HPL性能调优

环境

Ubuntu22.04 插电状态

笔记本关闭显示,不进入休眠,注销用户,在其他电脑上利用ssh连接

RAM:16G DDR3

CPU: i7-4980HQ 2.8Ghz 核心数:4 线程数:8

性能影响

- vscode-server的ssh远程编程进程可能会对性能有影响
- ssh的连接会影响性能
- 实验过程中,开启NextSSH软件中的数据监看功能,实时监测cpu、内存、网络波动
- 实验时开启了Wi-Fi功能
- 内存
- 笔记本发热降频

参考

- https://github.com/yunzhongOvO/Linpack-HPL
- https://www.intel.com/content/www/us/en/products/sku/83503/intel-core-i74980hq-processor-6m-cache-up-to-4-00-ghz/specifications.html
- https://www.intel.com/content/www/us/en/products/sku/75789/intel-xeon-processor-e52620-v2-15m-cache-2-10-ghz/specifications.html

理论峰值

Gfloat理论峰值：核心数 *主频* 每时钟周期浮点运算次数

本机CPU：Intel Core i7-4980HQ

CPU主频：2.8GHz
每时钟周期浮点运算次数：8
Gflops理论峰值：$4 * 2.8GHz * 8 = 89.6Gflops$

这部分网上资料搜索有难度,4980HQ使用了avx256指令集,同时有2个FMA,每时钟周期浮点运算次数应当是8*2=16

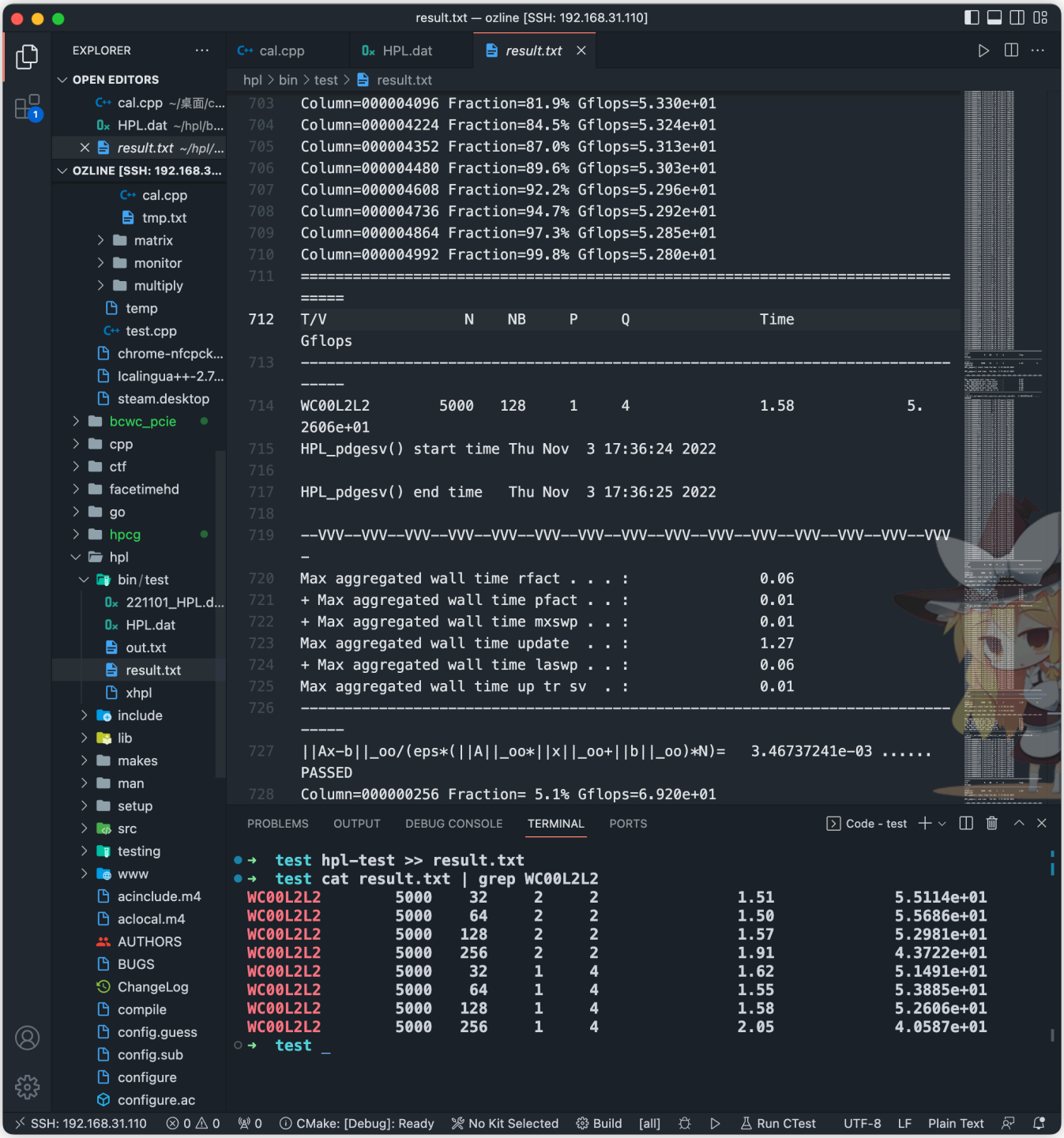
但是不确定FMA是否需要加入,这里只考虑了*8的情况

根据后续实际测试结果,应该是乘以8的

测试方式

- 局域网远程连接测试电脑,启用vscode-server
- 先行跑一些数据,预热机器
- `mpirun -np 8 ./xhpl`
- 开始测试,直接将测试结果输出到 `result.txt` 文件
- 利用命令 `cat result.txt | grep WC00L2L2` 获取答案

测试例图:



默认测试

```
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)
1            # of problems sizes (N)
5000         Ns
1            # of NBs
1            NBs
0            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
0            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2            NBMINs (≥ 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            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)
```

```
➜  test mpirun -np 8 ./xhpl
```

