University of Zurich[UZH]

High Performance Computing Lecture 8

Douglas Potter
Jozef Bucko
Noah Kubli

Compiler Suites

Cray Compiler (PrgEnv-cray)	Available on Cray Computers
GNU Compiler (PrgEnv-gnu)	Free for everyone
Intel Compiler (PrgEnv-intel)	Free for Students
Portland (PrgEnv-pgi)	Important for OpenACC (GPU)
Clang (Apple & Cray)	Open Source & GNU Compatible
Visual Studio (Windows)	Not usually High Performance

Compile with the Cray “wrappers”

```
dpotter@daint106:~> cc --version
Cray clang version 11.0.0  (1563bb12fb3eabb03515d4cc4aeaedd757aea56a)
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir: /opt/cray/pe/cce/11.0.0/cce-clang/x86_64/share/../bin
dpotter@daint106:~> CC --version
Cray clang version 11.0.0  (1563bb12fb3eabb03515d4cc4aeaedd757aea56a)
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir: /opt/cray/pe/cce/11.0.0/cce-clang/x86_64/share/../bin
dpotter@daint106:~> ftn --version
Cray Fortran : Version 11.0.0
dpotter@daint106:~>
```

Compile with the Cray “wrappers”

```
dpotter@daint106:~> module swap PrgEnv-cray PrgEnv-gnu
dpotter@daint106:~> cc --version
gcc (GCC) 10.1.0 20200507 (Cray Inc.)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

dpotter@daint106:~> CC --version
g++ (GCC) 10.1.0 20200507 (Cray Inc.)
…

dpotter@daint106:~> ftn --version
GNU Fortran (GCC) 10.1.0 20200507 (Cray Inc.)
…
```

Compile with the Cray “wrappers”

```
dpotter@daint106:~> module swap PrgEnv-gnu PrgEnv-pgi
dpotter@daint106:~> cc --version

pgcc (aka pgcc18) 20.1-0 LLVM 64-bit target on x86-64 Linux -tp haswell-64
PGI Compilers and Tools
Copyright (c) 2020, NVIDIA CORPORATION.  All rights reserved.
dpotter@daint106:~> CC --version

pgc++ 20.1-0 LLVM 64-bit target on x86-64 Linux -tp haswell-64
…
dpotter@daint106:~> ftn --version

pgf90 20.1-0 LLVM 64-bit target on x86-64 Linux -tp haswell-64
…
```

On Some Systems you use the MPI Wrappers

```
ubuntu@dpotte-gpu:~$ mpicc --version
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ubuntu@dpotte-gpu:~$ mpicxx --version
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
…

ubuntu@dpotte-gpu:~$ mpif90 --version
GNU Fortran (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
…
```

MPI compilation

Compilation directives:

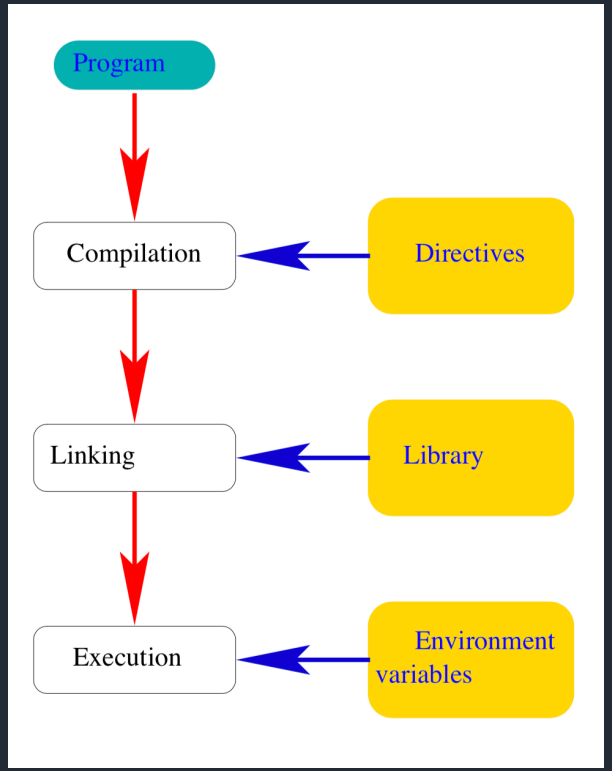
- You need to include “mpi.h” to call MPI functions.
  - This tells the compiler the parameters and their type.
- You need to tell the compiler where it can find “mpi.h” (`-I`).

Linking:

- You need to link against the MPI library (`-l`).
- You need to tell the linker where to find this library (`-L`).

Wrappers:

- The compiler wrapper (**`cc`** or **`mpicc`**) usually adds the necessary flags for you (`-L`, `-l`, and `-I`).



man gcc

-llibrary

Search the library named library when linking.

-Ldir

Add directory dir to the list of directories to be searched for -l.

-I dir

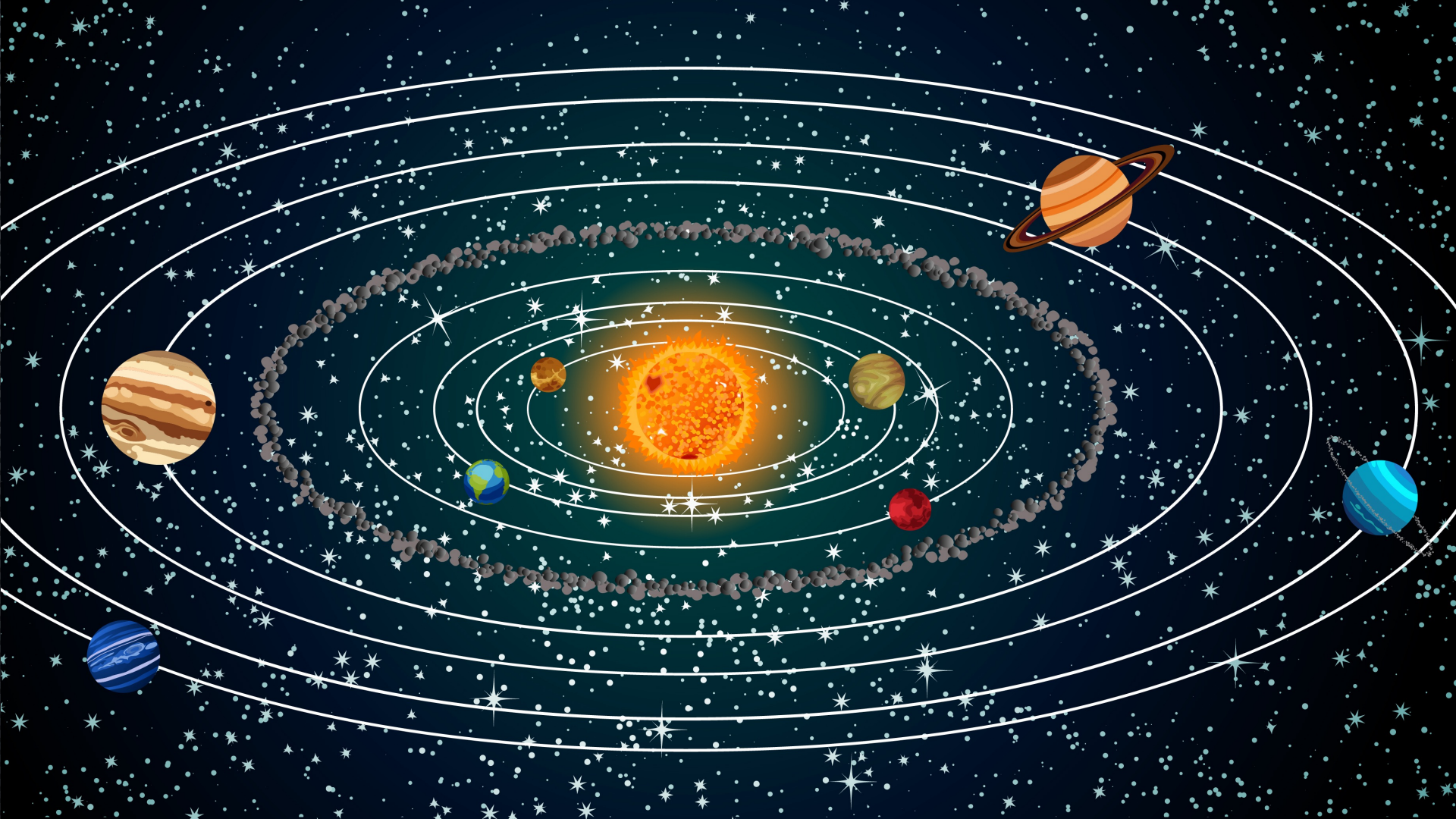
Add the directory dir to the list of directories to be searched for header files during preprocessing.

