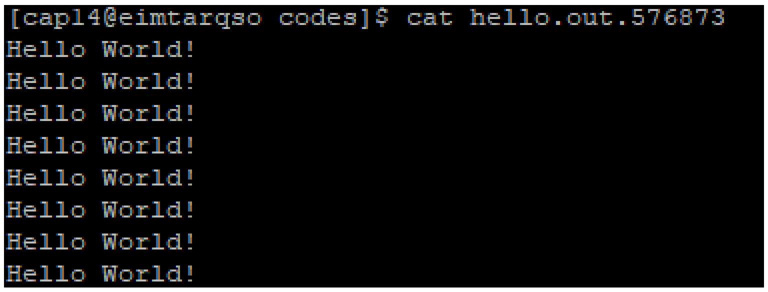
High-Performance Computer
Rovira e Virgili-UOC

Student: Felix

1. **What is the result of the execution of the MPI program discussed above? Why?**

The result of the program hello discussed is the following:



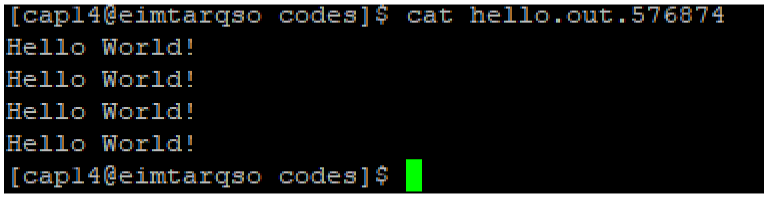
```
[capl4@eimtarqso codes]$ cat hello.out.576873
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Why?

This is a SPMD job, the syntax used mpirun with option -np 8 (np number of processes) in this case indicate to MPI runtime to start 8 processes.

The mpirun command controls several aspects of program execution in Open MPI. mpirun uses the Open Run-Time Environment (ORTE) to launch jobs

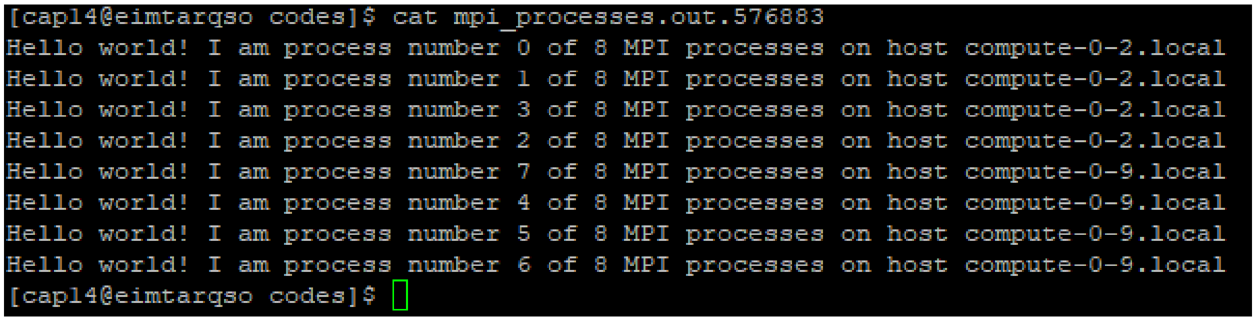
2. Explain what happens if you use the mpirun option "-np 4" instead of "-np 8" with the example above.



```
[capl4@eimtarqso codes]$ cat hello.out.576874
Hello World!
Hello World!
Hello World!
Hello World!
[capl4@eimtarqso codes]$
```

the syntax used mpirun with option -np 4 (np number of processes) in this case indicate to MPI runtime to start 4 processes.

3. **What is the result of the execution of the MPI program above? Why?**



