Overview of LucidWorks Big Data Software
September 16, 2012

1. Module name: Overview of LucidWorks Big Data software

2. Scope

This module introduces the basic concepts and the overview of LucidWorks Big Data software that is specifically designed for searching, discovery, and analysis of massive content sets.

3. Learning objectives

By the end of this module, students will be able to:

1. Apply the LucidWorks Big Data software to Information Retrieval tasks as in the Manning, Raghavan, and Schutze textbook.
2. Index documents using curl commands
3. Search using curl commands and the admin interface.
4. Select appropriate parameters when using the LucidWorks enterprise search engine (based on Apache Solr).

5. Level of effort required

a) Class time: 1.5 hours

b) Student time outside class:

   a. Preparation/reading: 2-3 hours

6. Relationships with other modules:

Related to module 7-a, which is indexing and searching.

Related to Apache Solr module in IR Software Package Modules.

7. Prerequisite knowledge/skills required

Familiarity with JSON [1] and XML [2].

9. Body of knowledge (theory + practice; an outline that could be used as the basis for class lectures)

9.1 Overview:

LucidWorks Big Data is an integrated platform for solving information access problems which provides the tools for search, discovery, and analysis of massive content sets.

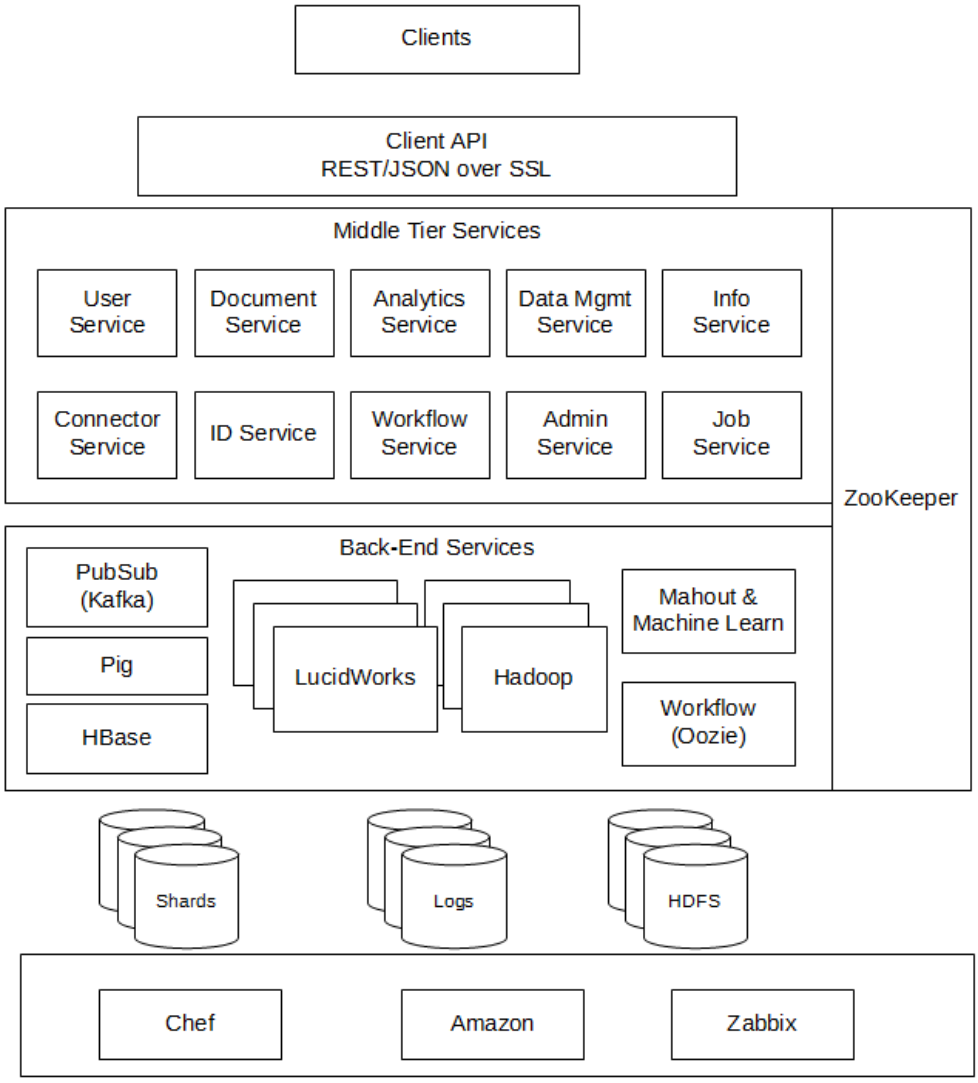
Built on leading open source components such as Apache Solr [3], Hadoop [4], Mahout [5] and others, in addition to the power of the LucidWorks Search Platform [6], it brings together previously disparate tools into a single tool suite with a unified, secure REST API for application integration and administration.