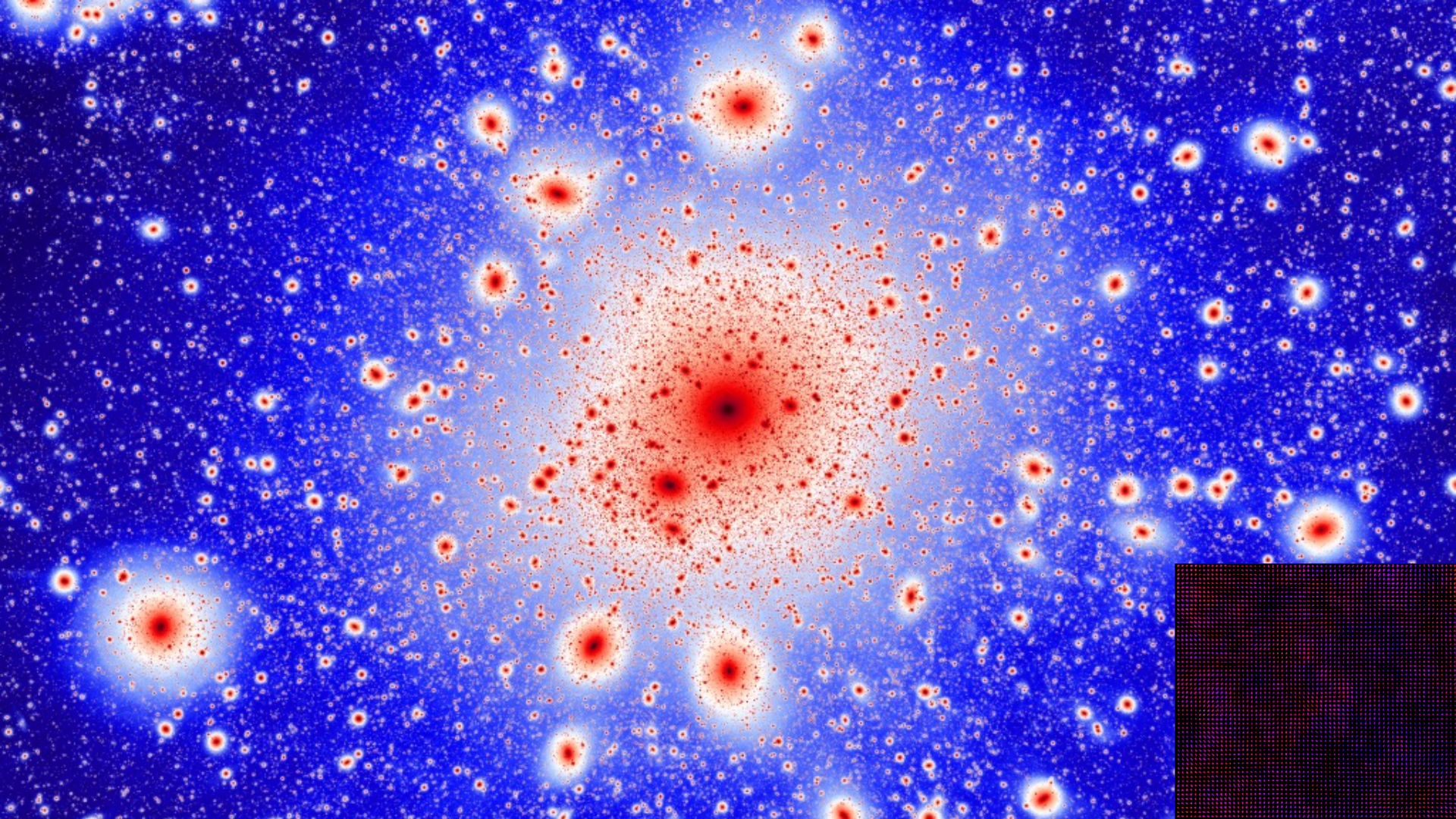
Example:

```
gcc -L/path/to/mpi/lib -lmpi -I/path/to/mpi/include -O3 -o mpitest mpitest.c
```

Gravity

The n-body method





EQUATIONS OF MOTION

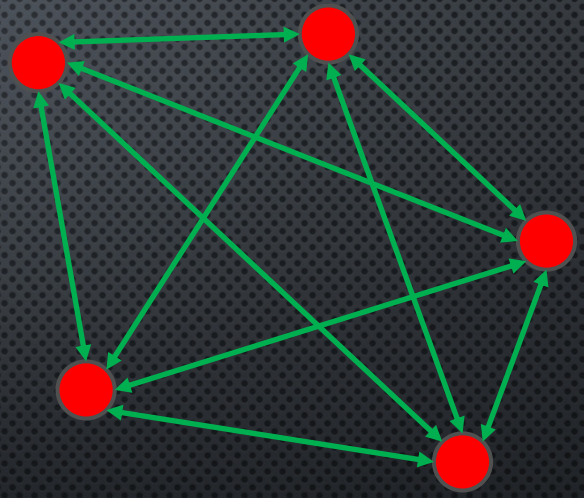
$$\boldsymbol{a} = \frac{d\boldsymbol{v}_i}{dt} = \sum_{j \neq i}^{N} G m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3}$$

EQUATIONS OF MOTION

$$\boldsymbol{a} = \frac{d\boldsymbol{v}_i}{dt} = \sum_{j \neq i}^{N} G m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3}$$

METHOD

PAIRWISE INTERACTIONS



The n-body code

```
#include <vector>
struct particle {
    float x, y, z;    // position
    float vx, vy, vz; // velocity
    float ax, ay, az; // acceleration
};
typedef std::vector<particle> particles;
int main(int argc, char *argv[]) {
    int N=20'000;    // number of particles
    particles plist; // vector of particles
    ic(plist,N);     // initialize starting position/velocity
    forces(plist);   // calculate the forces
    return 0;
}
```

Random Initial Conditions

```
#include <random>

// Initial conditions
void ic(particles &plist, int n) {
    std::random_device rd;  //Will be used to obtain a seed for the random number engine
    std::mt19937 gen(rd()); //Standard mersenne_twister_engine seeded with rd()
    std::uniform_real_distribution<float> dis(0.0, 1.0);

    plist.clear();       // Remove all particles
    plist.reserve(n);    // Make room for "n" particle
    for( auto i=0; i<n; ++i) {
        particle p { dis(gen),dis(gen),dis(gen),0,0,0,0,0,0 };
        plist.push_back(p);
    }
}
```

