CPSC 490: TorMR

Will Childs-Klein

Introduction

Parallel data processing methods have allowed people of all professions to analyze data at an unprecedented scale, yielding key insights in myriad industries and academic disciplines. Contemporaneously (and not entirely independently), data privacy has become an important and immediate issue in our society. Organizations in both the public and private sectors constantly collect or access data on their users or constituents, often without explicit consent by the individual. As computing continues to shift into "the cloud" from users' local systems, user data is relinquished to, and becomes centralized in, the small number of prevalent cloud service providers. In distributed parallel data processing, where clusters of multiple physical or virtual machines execute jobs in parallel, this introduces a potential for privacy and confidentiality compromises of users' data, as well as results of their computations. Even an entirely privacy-respecting benign cluster provider, the contemporary model of centralized cluster control provides an attack focus for malicious hackers. Though there are many different architectures for parallel and distributed processing, we will focus exclusively on the MapReduce paradigm in this project.

Problem Statement

So, how can we protect the privacy of users' computations while still affording the user all (or at least most) of the major features of modern distributed data processing architectures? We need a system that:

1. is decentralized (peer-to-peer) in architecture with granular data access control, to prevent a malicious peer from being able to see too great a portion of the user's data.[1]
2. preserves the users' anonymity[2], keeping it hidden from other peers in the cluster.

[1] this may be susceptible to sybil attacks
[2] or at least pseudonymity

3. maintains compatibility with existing MapReduce implementations without requiring the user to re-write large portions of their code base.

TorMR

We propose a system called TorMR which implements the Map Reduce paradigm, but manages clusters over the Tor protocol. Users who wish to accrue a cluster register a request with TorMR's bulletin-board, which is protected behind a Tor hidden service (THS). Other users who wish to volunteer their nodes' compute power register them as available workers with the bulletin board THS, and when enough nodes are available, a cluster is formed. Worker nodes on the cluster are unaware of each others' locations (IP addresses), as is the node requesting the compute job, as all communication is through Tor's hidden service protocol. We will elaborate on this protocol soon, but we will first enumerate the system's components.

Privacy Model

Components

Bulletin-Board THS

Job THS

Discovery & Cluster Management

Distributed Data Layer/File System

Protocol

Tools, Dependencies, & Technologies

Tor

Pachyderm MapReduce

git

Docker

CoreOS & etcd

Development & Deployment Methodologies

Source Testing

Vagrant

VirtualBox

Compute Cluster

Deliverables

1. code for simulations/proof of concept

2. code documentation
3. formal document containing description of project, implementation details & design decisions, and analysis of experimental results.

Future Work

- dynamic cluster management
- fully anonymous (self-hosted) data layer
- payment system (probably via BTC)
- pipeline implementation
  - this should be implemented as a dumb pipe for compute jobs
  - replace Bulletin-Board THS with job queue
  - instead of matching available workers with individual jobs, keep them as pool (i.e. all available workers are part of same super-cluster)
  - users register their jobs as a git repository
  - when job is up in queue, job is `git push`'d to the cluster where it is executed. data is read/results are written from/to specified storage buckets
  - as part of job definition, requester must specify storage bucket address.

Timeline & Milestones

Deadline	Goal
2/22/15	get MR sample MR job running on virtual cluster
3/01/15	
3/08/15	
3/15/15	
3/22/15	
3/29/15	
4/05/15	
4/12/15	
4/29/15	All done!