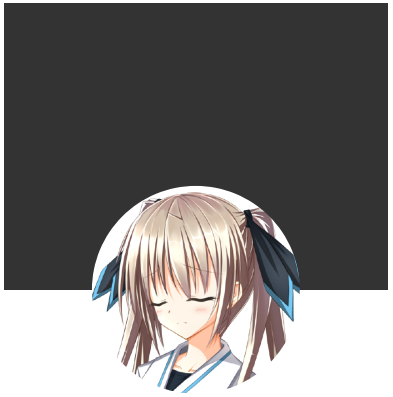


Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

奇技淫巧：非对称 MPI + OpenMP 并行

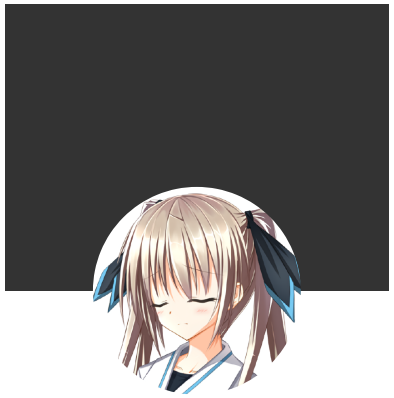
MPI + OpenMP/multi-threading 是如今大部分高性能计算程序的并行方式。一般情况下，各 MPI 进程地位对等（对称），可用的 CPU 资源和并行出来的线程数也相同。今天讲一个奇技淫巧：如何进行非对称 MPI + OpenMP 并行。

最近干活的时候遇到一个情况：一个纯 MPI 程序中有一个 eigen solver 只在 rank 0 上计算，因为矩阵尺寸不大（<2000），用 ScaLAPACK 或者其他 MPI 库并行得不偿失，然而单线程求解又有点慢。因此甲方想在这一步的时候让 rank 0 用多线程来做 eigen solver, 其他 MPI 进程休眠等待 rank 0 完成计算。

乍一看这下，这个需求似乎很好实现，只要在 rank 0 上手动将线程数设大一点然后调 LAPACK 就行了。这就是我碰到的第一个钉子：有些高性能的 MPI 库默认做了自动绑核以提高性能（比如 Intel MPI 默认 `I_MPI_PIN=1`，MVAPICH2 默认 `MV2_ENABLE_AFFINITY=1`），每个 MPI 进程只能在指定的几个核心上执行。不关掉 MPI 的自动绑核，rank 0 就没办法用属于其他 MPI 进程的核心。但是关掉了自动绑核，其他部分的性能可能会严重下降。因此，我们首先需要手动进行绑核：rank 0 绑到所有核心上，其他进程只绑到一个物理核心上。

现在的 CPU 大部分有超线程，手动绑核要求不能把两个 MPI 进程绑到同一个物理核心上。Linux 上同一个物理核心的超线程核心编号一般都是不连续的，相差的值就是总的物理核心数；但有些时候同一个物理核心的超线程核心编号也可能是连续的。我不想引入别的库来解决这个问题，所以自己撸了一个简单粗暴的检测方法：读 `/sys/devices/system/cpu/cpu0/topology/thread_siblings_list` 和 `/sys/devices/system/cpu/cpu1/topology/thread_siblings_list` 这两个文件，看一下 CPU0 和 CPU1 的第一个核心编号差了多少。有个物理核心编号的距离以后，我们就可以通过 `CPU_SET_S()` 和 `sched_setaffinity()` 来手动绑核了。

最后一个小坑是 rank 0 以外的 MPI 进程的休眠。`MPI_Barrier()` 或者 `MPI_Ibarrier()` + `MPI_Wait()` 的组合都会阻塞 CPU，不断查询其他进程是否到达同步点。一个奇技淫巧是每隔一段时间用 `MPI_Test()` 代替 `MPI_Wait()` 来检测同步是否完成，如果没完成就用 `usleep()` 继续休眠一段时间。这样，非 rank 0 进程休眠的时候就不会阻碍 rank 0 使用属于它们的核心了。



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

下面是样例测试脚本：

```
# Compile the program
mpiicc -Wall -g -O3 -xHost -mkl -qopenmp -std=c99 -o test_dsyev_omp test_dsyev_omp.c

# Disable Intel MPI auto pinning
export I_MPI_PIN=0

# Test the program with different available threads on rank 0
for i in 1 2 4 8; do
    export NTHREADS=$i
    mpirun -np 64 ./test_dsyev_omp
done
```

