# Linpack标准测试程序及其分析

1. **Linpack简介**  
    Linpack是国际上最流行的用于测试高性能计算机系统浮点性能的benchmark。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性能。  
    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。
2. **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。  
    理论浮点峰值＝CPU主频×CPU每个时钟周期执行浮点运算的次数×系统中CPU数。  
    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，下表是各种CPU的每个时钟周期执行浮点运算的次数。

实测浮点峰值是指Linpack值，也就是说在这台机器上运行Linpack测试程序，通过各种调优方法得到的最优的测试结果。在实际程序运行中，几乎不可能达到实测浮点峰值，更不用说理论浮点峰值了。这两个值只是作为衡量机器性能的一个指标。

**三．Linpack安装**

（1）安装MPICH2，并配置好环境变量，本书前面已作介绍。

（2）进入Linux系统，建议使用root用户，在/root下建立linpack文件夹，解压下载的Gotoblas和HPL文件到linpack文件夹下，改名为Gotoblas和hpl。

#tar xvf GotoBLAS-\*.tar.gz

#mv GotoBLAS-\*  ~/linpack/Gotoblas

#tar xvf  hpl-\*.tar.gz

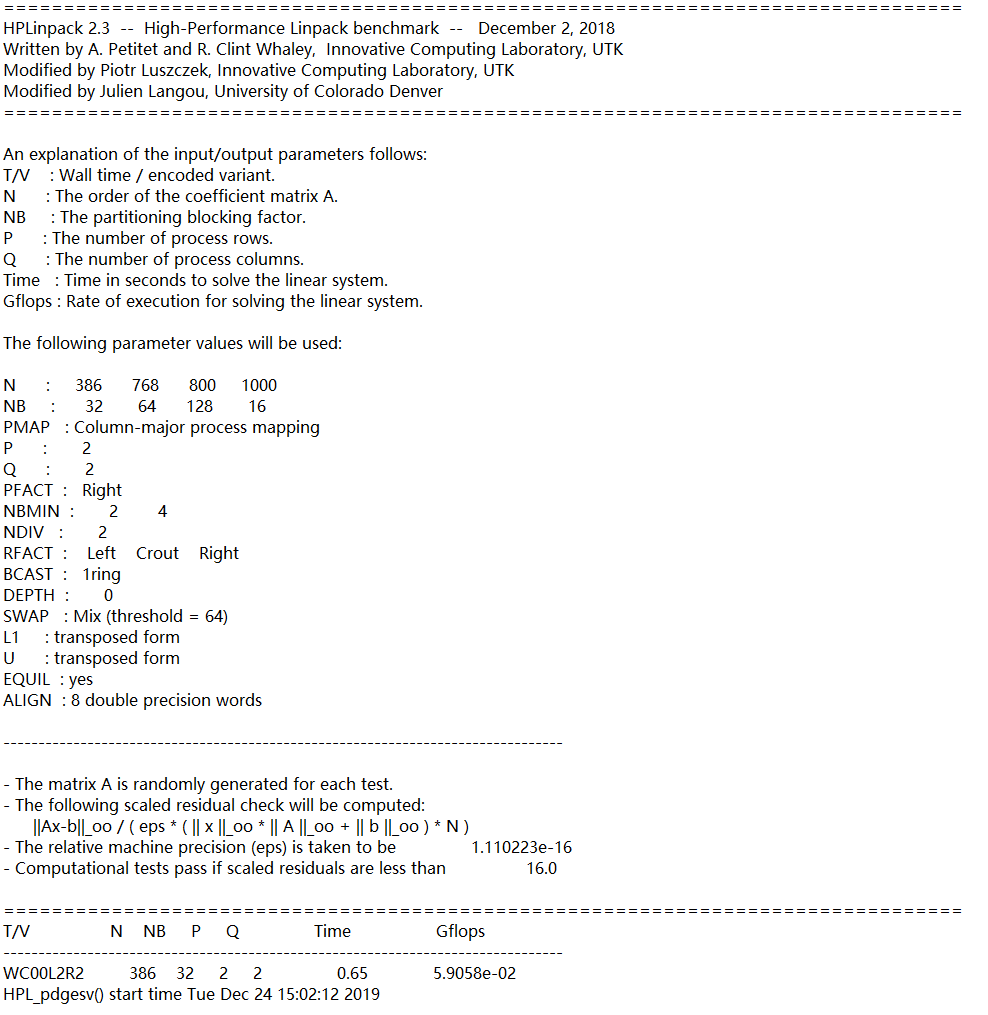
#mv hpl-\*  ~/linpack/hpl

（3）安装Gotoblas。

进入Gotoblas文件夹，在终端下执行./ quickbuild.64bit（如果你是32位系统，则执行./ quickbuild.31bit）进行快速安装，当然，你也可以依据README里的介绍自定义安装。如果安装正常，在本目录下就会生成 libgoto2.a和libgoto2.so两个文件。

（4）安装HPL。

进入hpl文件夹从setup文件夹下提取与自己平台相近的Make.<arch>文件，复制到hpl文件夹内，比如我们的平台为 Intel xeon，所以就选择了Make.Linux\_PII\_FBLAS，它代表Linux操作系统、PII平台、采用FBLAS库。



如果编译正常，在hpl/bin/Linux\_xeon目录下就会生成两个文件HPL.dat和xhpl。 HPL.dat文件是Linpack测试的优化配置文件，这个对测试的结果十分重要，xhpl为可执行程序。  
