International Journal of Database Theory and Application
Vol.8, No.3 (2015), pp.77-84
http://dx.doi.org/10.14257/ijdta.2015.8.3.08

Performance Comparison Between Hama and Hadoop

Shuo Li and Baomin Xu

School of Computer and Information Technology, Beijing Jiaotong University
Beijing 100044
2889130209@qq.com

Abstract

Massive scientific computations such as matrix, graph and network algorithms are very attractive when they come to modelling real-world data. Apache Hama is a pure BSP (Bulk Synchronous Parallel) distributed computing framework for massive scientific computations. In this paper, our experiments were conducted on a 4-node Hadoop cluster. We implement Monte Carlo algorithm of Pi in Hama and Hadoop under the same software and hardware environment. The experimental results show that Hama can achieve much higher performance than Hadoop in our testbed.

Keywords: *Hama; Hadoop; BSP; matrix computation; performance.*

1. Introduction

The growing demand for large-scale data processing has led to create a new parallel computing model MapReduce [7] for data intensive computing. MapReduce is a successful paradigm to perform large-scale data processing. However, MapReduce model does not efficiently support iterative computations [11]. Large scale scientific computations such as most machine learning and data mining applications involve iterative computations are often used as primary means for many data-intensive scientific applications. The computation core of many data-intensive scientific applications can be expressed as matrix/graph computations based on iterative aggregation-disaggregation methods. Matrix/graph computation has been a focus area in the HPC (High-performance computing) community for many years. With the prevalence of MapReduce, a cloud-based parallel computing framework [1,15], the studies on the high-performance massive scientific computations using cloud have been conducted [2,3,8,9,10,13].

Hama [2] is a distributed framework on Hadoop [4], the open source implementation of MapReduce, for high-dimensional matrix and graph computations. Like Pregel [3], Hama is heavily based on BSP model and also has some similarity to Hadoop. The key difference between Hadoop and Hama is BSP tasks can communicate to each other while MapReduce tasks cannot communicate with each other. Hama aims at a powerful tool for various scientific applications and providing the BSP primitives for developers, so messaging between tasks and a synchronization barrier. Owing to the Hama is a BSP [5] implementation over Hadoop, the BSP nodes and jobs management, including fault tolerance, and node-to-node communication directly leverage the facilities offered by Hadoop. The synchronization barrier is enforced using Zookeeper[6]. The architecture of Hama is illustrated in Fig.1. Hama supports MapReduce engine, BSP engine, and Dryad[12] engine. The MapReduce engine is used for matrix computations while BSP and Dryad engines are used for graph computations.

The performance of distributed computing frameworks impacts many important applications. In this paper, we choose matrix computations as the target for evaluation the performance of Hama. To evaluate the efficiency and accuracy of iterative algorithm on Hama, the selected algorithm needs to have a fixed execution results and suitable for parallelization. So, we select the Monte Carlo calculation of Pi [14] as experimental

ISSN: 2005-4270 IJDTA

algorithm. Specifically, we share our experience of implementing Monte Carlo Calculation of Pi with Hama and Hadoop, and present our preliminary results.

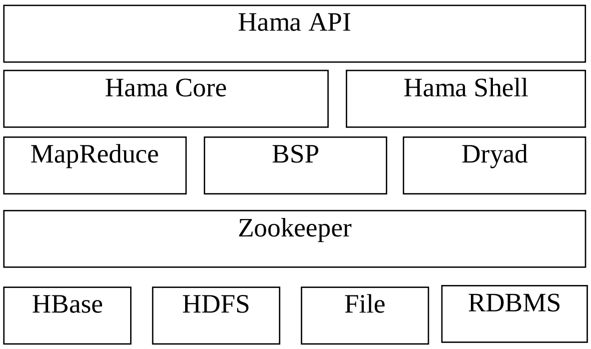

Figure 1. The Architecture of Hama

The remaining of this paper is organized as follows: Section 2 gives a detailed description of the Monte Carlo Calculation of Pi. Section 3 describes the experimental results. Section 4 draws the conclusions.

