湖南大學

HUNAN UNIVERSITY

计算机系统设计

选题名称：Linpack 标准测试程序及其分析

姓　　名：肖若愚

学　　号：201708010619

专业班级：物联 1702

Linpack 标准测试程序及其分析

一、Linpack 概述

Linpack 是国际上使用最广泛的测试高性能计算机系统浮点性能的基准测试。通过对高性能计算机采用高斯消元法求解一元 N 次稠密线性代数方程组的测试，评价高性能计算机的浮点计算性能。Linpack 的结果按每秒浮点运算次数（flops）表示。

很多人把用 Linpack 基准测试出的最高性能指标作为衡量机器性能的标准之一。这个数字可以作为对系统峰值性能的一个修正。通过测试求解不同问题规模的实际得分，我们可以得到达到最佳性能的问题规模，而这些数字与理论峰值性能一起列在 TOP500 列表中。

Linpack 测试包括三类，Linpack100、Linpack1000 和 HPL。Linpack100 求解规模为 100 阶的稠密线性代数方程组，它只允许采用编译 优化选项进行优化，不得更改代码，甚至代码中的注释也不得修改。Linpack1000 要求求解 1000 阶的线性代数方程组，达到指定的精度要求，可以在 不改变计算量的前提下做算法和代码上做优化。HPL 即 High Performance Linpack，也叫高度并行计算基准测试，它对数组大小 N 没有限制，求解问题的规模可以改变，除基本算法（计算量）不可改变外，可以采用其它任何优化方 法。前两种测试运行规模较小，已不是很适合现代计算机的发展。

HPL 是针对现代并行计算机提出的测试方式。用户在不修改任意测试程序的基础上，可以调节问题规模大小(矩阵大小)、使用 CPU 数目、使用各种优化方法等等来执行该测试程序，以获取最佳的性能。HPL 采用高斯消元法求解线性方程组。求解 问题规模为 N 时，浮点运算次数为(2/3 * N^3 − 2*N^2)。因此，只要给出问题规模 N，测得系统计算时间 T，峰值=计算量(2/3 * N^3 − 2*N^2)/计算时间 T，测试结果以浮点运算每秒（Flops）给出。HPL 测试结果是 TOP500 排名的重要依据。

计算机计算峰值简介：衡量计算机性能的一个重要指标就是计算峰值或者浮点计算峰值，它是指计算机每秒钟能完成的浮点计算最大次数。包括理论浮点峰值和实测浮点峰值。理论浮点峰值是该计算机理论上能达到的每秒钟能完成浮点 计算最大次数，它主要是由 CPU 的主频决定的。计算公式如下： 理论浮点峰值 = CPU 主频×CPU 每个时钟周期执行浮点运算次数×CPU 数量。

CPU 每个时钟周期执行浮点运算的次数是由处理器中浮点运算单元的个数及每个浮点运算单元在每个时钟周期能处理几条浮点运算来决定的，下表是常见 CPU 的每个时钟周期执行浮点运算的次数。

二、运行环境

硬件平台：2 台以上计算节点（单节点也可进行且无需网络结构）
网络环境：各计算节点间网络连通，网络结构推荐高带宽、低延时的 Voltaire Infiniband 网络
系统环境：Linux 平台 存储：各节点间拥有共享存储（并行文件系统最优）
软件：编译器、MPI、数学库、linpack 测试包等

三、测试运行

1． Linpack 安装条件：

在安装 HPL 之前，系统中必须已经安装了编译器、并行环境 MPI 以及基本线性代数子方程(BLAS)或矢量图形信号处理库(VSIPL)两者之一。

编译器必须支持 C 语言和 Fortran77 语言。并行环境 MPI 一般采用 MPICH，当然也可以是其它版本的 MPI，如 LAM－MPI。HPL 运行需要 BLAS 库或者 VSIPL 库，且库的性能对最终测得的 Linpack 性能有密切的关系。常用的 BLAS 库有 GOTO、Atlas、ACML、ESSL、MKL 等，我的测试经验是 GOTO 库性能最优。

2． 安装与编译：

第一步，从 www.netlib.org/benchmark/hpl 网站上下载 HPL 包 hpl.tar.gz 并解包，目前 HPL 的最新版本为 hpl 1.0a。

第二步，编写 Make 文件。从 hpl/setup 目录下选择合适的 Make.<arch>文件 copy 到 hpl/目录下，如：Make.Linux_PII_FBLAS 文件代表 Linux 操作系统、PII 平台、采用 FBLAS 库；Make.Linux_PII_CBLAS_gm 文件代表 Linux 操作系统、PII 平台、采用 CBLAS 库且 MPI 为 GM。HPL 所列都是一些比较老的平台，只要找相近平台的文件然后加以修改即可。修改的内容根据实际环境的要求，在 Make 文件中也作了详细的说明。主要修改的变量有：

