文章编号:1007 – 6492(2001)02 – 0007 – 05

The Study of MPP and User 's Effective Speed

WANG Wen – yi[1], XIN Xiao – nan[1], WANG Ruo – yu[2]

(1. College of Electrical & Information Engineering, Zhengzhou University of Technology, Zhengzhou 450002, China; 2. Henan University of Electric Power Workers, Zhengzhou 450001, China)

Abstract: Generally, the Massively Parallel Processor(MPP) has very high peak speed in theory, but the user 's effective speed in use is far from it. In the paper, the author takes Exemplar(SPP 1200) system for example to discuss the methods of increasing MPP user 's effective speed on cache missing ratio, localizability, hierarchical algorithms and so forth. These methods would be of great importance to more and more application of MPP machines.

Key words: cache missing latency; localizability; hierarchical algorithms; visualized tool

CLC munber: TP 311.11 **Document code**: A

Introduction

In recent years, some developed countries in the world are positively investing huge funds in the development and study of the Massively Parallel Processing(MPP) technique, which makes it become one important scale in judging a country 's scientific and technical level and synthetic national power. Because MPP system possesses good scalability , there is almost no limit to its theoretic peak speed. Now, the RISC – based processor 's MPP system is rapidly replacing costly traditional Vector Parallel Processor (VPP), forming the leading trend. For example, Paragon xp/s, SPP1600, T3E, CM – 5, SP – 2 etc., are the new type super parallel computers[1].

However, MPP still has some insufficiency because it is developing and researching stage. Although in some MPP system, such measures have been taken as increasing cache capacity on board and setting the multi – entry and the integer processor to improve pre – fetching function, these measures would be restricted in large – scale science computation by the programs and the algorithms. When the executive unit is to reference the data, and the data are not in cache, thus causing the cache delay. Therefore it makes the user 's effective speed lower obviously.

To VPP machine, whether it is IBM3090 or famous CRAY/YMP and CRAY – C90 etc., they are not much affected by the above – mentioned problem, for the users can easily make the effective speed up to more than 90% of the peak speed, by vector processing.

The Exemplar 's (namely, SPP 1200) manufacturer claims that its peak speed is 240 Mflops per processor (real * 4) or 120 Mflops per processor (real * 8).

I tested the speed on Exemplar (using 8 processors) and on CRAY – C90 (supposing its speed is 1Gflops per processor) respectively by using a large – scale linear problem——the lattice QCD (Quantum Chromodynamic) code. The measured results are given in table 1. The speeds tested by MILC group member of lattice on other parallel superscalar machines with the QCD code are shown in table 2.

From the above – mentioned results, we find that each of the computers displayed the same problem that the user 's effective speed is much less than the peak speed, especially for large – scale sparse problem, it is only about 30% .

Received date: 2000 – 10 – 12; **revision received date**: 2001 – 01 – 20

Biography: WANG Wen – yi(1947 –), male, born in Luogang, Henan province, professor of Zhengzhou University of Technology, research interest: software engineering and parallel algorithm.

Table 1 Lattice QCD code (matrix inversion) CXPA tested results

type	structure	iteration	CPU/s	wall clock/s	effective speed/(Mflops/processor)
CRAY - C90	VPP	109	165.0	186.0	1000
exemplar	MPP	109	5322.8	835.7	31

And I also got the NCSA members basic estimation for Exemplar about the average user's efgective speed in the CONVEX user's world - wide conference at Dallas. The estimation is that the Exemplar applied effective speed is between 25 Mflops per processor to 40 Mflops per processor. For especially large - scale science computing problem, they estimated that the Exemplar user's effective speed could be lower than 20 Mflops per processor.

The above - mentioned measured results and the estimation are consistent with the author's tested data on Exemplar.

