Solr Team

CS5604: Cloudera Search in IDEAL

Nikhil Komawar, Ananya Choudhury, Rich Gruss

Tuesday May 5, 2015

Department of Computer Science
Virginia Tech, Blacksburg

Outline

1. Schema design
2. Indexing
3. Custom Search

System Overview

1. CPU
    a. Intel i5 Haswell Quad core 3.3 Ghz Xeon
2. RAM
    a. 660 GB in total
    b. 32 GB in each of the 19 Hadoop nodes
    c. 4 GB in the manager node
    d. 16 GB in the tweet DB nodes
    e. 16 GB in the HDFS backup node
3. Storage
    a. 60 TB across Hadoop, manager, and tweet DB nodes
    b. 11.3 TB for backup
4. Number of nodes
    a. 19 Hadoop nodes
    b. 1 Manager node
    c. 2 Tweet DB nodes
    d. 1 HDFS backup node

Solr Schema: Design

```
<!-- twitter fields -->
<field name="id" type="string" indexed
<field name="tweet_id" type="long" ind
<field name="collection" type="text_ge
<field name="text" type="text_general"
<field name="created_at" type="tdate"
<field name="source" type="string" ind
<field name="user_screen_name" type="s
<field name="user_id" type="string" in
<field name="lang" type="string" index
<field name="retweet_count" type="tint
<field name="favorite_count" type="tin
<field name="_version_" type="tlong" i
<field name="contributors_id" type="st
<field name="coordinates" type="string
<field name="urls" type="string" index
<field name="hashtags" type="string" i
<field name="user_mentions_id" type="s
<field name="in_reply_to_user_id" type
       multiValued="false"/>
<field name="in_reply_to_status_id" ty
       multiValued="false"/>
```

```
<fields>
    <!-- webpages fields -->
    <field name="id" type="string" indexed=
    <field name="collection" type="string"
    <field name="title" type="text_general"
    <field name="domain" type="string" inde
    <field name="url" type="string" indexed
    <field name="text" type="text_general"
```

```
<!-- class analysis fields -->
<field name="ner_people" type="string" indexe
<field name="ner_locations" type="string" ind
<field name="ner_dates" type="string" indexed
<field name="ner_organizations" type="string"
<field name="cluster_id" type="string" indexe
<field name="cluster_label" type="string" ind
<field name="classification_vector_json" type
       multiValued="false"/>
<field name="classification_labels" type="str
       multiValued="true"/>
<field name="social_vector_json" type="string
       multiValued="false"/>
<field name="social_importance" type="double"
<field name="lda_dict_json" type="string" ind
<field name="lda_topics" type="string" indexe
<field name="urls_multiple" type="string" ind
<field name="hashtags_multiple" type="string"
       multiValued="false"/>
<field name="ner_people_multiple" type="strin
       multiValued="false"/>
<field name="ner_locations_multiple" type="st
       multiValued="false"/>
<field name="ner_dates_multiple" type="string
       multiValued="false"/>
<field name="ner_organizations_multiple" type
       multiValued="false"/>
<field name="lda_topics_multiple" type="strin
       multiValued="false"/>
<field name="classification_labels_multiple"
       multiValued="false"/>
```

Solr Schema: Effect

Solr idx size depends on:

- Number of fields
- Stored vs. not stored
- Type of field
  - example: string vs. text
- Index issues:
  - Recommended to add H/W
  - Alternatively, design schema and tune Java/Solr configurations

Solr Schema: Future

- Fewer stored fields
- NRT indexer
  - Consider using facet.method=fcs, helps during first request
- Index size lowering
  - D x S + U
    - D is the document count
    - S is the the size of the data type
    - ints - 4bytes, 8 bytes for doubles, U cumulative size of the unique field values