```
[capl4@eimtarqso codes]$ cat mpi_processes.out.576883
Hello world! I am process number 0 of 8 MPI processes on host compute-0-2.local
Hello world! I am process number 1 of 8 MPI processes on host compute-0-2.local
Hello world! I am process number 3 of 8 MPI processes on host compute-0-2.local
Hello world! I am process number 2 of 8 MPI processes on host compute-0-2.local
Hello world! I am process number 7 of 8 MPI processes on host compute-0-9.local
Hello world! I am process number 4 of 8 MPI processes on host compute-0-9.local
Hello world! I am process number 5 of 8 MPI processes on host compute-0-9.local
Hello world! I am process number 6 of 8 MPI processes on host compute-0-9.local
[capl4@eimtarqso codes]$ 
```

For this printing important to notice the next points

These lines below see 3.1 are the function calls and are getting information about MPI_COMM_WORLD for both functions the first argument is a communicator

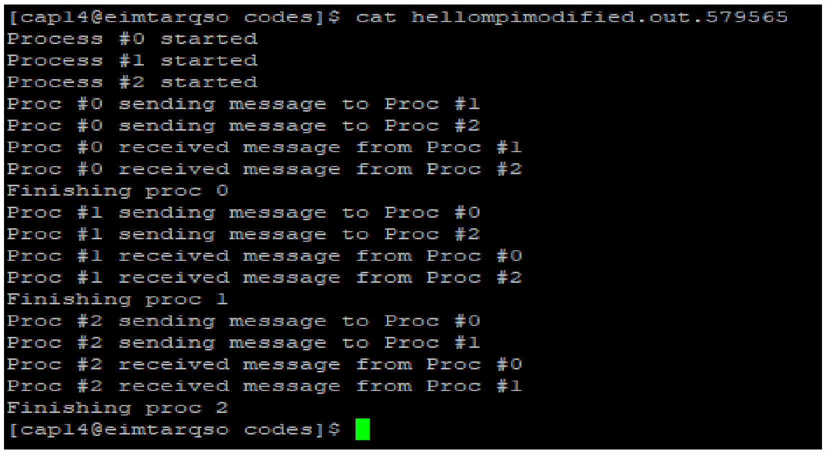
- MPI_Comm_size returns in its second argument the number of processes in the communicator that is 8.
- MPI_Comm_rank (Rank is a logical way of numbering processes) returns in its second argument the calling process rank in the communicator that is 0,1,2,3,4,5,6,7.

Also, we have the computer hostname with the nodes used in this case we are utilizing 8 cores and starting 8 processes node 2 (4 cores) and node 9 (4 cores)

3.1

```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
```

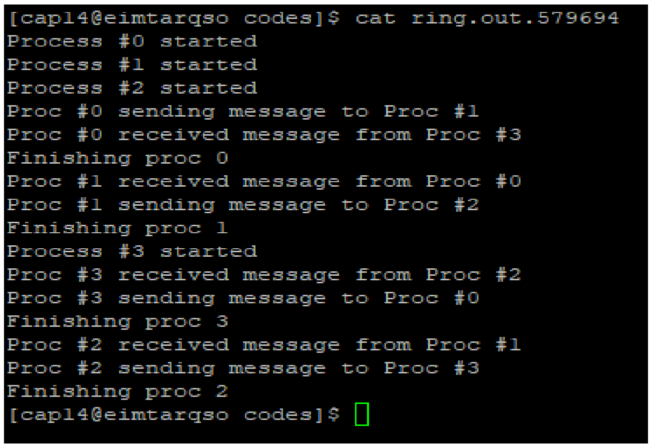
4. **Provide the code of the variant of hellompi.c as described above.**
   Attached: hellompimodified.c



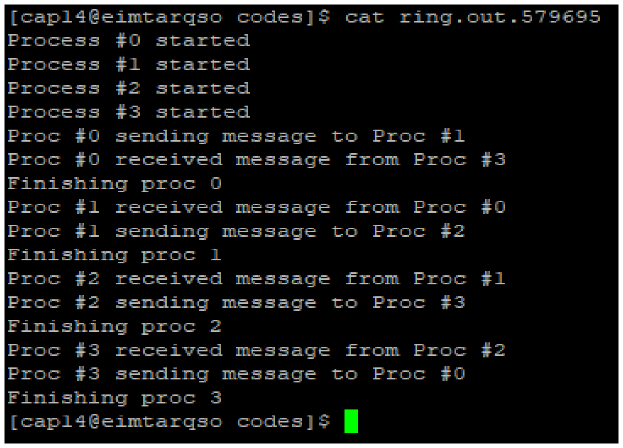
```
[capl4@eimtarqso codes]$ cat hellompimodified.out.579565
Process #0 started
Process #1 started
Process #2 started
Proc #0 sending message to Proc #1
Proc #0 sending message to Proc #2
Proc #0 received message from Proc #1
Proc #0 received message from Proc #2
Finishing proc 0
Proc #1 sending message to Proc #0
Proc #1 sending message to Proc #2
Proc #1 received message from Proc #0
Proc #1 received message from Proc #2
Finishing proc 1
Proc #2 sending message to Proc #0
Proc #2 sending message to Proc #1
Proc #2 received message from Proc #0
Proc #2 received message from Proc #1
Finishing proc 2
[capl4@eimtarqso codes]$ 
```

5. Provide the code of your ring program using MPI.

Attached ring.c