For additional background see [7].

9.2 Architecture [8]**:**

The picture below gives detailed information about several parts of the system that integrates several technologies. The core technologies are the LucidWorks Search Platform (built on Lucene and Solr) [6], and Apache Hadoop [9]. Surrounding are a number of other elements – Apache Mahout for analytics [10], Pig for Hadoop jobs [11], Zabbix [12], and Chef [13] for monitoring and provisioning.

.



The specific components of LucidWorks Big Data are described below [8].

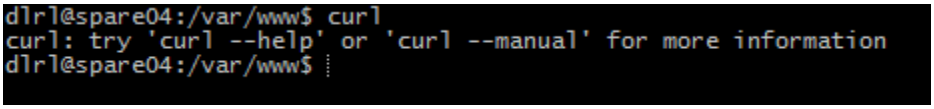
9.3 Software:

This software is installed in a server machine in the Digital Library Research Laboratory in Torg 2030 in Virginia Tech for the purpose of Virginia Tech Computer Science Fall 2012 CS5604 Course. We will be accessing the software using **cURL** [14] commands and through the admin interface [15] (http://fetcher.dlib.vt.edu:8888/solr/#/test_collection_vt) to search through indexed documents and to see the results.

9.4 How to use the software:

Please gain access to a **Unix system terminal** to be able to execute the commands. All Linux and Mac Systems have Unix terminals. If you are using windows and do not know how to gain access to one, please talk to the TA (tarekk@vt.edu) or ask at kiranch@vt.edu

To make sure your terminal can execute curl commands, type command 'curl' and it should give output like below screenshot.



9.5 JSON Format:

JSON (Java Script Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. Please note that every object in JSON is of the {key:value} format.

Example: The employees object is an array of 3 employee records (objects):

```
{
"employees": [
{ "firstName":"John" , "lastName":"Doe" },
{ "firstName":"Anna" , "lastName":"Smith" },
{ "firstName":"Peter" , "lastName":"Jones" }
]
}
```

For additional background see [1].

9.6 XML Format:

Extensible Markup Language (XML) defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

The above JSON object can be represented in XML as below.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
        <employees>
                <firstName>John</firstName>
                <lastName>Doe</lastName>
        </employees>
        <employees>
                <firstName>Anna</firstName>
                <lastName>Smith</lastName>
        </employees>
        <employees>
                <firstName>Peter</firstName>
                <lastName>Jones</lastName>
        </employees>
```

For additional background see [2].

By using the curl commands and through the admin interface, you will get more familiarity with the JSON and XML formats.

9.7 Indexing JSON type Documents:

Let's consider a command for indexing JSON documents and see how it works

