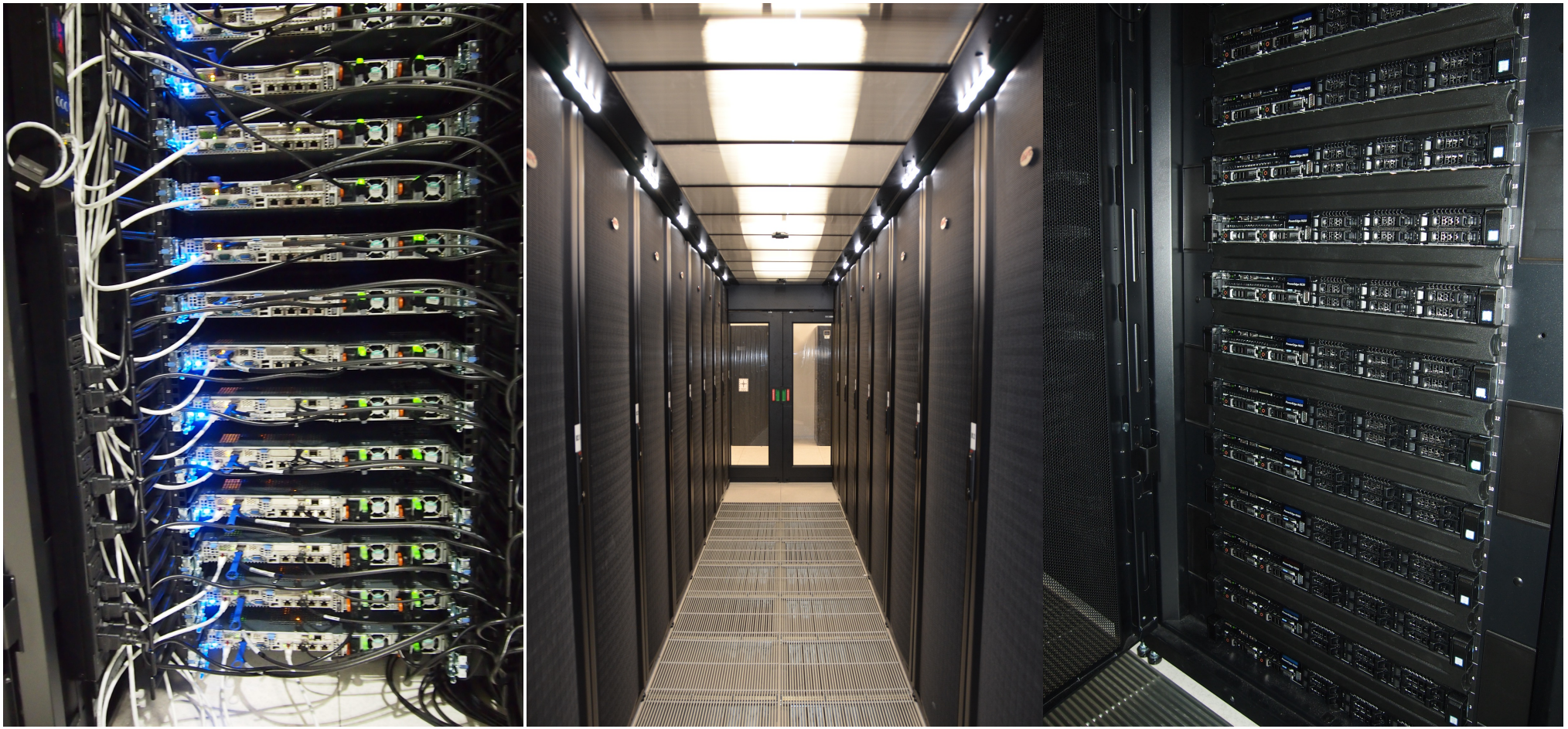
High Performance Computing for Data Science

Project topics & P2P communication benchmark
Lecture 10 - 13/10/2023

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024



HPC @ UniTrento

On-site visit October 20th

High Performance Computing for Data Science

Project topics & P2P communication benchmark
Lecture 10 - 13/10/2023

Yesterday's assignment

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

Scatter & Gather assignments

Exercise 1

I. How would you handle a scatter of a 15-element vector with 4 processes?
II. Would it work?
III. Are there other MPI calls that can help on that?

