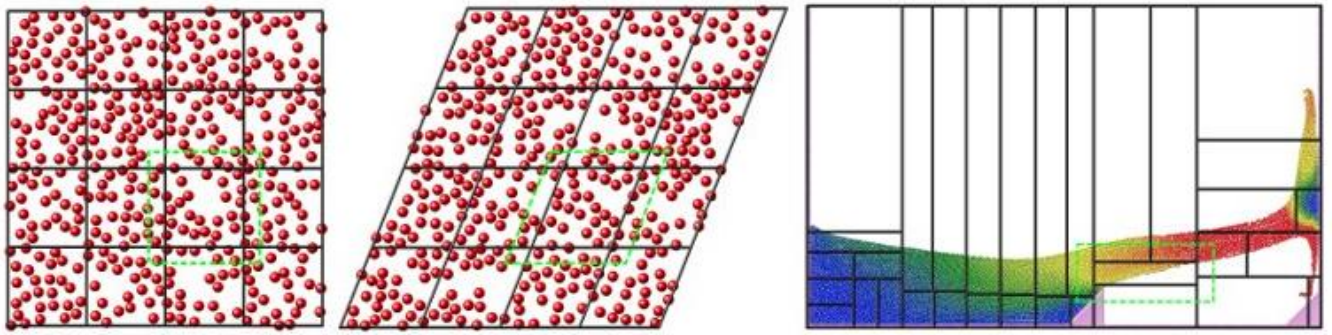
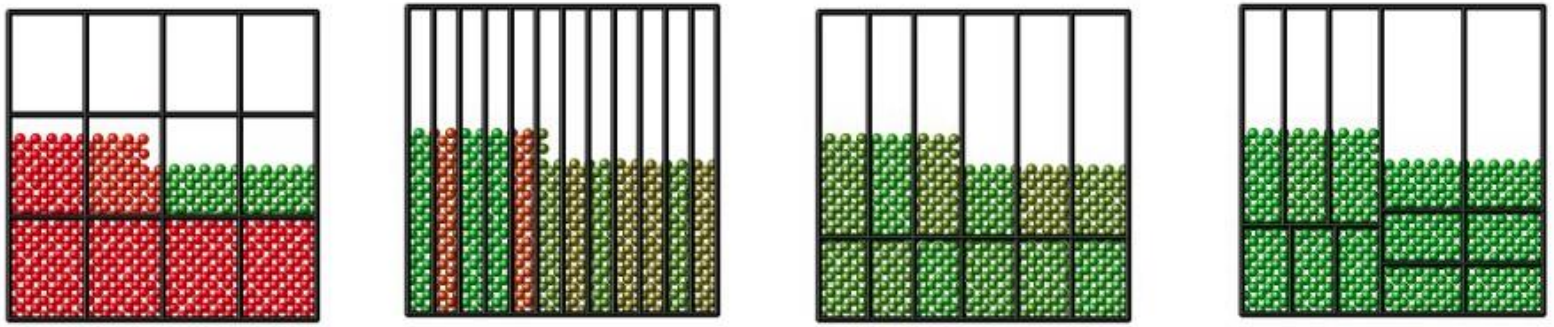
Topic: Decomposition for MPI

Dividing the data





Different decompositions for a 2d system with 12 MPI ranks

Load imbalance is reduced

https://docs.lammps.org/Developer_par_part.html

Error

```
  integer :: i
  real    :: s(1), total

! initialize the varaible s

  do i = 1, 5

      s(i) = i

  enddo
```

Is there any error in this program?

Topic: Tools

Error

./sum.x

Program received signal SIGBUS: Access to an undefined portion of a memory object.

Backtrace for this error:
#0 0x7F04D473CE08
#1 0x7F04D473BF90
#2 0x7F04D438C4AF
#3 0x400820 in MAIN__ at sum.f90:?
Bus error (core dumped)

- **Problems**
- Edit/compile/run times are timje consuming
- Too large data to display and analyze
- Print data for all processes

How to interpret such errors?