```
curl -u administrator:foo -X POST -H 'Content-type: application/JSON' -d
 '{"docs":[{"id":"foo","body":"body 1","title":"title1"},{"id":"foo2","body":"body 2","title":"title2"}]}'
http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents | python -
mjson.tool
```

a) 'curl' at the beginning specifies that this is a curl command and everything else provides parameters sent to the URL (API) [16] http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents

b) Parameters (arguments):

- **-u, -user <username:password>** Specify the username and password to use for server authentication**.** Please use here the username and password you are given through email by the TA. The commands won't work without proper authentication
- **-X, --request <command>** Specifies a custom request method to use when communicating with the HTTP server. We are using HTTP request method POST here to communicate with the server.
- **-H, --header <header>** We use –H to force the header to specify the content-type as JSON
- **-d, --data <data>** Sends the specified data in a POST request to the HTTP Server. The API [17] (/sda/v1/client/collections/*collection*/documents) parameters present in the LucidWorks documentation are 'docs' and 'options'.
    - We use the docs parameters to specify the JSON documents that are going to be indexed. In the command above, there are two JSON

documents with fields ‘id’, ‘body’, ‘title’. Please remember that ‘id’ should be unique for each document.

- We are posting all the arguments to the URL http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents.
    - The part (http://fetcher.dlib.vt.edu:8341) specifies the hostname and portname.
    - The remaining part (/sda/v1/client/collections/test_collection_vt/documents) is the API entry point for document indexing. This API [17] as shown in the documentation takes a parameter ‘collection’ which we gave as *‘test_collection_vt’.* Please note that we will be using the same collection name throughout this module.
    - There are many more APIs for other tasks listed in the LucidWorks Big Data API Reference [16]
- | python -mjson.tool – this is used to format the output in a JSON format. This is optional and need not be present.

When the above command is executed, the output should be:

```
dlrl@spare04:~$ curl -u administrator:foo -X POST -H 'Content-type: application/json' -d  '{"doc
s":[{"id":"foo","body":"body 1","title":"title1"},{"id":"foo2","body":"body 2","title":"title2"}
]}' http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents | pyt
hon -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   581    0   478  100   103   2489    536 --:--:-- --:--:-- --:--:--  2529
{
    "children": [
        {
            "children": [],
            "collection": "test_collection_vt",
            "createTime": 1347887045922,
            "id": "test_collection_vt",
            "status": "SUCCEEDED",
            "throwable": null
        },
        {
            "children": [],
            "collection": "test_collection_vt",
            "createTime": 1347887045926,
            "id": "test_collection_vt",
            "properties": {
                "service-impl": "HBaseDocumentService"
            },
            "status": "NOT_SUPPORTED",
            "throwable": null
        }
    ],
    "collection": "test_collection_vt",
    "createTime": 1347887045921,
    "id": "test_collection_vt",
    "status": "SUCCEEDED",
    "throwable": null
}
dlrl@spare04:~$ |
```

If you see something as below, then there is something wrong with the command that is executed. I have deliberately specified a wrong collection name in the command above so you can see the error response.

```
dlrl@spare04:~$ curl -u administrator:foo -X POST -H 'Content-type: application/json' -d  '{"doc
s":[{"id":"foo","body":"body 1","title":"title1"},{"id":"foo2","body":"body 2","title":"title2"}
]}' http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection/documents | python
 -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   589  100   486  100   103   2843    602 --:--:-- --:--:-- --:--:--  2892
No JSON object could be decoded
dlrl@spare04:~$
```

Please see the API [17] for more examples on indexing json documents.

9.8 Workflows [18]:

A user communicates with all or some of these components in the architecture (9.2) through the workflows which are invoked through the cURL[8] API commands. Workflows [18] describe processes that should be performed on data.

Starting a workflow starts a specific job, which can be monitored or cancelled with the Jobs API [19]. The Workflows API [18] is a way to update the definitions for each workflow and start the workflow job

There are two types of workflows.

- **_etl (Extract, transform, load):**
    - The etl workflow extracts data from a source, performs some transformations, and finally loads the data to Solr.
    - This main workflow contains other internal workflows which are initiated by calling attributes [20]
    - A full etl workflow with all the internal workflows turned to true will execute all the internal components in the below order as in the picture below.
    - A normal etl workflow will extract, vectorize and index the data without executing any internal workflows.
        - For example: A normal etl workflow is as in example below.

          Please note the inputDir parameter value, where we are specifying the data loaded in to Hadoop [9]. All data that is given as input in an etl workflow should be loaded in to Hadoop [9] first and then given as inputDirectory.

