Hadoop is an open source software framework that uses the well-known MapReduce model to process large-scale datasets. It’s widely used by many data processing companies including Google, Yahoo, Facebook, and LinkedIn. Most of these have dedicated Hadoop clusters, which have abundant memory to achieve high system throughput. However, many smaller companies, research institutes, and universities might only have access to high-performance computing (HPC) or ordinary commodity clusters, which are both memory-constrained compared to Hadoop.

The latest survey conducted by the International Data Corporation indicates that 67 percent of HPC systems are now used for big data analysis. It’s unclear whether the MapReduce model can reach its full potential in these constrained platforms. If it can’t, how might we reengineer the traditional Hadoop system toward this purpose?

In the forthcoming article “Mammoth: Gearing Hadoop towards Memory-Intensive MapReduce Applications” (IEEE Trans. on Parallel and Distributed Systems; doi: 10.1109/TPDS.2014.2345068), the authors conducted benchmarking experiments with Hadoop and observed inefficiencies in both memory usage and I/O operations. These deficiencies cause significant performance reduction in Hadoop, especially when the supporting platform’s memory is constrained.

The authors observed static and coarse-grained memory management inefficiencies in Map and Reduce tasks; unnecessary disk spilling during the Map/Reduce procedure; lack of coordination among the Map tasks with different memory demands; excessive I/O waits caused by the merge-sort procedure; excessive disk seeks caused by the parallel I/O; and the long-tail effect caused by an inappropriate priority setting for the file buffer. To tackle these problems, the authors developed a new MapReduce data processing system called Mammoth for memory-constrained systems.

Mammoth is a multithread execution engine that’s based on Hadoop but runs in a single Java virtual machine (JVM) on each node. Each Map or Reduce task on a node is executed as a thread in the engine, and all task threads can share memory at runtime. A memory-scheduling algorithm is developed in the execution engine to realize global memory management.

The authors further implemented the techniques of disk access serialization, multicaching, and shuffling from memory, and solved the problem of full garbage collection in the JVM. In addition, the authors designed a novel rule-based heuristic to prioritize memory allocation and revocation among execution units (mapper, shuffler, reducer, and so on), which maximizes the holistic benefits of the MapReduce job when scheduling each memory unit.

The authors conducted extensive experiments to compare Mammoth with Hadoop and another popular in-memory processing framework called Spark. The results show that Mammoth can dramatically improve performance in terms of execution time on memory-constrained clusters.

Hai Jin is a Cheung Kung Scholars Chair professor of computer science and engineering at Huazhong University of Science and Technology (HUST) and dean of the School of Computer Science and Technology at HUST. Contact him at hjin@hust.edu.cn.