Experiences Clustering High-Dimensional Data using pbdR

Sadika Amreen and Audris Mockus
University of Tennessee, Knoxville

Motivation

Challenges of Software Development for HPC (Basili et al)

- High complexity of middleware
- Complex architecture: nodes, network, storage
- Need for parallel processing, including parallel IO
- Development practices differ from “traditional” SE

HPC is not designed for big data/analytics

- No databases
- No analysis tools

Objective

Leverage pbdR - analytics middleware to simplify HPC

- Use R without having to master many layers of HPC infrastructure such as OpenMPI or ScalaPACK
- Evaluate the extent to which it improves productivity
  - Time to write and debug the program
  - Time to run it

By solving a real problem involving big data

Big Data Context: Open Source Repository Mining

All version control systems in open source

- 50 million projects
- 20 million unique developer IDs
- 1 Billion commits

Select IDs with >10 commits & >40 characters

~ 2.3 Million IDs

Use commit message text to identify developers

Problem Context: Open Source Repository Mining

Developer id in a commit: name and email -- spelled in multiple ways

- Coalescing of identities where multiple credentials belong to a single actor
- Determining correct credentials for a commit when multiple actors share a single credential i.e. organizational email

Workflow

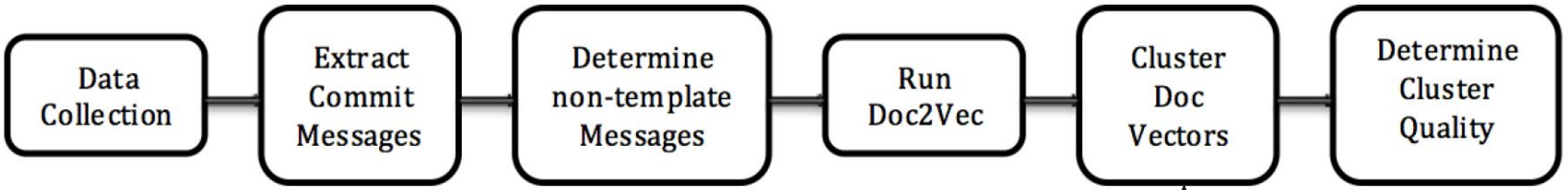
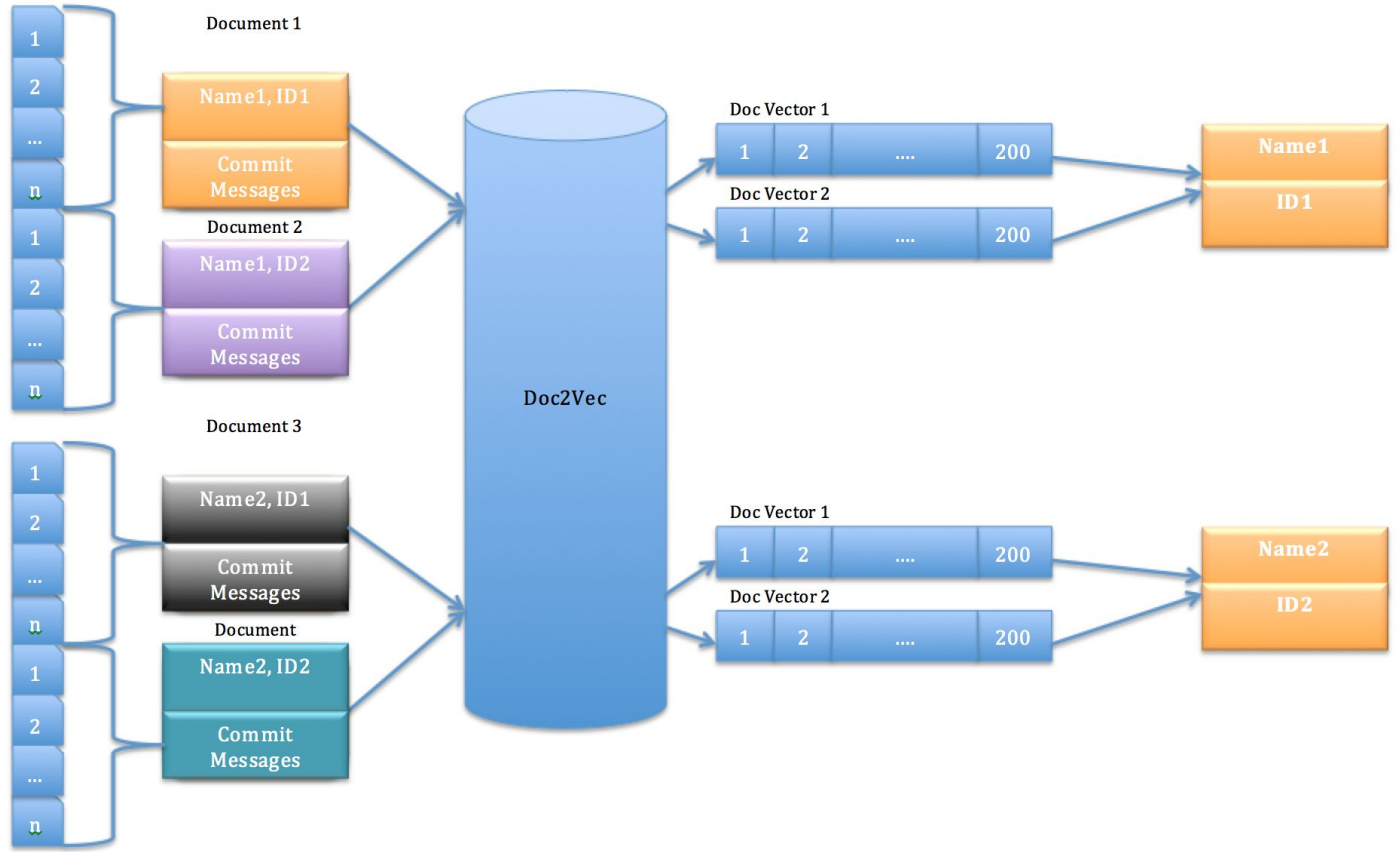


