HPL Performance Prevision to Intending System Improvement[1]

Zhang Wenli[1, 2], Chen Mingyu[1], and Fan Jianping[1]

[1] National Research Center for Intelligent Computing Systems,
Institute of Computing Technology, Chinese Academy of Sciences
[2] Graduate School of the Chinese Academy of Sciences,
NCIC, P.O. Box 2704, Beijing, P.R. China, 100080
zhangwl@ncic.ac.cn

Abstract. HPL is a parallel Linpack benchmark package widely adopted in massive cluster system performance test. On HPL data layout among processors, a law to determine block size NB theoretically, which breaks through dependence on trial-and-error experiments, is found based on in-depth analysis of blocked parallel solution algorithm of linear algebra equations and implementation mechanics in HPL. According to that law, an emulation model to roughly estimate HPL execution time is constructed. Verified by real system, the model is used to do some scientific prevision on the benefits to Linpack test brought by intending system improvement, such as respectively memory size increase, communication bandwidth increase and so on. It is expected to conduce to direct system improvement on optimizing HPL test in the future.

1 Introduction

Linpack[1] is a prevailing performance test benchmark at present. HPL[2] (high performance Linpack) is the first open standard parallel Linpack test package on large-scale distributed-memory parallel computing, used in Top500[3] test widely. In order to obtain optimal results, users can use any number of CPUs to any problem size and use various kinds of optimization methods based on Gaussian Elimination.

Performance test is actually to calculate the floating-point operation rate— Gflops. In LU factorization, it is

$$(2n^3/3 + 3n^2/2) / t. \tag{1}$$

Generally, to obtain HPL peak value, the problem size should be as large as close to 80% of total memory capacity. NB is influential to HPL test time, but till now, its determination mainly depends on experience, which brings about the deficiency of reliability emulation model of performance test.

[1] This research is supported by Chinese National High-tech Research and Development (863) Program (grants 2003AA 1Z2070) and by the foundation of Knowledge Innovation Program (grants 20036040), Chinese Academy of Sciences (CAS).

Therefore section 2 of this paper will probe the basis for determining NB theoretically, and try to construct an emulation model of HPL test. In section 3 the emulation model is verified further. Section 4 forecasts the Linpack test prevision with system improvement using the verified model. Finally section 5 concludes.

2 Model Introduction

The main problem of HPL is to solve dense linear algebra equations

$$Ax = b. \tag{2}$$

Clearly, LU factorization[4, 5, 6, 7] is the main part of linear equations solution, accounting for O (n^3) operations. The two-dimensional block-cyclic data distribution of dense matrix among processors is confirmed after series of analysis and comparison[8]. But the efficient determination of NB is still hanging.

2.1 Theoretical Determination of Block Size NB

In experiments, we found that the efficiency factor γ, defined as ratio of test time and the amount of operation, varies little with N gemination increase, but does distinct change with NB to a great extent. After our inference verified, it seems proper to choose suitable NB in quality referring to tendency of small-scale matrix efficiency curve. It will undoubtedly reduce the blindness of NB determination, as well as benefit emulation model construction. Detail description is in Reference [9].

2.2 Emulation by Constructing Model

Since computing is the main part in single processor, it is convinced to construct an emulation model for cluster based on computing and communication parts. By $T_{comm} = \alpha + \beta L$, communication time can be defined. Where α = Communication latency, β = 1/bandwidth, L = Communication package size. By efficiency factor γ, computing time can be emulated utilizing operation amount. Then according to analyzed HPL panel execution flow and execution logic, detailed in reference [9], an estimating emulation model was implemented. Using the law in section 2.1 proper NB is chosen, then emulation model was activated after acquiring the parameters correlative to architecture, such as communication latency, bandwidth and so on.

3 Model Verification

To emulation model, verification is done on an AMD64 node and part of Dawning 4000A, which listed No. 10 in the newest Top500 lists with 11.264 Tflops theoretical peak performance, respectively described in Table 1. Clearly, estimated results shown in Table 2 are really close to real ones, and obviously better than the rough estimate time, denoted as D in table, described in reference [1].

Table 1. Summary of tested architectures

Arch.	**Proc. no.**	**Freq.** (Ghz)	**Peak Perf.** (Gflops)	**Bandwidth** (Gb/s)	**Latency** (ns)	**Mem. Size per node** (GB)
AMD64 single Node（Ⅰ）	1×2	1.6	3.2	0.664	25000	2
Dawning 4000A Nodes（Ⅱ）	16×4	2.4	4.8	2.5	5000	8

Table 2. Comparison of real system record time and estimated one

Tested Arch.	**Matrix Dim.** N	**Proc. Array** P×Q	**Real Time** (s)	**Estimate Time** (s)	**Diff.** (%)	**D Time** (s)	**D Diff.** (%)
I	8000	2×1	70.04	70.25	0.2998	71.88	2.6271
	14140	1×2	354.88	354.13	-0.2113	356.49	0.4537
II	57780	2×2	8196.63	8159.42	-0.4540	8123.27	-0.8950
	115760	2×8	16625.72	16552.34	-0.4414	16309.29	-1.9032

Note: The arch. I and II are corresponding to the ones in Table 2.

The above verification further strengthens the credibility of emulation model.