Exercise 2

I. Try to implement **matrix-vector** multiplication example

Exercise 3

I. MPI_SendRecv

High Performance Computing for Data Science

Project topics & P2P communication benchmark
Lecture 10 - 13/10/2023

Project topics

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

Project topics (I)

1. Parallel CountSketch
2. Parallel MergeSort
3. Parallel tridiagonal linear system
4. Parallel numerical integration using Romberg's method
5. Parallel numerical integration using Gaussian Quadrature
6. Parallel Huffman Coding and decoding
7. Parallel Matrix Multiplication: SUMMA Algorithm
8. Parallel FFT
9. Parallel closest pair of points (among n input points)
10. Parallel Outer-Product Matrix Multiplication
11. Parallel Snyder Matrix Multiplication
12. Parallel Minimum Spanning Tree (Boruvka's algorithm)
13. Parallel Minimum Spanning Tree (Kruskal's algorithm)
14. Parallel Moth-Flame Optimization Algorithm
15. Parallel Raven Roosting Optimization Algorithm
16. Parallel Emperor Penguin Optimization Algorithm
17. Parallel Shark Smell Optimization
18. Parallel Grey Wolf Optimization
19. Parallel Fish School Search
20. Parallel Dragonfly Algorithm
21. Parallel root-finding of a polynomial
22. Parallel A-Star
23. Parallel All Pairs Shortest Paths

Project topics (II)

24. Parallel Bat Algorithm
25. Parallel solver per Aristotle's Number Puzzle
26. Parallel solver per Futoshiki puzzle (board at least 5 x 5, if possible 10 x 10)
27. Parallel Fast Hartley Transform
28. Parallel Bareiss algorithm
29. Parallel Levinson-Durbin algorithm
30. Parallel REQ (quantiles)
31. Parallel conjugate gradient
32. Parallel connected components
33. Parallel Q-Digest (quantiles)
34. Parallel Count-Min sketch
35. Parallel CPC (Compressed Probabilistic Counting) Sketch
36. Parallel Hyper Log Log Sketch
37. Parallel GSA (Gravitational Search Algorithm)
38. Parallel EM (Expectation-Maximization) clustering
39. Parallel CHARM (closed frequent itemsets mining)
40. Parallel Singular Value Decomposition
41. Parallel Spectral Clustering
42. Parallel Angle-Based Clustering
43. Parallel BIRCH Clustering
44. Parallel LU factorization
45. Parallel Selection
46. Parallel Spline Interpolation

Project topics (III)

47. Parallel GSA (Gravitational Search Algorithm)
48. Parallel BFR clustering (Bradley, Fayyad, Reina)
49. Parallel Harmony Search
50. Parallel Grasshopper algorithm
51. Parallel Flower Pollination Algorithm
52. Parallel Hough Transform
53. Parallel Discrete Haar transform
54. Parallel Matrix Inversion
55. Parallel Sinkhorn factorization of a square matrix with positive entries
56. Parallel SVD factorization
57. Parallel Bareiss algorithm for computing the determinant of a matrix
58. Parallel MergeSort
59. Parallel Artificial Gorilla Troops Optimizer
60. Parallel Stencil Computation
61. Parallel Hitori solver
62. Parallel Cholesky factorization of a dense, symmetric and positive definite square n x n matrix A
63. Parallel QR factorization
64. Parallel Power Method for eigenvalues
65. Parallel linear regression
66. Parallel SLASH
67. Parallel Matrix Inversion
68. Parallel Autoencoder

Parallel data transfer topic

Objective: Development of a parallel application that does parallel download of a file

Scenarios:

- the file is available on multiple servers
- the servers can have different bandwidth with respect to our HPC center
- Files are in netcdf format (scientific format for climate data; C examples on the use of the API are available you can grab some code for I/O from there)

Challenges:

- Parallel data transfer (core part)
- Load balancing optimization
- NetCDF (it is a scientific data format, so it joins data + metadata)

Test Case

- Real scenarios with multiple servers from the Earth System Grid Federation. We'll choose three servers in different countries (France, UK, Germany, which own the biggest repositories in the federation.