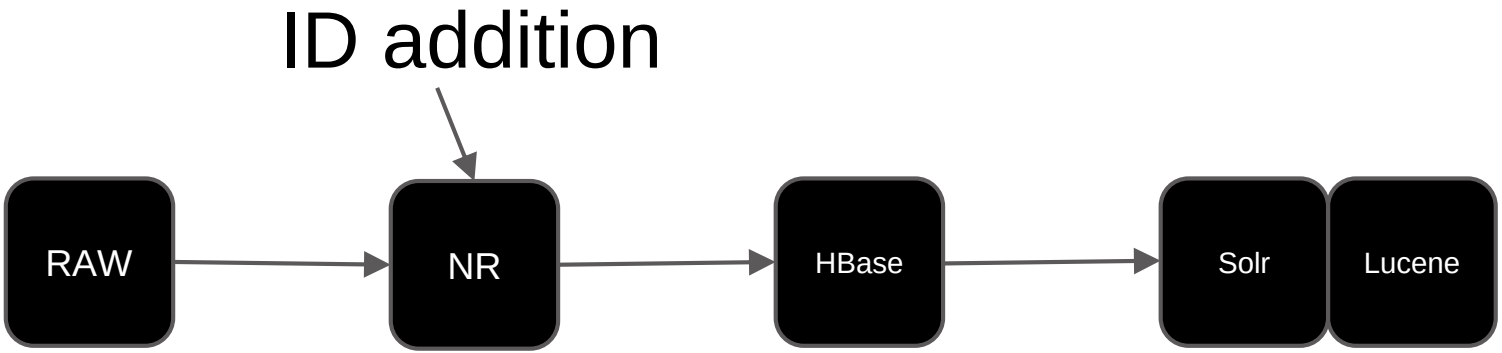
Document ID: Design

Syntax: Collection_name--Counter_value
Use: Noise, Reduction, HBase, Solr/Lucene

Document ID: Effect

- Affects Lucene indexing
- Does not affect Solr index size
- Fastest: Zero padded sequential
- Slowest: Random UUID generated using some languages (UUID v4)
- Our current design
  - Performance tradeoffs
  - Somewhere in the middle

ID generation in pipeline



Document ID: Future

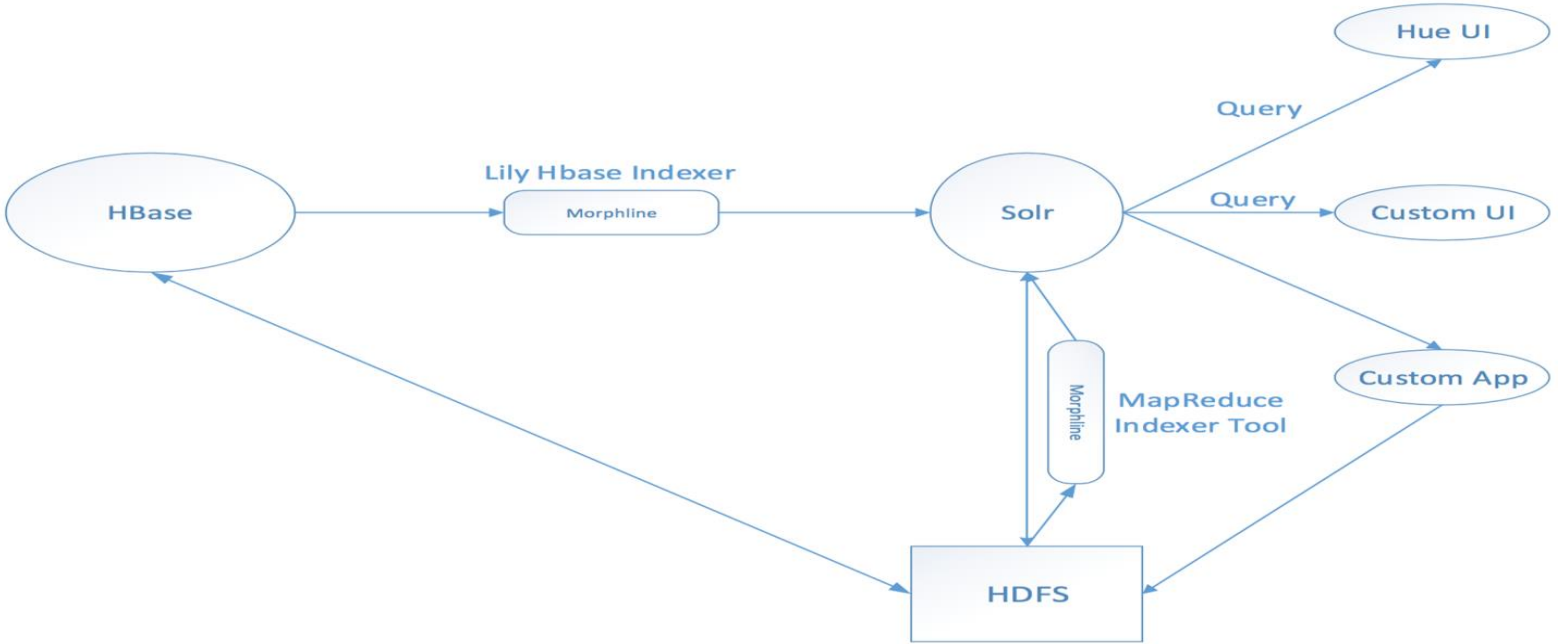
Preprocessing

- Current:
  - Concurrency
  - `test_and_set`
  - Batch processing
- Recommendation:
  - Binary encoded UUID
  - Parallel
  - CPU hit

Querying

- Current:
  - Faster fetching
  - Lower disk I/O
- Recommendation:
  - Sequentially assigned value or unhashed timestamp
  - Batch processing vs. asynchronous processing

Indexing


Hue UI
Query
HBase
Lily Hbase Indexer
Morphline
Solr
Query
Custom UI
Custom App
Morphline
MapReduce
Indexer Tool
HDFS

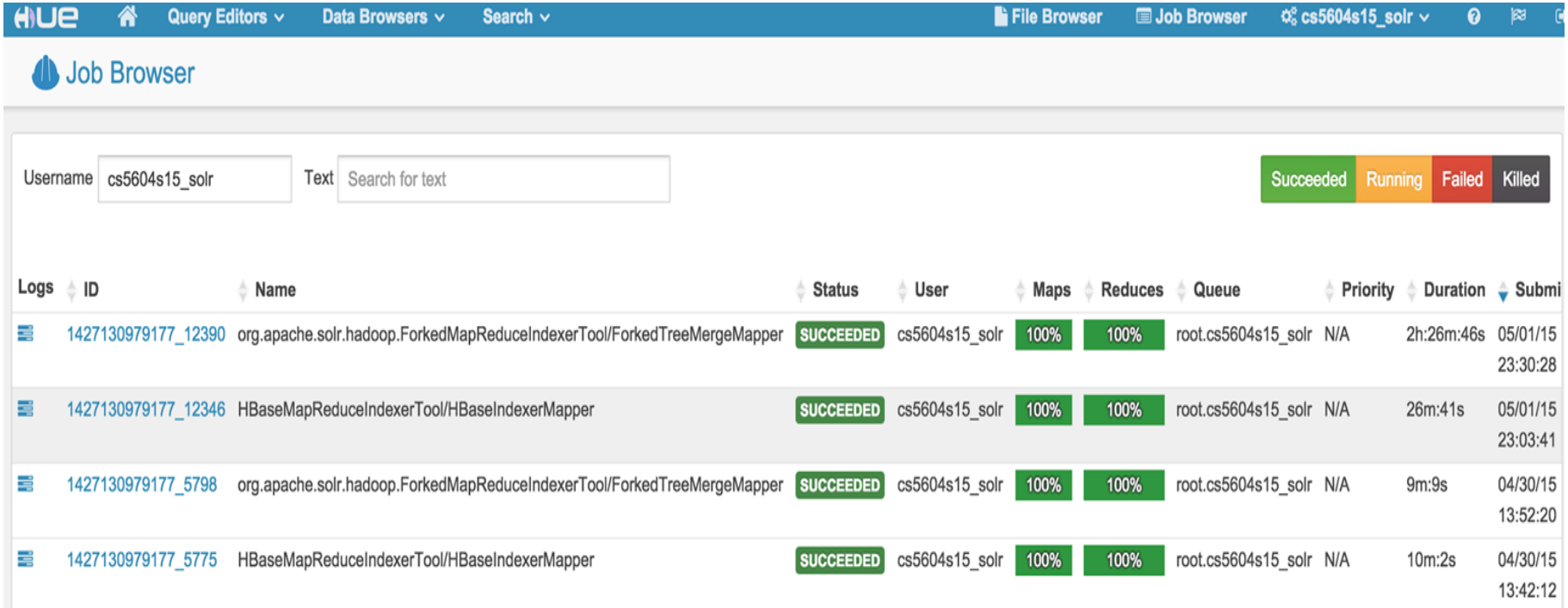
Hbase Indexer - Morphline

```
morphlines: [
        {
            commands: [
                {
                    extractHBaseCells {
                        mappings: [{
                                inputColumn: "original:text_clean"
                                outputField: "text"
                                type: string
                                source: value
                            }
                                                        {
                                inputColumn: "analysis:ner_people"
                                outputField: "ner_people_multiple"
                                type: string
                                source: value
                            }
                             ] }}
                  {
                    split {
                        inputField: "ner_people_multiple"
                        outputField: "ner_people"
                        separator: "|"
                    }
                }
```

