Does RDMA-based Enhanced Hadoop MapReduce Need a New Performance Model?
The Ohio State University, {rahmanmd,luxi,islamn,panda}@cse.ohio-state.edu

Abstract
Recent studies [17, 12] show that leveraging benefits of high performance interconnects like InfiniBand, MapReduce performance in terms of job execution time can be greatly enhanced by using additional features like in-memory merge, pipelined merge and reduce, and prefetching and caching of map outputs. In this paper, we validate that it is time to have a new performance model for the RDMA-based design of MapReduce over high performance interconnects. Our initial results derived from the proposed analytical model matches the experimental results within a 3-5% range.

1 Motivation
Authors in [17, 12] present enhanced designs and algorithms for the RDMA-based MapReduce framework. With these design changes, MapReduce job execution can be greatly accelerated by leveraging the benefits of high-performance interconnects. The high performance design of Hadoop (Hadoop-RDMA) [3] also shows significant performance benefits achievable through RDMA-capable interconnects using enhanced designs of various components (HDFS [6], MapReduce [12], RPC [9]) inside Hadoop. On the other hand, much performance modeling research [4, 8, 2, 1, 13, 5, 7, 10, 11] has been carried out to deeply analyze the default MapReduce framework. But, because of the inherent architectural changes, these models are not appropriate for performance prediction of RDMA-based enhanced MapReduce. For example, Table 1 captures the performance evaluation for the Sort benchmark using default Hadoop [16] and enhanced MapReduce with RDMA [12] and compares these with the performance model in [4]. This clearly illustrates the necessity of a new model for the enhanced design of MapReduce.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort (20GB)</td>
<td>707 sec</td>
<td>691.78 sec</td>
<td>324 sec</td>
</tr>
</tbody>
</table>

Table 1: Comparison using Sort

2 Our Approach
For the RDMA-based enhanced design of MapReduce, all of the new features are added inside the ReduceTask. Thus, to predict the performance correctly for this design, we approach to model the performance of the ReduceTask from scratch. In the default MapReduce framework, execution time for a single ReduceTask, \( t_{RT} \) is calculated from the execution times of different phases in the ReduceTask.

\[
I_{RT} = I_{shuffle} + I_{merge} + I_{reduce} 
\]

For the RDMA-based design, on the other hand, \( I_{RT} \), will not be as simple as the default one. Because of the fully overlapping feature among these three phases, \( I_{RT} \) can be rewritten as:

\[
I_{RT} = \max(I_{shuffle}, I_{merge}) + \alpha \cdot I_{reduce} 
\]

\( \alpha \) represents the fraction of the total data that resides in memory yet to be reduced, while both shuffle and merge phases have completed their execution. Also, because of the architectural changes in the enhanced design, all of the parameters \( I_{shuffle}, I_{merge}, \) and \( I_{reduce} \) need to be re-modeled to incorporate all of the new design enhancements.

3 Contribution

We validate our model for enhanced MapReduce using terasort [15] on Stampede [14]. We vary the cluster size from 8 to 128, while increasing the data size exponentially from 40 GB to 640 GB. As shown in Figure 1, we observe that the model successfully validates the experimental results with a difference of 3-5% range.
References