 如果是集群测试，就将linpack目录复制到机群中其余节点相同的目录下。  
至此，安装就算完成了。现在可以进行简单的测试.  
 进入hpl/bin/Linux\_xeon目录，先启动mpd服务，在终端运行命令启动测试，就会看到测试开始了，但得到的数据会比较差，有时可能连最佳值的1%都达不到，这就需要根据自己平台的情况进行测试优化。

1. **Linpack优化方法**  
    在Linpack测试的理论分析工作中，需要解决矩阵的规模、矩阵分块、进程的映射、L通信、U通信、负载均衡等问题。而Linpack的测试结果受多方面因素的影响，包括与算法相关的参数设置、CPU的架构数量和效率、内存容量、互联网络的通信性能、系统规模等。这些因素共同决定了系统Linpack的测试的最终结果。  
   每个参数的含义及一般配置：

（1）

"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

（2）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。因此，对于一般系统而言，要尽量增大矩阵规模N的同时，又要保证不使用系统缓存。因为操作系统本身需要占用一定的内存，除了矩阵（N × N）之外，HPL还有其他的内存开销，另外通信也需要占用一些缓存。矩阵占用系统总内存的80%左右为最佳，即N × N × 8 = 系统总内存×80%。

（3）为提高数据的局部性，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。

（4）在消元过程中，zHPL采用每次完成NB列的消元，然后更新后面的矩阵。这NB的消元就是L的分解。每次L的分解只在一列处理器中完成。对每一个小矩阵作消元时，都有3种算法：L、R、C，分别代表Left、Right和Crout。在LU分解中，具体的算法很多，测试经验，NDIVs选择2比较理想，NBMINs 4或8都不错。而对于RFACTs和PFACTs，对性能的影响不大。

（5）L和U的数据存放格式，若选择"transposed"，则采用按列存放，否则按行存放。

（6）U的广播算法，U的广播为列向广播，HPL提供了3种U的广播算法：二元交换（Binary Exchange）法、Long法和二者混合法。SWAP="0"，采用二元交换法；SWAP="1"，采用Long法；SWAP="2"，采用混合法。

**五．Linpack实际测试结果**  
当上面介绍的一切都顺利完成后，就可以对系统进行实际测试了，步骤如下：  
当测试顺利完成后，会得到如下的结果：  
以上的数据显示，最大的数值为7.699e+00Gflops，即为76.99亿次每秒。