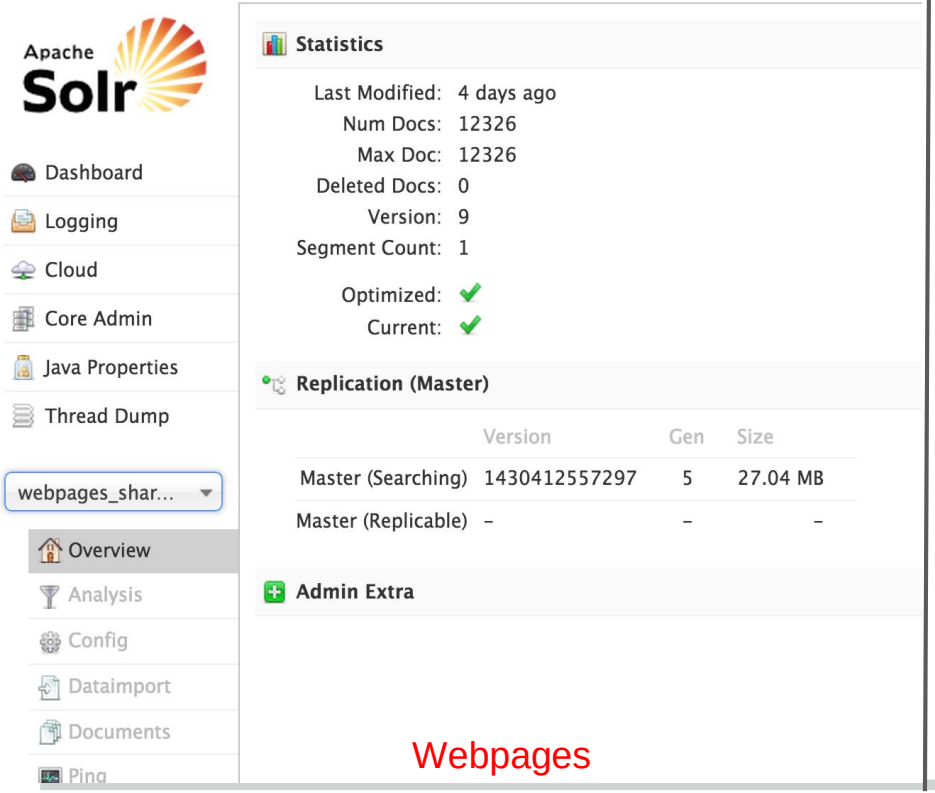
maps to HBase column family and column qualifier

defined in schema.xml

Indexing - Jobs



Results


Apache Solr
Dashboard
Logging
Cloud
Core Admin
Java Properties
Thread Dump
webpages_shar...
Overview
Analysis
Config
Dataimport
Documents
Ping
Statistics
Last Modified: 4 days ago
Num Docs: 12326
Max Doc: 12326
Deleted Docs: 0
Version: 9
Segment Count: 1
Optimized:
Current:
Replication (Master)
Version Gen Size
Master (Searching) 1430412557297 5 27.04 MB
Master (Replicable) - - -
Admin Extra
Webpages