Force Calculations

```
void forces(particles &plist) {
    int n = plist.size();
    for(int i=0; i<n; ++i) {                    // We want to calculate the force on all particles
        plist[i].ax = plist[i].ay = plist[i].az = 0;  // start with zero acceleration
        for(int j=0; j<n; ++j) {                          // Depends on all other particles
            if (i==j) continue; // Skip self interaction
            auto dx = plist[j].x - plist[i].x;
            auto dy = plist[j].y - plist[i].y;
            auto dz = plist[j].z - plist[i].z;
            auto r = sqrt(dx*dx + dy*dy + dz*dz);
            auto ir3 = 1 / (r*r*r);
            plist[i].ax += dx * ir3;
            plist[i].ay += dy * ir3;
            plist[i].az += dz * ir3;
        }
    }
}
```

$$\boldsymbol{a} = \frac{d\boldsymbol{v}_i}{dt} = \sum_{j \neq i}^{N} \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3}$$

How does this perform? No idea!

```
dpotter@daint106:~/hpc1a/Y7> CC --version
g++ (GCC) 10.1.0 20200507 (Cray Inc.)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

dpotter@daint106:~/hpc1a/Y7> CC -o nbody nbody.cxx
dpotter@daint106:~/hpc1a/Y7> ./nbody
dpotter@daint106:~/hpc1a/Y7>
```

The “time” command

```
dpotter@daint106:~/hpc1a/Y7> time ./nbody

real    0m20.061s
user    0m20.006s
sys     0m0.030s
dpotter@daint106:~/hpc1a/Y7> /usr/bin/time ps -fu dpotter f
UID        PID  PPID  C STIME TTY      STAT   TIME CMD
dpotter  24319 23916  0 10:08 ?        S      0:00 sshd: dpotter@pts/46
dpotter  24320 24319  0 10:08 pts/46   Ss     0:00  \_ -bash
dpotter  16118 24320  0 10:59 pts/46   S+     0:00      \_ /usr/bin/time ps -fu dpotter f
dpotter  16119 16118  0 10:59 pts/46   R+     0:00          \_ ps -fu dpotter f
0.01user 0.03system 0:00.06elapsed 85%CPU (0avgtext+0avgdata 5484maxresident)k
0inputs+0outputs (0major+635minor)pagefaults 0swaps
```

Oops, forgot to optimize

```
dpotter@daint106:~/hpc1a/Y7> CC -O3 -o nbody nbody.cxx
dpotter@daint106:~/hpc1a/Y7> time ./nbody

real    0m4.335s
user    0m4.303s
sys     0m0.021s
dpotter@daint106:~/hpc1a/Y7> CC -O3 -ffast-math -o nbody nbody.cxx
dpotter@daint106:~/hpc1a/Y7> time ./nbody

real    0m3.976s
user    0m3.928s
sys     0m0.037s
```

New compiler flag “-g” (man gcc)

GCC(1) GNU GCC(1)

NAME

gcc - GNU project C and C++ compiler

…

Options for Debugging Your Program

To tell GCC to emit extra information for use by a debugger, in almost all cases **you need only to add -g** to your other options.

GCC allows you to use -g with -O. **The shortcuts taken by optimized code may occasionally be surprising:** some variables you declared may not exist at all; flow of control may briefly move where you did not expect it; some statements may not be executed because they compute constant results or their values are already at hand; some statements may execute in different places because they have been moved out of loops. Nevertheless it is possible to debug optimized output. This makes it reasonable to use the optimizer for programs that might have bugs.

If you are not using some other optimization option, consider using -Og with -g. With no -O option at all, some compiler passes that collect information useful for debugging do not run at all, so that -Og may result in a better debugging experience.

-g Produce debugging information in the operating system's native format (stabs, COFF, XCOFF, or DWARF). GDB can work with this debugging information.

Effect of “-g” with gcc

```
dpotter@daint102:~/hpc1a/Y7> CC -g -O3 -ffast-math -o nbody nbody.cxx
dpotter@daint102:~/hpc1a/Y7> ls -l nbody
-rwxr-xr-x 1 dpotter uzh0 126464 Mar 22 10:53 nbody
dpotter@daint102:~/hpc1a/Y7> strip nbody
dpotter@daint102:~/hpc1a/Y7> ls -l nbody
-rwxr-xr-x 1 dpotter uzh0 17016 Mar 22 10:54 nbody

dpotter@daint102:~/hpc1a/Y7> CC -O3 -ffast-math -o nbody nbody.cxx
dpotter@daint102:~/hpc1a/Y7> ls -l nbody
-rwxr-xr-x 1 dpotter uzh0 25168 Mar 22 10:53 nbody
dpotter@daint102:~/hpc1a/Y7> strip nbody
dpotter@daint102:~/hpc1a/Y7> ls -l nbody
-rwxr-xr-x 1 dpotter uzh0 17016 Mar 22 10:55 nbody
```

Effect of “-og” with gcc

```
dpotter@daint106:~/hpc1a/Y7> CC -O3 -ffast-math -o nbody nbody.cxx
dpotter@daint106:~/hpc1a/Y7> time ./nbody

real    0m3.976s
user    0m3.928s
sys     0m0.037s
dpotter@daint106:~/hpc1a/Y7> CC -Og -ffast-math -o nbody nbody.cxx
dpotter@daint106:~/hpc1a/Y7> time ./nbody

real    0m4.862s
user    0m4.819s
sys     0m0.024s
```

Using Craypat – must use queue system

```
dpotter@daint106:~/hpc1a/Y7> module load perftools-lite

dpotter@daint106:~/hpc1a/Y7> CC -Og -ffast-math -o nbody nbody.cxx

WARNING: PerfTools is saving object files from a temporary directory into directory
'/users/dpotter/.craypat/nbody/18328'

INFO: creating the PerfTools-instrumented executable 'nbody' (lite-samples) ...OK

dpotter@daint106:~/hpc1a/Y7> time ./nbody

pat[FATAL][0]: PMI_Init failed


real    0m2.144s

user    0m0.022s

sys     0m0.084s
```

Using Craypat

```
dpotter@daint106:~/hpc1a/Y7> cat nbody.job
#!/bin/bash -l
#SBATCH --account=uzg2
#SBATCH --job-name=nbody
#SBATCH --time=00:5:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-core=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --partition=debug
#SBATCH --constraint=gpu

srun nbody
dpotter@daint106:~/hpc1a/Y7> sbatch nbody.job
Submitted batch job 30052444
```

Using Craypat

```
CrayPat/X:  Version 20.10.0 Revision 7ec62de47  09/16/20 16:54:26

##################################################################
#                                                                #
#            CrayPat-lite Performance Statistics                 #
#                                                                #
##################################################################

CrayPat/X:  Version 20.10.0 Revision 7ec62de47  09/16/20 16:54:26
Experiment:                   lite  lite-samples
Number of PEs (MPI ranks):       1
Numbers of PEs per Node:         1
Numbers of Threads per PE:       1
Number of Cores per Socket:     12
Execution start time:  Mon Mar 22 10:30:17 2021
System name and speed:  nid02356  2.601 GHz (nominal)
Intel Haswell    CPU  Family:  6  Model: 63  Stepping:  2
```

```
DRAM:  64 GiB DDR4-2400 on 2.6 GHz nodes


Avg Process Time:          4.38 secs
High Memory:               22.0 MiBytes 22.0 MiBytes per PE
Observed CPU clock boost: 126.5 %
Percent cycles stalled:    59.7 %
Instr per Cycle:           1.38
```

