基于 HPL 测试的集群系统性能分析与优化*)

王晓英　都志辉

(清华大学计算机与科学系　高性能计算研究所　北京 100084)

摘　要　集群系统以很高的性价比和良好的可扩展性而成为当今高性能研究的一大新热点，如何评价与优化集群系统的特性也成为一个很关键的问题。本文通过采用全世界 TOP500 计算机排名所用的 HPL 基准测试，对集群的浮点运算性能进行评测和分析，根据在集群系统上进行的 HPL 基准测试的结果分析总结了影响其整体性能的重要因素，并对这些方面一一给出具体的分析，提出了针对集群系统具有实际意义和研究价值的优化方向和优化方法。

关键词　集群系统，系统性能，HPL，基准测试，优化

Performance Analysis and Optimization for Cluster System Based on HPL Benchmark

WANG Xiao-Ying　DU Zhi-Hui

(Department of Computer Science, Tsinghua University, Beijing 100084)

Abstract　Cluster system has become increasingly significant in High Performance Computing(HPC) research field because of its high performance/cost ratio and good scalability. How to evaluate and optimize the performance of cluster system becomes a key issue in this field. The World TOP500 computer rank list uses HPL as a standard benchmark, from which the optimized performance of clusters on floating-point operations can be obtained. Specific tests are carried out for a typical cluster system, and then some most important factors which affect the performance of the whole system are studied. With the analysis of these factors in detail, this article proposed some methods of optimization for cluster systems, which have both practical significance and researching value.

Keywords　Cluster system, System performance, HPL, Benchmark, Optimization

1　引言

将多台同构或异构的计算机节点用高速内部网络连接起来协同完成特定的任务便构成集群系统，集群计算机已经成为当前高性能研究的一项热点。这里所要研究的是高性能计算集群[1](High Performance Computing Cluster)。作为一种新型的并行计算系统，集群系统能够以较低的价格提供很高的计算性能，改变了传统的超级计算的概念，是计算机体系结构上的一个重要突破。

对一个系统，评测其计算性能、计算效率是很重要且有意义的，本文的目的就在于根据测试结果深入分析和研究提高集群在实际应用中效率的可行方法。实验中所采用的集群系统是采用 Symmetric MultiProcessor (SMP)的计算机为节点的结构。SMP 系统一般含有 2～8 个微处理器，采用总线连接，共享内存储器。因此 SMP 集群结合了共享存储和分布存储这两种体系结构，拥有混合的存储模式。

2　Linpack 与 HPL

2.1　概述

Linpack[2]是在高性能计算领域最出名和使用最广泛的基准测试，使用线性代数方程组，利用选主元高斯消去法在分布式内存计算机上按双精度(64 bits)算法，测量求解稠密线性方程组所需的时间。Linpack 的结果按每秒浮点运算次数(flops)表示。

Linpack 发展至今已经有几种版本，其中 High Performance Linpack (HPL)是针对大规模并行计算系统的测试，也已经成为国际标准的 Linpack 基准测试程序，一般用于 TOP500[3]超级计算机上的并行超级计算机。HPL 的特点是自由度要大很多，使用者可以根据需要选择矩阵的规模，分块大小，分解方法等各种参数。HPL 软件包需要在配备了 MPI 环境下的系统中才能运行，还要底层有线性代数子程序包 BLAS[7]或 VSIPL[4]的支持。它不仅提供了完整的 Linpack 测试程序，还进行了全面细致的计时工作，最后可以得到求解的精确性和计算所花费的总时间。

目前，用 Linpack 基准测试出的最高性能指标已经成为衡量机器性能的标准之一，这是一个比较成熟的测试标准[5]。

2.2　HPL 算法简要描述

该软件包是用来求一个 N 维的线性方程组 $Ax=b$ 的解，首先通过选局部列主元的方法对 $N\times(N+1)$ 的 $[A\ b]$ 系数矩阵进行 LU 分解成如下形式：

$[A,b]=[[LU],y]$

由于下三角矩阵 L 因子所作的变换在分解的过程中也逐步应用到 b 上，所以最后方程组的解 x 就可以由转化为求解上三角矩阵 U 作为系数矩阵的线性方程组 $Ux=y$ 从而得到。