Debugging

- What (another) program was doing at the moment it crashed?
- Interactive debugger –> monitor and control the behaviour of running program
- Advantages:
  - Make your program stop on specified conditions
  - Examine what has happened, when your program has stopped
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.
  - Use GDB to debug programs written in C, C@t{++}, Fortran, etc

Debugging tools

Comman method of debugging is to use ‘write statements’

Debuggers:
Gdb
Ddd
TotalView
DDT
Idb
Valgrind

gdb

Compile the program with ‘-g’ to enable building symbol table with gfortran compiler

How to use gdb: In terminal, use ‘gdb executable’

Topic: Tools

Working with gdb

```
 $ gdb ./sum.x
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./sum.x...done.
(gdb) run
Starting program: /home/sandeep/Dropbox/iitkgp/teaching/HPC_CD61004/sandeep_lectures/L23_md_mpi
/gdb/sum.x

Program received signal SIGBUS, Bus error.
0x0000000000400820 in test () at sum.f90:11
11            s(i) = i
(gdb)
```

Useful commands with gdb

run or r –> executes the program from start to end.

break or b line –> sets breakpoint or STOP on a particular line.

Eg: break 12, break subroutine, info break,

delete

next or n -> executes next line of code, but don't dive into functions.

step –> go to next instruction, diving into the function.

print or p –> print the variable value at present step.

list or l –> displays source code.

quit or q –> exits.

Read at https://web.eecs.umich.edu/~sugih/pointers/gdbQS.html

Topic: Tools

Example

```
(gdb)
(gdb) break 3
Breakpoint 1 at 0x400872: file 3_sin.f90, line 3.
(gdb) run
Starting program: /home/sandeep/Dropbox/iitkgp/teaching/HPC_CD61004/sandeep_lectures/L23_md_mpi
/gdb/3_sin.x

Breakpoint 1, test () at 3_sin.f90:9
9         do i = 1, 5
(gdb) next
11            s(i) = sin(real(i))
(gdb) next
9         do i = 1, 5
(gdb) next
11            s(i) = sin(real(i))
(gdb) next
9         do i = 1, 5
(gdb) print i
$1 = 2
(gdb) print s(2)
$2 = 0.909297407
(gdb)
```

- **what do you notice about the loop and the function?**
- **Set another breakpoint: eg. *break 12* and do *cont*. What happens?**
- **Use *step* → when the execution stops, use *where* and *list***

Timers

Measure the execution time of a code

```
program test_secnds
   implicit none
   integer :: i, j
   real(4) :: t1, t2, x

   call cpu_time(t1)

   do i = 1, 10000000
     do j = 1, 100
        x=i+j
     end do
   end do

   call cpu_time(t2)

   write(*,*) "time ", t2-t1, " seconds."
end_program test_secnds
```

Profiling using gprof

Compile using the flag '-pg'

Eg: gfortran -pg sum.f90 -o sum.x

And run the executable. It creates *gmon.out* file which is non-readable format

Use command *gprof* to get timings (it reads gmon.out file),

Eg: gprof ./sum.x OR

- Measure the execution time at the subprogram level
- Reports how many times the subprogram or function is called, who and whom it called, time spent, etc

- It prints two profles – Flat profile and Call graph

Example

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 99.37      8.41     8.41    30001   280.21   280.21  force_calc_
  0.71      8.47     0.06    30000     2.00   282.22  integrate_
  0.00      8.47     0.00        1     0.00     0.00  initialize_
```

Topic: Tools

```
                     Call graph


granularity: each sample hit covers 2 byte(s) for 0.12% of 8.47 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    8.47                 MAIN__ [1]
                0.06    8.41   30000/30000       integrate_ [2]
                0.00    0.00       1/30001       force_calc_ [3]
                0.00    0.00       1/1           initialize_ [4]
-----------------------------------------------
                0.06    8.41   30000/30000       MAIN__ [1]
[2]    100.0    0.06    8.41   30000         integrate_ [2]
                8.41    0.00   30000/30001       force_calc_ [3]
-----------------------------------------------
                0.00    0.00       1/30001       MAIN__ [1]
                8.41    0.00   30000/30001       integrate_ [2]
[3]     99.3    8.41    0.00   30001         force_calc_ [3]
-----------------------------------------------
                0.00    0.00       1/1           MAIN__ [1]
[4]      0.0    0.00    0.00       1         initialize_ [4]
-----------------------------------------------


Index by function name

   [3] force_calc_  _          [4] initialize_             [2] integrate_
```