Linpack

Gao Cuiying Chen JinYu Xiao Jianwei Fang zhu Wu Weizheng

Abstract

Linpack is now internationally the most popular benchmark for testing floating-point performance of high-performance computer systems. This article introduces Linpack's installation environment, installation process, and HPL.data parameters. This paper also introduces the Linpack test based on standalone test, which includes the influence of some important factors such as different problem sizes, number of nodes, matrix data block NB, processor network topology $P \times Q$ and so on. Draw the conclusion:

1 Introduction

Linpack (Linear system package)[1] is a tool for testing the performance of floating-point computing in high-performance computer systems, but also the performance of the international TOP500[2] supercomputer evaluation tools. The test uses Gaussian elimination method to solve the dense one-dimensional N-order linear equations. In the calculation, the coefficient matrix is processed by the block-recursive LU decomposition method, the original linear equations are converted into the upper triangular equations, and the solutions to the system of equations are obtained one by one by replacing them. The computational part of the method is mainly done by calling the functions of the Basic Linear Algebra Subprograms (BLAS). The BLAS library mainly includes three kinds of Goto BLAS, ATLAS and Generic BLAS. Goto BLAS Library takes into account the idea of modern processors' Cache and Translation Lookaside Buffers (TLBs) that use block-wise computation and block-wise storage while being optimized for different processors that enable Goto BLAS Library in the test has been widely used. OpenBLAS is an optimized BLAS library based on GotoBLAS2 1.13 BSD version. In this paper, OpenBLAS library as a method of implementation. This report uses the Lenovo ideapad 3150 to test the results for different problem scales, different block sizes for the same size matrix, and different processor topologies.

2 Installation of Linpack

Before installing HPL, the system must already have the compiler, parallel environment MPI, and either one of Basic Linear Algebra Subprograms (BLAS) or Vector Signal and Image Processing Library (VSIPL) installed. The compiler must support C and Fortran 77 languages. Parallel environment MPI generally use MPICH, of course, can also be other versions of MPI, such as LAM-MPI. HPL runs require BLAS or VSIPL libraries, and library performance is closely related to the final measured Linpack performance. Commonly used BLAS library GOTO, Atlas, ACML, ESSL, MKL and so on.

Download $hpl-2.1.zip$ package and unzip, copy $Make.<Arch>$ file to the hpl root directory $hpl-2.1$, according to their own file path configuration. Make file, and then compile, execute the compile command in the $/hpl$

directory. After doing this, there will be two files in directory */hpl/bin/Arch*. They are HPL.dat and xhpl. The latter is an executable file, and the former is a common file which is used to control the parament of testing Linpack.

3 Introduce of Paraments in HPL.dat

1. Lines 1 and 2 are comment lines and do not need to be changed
2. Lines 3 and 4 specify the format of the result file. When "device out" is "6", the test result is output to a standard output (stdout). When "device out" is "7", the test result is output to standard error output (stderr). When "device out" is other values, the test result is output to the file specified in the third line.
3. Lines 5 and 6 illustrate the number and size of matrix N. The larger the size of the matrix N, the greater the proportion of valid calculations,the higher the performance of system floating-point processing,But the increase of matrix size N will lead to the increase of memory consumption. Once the actual system memory space is insufficient, Using caching, performance can be drastically reduced.It is best that matrix occupies about 80 percent of the total memory of the system .
4. Lines 7 and 8 illustrate the size of the solution matrix N. To improve the locality of data, improve overall performance, HPL uses a partitioned matrix algorithm.
5. Line 9 is an introduction to the selection of processor arrays that are listed in columns or in rows.
6. Lines 10-12 illustrate the two-dimensional processor grid ($P \times Q$) and the two-dimensional processor grid($P \times Q$). There are several requirements, $P \times Q$ = cores number = number of processes. In general, A process that corresponds to a CPU can get the best performance.
7. Line 13 shows the accuracy of the test.
8. Lines 14-21 specify the matrix L decomposition method. In the elimination process, Each time zHPL completes the elimination of the NB column, and then update the following matrix. The elimination of this NB is the decomposition of L.

4 Result and analysis

- Information of our platform

Processor	Intel(R) Core(TM) i5-7200 CPU @ 2.5GHz 2.7GHz
Cache Size	3072kB
Memory	8GB
MPI	MPICH
Blas	Openblas-0.2.20
Operating System	Linux Ubuntu 16.10
Compiler	Fortran77Fortran90
HPL	HPL2-1

All of our tests are based on a laptop whose information is shown above. Before testing, we calculate the theoretical floating-point peak /cite[]. It is about 80Gflops. But by now, we only get a result about 58Gflops. So there is still much to improve.