Potential impact

- The community could use it if available with a simple Command Line Interface

Multiple teams can be engaged for this project. Please contact me if you are interest in this topic

High Performance Computing for Data Science

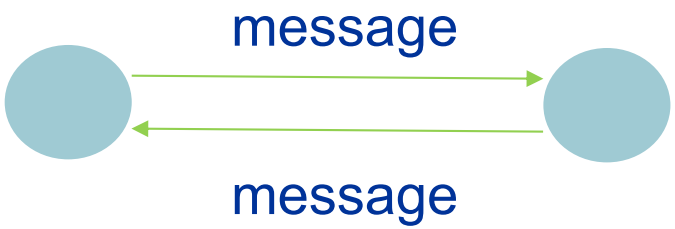
Project topics & P2P communication benchmark
Lecture 10 - 13/10/2023

Point-to-point communication benchmark

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

Benchmarking p2p communication

Write a program to measure the time it takes to send 1, 2, 4, ..., 1M C bytes (arrays) from one processor to another using ***MPI_Send*** and ***MPI_Recv***.



New concepts:

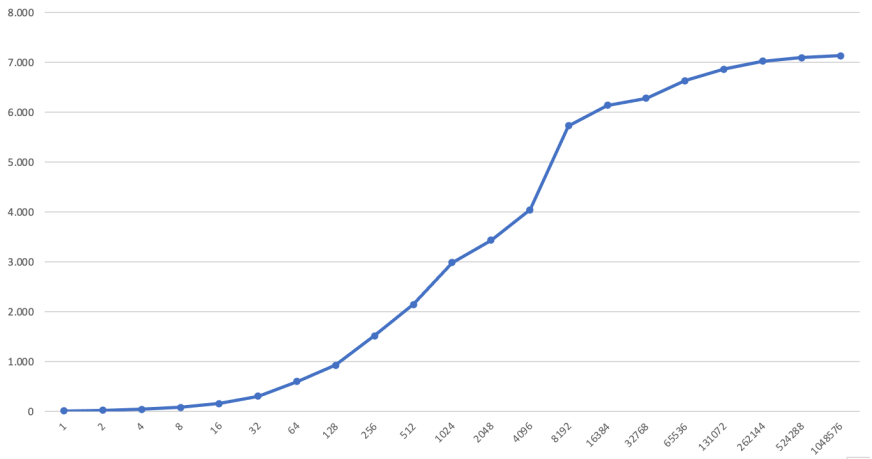
- *PBS "placing" strategies (class)*
- *MPI_Wtime (class)*
- *MPI_Sendrecv (home)*