为了保证良好的负载平衡和算法的可扩展性，数据是以循环块的方式分布到一个 $P\times Q$ 的由所有进程组成的 2 维网格中。$N\times(N+1)$ 的系数矩阵首先在逻辑上被分成许多 $NB\times NB$ 大小的数据块，然后循环的分配到 $P\times Q$ 进程网格上去处理。这个分配的工作在矩阵的行、列两个方向同时进行。如图 1 所示。图 1(a)为一个矩阵被分成 8×8 个小块，要将它们分配到一个 2×3 的进程阵上，相同颜色的块被分配到同一个进程中去；图 1(b)为分配之后各个数据块在进程中的分布情况。

*)本文受到清华大学 985 基金、华为基金以及 IBM 基金(JC2001027)资助。王晓英　博士生，主要研究方向为并行计算与分布式计算；都志辉　副教授，主要研究方向为集群计算、网络计算以及 P2P 计算。

前面所提到的 N,NB,P,Q 都是可以根据集群的具体配置和用户需要而随时修改的，也是 HPL 测试中十分关键和重要的几个参数。除了这几个主要参数以外，HPL 还提供了很多可改变的参数，在开始测试前通过修改 HPL.dat 文件的内容来设定。

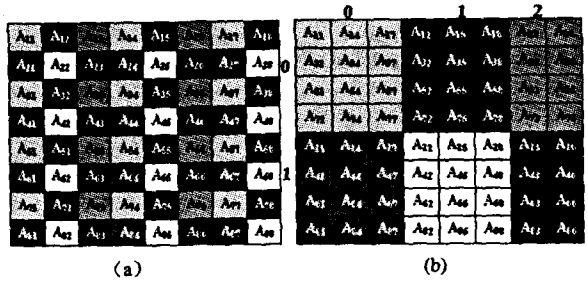


图 1　矩阵分块在各进程上的分布

3　影响性能的因素

下面通过具体的测试来寻找影响集群系统性能的主要因素。实验以“深超-21C”集群系统为测试平台，“深超-21C”是清华大学为深圳大学研制的高性能集群系统，其整体架构如图 2 所示。共有 128 个双 CPU 的 SMP 节点，每个节点物理内存为 1G，前 64 个节点为双 Intel Xeon 3.06G 处理器，后 64 个节点为双 Intel Xeon 2.80G 处理器。该系统的理论峰值速度为 1.5Tflops。

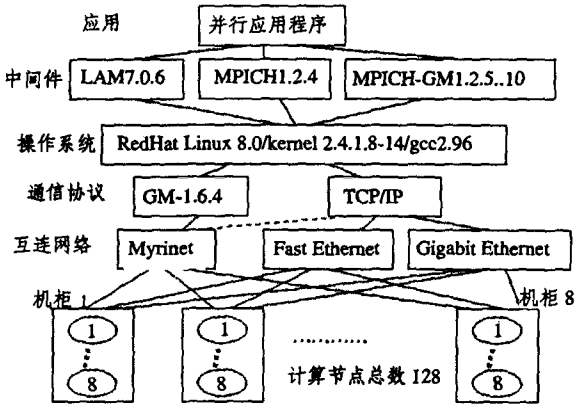


图 2　测试平台的整体架构

3.1　底层代数运算库-BLAS

BLAS(Basic Linear Algebra Subprograms)是实现高效率的向量、矩阵运算的核心，HPL 包中有大量的浮点运算是通过调用 BLAS 库实现的。这里测试 3 种不同的实现：

1) ATLAS(Automatically Tuned Linear Algebra Software)，这个工具可以自动为很多种现今的系统类型产生一个完整的经过优化的 BLAS 库。

2) Intel Math Kernel Library(MKL6.1)[10]，Intel 提供的数学核心库，用于在 Intel 平台上的工程、科学计算等等。

3) 由德克萨斯州大学的 Kazushige Goto 开发的高效的 BLAS 库，用 Fortran 编写，以下简称 GOTO BLAS。

虽然这些都是对相同的代数运算功能的实现，但不同的实现方法所生成的库的运行效率会相差很多。HPL 的核心运算都是通过调用底层 BLAS 库来实现的，因此 BLAS 实现的效率如何对 HPL 测试结果有着很大的影响。GOTO BLAS 是相对比较新的 BLAS 库，由于其很高的矩阵乘法的运算效率而逐渐得到广泛的使用。我们在实验中，用 32 个 2.80G 节点进行以上 3 种库的测试，得到的结果如表 1 所示。