```
curl -X POST -H 'Content-type: application/json' -d
'{"collection":"test_collection_vt","inputDir":"hdfs://128.173.49.66:
50001/input/reuters/*/*.txt","inputType":"text/plain"}'
http://fetcher.dlib.vt.edu:8341/sda/v1/workflow/_etl
```

The above example would give an output as below.

```
{"id":"0000094-120928210911692-oozie-hado-
W","workflowId":"_etl","createTime":1350349329000,"status":"RUN
NING","children":[],"throwable":null}
```

The state of the workflow can be checked at
http://spare04.dlib.vt.edu:11000/oozie/

All sub-workflows can be started using different parameters in [20]

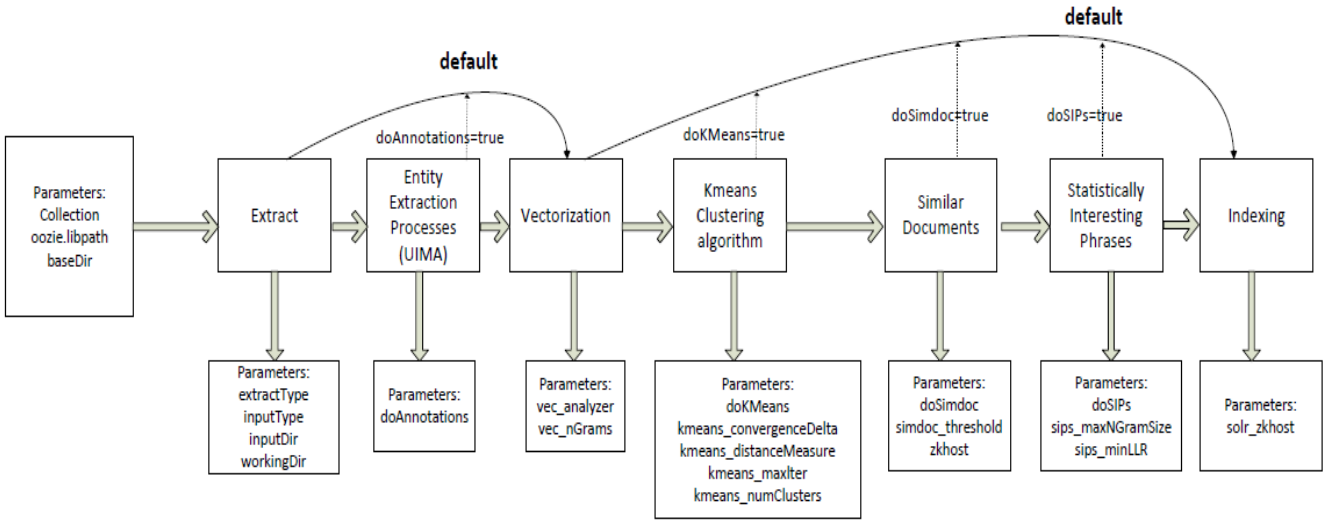
- For Example: A etl workflow with internal workflows executed is as in the example below

  In the example, we are enabling internal workflows KMeans [21], StatisticallyInterestingPhrases, and not enabling annotations [22], similar documents

