Hadoop Map-Reduce Tuning and Debugging

Arun C Murthy

acmurthy@apache.org

Yahoo! Grid Team, CCDI

Existential Angst: Who Am I?

- Lowly Engineer, CCDI Yahoo!
  - Design, review, and implement features in Hadoop, specifically Map-Reduce (and security)
  - Working on Hadoop full time since March 2006
- Apache Software Foundation
  - Hadoop Core Committer
  - Member of Hadoop Program Management Committee

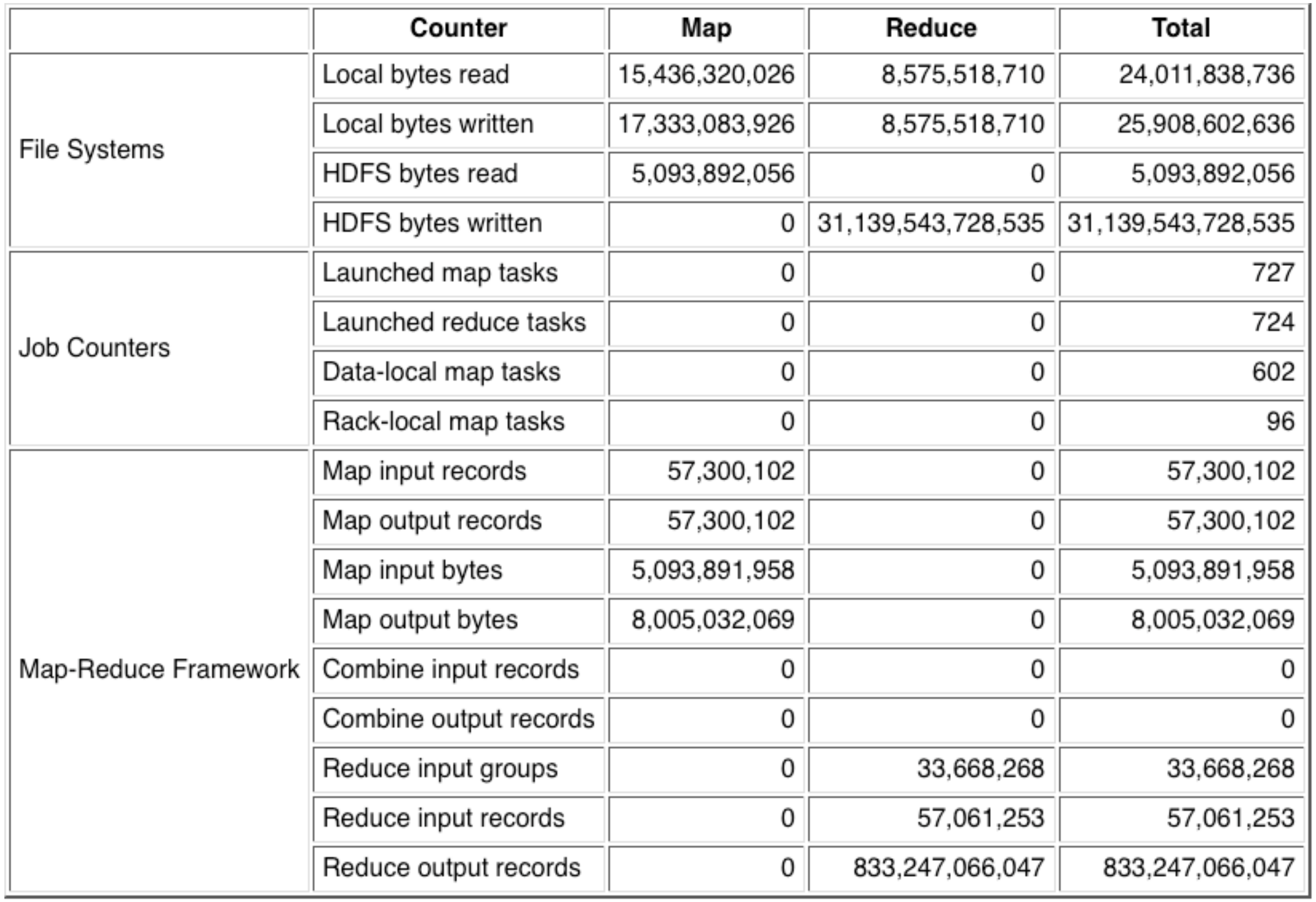
Topical Matters

- Peek inside your MR application
- Tuning
- Debugging (god forbid!)