Practice with *malloc/free, MPI_Send & MPI_Receive*

Used to estimate the bandwidth

Expected output

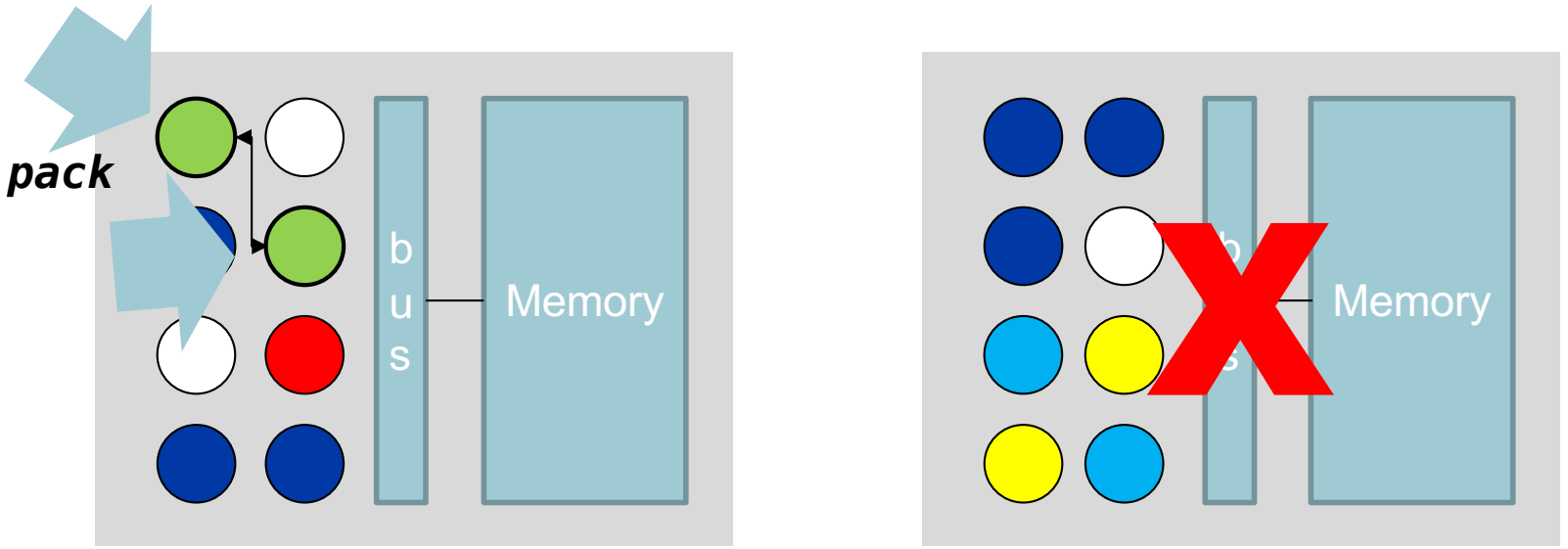
- A **2-process MPI** application
- **Print** the size, time, and rate in MB/sec for each message length
- **Plot** the **MB/sec** in a chart (message size on the X axis, use log scale)



n	time (sec)	Rate (MB/sec)
1	0.000001	6.967282
2	0.000001	12.631068
4	0.000001	28.802088
8	0.000001	55.831002
16	0.000001	106.175428
32	0.000001	211.339661
64	0.000001	438.261969
128	0.000002	639.675980
256	0.000002	1010.580540
512	0.000003	1493.901668
1024	0.000005	1762.037865
2048	0.000010	1655.891006
4096	0.000018	1796.587627
8192	0.000024	2681.735678
16384	0.000042	3097.215853
32768	0.000103	2539.287824
65536	0.000207	2531.978417
131072	0.000414	2531.249791
262144	0.000829	2529.793794
524288	0.001635	2565.393517
1048576	0.003264	2570.359944

PBS Placing strategies (1)