可以看出，用 Intel MKL 比 ATLAS 得到的结果稍好一些，采用 GOTO 开发的 BLAS 库的结果最优。对于 GOTO BLAS 的具体实现方法可参见文[8]。以下测试中统一采用 GOTO BLAS 进行底层代数运算。

表 1　底层 BLAS 库的不同实现对集群性能指标的影响
(互连网络：Myrinet，N=40000，处理器分布为 4×8)

BLAS 库	Gflops	效率
ATLAS	94.28	52.6%
MKL	107.2	59.8%
GOTO	116.9	65.2%

3.2　互连网络

集群系统内部的互连网络对于整个系统的性能来说是十分重要的，因为通信部分往往是集群系统性能的瓶颈，必须用专用的高速互连才能满足高速计算的需要。在“深超-21C”中，我们采用高速网络 Myrinet-2000[6] 作为内部互连，同时，采用 100Mb/s 的以太网作为容错网络连接。以同样的参数配置在两种不同的通信网络上运行 HPL 测试，结果见图 3。

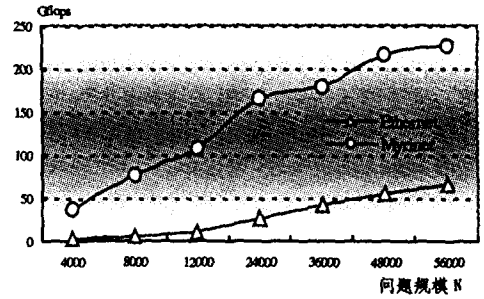


图 3　对于不同互连网络的测试结果

可以看出，使用高速 Myrinet 网络使整个集群系统的性能提高了很多。Myrinet-2000 高速网络的单向传输带宽可以达到 2.0Gb/s，短消息的通信延迟可以低于 10 微秒(usec)。在 64 位/64MHz 的 PCI 插槽上，Myrinet 卡的 PCI64C 接口可以达到双向持续的数据存储速率的极限值 500Mbytes/s，当然实际上能达到的速率还要受到主板上 PCI 总线读写速率的限制。

通信性能对提高集群整体性能有十分重要的意义。要进行并行运算，节点与节点间的通信操作是必不可少的，所以并行计算的总体性能不仅仅依赖于节点本身的运算速度，节点间的通信速度更为关键。

3.3　测试中所使用的参数

HPL 测试中选择的参数与测试的结果有很大的关系。从 HPL 的算法可以了解到了各个参数的含义，下面通过实际测试来考查对集群性能带来较大影响的几个重要参数的作用：

1) *测试中的问题规模 N*　这里的 N 也就是要求解的线性方程 $Ax=b$ 中 A 的维数。N 的选取需要考虑内存容量的制约关系，有一个达到最佳性能的上限值。可通过如下公式来估计这个值，其中 Mem 表示每个节点的物理内存大小。公式的含义为使用大约系统内存总和的 80%来进行 Linpack 测试，20%左右留给其他进程。

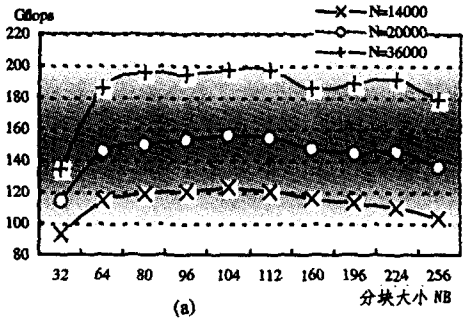
$$N \approx \sqrt{(\text{节点数} \times Mem/8) \times 80\%}$$

测试中选用的 N 不应超过这个值，否则将引起内存与磁盘交换的增加从而降低效率。（如果系统中运行的其他进程很少，可以再适当加大 N）。

按照这个公式估算用 32 个节点测试时应采用的最佳值为：$\sqrt{(32*1G/8)}*80\%=56569$。从图 3 中可以看出，对于小于 56000 的各个 N 的测试中，N 越大，得到的 Linpack 速度值越高。在实际的测试中，为了避免引起不必要的内存与磁盘的交换，可以先关闭内存和磁盘的交换区再进行测试。

2）*矩阵分块的大小 NB*　如 2.2 中所述，系数矩阵被分成 $NB\times NB$ 的循环块被分配到各个进程当中去处理，这里 NB 的大小也就是计算粒度，在很大程度上影响计算性能的优劣，粒度过粗或过细都会导致性能的下降，所以需要选择合适的粒度值。

