Distributed Computing Project Proposal

Project Title:

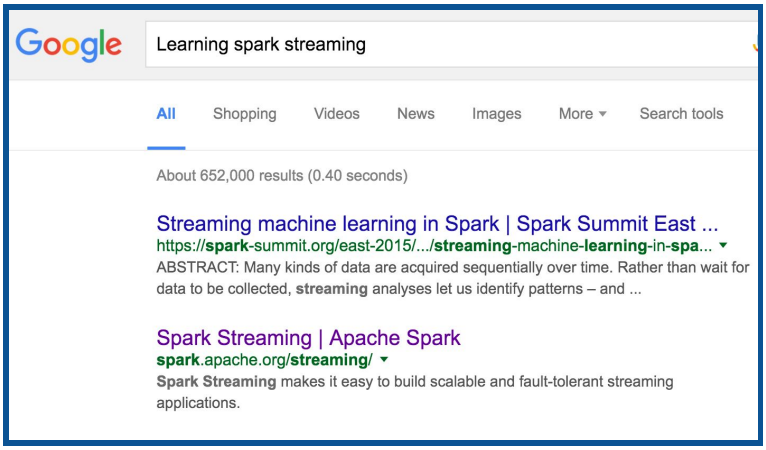
“Search Optimization in a Elastic Database”

Candidates:

Swetha Reddy (20357646)
Tate Campbell (20352330)
Abhishek Singh (20363537)

1. Project Description

Whenever we have doubts about a topic, the first thing we do is go to our favorite search engine (Google, Yahoo, etc.) to get a list of websites which potentially contain answers for our search. What drives this search engine?



Welcome to the world of *Web Indexing*, which ensures that we get the best possible matches for our search input from trillions of links. Our project involves creating a search algorithm that explores our elastic multi-tenant database and reports the most relevant search results for our search input. The web crawling and indexing would be done on a multi-node setup. We will be using three different, competing technologies to arrive at the best solution, which shall then become our final product.

2. Project Methodology

We will scrape data from sites such as www.nytimes.com in the pySpark environment and store the results in our elastic database. Once the data has been stored we will perform indexing on this database and execute local search. These searches will be gauged on the speed of search. We will compare the results from Spark, Hive and Python environments. Based on the tests performed across the 3 environments, we will be able to rank them. The performance criteria: time taken, relevance of the results, and number of unique results.

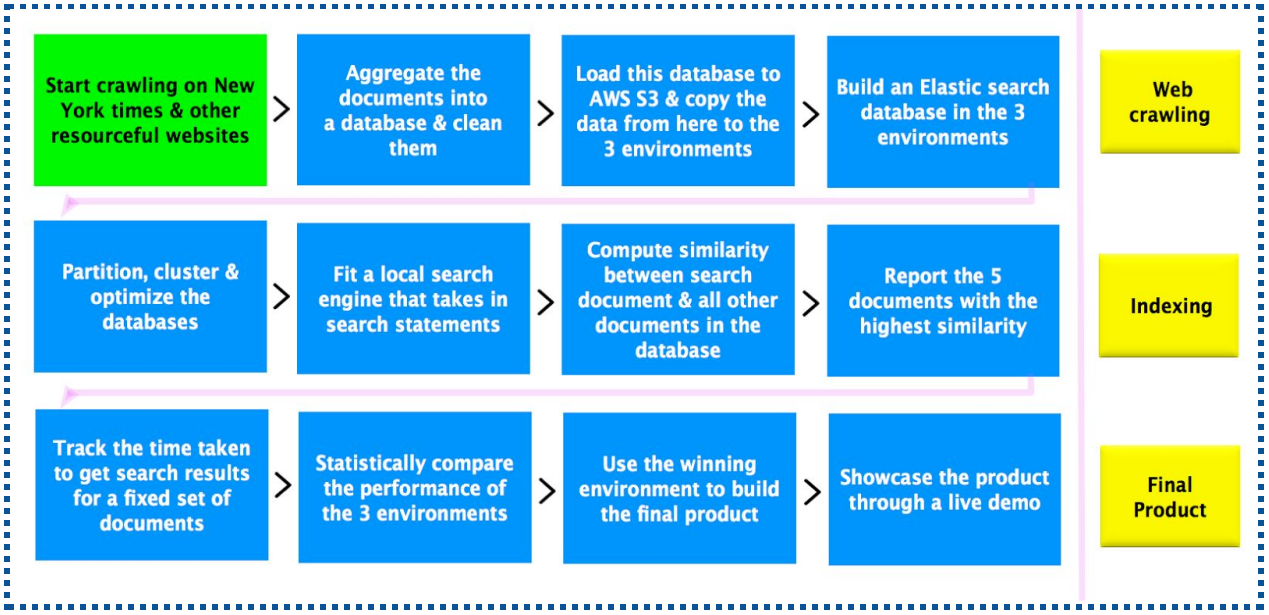
The four major components of this project are:

1. Build a optimized database with a large number of clean documents.
2. Compute similarity scores between search input and documents in database.
3. Benchmark the results in a distributed environments over multiple nodes.
4. Present the results visually (D3, plotly) and search engine via live demo.

3. Evaluation Criteria

We will be using a set of documents to benchmark the performance of the 3 different environments. We will enter these documents in our search tab (1 at a time) and wait for the search engine to show up 5 search results with the highest similarity score. The **time taken** by the 3 different search engines (Spark, Python & Hive Based) for computing search results would be our **evaluation criteria**. We will track the time taken by these 3 engines for a list of documents. The Engine that takes the least time based on statistical analysis (T-tests, Box Plots) would be declared our winning search engine from the 3 cases. The flowchart below depicts the steps.

4. Flow chart



5. Expected Timeline

We are currently researching technologies for the indexing problem but we will have our initial results by **30th April, 2016**. Our approach would be to continually improve & optimize our local search engine at both local & distributed level. Final product will be done by the day of the demo.

6. Technology

Ours would be a Spark based project at its core, as traditionally Spark has outperformed other technologies in terms of speed & performance. Our evaluation criteria here is speed of search for getting the Top 5 results. Performance of search on Spark will be benchmarked against the performances of two other popular tools:

1. **Hive**: Writing optimised HiveQL queries to get search results from the indexed database.
2. **Python**: Running Python in multithreaded mode to get search results from the database.

So it would be a competition between the Spark vs Hive and Python, for this use case of local searches. To present the results we will be using visualization packages such as Plotly & D3. In case the time allotted to each presentation is between 10 - 15 mins, we will do a live demo hosting our results through the web (Flask), backed on a EC2 instance of AWS.

7. Sources

Since Web scraping & indexing at a large scale requires optimum use of tools. We will be referring a variety of links, papers & videos for our learning & testing purposes. Some of the links we find useful are:

1. http://www1.cs.columbia.edu/nlp/theses/dave_evans.pdf
2. http://nlp.stanford.edu/IR-book/pdf/20crawl.pdf
3. http://www.micsymposium.org/mics_2005/papers/paper89.pdf
4. http://docs.python-guide.org/en/latest/scenarios/scrape/
5. http://www-rohan.sdsu.edu/~gawron/python_for_ss/course_core/book_draft/web/web_intro.html
6. http://dlab.berkeley.edu/blog/scraping-new-york-times-articles-python-tutorial

Apart from our own findings, we will refer to suggested readings from **Professor Maria Daltayanni**.

8. Individual Responsibilities

This being a group project we have split the task nearly equally in terms of complexity & quantum of work. As a team we shall keep helping each other for all possible concerns, however each one of us would be taking ownership of specific parts of the project. The details are as such

1. API based Web Crawling - (Tate, Swetha, Abhishek)
2. Elastic database creation & management - Swetha (Primary), Abhishek (Secondary)
3. Python Implementation (Local & Distributed) - Abhishek
4. PySpark Implementation (Local & Distributed) - Tate
5. Hive Implementation (Local & Distributed) - Swetha
6. Documentation & Plots (Reports & Presentations) - Abhishek (Primary), Swetha (Secondary)
7. Interactive web based demo - Tate (Primary), Swetha (Secondary)

Although we are confident of our current estimates of workload and have distributed the same amongst us but it is likely that some extra work may emerge. In that circumstance, we shall split the work on an ad-hoc basis.

9. Final Deliverables

We will follow the format as laid out in the instruction for the deliverables of this project.

1. Progress Report & Final Report through Canvas
2. Codes from python, Hive, Spark, D3, bash command & others through Github
3. Presentation (pdf) slides through Canvas
4. Visualization & demo links via hyperlinks

On the day of the presentation we plan on sharing the below deliverables for the audience:

1. Presentation deck that shall walk through the concept, working & application.
2. D3 & Plotly web charts to visually present the performance metrics using localhost.
3. A live demo of our algorithm working through the local host (conditioned on time sufficiency).