6.14 [15/15/10/10] <6.2, 6.3> MapReduce enables large amounts of parallelism by having data-independent tasks run on multiple nodes, often using commodity hardware; however, there are limits to the level of parallelism. For example, for redundancy MapReduce will write data blocks to multiple nodes, consuming disk and, potentially, network bandwidth. Assume a total dataset size of 300 GB, a network bandwidth of 1 Gb/s, a 10 s/GB map rate, and a 20 s/GB reduce rate. Also assume that 30% of the data must be read from remote nodes, and each output file is written to two other nodes for redundancy. Use Figure 6.6 for all other parameters.

a. [15] <6.2, 6.3> Assume that all nodes are in the same rack. What is the expected runtime with 5 nodes? 10 nodes? 100 nodes? 1000 nodes? Discuss the bottlenecks at each node size.

b. [15] <6.2, 6.3> Assume that there are 40 nodes per rack and that any remote read/write has an equal chance of going to any node. What is the expected runtime at 100 nodes? 1000 nodes?

c. [10] <6.2, 6.3> An important consideration is minimizing data movement as much as possible. Given the significant slowdown of going from local to rack to array accesses, software must be strongly optimized to maximize locality. Assume that there are 40 nodes per rack, and 1000 nodes are used in the MapReduce job. What is the runtime if remote accesses are within the same rack 20% of the time? 50% of the time? 80% of the time?

d. [10] <6.2, 6.3> Given the simple MapReduce program in Section 6.2, discuss some possible optimizations to maximize the locality of the workload.

a. 300/5 = 60 GB → data per node for 5 nodes

60 * .30 = 18 GB → read from remote (data per node)

18 / 100 = 180 seconds → remote access time (bandwidth (MB/s))

60 - 18 / 200 (MB/s) = 42/200 = 210 seconds → local disk access

60 * 10 s/GB = 600 seconds → map time

60 * 20 s/GB = 1200 seconds → reduce

(180*2) + (210*2) + 600 + 1200

360 + 420 + 600 + 1200

= 2,580 seconds total

the primary bottleneck is in the reduce stage with the next being in the data transfer stage.

300/10 = 30 GB → data per node for 10 nodes

30 (data per node) * .30 = 9 GB → read from remote

9 / 100 (bandwidth (MB/s)) = 90 seconds → remote access time

30 - 9 / 200 (MB/s) = 21 / 200 = 105 seconds → local disk access

30 * 10 s/GB = 300 seconds → map time

30 * 20 s/GB = 600 seconds → reduce

90 + 105 + 300 + 600 = 1,095 seconds total

the primary bottleneck is in the reduce stage with the next being in the data transfer stage.

300/100 = 3 GB → data per node for 100 nodes

3 (data per node) * .30 = .9 GB → read from remote

.9 / 100 (bandwidth (MB/s)) = 9 seconds → remote access time

3 - .9 / 200 (MB/s) = 2.1 / 200 = 10.5 seconds → local disk access

3 * 10 s/GB = 30 seconds → map time

3 * 20 s/GB = 60 seconds → reduce

9 + 10.5 + 30 + 60 = 109.5 seconds total

the primary bottleneck is in the reduce stage with the next being in the data transfer stage.

$300/1000 = 0.3$ GB → data per node for 1000 nodes

0.3 (data per node) $* .30 = .09$ GB → read from remote

$.09/100$ (bandwidth MB/s) $= .9$ seconds → remote access time

$0.3 - .09 / 200$ MB/s $= .21 / 200 = 1.05$ seconds → local disk access

$.3 * 10$ s/GB $= 3$ seconds → map time

$.3 * 20$ s/GB $= 6$ seconds → reduce

$.9 + 1.05 + 3 + 6 = 10.95$ seconds total

the primary bottleneck is STILL in the reduce stage with the next being in the data transfer stage.

b.

100 nodes

$100/40 = 2.5$ racks

$300/100 = 3$ GB → data per node

3 GB $* 0.3 = 900$ MB

↳ 900 MB $* 2/3 = 600$ MB → read

$600/10$ (bandwidth MB/s) $= 60$ seconds → remote access

$300/100 = 3$ seconds → local

$60 + 30 + (60 * 2) + (3 * 2) = 216$ seconds

the greatest bottleneck is the data transfer.

1000 nodes

1000 / 40 = 25 racks

300 / 1000 = .3GB → data per node

.3GB * 0.3 = 90 MB

↳ 90 MB * 2/3 = 60 MB → read

60 / 10 (bandwidth MB/s) = 6 seconds → remote access

30 / 100 = 0.3 seconds → local

6 + 3 + (6 * 2) + (.3 * 2) = 21.6 seconds

the greatest bottleneck is the data transfer.

C.

20% of the time → 900 * 20% = 180 same rack, 720 remote

720 / 10 MB/s = 72 seconds → remote access

180 / 100 MB/s = 1.8 seconds → local

[(72 + 1.8) * 2] + 30 + 60 = 237.6 seconds total

50% of the time → 450 same, 450 remote

450 / 10 MB/s = 45 seconds → remote access

450 / 100 MB/s = 4.5 seconds → local

[(45 + 4.5) * 2] + 30 + 60 = 189 seconds total

80% of the time → 720 same, 180 remote

180 / 10 MB/s = 18 seconds → remote access

720 / 100 MB/s = 7.2 seconds → local

$[(18 + 7.2) * 2] + 30 + 60 = 140.9$ seconds total

d. Some possible optimizations that would maximize the locality of the workload would be decreasing duplicate words between the map phase and reduce phase which would allow more data to properly fit in node caches overall improving efficiency. The system could also calculate the number of bytes in the map phase as well as properly placing data for reduction which minimizes data transfer within racks.