HAMR Technical Report: Graph

PageRank

HAMR vs Hadoop and Spark

Version 0.3
Nov 10, 2014

hamr-info@hamrtech.com
www.hamrtech.com
302.797.4267

Key Findings: *HAMR™ outperforms both Hadoop and Spark by up to 16x.*
HAMR is 10x more memory efficient than Spark.

HAMR is significantly more efficient than MapReduce in both memory and processor utilization. Memory efficiency allows far less spilling to disk; direct processing is *one* of the sources of greater processor utilization. These efficiencies translate into speedups of up to 16x versus Hadoop and 7x versus Spark. HAMR is capable of processing up to 10x larger datasets in memory than Spark.

HAMR is able to achieve these improvements with a simple API that does not require the user to write a more complex program.

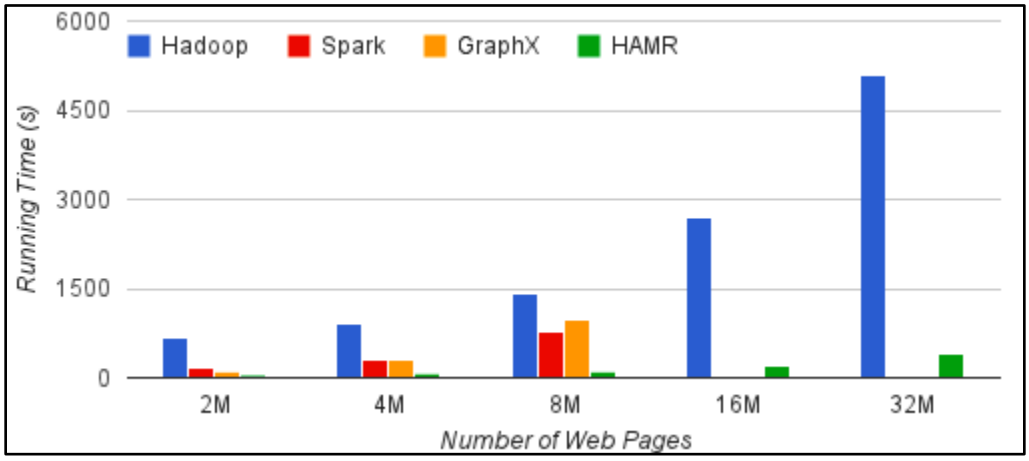
Introduction

The PageRank algorithm, named after Larry Page, one of the founders of Google, works by counting the number and quality of links to a web page to determine a rough estimate of how important the page is. The underlying assumption is that more important websites are likely to receive more links from other websites. The input data is automatically generated Web data whose hyperlinks follow the Zipfian distribution. Intel HiBench provides the Hadoop version of the PageRank benchmark and its input data generator, while Spark includes two versions of PageRank in its distribution, one in Java and another in Scala using the GraphX API. In this document the performance of the PageRank benchmark is compared between Apache Hadoop, Apache Spark, and HAMR.

Results

The following graph presents a performance comparison between the HAMR, Hadoop, Spark, and GraphX[1] PageRank implementations.