Using Craypat

```
Notes for table 1:

  This table shows functions that have significant exclusive sample
    hits, averaged across ranks.
  For further explanation, see the "General table notes" below,
    or use:  pat_report -v -O samp_profile ...
```

```
Table 1:  Profile by Function

  Samp% |  Samp | Imb. |  Imb. | Group
        |       | Samp | Samp% |  Function=[MAX10]

 100.0% | 433.0 |   -- |    -- | Total
|-----------------------------------------------------------
| 100.0% | 433.0 |   -- |    -- | USER
||----------------------------------------------------------
||  83.1% | 360.0 |   -- |    -- | forces
||  16.9% |  73.0 |   -- |    -- | std::vector<>::operator[]
|===========================================================
```

Using Craypat

```
Notes for table 2:

  This table shows functions, and line numbers within functions, that
    have significant exclusive sample hits, averaged across ranks.
  For further explanation, see the "General table notes" below,
    or use:  pat_report -v -O samp_profile+src ...

Table 2:  Profile by Group, Function, and Line

  Samp% |  Samp | Imb. |  Imb. | Group
        |       | Samp | Samp% |  Function=[MAX10]
        |       |      |       |   Source
        |       |      |       |    Line
```

```
 100.0% | 433.0 |   -- |    -- | Total
|-----------------------------------------------------------
| 100.0% | 433.0 |   -- |    -- | USER
||----------------------------------------------------------
||  83.1% | 360.0 |   -- |    -- | forces
|||---------------------------------------------------------
3||  80.1% | 347.0 |   -- |    -- | dpotter/hpc1a/Y7/nbody.cxx
||||--------------------------------------------------------
4|||   2.8% |  12.0 |   -- |    -- | line.32
4|||   1.8% |   8.0 |   -- |    -- | line.36
4|||   4.4% |  19.0 |   -- |    -- | line.37
4|||  65.4% | 283.0 |   -- |    -- | line.39
4|||   3.2% |  14.0 |   -- |    -- | line.40
4|||   2.5% |  11.0 |   -- |    -- | line.41
```

Using Craypat

```
dpotter@daint106:~/hpc1a/Y7> cat -n nbody.cxx | sed -n 28,43p
    28      void forces(particles &plist) {
    29          int n = plist.size();
    30          for(int i=0; i<n; ++i) {
    31                  plist[i].ax = plist[i].ay = plist[i].az = 0;
    32              for(int j=0; j<n; ++j) {
    33                      if (i==j) continue;
    34                      auto dx = plist[j].x - plist[i].x;
    35                      auto dy = plist[j].y - plist[i].y;
    36                      auto dz = plist[j].z - plist[i].z;
    37                      auto r = sqrt(dx*dx + dy*dy + dz*dz);          4|||   4.4% |  19.0 |   -- |    -- | line.37
    38                      auto ir3 = 1 / (r*r*r);
    39                      plist[i].ax += dx * ir3;                       4|||  65.4% | 283.0 |   -- |    -- | line.39
    40                      plist[i].ay += dy * ir3;                       4|||   3.2% |  14.0 |   -- |    -- | line.40
    41                      plist[i].az += dz * ir3;
    42                  }
    43          }
```

Sampling

PAT_RT_SAMPLING_INTERVAL

Specifies the interval, in microseconds, at which the instrumented executable is sampled.

To specify a random interval, use the following format:

```
lower-bound,upper-bound[,seed]
```

After a sample is captured, the interval used for the next sampling interval is generated using rand(3) and will be between lower-bound and upper-bound. The initial seed (seed) for the sequence of random numbers is optional. See srand(3) for more information.

This environment variable affects sampling experiments. It can also be used to control trace-enhanced sampling experiments, provided the program is instrumented for tracing but the PAT_RT_EXPERIMENT environment variable is used to specify a sampling-type experiment, and subject to the PAT_RT_SAMPLING_MODE environment variable setting.

Default: 10000 (microseconds)

Using Craypat without optimization “-O0”

```
34        auto dx = plist[j].x - plist[i].x;          4|||   3.1% |    50.0 |   -- |    -- | line.34
35        auto dy = plist[j].y - plist[i].y;          4|||   3.5% |    56.0 |   -- |    -- | line.35
36        auto dz = plist[j].z - plist[i].z;          4|||   3.6% |    59.0 |   -- |    -- | line.36
37        auto r = sqrt(dx*dx + dy*dy + dz*dz);       4|||   6.0% |    97.0 |   -- |    -- | line.37
38        auto ir3 = 1 / (r*r*r);                     4|||  25.9% |   419.0 |   -- |    -- | line.38
39        plist[i].ax += dx * ir3;                    4|||  11.2% |   181.0 |   -- |    -- | line.39
40        plist[i].ay += dy * ir3;                    4|||   2.8% |    46.0 |   -- |    -- | line.40
41        plist[i].az += dz * ir3;                    4|||   3.8% |    62.0 |   -- |    -- | line.41
```

Craypat summary

- Load “perftools-lite” module (or “perftools” for advanced)
- Compile with “-g” for debug information
- Consider effect of optimization
- Run using the batch queue system

app2

```
dpotter@daint105:~/hpc1a/Y7> ls -ld nbody+13229-3508s/
drwxr-x--- 5 dpotter uzh0 8192 Mar 22 11:31 nbody+13229-3508s/
dpotter@daint105:~/hpc1a/Y7> ls -l nbody+13229-3508s/
total 171
drwxr-x--- 2 dpotter uzh0   4096 Mar 22 11:31 ap2-files
-rw-r--r-- 1 dpotter uzh0 167936 Mar 22 11:31 index.ap2
drwxr-x--- 2 dpotter uzh0   4096 Mar 22 11:31 rpt-files
drwxr-x--- 2 dpotter uzh0   4096 Mar 22 11:31 xf-files
dpotter@daint105:~/hpc1a/Y7> app2 nbody+13229-3508s/
```

University of Zurich[UZH]

app2

nbody+13229-3508s

File Compare View Help

About Apprentice2 | nbody+13229-3508s

Overview

Profile

CPU

Computation
100.00%

Function/Region Profile

100.0% = forces

Memory Utilization

Process HiMem = 22.0 MiBytes

Load Imbalance
(samples)