2. Monte Carlo Calculation of Pi

The Monte Carlo method provides an approximate solution to a variety of mathematical problems by performing statistical sampling experiments on a computer. The calculation of Pi using the Monte Carlo method can be described as:

If a circle of radius r is inscribed inside a square with side length 2r, then the area of the circle will be πr^2 and the area of the square will be $(2r)^2$ .The ratio of the area of the circle to the area of the square is $\pi / 4$. This means that, if you randomly pick points $\{(x_i, y_i)\}_{i=1}^{n}$ inside the square. With approximately $\pi / 4$, those points should fall inside the circle, because the circle has $\pi / 4$ the area of the square. So, we can estimate π as:

$$\pi \approx 4\frac{m}{n}$$

Where n is the number of points inside the square. m is the number of points that satisfy $x_i^2 + y_i^2 \leq 1$ in circle.

To calculate each significant digit there will have to be about 10 times as the number of iterations to calculate the preceding significant digit.

3. Experimental Results

The evaluation of Hama has been performed on 4-nodes Hadoop Cluster. Each node consists of a quad core processor Intel Xeon X3470 and 16GB of main memory. The operation system is Redhat Enterprise Linux 5. Hadoop-0.20.2 and Hama-0.4.0-incubating have been installed on the cluster. In addition, the number of both Hadoop's map and Hama's BSPPeer are fixed at 20.

In this evaluation, we used both Hama and Hadoop to implement Monte Carlo calculation of Pi. Through many experiments, we found that the time growth trends can be divided into three stages. Correspondingly, the number of iterations is divided into three intervals: 1) advantage interval 0~10^5. Hama gives better performance than Hadoop; 2) available interval 10^5~1.5*10^7. The performance of Hama is gradually reduced as the problem size becomes larger; 3) disadvantage interval 1.5*10^7~10^8. The performance of Hadoop is better than that of Hama.

3.1 Advantage Interval

The experimental results of both Hama and Hadoop algorithm are shown in Table 1 and Table 2, respectively, and graphed in Fig.2.

Table 1. Advantage Interval – Hama

number of iterations	Hama						mean value
0	elapsed time (seconds)	3.381	3.352	3.363	3.369	3.361	3.3652
	value of Pi	N/A	N/A	N/A	N/A	N/A	N/A
10	elapsed time (seconds)	3.364	3.368	3.358	3.359	3.367	3.3632
	value of Pi	3.2600	3.3400	3.2200	3.2600	3.0200	3.2398
10^2	elapsed time (seconds)	12.202	12.171	12.175	12.184	12.171	3.3612
	value of Pi	3.17	3.188	3.13	3.132	3.186	3.1612
10^3	elapsed time (seconds)	12.181	12.178	12.178	12.179	12.172	3.3664
	value of Pi	3.1389	3.1419	3.1461	3.1449	3.1457	3.1435
10^4	elapsed time (seconds)	12.180	12.182	12.183	12.170	12.171	3.364
	value of Pi	3.1441	3.1387	3.1444	3.1403	3.1422	3.1413
10^5	elapsed time (seconds)	12.180	12.178	12.174	12.177	12.168	3.3578
	value of Pi	3.1423	3.1404	3.1417	3.1429	3.1413	3.1417

