The Case for Tiny Tasks in Compute Clusters


* UC Berkeley, + ICSI
Map Reduce/Spark/Dryad Job
Use smaller tasks!

**Today's tasks**

<table>
<thead>
<tr>
<th>Slots</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**Tiny Tasks**

<table>
<thead>
<tr>
<th>Slots</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Why?  How?  Where?
Why?  How?  Where?
Problem: Skew and Stragglers

Contended machine?  
Data skew?
Benefit: Handling of Skew and Stragglers

As much as 5.2x reduction in job completion time!
Problem: Batch and Interactive Sharing

Clusters forced to trade off utilization and responsiveness!

 tijd

High priority interactive job arrives
Benefit: Improved Sharing

High-priority tasks not subject to long wait times!
Benefits: Recap

(1) Straggler mitigation

(2) Improved sharing

Mantri (OSDI ‘10)
Scarlett (EuroSys ‘11)
SkewTune (SIGMOD ‘12)
Dolly (NSDI ‘13)
...

Quincy (SOSP ‘09)
Amoeba (SOCC ‘12)
...
Why?  How?  Where?
Scheduling requirements:

High Throughput (millions per second)

Low Latency (milliseconds)

Distributed Scheduling (e.g., Sparrow Scheduler)
Use existing thread pool to launch tasks
Use existing thread pool to launch tasks
+ Cache task binaries

Task launch = RPC time (<1ms)
Smallest efficient file block size: 8MB

Distribute Metadata

(à la Flat Datacenter Storage, OSDI ‘12)
Schedule task
Launch task
Read input data
Execute task + read data for next task

Tons of tiny transfers!

Framework-Controlled I/O
( enables optimizations, e.g., pipelining )
How low can you go?

8MB disk block

100’s of milliseconds
Why?  How?  Where?
Original Reduce Phase

Reduce Task 1

Tiny Tasks = ?
Splitting Large Tasks

• Aggregation trees
  — Works for functions that are associative and commutative

• Framework-managed temporary state store

• Ultimately, need to allow a small number of large tasks
Tiny tasks mitigate stragglers + Improve sharing

Distributed scheduling → Launch task in existing thread pool → Distributed file metadata → Pipelined task execution

Questions? Find me or Shivaram: