DistCI – Continuous Integration at Scale

Highly available, fault tolerant, scalable and extensible continuous integration system

https://github.com/F-Secure/distci

APLv2

Speaker Intro

Heikki Nousiainen

Lead Software Architect, Cloud Technology Strategy

Heikki.Nousiainen@F-Secure.com

Context

- 600 R&D personnel
- 30+ Jenkins instances
- 1800+ Executors
- 4500+ Jobs
- 15000 – 18000 VM instances spun up every working day

- Challenges:
  - Stability
    - Number of concurrent builds
    - Plugin stability
  - Availability – downtime equals to loss of work
  - Job state/progress lost when Jenkins goes into maintenance
    - Long running tasks

Goals

- High availability
  - Always available for new progress
- Fault tolerance
  - Minimal impact of any node failure; lose at most progress from single node
- Fault recovery
  - Facilitate automatic restarts of interrupted progress
- Horizontal scalability
  - 1000s of configured jobs, 100s of concurrent builds

Key Concepts

- Reversal of control flow
  - Eliminate active master
  - Pull instead of push
  - Maintain a repository or a global state
    - All progress and state transitions are made by calling applications, clients or workers
    - State is not actively tracked, but updated by workers
  - Work is scheduled via posted tickets

Key Concepts

- Simplified workers
  - Independent
  - Do one thing and do it well
  - Easy introduction of new workers and functionality
  - Chain multiple workers for rich functionality
  - Workers need no pre-registration, and are easily scaled up/down based on load

- Examples:
  - Git checkout
  - execute script
  - publish artifacts

Implementation

- Ceph, distributed object storage and filesystem
  - Provides availability and fault tolerance for persistent data
- ZooKeeper, distributed coordination service
  - Provides synchronization for ticket assignment, as well as distributed locking where needed
  - Available and fault tolerant
  - Ephemeral locks; if caller goes away, lock is freed

Implementation

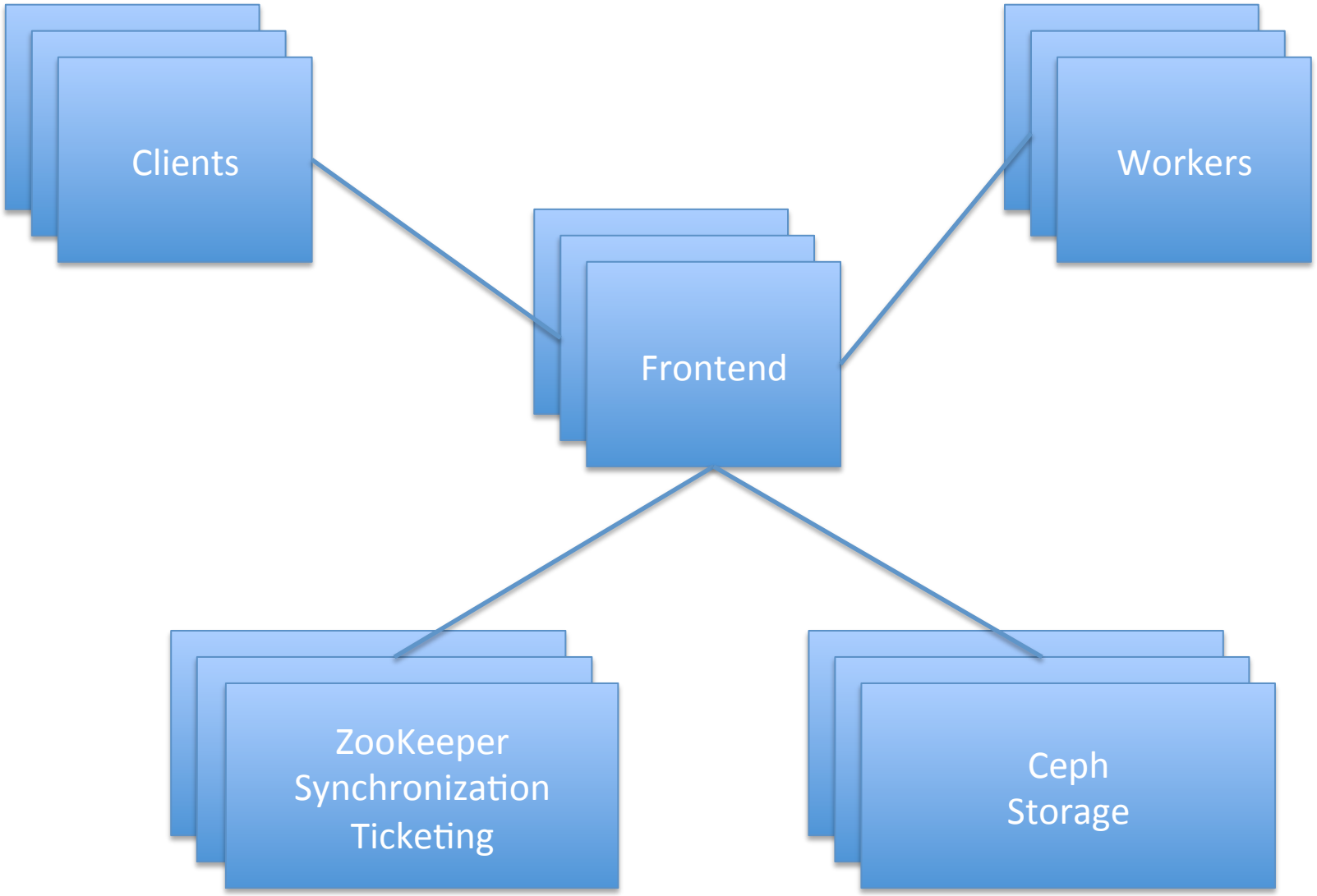
- REST/JSON Frontend to tie these together
  - Provides
    - Views to tickets and repository
    - Hooks for manipulating state
  - Does not hold state
  - Does not do active processing
  - Scaling is straightforward addition of nodes

Implementation

- Extremely simple workers
- Retry requests until successful

- Either long lived or spawned afresh after each task completion
  - VM instances
  - LXC instances

Implementation



Results

- Test setup in AWS
  - 3 DistCI frontends
  - 3 ZooKeeper nodes
  - 3 Ceph monitors
  - 6 Ceph OSD storage nodes
  - 10 worker nodes
- Distributed over 3 availability zones

Results

- Availability
  - Able to sustain loss of one full Availability Zone
- Fault Tolerance
  - In the event of AZ failure, only progress on the worker nodes in the AZ is lost
- Fault Recovery
  - Task restarts not yet implemented, but certainly seem possible
- Scalability
  - Able to host 10000 projects and run 500 concurrent builds

Future

- Concept looks promising, but DistCI is not quite ready for general use
- Focus on documentation, tooling

- In the coming 3 months, launch public free-to-use service for Open Source projects

- Brave early adopters & contributors most welcome!