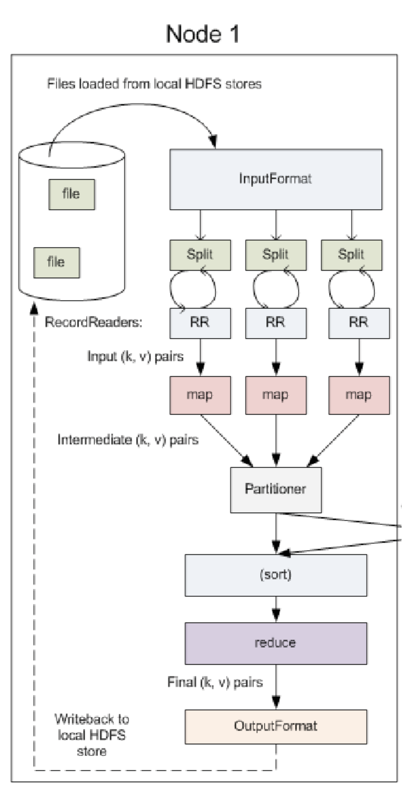
Understanding Filesystem Imbalance in Hadoop

Andrew Ferguson
adf@cs.brown.edu

Rodrigo Fonseca
rfonseca@cs.brown.edu

BROWN

Hadoop Architecture

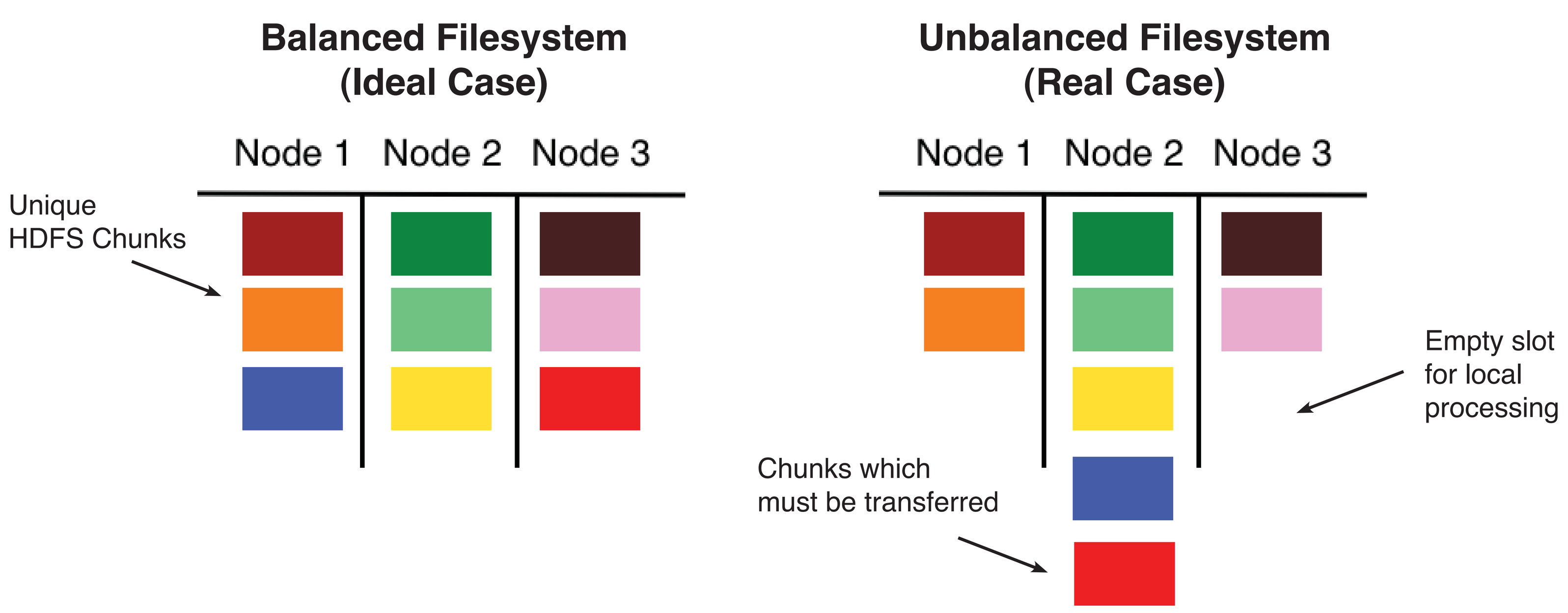


- Users submit **jobs** to Hadoop
- Jobs consist of map and reduce **tasks** executed by TaskTrackers
- Each map task processes one **chunk** from Hadoop distributed filesystem
- Chunk locations are known as the **input split**, which is computed in advance
- Tasks try to read the closest chunk, stored **locally**, **rack-locally**, or **remotely**

Typical Deployment

- **Drives** are placed 2-4 per node
- **Nodes** are organized into racks with full bandwidth
- **Racks** are connected at 1:5 or 1:8 bandwidth

Filesystem Imbalance in HDFS



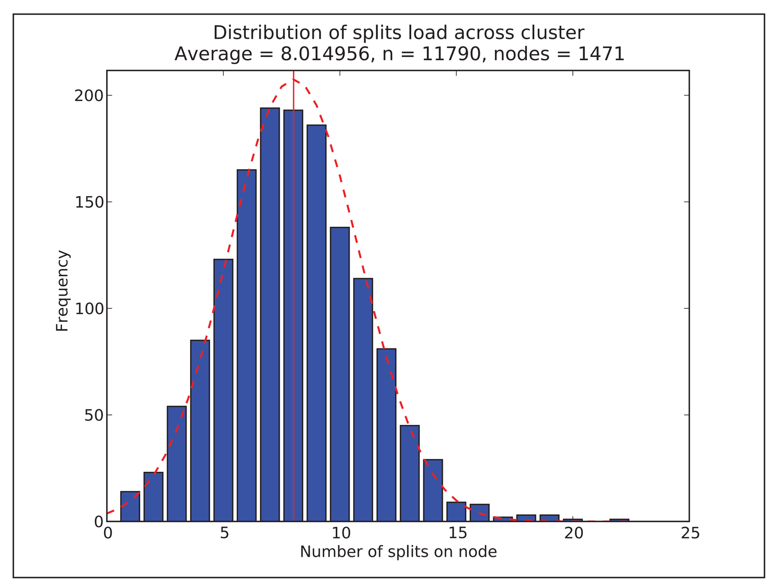
- HDFS places chunks uniformly at random in the cluster
- The number of chunks on each node is the sum of i.i.d. Bernoulli random variables, which is *binomially distributed*
- When a block is not available locally, it must be read over a (relatively) slow network link, and compete for resources

Imbalance in the Real World

- Analyzed 93 jobs from a large company of varying sizes (34 tasks to 11,340 tasks) with a total of 41,377 tasks in total
- 13,299 tasks (32.14%) had input data local to the rack; 2,938 (7.1%) fetched data from another rack; the rest had local data
- This problem is worse for small jobs:

Job Size:	Small	Large
Number of Tasks:	181	2936
Local Tasks:	22 (12.15%)	2099 (71.49%)
Rack-Local Tasks:	111 (61.33%)	700 (23.84%)
Remote Tasks:	48 (26.52%)	137 (4.67%)

- Observed input split distributions match predictions:



Solid red line at mean (8.015).
Dashed red line is normal distribution with $\mu = 8.015$, $\sigma^2 = 7.997$.

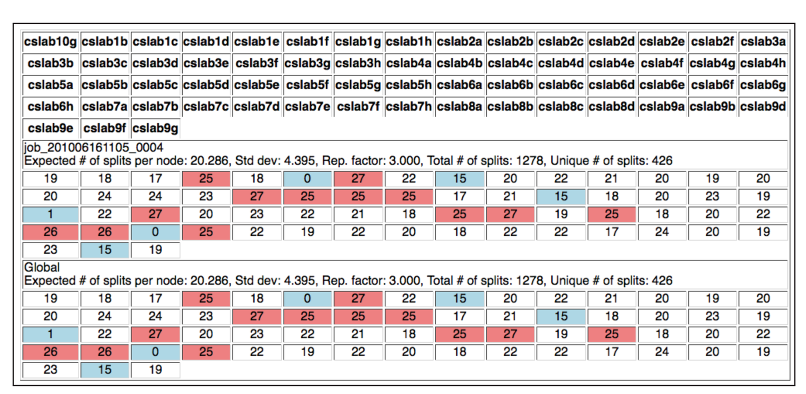
$$\mathrm{E}c_n = S \times \frac{r}{N} = \frac{rS}{N} \qquad c_n \sim \mathcal{N}\left(\frac{rS}{N}, \frac{r(N-r)S}{N^2}\right)$$

r = replication factor of each chunk the DFS
N = number of nodes in the cluster
S = number of unique chunks in the input splits
c_n = number of chunks on node n