ARCH： 必须与文件名 Make.<arch>中的<arch>一致

TOPdir：指明 hpl 程序所在的目录

MPdir： MPI 所在的目录

MPlib： MPI 库文件

LAdir： BLAS 库或 VSIPL 库所在的目录

LAinc、LAlib：BLAS 库或 VSIPL 库头文件、库文件

HPL_OPTS：包含采用什么库、是否打印详细的时间、是否在 L 广播之前拷贝 L

若采用 FLBAS 库则置为空，采用 CBLAS 库为“-DHPL_CALL_CBLAS”，采用 VSIPL 为“-DHPL_CALL_VSIPL”

“-DHPL_DETAILED_TIMING”为打印每一步所需的时间，缺省不打印

“-DHPL_COPY_L”为在 L 广播之前拷贝 L，缺省不拷贝（这一选项对性能影响不是很大）

CC： C 语言编译器

CCFLAGS：C 编译选项

LINKER：Fortran 77 编译器

LINKFLAGS：Fortran 77 编译选项（Fortran 77 语言只有在采用 Fortran 库是才需要）

3、对 HPL 文件进行配置

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
384 384 384 384   Ns
4            # of NBs
32 64 128 256        NBs
1            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
2            Ps
2            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

第 1 行 HPLinpack benchmark input file
第 2 行 Innovative Computing Laboratory, University of Tennessee
前两行为说明性文字，不用作修改
第 3 行 HPL.out output file name (if any)
第 4 行 6　device out (6=stdout,7=stderr,file)
device out"为"6"时，测试结果输出至标准输出（stdout）
"device out"为"7"时，测试结果输出至标准错误输出（stderr）
"device out"为其它值时，测试结果输出至第三行所指定的文件中
可以通过设置此处用来保存测试结果。**(这里我们将结果输出至标准输出)**
第 5 行 1　# of problems sizes (N)

选择矩阵的数量 如 1 则为第一个矩阵。**(这里我们设置矩阵的数量为 4)**
第 6 行 10240 26680 28800 30720 29 30 34 35 Ns
矩阵的规模 N 越大，有效计算所占的比例也越大，系统浮点处理性能也就越高；但与此同时，矩阵规模 N 的增加会导致内存消耗量的增加，一旦系统实际内存空间不足，使用缓存、性能会大幅度降低。
由于之前采用了大页面内存系统，所以此处计算规模的大小，应以设置的大页面内存总量做计算。计算方式为：N*N*8=大页内存总量*0.8，内存总量换算为字节。
而且规模的大小最好为 384 的倍数。
(其实我一开始也想把矩阵搞大点的，但是我虚拟机非常卡，电脑有点垃圾，所以一开始矩阵搞的很大以后跑的又慢，后面索性就把矩阵规模调的比较小了，这里我用了 4 个 384 的矩阵)
第 7 行 1 # of NBs
第 8 行 128 2 3 4   NBs
提高数据的局部性，从而提高整体性能，HPL 采用分块矩阵的算法。分块的大小对性能有很大的影响，NB 的选择和软硬件许多因素密切相关。NB 值的选择主要是通过实际测试得到最优值。但 NB 的选择上还是有一些规律可寻，如：NB 不可能太大或太小，一般在 256 以下；NB × 8 一定是 Cache line 的倍数等。例如，我们的 L2 cache 为 1024K, NB 就设置为 192 另外，NB 大小的选择还跟通信方式、矩阵规模、网络、处理器速度等有关系。一般通过单节点或单 CPU 测试可以得到几个较好的 NB 值，但当系统规模增加、问题规模变大，有些 NB 取值所得性能会下降。所以最好在小规模测试时选择 3 个左右性能不错的 NB，再通过大规模测试检验这些选择。此处一般选择 128。
(这里我们的 NB 用的 256)
第 10 行 3 # of process grids (P x Q)
第 11 行 2 1 4 Ps
第 12 行 2 4 1 Qs
）第 10 ~ 12 行说明二维处理器网格（P × Q），二维处理器网格（P × Q）的有以下几个要求。
P × Q = 系统 CPU 数 = 进程数。一般来说一个进程对于一个 CPU 可以得到最佳性能。对于 Intel Xeon 来说，关闭超线程可以提高 HPL 性能。P≤Q；一般来说，P 的值尽量取得小一点，因为列向通信量（通信次数和通信数据量）要远大于横向通信。P = 2n，即 P 最好选择 2 的幂。HPL 中，L 分解的列向通信采用二元交换法（Binary Exchange），当列向处理器个数 P 为 2 的幂时，性能最优。例如，当系统进程数为 4 的时候，P × Q 选择为 1 × 4 的效果要比选择 2 × 2 好一些。 在集群测试中，P × Q = 系统 CPU 总核数。如系统为总核数为 16 核，则 P*Q 值应该为 4.**(这里我们用的单进程测试 P* Q 为 1)**
后面就基本都没有作什么修改了，配置文件就这样吧!

4、运行结果
结果输出到了一个 txt 文档里，结果已附在了文档中，最后发现矩阵大小还是比较影响结果的，一开始 N 调的很大确实跑了好久…