Hanoi University of Science and Technology
School of Information and Communication Technology

Task Scheduling In Real-time System

Quang Khanh
khanh.tq170083@gmail.com

2nd June 2021

Content

Table of Contents

Google's Borg System

Google's Borg system is a cluster manager that runs millions of jobs, from many thousands of different applications, across a number of clusters each with up to ten thousands of machines

Borg cell is a set of machines grouped together

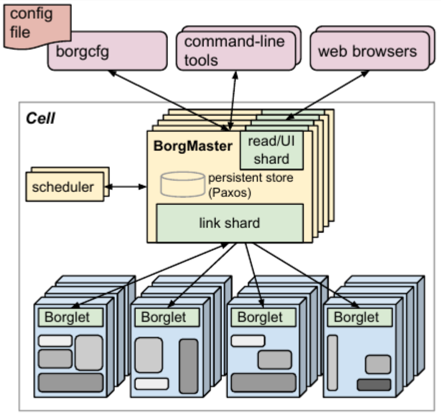


Figure: Borg architecture

Google's Borg System

BorgMaster is a logically centralized controller over a borg cell

Borglet is an agent process that runs on each machine in a cell

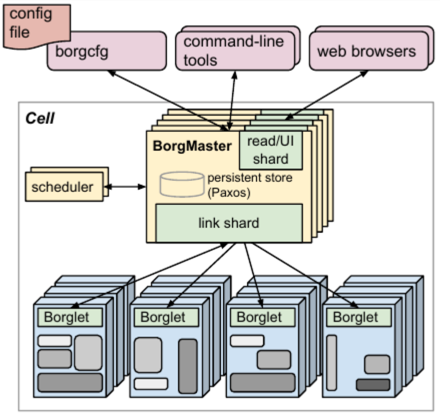


Figure: Borg architecture

Task scheduling in Borg system

When a job is submitted

It is added to pending queue

Scheduler

Scheduler scans the pending queue and assigns tasks to machines by scheduling algorithm. The algorithm has two parts:

- **feasiable checking:** to find machines on which the task could run
- **scoring:** to pick one machine to assign task to.

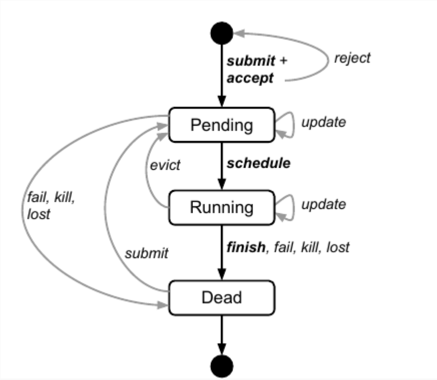


Figure: Borg architecture

Task scheduling algorithm in Borg system

Feasible checking

The scheduler find a set of machines that:

- meet the task's constraints (dependencies, packages, ...)
- have enough available resources

Scoring

The score is based on some criteria such as minimizing the number and priority of preempted tasks, and resources-related criteria are:

- **worst fit**: speading workload across all the machines uniformly
- **best fit**: tries to fill machines as tightly as possible

Table of Contents

Drawbacks

Execution time

Due to lack of certainty about how long the tasks' instructions are, the scheduler cannot estimate execution time of the tasks

Service quanlity

The solution cannot optimize execution time, leading to longer waiting-time to user

Drawbacks

Eviction

When resources of machines are not available, some running tasks would be evicted to release resources for higher-prioirty tasks

Service quanlity

The user will have to wait longer to finish their job

System effectiveness

The resources spent for the evicted tasks would be wasted. The google trace data published in 2011 shows that up to 60% of CPU resource is wasted by tasks that do not complete successfully.

Table of Contents

Length Estimation

Divide tasks to two types:

- Long-running: the task as an always running service, unspecified its instructions
- Batch-job (short-running): the task which has specific instructions depending on only its input, which takes from a few seconds to a few days to complete.