Apache Solr
Dashboard
Logging
Cloud
Core Admin
Java Properties
Thread Dump
tweets_shard1_...
Overview
Analysis
Config
Dataimport
Documents
Ping
Statistics
Last Modified: about 16 hours ago
Num Docs: 3444305
Max Doc: 3444305
Deleted Docs: 0
Version: 3
Segment Count: 1
Optimized:
Current:
Replication (Master)
Version Gen Size
Master (Searching) 1430730434200 2 764.78 MB
Master (Replicable) - - -
Admin Extra
Tweets

Search

1: Adjust boost levels in Query Parser (solrconfig.xml)

```
<requestHandler name="/select" class="solr.SearchHandler">
    <lst name="defaults">
        <str name="defType">edismax</str>
        <str name="qf">
            text
            collection^3
            hashtags^3
            cluster_label^2.5
            lda_topics^2.0
            ner_people^2.0
            ner_locations^2.0
            ner_organizations^2.0
        </str>
        <str name="echoParams">explicit</str>
        <int name="rows">10</int>
        <str name="df">text</str>
        <str name="fl">*,score</str>
    </lst>
    <arr name="last-components">
        <str>idealSocialBoostComponent</str>
        <str>idealTopicSupplementComponent</str>
    </arr>
</requestHandler>
```

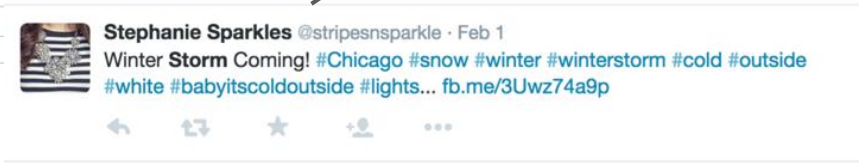
```
<searchComponent name="idealSocialBoostComponent" class="edu.vt.dlib.ideal.solr.IDEALSocialBoostComponent"/>
<searchComponent name="idealTopicSupplementComponent" class="edu.vt.dlib.ideal.solr.IDEALTopicSupplementCompone
```

These numbers are **conjectural**. A more disciplined approach:

Relevance = f(q,d)

For each of 100 queries, induce a logistic regression model. Find average contribution of each field to relevance.

Search




```
{
  "text": "Winter Storm Coming    storm     ",
  "user_screen_name": "stripesnsparkle",
  "urls": [
    "http://t.co/zuzA8x1bhe"
  ],
  "tweet_id": 383430624595510700O,
  "cluster_id": "winter_storm_S--c5",
  "collection": "winter_storm_S",
  "lang": "English",
  "id": "winter_storm_S--299751",
  "source": "twitter-search",
  "created_at": "2015-02-01T06:07:32Z",
  "cluster_label": "storm",
  "user_id": "2914255473",
  "hashtags": [
    "Chicago",
    "snow",
    "winter",
    "winterstorm",
    "cold",
    "outside",
    "white",
    "babyitscoldoutside"
  ],
  "coordinates": [
    "0.0,0.0"
  ],
  "_version_": 1500228833549222000
},
```

Search

2: Reorder documents on the basis of Social Importance Score.

solr.SearchHandler

```
for( SearchComponent c : components ) {
  c.prepare(rb);
}

            ...

for( SearchComponent c : components ) {
  c.process(rb);
}
```

edu.vt.dlib.ideal.solr.SocialBoostComponent

```
DocIterator iterator = docList.iterator();
while (iterator.hasNext()) {

    int docId = iterator.nextDoc();

    float score = iterator.score();

    Document d = rb.req.getSearcher().doc(docId);
    float socialBoost = 0;
    IndexableField socialImportanceField = d.getField("social_importance");
    if(socialImportanceField != null) {
        Number socialImportance = socialImportanceField.numericValue();
        if (socialImportance != null) {
            socialBoost = socialImportance.floatValue();

        }
        score = score + socialBoost;

    }
    scoreDocs[idx++] = new ScoreDoc(docId, score);
}
```