Table 2. Advantage Interval – Hadoop

number of iterations	Hadoop						mean value
0	elapsed time (seconds)	64.275	64.776	64.368	64.705	64.243	64.4734
	value of Pi	N/A	N/A	N/A	N/A	N/A	N/A
10	elapsed time (seconds)	64.26	64.233	64.265	64.173	64.254	64.237
	value of Pi	3.27	3.35	3.58	3.24	3.17	3.322
10^2	elapsed time (seconds)	64.103	64.259	64.233	64.285	64.216	64.2192
	value of Pi	3.179	3.183	3.129	3.134	3.168	3.1586
10^3	elapsed time (seconds)	64.253	64.205	64.221	64.237	64.126	64.2084
	value of Pi	3.1428	3.1436	3.1457	3.1439	3.1412	3.14344
10^4	elapsed time (seconds)	64.233	64.233	64.233	64.233	64.233	64.233
	value of Pi	3.1414	3.1409	3.1415	3.1425	3.1456	3.14238
10^5	elapsed time (seconds)	64.24	64.213	64.384	64.241	64.354	64.2864
	value of Pi	3.1414	3.1417	3.1415	3.1412	3.146	3.14236

As seen in Table 1 and Table 2, For Hama, the execution time is maintained at between 3.1 to 3.4 seconds. For Hadoop, the execution time is maintained at between 64.2 to 64.5 seconds. There is no apparent difference in total execution time between Hama and Hadoop algorithm. It can be inferred that the execution time of Hama is mainly consumed in the BSP function while the execution time of Hadoop is mainly consumed in the MapReduce functions when the number of iterations below 10^5.

As seen from Fig.2, The computing efficiency of Hama is about 95 percent higher than that of Hadoop. This phenomenon is caused by the following reasons: Hama does not have a large amount of data read and write operations while Hadoop presences of a large number of I/O operations.

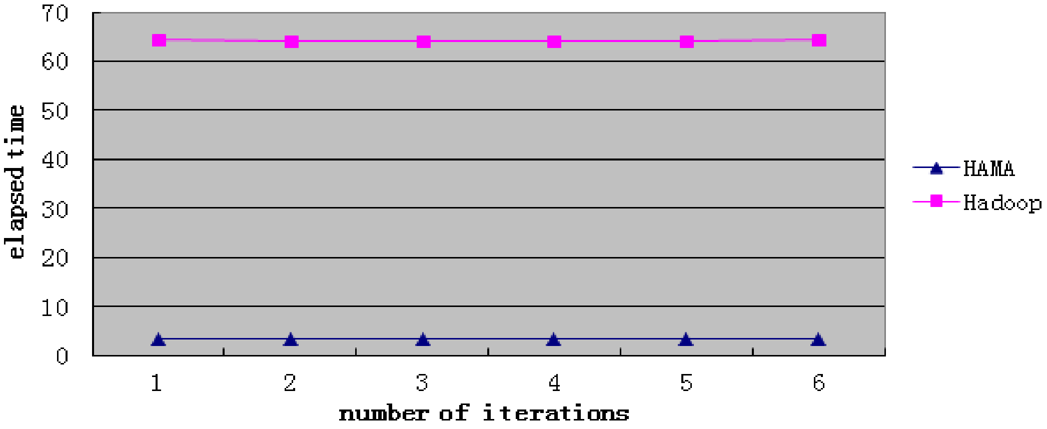

Figure 2. Advantage Interval

3.2 Available Interval

Table 3. Available Interval –Hama

number of iterations	Hama						mean value
10^6	elapsed time (seconds)	6.364	6.37	6.362	6.374	6.361	6.3662
	value of Pi	3.1419	3.1413	3.1417	3.1417	3.142	3.14172
2X 10^6	elapsed time (seconds)	9.366	9.35	9.402	9.397	9.373	9.3776
	value of Pi	3.1417	3.1417	3.1417	3.1416	3.1418	3.1417
4X 10^6	elapsed time (seconds)	18.307	18.384	18.322	18.308	18.36	18.3362
	value of Pi	3.1416	3.1417	3.1417	3.1412	3.1413	3.1415
6X 10^6	elapsed time (seconds)	27.333	27.286	27.377	27.429	27.418	27.3686
	value of Pi	3.1415	3.1416	3.1415	3.1415	3.1414	3.1415
8X 10^6	elapsed time (seconds)	36.33	37.091	36.357	36.435	36.658	36.5742
	value of Pi	3.1416	3.1415	3.1414	3.1418	3.1414	3.14154
10^7	elapsed time (seconds)	45.376	44.992	45.651	45.241	42.576	44.7672
	value of Pi	3.1416	3.1416	3.1417	3.1414	3.1415	3.14156
1.5X 10^7	elapsed time (seconds)	66.346	66.417	67.753	66.195	66.628	66.6678
	value of Pi	3.1415	3.1415	3.1411	3.1416	3.1419	3.14152