Long-running tasks

Find feasible machines and pick one of them for the task

Short-running tasks

Predict the intructions and find a machine in which the execution time is minimized

Proposed task scheduling pipeline

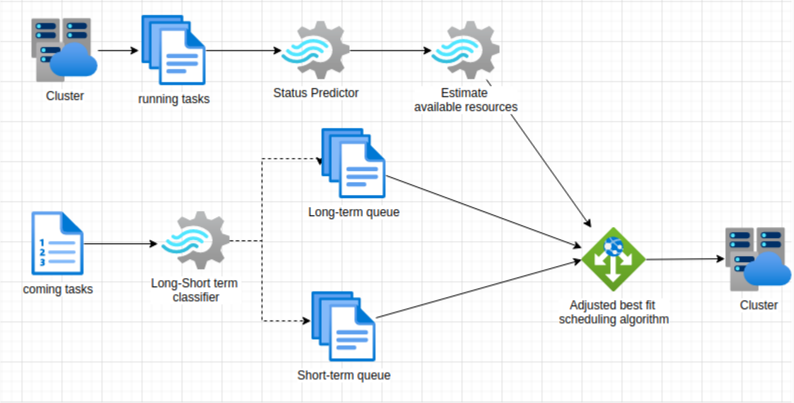


Figure: Task scheduling pipeline

Long-short term classifier

Prior-knowledge classifier

Basing on what kind of applications submitting tasks

Logistic Classifier

A logistic regression model to predict label of tasks

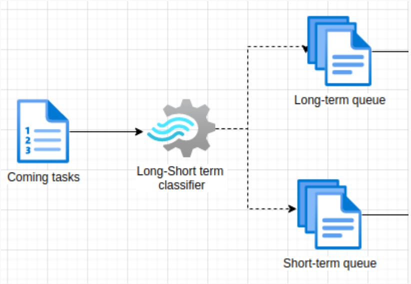


Figure: Long-short term classifier

Long-term task scheduling

Figure: Long term task scheduling

We should schedule long-term tasks first because in reality, they are always belonged to sensitive jobs such as Gmail, Docs in google trace data

Scheduling algorithm

- Best fit: Optimize resources utilization
- Worst fit: Optimize time execution and workload balancing

Proposed task scheduling pipeline

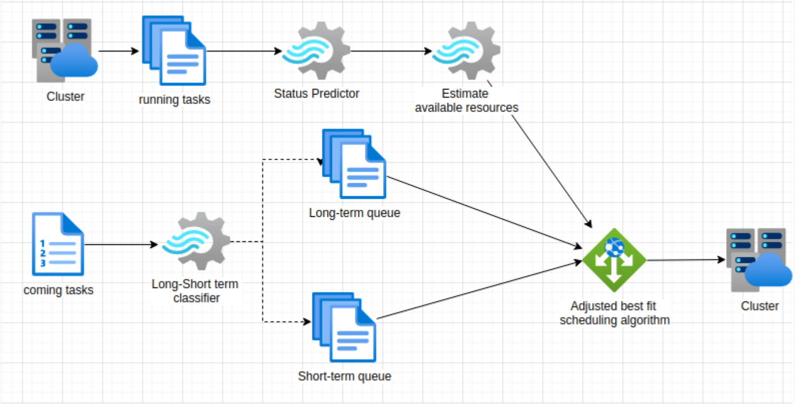


Figure: Task scheduling pipeline

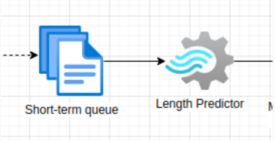


Figure: Length predictor

Task information

- resources_request
- meta_data (priority, programming language, ...)
- length: the number of instructions

Length estimation model

We expect that there is a machine learning model fitting task length data

Minimizing execution time

Task execution time fomula

$$T = \frac{length}{MIPS * cpu\_usage}$$

Make-span

$$makespan = max(T_1, T_2, ..., T_n)$$

Algorithm

Input: tasks, machines
Output: task-machine mapping
Objective: find a solution that minimize make-span of tasks

Minimizing execution time algorithm

Algorithm

Algorithm 1 pseudocode for the calculation of

Input: *L_queue*, *S_queue*, *vms*

Output: Map<Task, Vm>

```
usage ← estimate_usage(vms)
L_running_usage ← get_L_running(usage)
S_running_usage ← get_S_running(usage)
L_solution ← bestfit(L_queue, L_running_usage)
L_usage ← update(L_running_usage, L_solution)
S_solution ← bestfit(S_queue, S_running_usage)
return merge(L_solution, S_solution)
```