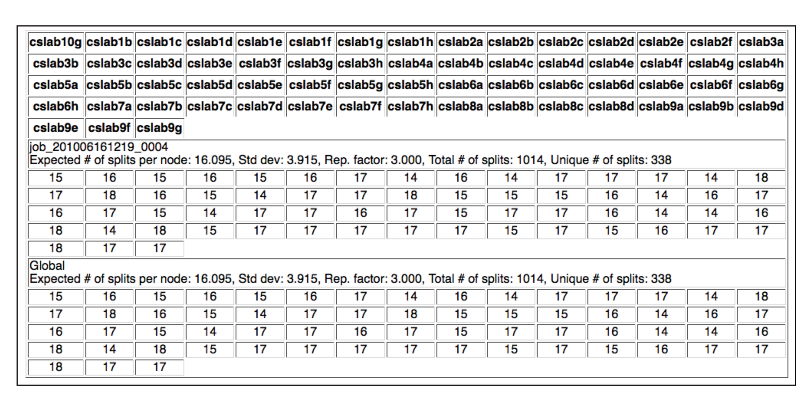
Hypothesis:
Round-robin placement will decrease the variance of the splits distribution and yield improved performance.

Visualizing HDFS Chunk Placement

- The Hadoop JobTracker was modified to display in real-time the number of potentially local tasks remaining on each node, both for each job and globally across all jobs
- White cells are within one standard deviation of the average, while red nodes are one s.d. above, and blue one s.d. below