测试过程中电脑数据情况如图所示

测试结果

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L2        5000     1     2     2              27.06             3.0806e+00
```

不知道是理论出错还是什么原因,的确是只有3.08Gflops...

实际理论比值约为1.76%,考虑到可能是NB值过小原因

参数优化

矩阵规模N

根据公式 $N * N * 8 = 物理内存容量 * 0.8$

这里物理内存容量单位是bytes,这台机子的内存为16G,那么我们经过计算需要的N值大约为41448.6

考虑到我们实际上是取了80%,而可以考虑取到80-85%,再根据经验贴考虑384的倍数

那么我们可以确定N值为**41472**,也就是说要在这个范围内测试

这里考虑暂时以n=5000为测试

疑问:这里的公式似乎有错?我认为取不到41472

参考的一篇文档,47G内存N可以到52900,类比一下16G内存应该只能到三分之一甚至更低?

后续的数据也可以看出,实际上N的取值范围对于本机来说大约在10000左右

二维处理器网格P*Q

根据公式:$P * Q = 进程数 = 系统CPU数$

一般来说一个进程对于一个CPU可以得到最佳性能

一般情况下,P应当略小于Q,因为列向通信量的通信次数和通信数据量大于横向通信.

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

综上,确定 `P=2 Q=2` 和 `P=1 Q=4` 两组数据

矩阵分块NB

HPL 使用块大小 NB 进行数据分布和计算粒度。从数据分布的角度来看，NB越小，负载平衡越好。从计算的角度来看，NB 值太小可能会限制计算性能，消息的数量将增加。NB的选择和软硬件许多因素密切相关。

NB一般在256以下，NB的最优值主要通过实际测试来得到。

本实验中选取 `32、64、128、256` 五组数据

PMAP process mapping

HPL文档中介绍，按列的排列方式适用于节点数较多、每个节点内CPU数较少的系统；按行的排列方式适用于节点数较少、每个节点内CPU数较多的大规模系统。在机群系统上，按列的排列方式的性能远好于按行的排列方式，此处一般选择1。

测试中选择 `1`

第一次测试

单个测试ID测试约10次,数据取平均值

N=5000

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	5000	2*2	32	1	1.51	5.5114e+01	61.51%
2	5000	2*2	64	1	1.50	5.5686e+01	62.15%
3	5000	2*2	128	1	1.57	5.2981e+01	59.13%
4	5000	2*2	256	1	1.91	4.3722e+01	48.80%
5	5000	1*4	32	1	1.62	5.1491e+01	57.47%
6	5000	1*4	64	1	1.55	5.3885e+01	60.14%
7	5000	1*4	128	1	1.58	5.2606e+01	58.71%
8	5000	1*4	256	1	2.05	4.0587e+01	45.30%

第一轮测试数据如表,效果最好的是2号数据

N=10000

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	10000	2*2	32	1	13.87	4.8061e+01	53.64%
2	10000	2*2	64	1	10.96	6.0826e+01	67.89%
3	10000	2*2	128	1	10.44	6.3850e+01	71.26%
4	10000	2*2	256	1	11.90	5.6054e+01	62.56%
5	10000	1*4	32	1	13.83	4.8225e+01	53.82%
6	10000	1*4	64	1	10.08	6.1314e+01	68.43%
7	10000	1*4	128	1	10.91	6.1104e+01	68.20%
8	10000	1*4	256	1	11.94	5.5839e+01	62.32%

第二轮测试数据如表,效果最好为3号数据

N=15000

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	15000	2*2	32	1	52.85	4.2578e+01	47.52%
2	15000	2*2	64	1	45.03	4.9978e+01	55.78%
3	15000	2*2	128	1	41.98	5.3607e+01	59.83%
4	15000	2*2	256	1	48.16	4.6725e+01	52.15%
5	15000	1*4	32	1	73.10	3.0786e+01	34.36%
6	15000	1*4	64	1	55.96	4.0213e+01	44.88%
7	15000	1*4	128	1	53.44	4.2106e+01	46.99%
8	15000	1*4	256	1	52.82	4.2603e+01	47.55%

第三轮测试数据如表,效果最好为3号数据

N=20000

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	20000	2*2	32	1	138.93	3.8393e+01	42.85%