Table 2 MIMD QCD code performance in megaflops per node

machine type	nodes	lattice size	Mflops/node
Paragon xp/s	1	8^4	28
	128	$16 * 32^3$	23
	256	32^4	22
CM - 5(conjugate gradient)	128	$24^3 * 12$	40
CM - 5(full code)	128	$24^3 * 12$	23
T3D	64	$16^2 * 32^2$	31
	128	$16 * 32^3$	30
SP - 2 (thin nodes)	1	$12^3 * 6$	36
	8	$12^2 * 24^2$	34
	16	$12 * 24^3$	32
	32	24^4	30

Under normal conditions, the execution cycle of a typical instruction contains four phases, that is, instruction fetch, decode, execute and write back. Under the condition that: all operands can be fetched within cache, we could get the speed as one operation per cycle after 3 cycles. Accordingly, the formula of computing peak performancespeed is

$$V = F * m ,$$

where, F is the time clock frequency and m is the number of the floating function units. Exemplar computer with HP - PA7200 processor has $F = 120$ MHz and $m = 2$, therefore its peak speed is

real * 4 : $120 * 2 = 240$(Mflops/processor);

real * 8 : $120 * 2/2 = 120$(Mflops/processor).

1 Cache Missing Latency

Customarily, we call it a hit that an operand is fetched within cache, otherwise a miss. When cache missing latency happened, the pipeline actions would delay some time from waiting for the data to come. It means that we need more time cycles to perform one instruction. So we can say, cache missing latency is the main cause in leading to user's effective speed slowdown.

The number of time cycles wasted by cache missing mainly depended on the memory bandwidth (of cause, it is also concerned with the programming). The memory bandwidth of RISC - based parallel computer usualmy is lower than that the pipeline data flow requires. And the memory hierarchical technique of parallel computer makes it much lower. For example, Exemplar of uhe peak speed with 120Mflops per processor (cache line capacity is 500 kB) requires such a data flow bandwidth that is

$$120 * 8 = 960 \text{ MB/s} ,$$

but its memory bandwidth at all levels are as follows:

memory - memory 41 MB/s;

cache - memory 49 MB/s;

memory - cache 88 MB/s;

cache - cache 352 MB/s.

Obviously, the difference between these bandwidths and the peak speed required bandwidth 960 MB/s is great. Although Exemplar with HP - PA7200 processor has adopted the 'pre - fetching' function to reduce cache missing latency, above - mentioned problem can't still be avoided, when the user's data set is large, such as in computing the matrix problem of large

lattice size discrete partial differential equations. For MPP system, this is a general characteristic problem. The tests show that MPP system 's Cache Missing Ratio (CMR) is larger than 50% for most large – scale application computation, for some special problems such as scatter or gather (in the linear algorithm routine lib.), CMR even more than 90%.

According to some hardware experts, because of the development of science and technology, the peak speed of general RISC – based superscalar processors will be doubled and their bandwidth will also increase by about 25% to 50% in the next two years. It looks like that the conflict between peak speed and bandwidth will still be increased. That is to say, the difference between the user 's effective speed and the peak speed will be much greater.

In principle, the VPP machine should not be affected by cache missing, because it has so many vector registers and has considerable bandwidth, and CRAY – C90 is a typical example. However, the VPP is limited in the number of processors in system structure, so it is difficult to make VPP 's float point speed reach or exceed 1Tflops. But on the stage, many world – level significant challenging subjects urgently require this property.

Because MPP system has the advantages of scalability, good cost performance and very high peak speed etc., now, many famous institutes, national laboratories and supercomputing centers in the world turn to it, which causes VPP machine to go down rapidly in the real market.

To sum up, the users must deal with MPP 's effective speed problem conscientiously.

In the case of cache missing latency dominating, the user 's highest effective speed can be expressed as follows:

$$V(\text{effect}) \leqslant V(\text{peak}) * (1.0 - \text{CMR}),$$

where CMR equals cache missing ratio which is a measurable number and

CMR = (cache missing latency)/(CPU time) (%).

2 Memory Hierarchical Structure and Localizability Skill[2]

The key to raising MPP system user 's effective speed is how to reduce the cache missing latency. The following methods are to be considered by the users.