Counters …

- Often MR applications have countable ‘events’
- For e.g. the Map-Reduce framework ‘counts’ the bytes read/write on HDFS and the local filesystem
- To define your own:
  - `static enum Counter {C1, C2}`
  - `reporter.incrCounter{Counter.C1, 1}`

Counters continued…



	Counter	Map	Reduce	Total
File Systems	Local bytes read	15,436,320,026	8,575,518,710	24,011,838,736
	Local bytes written	17,333,083,926	8,575,518,710	25,908,602,636
	HDFS bytes read	5,093,892,056	0	5,093,892,056
	HDFS bytes written	0	31,139,543,728,535	31,139,543,728,535
Job Counters	Launched map tasks	0	0	727
	Launched reduce tasks	0	0	724
	Data-local map tasks	0	0	602
	Rack-local map tasks	0	0	96
Map-Reduce Framework	Map input records	57,300,102	0	57,300,102
	Map output records	57,300,102	0	57,300,102
	Map input bytes	5,093,891,958	0	5,093,891,958
	Map output bytes	8,005,032,069	0	8,005,032,069
	Combine input records	0	0	0
	Combine output records	0	0	0
	Reduce input groups	0	33,668,268	33,668,268
	Reduce input records	0	57,061,253	57,061,253
	Reduce output records	0	833,247,066,047	833,247,066,047

Tuning Map-Reduce Applications

- Where do I start?
  - User code (the less said, the better!)
    - Use `configure` and/or `close`
    - Use the `OutputCommitter` and `setup`/`cleanup` tasks
  - The framework
    - Input
    - Data-path
    - Output

