTEN)
```

Set up Apache Solr: Check if Solr is Running

```
# http:localhost:8983/solr/
```



Set up Apache Solr: Index XML Documents

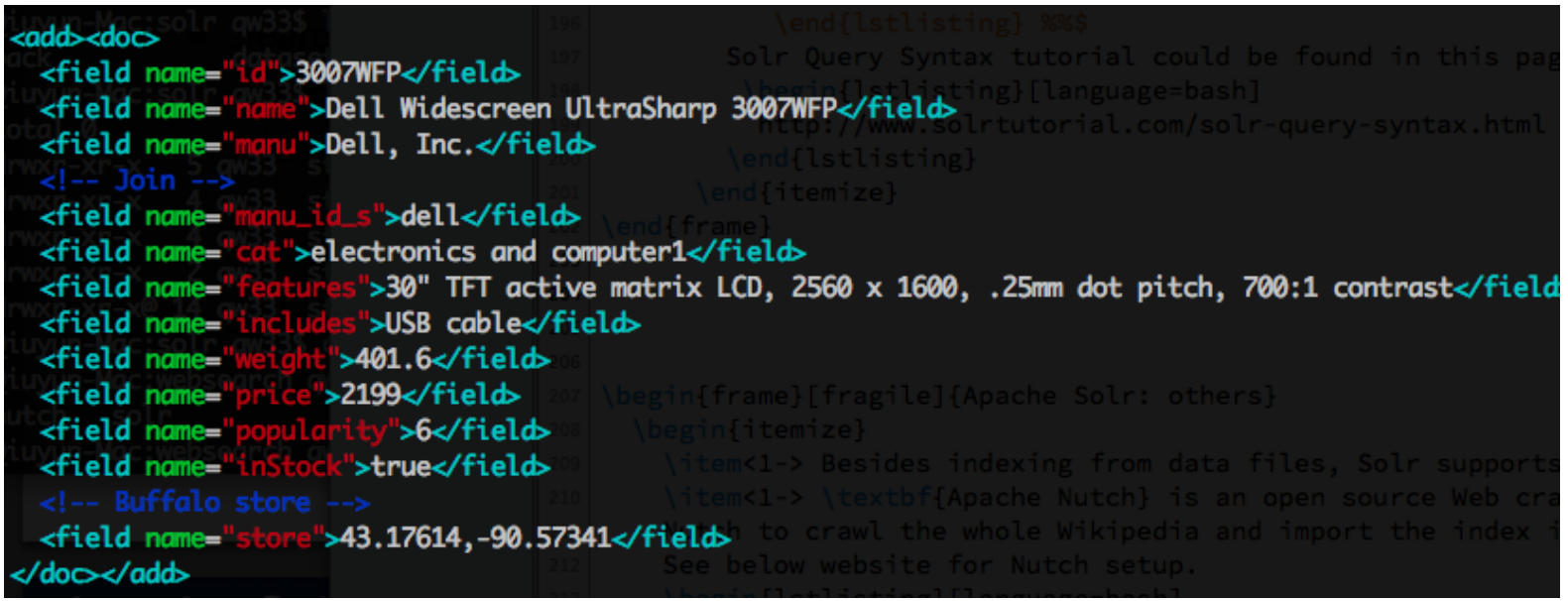
```
# cd solr-4.10.2/example/exampledocs
```

- Create search index for XML documents:

```
qw33@frigate:~/websearch/solr/solr-4.10.1/example/exampledocs$ ls
books.csv              ipod_video.xml     monitor.xml   solr-word.pdf
books.json             manufacturers.xml  mp500.xml     solr.xml
gb18030-example.xml    mem.xml            post.jar      test_utf8.sh
hd.xml                 money.xml          post.sh       utf8-example.xml
ipod_other.xml         monitor2.xml       sd500.xml     vidcard.xml
```

Set up Apache Solr: Index XML Documents

- monitor.xml:



```
<add><doc>
  <field name="id">3007WFP</field>
  <field name="name">Dell Widescreen UltraSharp 3007WFP</field>
  <field name="manu">Dell, Inc.</field>
  <!-- Join -->
  <field name="manu_id_s">dell</field>
  <field name="cat">electronics and computer1</field>
  <field name="features">30" TFT active matrix LCD, 2560 x 1600, .25mm dot pitch, 700:1 contrast</field
  <field name="includes">USB cable</field>
  <field name="weight">401.6</field>
  <field name="price">2199</field>
  <field name="popularity">6</field>
  <field name="inStock">true</field>
  <!-- Buffalo store -->
  <field name="store">43.17614,-90.57341</field>
</doc></add>
```

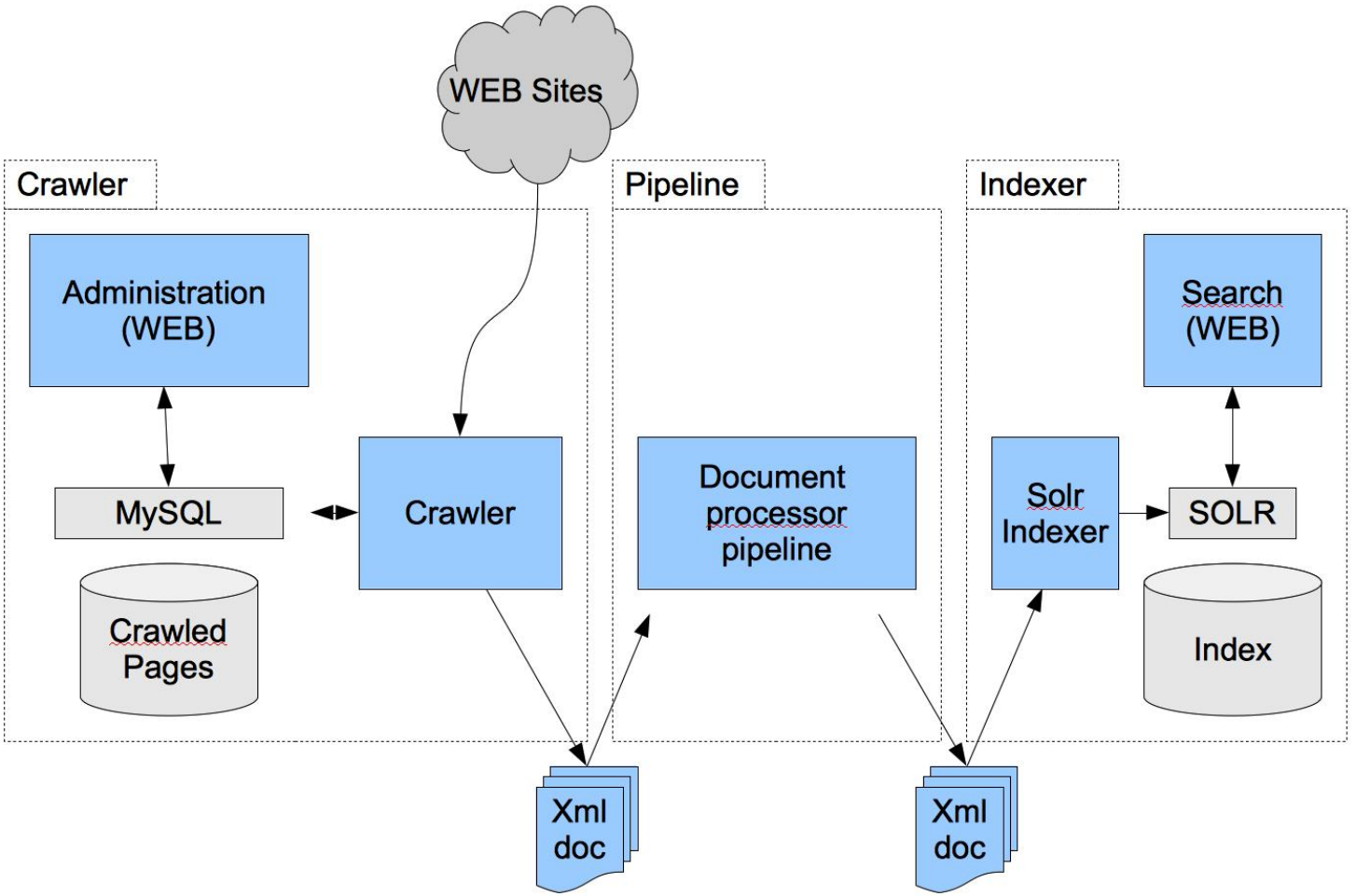
Index one XML document:

```
# ./post.sh monitor.xml
```

Index all XML documents:

```
# ./post.sh *.xml
```

Set up Apache Solr: Index XML Documents



Set up Apache Solr: Submit a Search Query

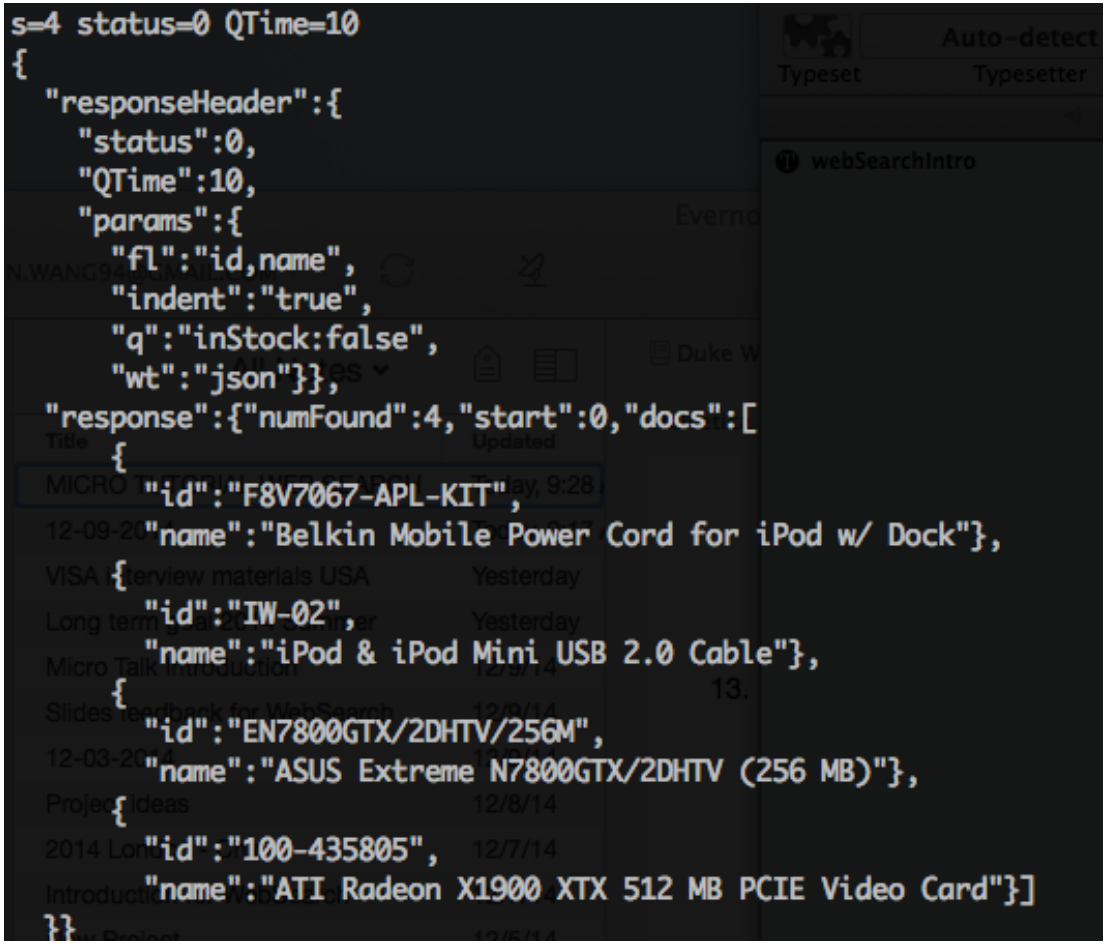
- Submit an example query to retrieve name and id of all documents with inStock=false:

```
# curl "http://localhost:8983/solr/
    collection1/select?q=inStock:false&wt=json
    &fl=id,name&indent=true"
```

  - Kernel name: collection1
  - Select operator: inStock=false
  - Return format: json (support Json, XML)
  - Return fields: id, name
  - Return format with indent on

Set up Apache Solr: Submit a Search Query

- Return from the command:



```
s=4 status=0 QTime=10
{
  "responseHeader":{
    "status":0,
    "QTime":10,
    "params":{
      "fl":"id,name",
      "indent":"true",
      "q":"inStock:false",
      "wt":"json"}},
  "response":{"numFound":4,"start":0,"docs":[
      {
        "id":"F8V7067-APL-KIT",
        "name":"Belkin Mobile Power Cord for iPod w/ Dock"},
      {
        "id":"IW-02",
        "name":"iPod & iPod Mini USB 2.0 Cable"},
      {
        "id":"EN7800GTX/2DHTV/256M",
        "name":"ASUS Extreme N7800GTX/2DHTV (256 MB)"},
      {
        "id":"100-435805",
        "name":"ATI Radeon X1900 XTX 512 MB PCIE Video Card"}]
  }}
```

- Solr Query Syntax tutorial at this page:
  `www.solrtutorial.com/solr-query-syntax.html`

Set up Apache Solr: Crawl Datasets

- Solr indexes from data files or crawled websites.
- **Apache Nutch** is open-source web crawler. Use Nutch to crawl websites and then import the index into Solr.

- See below website for Nutch setup.
  `http://wiki.apache.org/nutch/NutchTutorial/`
  `http://opensourceconnections.com/blog/2014/05/24/crawling-with-nutch/`

Set up Wikipedia Search: Download Datasets

Wikipedia search is already set up in the image:

```
$ cd ~/solr.4.10.1/
```

The following steps are already done for you.

- Download wikimedia commons in XML format (11GB) and decompress (47GB).

```
$ wget http://dumps.wikimedia.org/enwiki
    /20140903/
```

```
$ bzip2 -d enwiki-20140903-pages-articles-
    multistream.xml.bz2
```

Set up Wikipedia Search: Data Import Handler

- Use DataImportHandler to index big dataset. Edit file:

```
$ vim example/solr/collection1/conf/data-
    config.xml
```

```
<dataConfig>
        <dataSource type="FileDataSource" encoding="UTF-8" />
        <document>
        <entity name="page"
                processor="XPathEntityProcessor"
                stream="true"
                forEach="/mediawiki/page/"
                url="/home/qw33/websearch/dataset/enwiki-20140903-pages-articles-multistream.xml"
                transformer="RegexTransformer,DateFormatTransformer"
                >
            <field column="id"        xpath="/mediawiki/page/id" />
            <field column="title"     xpath="/mediawiki/page/title" />
            <field column="revision"  xpath="/mediawiki/page/revision/id" />
            <field column="user"      xpath="/mediawiki/page/revision/contributor/username" />
            <field column="userId"    xpath="/mediawiki/page/revision/contributor/id" />
            <field column="text"      xpath="/mediawiki/page/revision/text" />
            <field column="timestamp" xpath="/mediawiki/page/revision/timestamp" dateTimeFormat="yyyy-MM-dd'T'hh:mm:ss'Z'" />
            <field column="$skipDoc"  regex="^#REDIRECT .*" replaceWith="true" sourceColName="text"/>
        </entity>
        </document>
</dataConfig>
```

Set up Wikipedia Search: Data Import Handler

- Register DataImportHandler in Solr configuration file:

```
$ vim example/solr/collection1/conf/
    solrconfig.xml
```

Set up Wikipedia Search: Data Import Handler

- Add DataImportHandler library:
  - Check if solr-dataimporthandler-*.jar is in directory
    `$ solr-4.10.2/dist`
  - Include the library by adding the following line to Solr configuration file: solrconfig.xml

```
<lib dir="../../../dist/" regex="solr-
    dataimporthandler-.*\.jar" />
```

Set up Wikipedia Search: Create the Index

- Ready to create the index for wikipedia dataset. Run:

```
$ curl "http://localhost:8983/solr/
    collection1/dataimport?command=full-import
    "
```

- Command returns immediately. Index is saved in directory: example/solr/collection1/data/index. This process takes 3-4 hours.

Prepare Search on MARSSx86: File Transfer

- Switch to MARSSx86 QEMU:

```
$ cd marss.dramsim
$ ./qemu/qemu-system-x86_64 -m 4G -drive file
   =demo.qcow2,cache=unsafe -nographic -
   simconfig demo.simcfg
```

- Copy search engine from physical machine into MARSSx86. Reduce time to create index. From inside the image, run:

```
# scp -r username@machine:solr-4.10.2 .
```

- Check and release write lock:

```
# rm /example/solr/collection1/data/index/
   write.lock
```

Prepare Search on MARSSx86: Start Wikipedia Engine

- Start the search engine:

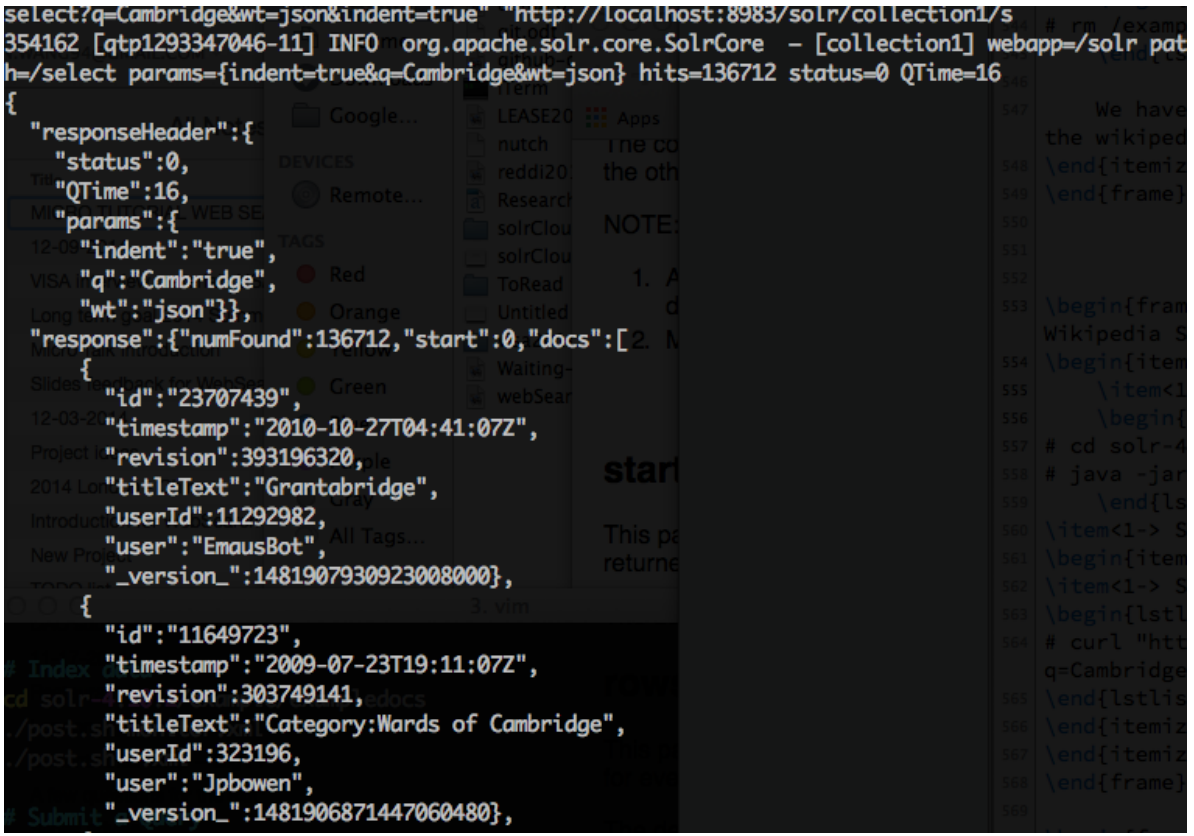
```
# cd solr-4.10.1/example
# java -jar start.jar &
```

- Submit single-word for query:

```
# curl "http://localhost:8983/solr/
    collection1/select?q=Cambridge&wt=json&
    indent=true"
```

Prepare Search on MARSSx86: Start Wikipedia Engine

- Display the top 10 responses
- Count all the hits
- Return the response time in ms



Prepare Search on MARSSx86: Start Wikipedia Engine

- Submit phrase for query:

```
# curl "http://localhost:8983/solr/
    collection1/select?q=\"Computer+
    architecture\"&wt=json&indent=true"
```

Prepare Search on MARSSx86: Warm Up Queries

- Configure warm up queries with first search events. Edit /solr-4.10.1/example/solr/collection1/conf/solrconfig.xml

```
<listener event="firstSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <lst> <str name="q">Australia</str>
              <str name="start">0</str>
              <str name="rows">100000</str>
        </lst>
        <lst> <str name="q">university</str>
              <str name="start">0</str>
              <str name="rows">100000</str>
        </lst>
        <lst> <str name="q">rabbit</str>
              <str name="start">0</str>
              <str name="rows">100000</str>
        </lst>
      </arr>
</listener>
```

Prepare Search on MARSSx86: Create Checkpoints

- Prepare PTLSim calls: create_checkpoint.c

```
#include <stdio.h>
#include <stdlib.h>
#include "ptlcalls.h"
int main(int argc, char ** argv){
    if (argc >1){
        char * chk_name = argv[1];
        printf("Creating checkpoint %s\n",
            chk_name);
        ptlcall_checkpoint_and_shutdown(
            chk_name);
    }
    else{
        printf("No checkpoint name was
            provided.\n");
    }
    return EXIT_SUCCESS;
}
```

Prepare Search on MARSSx86: Create checkpoints

- PTLSim: stop_sim.c

```
#include "ptlcalls.h"
#include <stdio.h>
int main(int argc, char ** argv){
    printf("Stopping simulation\n");
    ptlcall_switch_to_native();
    return EXIT_SUCCESS;
}
```

Compile those functions with gcc into binary files.

```
# make
```

- Prepare search queries: singleWord.sh

```
#!/bin/bash
curl "http://localhost:8983/solr/collection1/
   select?q=rabbit&wt=xml"
}
```

Prepare Search on MARSSx86: Create Checkpoints

- Run create_checkpoint binary and give a checkpoint name

```
cd ~/; ~/create_checkpoint singleWord; bash
    tests/singleWord.sh; ~/stop_sim
```

Prepare Search on MARSSx86: Create Checkpoints

- Put all together in the create_checkpoint.py.
  - Change directory into /solr/example
  - Start the search engine
  - Wait for it to set up
  - Run create checkpoint binary
  - Run the search query
  - Stop the simulation

```
cd websearch/solr-4.10.1/example && java -jar
    start.jar &> out.log & sleep 400 & cd ~/;
    ~/create_checkpoint singleWord; bash
   tests/singleWord.sh ; ~/stop_sim
```

Prepare Search on MARSSx86: Simulate Queries

- Add the checkpoint singleWord to the configuration file: marss.dramsim/util/util.cfg.

```
[suite micro2014]
checkpoints= helloWorld, singleWord
```

- Run the query from created checkpoint

```
$ cd marss.dramsim
$ python util/run_bench.py -c util/util.cfg -
    d testdir --chk-name=singleWord demo
```

Agenda

- Goals:
  - Be able to study real-world search engine that uses a large index, processes diverse queries
  - Be able to simulate search and queries
- Outline:
  - Introduce Apache Solr
  - Set up Apache Solr
  - Prepare Wikipedia search engine
  - Set up search on MARSSx86

Tutorial Schedule

Time	Topic
09:00 - 10:00	Setting up MARSSx86 and DRAMSim2
10:00 - 10:15	Break
10:15 - 10:45	**Web search simulation**
10:45 - 11:15	GraphLab simulation
11:15 - 12:00	Spark simulation
12:00 - 13:00	Questions, Hands-on Session