1. #PBS –l select=2:ncpus=1:mem=2gb –l *place*=***pack***



Free core

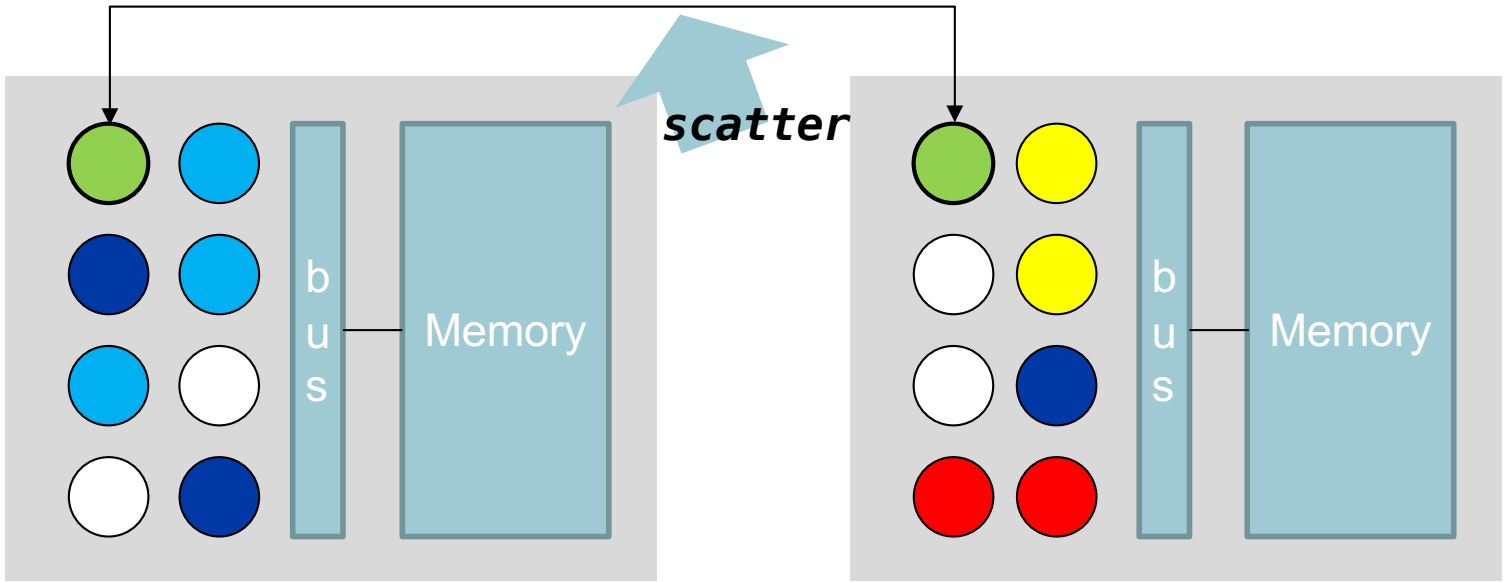
Cores allocated to my app

Cores allocated to other users' apps

2 chunks (select 2) by 1 core (ncpus 1) on the same node (pack)

PBS Placing strategies (2)