```
[cap14@eimtarqso codes]$ cat ring.out.579694
Process #0 started
Process #1 started
Process #2 started
Proc #0 sending message to Proc #1
Proc #0 received message from Proc #3
Finishing proc 0
Proc #1 received message from Proc #0
Proc #1 sending message to Proc #2
Finishing proc 1
Process #3 started
Proc #3 received message from Proc #2
Proc #3 sending message to Proc #0
Finishing proc 3
Proc #2 received message from Proc #1
Proc #2 sending message to Proc #3
Finishing proc 2
[cap14@eimtarqso codes]$
```



```
[cap14@eimtarqso codes]$ cat ring.out.579695
Process #0 started
Process #1 started
Process #2 started
Process #3 started
Proc #0 sending message to Proc #1
Proc #0 received message from Proc #3
Finishing proc 0
Proc #1 received message from Proc #0
Proc #1 sending message to Proc #2
Finishing proc 1
Proc #2 received message from Proc #1
Proc #2 sending message to Proc #3
Finishing proc 2
Proc #3 received message from Proc #2
Proc #3 sending message to Proc #0
Finishing proc 3
[cap14@eimtarqso codes]$
```

6. What is the order of the messages in the output? Why?

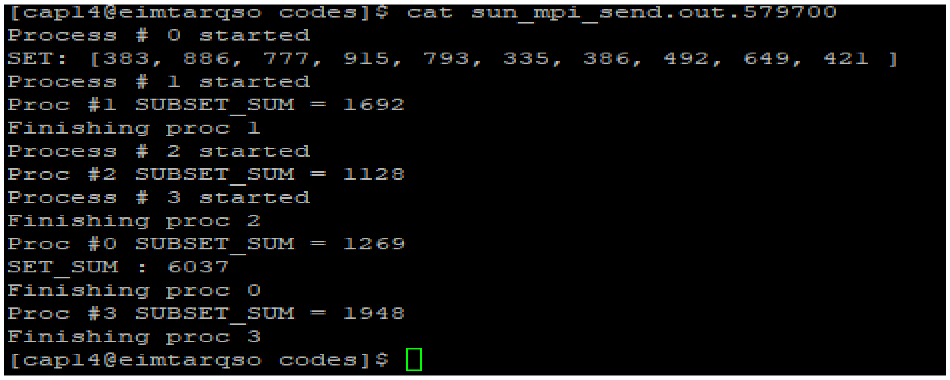
Although the MPI standard doesn't specify which processes have access to which I/O devices, virtually all MPI implementations allow all the processes in MPI COMM WORLD full access to stdout and stderr, so most MPI implementations allow all processes to execute printf and fprintf(stderr, ...). However, most MPI implementations don't provide any automatic scheduling of access to these devices. That is if multiple processes are attempting to write to, say, stdout, the order in which the processes' output appears will be unpredictable. Indeed, it can even happen that the output of one process will be interrupted by the output of another process.

The reason this happens is that the MPI processes are "competing" for access to the shared output device, stdout, and it's impossible to predict the order in which the processes' output will be queued up. Such competition results in **nondeterminism**.