Energy Usage

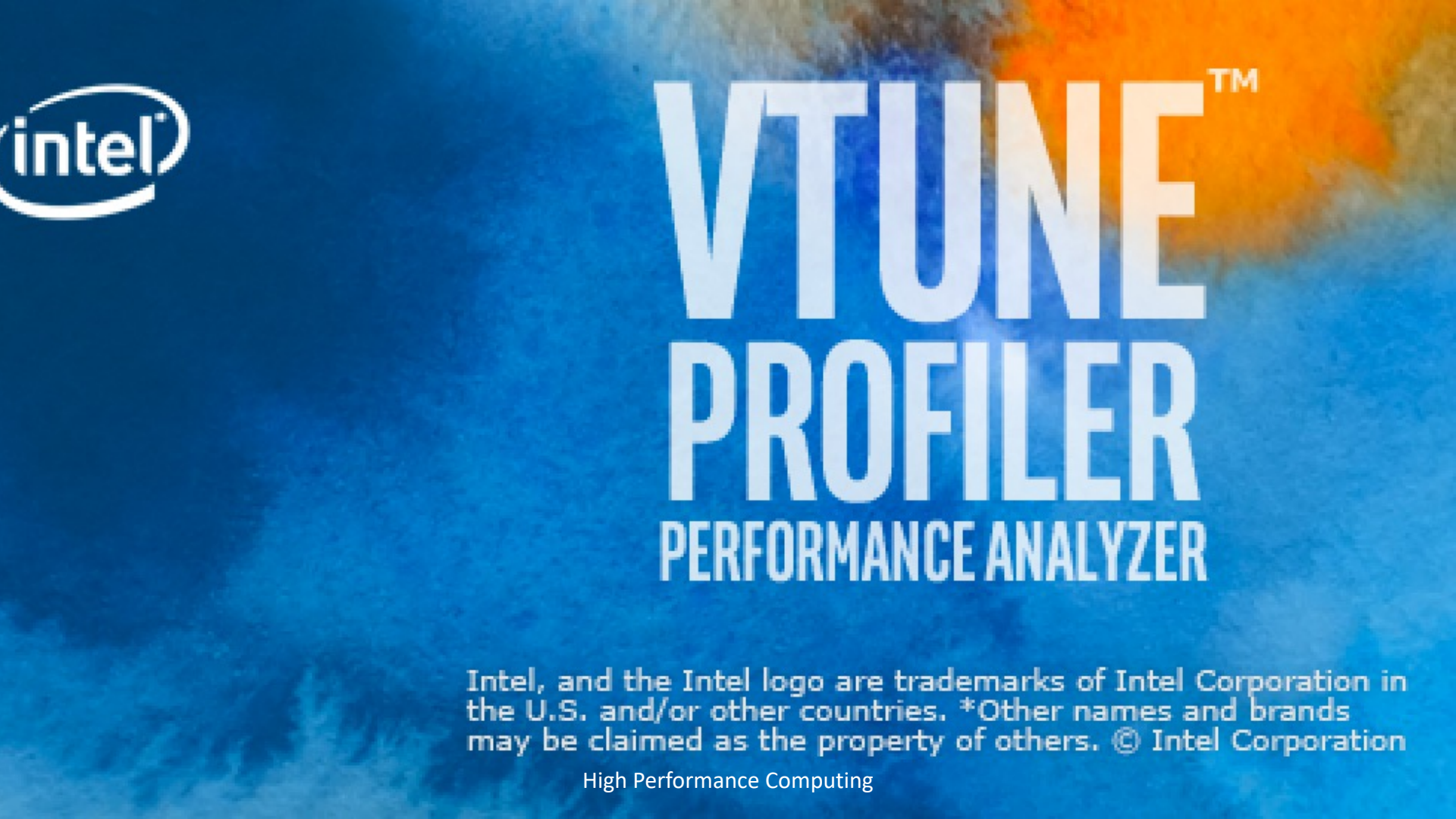
Node Energy (J) 382

Data Movement

No data collected.

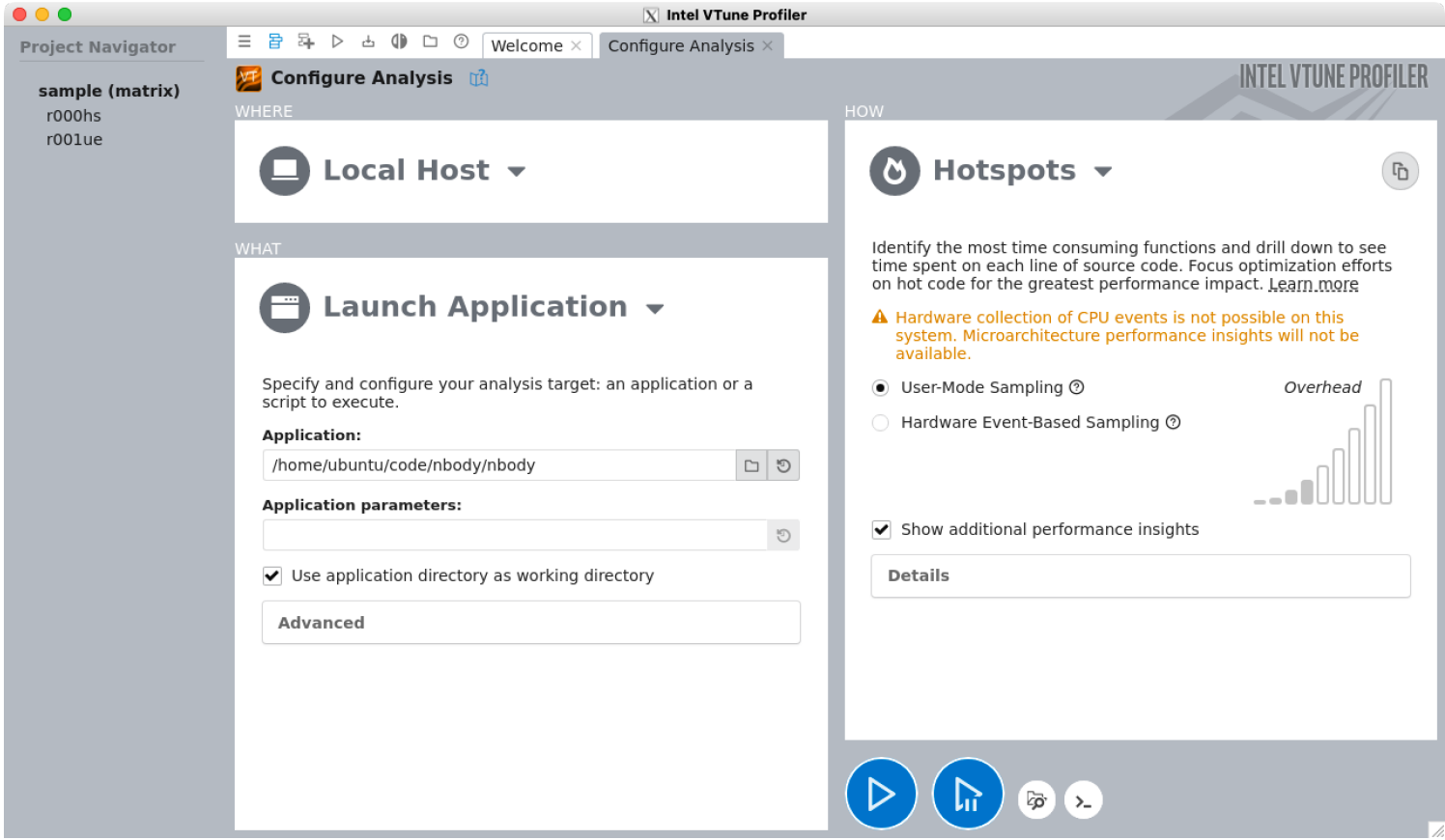
Wallclock time: 3.690296s

nbody+13229-3508s (0.069s)


intel
VTUNE™
PROFILER
PERFORMANCE ANALYZER
Intel, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others. © Intel Corporation