Figure 1: Concept of Workflow

Leverage pbdR

Application of Doc2Vec to Identify Individuals in Online Communities



Data

Doc2Vec model with original credentials used as a document identifier (commit message as document)

Cluster the resulting

- 200-dimensional vectors
- 2.3M identifiers

The distance matrix is very large: a good target for HPC

Results: Parallel Reading of Data

Data stores in a compressed csv (8GB)

Split across a thousand “chunks” (compressed via gzip)

Read across ranks using Reader()

Store in ddmatrix : based on ScalaPACK

Results: Reading Data in Parallel

Data distribution considers

(1) Maximizing processor utilization and minimizing communication overhead

(2) The distance matrix (2.3M X 2.3M) created for clustering jobs, will consume ~10K times more memory than the input data (2.3M X 200)

Results: Prototyping Clustering

Synthetic small sample (108 observations)

- 2 Dimensional
- Small std dev within clusters
- Large std dev between clusters

This enables us to create “clean” cluster samples that any clustering algorithm could easily recover.

Results: Pkmeans

Run Parallel K-Means (pbdR ‘pmclust’ package)

Simulated 12 clusters with 9 data points in each cluster (K = 12,p = 2,n = 108)

PK-Means reconstructs clusters when centroids are initialized with actual means

Getting Accurate Results

Table 1: pkmeans: K=12, p=2, n=108, Procs = 9, Node = 1

Iteration	Min Error Iteration	Time(min)
50,000	822	43.55
20,000	1568	17.22
10,000	40	8.46
5,000	227	4.24
1,000	50	0.86

Cluster Quality: Sum of squared distance from each data point to its corresponding centroid

Pkmeans

Table 2: pkmeans: K=10, p=2, n=1000, Procs = 40, Node = 3

Iteration	Min Error Iteration	Time(min)
15,000	35	37.30
10,000	48	24.37
5,000	24	12.32

Did not converge at n = 2000 << 2M and at dim = 2 << 200 and K = 20 << 1M

Table 3: pkmeans: K=20, p=2, n=2000, Procs = 200, Node = 13

Run	Min Error	Min Error Iter	Time(min)
1	159.05	1235	24.54
2	155.62	1119	24.31
3	157.87	817	24.41
4	156.01	1729	24.17
5	155.75	1153	24.63

Challenges

Lack of adequate documentation in scientific software

Software could not be used as “black-box”

Bugs as a result of

- Few users and developers to detect and fix bugs
- Massive number of configurations options
    - due to hardware and software

Current Work

Before: Random partitioning of data

Now: Partition based on distance between vectors, i.e. authors with similar vectors will be placed in same ranks

Run clustering algorithm within each partition

Challenge: Extreme disparity between partition sizes

Summary

Prototyping is crucial, but cannot prototype on sufficiently large data

Do only what is absolutely necessary on HPC env.

- Once we partition the data we plan to move away from the distributed environment

Debugging is time consuming, because we often cannot output results due to size

How to address the issues?

Need more focus on documentation

- More vignettes illustrating how to use
- How to use effectively (%*% VS crossproduct())

Need better testing tools

- To account for millions of configuration options

Need more development resources

- pbdR has only one PhD student who works on it