Table 4. Available Interval – Hadoop

number of iterations	Hadoop						mean value
10^6	elapsed time (seconds)	64.242	64.254	64.252	64.256	64.352	64.242
	value of Pi	3.1415	3.1409	3.1412	3.1415	3.1416	3.14134
2X 10^6	elapsed time (seconds)	64.247	64.223	64.209	64.28	64.219	64.247
	value of Pi	3.1415	3.1419	3.1414	3.1419	3.1415	3.14164
4X 10^6	elapsed time (seconds)	64.236	64.295	64.516	64.228	64.299	64.236
	value of Pi	3.1413	3.1419	3.1414	3.1415	3.1415	3.14152
6X 10^6	elapsed time (seconds)	64.241	64.224	64.242	64.296	64.341	64.241
	value of Pi	3.1415	3.1416	3.1418	3.1419	3.1414	3.14164
8X 10^6	elapsed time (seconds)	64.449	64.489	64.475	64.639	64.039	64.449
	value of Pi	3.1415	3.1412	3.1419	3.1419	3.1415	3.1416
10^7	elapsed time (seconds)	64.377	64.392	64.757	64.354	64.257	64.4274
	value of Pi	3.1416	3.1416	3.1416	3.1417	3.1416	3.14162
1.5X 10^7	elapsed time (seconds)	64.383	64.801	64.203	64.326	64.923	64.5272
	value of Pi	3.1415	3.1416	3.1416	3.1417	3.1413	3.14154

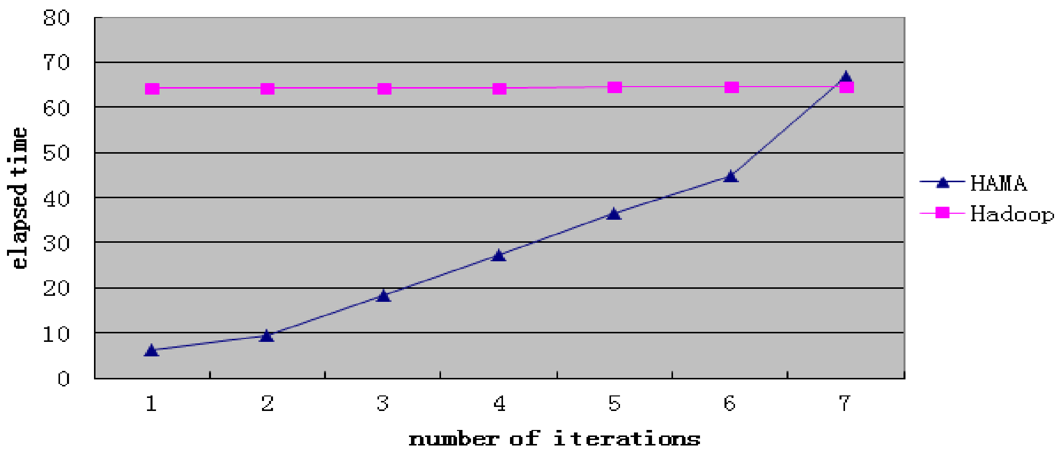

Figure 3. Available Interval

As seen in Table 3 and Table 4, For Hama, there is significant fluctuation in execution time. For Hadoop, the execution time to complete computing tasks is maintained at between 64.2 to 64.5 seconds.

As seen from Fig.3, the mainly execution time of Hama is iterative calculation while the execution time of BSP function is not increased significantly. The total execution time of Hadoop is mainly consumed in MapReduce functions when the number of iterations below 1.5×10^7.