2.1 Increasing the localizability by going to small sub – data space

In programming, the data arrays (vector, matrix, etc.) to be operated should be split into several pieces as small as possible, making each thread only operate one sub – space. That can help the compiler to locate your data address and improve the pre – fetching function. I tested this strategy by using the lattice QCD code (a matrix – vector multiapplication routine). The measured results is shown in table 3. From the table we can clearly see that the cache latency is affected by data array decomposition.

Particularly, you can make all the operands stay in cache for a long time and get a super – speed up (cache alignment), if you are lucky. I happened to get this case by using a CFD code to solve a 160 * 160 two – dimensional Navier – Stockes equations through simultaneous Over – relaxation (SOR) algorithm. The measured data cache latency is also shown in table 3. The user 's effective speed is 109 Mflops per processor, very close to the exact peak speed 120 Mflops per processor (real * 8). At the same time, I got the wall clock speed up than 8 (using 8 processors).

Table 3　The effect of data decomposition on parallel computer runtime

function	long array/s	CMR/%	sub – array number/s	CMR/%
$V = M * v$	420.0	74.9	273.6(4)	72.6
navier – stockes	46.8	28.8	8.78(8)	9.2

2.2 Appointing memory level in programming, making the operands stay in memory as low level as possible

Artificially appointing memory level will greatly affect the operating speed. The memory of general work stations, such as SUN, HP, IBM RS/6000 etc. has three levels as 0,1 and 2 (if the register on CPU is regarded as the first level), but shared memory MPP usually has five levels as 0,1,2,3 and 4. Exemplar has adopted the memory hierarchical technique, its memory is divided into cache, local memory and shared global memory from a low level to a high level[4]. The higher the memory level is, the lower the transfer bandwidth is. For example, the level of the SPP1200 system 's shared global memory is the highest, but the bandwidth

from it to cache is only 56 kB/s.

Using the lattice QCD code (a matrix – vector multiplication routine), I can also test the influence of appointing or not appointing memory classifying in programming on the effective speed. The measured results are shown in table 4.

Table 4 The effect of appointing memory level in programming on parallel computer runtime

function	calculation time/s		CMR/%	
	not classified	node – private	not classified	node – private
$V = M * v$	348.2	273.6	77.1	72.6

2.3 Using as new algorithms as possible to increase 'cache using ratio'[3]

Recently, the experts of numerical algorithm fields have developed a lot of methods called 'hierarchical algorithms'. And they have widely been applied to newly published mathematical routine library. The basic idea is substituting 'block' method for traditional method, that is, 'less data and more computing'. Consider four kinds of computing models in table 5, we can find that the higher the class is, the shorter the data cache missing latency would be, that is, the higher its CUR is.

Table 5 Classify of various computing functions

class	computing function	CUR = # performance/ # data reference
0	vector(or matrix) assignment	~ 1/2
1	vector – vector $V = B + a * v$	~ 3/3
2	vector – matrix multiplication $V = M * v$	~ 2
3	matrix – matrix multiplication $M = L * R$	~ n

note: ~ 'means approximate value.

I tested the CMR of class 0 and class 2 by using the lattice QCD code (compare a vector assignment routine and a matrix – vector multiplication routine). The results are shown in table 6.

I also tested the following three kinds of computing models with 3000 * 3000 matrix and 3000 vector by designing program. The data cache missing latencies are shown in table 7.

Similarly, the VECLIB (Convex Inc. general routine library about linear algebra algorithms) routines are also tested with 4 processors, and the measured results are shown in table 8. The table shows that the VECLIB user's effective speed upper bound are very close to the exact effective speed, which is mainly because VECLIB's codes have been well optimized by the manu – facturer.

From the data shown in table 6, table 7 and table 8, it is not difficult to find such a fact that the effective speed is very low in executing the vector (or matrix) assignment operation, whether the users use VECLIB codes or self – designed Fortran – 90 programs[5]. The author suggests that the users should avoid using them as much as possible and use other methods.