下面是样例测试代码：

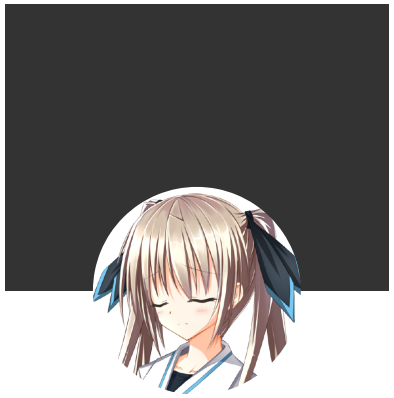
```
#define _XOPEN_SOURCE 500 // For srand48(), drand48(), usleep()

// For sched_setaffinity
#define  _GNU_SOURCE
#include <sched.h>
#include <unistd.h>  // Also for usleep()

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <omp.h>
#include <mkl.h>
#include <mpi.h>

void test_dgemm(const int rank, const int n)
{
    mkl_set_dynamic(0);
    mkl_set_num_threads(1);

    size_t mat_msize = sizeof(double) * n * n;
    double *A = (double*) malloc(mat_msize);
    double *B = (double*) malloc(mat_msize);
    double *C = (double*) malloc(mat_msize);
    for (int i = 0; i < n * n; i++)
    {
        A[i] = drand48();
        B[i] = drand48();
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

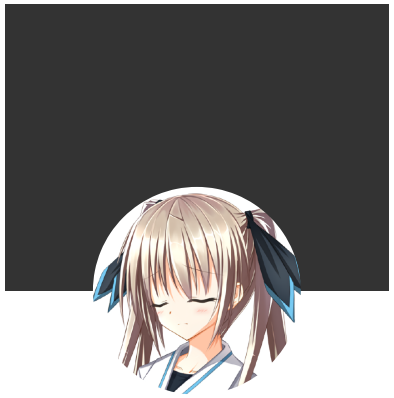
Home

Archives

About

SiteXC

```
29      }
30      cblas_dgemm(
31          CblasRowMajor, CblasNoTrans, CblasNoTrans, n, n, n,
32          1.0, A, n, B, n, 0.0, C, n
33      );
34      MPI_Barrier(MPI_COMM_WORLD);
35
36      double st = omp_get_wtime();
37      for (int i = 0; i < 5; i++)
38      {
39          cblas_dgemm(
40              CblasRowMajor, CblasNoTrans, CblasNoTrans, n, n, n,
41              1.0, A, n, B, n, 0.0, C, n
42          );
43      }
44      double ut = (omp_get_wtime() - st) * 1000.0 / 5.0;
45      printf("Rank %2d single thread %d*%d*%d dgemm used %.3lf ms\n", rank, n, n, n, ut);
46
47      mkl_set_dynamic(1);
48  }
49
50  void test_dsyev(const int rank, const int n)
51  {
52      int my_nthreads = 1;
53      char *env_ntheads = getenv("NTHREADS");
54      if (env_ntheads != NULL) my_nthreads = atoi(env_ntheads);
55      if (my_nthreads < 1) my_nthreads = 1;
56
57      int save = mkl_get_max_threads();
58      mkl_set_dynamic(0);
59      mkl_set_num_threads(my_nthreads);
60
61      size_t mat_msize = sizeof(double) * n * n;
62      double *A  = (double*) malloc(mat_msize);
63      double *B  = (double*) malloc(mat_msize);
64      double *A0 = (double*) malloc(mat_msize);
65
66      for (int i = 0; i < n; i++)
67      {
68          for (int j = 0; j < i; j++)
69          {
70              double val = drand48();
71              A[i * n + j] = val;
72              A[j * n + i] = val;
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