```
curl -X POST -H 'Content-type: application/json' -d
‘{"collection":"test_collection_vt","inputDir":"hdfs://128.173.49.66:50001
/input/reuters/*/*.txt","inputType":"text/plain","solr_zkHost":"128.173.49.
66:2181/solr",
"doKMeans":"true","doSIPs":"true","doSimdoc":"false","zkhost":"128.17
3.49.66:2181","sips_minLLR":"50","doAnnotations":"false"}’
http://fetcher.dlib.vt.edu:8341/sda/v1/workflow/_etl
```

The picture below shows the sub-workflows present within the main workflow _etl and the parameters associated with each sub-workflow.

More information about parameters can be found at [20]



Sub-workflows (Discovery) [23]**:**

The Discovery functionality in LucidWorks uses Apache Mahout machine-learning algorithms to provide information about the documents in a collection. Please refer to [23] for more information on queries and definitions.

- **_log_analysis** [24]:
    - The _log_analysis workflow runs a series of Pig scripts across query and Click Scoring logs and produces the metrics that can be later used when running the Collection analysis API [24].

How Solr Indexing Works:

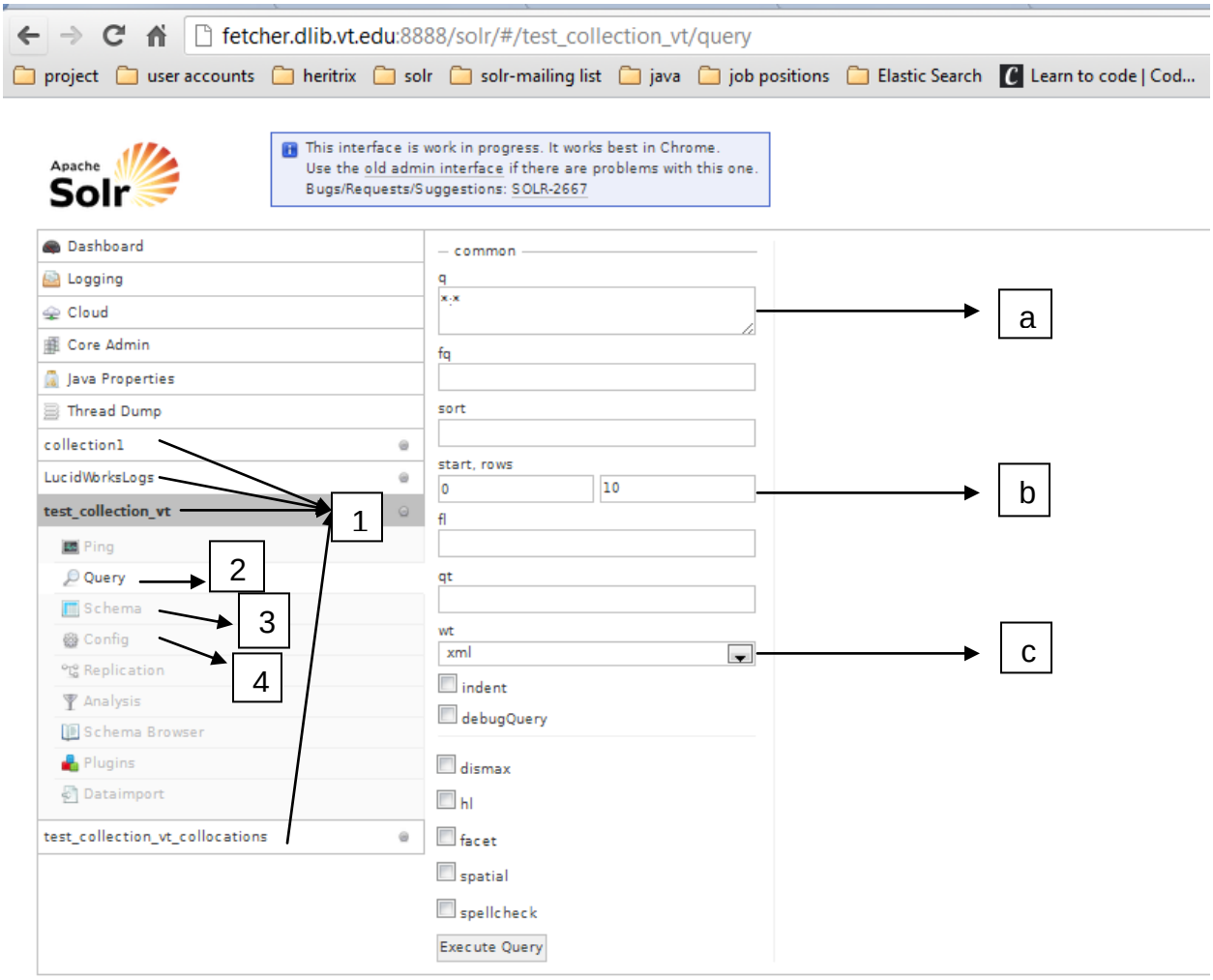
To learn in more detail about how Solr indexing works, please see [25] wiki.

Searching through Indexed documents:

Documents can be searched by using the curl commands as listed [26] in the documentation or through the admin interface [15].

Below is a screenshot of the query admin interface. Please see the labels and explanations to get familiar with the admin interface.

1. There are 4 collections in the picture below, each of which has their own schema, config, data, and indexes.
    a. For the Class CS5604 we will be using the collection name ‘test_collection_vt’ throughout the semester for sending documents for indexing or when retrieving documents based on queries.
2. This is the query syntax [27] for the collection ‘test_collection_vt’. The query syntax has many parameters which are explained in [27].
    a. We specify the query in the ‘q’ text box. A wildcard is specified in the screenshot above, which will retrieve all the documents present in the index.