Table 6 CMR of various computing functions

class	computing function	CMR/%
0	$V = 0$	93.6
2	$V = M * v$	72.6

Table 7 CMR and Mflops of various computing functions(by Convex Inc. general routine library)

class	computing function	CMR/%	upper bound speed/ (Mflops/processor)
1	$V = v + a * v$	80.0	24.0
2	$V = M * v$	68.9	37.3
3	$M = M_1 * M_2$	15.8	101.1

Table 8 CMR and Mflops of various computing functions (by user's testing program)

class	computing function	CMR/%	upper bound speed/ (Mflops/processor)
0	$V(i) = V_1(\mathrm{ind}(i))$	79.9	24.1
1	$V = a * V_1 + v$	72.1	33.5
2	$V = M * V_1$	44.8	66.2
3	$M_1 = M_2 * M_3$	21.8	93.8
1	$V = a * v_1 + v$	74.5	61.2
2	$V = M * V_1$	43.2	136.3
3	$M_1 = M_2 * M_3$	16.7	199.9

note: ahead four is real * 8, behind three is real * 4.

2.4 Fully utilizing the system software tools to improve the application program's quality

MPP system is usually provided with the supporting en-

vironment or the software tools for parallel programming. The users can conveniently use them to get many parameters such as the program parallelism, Mflops, cache missing ratio and the communication or data transfer delay between processors etc. These parameters can help users to know well system running status and to further improve the application program quality. For example, CXPA (Convex X – window based Performance Analysis tool) is a software tool on Exemplar for analyzing computer running status.

CXPA is easy to operate. The users need only to do the following operations:

fc – CXPA ... fortran.f

or cc – CXPA ... cfile.c

type CXPA a.out&

You will go into a X – window interface. Then the visualized graphic would quid the users to fulfilling the measuring of the data cache latency and the information of more variables. After test running, CXPA creates a profile which can be read by the user's clicked 'profile' term. In the profile, you can get the data reports or the graphic reports which reflect the routine effect.

3 Conclusion

This paper takes only Exemplar computer for example to discuss the problem how to increase MPP user's effective speed in practice. We can say that the content is only a side of massively parallel processing computer in applications. The parallel programming and the parallel computer's structure property are closely related to each other. Therefore, its development and efficient realization are much more difficult than those of serial programming. As the MPP's technique develops quickly, how to make its practical property and peak property achieve perfect unanimity has become a realistic problem to be solved urgently. Therefore, it is especially important to train and bring up a large number of qualified personnel who not only know the MPP's system structure but also have high level in parallel programming.

Reference:

[1] BONNIGER T, ESSER R, KREKEL D. A comparative description of massively parallel computers[J]. Parallel Computing, 1995,21(2):199 – 232.

[2] HWANG Kai. Advanced Computer Architecture: Parallelism Scalability Programmability[M]. New York: McGraw – Hill Inc,1993.

[3] HILLS W D, STEELE G L. Data parallel algorithms[J]. Commun ACM,1986,29(12):1170 – 1179.

[4] Richardson T X. Exemplar SPP1200/XA Scalable Computing System[R]. New York: Convex Computer Corporation,1995.118 – 143.

[5] DONG Shao – jing, WANG Wen – yi. A parallelization test on SPP1200 of a large scalar matrix computation[A]. Hewlett Packard Convex Technology Center: Proceeding of the 11th Annual Convex User Group Worldwide Conference[C]. Dallas: Hewlett Packard Convex Technology Center,1996.

MPP 和用户有效速度的研究

王文义[1], 辛小南[1], 王若雨[2]

(1.郑州工业大学电气信息工程学院,河南 郑州 450002;2.河南电力职工大学,河南 郑州 450001)

摘　要:大规模并行处理机(Massively Parallel Processor,MPP)一般都具有极高的理论峰值速度,但用户在实际应用中的有效速度却往往与之大相径庭.以 Exemplar(SPP1200)系统为例,着重在 cache 缺失率、可定域性和层次算法等方面探讨了如何提高 MPP 用户有效速度这一问题,这对我国大规模并行机的推广应用将具有一定的实际意义.

关键词:cache 缺失延迟;可定域性;层次算法;可视化工具