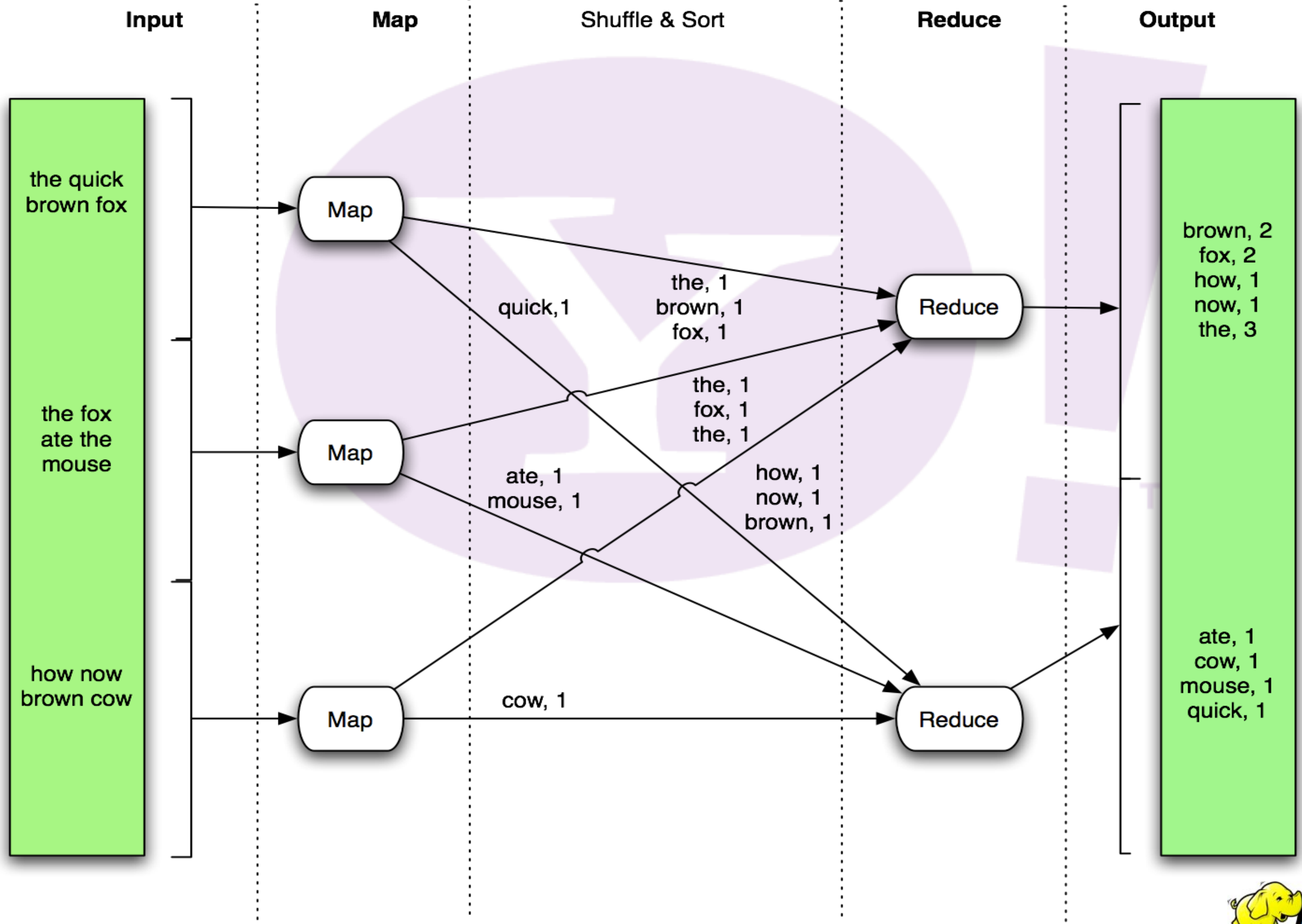
Tuning – A Diversion

- Tell HDFS and Map-Reduce about your network!
  - Rack locality script: `topology.script.file.name`
- Number of maps
  - Data locality
- Number of reduces
  - You do not, I repeat, do not, need a single output file!

