Using Straggler Replication to Reduce Latency in Large-scale Parallel Computing

Da Wang, Gauri Joshi, Gregory Wornell
Problem: Stragglers in Parallel Computing

- A job with hundreds of parallel tasks
- Machine response time can vary due to virtualization, congestion etc.
- The slowest tasks are the bottleneck in job completion

[Dean “Tail at Scale” 2013]

<table>
<thead>
<tr>
<th>Latency</th>
<th>50%ile</th>
<th>99%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 task finishes</td>
<td>1ms</td>
<td>10ms</td>
</tr>
<tr>
<td>All tasks finish</td>
<td>40ms</td>
<td>140ms</td>
</tr>
</tbody>
</table>
Solution: Replication of Stragglers

Re-run the stragglers when \( p \) fraction of tasks are remaining

Task 1

Task 2

Task 3

Task n

Task n
Related Previous Work

Task Replication in Systems Literature
- First used in MapReduce [Dean 2008] via back-up tasks
- Further developed in [Zaharia 2008], [Ananthanarayanan 2010] etc

Used in practice, but little theoretical analysis so far

Our Contributions
- Provide design insights on how to schedule task replication to reduce delay, with efficient use of additional resources

System Model

- A job with $n$ parallel tasks, $n$ is large
- Finish time of a task, $X \sim F_X$, i.i.d. across machines.

Remark on the i.i.d assumptions:

- From cloud user’s point of view, all rented machines are approx. identical.
Performance Metrics

- Expected Latency $E[T] =$ Expected Time when all tasks finish
- Expected Cost $E[C]=$ Expected total machine time spent, normalized by # of tasks

Remark on cost metric

- There could be other costs – network, memory usage, etc
Outline

Analysis of $E[T]$ and $E[C]$ using Extreme Value Theory
- Tail behavior of $F_X$ is a key factor affecting the $E[T]$-$E[C]$ trade-off

Heuristic Algorithm to find best replication strategy
- Compare with back-up tasks in MapReduce using Google trace data
Replication Policy: When to re-run?

- Re-run the stragglers when $p$ fraction of tasks are left

![Diagram showing task statuses](image)

$\checkmark$ Task 1
$\checkmark$ Task 2
$\checkmark$ Task 3
Task 4

$p = 25\%$
Replication Policy: When to re-run?

- Re-run the stragglers when $p$ fraction of tasks are left

$p = 25\%$

![Diagram showing replications and stragglers with $p = 25\%$](image)
Replication Policy: How many replicas?

- Re-run the stragglers when $p$ fraction of tasks are left
- Run $r$ additional replicas

\[ p = 25\% \]
\[ r = 1 \]

Wait for any 1 out of $r+1$ replicas to finish, and cancel the rest
Replication Policy: Relaunch or not?

- Re-run the stragglers when $p$ fraction of tasks are left
- Run $r$ additional replicas

$p = 25\%$
Relaunch

Relaunching the original task on another machine
Replication Policy: Relaunch or not?

- Re-run the stragglers when $p$ fraction of tasks are left
- Run $r$ additional replicas

$p = 25\%$
$r = 1$
No Relaunch

Keeping the original task
Problem Formulation

Given $n$ tasks, and task finish time distribution $F_X$,

Design Parameters
- $p$: Fraction of tasks left when we replicate
- $r$: Number of additional replicas
- Relaunch original straggling task or not

Performance Metrics
- Latency $E[T]$
- Cost $E[C]$
Evaluating Expected Latency $E[T]$

- Wait for $(1-p)n$ tasks to finish
- Launch replicas of the $pn$ stragglers
  - Time for 1 out $r+1$ copies to finish $Y \sim F_Y = g(F_X, r, \text{kill/keep})$
  - For e.g. $r = 1$ with task-killing $\Rightarrow (1-F_Y) = (1-F_X)^2$

$$T = X_{(1-p)n:n} + Y_{pn:pn}$$

Notation $X_{k:n}$: $k^{th}$ smallest of $n$ i.i.d. rvs $X_1, X_2, .. X_n$
Evaluating Expected Latency $E[T]$

$$E[T] = E[X(1-p)n:n] + E[Y_{pn}:pn]$$

Central Value Theorem \text{ for } n \to \infty

$$F_{X}^{-1}(1 - p)$$

Extreme Value Theorem \text{ for } n \to \infty

Different behavior for Exponential, Light or Heavy tailed $Y$

Asymptotic approx for $n \to \infty$ is close to simulation even for $n \sim 300$
Exercise: Task Execution Time $X \sim \text{Exp}(\mu)$

\[
\mathbb{E}[T] = \mathbb{E}[X^{(1-p)n:n}] + \mathbb{E}[Y_{pn:pn}]
\]