Proposed task scheduling pipeline

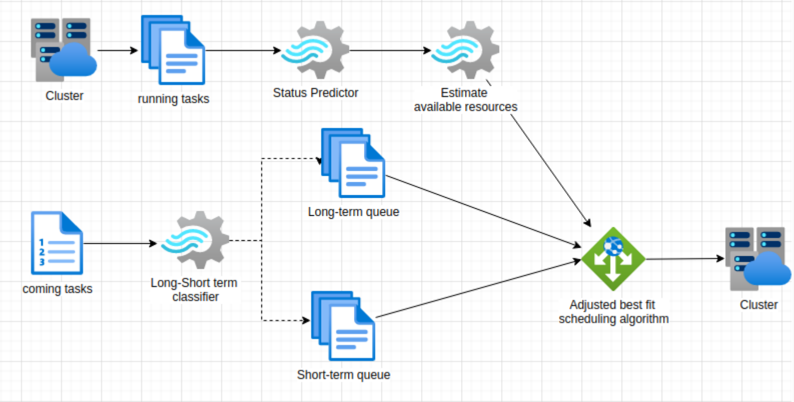


Figure: Task scheduling pipeline

Update long-short term

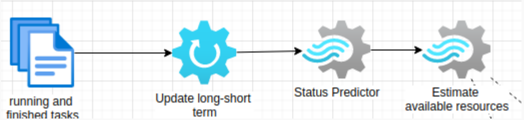


Figure: Update long-short term

When the duration of a short-term task is extremly longer than the expected, the task would be re-labled as long-term

$$T = \frac{length}{MIPS * cpu\_usage} + \epsilon$$

Update long-short term

Task execution time

$$T \sim \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(T-\mu)^2}{2\sigma^2}}$$

Task's duration

$$P(T > duration) = \int_{duration}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(t-\mu)^2}{2\sigma^2}} \, dt$$

$$= \frac{1}{2} - \phi\left(\frac{duration - \mu}{\sigma}\right)$$

α threshold

$P(T > duration) < \alpha \rightarrow$ task is re-labeled as long-term

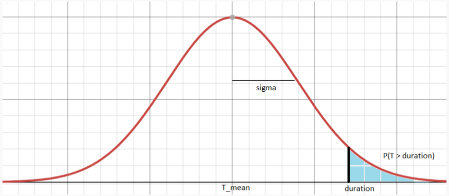


Figure: Task execution time distribution

Status predictor

Uncertain state

- tasks are scheduled over $state_1$
- tasks are executed over $state_2$

$state_1$ and $state_2$ are probably biased.

Estimate $state_2$

Basing on $state_1$, we want to estimate $state_2$ in order to get more accuracy environment information

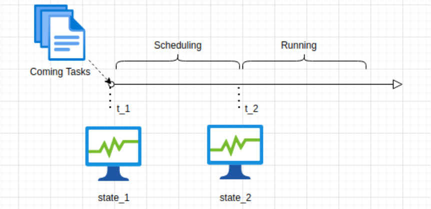


Figure: Scheduling process

Status predictor

Task state

Idea: use Markov chain to estimate transition probabilities between the states

Markov chain drawbacks

In experiment, the idea poses many problems :

- transition probabilities also depend on the duration of task
- cpu usage is also correlated to probability

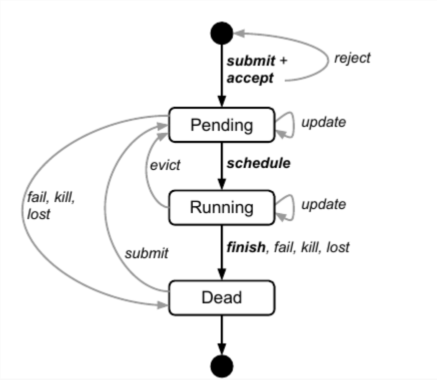


Figure: Task states transition

Status predictor

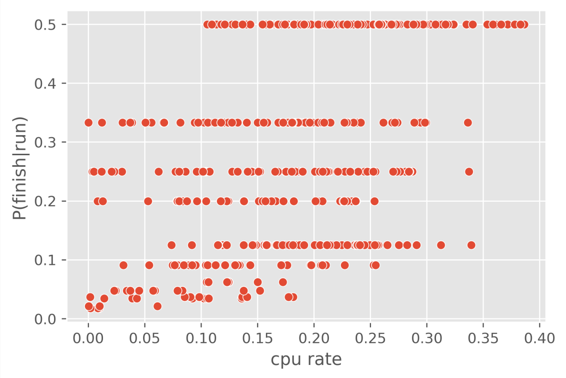


Figure: cpuRequest vs transition probability

```
Call:
lm(formula = prob ~ cpuRequest, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.304739 -0.124903  0.003684  0.117100  0.283832

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.12018    0.01649   7.287 1.59e-12 ***
cpuRequest   0.91220    0.08247  11.061  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1433 on 419 degrees of freedom
Multiple R-squared:  0.226,     Adjusted R-squared:  0.2242
F-statistic: 122.3 on 1 and 419 DF,  p-value: < 2.2e-16
```