This is because Hama optimization operation focuses on BSP functions while there has no specialized optimization for the calculation process itself in Hama.

3.3 Disadvantage Interval

The experimental results are shown in Table 5 and Table 6, and graphed in Fig.4. Fig.4 shows significant performance degradation of the Hama implementation compared to Hadoop. This indicates that computing task itself consumes the main part of execution time when computation over a certain size.

The computing efficiency of Hadoop is about 35 percent higher than that of Hama. This phenomenon is caused by the following reasons: Intermediate results occupy a lot of hardware resources when the amount of calculation is increased. So, the hardware resources allocated to the MapReduce tasks and BSP tasks are substantially reduced. In the case, it leads to decline in operating efficiency for Hama and Hadoop.

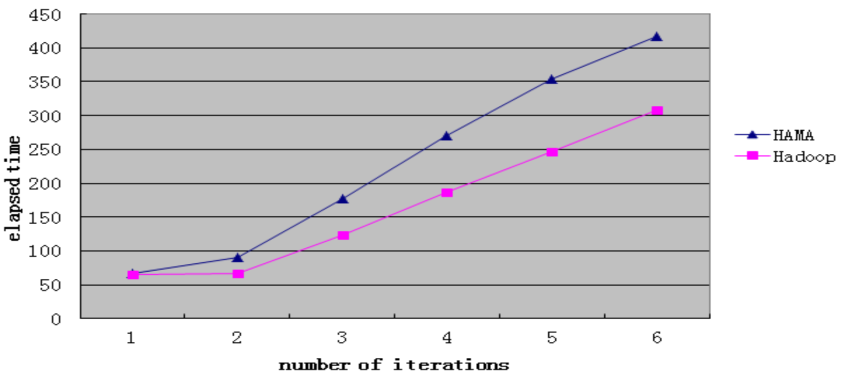

Figure 4. Disadvantage Interval

Table 5. Disadvantage Interval –Hama

number of iterations	Hama						mean value
1.5X 10^7	elapsed time (seconds)	66.346	66.417	67.753	66.195	66.628	66.6678
	value of Pi	3.1415	3.1415	3.1411	3.1416	3.1419	3.14152
2X 10^7	elapsed time (seconds)	89.435	89.761	89.348	91.825	89.419	89.9576
	value of Pi	3.1417	3.1414	3.1412	3.1415	3.1415	3.14146
4X 10^7	elapsed time (seconds)	178.013	175.905	177.346	179.697	178.314	177.855
	value of Pi	3.1415	3.1416	3.1415	3.1415	3.1413	3.14148
6X 10^7	elapsed time (seconds)	269.085	273.68	266.347	271.583	266.807	269.5004
	value of Pi	3.1416	3.1414	3.1413	3.1418	3.1415	3.14152
8X 10^7	elapsed time (seconds)	356.368	348.584	355.018	360.139	352.634	354.5486
	value of Pi	3.1415	3.1415	3.1415	3.1417	3.141	3.14144
10^8	elapsed time (seconds)	415.18	422.671	413.336	416.892	417.826	417.181
	value of Pi	3.1415	3.1415	3.1415	3.1415	3.1418	3.14156