2. `#PBS -l select=2:ncpus=1:mem=2gb -l place=`***`scatter`***



Free core

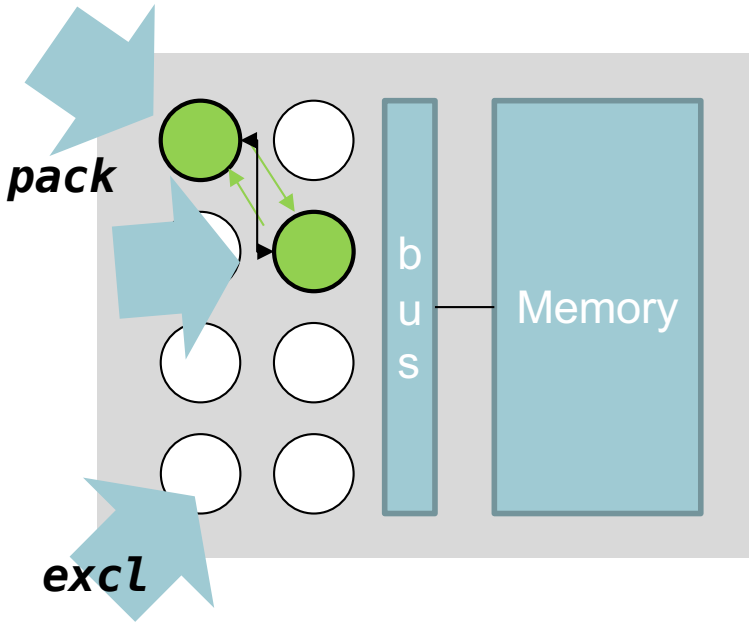
Cores allocated to my app

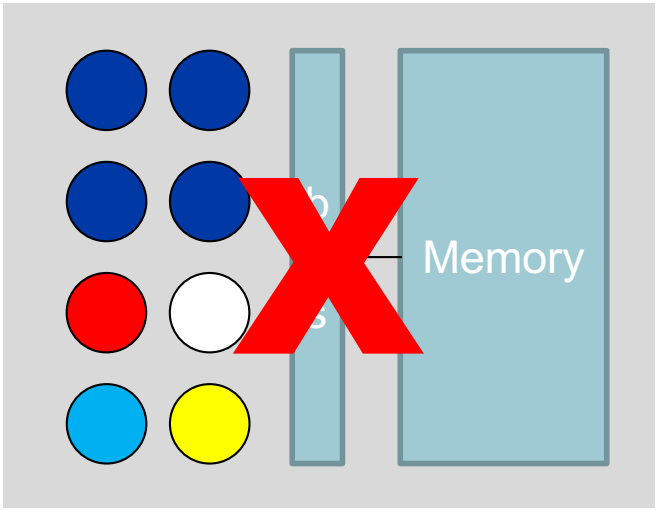
Cores allocated to other users' apps

2 chunks (select 2) by 1 core (ncpus 1) on two different nodes (scatter)

PBS Placing strategies (3)