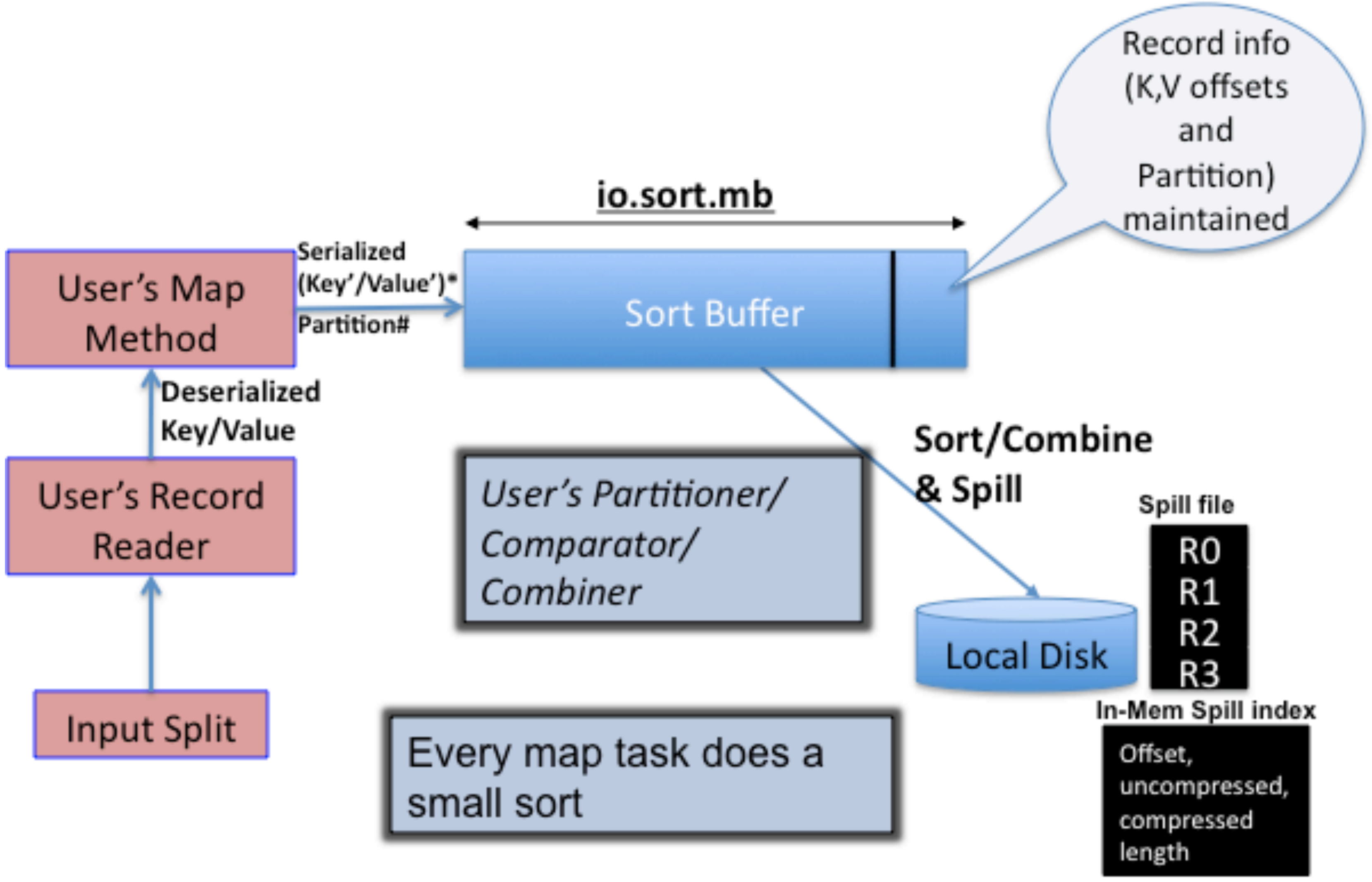
Tuning – Step One of Three: The Input

- Amount of data processed per Map
  - Consider fatter maps
  - Custom `InputFormat`:
    - `InputSplit`
    - `RecordReader`