Table 6. Disadvantage Interval –Hadoop

number of iterations	Hadoop						mean value
1.5X 10^7	elapsed time (seconds)	64.383	64.801	64.203	64.326	64.923	64.5272
	value of Pi	3.1415	3.1416	3.1416	3.1417	3.1413	3.14154
2X 10^7	elapsed time (seconds)	67.419	67.094	67.849	67.419	67.443	67.4448
	value of Pi	3.1415	3.1417	3.1416	3.1416	3.1415	3.14158
4X 10^7	elapsed time (seconds)	124.465	121.925	123.474	120.655	127.432	123.5902
	value of Pi	3.1417	3.1415	3.1418	3.1413	3.1416	3.14158
6X 10^7	elapsed time (seconds)	184.548	187.842	182.788	185.606	189.517	186.0602
	value of Pi	3.1416	3.1415	3.1416	3.1415	3.1412	3.14148
8X 10^7	elapsed time (seconds)	244.618	254.173	241.623	246.868	248.92	247.2404
	value of Pi	3.1415	3.1415	3.1415	3.1414	3.1415	3.14148
10^8	elapsed time (seconds)	304.674	306.866	305.379	315.241	307.539	307.9398
	value of Pi	3.1415	3.1415	3.1415	3.1416	3.1415	3.14152

4. Conclusions

In this paper, we have evaluated the performance of Hama with Monte Carlo calculation of Pi. The results show that Hama shows higher computing performance than the Hadoop implementation when the number of iterations is less than a certain value n. The value of n depends on the experimental conditions. Otherwise, Hama has worse performance than Hadoop. We hope that the experimental results presented in this paper would be useful for the future development of Hama.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC 61370060).

References

[1] B. Xu, C. Zhao, E. Hu, and B. Hu, "Job Scheduling Algorithm Based on Berger Model in Cloud environment", Advances in Engineering Software, vol. 42, no. 7, **(2011)**,pp. 419-425.

[2] E.J. Yoon, J. Kim, S. Jin, J.-S. Kim, S. Maeng, "Hama: An Efficient Matrix Computation with the MapReduce Framework. 2010 IEEE Second International Conference on Cloud Computing Technology and Science", **(2010)**, pp. 721-726.

[3] G. Malewicz, M. H. Austern, A.J.C Bik, J. C. Dehnert, I. Horn, N. Leiser, G. Czajkowski, "Pregel: a system for large-scale graph processing", ACM SIGMOD International Conference on Management of data, **(2010)**, pp.135-146.

[4] "Apache Hadoop" en.wikipedia.org/wiki/Apache_Hadoop

[5] "Bulk Synchronous Parallel", http://en.wikipedia.org/wiki/Bulk_synchronous_parallel

[6] P. Hunt, M. Konar, F. P. Junqueira, B. Reed, "ZooKeeper: Wait-free Coordination for Internet-scale Systems", USENIX Annual Technology Conference, **(2010)**, pp.11-11.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004: 137-150.

[8] U. Kang, C. E. Tsourakakis, C. Faloutsos, "PEGASUS: Mining Peta-Scale Graphs", Knowledge and

Information Systems, vol. 27, no. 2, **(2011)**, pp.303-325.

[9] "JPregel", url: http://kowshik.github.com/JPregel/.

[10] "Apache Giraph", url: http : / /giraph .apache.org/

[11] E. Elnikety, T. Elsayed, H. E. Ramadan, "iHadoop: Asynchronous Iterations for MapReduce", IEEE Third International Conference on Cloud Computing Technology and Science, Nov. 29 -Dec. 1 2011:81- 90.

[12] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", The 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, **(2007)**, pp. 59-72.

[13] "Twister", url: http://www.iterativeMapReduce.org/#home.

[14] http://math.fullerton.edu/mathews/n2003/montecarloPimod.html.

[15] B. Xu, N. Wang, C. Li, "A cloud computing infrastructure on heterogeneous computing resources", Journal of Computers, **(2011)**, vol. 6, no. 8, pp. 1789-1796.

Authors

Shuo Li, he is an undergraduate student of Computer and Information Technology, Beijing Jiaotong University, China. He prefer to get a higher degree in the field of Cloud Computing, Big Data. He makes efforts to gain more attention of this field.

Baomin XU, Dr. Xu Baomin is an associate professor in School of Computer and Information Technology, Beijing Jiaotong University, China. His research interests include Large Scale Data Analysis Based-on Cloud Computing, and Complex Network (Link Prediction, Community Detection).Until now he has published more than 50 research papers in referred journals and proceedings.