That is, the actual output will vary from one run to the next. In any case, if we don't want output from different processes to appear in a random order, it's up to us to modify our program accordingly. For example, we can have each process other than 0 send its output to process 0, and process 0 can print the output in process rank order.

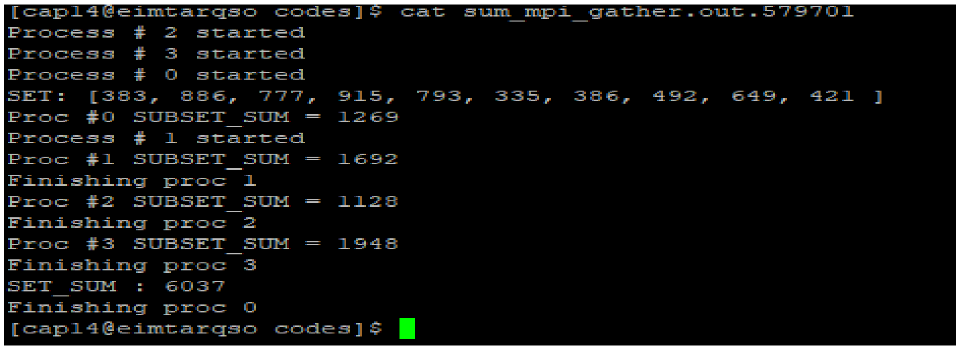
7. Provide three simple parallel programs to sum a set of random numbers based on the variants described below. Each MPI process shall generate and sum a subset of random numbers locally, and then compute the grand total by adding the local sum from each process, following the following strategies:

7.a: a "master" process receives the local sums using MPI_Send.
Attached: sum_mpi_send



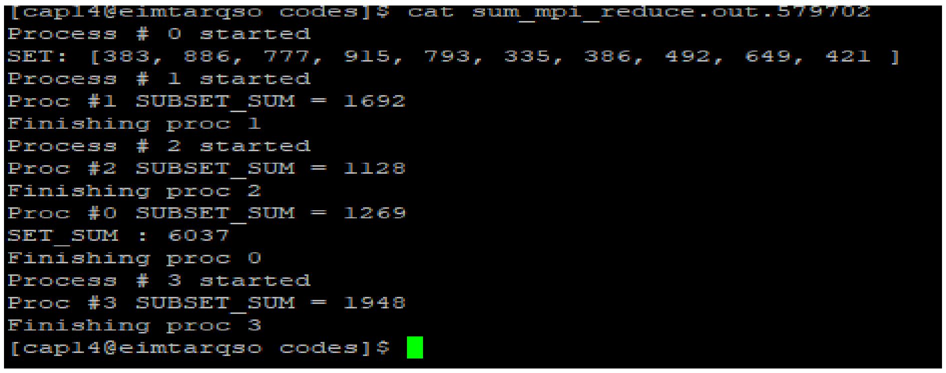
```
[capl4@eimtarqso codes]$ cat sun_mpi_send.out.579700
Process # 0 started
SET: [383, 886, 777, 915, 793, 335, 386, 492, 649, 421 ]
Process # 1 started
Proc #1 SUBSET_SUM = 1692
Finishing proc 1
Process # 2 started
Proc #2 SUBSET_SUM = 1128
Process # 3 started
Finishing proc 2
Proc #0 SUBSET_SUM = 1269
SET_SUM : 6037
Finishing proc 0
Proc #3 SUBSET_SUM = 1948
Finishing proc 3
[capl4@eimtarqso codes]$
```

7.b: a "master" process receives the local sums using MPI_Gather.
Attached: sum_mpi_gather



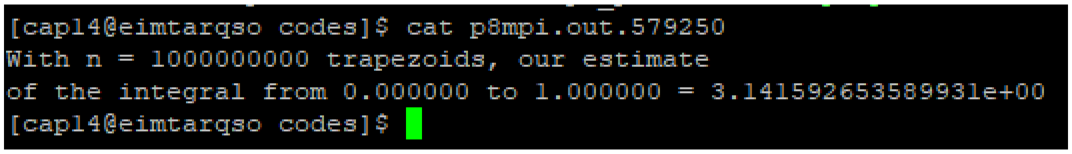
```
[capl4@eimtarqso codes]$ cat sum_mpi_gather.out.579701
Process # 2 started
Process # 3 started
Process # 0 started
SET: [383, 886, 777, 915, 793, 335, 386, 492, 649, 421 ]
Proc #0 SUBSET_SUM = 1269
Process # 1 started
Proc #1 SUBSET_SUM = 1692
Finishing proc 1
Proc #2 SUBSET_SUM = 1128
Finishing proc 2
Proc #3 SUBSET_SUM = 1948
Finishing proc 3
SET_SUM : 6037
Finishing proc 0
[capl4@eimtarqso codes]$
```

7.c: a "master" process receives the local sums using MPI_Reduce.
Attached: sum_mpi_reduce



