Orchestration Proposal Presentation

Robert Hensing Robin Kuipers Jelle Postma

January 5, 2016

Orchestration?

Orchestration?

Wikipedia: Orchestration describes the automated arrangement, coordination, and management of complex computer systems, middleware and services.

Taxonomy

Wikipedia level research :)

Taxonomy

- Arrangement
- Coordination
- Management

Taxonomy

- Arrangement
  - Allocating machines
  - Deploying applications
- Coordination
- Management

Taxonomy

- Arrangement
- Coordination
  - Peer discovery in distributed app?
- Management

Taxonomy

- Arrangement
- Coordination
- Management
  - Monitoring
  - Configuration

Existing products ...

... from the hadoop community

ZooKeeper A coordination service for distributed applications

Ambari Web UI for provisioning, managing and monitoring other hadoop products

Chukwa A monitoring / data collection system piggybacking on the storage and compute systems in hadoop

ZooKeeper

- Very simple interface
- Maintenance, coordination, management, consensus...
- Synchronisation for distributed systems
- Aims to implement functionality that is hard to implement but easy to use

Ambari

- Web-based interface
- Installation, management and monitoring of Hadoop clusters
- Easily integratable with its RESTful APIs
- Very wide range of features

Chukwa

- Data collection
- Mostly used for monitoring and analyzing systems
- Builds on Storage and Computation systems
- Very scalable and robust

Choosing between the three

- Chukwa
- Ambari
- Zookeeper

Choosing between the three

- Chukwa
  - Too heavily based on other systems
  - Therefore an interesting project, but not the scope that we are looking for
- Ambari
- Zookeeper

Choosing between the three

- Chukwa
- Ambari
  - Too large; would have to pick subset of functionality
  - Choice would again be very dependant on work of other groups
- Zookeeper

Choosing between the three

- Chukwa
- Ambari
- Zookeeper
  - Focusses on concurrent programming
  - Has a core which we can start implementing, and continue from there
  - Great for a small project like this

Choice of project

TimeKeeper

- A distributed data store for cloud management
- Focus on consistency, availability, scalability
- Not unlike ZooKeeper

Project

- Deviation from sequential use: for sequential use, we have roughly @Data.Map@ with change notifications. Requirement: availibility, scalability.
- We will implement the core logic of an in memory consistent/available data store. Stages, at least three, for example:
  - Serve from one process
  - Serve through multiple processes: master/worker
  - Serve more efficiently with multiple processes: implement sharding (how to deal with consistency?)

Experimentation

- Measure req/s for write-intensive vs. read-intensive workloads
- Measure req/s for single point of synchronisation vs. sharded
- Measure effect of simulated intercontinental latency on throughput

Questions?