在 32 个节点上开辟 64 个进程，用 10 种不同的 NB 值进行 3 种不同问题规模的 HPL 测试，结果如图 4(a)所示。



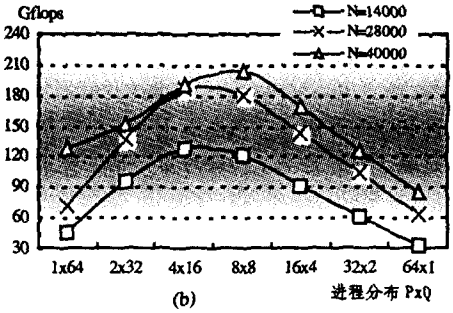


图 4　分块大小和进程分布对集群性能指标的影响

可以看出，采用 104 作为矩阵分块的大小是比较合适的，而 NB 偏小或偏大都会使性能有所下降。这个最佳值和使用的 BLAS 库关系很大，根据测试的经验，使用 GOTO BLAS 时选择 104 结果最好，使用 ATLAS 时应选择 80 的倍数，使用 MKL 时选择 64 的倍数比较好。最佳 NB 的值并不是固定的，跟具体的测试环境有关，需要经过多次实验才能找到。

3）*进程网格的分布 $P\times Q$*　进程网格的分布会影响到数据在各个处理器上的分布情况，而什么样的进程分布最合适取决于实际使用的通信网络架构是如何的。在 64 个处理器上测试了 7 种分布方式，结果如图 4(b)所示。

从图中可看出，P 和 Q 越接近测试结果越好（4×16 以及 8×8）。而在同样的分解方式下 $P<Q$ 比 $P>Q$ 的结果要好（例如 1×64 好于 64×1，2×32 好于 32×2）。

3.4　进程与处理器的对应关系

运行并行程序时，在每个处理器上分配一个进程，而对于 SMP 节点构成的集群来说，每个节点不只一个处理器。假设每节点拥有 s 个处理器，共 m 个节点，那么启动的进程数应为 $s*m$。进程的序号为 $0,1,\cdots,s*m-1$，这样进程和处理器就可以有不同的对应关系，典型的两种不同方式如下：（将节点 i 的 j 号处理器表示为 $P[i,j]$，序号均从 0 开始）

1）同一节点的处理器顺序排列，这样处理器和进程的对应方式为：

$P[i,j]\Leftrightarrow$进程$(i*s+j)$

2）按节点循环排列，共循环 s 次，这样处理器和进程的对应关系为：

$P[i,j]\Leftrightarrow$进程$(j*m+i)$

至于这两种排列方式哪一种性能更好，取决于要运行的程序的通信模型。对于以上两种情况进行 Linpack 测试的结果如图 5 所示，采用第 2 种对应方式的计算效率略高于第 1 种方式。

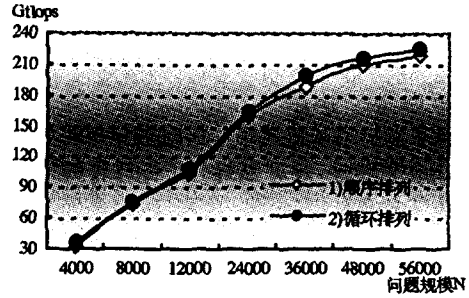


图 5　进程与处理器的对应关系对性能的影响

4　结果分析

实验结果表明，影响集群性能的因素中，包括硬件和软件两个方面，硬件方面包括处理器的主频、内存容量、互连网络的带宽延迟等等；软件方面包括测试数据规模等引起的 Cache 命中率、通信中的开销。一般对于一个特定的集群来说，硬件设施是已经确定的，所以优化工作应当尽可能地发挥硬件性能，同时要减少软件上的开销。从另一角度来看，集群是许多个节点松耦合方式互连而构成的，可以从单节点性能和多节点协作的性能两方面来分析。

4.1　单个节点的效率

要提高集群性能，首先抛开通信的因素，对单个节点进行优化，使计算速度加快。处理器的峰值速度由它的主频和每周期的浮点运算次数所决定，在峰值速度已经确定的情况下，对处理器效率所做的优化目的就是要使实际运算速度尽量接近峰值速度。