Central Value Theorem \( n \to \infty \)

\[
F_X^{-1}(1 - p)
\]

Extreme Value Theorem \( n \to \infty \)

Different behavior for Exponential, Light or Heavy tailed Y
Comparing Theoretical Analysis with Simulations

\[ X \sim 1 + \text{Exp}(1), \text{ and } n = 400 \]

\[ \mathbb{E}[T] \]

\[ n \]

\[ E[T] \]

\[ p = 0.1 \]

\[ p = 0.2 \]

\[ r = 1 \text{ & kill} \]

\[ r = 1 \text{ & keep} \]

\[ r = 2 \text{ & kill} \]

\[ r = 2 \text{ & keep} \]

Gauri Joshi

Straggler Replication to Reduce Latency
Evaluating Expected Cost $E[C]$

$$E[C] = \frac{1}{n} \sum_{i=1}^{(1-p)n} E[X_{i:n}] + \frac{np}{n} E[T^{(1)}] + \frac{1}{n} \sum_{j=1}^{pn} (r + 1) E[Y]$$

$$= \int_0^{1-p} F_X^{-1}(h) dh + pF_X^{-1}(1 - p) + (r + 1)pE[Y] + O(1/n)$$

By Central Value Theorem
Case: Shifted Exponential (Exp. tail)

$X \sim 1+\text{Exp}(1)$, and $n = 400$

- Increasing $p$ and $r$ reduces latency but increases cost
- Killing a straggling task never helps!

---

Gauri Joshi

Straggler Replication to Reduce Latency
Case: Pareto (Heavy tail)

\[ X \sim \text{Pareto}(2, 2) \text{ and } n = 400 \]

Latency and cost both reduce for small \( p! \)
Outline

Analysis of $E[T]$ and $E[C]$ using Extreme Value Theory
- Tail behavior is a key factor affecting the $E[T]$-$E[C]$ trade-off

Heuristic Algorithm to find best replication strategy
- Comparing with back-up tasks in MapReduce using Google trace data
Example: Job with 1026 tasks

Google Cluster Data

![Graph showing task duration vs. fraction of samples]

Straggling Tasks
Simulations using Google Cluster Data

Latency-Cost Trade-off

Careful choice of replication strategy can be better than the default in MapReduce

Gauri Joshi

Straggler Replication to Reduce Latency
Example: Job with 488 tasks

Google Cluster Data

Gauri Joshi
Straggler Replication to Reduce Latency
Simulations using Google Cluster Data

Latency-Cost Trade-off

Gauri Joshi

Straggler Replication to Reduce Latency
Heuristic Search of the Best Strategy

May be hard to use our analysis to optimize the strategy for any $F_X$

- Analysis of $E[T]$ and $E[C]$ after replication can be hard

**ESTIMATION**

- Estimate $F_X$ from traces of task execution time
- Use empirical $F_X$ to estimate $J = E[T] + \mu E[C]$ for given $p$, $r$, relaunch/not

**HEURISTIC ALGORITHM**

1. For given $p$, choose $r$, and the kill/keep strategy that minimizes $J$
2. Perform gradient descent on $p$
Heuristic Algo: Resulting $E[T]$ and $E[C]$

- Run heuristic algorithm with different $\mu$, to minimize $J = E[T] + \mu E[C]$
- $r=1$, without relaunch ($l=1$): Back-up tasks option in MapReduce

\[
\begin{align*}
\text{Baseline: } & p = 0 \\
\mu = 3 \quad & \text{Heuristic: } (p, r, l) = (0.122, 2, 1) \\
\mu = 2 \quad & \text{Heuristic: } (p, r, l) = (0.086, 4, 1) \\
\mu = 1 \quad & \text{Heuristic: } (p, r, l) = (0.145, 5, 1)
\end{align*}
\]
Concluding Remarks

SUMMARY
- Tail behavior is important in choosing the right policy
  - Pareto, Shifted Exponential etc.
- Heuristic algorithm to find good replication policy given traces of execution time

RELATED AND FUTURE WORK
- Queueing of jobs (next class)
- Online algorithm to learn $F_X$ and schedule simultaneously