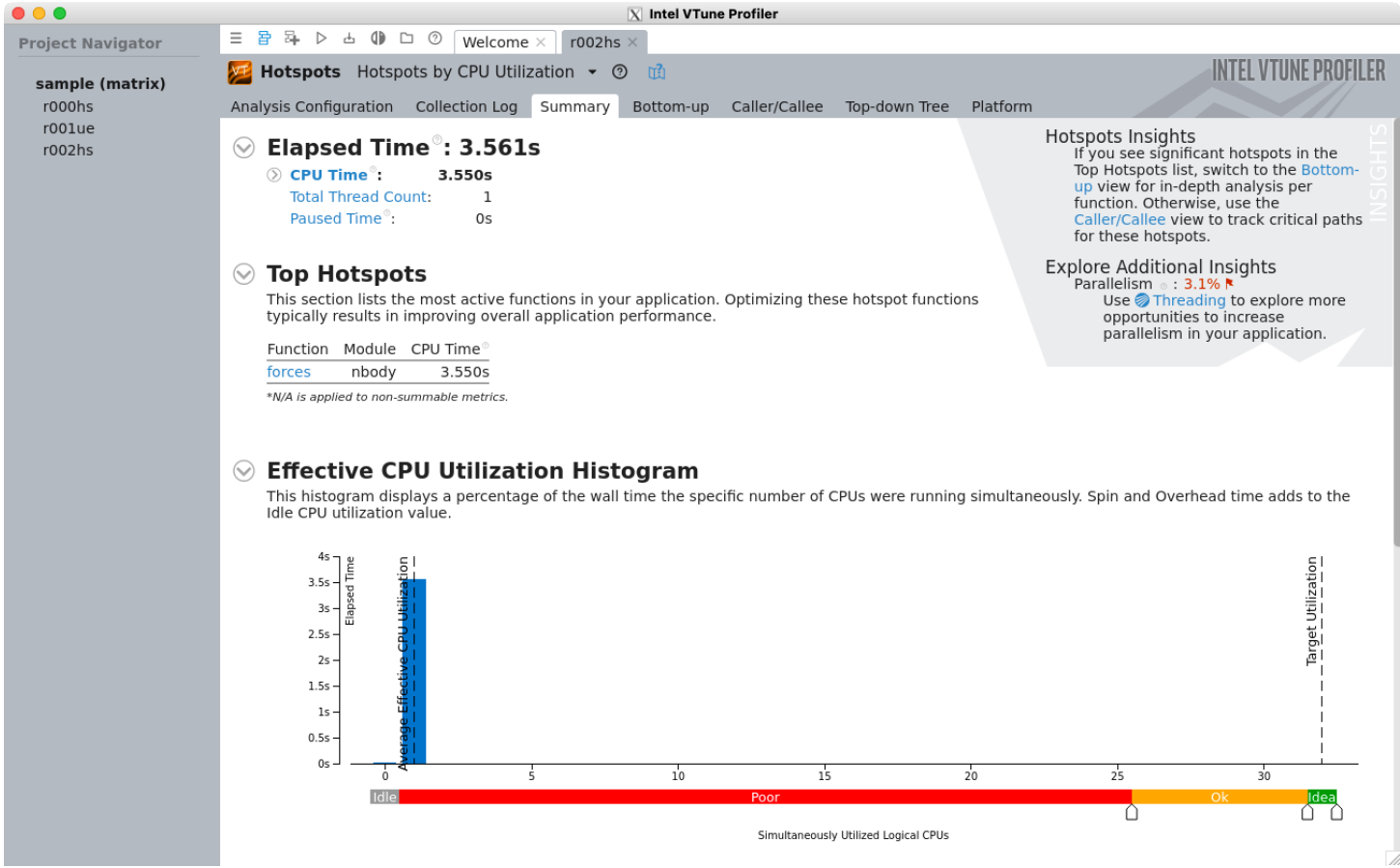
University of Zurich[UZH]

VTUNE



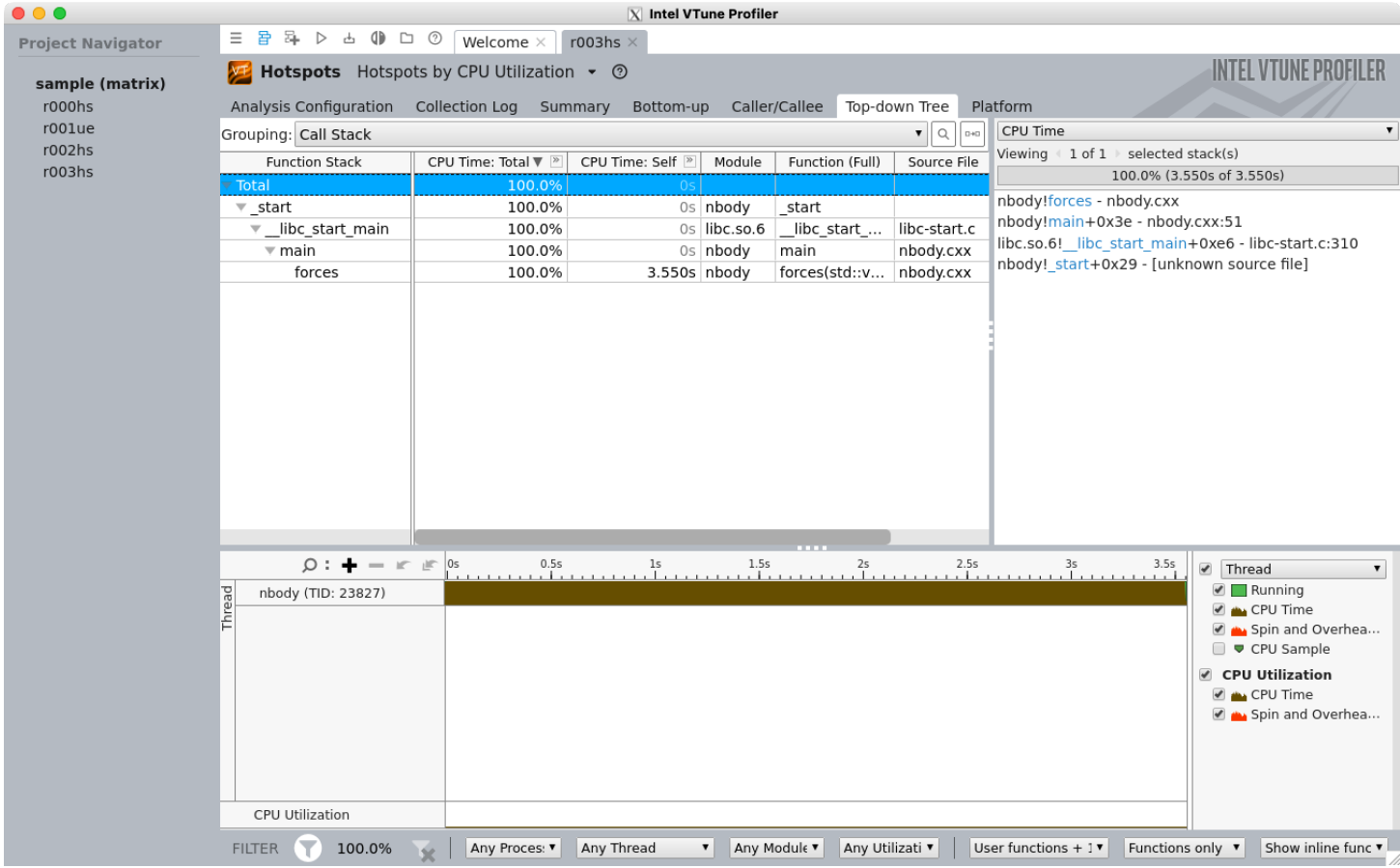
University of Zurich[UZH]