Figure: Summary statistics

Bayesian network predictor

Bayesian network

$$P(S_{i+1}|T_{i+1}, cpu, S_i) = \mathcal{F}_{S_{i+1}}(S_i, T_{i+1}, cpu, \theta)$$

Probability function

$$\mathcal{A}_{S_{i+1}}(S, T, cpu, \theta) = \theta_0^{S_{i+1}} + \theta_1^{S_{i+1}} * S + \theta_2^{S_{i+1}} * T + \theta_3^{S_{i+1}} * cpu$$

$$\mathcal{F}_{S_{i+1}}(S_i, T_{i+1}, cpu, \theta) = \frac{\mathcal{A}_{S_{i+1}}(S, T, cpu, \theta)}{\sum_{S_k \in \{S\}} \mathcal{A}_{S_k}(S, T, cpu, \theta)}$$

state\state	pending	running	dead
pending	p11(duration, cpu)	p12(duration, cpu)	1 - p11 - p12
running	p21(duration, cpu)	p22(duration, cpu)	1 - p21 - p22
dead	p31(duration, cpu)	0	1 - p31

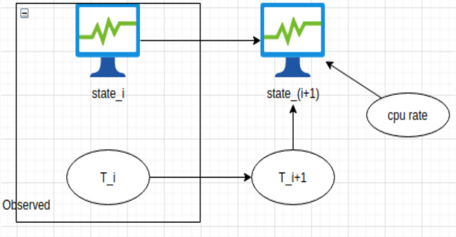


Figure: Transition network

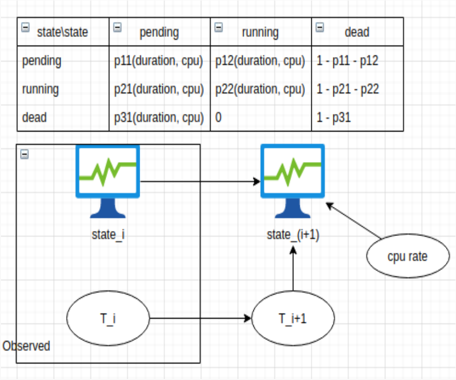
Bayesian network predictor

Probability factorization

$$
\begin{aligned}
P(S_{i+1}|S_i, T_i, cpu) &= \int P(S_{i+1}, T_{i+1}|S_i, T_i, cpu)dT_{i+1} \\
&= \int P(S_{i+1}|S_i, T_{i+1}, cpu)P(T_{i+1}|T_i)dT_{i+1} \\
&= E_{T_{i+1}\sim\mathcal{N}(T_i+\mu,\sigma^2)}\mathcal{F}_{S_i,cpu,\theta}(T_{i+1}) \\
&\sim \mathcal{F}_{S_i,cpu,\theta}(T_i+\mu)
\end{aligned}
$$

Use gradient ascend to find θ that maximize likelihood

$$\mathcal{L}(\theta : d) = log(\mathcal{F}_{S_i,cpu,\theta}(T_i+\mu))$$



state\state	pending	running	dead
pending	p11(duration, cpu)	p12(duration, cpu)	1 - p11 - p12
running	p21(duration, cpu)	p22(duration, cpu)	1 - p21 - p22
dead	p31(duration, cpu)	0	1 - p31

Figure: Transition network

Resources estimation

Tasks assigned to machines

$\mathcal{S}$ is a set of tasks which are assigned to the machine and haven't finished