3. `#PBS -l select=2:ncpus=1:mem=2gb -l place=`***`pack:excl`***





Free core

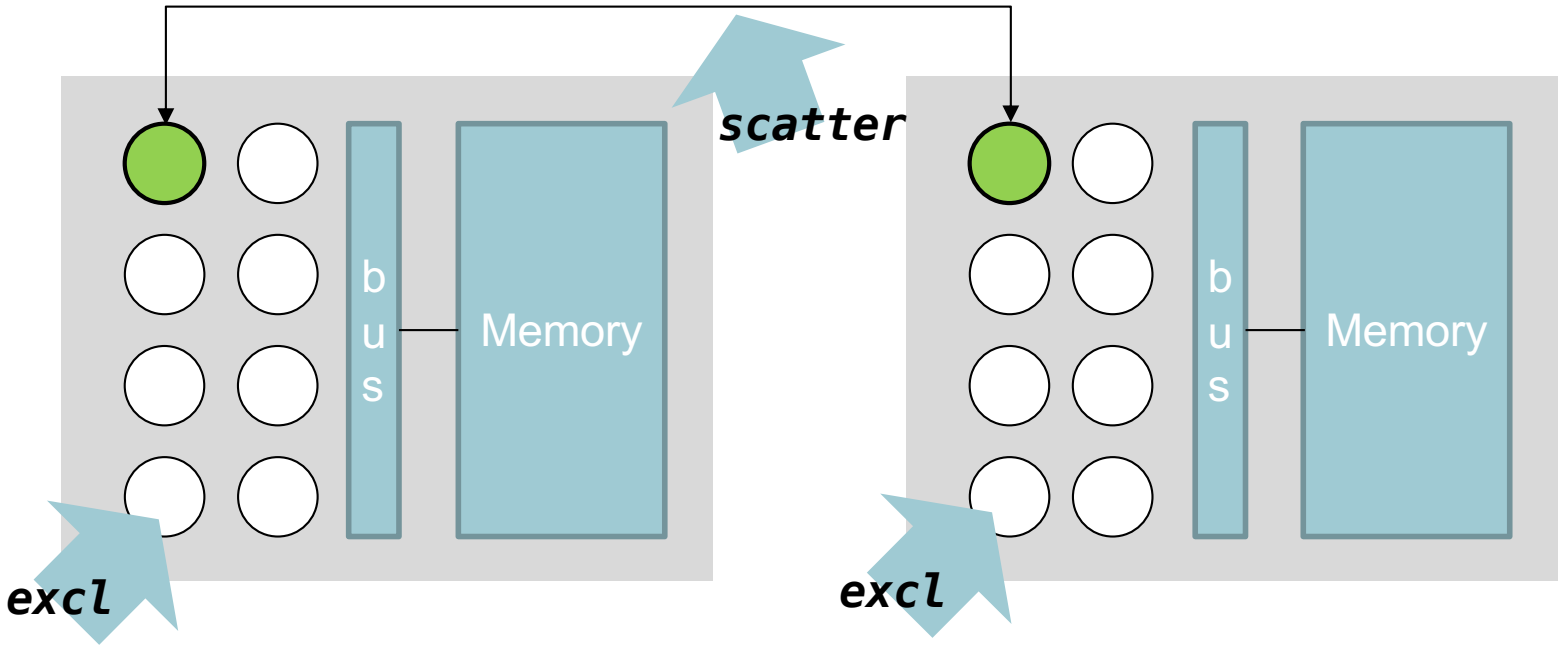
Cores allocated to my app

Cores allocated to other users' apps

2 chunks (select 2) by 1 core (ncpus 1) on the same node (pack) with exclusive allocation (excl)

PBS Placing strategies (4)

4. `#PBS -l select=2:ncpus=1:mem=2gb -l place=`***`scatter:excl`***



Free core

Cores allocated to my app

2 chunks (select 2) by 1 core (ncpus 1) on two different nodes (scatter) with exclusive allocation (excl)

Run over multiple configurations

```
#PBS -l select=2:ncpus=1:mem=2gb -l place=pack
#PBS -l select=2:ncpus=1:mem=2gb -l place=scatter
#PBS -l select=2:ncpus=1:mem=2gb -l place=pack:excl
#PBS -l select=2:ncpus=1:mem=2gb -l place=scatter:excl
```

Important for your project

New MPI calls: MPI_Wtime

MPI_Wtime

Returns an elapsed time on the calling processor

Synopsis

```
double MPI_Wtime( void )
```

Return value

Time in seconds since an arbitrary time in the past.

Notes

This is intended to be a high-resolution, elapsed (or wall) clock. See `MPI_WTICK` to determine the resolution of `MPI_WTIME`. If the attribute `MPI_WTIME_IS_GLOBAL` is defined and true, then the value is synchronized across all processes in `MPI_COMM_WORLD`.

Notes for Fortran

This is a function, declared as `DOUBLE PRECISION MPI_WTIME()` in Fortran.

See Also also: MPI_Wtick, MPI_Comm_get_attr, MPI_Attr_get

Location:src/mpi/timer/wtime.c

https://www.mpich.org/static/docs/v3.1/www3/MPI_Wtime.html

gettimeofday

```
GETTIMEOFDAY(2)          FreeBSD System Calls Manual          GETTIMEOFDAY(2)

NAME
     gettimeofday, settimeofday -- get/set date and time

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <sys/time.h>

     int
     gettimeofday(struct timeval *tp, struct timezone *tzp);

     int
     settimeofday(const struct timeval *tp, const struct timezone *tzp);

DESCRIPTION
     The system's notion of the current Greenwich time and the current time
     zone is obtained with the gettimeofday() system call, and set with the
     settimeofday() system call.  The time is expressed in seconds and mi-
     croseconds since midnight (0 hour), January 1, 1970.  The resolution of
     the system clock is hardware dependent, and the time may be updated con-
     tinuously or in "ticks".  If tp or tzp is NULL, the associated time in-
     formation will not be returned or set.

     The structures pointed to by tp and tzp are defined in <sys/time.h> as:

     struct timeval {
             time_t          tv_sec;         /* seconds */
             suseconds_t     tv_usec;        /* and microseconds */
     };

     struct timezone {
             int     tz_minuteswest; /* minutes west of Greenwich */
             int     tz_dsttime;     /* type of dst correction */
     };

     The timezone structure indicates the local time zone (measured in minutes
     of time westward from Greenwich), and a flag that, if nonzero, indicates
     that Daylight Saving time applies locally during the appropriate part of
     the year.
```

MPI_Sendrecv (Assignment)

