SMC 2018 Data Challenge #4

Ketan Maheshwari (km0@ornl.gov)

July 23, 2018

Introduction

This report presents my response to the SMC 2018 Data Challenge 4–Acedamic Data Mining. The challenge is addressed by combining the simplicity of the classic Linux tools with the power of a modern scalable parallel scripting platform.
The report describes the tools used, their rationale, solutions to the problems and a glance at the results. Source code for solutions and results may be found at https://github.com/ketancmaheshwari/SMC18. The source code for the core solutions are in files `src/prob*.awk`. The source code for the Swift parallelization of the core solutions are in files `src/runprob*.swift`. All the results are in `results/` directory.

Tools used

Software

Classic Linux tools such as `awk`, `sort`, `grep`, `tr`, `sed` and `bash` are used. For the bulk of processing, `awk` is used. Syntax of awk programs is known to be terse and hard to read by some accounts. I have taken special care to make the programs as readable as possible. ANL's Swift (`swift-lang.org/Swift-T`) parallel scripting tool is used to run the awk programs in parallel over the dataset to radically improve performance. Swift uses MPI based communication and load-balancing model to parallelize over shared as well as distributed memory architectures.

Hardware

A large-memory (24 T) SGI system with 512-core Intel Xeon (2.5GHz) was used for all the processing. All the IO is memory bound (/dev/shm).

Rationale

1. `Awk` is concise, fast, and expressive – especially for text processing applications.

2. SQL based tools and databases are hard to port, parallelize, and maintain.
3. Traditional Linux tools are portable and available on almost all the standard Linux distributions.
4. Alternative tools such as modern Python libraries have speed limitations, portability concerns and considerable learning curve. Some of them are still evolving.
5. Swift was used over GNU parallel for its portability to both shared memory systems and distributed memory clusters.

Data

The original data was in two sets (*aminer* and *mag*) of json files (total 322 files) each file containing a million records. The data also consisted of a file with a list of duplicate records. An `awk` script (`src/filterdup.awk`) is used to exclude the duplicate records from the aminer dataset. As a result, there are over 256 million (256,382,605 to be exact) unique records to process. The data occupies 329G of disk space. Several fields in the data are `null`. Those records were avoided where relevant. Additionally, records related to non-English publications were avoided where needed.
The json files were converted to tabular files with 19 original columns and one additional column called *"num_ authors"* showing the number of authors for a given publication record.

Auxiliary data

In addition to the existing data, I use four lists: 1) A list of cities and their lat-long coordinates (3,517) of world with population 100,000 or above; 2) A list of countries (190); 3) A list of all the universities and research institutes (8,984); 4) A list of stop-words to avoid in some of the results (161 words).

Solutions

Preprocessing

The `jq` (stedolan.github.io/jq) tool was used to transform json data to tabular format (`src/json2tabular.sh`). Further curation of table data was done by removing extraneous space, square brackets, escape characters and quotes using `sed`.

Parallelization and Speedup

Each of the solution is run in parallel over the 334 data files. This has resulted in radical speedup of each of the problems. As mentioned in the description of the solutions in the sections below, none of the solution has taken more than an hour of runtime.

Problem1

Identify the individual or group of individuals who appear to be the expert in a particular field or sub-field.
This problem is solved in two ways. First part identifies all the entries with citations higher than 500 for a given search topic (`results/meditation_highly_cited.txt`). The second part finds the names of authors whose names are repeating for queried topic with atleast a certain number of citations in each entry. This will give a reasonable idea of who are the expert figures in a given research area. One such result in `results/cancer_research_topauths.txt` shows authors in cancer research with more than one publication with at least a 1,000 citations. Along side is a result of a query the all-time list of most cited papers with a threshold of 20,000 in `results/top_papers.txt`. The parallel implementation of the task finishes in 25 seconds.

Problem2

Identify topics that have been researched across all publications.
This problem is solved by identifying most frequently used words in the collection. Several common words (aka stop-words) are filtered from title, abstract and keywords and top 1,000 words across the whole collection was found. At over 23 million, the word "patients" has most frequent occurrence. The full list of top 1,000 words may be found in `/results/top_1K_words_kw_abs_title.txt`. The parallel implementation of the task finishes in 9 minutes.

Problem3

visualize the geographic distribution of the topics in the publications.
This problem is solved by identifying the author's affiliations for the records that has user supplied keyword in them. The affiliation is searched against three databases–cities, universities and countries to find out the geographic locations for that research. The results are aggregated to present a list of centers for which a given keyword appears most frequently. For cities, the results are plotted on world map. One such example is shown in `/results/bird_research_cities.png` for the keyword "birds". The directory also contains other similar example results such as epilepsy, opioid, meditation research by universities and by countries. The parallel implementation of the task finishes in 25 seconds.

Problem4

Identify how topics have shifted over time.
This problem may be solved in three distinct ways.
The first part processes the database to find out yearwise occurence of user defined topics. It generates a list of years and the number of times **both** topics has occured in a single publication in that year. For example, the result in `results/obesity_sugar.pdf` (plot courtsey S. Somnath) shows how the terms "obesity" and "sugar" have trended together in publications over the years.

The second part finds the papers that has highest impact in each year and extracts the keywords in those papers. The impact is computed by the paper that was cited the most in that year. The result for this task are in `results/yearwise_trending_keywords.txt` in the form of year, keywords, citations triplet.
The third part find the top 10 most frequently occurring terms each year to find how the topics get in and out of trend over the years. An animation showing a bubble plot of words changing between the year 1800 and 2017 may be found at `results/freqwordsoveryrs.mkv`. A file list of all the words may be found in the directory `results/trending_words_by_year`.
Parallelizing the third part was most challenging because it involved a two-level parallel nested foreach loop. The outer loop iterates over the years and the inner loop iterates over the input files. The parallel implementation of the task finishes in 48 minutes.

Problem5

Given a research proposal, determine whether the proposed work has been accomplished previously.
This problem has a simple solution: Find the keywords on a new proposal. If those keywords appear in an existing publication record, it is a suspect. A broad list of suspects may be found with logical OR between keywords which could be narrowed with logical AND. The keywords may be arbitrarily combined in logical ORs and ANDs. The results file `/results/suspects.txt` shows over 1,400 suspects for an AND combination of keywords: battery, electronics, lithium, and energy from English language papers. The parallel implementation of the task finishes in 26 seconds.

Postprocessing

Some results were postprocessed for sorting and merging purposes using Linux `sort, uniq` commands which took negligible time. Some of the results obtained were postprocessed for visulization using the `D3` (d3js.org) graphics framework and ffmpeg libraries.

Summary

In this work I show how the classical Linux tools may be leveraged to solve modern problems. I show that hundreds of millions of records may be processed in under a minute at scale using Swift. I show the results that offer insight into the data without employing any formal sophisticated algorithms.

Acknowledgements.

I acknowledge Brian Zachary, Suhas Somnath, Drew Schmidt for their helpful inputs. I acknowledge CADES for the resources for this project.