```
[capl4@eimtarqso codes]$ cat sum_mpi_reduce.out.579702
Process # 0 started
SET: [383, 886, 777, 915, 793, 335, 386, 492, 649, 421 ]
Process # 1 started
Proc #1 SUBSET_SUM = 1692
Finishing proc 1
Process # 2 started
Proc #2 SUBSET_SUM = 1128
Finishing proc 2
Proc #0 SUBSET_SUM = 1269
SET_SUM : 6037
Finishing proc 0
Process # 3 started
Proc #3 SUBSET_SUM = 1948
Finishing proc 3
[capl4@eimtarqso codes]$
```

8. Provide a parallel version of p8.c using MPI. Explain your design decisions.



```
[capl4@eimtarqso codes]$ cat p8mpi.out.579250
With n = 1000000000 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 3.141592653589931e+00
[capl4@eimtarqso codes]$
```

Attached: p8mpi.c

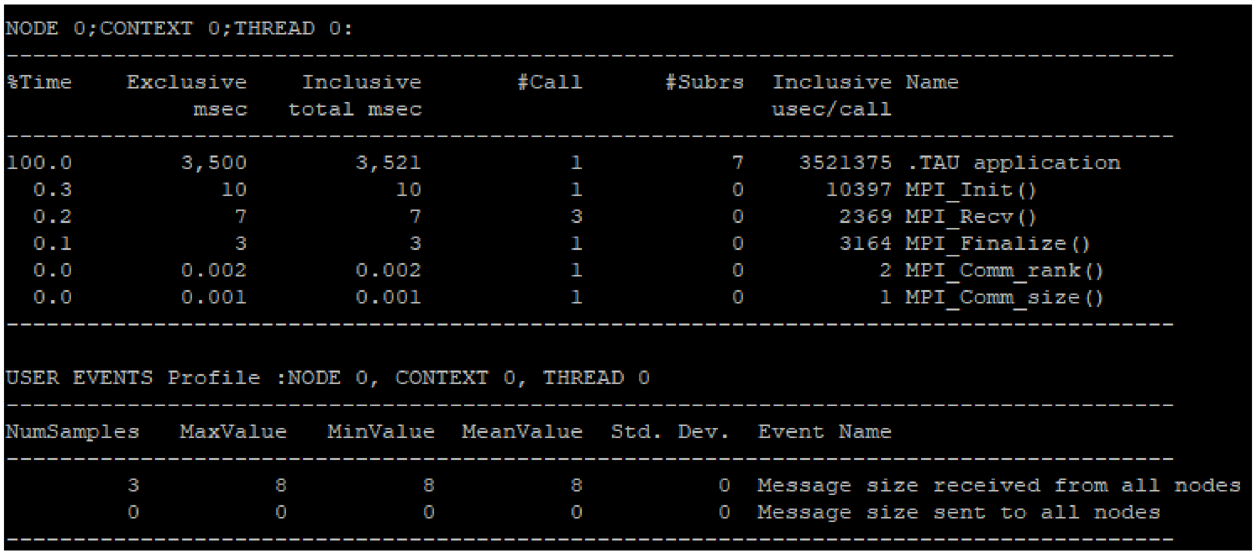
The code is hardwired in order to get the same result as in the serial version

Design:

1. Partitioning the problem solution into task