PageRank: Hadoop, Spark, GraphX, and HAMR

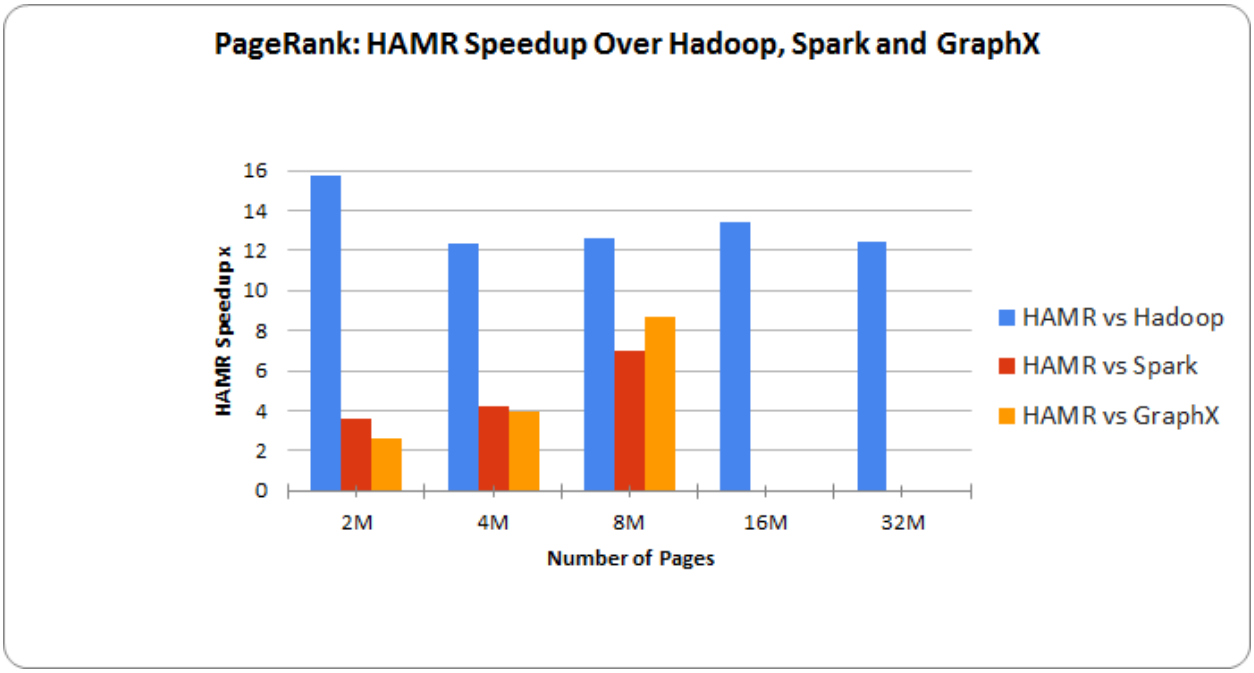


For data sets from 1M to 32M pages HAMR clearly outperforms Hadoop and Spark. At 8 million pages Spark begins to suffer from garbage collector pauses causing the running time to disproportionately increase. At 16 million pages Spark was unable to reliably complete the execution, frequently failing with

[1] GraphX is a Spark graph library.

OutOfMemoryError exceptions, other odd exceptions such as IndexOutOfBoundsException, and finally failing after an hour of retries. In an attempt to continue the benchmarks we modified the Spark configuration and Java implementation of PageRank to be more memory efficient by enabling Kryo serialization and allowing RDDs to spill to disk. This allowed Spark to execute the 16 million page data set in 23 minutes compared to HAMR's 3 minutes. Modifying the RDD StorageLevel did not seem to alleviate the memory usage problems and we were unable to get a successful execution of Spark on the 32 million dataset.

The following diagram and table summarize HAMR's speedup[2] over Hadoop, Spark, and GraphX. HAMR is 16x faster than Hadoop on the 2 million page dataset.



	Speedup over Hadoop	Speedup over Spark	Speedup over GraphX
Minimum	12.37	3.61	2.60
Maximum	15.72	7.00	8.69
Average	13.29	4.95	5.08

HAMR Support for PageRank

PageRank demonstrates the usage of several different KeyValueStores that HAMR implements, and requires complex interaction with them. A KeyValueStore is a node in the workflow graph that is a reliable, distributed data structure for storing key/value pairs. It has ports for connecting to Flowlets and Resources.