Tuning – Step Two of Three: The Data-Path

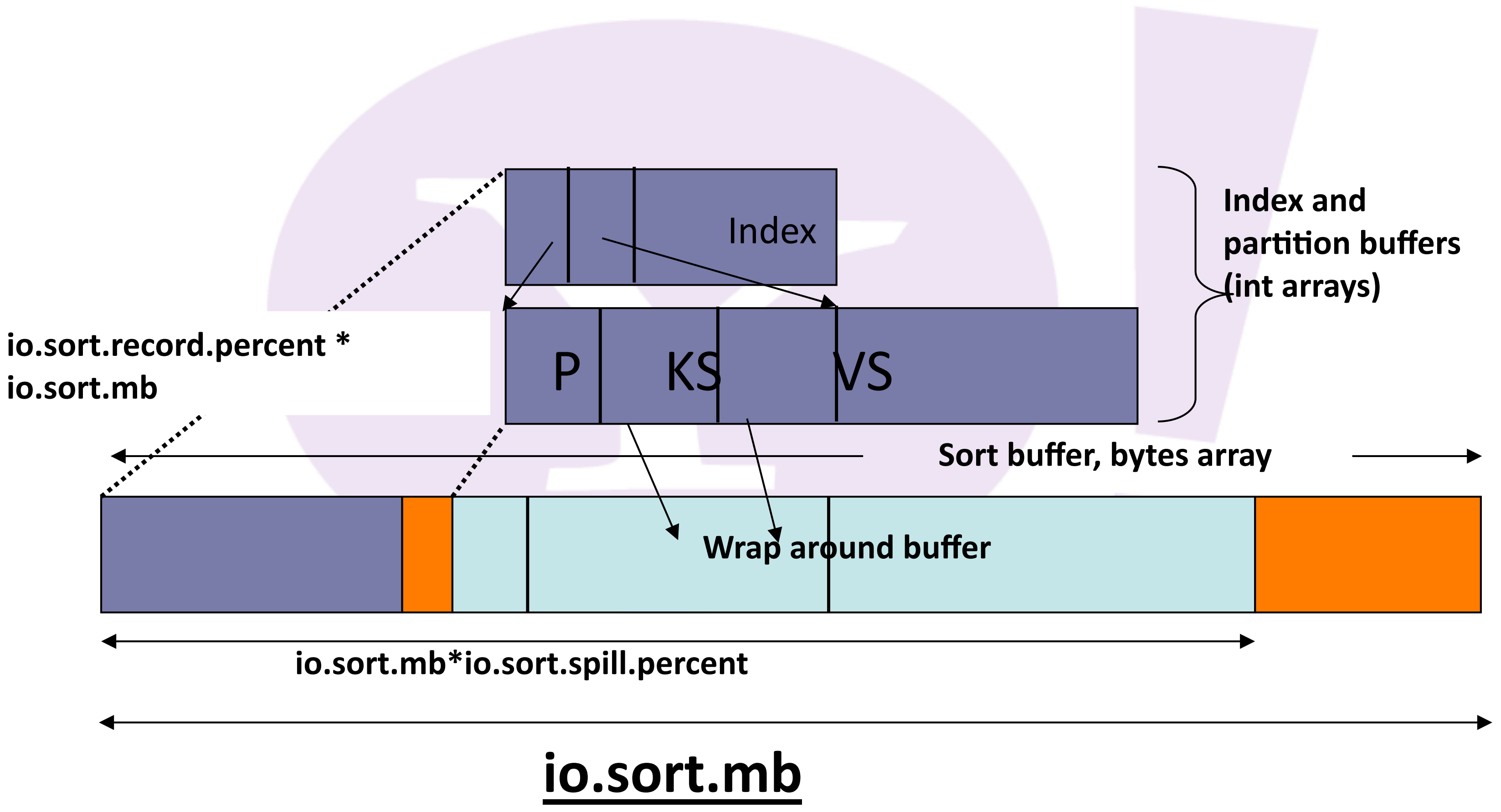


Tuning – The Mapper



Record info (K,V offsets and Partition) maintained
io.sort.mb
User's Map Method
Serialized (Key'/Value')*
Partition#
Sort Buffer
Deserialized Key/Value
User's Record Reader
Input Split
User's Partitioner/ Comparator/ Combiner
Sort/Combine & Spill
Spill file
R0
R1
R2
R3
Local Disk
In-Mem Spill index
Offset, uncompressed, compressed length
Every map task does a small sort

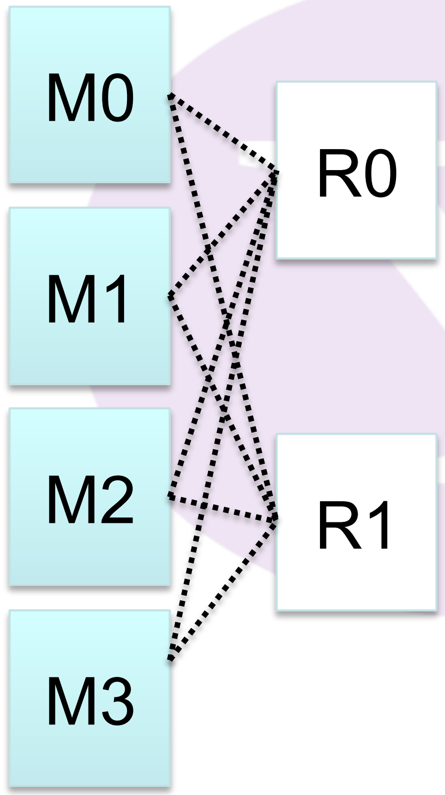
Tuning – The Mapper


Index
P
KS
VS
Index and partition buffers (int arrays)
io.sort.record.percent * io.sort.mb
Sort buffer, bytes array
Wrap around buffer
io.sort.mb*io.sort.spill.percent
io.sort.mb

Tuning – The Mapper

- `io.sort.mb`
  - Controls the sort buffer size
- `io.sort.factor`
  - Controls the number of files simultaneously merged (recommendation: 100)
- `io.sort.record.percent`
  - Controls the number of records that can be collected
- `io.sort.spill.percent`
  - Soft limit that controls when sort/spill starts

Tuning – The Shuffle



Shuffle: MxR number of intermediate file transfers to copy map-outputs to the reducers

