Pigeon: an Effective Distributed, Hierarchical Datacenter Job Scheduler

Zhijun Wang, Huiyang Li, Zhongwei Li, Xiaocui Sun, Jia Rao, Hao Che and Hong Jiang

University of Texas at Arlington
Datacenter job scheduling challenges-I

- **Large scale**

Cluster size is large

Tens of thousands of nodes/workers

The number of tasks in a job can be larger

Tens of thousands of tasks in a job

-- More than 50K tasks in a job in the Cloudera trace
Datacenter job scheduling challenges-II

- **Heterogeneous workload**

  Short jobs (e.g., user facing applications)
  ---call for short response time

  Long jobs (e.g., Data backup)
  --call for mean response time guarantee
Centralized job scheduling

**Scalability problem**

A scheduler manages all the workers’ resources in a cluster.
Distributed scheduling - Sparrow

- **Low efficiency**: unbalanced probing

A scheduler needs to maintain all probes.
Hybrid scheduling-Eagle, Hawk

- All short jobs are put to reserved workers
- Scalability problem

Centralized Scheduler

Distributed Scheduler

Reserved workers: only serve short job tasks
Pigeon

Contributions

1. Introduce a master level for task distribution
   New architecture, hierarchical job scheduler

2. Fully solve scalability problem

3. High efficiency
Overview of Pigeon

Centrally manage a group of workers

Receive tasks from job schedulers

Dispatch tasks to workers

Master is job agnostic
Job scheduling in Pigeon

- Weighted fair queue (W)
- Idle worker list

Distributed Scheduler

workers
Why is Pigeon better?

Solve key challenges in existing schedulers

**Scalable:** greatly reduce status maintenance costs in job schedulers

  - Group size 100: # of master is 1% # of workers, reduce 99% status maintenance cost

**Efficiency:**
  - Remove head-of-line blocking
  - Have statistical multiplexing gain within a group

Group size 100: run at 90% load, the probability of a task finding an idle worker in a group is \(1 - 0.9^{100} = 99.99734\%\)
Modeling and Analysis

Consider a single type of jobs, the fanout degree in a job is less than the number of masters. The task queuing time in a master is a M/M/K queue (K is the group size).

Zero queueing time: job without queuing time,
The task execution time in a job is the same

Running at 30% higher utilization

50 tasks per Job

100 tasks per Job
Evaluation--Implementation

- Spark plug-in, Amazon EC2 cloud
- 120-worker cluster (3 groups in Pigeon)
- Measurement metrics: 50th, 90th and 99th percentile short and long job completion time
- Compare with state-of-the-art schedulers: Eagle and Sparrow
- Source codes: https://github.com/ruby-/pigeon/
Pigeon vs Eagle--Implementation

Eagle normalized to Pigeon

20x~30x short job performance gains
Pigeon vs Sparrow--Implementation

Sparrow normalized to Pigeon

Pigeon works in a real cluster
Evaluation—Large Scale Simulation

- Event-driven simulator
- Google, Yahoo and Cloudera traces
- Cluster size 3000--19000 workers
- Measurement metrics:
  - 50th, 90th and 99th percentile short and long job completion time
- Compare with state-of-the-art hybrid scheduler: Eagle
Pigeon is really scalable and efficient

Google trace

Slowdown = job completion time / job execution time

Big performance gains for short job at high loads
Slightly better performance gains for long jobs
Conclusion

Pigeon: a new distributed and hierarchical job scheduler, new scheduling architecture

1. Excellent scalability
   better than existing schedulers

2. High efficiency with multiplexing
Thank you!
Questions ??