   Find the area of a single trapezoid estimating the integral of f(x) and Compute the sum of this areas.
2. Identify the communication channel
   The communication channels will join each of the trapezoidal areas (first task) with the addition (second task)
3. Aggregate the task into combined tasks thus we use more trapezoids than cores.
   A way to go is split the interval [a,b] up into comm_sz subintervals. If comm_sz evenly divides n, the number of trapezoids, we can apply the trapezoidal rule with n/comm_sz trapezoids to each of the comm_sz subinterval.
4. Map the composite task to cores
   One process adds the estimate value normally process cero

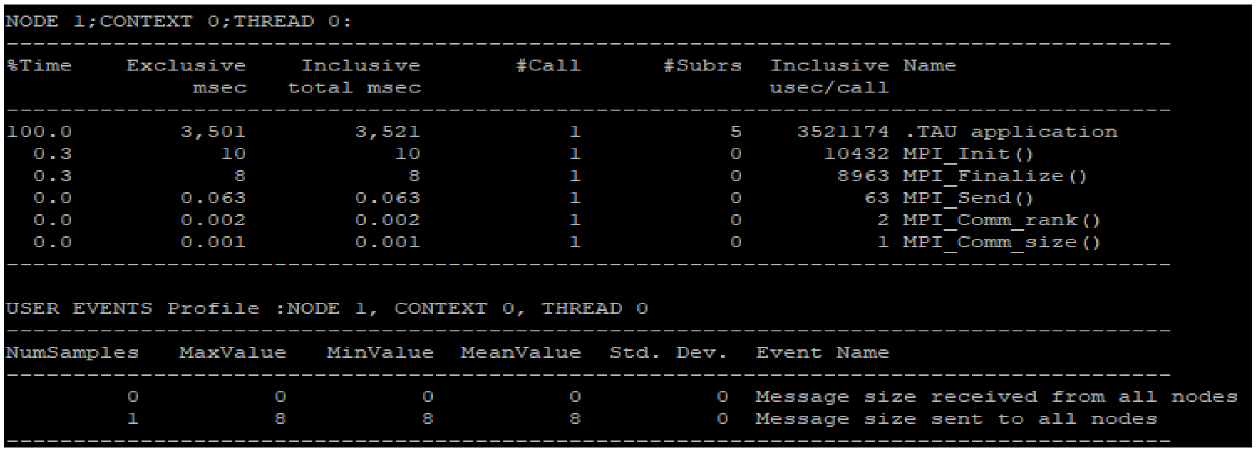
9. **Perform a brief evaluation of your parallel implementation of p8.c using the tools introduced above. You can provide a summary of statistics, screenshots, and the comments that you consider appropriate.**

Node 0

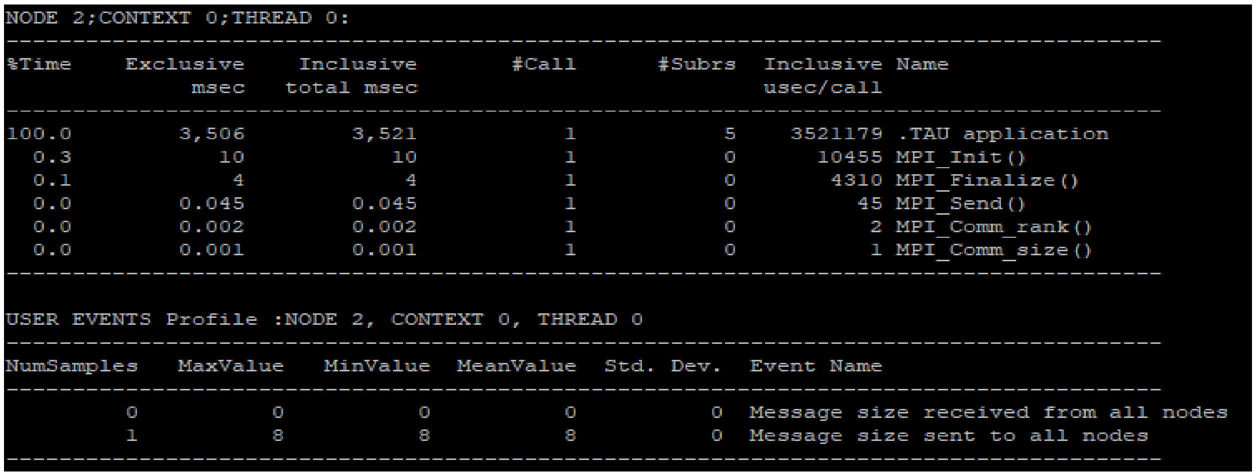