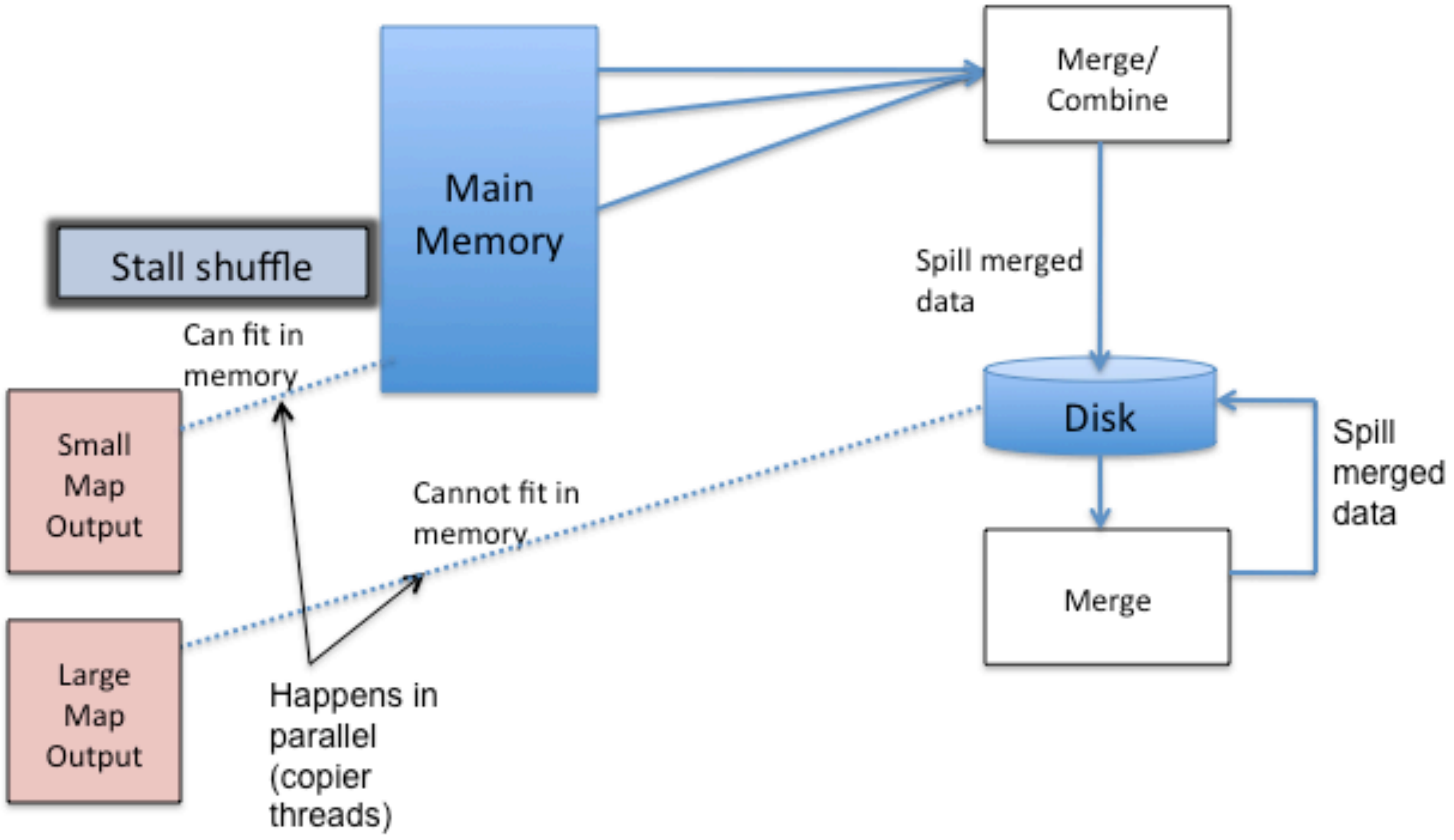
Tuning – The Shuffle

- Map-side
  - `Partitioner`
  - Use `combiners`: the faster way to copy data is to do less of it!
  - Compression for map-outputs
    - `mapred.compress.map.output`
    - `mapred.map.output.compression.codec`
    - `Native compression libraries (lzo)`