VTUNE



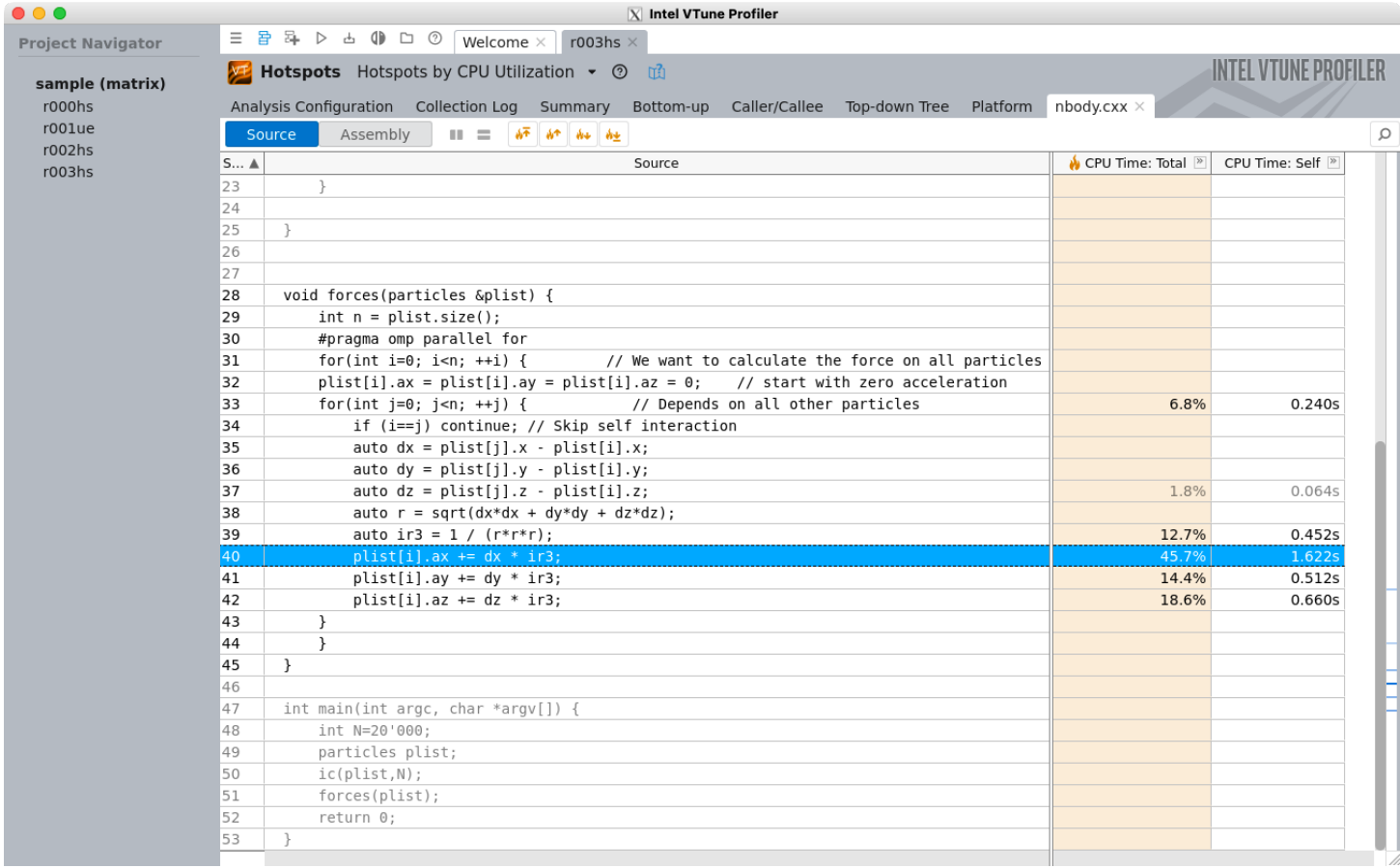
University of Zurich[UZH]

VTUNE



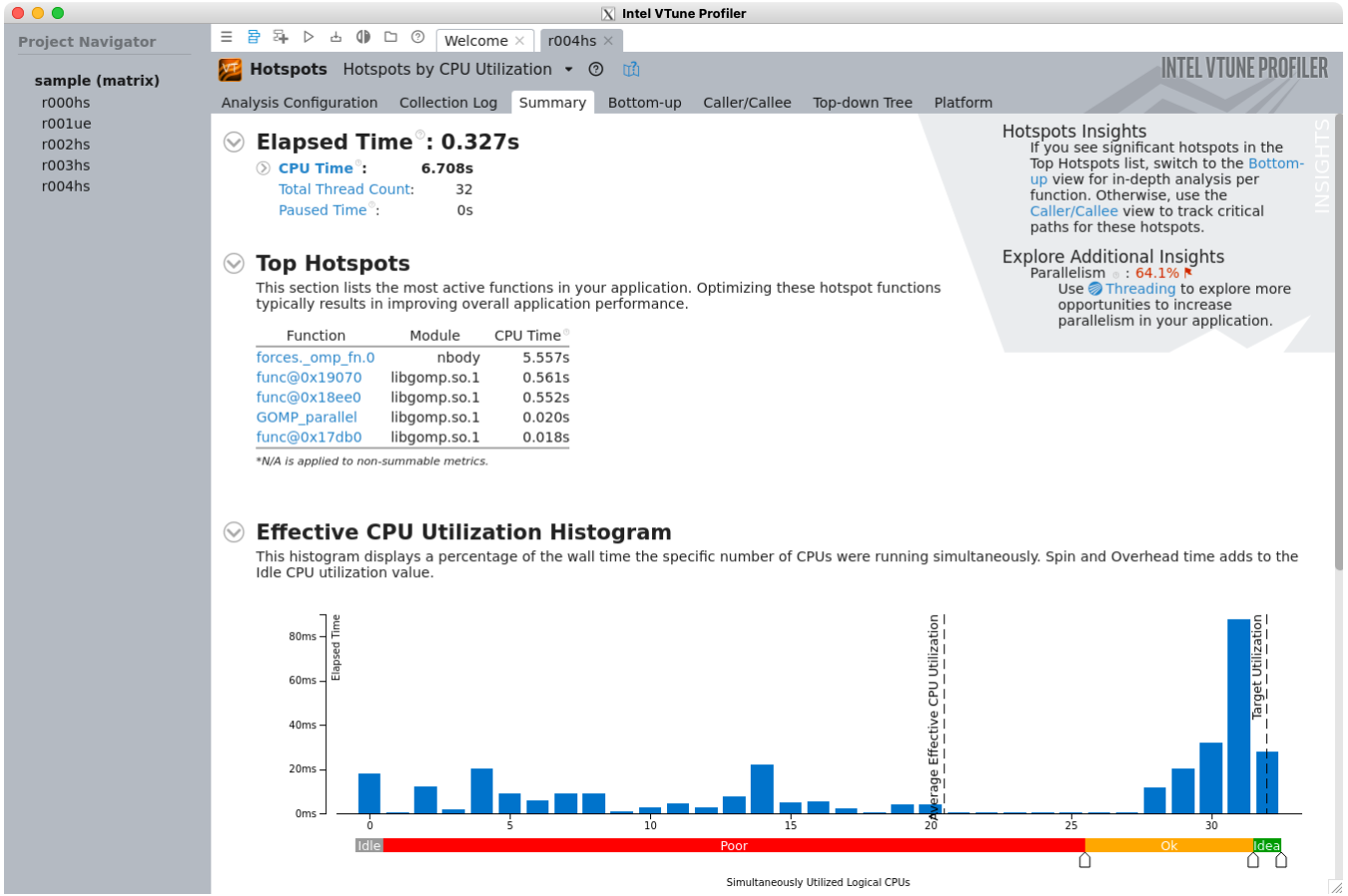
University of Zurich[UZH]

VTUNE



University of Zurich[UZH]

VTUNE



Self timing

- Basic concept
  - Get the current time
  - Do some calculations
  - Get the new time
  - The difference is how long it took
- What function(s) can we use?

time()

```
TIME(2)                                Linux Programmer's Manual                                TIME(2)

NAME
       time - get time in seconds

SYNOPSIS
       #include <time.h>

       time_t time(time_t *tloc);

DESCRIPTION
       time() returns the time as the number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).

       If tloc is non-NULL, the return value is also stored in the memory pointed to by tloc.

RETURN VALUE
       On success, the value of time in seconds since the Epoch is returned.  On error, ((time_t) -1) is returned, and errno is set
appropriately.
```