Search

3: Supplement result list, if necessary, by retrieving documents from collections that include topics related to the search. Get topic list straight from HBase.


```
@Override
public void process(ResponseBuilder rb) throws IOException {

    DocListAndSet results = rb.getResults();

    SortSpec sortSpec = rb.getSortSpec();
    int len = sortSpec.getCount();
    int offset = sortSpec.getOffset();

    DocList docList = rb.getResults().docList;
    if (docList.size() < MIN_DOCS) {
        String collectionName = getDominantCollection(rb);
        TopicModel topicModel = getCollectionTopicModel(collectionName);
        String topTopics = topicModel.getTopTopics();

        QueryParser queryParser = new QueryParser("name", new StandardAnalyzer());
        Query query = null;
        try {
            query = queryParser.parse("q=" + topTopics);

        } catch (ParseException e) {
            e.printStackTrace();
        }

        TopDocs additionalDocs = rb.req.getSearcher().search(query, 100);
        ScoreDoc[] scoreDocs = additionalDocs.scoreDocs;
        int totalHits = additionalDocs.totalHits + docList.matches();

        int[] docs = new int[scoreDocs.length];
        float[] scores = new float[scoreDocs.length];
```

```
private TopicModel getCollectionTopicModel(String collectionName) throws IOException {

    TopicModel topicModel =new TopicModel(collectionName);
    HTable table = new HTable(conf, "collection_metadata");
    Get get = new Get(Bytes.toBytes(collectionName));
    byte[] val = table.get(get).getValue(b("analysis"), b("lda"));
    String jsonTopicModel = new String(val, StandardCharsets.UTF_8);

    Map<String, Double> map = new Gson().fromJson(jsonTopicModel, new TypeToken<HashMap<String, Double>>() {}.getType(
    topicModel.setTopicProbabiliities(map);

    return topicModel;

}
```

Search Results

(first 1000 results)

{'query': 'election', 'num_results': 637498, 'precision': 0.998, 'time': 0.05295705795288086}
{'query': 'elect', 'num_results': 3247, 'precision': 0.978, 'time': 0.04558682441711426}
{'query': 'revolution', 'num_results': 13048, 'precision': 0.95, 'time': 0.04502081871032715}
{'query': 'uprising', 'num_results': 1769, 'precision': 0.851, 'time': 0.042988777716064453}
{'query': 'storm', 'num_results': 429329, 'precision': 0.999, 'time': 0.04975700378417969}
{'query': 'winter', 'num_results': 409987, 'precision': 0.999, 'time': 0.04920697212219238}
{'query': 'ebola', 'num_results': 306827, 'precision': 1.0, 'time': 0.0451481342315673 8}
{'query': 'disease', 'num_results': 6802, 'precision': 0.993, 'time': 0.041940927505493164}
{'query': 'bomb', 'num_results': 33924, 'precision': 0.857, 'time': 0.040463924407958984}
{'query': 'explosion', 'num_results': 1224, 'precision': 0.284, 'time': 0.04803609848022461}
{'query': 'crash', 'num_results': 274014, 'precision': 0.995, 'time': 0.04688715934753418}
{'query': 'plane crash', 'num_results': 193046, 'precision': 1.0, 'time': 0.056591033935546875}
{'query': 'shooting', 'num_results': 5366, 'precision': 0.744, 'time': 0.04262495040893555}
{'query': 'paris shooting', 'num_results': 446, 'precision': 0.446, 'time': 0.20793604850769043}
{'query': 'terrorist attack', 'num_results': 1143, 'precision': 0.768, 'time': 0.0426750183105468 75}

Future Work

1. Relevance feedback to derive logistic regression, evaluate with F1 score.
2. More Solr nodes -> more index shards.
3. Boost by length
4. Innovative inputs (social graph)
5. More sophisticated traversal of hierarchical classification

Acknowledgement

We are especially thankful to

- The NSF grant IIS - 1319578, III: Small: Integrated Digital Event Archiving and Library (IDEAL) for the funding that supported the infrastructure and the data used in the project.

- Dr. Edward A. Fox for being the guide on the side for us and for coordinating efforts of all the teams to help us make it a successful class project.

- The GTA (Sunshin), the GRA (Mohamed) and other students of the class who supported us with ideas as well as efforts during the semester.

- The authors and the contributors of open source projects, blogs and wiki pages from where we borrowed some ideas and solutions.

Thank you!

Q & A