linpack 测试及其分析

一、linpack 简介

linpack 是线性系统软件包(Linear system package) 的缩写。通过利用高性能计算机，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。当 Linpack 求解问题规模为 N 时，浮点计算次数为(2/3 * N^3+3/2N^2) ,若计算时间为 t,则 HPL 测试值=计算量 x (2/3 * N^3+3/2N^2) /t。[1]

Linpack 测试包括三类，Linpack100、Linpack1000 和 HPL。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，因此现在使用较多的测试标准为 HPL，因此本次讨论的主要是 HPL。

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。

二、HPL 算法简介

HPL 软件包是用来求解线性方程组 A x = b 的解。其中 $A=(a_{i\ j})_{n*n}$ 为非奇异矩阵；$b=(b_1, b_2, \dots ,b_n)^T$, $x=(x_1, x_2, \dots ,x_n)^T$, A 与 b 均已知，而 x 是待求的 n 维向量。HPL 采用的方法为经典的 LU 分解法。而 LU 分解的算法最经典的是高斯消去法，有消元过程和回代过程两步。

首先需要将 A 矩阵分解为下三角矩阵 L 和上三角矩阵 U 两个矩阵。使得 L U = A , 则 A x = b 可以转化为 L U x = b , 令 U x = y ,则可以将问题转化为 L y = b ，解得 y 后再将 y 代入 U x = y 中，解得 x。因为 L 和 U 都是三角矩阵，因此易使用追赶法得到解。“追”即分解过程，“赶”即是回代过程。

三、linpack 测试程序编译配置

测试环境：测试环境为自行组装的台式主机，机器配置为八个 Intel Core i7-6700K CPU @4.00GHz *8, 16G DDR4 内存，200G 机械硬盘，操作系统为 Ubuntu 18.04.3 LTS, MPI 和 HPL 的版本分别为 mpich-3.2 和 HPL2.3, BLAS 选择 OpenBLAS 软件包。

OpenBLAS 以及 mpich-3.2 的安装这里不做详细介绍，只对 HPL 编译选项进行描述：在 setup 文件夹下找到 Makefile.Linux_PII_FBLAS 文件，将文件内容拷贝到 Makefile.test。修改后内容如下：

```
SHELL        = /bin/sh

CD           = cd
CP           = cp
LN_S         = ln -s
MKDIR        = mkdir
RM           = /bin/rm -f
TOUCH        = touch

ARCH         = test

TOPdir       = /home/drone/linpack/hpl-2.3
INCdir       = $(TOPdir)/include
BINdir       = $(TOPdir)/bin/$(ARCH)
LIBdir       = $(TOPdir)/lib/$(ARCH)

HPLlib       = $(LIBdir)/libhpl.a

MPdir        = /home/drone/linpack/setmpich
MPinc        = -I$(MPdir)/include
MPlib        = $(MPdir)/lib/libmpi.a

LAdir        = /home/drone/linpack/setblas
LAinc        =
LAlib        = $(LAdir)/lib/libopenblas.a $(LAdir)/lib/libopenblas.so

F2CDEFS      = -DAdd__ -DF77_INTEGER=int -DStringSunStyle

HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) $(LAinc) $(MPinc)
HPL_LIBS     = $(HPLlib) $(LAlib) $(MPlib)

HPL_OPTS     =

HPL_DEFS     = $(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES)

CC           = /home/drone/linpack/setmpich/bin/mpicc
CCNOOPT      = $(HPL_DEFS)
CCFLAGS      = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W
CCFLAGS      = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W -Wall -fuse-ld=gold -pthread -lm

LINKER       = /home/drone/linpack/setmpich/bin/mpif77
LINKFLAGS    = $(CCFLAGS)


ARCHIVER     = ar
ARFLAGS      = r
RANLIB       = echo
```

其中需要设置的量及其代表的意义。其余值不变。

```
ARCH        = test                          //需要和 Makefile 后面的名称一致
TOPdir      = /home/drone/linpack/hpl-2.3   //hpl 代码包解压位置
MPdir       = /home/drone/linpack/setmpich  //mpi 库安装位置
LAdir       = /home/drone/linpack/setblas   //blas 库安装位置

//blas 库文件绝对路径
LAlib       = $(LAdir)/lib/libopenblas.a $(LAdir)/lib/libopenblas.so

//安装的 mpi 的 c 语言编译器
CC          = /home/drone/linpack/setmpich/bin/mpicc

//安装的 mpi 的 c 语言编译器选项
CCFLAGS     = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W -Wall -fuse-
ld=gold -pthread -lm

//安装的 mpi 的 c 语言链接器
LINKER      = /home/drone/linpack/setmpich/bin/mpif77
```

在设置完成后在 bin 文件夹下会生成 xhpl 文件作为测试程序，hpl.dat 为输入配置文件。

运行 mpirun -np X ./xhpl 即可运行程序。

四、linpack 测试结果分析

HPL.dat 文件是 linpack 测试中需要输入的文件，以下是本次测试的一些配置。

```
N      :  20000   21000
NB     :    192     232     256
PMAP   : Column-major process mapping
P      :      2
Q      :      2
PFACT  :   Left    Crout   Right
NBMIN  :      2       4
NDIV   :      2
RFACT  :   Left    Crout   Right
BCAST  :  1ring
DEPTH  :      0
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words
```

其中 n 表示所要解决的方程组的大小，NB 表示分块大小，P*Q 表示所开的进程数量。其余数字不需要更改。

摘取以下两种结果会发现 NB 分块大小会对计算速率有影响，这算是 linpack 中 LU 分解算法导致的原因。测算出来本次测试单个节点的峰值速率为当 N 为 21000 时分块大小为 192 时的速率为 87.307Gflops.

WC00L2R4			
N	NB	Time	Gflops
20000	192	63.92	83.45
20000	232	62.29	85.633
20000	256	64.86	82.233
21000	192	70.72	87.307
21000	232	73.79	83.675
21000	256	76.38	80.836

WC00L2C4			
N	NB	Time	Gflops
20000	192	93.84	56.841
20000	232	86.53	61.643
20000	256	101.75	52.42
21000	192	109.8	56.237
21000	232	94.87	65.083
21000	256	116.27	53.108

以上数据均是整理出来的，最终计算结果都是通过的。