例如 3.1 节中将 BLAS 库采用的 GOTO BLAS，针对处理器硬件特点进行优化，将性能指标提升了很多。GOTO BLAS 库的设计思想主要基于如下考虑：

——浮点运算单元进行浮点运算的速度与从第二级 cache 经过的浮点数据流的速度相比的话，比例很小。因此，启用第二级 cache 处理这些数据流是一笔不必要的开销。

——很大一部分数据流的开销来自于阻塞 CPU 的 TLB（Translation Look-aside Buffers 转换表缓冲区）不命中，这种不命中很多是可以避免的，不能避免的也可以分摊到大量的计算当中去。

可见由于 GOTO BLAS 有效地提高了二级缓存的利用率，大大降低了 TLB 的不命中率，减少了不必要的开销，因此大大提高了性能。作为 HPL 需要频繁调用的底层库，BLAS 自身的高效率对 Linpack 测试的性能指标产生了显著的影响。

除了处理器以外，节点的物理内存容量对其计算效率也

是有一定的影响的。尤其对于SMP节点，内存越大，CPU访问内存引起的冲突概率就越小，从而提高系统中程序的运行速度。用两个SMP节点测试发现，将内存容量从512MB增加至1GB，运算效率从76.3%提升到80.5%。另一方面这也是由于内存容量大从而能够进行更大问题规模测试的关系。

4.2 节点间的通信

归根结底，通信是集群系统的瓶颈所在，因为进程间的通信始终是在计算之外的一笔开销。其它在软件上的改变影响系统性能的本质原因是影响了通信开销所占比例。

1)*问题规模N的选择* 问题规模的大小也决定了计算量的大小，N越大，需要花费的时间越长，需要占用的内存空间就越多。如果选择的N超过一定值，内存就不够用而发生磁盘交换，从而大大降低系统的计算效率。所以选择N之前应该首先按照前面所提到公式估算一下。在这个临界值之下，N越大，性能指标就越高，是因为N变大了，那么计算量相对于通信量的开销也就增大，即N较大时计算量/通信量比也较大。

2)*数据块大小NB的选择* 从数据分布的角度上来看，NB越小，分块就越多，也就越有利于各进程的负载平衡；但从计算的角度上来看，太小的NB，会限制系统的计算性能，因为在最高一级的存储结构中数据重用很少，而且需要交换的消息增多。NB的最佳值跟N的大小有关，也跟整个系统的计算/通信性能比有关。

衡量NB是否最佳的根本原因还是看它产生的通信开销是否达到最小，从通信角度来看，NB太小会大大地增加进程间的通信量，降低计算所占比重；而NB太大，影响各个处理器的负载平衡，同样也会增加通信开销。所以，如果计算性能比起通信性能来要优越些，则适合选择较大的NB值；反之，可以尝试选择较小的NB值。

3)*处理器网格分布的选择* 从测试结果可以看到P和Q比较接近且P小于Q的时候，效率最高，原因如下：P和Q接近时有利于进程之间的负载平衡，行列方向通信能够保持平衡，否则可能同一对处理器之间要进行大量消息的通信；在此前提下，P要稍小于Q，因为根据Linpack的算法，列向通信量要大于行向通信量，使列向的进程稍多一些可以减少通信开销。

4)*进程和处理器的对应关系* 选择进程和处理器的对应关系时应考虑程序通信模型的特点。对于Linpack，矩阵是以转置形式存放的，在计算中通信最频繁的Panel分解中，列与列之间的通信比较多。3.4节中所述的第1种方式中，同一列的分块分配到相同节点的处理器上进行处理，容易造成一对节点间有多对进程需要通信的情况，于是造成网络端口的竞争、阻塞和等待，从而浪费了一定时间。而第2种方式可以将相同节点的处理器分隔开来，在很大程度上减少了这种多处理器竞争同一资源的状况，因此计算所用总时间较少。可参见图5的测试结果。

5 优化方向和优化方法

以上的测试数据和分析结果虽然只基于HPL测试这一个方面，但目的是要通过测试来寻找和总结提高集群性能指标的有效方法，对集群系统的性能进行优化。根据前面的结果分析，可以总结出对集群系统优化方法如下：