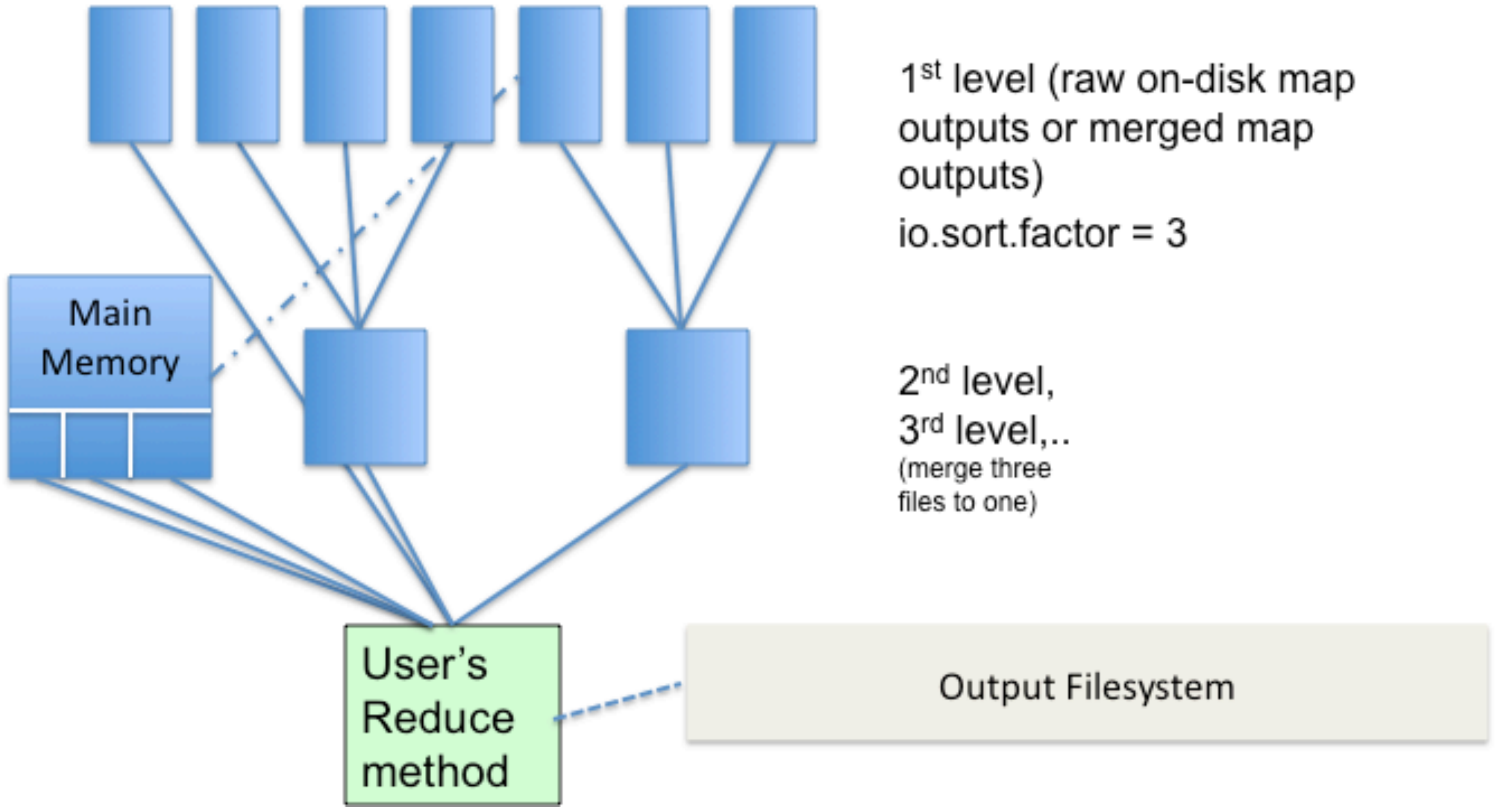
Tuning – The Shuffle

- TaskTracker
  - Jetty threads on the TaskTracker
    - `tasktracker.http.threads`
  - In-memory Index Cache
    - `mapred.tasktracker.indexcache.mb`

Tuning – The Shuffle



Tuning – The Shuffle



1st level (raw on-disk map outputs or merged map outputs)

io.sort.factor = 3

2nd level,
3rd level,..
(merge three files to one)

Tuning – The Shuffle

- Reduce-side
  - mapred.reduce.parallel.copies
  - mapred.reduce.copy.backoff
  - mapred.job.shuffle.input.buffer.percent
  - mapred.job.shuffle.merge.percent
  - mapred.inmem.merge.threshold
  - mapred.job.reduce.input.buffer.percent

Tuning – Step Three of Three: The Output

- `OutputCommitter`
- `MultipleOutputs / MultipleOutputFormat`
- Do you really need 3 replicas?