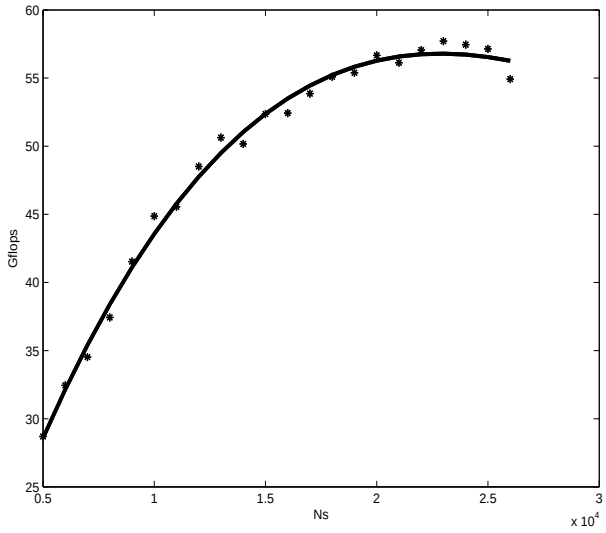

Figure 1: Influence of NS on test results

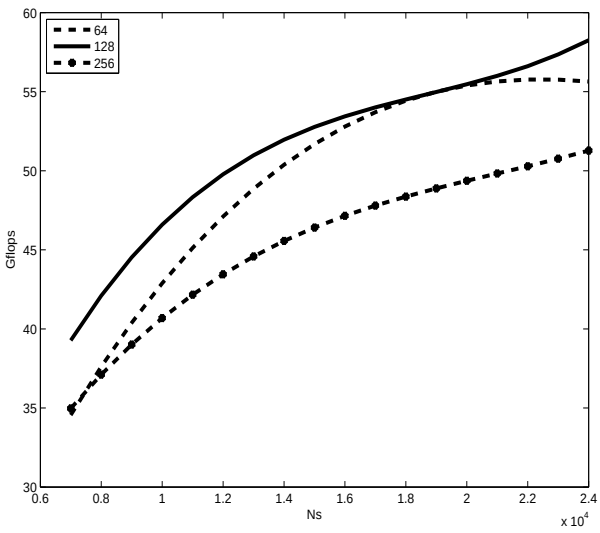

Figure 2: Effect of block size on test results

From Figure 1, we can see that each time NS increases with a gradient of 1000. Within a certain range, the performance of Linpack shows a significant linear increase. From the figure we can see that calculation of Linpack shows a declining trend after NS exceeds 23000. As the value increases, the memory of the computer continues to be depleted. When the NS value reaches 23,000, the memory of the computer consumes about 84 percent, slightly more than the recommended value of 80 percent [3]. The speed of CPU data processing is slower than that of the memory data supply. So the overall performance showed a downward trend.

The impact of different data allocation and computing granularity on parallel computing is very big. As we can see from Figure 1, the computational performance of different problem scales increases slowly with the increase of NB value. when NB equals to 128, our computer gets the highest ability of floating-point computing. On the one hand, if NB is too small, nodes frequently pass messages, resulting in long communication time and short calculation time. On the other hand, if NB too big. it easily lead to unbalanced load and will increase the cost of communication. so the size of the matrix block. NB if of much importance. In fact, NB can not be too small nor too large.

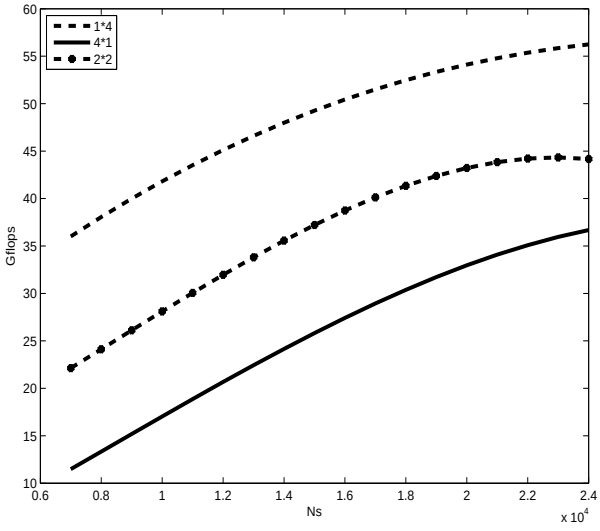

Figure 3: Test results for different processor topologies

Bidimensional progress array has a greater impact on performance. As shown in Figure 3, the calculated performance is optimal at

$P \times Q = 1 \times 4$. The computational performance of different Bidimensional progress arrays increases as the sizes of the problem, and then slows down.

5 Conclusion

References

[1] J. J. Dongarra, "The linpack benchmark: An explanation," in *Supercomputing*, pp. 456–474, Springer, 1988.

[2] Y. Deng, P. Zhang, C. Marques, R. Powell, and L. Zhang, "Analysis of linpack and power efficiencies of the worlds top500 supercomputers," *Parallel Computing*, vol. 39, no. 6, pp. 271–279, 2013.

[3] X. Z. Huang Kai, "Scalable parallel computing: technology, structure and programming," 2010.