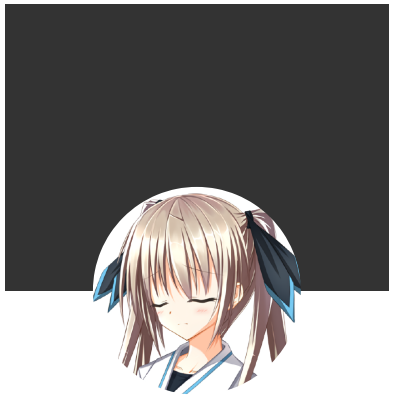
Home

Archives

About

SiteXC

```
73          }
74      }
75
76      memcpy(A, A0, mat_msize);
77      LAPACKE_dsyev(LAPACK_ROW_MAJOR, 'V', 'U', n, A, n, B);
78      for (int i = 0; i < 5; i++)
79      {
80          memcpy(A, A0, mat_msize);
81          double st = omp_get_wtime();
82          LAPACKE_dsyev(LAPACK_ROW_MAJOR, 'V', 'U', n, A, n, B);
83          double ut = (omp_get_wtime() - st) * 1000.0;
84          printf("Rank %d LAPACKE_dsyev use %.3lf ms\n", rank, ut);
85      }
86
87      printf("\n");
88
89      free(A);
90      free(B);
91      free(A0);
92
93      mkl_set_dynamic(1);
94      mkl_set_num_threads(1);
95  }
96
97  int get_phys_core_id_dist()
98  {
99      int core0_id, core1_id;
100     FILE *inf0 = fopen("/sys/devices/system/cpu/cpu0/topology/thread_siblings_list", "r");
101     FILE *inf1 = fopen("/sys/devices/system/cpu/cpu1/topology/thread_siblings_list", "r");
102     fscanf(inf0, "%d,", &core0_id);
103     fscanf(inf1, "%d,", &core1_id);
104     fclose(inf0);
105     fclose(inf1);
106     return (core1_id - core0_id);
107 }
108
109 // This function is based on Kent Milfeld's <milfeld@tacc.utexas.edu> code
110 int set_core_affinity(const int ncore, const int ntarget, const int *cores)
111 {
112     cpu_set_t *mask = CPU_ALLOC(ncore);
113     size_t size = CPU_ALLOC_SIZE(ncore);
114     CPU_ZERO_S(size, mask);
115
116     if (ntarget < 0)
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

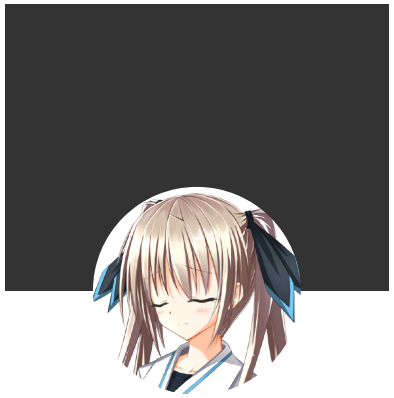
Home

Archives

About

SiteXC

```
117     {
118         // Allow to run on all cores
119         for (int core_id = 0; core_id < ncore; core_id++)
120             CPU_SET_S(core_id, size, mask);
121     } else {
122         for (int i = 0; i < ntarget; i++)
123         {
124             int core_id = cores[i];
125             CPU_SET_S(core_id, size, mask);
126         }
127     }
128 
129     return sched_setaffinity((pid_t) 0, size, mask);
130 }
131 
132 void bind_rank0_to_all_cores(const int nproc, const int rank)
133 {
134     int phys_core_id_dist = get_phys_core_id_dist();
135     if (rank == 0)
136     {
137         int *cores = (int*) malloc(sizeof(int) * nproc);
138         for (int i = 0; i < nproc; i++)
139             cores[i] = i * phys_core_id_dist;
140         set_core_affinity(nproc, nproc, cores);
141         free(cores);
142     } else {
143         int core_id = rank * phys_core_id_dist;
144         set_core_affinity(nproc, 1, &core_id);
145     }
146 }
147 
148 void MPI_Wait_nonblocking(MPI_Request *req, const int microseconds)
149 {
150     int flag;
151     MPI_Status status;
152     MPI_Test(req, &flag, &status);
153     while (!flag)
154     {
155         usleep(microseconds);
156         MPI_Test(req, &flag, &status);
157     }
158 }
159 
160 int main(int argc, char **argv)
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

```
161 {
162     MPI_Init(&argc, &argv);
163 
164     int rank, nproc;
165     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
166     MPI_Comm_size(MPI_COMM_WORLD, &nproc);
167 
168     MPI_Barrier(MPI_COMM_WORLD);
169 
170     int n = 1000;
171     if (argc >= 2) n = atoi(argv[1]);
172     if (rank == 0) printf("Matrix size = %d\n", n);
173 
174     srand48(114514 + rank);
175 
176     bind_rank0_to_all_cores(nproc, rank);
177 
178     test_dgemm(rank, 1000);
179     MPI_Barrier(MPI_COMM_WORLD);
180 
181     if (rank == 0) test_dsyev(rank, n);
182 
183     MPI_Request req;
184     MPI_Ibarrier(MPI_COMM_WORLD, &req);
185     MPI_Wait_nonblocking(&req, 10000);
186 
187     MPI_Finalize();
188 }
```

顺带吐槽一句， `LAPACKE_dsyev()` 的并行扩展性真的不行……

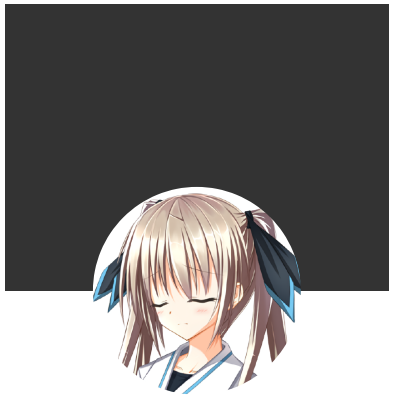
发布于 2019-09-02

tags: { C } { MPI } { OpenMP } { Affinity }

1 comment

Anonymous

Leave a comment



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

ⓘ Markdown is supported

Login with GitHub Preview

LiangJiuyang commented 12 months ago

博主你好，今天看到你的博客学到了很多小(奇)技(淫)巧，感谢！

我现在的博士工作主要是数学结合高性能计算的快速算法研究，经常感觉到自己自学的并行计算离真正前沿的“高性能”还有很大的差距。期待以后有机会和你交流学习！