Effective Straggler Mitigation

Attack of the Clones

Greg Lyras & Johann Mifsud
Outline

- Introduction
- Related Work
- Goals
- Design
- Results
- Evaluation
- Summary
What are Stragglers?

- Tasks that take more time to complete than regular
- Why?
  - Faulty Disc
  - High loads
- Unpredictable in the general case
- Why are “stragglers” a problem?
  - Reduces performance. Job is done only when all the tasks are finished.
Environment

→ Metric: Progress Rate
  ◆ the size of input data divided by duration
→ Use work load of Facebook/Bing
  ◆ 150-node cluster

<table>
<thead>
<tr>
<th></th>
<th>Facebook</th>
<th>Microsoft Bing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates</td>
<td>Oct 2010</td>
<td>May-Dec* 2009</td>
</tr>
<tr>
<td>Framework</td>
<td>Hadoop</td>
<td>Dryad</td>
</tr>
<tr>
<td>File System</td>
<td>HDFS [9]</td>
<td>Cosmos</td>
</tr>
<tr>
<td>Jobs</td>
<td>375K</td>
<td>200K</td>
</tr>
<tr>
<td>Cluster Size</td>
<td>3,500</td>
<td>Thousands</td>
</tr>
</tbody>
</table>

* One week in each month

Table 1: Details of Facebook and Bing traces.
Straggler Mitigation

Blacklisting
- AIM: find machines that are not performant. (Faulty Disks)
- FB & MS blacklist 10% of their machines
- Stragglers occur only on blacklisted machines? What can create stragglers?

Speculation Execution
- AIM: Analyse progress of the tasks of a job and duplicate tasks that are slower
- LATE and MANTRI
- Getting the analysis data takes time
State-of-the-art: Small vs Large jobs

- Large jobs handled well
- Small jobs handled inefficiently
  - LATE 8x slower
  - MANTRI 6x slower

![Graph showing median progress rate for Facebook and Bing](a) LATE

![Graph showing median progress rate for Facebook and Bing](b) Mantri
Importance of small tasks

➔ 90% jobs consume 6% (FB) or 11% (BING)
➔ only 2% of the overall resources

(a) Heavy-tail

(b) Power Law

\[ y = (9E+06)x^{-1.9} \]
Goals

➔ Speedup execution
  ◆ Potential average completion time
    ● Facebook - 47%
    ● Bing - 29%
  ◆ Potential for small jobs
    ● Facebook - 49%
    ● Bing - 38%

➔ Compete against State of the Art

➔ Keeping resource usage low
  ◆ only an extra 5%
Solution

Proactive Approach

◆ Launch multiple clones without analysis
◆ Use result of the clone that finishes first
◆ Eschew waiting
◆ Cloning of 90% utilization increase by 3%

Are we done already?

◆ What should we clone?
◆ Should there be a limit of resources used?
◆ What if the cluster is full?
◆ Intermediate data?
Granularity - Jobs

- Multiple clones of the entire job
- Result is taken from the first job that finishes
- Advantages
  - Simple
  - Easy to implement
Granularity - Tasks

→ Multiple clones of the same task
→ Result taken from the clone that finishes first in the group
→ Complex to implement
Granularity - Performance

(a) Job-level Cloning

(b) Task-level Cloning
Budget Cloning

- Limit number of clones
- Stops cloning under heavy loads
- Creates equal number of clones to all tasks

```
1: procedure CLONE(n tasks, p, ε)
   C: Cluster Capacity, U: Cluster Utilization
   β: Budget in fraction, BU: Utilized budget in #slots
   c = \left\lfloor \log \left( \frac{1 - (1 - \epsilon)^{1/n}}{\log p} \right) \right\rfloor
2: if (BU + c \cdot n) ≤ (C \cdot β) and (U + c \cdot n) ≤ τ then
   > Admission Control: Sufficient capacity to create c clones for each task
3:   for each task t do
4:     Create c clones for t
   BU ← BU + c \cdot n
```
Intermediate Data

➔ Extreme Strategies
  ◆ Contention Avoidance Cloning (CAC)
  ◆ Contention Cloning (CC)

➔ Novelty
  ◆ Delay Assignment
Contestion Avoidance Cloning (CAC)

- Exclusive read
- Weakness:
  Diminish benefit from cloning
Contention Cloning (CC)

➔ All read from the first finished upstream clone

➔ Weakness:
  ◆ Slow down of transfer of data between phases
Statistic of CC/CAC

→ CAC requires more clones to decrease straggling
→ CC affects performance because of slowdown of transfer

Figure 5: CAC vs. CC: Probability of a job straggling.
Delay Assignment

- Wait a time $\omega$ to get exclusive access
- Contention otherwise
Delay Assignment - in search for $\omega$

- Calculate expected duration for reading
- Estimate overall duration of the task
- Find $\omega$ that minimizes the duration
Dolly Design Choices - Proposal

➔ Clone Tasks
➔ Delay Assignment
➔ Budget Cloning
➔ On Hadoop
Results - Overall Improvement

➔ Significant speedup
➔ Even on large jobs
Results - Delay assignment (more is better)

- Bing workload (worse for them)
- Delay assignment works for most of the Bins (1.5 x-2x faster than CAC)
Results - In search of $\beta$ (cloning budget)

- On average a $\beta$ 3-5% is enough.
- Gains by eliminating stragglers in small jobs
- So small budget

![Graphs showing reduction in completion time with increasing cloning budget]
Results - Budget Policy

➔ Preemption, little gain (1-2%)
➔ “pure”, big difference (14%)
◆ large jobs deny resources to small ones
➔ Current Budget policy is good enough
Evaluation

→ Identify a weakness in the State-of-the-Art
→ Prove it exists
→ Specify their target (small jobs)
→ Real life production workload (Bing and Facebook)
→ They reach their goals
  ◆ Speedup 34% to 46% (LATE and Mantri, + blacklisting)
  ◆ Only 5% overhead on resources!
Evaluation (cont’d)

- Google’s MapReduce
  - Backup Tasks
- Amazon Dynamo
  - Backup requests - hinted handoff, R/W/N
Future work

- Compatibility with caching systems
- Multiple frameworks
- Deal with changes in job distributions
Summary

- Stragglers still remain after state-of-the-art mitigation
- Clones to the rescue (small jobs, low resources)
- Intermediate data (delay assignment)
- Result 34-46% speedup w/ 5% extra resources
Thank you for your attention

→ Questions? (Ask! 😊 )