4 Prevision of Potential Test Performance

Referring to system status of Dawning 4000A, rough forecast is attempted on potential performance to be brought by intending system improvement by the above model. It can be beneficial to leverage efforts on architecture optimization.

4.1 When Memory Size Changes by Gemination

The adjustment of memory size is simulated by matrix dimension N. Estimated results in Table 3 indicate that, with memory size increasing by gemination, the increase scale of system efficiency decreases and the difference is only several permillage. Although reference [10] arguments that there is no limitation for the factorization of huge matrix, due to the time completing once row copy of 9000 elements is 1.8 milliseconds, it seems that reducing matrix size to store in row and column simultaneously is more of feasibility than enlarging the matrix size auxiliary by hard disk prefetching. Moreover, in large scale, the difference between Dongarra estimated time and the model estimated is not up to 3%, which assures the model estimation believable to some extent.

Table 3. To estimate performance improving scale with simulated memory change

Mem. Size (G)	**Matrix Dim.** N	**Estimate Time** (s)	**D Time** (s)	**FP Op Rate** (Gflops)	**Gflops Diff** (Gflops)	**Effi.** (%)	**Incr. Scale** (%)
2560	509120	12609.8	12256.2	6976.8		61.9391	
5120	720000	35164.0	34476.2	7076.3	99.4847	62.8223	0.8832
10240	1018230	98483.1	97134.4	7146.3	70.0316	63.4441	0.6217
20480	1440000	276643.6	273984.8	7195.7	49.3921	63.8826	0.4385

4.2 When Bandwidth Increases by Ten Times

Based on estimated results shown in Table 4, to gigabit per second bandwidth level, it will be only 0.8% performance improvement to HPL with ten times increase of bandwidth. It is obviously inferior to the former ten times improvement to Gb/s acquiring 8.6% increase. The developing potential distinctly diminishes.

Table 4. To estimate performance improving scale with bandwidth change

Bandwidth (Gb/s)	**Estimate Time** (s)	**FP Op Rate** (Gflops)	**Efficiency** (%)	**Incr. Scale** (%)
0.25	40768.19	6103.6	54.1864	
2.5	35163.99	7076.3	62.8223	8.6359
25	34714.57	7167.9	63.6356	0.8133

4.3 When Latency Decreases by Thousand Times

Clearly, results in Table 5 show that in large scale once the latency reduces to nanosecond level its decrease is of little real meaning to performance improvement.

Table 5. To estimate performance improving scale with latency change

Latency	**Estimate Time** (s)	**D Time** (s)	**Time Diff.** (%)
5ms	53756.93	53050.33	1.3144
5us	35182.56	34494.77	1.9549
5ns	35163.99	34476.21	1.9559

4.4 When Efficiency Factor Varies

The estimated results in Table 6 indicate that, varying +/- 0.1 from current γ value 0.35 causes system floating point operation rate to change by a large scale, and the increased scale increases with further improvement. Unfortunately, γ is acting by complex factors, related to main frequency and number of FPU etc. Limited by its complexity, further insulated analysis on correlated factors for γ should be done.

Table 6. To estimate performance improving scale with efficiency factor change

Effi. factor	**Estimate Time** (s)	**FP Op Rate** (Gflops)	**Incr. Scale** (%)
0.45	45438.57	5476.2	
0.35	35441.02	7021.0	28.2090
0.25	25443.48	9779.8	39.2931

5 Conclusions

Theoretical analysis and actual system verification show that, it is feasible and credible to use the characteristic of efficiency factor to determine NB and further construct emulation model to estimate HPL execution time. The estimated results demonstrate that to expectative system improvements, such as memory size, communication bandwidth etc., HPL performance is not improved as obviously as that brought by little change of γ. It further indicates that updating computation accounts for absolute portion of Linpack test, whose little change will cause obvious performance increase. Yet due to acting by complex factors, work here corresponding to γ is to be done further. The model is to be amended to do more precise forecast.

References

1. Jack J. Dongarra, Piotr Luszczek and Antoine Petitet. The LINPACK Benchmark: Past, Present, and Future, Concurrency and Computation: Practice and Experience 15, 2003
2. A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary. HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, http://www.netlib.org/benchmark/hpl/
3. Hans W. Meuer, Erik Strohmaier, Jack J. Dongarra and H.D. Simon. Top500 Supercomputer Sites, 17th edition, November 2 2001. (The report can be downloaded from http://www.netlib.org/benchmark/top500.html)
4. Zhang BL, etc. Theory and method of numeric parallel computing, Beijing: National defense industry press, 1999,7
5. Lin CS. Numeric computing method (Column A), Beijing: Science press, 1998
6. Chen GL. Parallel computing: structure, algorithm, programming (modified version), Beijing: Advanced education press, 2003.8

7. Sun ZZ. Numeric analysis (second edition), Nanjing: South-east university press, 2002.1
8. http://www.cs.utk.edu/~dongarra/WEB-PAGES/SPRING-2000/lect08.pdf
9. Zhang Wenli, Fan Jianping, Chen Mingyu. Efficient Determination of Block Size NB for Parallel Linpack Test. Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), MIT, Received.
10. Eddy Caron, Gil Utard. On the performance of parallel factorization of out-of-core matrices, parallel computing, 30 (2004) 357-375