University of Zurich[UZH]

gettimeofday()

```
GETTIMEOFDAY(2)                                          Linux Programmer's Manual
GETTIMEOFDAY(2)

NAME

       gettimeofday, settimeofday - get / set time

SYNOPSIS

       #include <sys/time.h>

       int gettimeofday(struct timeval *tv, struct timezone *tz);

       int settimeofday(const struct timeval *tv, const struct timezone *tz);


DESCRIPTION

       The  functions  gettimeofday()  and settimeofday() can get and set the time as well as a timezone.

       The tv argument is a struct timeval (as specified in <sys/time.h>), and gives the number of seconds and microseconds

       since the Epoch (see time(2)).  The tz argument is a struct timezone:

           struct timezone {

               int tz_minuteswest;     /* minutes west of Greenwich */

               int tz_dsttime;         /* type of DST correction */

           };
```

gettime.c

```
struct timeval {
    time_t       tv_sec;    /* seconds since Jan. 1, 1970 */
    suseconds_t  tv_usec;   /* and microseconds */
    };
```

```
#include <sys/time.h>

double getTime(void) {
    struct timeval tv;
    struct timezone tz;
    gettimeofday(&tv, &tz);
    return tv.tv_sec + 1e-6*(double)tv.tv_usec;
}
```

clock_gettime() and clock_getres()

SYNOPSIS

```
#include <time.h>
int clock_getres(clockid_t clk_id, struct timespec *res);
int clock_gettime(clockid_t clk_id, struct timespec *tp);
```

DESCRIPTION

The function clock_getres() **finds the resolution** (precision) of the specified clock clk_id, and, if res is non-NULL, stores it in the struct timespec pointed to by res. The resolution of clocks depends on the implementation and cannot be configured by a particular process. If the time value pointed to by the argument tp of clock_settime() is not a multiple of res, then it is truncated to a multiple of res.

The functions clock_gettime() and clock_settime() retrieve and set the time of the specified clock clk_id.

All implementations support the system-wide real-time clock, which is identified by CLOCK_REALTIME. Its time represents **seconds and nanoseconds** since the Epoch. When its time is changed, timers for a relative interval are unaffected, but timers for an absolute point in time are affected.

Sufficiently recent versions of glibc and the Linux kernel support the following clocks:

CLOCK_MONOTONIC

Clock that cannot be set and represents monotonic time since some unspecified starting point. This clock is not affected by discontinuous jumps in the system time (e.g., if the system administrator manually changes the clock), but is affected by the incremental adjustments performed by adjtime(3) and NTP.

High resolution C++ timer

std::chrono::high_resolution_clock

Defined in header <chrono>

class high_resolution_clock; (since C++11)

Class std::chrono::high_resolution_clock represents the clock with the smallest tick period provided by the implementation.

It may be an alias of std::chrono::system_clock or std::chrono::steady_clock, or a third, independent clock.

std::chrono::high_resolution_clock meets the requirements of TrivialClock.

Example C++ program

```
dpotter@daint102:~/hpc1a/Y7> cat c++clock.cxx
#include <iostream>
#include <chrono>
int main(int argc, char*argv[]) {
    using namespace std::chrono;
    using clock = high_resolution_clock;
    double tick = 1.0 * clock::period::num / clock::period::den;
    auto t1 = clock::now();
    std::cout << "Timer resolution is " << tick << " seconds" << std::endl;
    auto t2 = clock::now();
    auto elapsed = duration_cast<duration<double>>(t2 - t1);
    std::cout << "Elapsed time " << elapsed.count() << " seconds" << std::endl;
    return 0;
    }

dpotter@daint102:~/hpc1a/Y7> CC -O3 -o c++clock c++clock.cxx
dpotter@daint102:~/hpc1a/Y7> ./c++clock
Timer resolution is 1e-09 seconds
Elapsed time 4.1871e-05 seconds
dpotter@daint102:~/hpc1a/Y7> ./c++clock
Timer resolution is 1e-09 seconds
Elapsed time 3.7361e-05 seconds
```

Example Python

```
Python 3.9.1 (default, Jan  8 2021, 17:17:43)
[Clang 12.0.0 (clang-1200.0.32.28)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> help(time.perf_counter)
perf_counter(...)
    perf_counter() -> float

    Performance counter for benchmarking.
>>> t1=time.perf_counter() ; t2 = time.perf_counter() ; print(t1,t2,t2-t1)
31.597328265 31.597329154 8.889999989492026e-07
```

Advanced -- “cycle.h”

```
/************************************************************************/
/* To use the cycle counters in your code, simply #include "cycle.h" (this
   file), and then use the functions/macros:

                ticks getticks(void);

   ticks is an opaque typedef defined below, representing the current time.
   You extract the elapsed time between two calls to gettick() via:

                double elapsed(ticks t1, ticks t0);

   which returns a double-precision variable in arbitrary units.  You
   are not expected to convert this into human units like seconds; it
   is intended only for *comparisons* of time intervals.

   (In order to use some of the OS-dependent timer routines like
   Solaris' gethrtime, you need to paste the autoconf snippet below
   into your configure.ac file and #include "config.h" before cycle.h,
   or define the relevant macros manually if you are not using autoconf.)
*/
```

One possible implementation

```
/*
 * X86-64 cycle counter
 */
#if (defined(__GNUC__) || defined(__ICC)) &&
defined(__x86_64__)
typedef unsigned long long ticks;

static __inline__ ticks getticks(void) {
    unsigned a, d;
asm volatile("rdtsc" : "=a" (a), "=d" (d));
    return ((ticks)a) | (((ticks)d) << 32);
    }
```

Time Stamp Counter

From Wikipedia, the free encyclopedia

The **Time Stamp Counter** (**TSC**) is a 64-bit register present on all x86 processors since the Pentium. It counts the number of cycles since reset. The instruction `RDTSC` returns the TSC in EDX:EAX. In x86-64 mode, `RDTSC` also clears the higher 32 bits of RAX and RDX. Its opcode is `0F 31`.[1] Pentium competitors such as the Cyrix 6x86 did not always have a TSC and may consider `RDTSC` an illegal instruction. Cyrix included a Time Stamp Counter in their MII.

Comparison of time

```
Gravity Calculated, Wallclock: 147.719104 secs
  Gflops:2671816.5, Total Gflop:3.95e+08
% computing: max=  97.957  avg=  87.266 of 48400 std-dev=   2.396
%   waiting: max=   6.545  avg=   3.399 of 48400 std-dev=   0.812
%   syncing: max=  16.694  avg=   9.335 of 48400 std-dev=   2.280
```

Summary

- Compiler Wrappers (cc,CC,ftn,mpicc,…)
- Timing a program (**time** command)
- Profiling using sampling (craypat)
- Other alternatives
  - Intel VTUNE
  - AMD µProf
- Build in instrumentation (for phases)

Data Layout (C++) – Indexes start at zero!

```
blitz::Array<float,2> r(1000,2)
blitz::Array<float,1> d2(1000);

for(i=0; i<1000; ++i) {
  d2(i) = r(i,0)*r(i,0) + r(i,1)*r(i,1);
}
```

Data Layout (Fortran) – Indexes start at one!

```
real, dimension(1000,2) :: r
real, dimension(1000) :: d2

do i = 1, 1000
  d2(i) = r(i,1)*r(i,1) + r(i,2)*r(i,2)
end do
```

$$d^2 = x^2 + y^2$$

Data Layout (Fortran)

```
real, dimension(2,1000) :: r
real, dimension(1000) :: d2

do i = 1, 1000
  d2(i) = r(1,i)*r(1,i) + r(2,i)*r(2,i)
end do
```

Data Layout (Fortran)