MPI_Sendrecv

Sends and receives a message

Synopsis

```
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 int dest, int sendtag,
                 void *recvbuf, int recvcount, MPI_Datatype recvtype,
                 int source, int recvtag, MPI_Comm comm, MPI_Status * status)
```

Input Parameters

sendbuf
: initial address of send buffer (choice)

sendcount
: number of elements in send buffer (integer)

sendtype
: type of elements in send buffer (handle)

dest
: rank of destination (integer)

sendtag
: send tag (integer)

recvcount
: number of elements in receive buffer (integer)

recvtype
: type of elements in receive buffer (handle)

source
: rank of source (integer)

recvtag
: receive tag (integer)

comm
: communicator (handle)

Output Parameters

recvbuf
: initial address of receive buffer (choice)

status
: status object (Status). This refers to the receive operation.

Useful for tomorrow!

High Performance Computing for Data Science

Project topics & P2P communication benchmark
Lecture 10 - 13/10/2023

Point-to-point communication benchmark Solution

Prof. Sandro Fiore, Ph.D.
Department of Information Engineering and Computer Science (DISI)
University of Trento, 2023-2024

P2P Communication code (part I)

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

#define NUMBER_OF_TESTS 10

int main( argc, argv )
int argc;
char **argv;
{
    double       *buf;
    int          rank;
    int          n;
    double       t1, t2, tmin;
    int          i, j, k, nloop;
    MPI_Status   status;

    MPI_Init( &argc, &argv );

    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    if (rank == 0)
        printf( "Kind\t\tn\ttime (sec)\tRate (MB/sec)\n" );

    for (n=1; n<1100000; n*=2) {
        if (n == 0) nloop = 1000;
        else        nloop  = 1000/n;
        if (nloop < 1) nloop = 1;

        buf = (double *) malloc( n * sizeof(double) );
        if (!buf) {
            fprintf( stderr,
                     "Could not allocate send/recv buffer of size %d\n", n );
            MPI_Abort( MPI_COMM_WORLD, 1 );
        }
        tmin = 1000;
```

P2P Communication code (part II)

```
        for (k=0; k<NUMBER_OF_TESTS; k++) {
            if (rank == 0) {
                /* Make sure both processes are ready */
                MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 1, 14,
                              MPI_BOTTOM, 0, MPI_INT, 1, 14, MPI_COMM_WORLD,
                              &status );
                t1 = MPI_Wtime();
                for (j=0; j<nloop; j++) {
                    MPI_Send( buf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD );
                    MPI_Recv( buf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD,
                              &status );
                }
                t2 = (MPI_Wtime() - t1) / nloop;
                if (t2 < tmin) tmin = t2;
            }
            else if (rank == 1) {
                /* Make sure both processes are ready */
                MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
                              MPI_BOTTOM, 0, MPI_INT, 0, 14, MPI_COMM_WORLD,
                              &status );
                for (j=0; j<nloop; j++) {
                    MPI_Recv( buf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD,
                              &status );
                    MPI_Send( buf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD );
                }
            }
        }
        /* Convert to half the round-trip time */
        tmin = tmin / 2.0;
        if (rank == 0) {
            double rate;
            if (tmin > 0) rate = n * sizeof(double) * 1.0e-6 /tmin;
            else          rate = 0.0;
            printf( "Send/Recv\t%d\t%f\t%f\n", n, tmin, rate );
        }
        free( buf );
    }

    MPI_Finalize( );
    return 0;
}
```

https://www.mcs.anl.gov/research/projects/mpi/tutorial/mpiexmpl/src3/pingpong/C/main.html

Feedback form

- Throughout the entire HPC4DS course there will be a form available for your feedback
  - Please provide any comment about:
    - Pros
    - Cons
    - Aspects that were not clear enough during the class
    - Any other feedback you think can be relevant for the course
    - …
- Information are gathered in an anonymous way

https://docs.google.com/forms/d/e/1FAIpQLScSVpNPfMP8poNNz-nLl9xlKlHkeBM2x_JgWyA0qQu6mOBl5A/viewform