Standard Placement

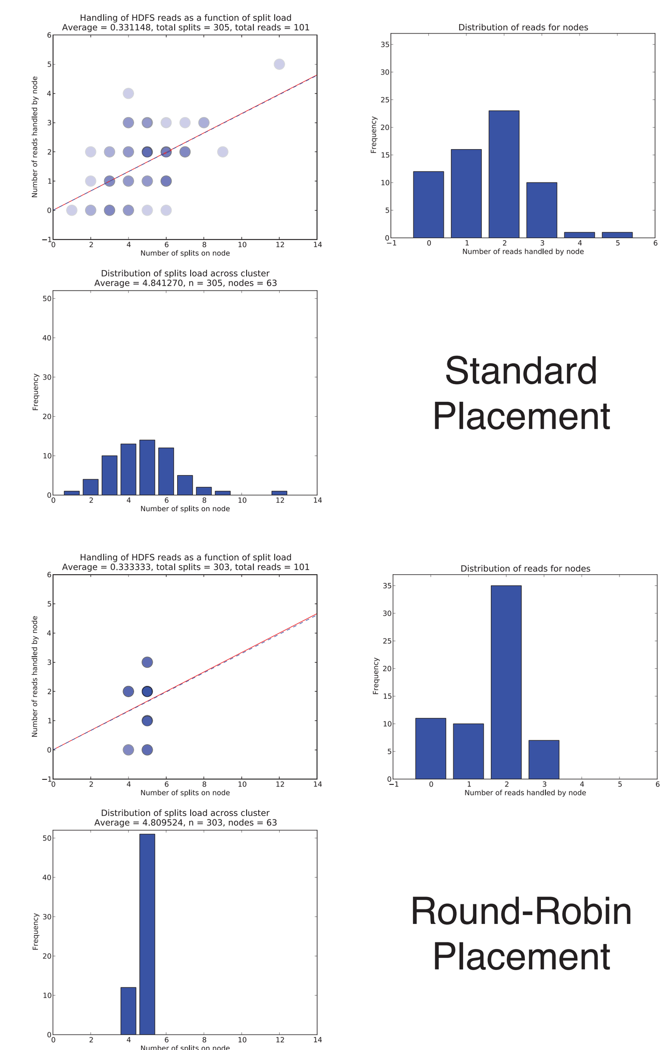


Round-Robin Placement

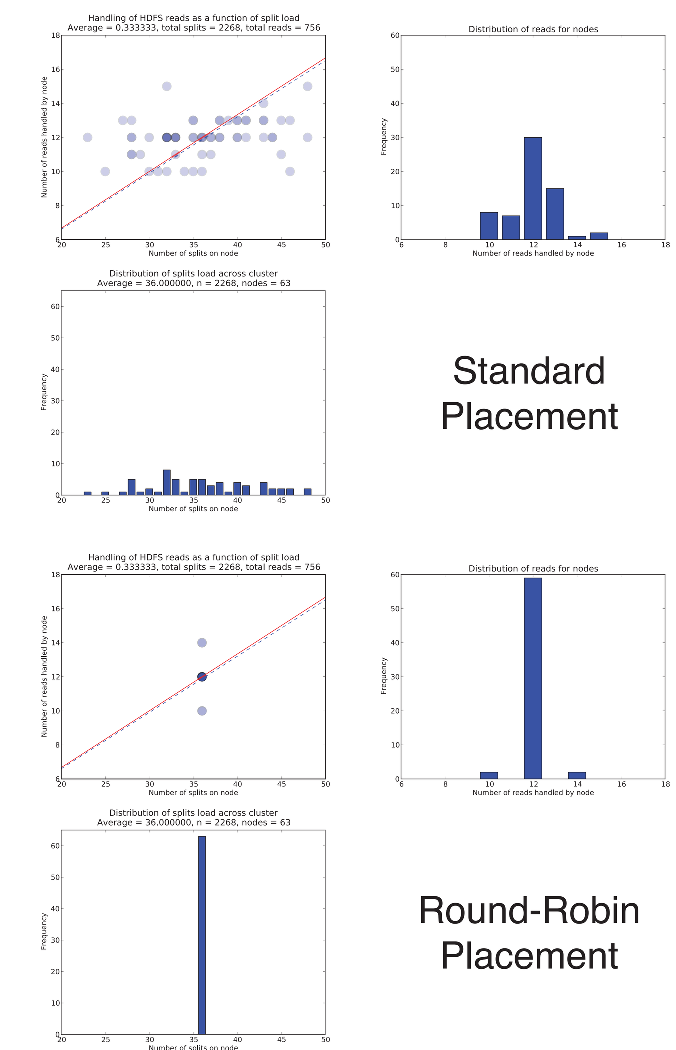
Evaluating Round-Robin Placement

- Hadoop ver. 0.20.1 was augmented with round-robin placement in addition to uniform-at-random (standard placement)
- Experiments were run on a cluster of 63 nodes (21 per rack) with two additional nodes as masters. Nodes had 4 x 2.4 GHz CPUs, 3 GB of RAM, and gigabit Ethernet connections

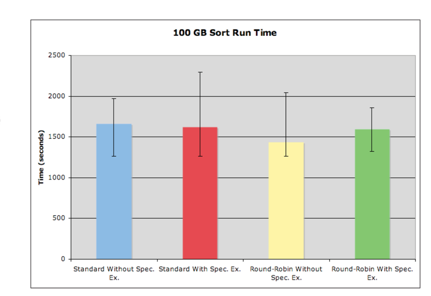
10 GB Grep Benchmark
(I/O Intensive)



Pi-Estimating Benchmark
(CPU Intensive)



- 100 GB Sort Benchmark achieves an 11.5% speed-up using round-robin placement



	Average Run Time	Maximum Run Time	Minimum Run Time
Std without Spec. Execution	1656.9	1969	1263
Std. with Spec. Execution	1618.8	2293	1265
R-R without Spec. Execution	**1433.2**	2042	1260
R-R with Spec. Execution	1595.4	1861	1325

Ten runs were conducted for each configuration

Conclusions and Next Steps

- The performance improvements from round-robin placement illustrate the benefits of a more-balanced filesystem.
- In future work, we will examine whether round-robin-like block placement can improve the performance of the new “delay-scheduling” technique, and construct a characterization of the theoretically best read pattern for a given input split.