Tuning - Miscellaneous

- Speculative execution
- Heap size for the child
  - `mapred.child.java.opts`
- Re-use jvm for maps/reduces
  - `mapred.job.reuse.jvm.num.tasks`
- Last, not least: Raw Comparators

Tuning - RawComparator

```
public class MyKeyClass implements WritableComparable {
 // Some data
  private int counter;
  private Text bigText;

  public void write(DataOutput out) throws IOException {
     out.writeInt(counter);
     bigText.write(out);
  }

  public void readFields(DataInput in) throws IOException {
     counter = in.readInt();
     bigText.readFields(in);
  }

  public int compareTo(MyKeyClass o) {
    int thisCounter = this.counter;
    int thatCounter = o.counter;
    return (thisCounter < thatCounter ? -1 : (thisCounter==thatCounter ? 0 :
     1));
  }
}
```

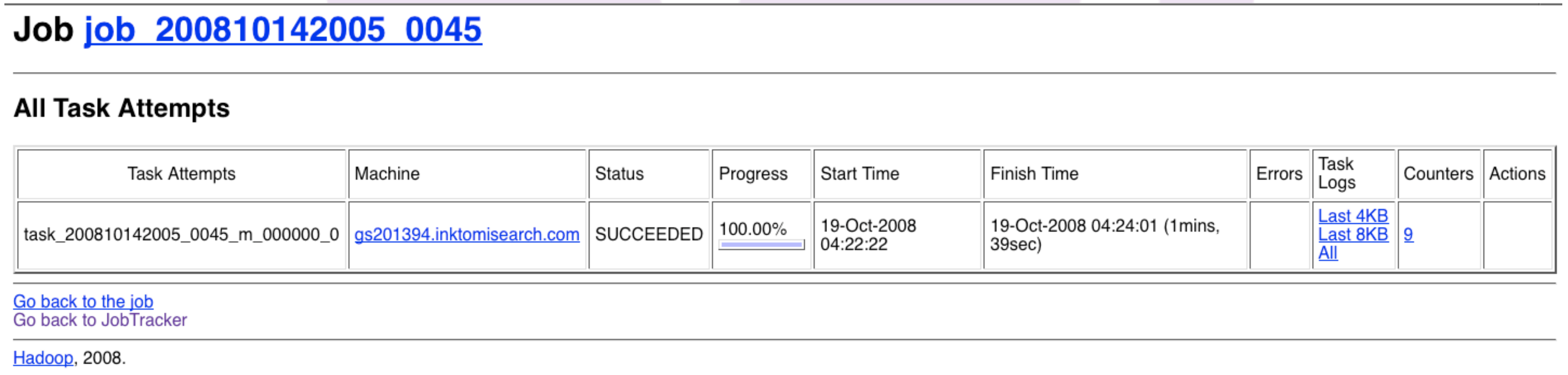
```
public static class Comparator
extends WritableComparator {
  public Comparator() {
    super(MyKeyClass.class);
  }

  public int compare(byte[] b1, int s1, int l1,
                     byte[] b2, int s2, int l2) {
    int n1 = WritableComparator.readInt (b1,s1);
    int n2 = WritableComparator.readInt(b2,s2);
    return (n1 < n2)  ? -1 : (n1 == n2) ? 0 : 1;
  }
}
```

Profiling

- Set `mapred.task.profile` to **`true`**
- Profile a small range of maps/reduces
  - `mapred.task.profile.{maps|reduces}`
- hprof support is built-in
- Use `mapred.task.profile.params` to set options for the debugger
- Possibly `DistributedCache` for the profiler's agent

Debugging – Oh no!

- Advanced technology
  - `stderr` – Hold on! Where do we find it?



Job job_200810142005_0045

All Task Attempts

Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions
task_200810142005_0045_m_000000_0	gs201394.inktomisearch.com	SUCCEEDED	100.00%	19-Oct-2008 04:22:22	19-Oct-2008 04:24:01 (1mins, 39sec)		Last 4KB Last 8KB All	9	

Go back to the job
Go back to JobTracker

Hadoop, 2008.

Debugging continued…

- Run job with 'Local Runner'
  - Set `mapred.job.tracker` to **`local`**
  - Runs application in single process/ thread
- Run on a single-node cluster i.e. your dev-box, with sampled data
- Set `keep.failed.task.files` to `true` `and use the IsolationRunner`

Questions?

- For more information:
  - Website: http://hadoop.apache.org/core
  - Mailing lists:
    - core-dev@hadoop.apache.org
    - core-user@hadoop.apache.org
  - IRC: #hadoop on irc.freenode.org