In the initialization phase of the algorithm the input file is read from HDFS, the Graph KeyValueStore is built, and the Ranks KeyValueStore is initialized. The initialization workflow is visualized in Figure 1:

[2] (Hadoop | Spark | GraphX running time) / (HAMR running time)

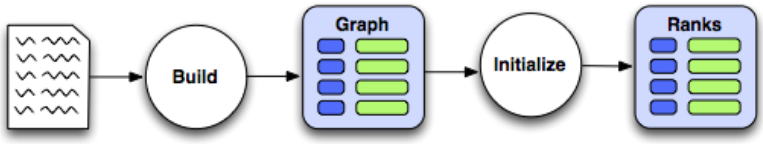


Figure 1: Abstract Representation of HAMR PageRank Initialization

Next the iterative part of the algorithm is executed. For each iteration the rank contribution for each edge is summed per page. Once all pages are accumulated the iteration updates the Ranks KeyValueStore and tests for convergence. In order to maintain compatibility with the Spark implementation, the iteration workflow is executed a fixed number of times. The iteration workflow is visualized in Figure 2:

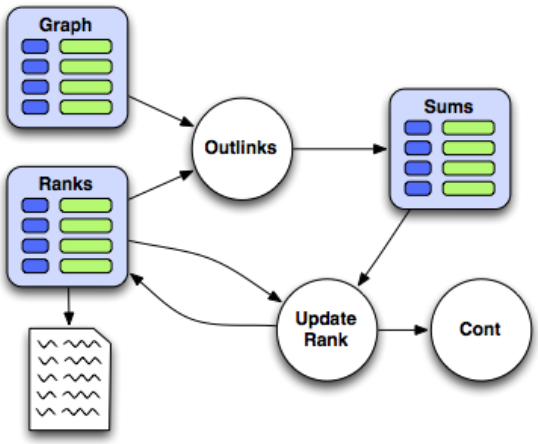


Figure 2: Abstract Representation of HAMR PageRank Processing

Conclusion

HAMR is significantly more efficient than Hadoop and Spark in both memory and processor utilization. This efficiency translates into performance speedups of up to 16x versus Hadoop and 7x versus Spark. HAMR is capable of executing data sets up to 10x larger than Spark in memory, and can execute even larger datasets without modification to the source code. HAMR is able to achieve these improvements with a simple API that does not require the user to write a more complex program. The HAMR implementation of PageRank is approximately the same number of lines of source code as the Spark implementation. This demonstrates that the ease of application development is roughly comparable between the two systems.

Testbed Description

We compare our HAMR implementation of the PageRank benchmark to the Hadoop implementation provided in the Intel HiBench Benchmarking Suite running Hadoop 2.2.0, and to the Spark and GraphX implementations provided with the Spark 1.0.1 distribution.

We report execution time in seconds collected as an average of three runs, each with 5 iterations, for graphs ranging from 2 million to 32 million pages.

HAMR, Hadoop, and Spark read the input graph from the HDFS file system. HAMR and Hadoop write the results to HDFS while Spark's examples write the results to stdout, which may impact the performance comparison. We configure Hadoop and HDFS according to the convention set by Hortonworks (http://goo.gl/4XcCbY). The HDFS block replication is set to 3 with a block size of 128 MB. While these parameters are tweakable for a given benchmark, we attempt to use a configuration most representative of standard clusters.

Spark was executed using YARN with 4 executors, 4 GB of driver memory, 16 GB of executor memory, and 16 executor cores. This configuration provided optimal performance across all runs.

The testing environment is a four node cluster with the specifications listed below. The benchmark was executed on the same cluster, with the same number of compute nodes, and with the same input data for Hadoop, Spark, Spark with GraphX, and HAMR.

- System: 4 Node Dell PowerEdge R420 2 x Xeon® E5-2450 32 GB 1600 MHz DDR3, 4 x SATA 2 (3 GB/s)
- Network: 4 x QDR Infiniband
- OS: CentOS release 6.5
- Java: JDK 1.7.0 u55

HAMR™ is the next generation in-memory real-time software appliance enabling out of the box Big Data analytics. While maintaining the best aspects of MapReduce found in Apache Hadoop 2.0, HAMRTech has developed a novel Big Data programming model and runtime inspired by MIT dataflow. HAMR leverages innovative Flowlets™ and Key/Value Stores to reduce idle time and costly spilling of data to disk, saving both time and energy. Furthermore, HAMR provides access to all enterprise data, and supports batch, streaming, and real-time analytics, making HAMR a fast, efficient, and more complete Big Data solution.