2	20000	2*2	64	1	130.79	4.0783e+01	45.52%
3	20000	2*2	128	1	123.08	4.3337e+01	48.37%
4	20000	2*2	256	1	129.60	4.1156e+01	45.93%
5	20000	1*4	32	1	178.66	2.9855e+01	33.32%
6	20000	1*4	64	1	152.13	3.5061e+01	39.13%
7	20000	1*4	128	1	141.92	3.7583e+01	41.95%
8	20000	1*4	256	1	131.40	4.0594e+01	45.31%

第四轮测试数据如表,效果最好的仍为3号数据

小结

可以发现,这四组数据中

- 关于N的甜点值大约在10000附近
- 效果最好的除了N=5000的情况,其余均为3号数据,而N=5000的情况下,2号数据和3号数据差异不大

在后续测试中,将会以三号数据的情况为基准,N=10000开始测试

第二次测试

N=8192

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	8192	2*2	96	1	5.84	6.2826e+01	70.12%
2	8192	2*2	128	1	6.31	5.8132e+01	64.88%
3	8192	2*2	160	1	7.16	5.3695e+01	59.83%
4	8192	2*2	96	0	6.24	5.8795e+01	65.62%
5	8192	2*2	128	0	6.71	5.4660e+01	61.00%
6	8192	2*2	160	0	6.78	5.4043e+01	60.32%

PMAP process mapping=1时效果最好

NB=96时的效果最好

N=10240

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	10240	2*2	96	1	11.10	6.4515e+01	72.00%
2	10240	2*2	128	1	12.80	5.8132e+01	62.43%
3	10240	2*2	160	1	14.19	5.0474e+01	56.33%
4	10240	2*2	96	0	11.54	6.2030e+01	69.23%
5	10240	2*2	128	0	15.31	4.6761e+01	52.19%
6	10240	2*2	160	0	16.91	4.2343e+01	47.26%

PMAP process mapping=1时效果最好

NB=96时的效果最好

N=12288

ID	N	P*Q	NB	PMAP process mapping	耗时/s	测试结果/Gflops	理论实际比值
1	12288	2*2	96	1	19.83	6.2392e+01	69.63%
2	12288	2*2	128	1	25.31	5.8132e+01	54.55%
3	12288	2*2	160	1	20.94	5.9075e+01	65.93%
4	12288	2*2	96	0	26.67	6.1447e+01	68.58%
5	12288	2*2	128	0	23.10	5.3557e+01	59.77%
6	12288	2*2	160	0	26.67	4.6387e+01	51.77%

PMAP process mapping=1时效果最好

NB=96时的效果最好

需要注意：

经过多轮测试,只改变NB值,理论实际比值在这一组测试中并非递减,存疑

小结

- PMAPprocess mapping = 1
- P*Q = 2
- NB范围中值更新为96
- N范围中值更新为10240,缩小区间

第三次测试

新的测试方法

- 使用命令 `hpl-test > result.txt && cat result.txt | grep WR00L2L2` ,其中 `hpl-test` 对应上文的测试命令
- 对于9组测试,均取10次结果的平均值作为数据.在这里我使用了word的文本转表格、再将表格导入excel,excel自动计算十次答案的平均值

数据结果

ID	N	P*Q	NB	平均耗时/s	平均测试结果/Gflops	理论实际比值
1	9216	2*2	80	9.221	5.7660E+01	64.35%
2	9216	2*2	96	9.834	5.3615E+01	59.84%
3	9216	2*2	112	10.26	5.1499E+01	57.48%
4	10240	2*2	80	16.027	4.4861E+01	50.07%
5	10240	2*2	96	13.957	5.1972E+01	58.00%
6	10240	2*2	112	14.394	5.0079E+01	55.89%
7	11264	2*2	80	19.781	4.8580E+01	54.22%
8	11264	2*2	96	19.373	4.9618E+01	55.38%
9	11264	2*2	112	19.591	4.9121E+01	54.82%

小结

- N范围以9216为基础值
- NB范围以80为基础值
- 在测试过程中,发现由于电脑散热原因部分测试结果数据整体较低,但个人认为不影响理论实际比值
- 在测试过程中,单次测试获得的最高理论实际比值为71.09%,发生在一号数据中

第四次测试

新的测试方法

- 关闭NextSSH的数据监看
- 关闭vscode-server连接
- 编写shell脚本自动化实现测试
- 只取十次测试中最好结果(考虑发热问题)

shell脚本