Resources usage estimation

$$resource\_usage = \sum_{task \in \mathcal{S}} P(running) * resouce\_usage\_of\_task$$

$$available\_resource = total\_capacity - resource\_usage$$

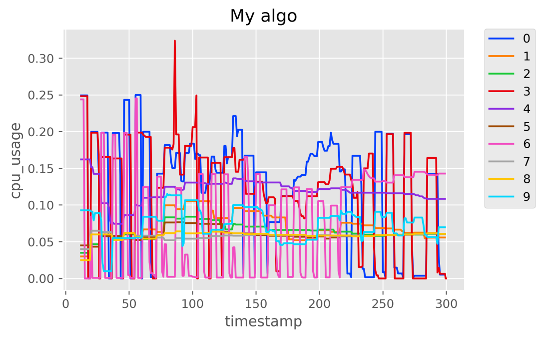
- P(running): probability that the task continue running after scheduling time

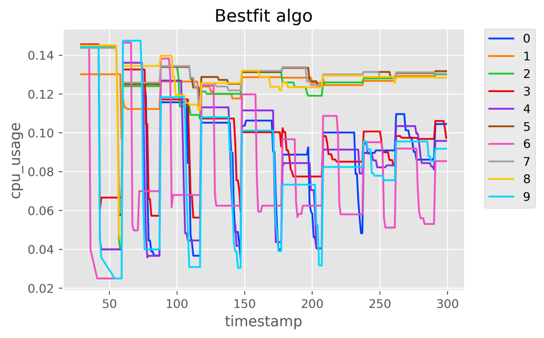
Scheduling information

Coming tasks would be scheduled over resources estimation, not observed resources at scheduling time

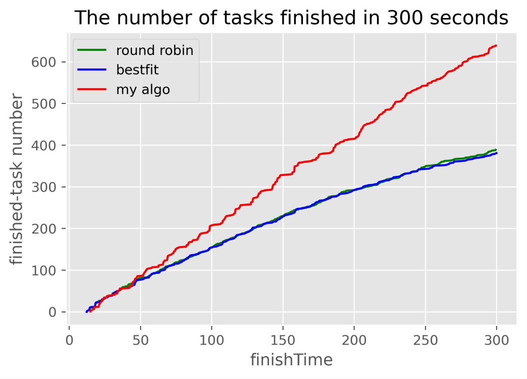
Table of Contents

Cpu usage experiment



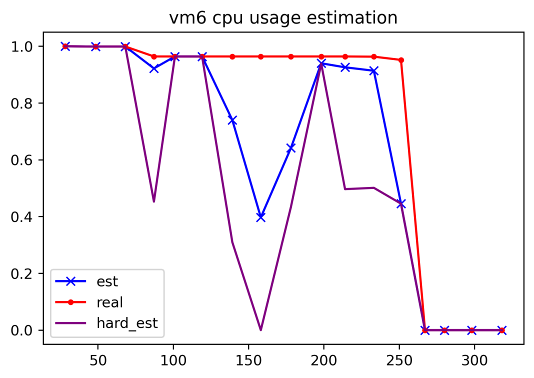


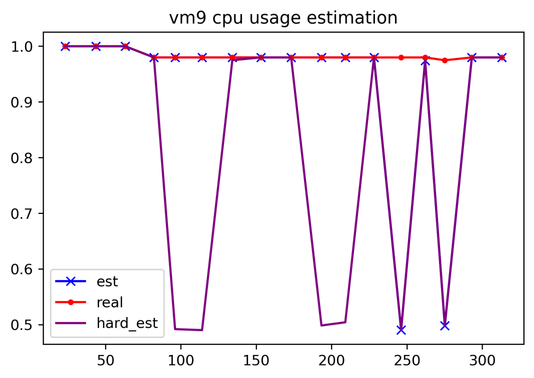
Number of tasks finished



Statistics	RoundRobin	BestFit	MyAlgo
count	390.00	382.00	640.00
mean	28.14	24.12	13.15
std	26.74	22.77	15.51
min	1.99	2.00	2.62
25%	8.83	7.83	5.38
50%	21.25	18.07	7.23
75%	36.46	31.19	14.82
max	148.15	131.54	109.16

Resouces estimation





Resources estimation

Comparison between available-resouces estimation vs no estimation

	statistic	no-estimated	estimated
0	count	366.000000	378.000000
1	mean	32.177039	30.375850
2	std	23.648688	23.509136
3	min	6.644639	6.581494
4	25%	17.068980	15.700250
5	50%	27.211863	24.485500
6	75%	37.649000	33.975500
7	max	141.561000	138.182000

Thank you for listening!