1)*针对处理器特点提高效率* 针对CPU内部结构的特点，如流水线技术，寄存器，高速缓存等等，采取一定的方法来提高其计算的效率，是一种可行的有效的优化方案。从测试结果看，改用GOTO BLAS进行底层代数运算，可将性能提升10%～20%。另外，针对不同的CPU使用更有效的编译器，并利用合适的参数开关相组合，也是一种处理器效率的优化方法。

2)*改进通信网络，减少传输延迟* 从硬件体系上来看，采用Myrinet网络比采用以太网要改善很多，可将性能提升到原来的3～4倍。其软件包GM[11]提供了高效的用户层通信协议，GM在Myrinet的LANai芯片上运行控制程序(MCP)来承担大部分通信处理工作，减轻了主机的开销。专用于集群的高速网络还有InfiniBand[9]，QsNet[12]等。考虑到未来网络传输速度将不断加快，总线的带宽也将会成为系统数据通路中的瓶颈部分。InfiniBand正是利用可交换、点对点的网络来提供不同于传统PCI的架构，以及数倍于PCI总线的带宽。另一方面，QsNet也具有很高的带宽和很低的延迟，然而基于QsNet之上的应用与InfiniBand比起来相对较少。

3)*通过软件方法降低通信开销* 硬件结构已经确定的前提下，可通过软件方法来降低通信开销。通过合理的并行方案的设计，可以使完成同样任务的情况下减少花费在数据传输上的时间。例如前面对HPL参数的修改，以及处理器和进程对应关系的变换，都起到了减小通信开销比的作用。测试中通过合理的调节，可将集群的实际运算饱和速率提高3%～8%。编写并行程序时应充分考虑到如何尽量发挥多处理器的有利特点，提高加速比。

结束语 本文通过集群系统上的HPL基准测试，考查了一些对性能影响较大的重要因素，并分析测试结果提出了集群系统可优化的基本方向。当然，只通过一项HPL测试来评定集群的性能是不全面的，但至少从一个很重要的侧面反映了集群系统的性能优劣。目前一种面向高性能计算机的综合基准测试程序包HPC Chanllenge[13]受到众多关注与讨论，该程序包在HPL的基础上扩充了对存储性能方面的测试，针对于多种应用的不同存储访问模型，其中HPL代表了时间局部性和空间局部性都很高的一种典型应用。我们的下一步工作是基于该综合测试包来评价集群的性能，找出HPL的不足之处，对集群的更多重要方面的性能进行分析和优化。

参考文献

1 Buyya R著，郑纬民，石威，汪东升，等译. 高性能集群计算：结构与系统/(美)[M]. 北京：电子工业出版社，2001
2 Bunch J, Dongarra J, Moler C, et al. LINPACK users' guide [M]. SIAM, Philadelphia: PA, 1979
3 Dongarra J, Meuer H W, Strohmaier E, eds. TOP500 Report 1996 [J]. SUPERCOMPUTER, 1997, 13(1): 89～120
4 Richards M. Introduction to the 1st VSIPL Tutorial&User's Conference [EB/OL]. http://www.vsipl.org/CD/richards1.pdf, 2002
5 都志辉，吴博，刘鹏，等. LINPACK与机群系统的LINPACK测试[J]. 计算机科学，2002，29(5)：8～10
6 Boden N, et al. Myrinet - a Gigabit-per-second local-area-network [J]. IEEE Micro, 1995, 15(1): 29～36
7 Kagstrom B, Ling P, Van Loan C. Portable High Performance GEMM-based Level 3 BLAS [A]. In: R F Sincovec, et al. eds. Parallel Processing for Scientific Computing [C]. Philadelphia: SIAM, 1993. 339～346
8 Goto K, van de Geijn R. On Reducing TLB Misses in Matrix Multiplication. Technical Report TR-2002-55. FLAME Working Note #9, The University of Texas at Austin: Department of Computer Sciences, 2002
9 InfiniBand Trade Association. InfiniBand(tm) Architecture Specification [R]. Release 1.1, 2002, 1
10 Intel Math Kernel Library. Webpage. http://www.intel.com/software/products/mkl/
11 GM reference manual [EB/OL]. http://www.myri.com/scs/GM/doc/refman.pdf
12 Petrini F, Feng Wu-chun, Hoisie A, et al. The Quadrics Network (QsNet): High-Performance Clustering Technology [J]. IEEE Micro, 2002, 22(1): 46～57
13 HPC Challenge Benchmark [EB/OL]. http://icl.cs.utk.edu/hpcc