```
#!/bin/bash
cat /dev/null > result.txt
for i in {1..10};
do
    echo "Solving process : $i"
    mpirun -np 8 ./xhpl | grep WR00L2L2 >> result.txt
done
```

测试参数

```
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)
3            # of problems sizes (N)
8704 9216 9728          Ns
3            # of NBs
72 80 88    NBs
0            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
0            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2            NBMINs (≥ 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            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)
```

数据结果(只取各个参数最好成绩)

ID	N	P*Q	NB	耗时/s	测试结果/Gflops	理论实际比值
1	8704	2*2	72	7.19	6.1133e+01	68.23%
2	8704	2*2	80	7.58	5.8030e+01	64.77%
3	8704	2*2	88	8.38	5.2490e+01	58.58%
4	9216	2*2	72	9.10	5.7354e+01	64.01%
5	9216	2*2	80	9.43	5.5361e+01	61.79%
6	9216	2*2	88	10.78	4.8403e+01	54.02%
7	9728	2*2	72	12.92	4.7531e+01	53.05%
8	9728	2*2	80	10.41	5.8965e+01	65.81%
9	9728	2*2	88	10.97	5.5965e+01	62.46%

性能损失

根据 `lm-sensors` 提供的数据,测试结束一段时间后cpu温度仍然在50多度

```
Adapter: ISA adapter
Package id 0:  +55.0°C  (high = +84.0°C, crit = +100.0°C)
Core 0:        +51.0°C  (high = +84.0°C, crit = +100.0°C)
Core 1:        +50.0°C  (high = +84.0°C, crit = +100.0°C)
Core 2:        +52.0°C  (high = +84.0°C, crit = +100.0°C)
Core 3:        +50.0°C  (high = +84.0°C, crit = +100.0°C)
```

应该是撞温度墙了,导致频率下降,所以后续测试并不准确,各个数据之间拉不开差距

小结

- 发现一些问题所在:温度、功耗、连续测试变为压测
- 后面测试不应该安排这么紧密

其他

本次测试数据保留在`result.txt`里,可以参考

第五次测试

新的测试方法

- 利用 `cpufrequtils` 开启cpu高性能模式
- 将电脑移动至开阔地带
- 利用风扇辅助散热
- 只测试4次,每次只测试一组数据,取最高值
- 监控CPU温度,到45度左右在开始测试

shell脚本

```
#!/bin/bash
cat /dev/null > result.txt
echo "launch performance mode"
for i in {0..7}
do
    sudo cpufreq-set -g performance -c $i
done
for i in {1..4};
do
    echo "Solving process : $i"
    mpirun -np 8 ./xhpl | grep WR00L2L2 >> result.txt
done
```

数据

ID	N	P*Q	NB	耗时/s	测试结果/Gflops	理论实际比值
1	10240	2*2	96	11.28	6.3468e+01	70.83%
2	10240	2*2	128	11.53	6.2105e+01	69.31%
3	9216	2*2	96	8.94	5.8401e+01	65.18%
4	9216	2*2	128	8.42	6.1987e+01	69.18%
5	8704	2*2	96	6.69	6.5727e+01	73.36%
6	8704	2*2	128	7.63	5.7631e+01	64.32%

结论

有突破点了,1号和5号数据都超过了70!还刷新到了73.36%!

第六次测试

从内存确定N的值

根据 `free -g` 获取到的数据

```
           total        used        free      shared  buff/cache   available
内存:         15           1          10           0           3          13
交换:          1           0           1
```

我们注意到实际上可用空间是10g,那么重新计算的出的N的值为35895左右,离他最近的384的倍数是35712

分别试试这两个

N=35895

可以看见这里面的物理内存剩余稳定在3.7g可用,说明还没压榨完,可以考虑继续提高n的值

测试得出Gflops = 6.1495e+01, 理论实际比值为68.63%

N=38400

考虑到还有内存剩余,我们尝试N=38400

测试得出:Gflops = 6.1320e+01, 理论实际比值为68.44%

总结

最好成绩

实际/理论*100% = 73.36%

参数如下

```
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)
1            # of problems sizes (N)
8704         Ns
1            # of NBs
96           NBs
0            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
0            PFACTs (0=left, 1=Crout, 2=Right)
```

```
1            # of recursive stopping criterium
2            NBMINs (≥ 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            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)
```

优化方向

- 经过前期摸爬滚打,优化方向大致在N、NB、P、Q上
- 需要考虑CPU散热问题
- 通常以2的整数倍效果好,尤其是2的幂