    b. The rows parameter will specify how many results to be returned for the query.
    c. The ‘wt’ parameter shows in which format we want the results output to be presented.
3. This is the schema [25] file for the collection.
4. This is the configuration [28] file for Apache Solr.



10. Resources (required readings for students; additional suggested readings for instructor and students)

1. http://www.json.org/ and http://en.wikipedia.org/wiki/JSON
2. http://en.wikipedia.org/wiki/XML
3. http://wiki.apache.org/solr/
4. http://wiki.apache.org/hadoop/
5. http://mahout.apache.org
6. http://www.LucidWorks.com/products/LucidWorks-search
7. http://lucidworks.lucidimagination.com/display/bigdata/Overview
8. http://lucidworks.lucidimagination.com/display/bigdata/System+Architecture
9. http://hadoop.apache.org/
10. http://mahout.apache.org/
11. http://pig.apache.org/
12. http://www.zabbix.com/
13. http://www.opscode.com/chef/
14. http://curl.haxx.se/docs/manpage.html
15. http://fetcher.dlib.vt.edu:8888/solr/#/test_collection_vt

16. http://lucidworks.lucidimagination.com/display/bigdata/LucidWorks+Big+Data+API+Reference
17. http://lucidworks.lucidimagination.com/display/bigdata/Document+Indexing
18. http://lucidworks.lucidimagination.com/display/bigdata/Workflows
19. http://lucidworks.lucidimagination.com/display/bigdata/Jobs
20. http://lucidworks.lucidimagination.com/display/bigdata/Workflows#Workflows-api3
21. http://en.wikipedia.org/wiki/K-means_clustering
22. http://uima.apache.org/
23. http://lucidworks.lucidimagination.com/display/bigdata/Discovery#Discovery-Clustering
24. http://lucidworks.lucidimagination.com/display/bigdata/Collection+Analysis
25. http://wiki.apache.org/solr/SchemaXml
26. http://lucidworks.lucidimagination.com/display/bigdata/Document+Retrieval
27. http://wiki.apache.org/solr/CommonQueryParameters
28. http://wiki.apache.org/solr/SolrConfigXml

11. Exercises / Learning activities

12. Evaluation of learning objective achievement (graded exercises or assignments)

13. Glossary

14. Additional useful links:

15. Contributors (authors of module, reviewers of module)

Author: Kiran Chitturi

Reviewer: Dr. Edward A. Fox