Lecture 16.4
Spark Performance

EN 600.320/420
Instructor: Randal Burns
10 April 2017
Spark Concepts Review

- Lazy evaluation and pipelining
- Distributed memory and reuse (persist)
  - Iterative algorithms
- Data parallelism through partitioning
  - Hash and range partitions
- Fault tolerance
  - Lineage
  - Checkpoints: wide-dependencies and long lineages
How much better?

- Lots: for iterative algorithms
  - Hadoop is the WRONG tool, but people have used it anyway because of the easy access to massive parallelism

Figure 7: Duration of the first and later iterations in Hadoop, HadoopBinMem and Spark for logistic regression and k-means using 100 GB of data on a 100-node cluster.
Spark/DSM/MPI

- I can do all of these things in distributed shared memory and the message passing interface
  - Yes, but must manage distribution and recovery
  - Spark provides transparent parallelism for a broader class of applications than MR and easier than other frameworks
Spark/DSM/MPI

- Chart is a little unfair, but makes the same point

<table>
<thead>
<tr>
<th>Aspect</th>
<th>RDDs</th>
<th>Distr. Shared Mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads</td>
<td>Coarse- or fine-grained</td>
<td>Fine-grained</td>
</tr>
<tr>
<td>Writes</td>
<td>Coarse-grained</td>
<td>Fine-grained</td>
</tr>
<tr>
<td>Consistency</td>
<td>Trivial (immutable)</td>
<td>Up to app / runtime</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>Fine-grained and low-overhead using lineage</td>
<td>Requires checkpoints and program rollback</td>
</tr>
<tr>
<td>Straggler mitigation</td>
<td>Possible using backup tasks</td>
<td>Difficult</td>
</tr>
<tr>
<td>Work placement</td>
<td>Automatic based on data locality</td>
<td>Up to app (runtimes aim for transparency)</td>
</tr>
<tr>
<td>Behavior if not enough RAM</td>
<td>Similar to existing data flow systems</td>
<td>Poor performance (swapping?)</td>
</tr>
</tbody>
</table>

Table 1: Comparison of RDDs with distributed shared memory.