```
NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,500        3,521           1           7    3521375 .TAU application
  0.3           10           10           1           0      10397 MPI_Init()
  0.2            7            7           3           0       2369 MPI_Recv()
  0.1            3            3           1           0       3164 MPI_Finalize()
  0.0        0.002        0.002           1           0          2 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         3          8          8          8          0  Message size received from all nodes
         0          0          0          0          0  Message size sent to all nodes
---------------------------------------------------------------------------------------
```

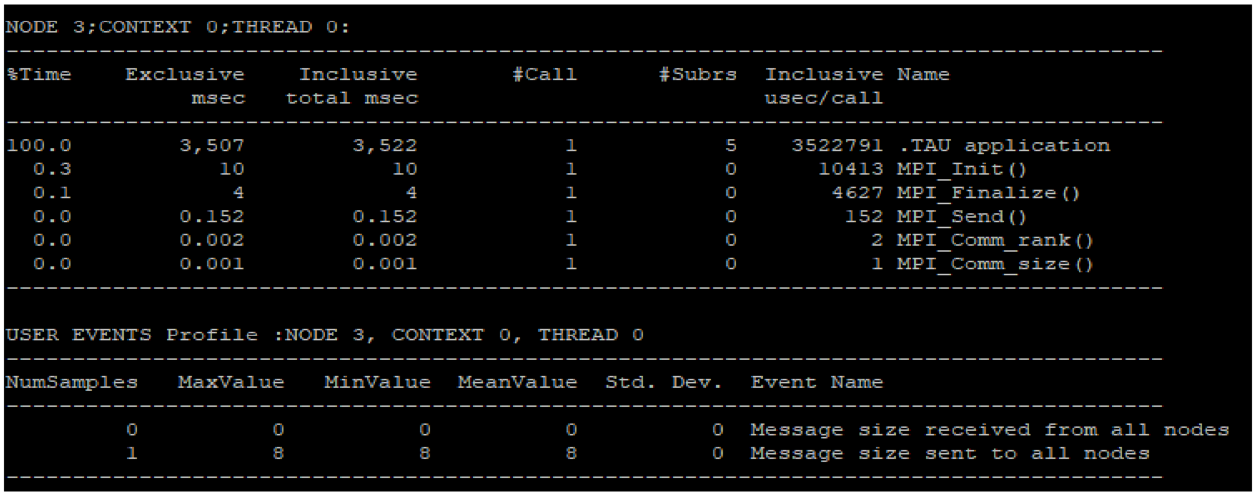
Node 1


```
NODE 1;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,501        3,521           1           5    3521174 .TAU application
  0.3           10           10           1           0      10432 MPI_Init()
  0.3            8            8           1           0       8963 MPI_Finalize()
  0.0        0.063        0.063           1           0         63 MPI_Send()
  0.0        0.002        0.002           1           0          2 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 1, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         0          0          0          0          0  Message size received from all nodes
         1          8          8          8          0  Message size sent to all nodes
---------------------------------------------------------------------------------------
```

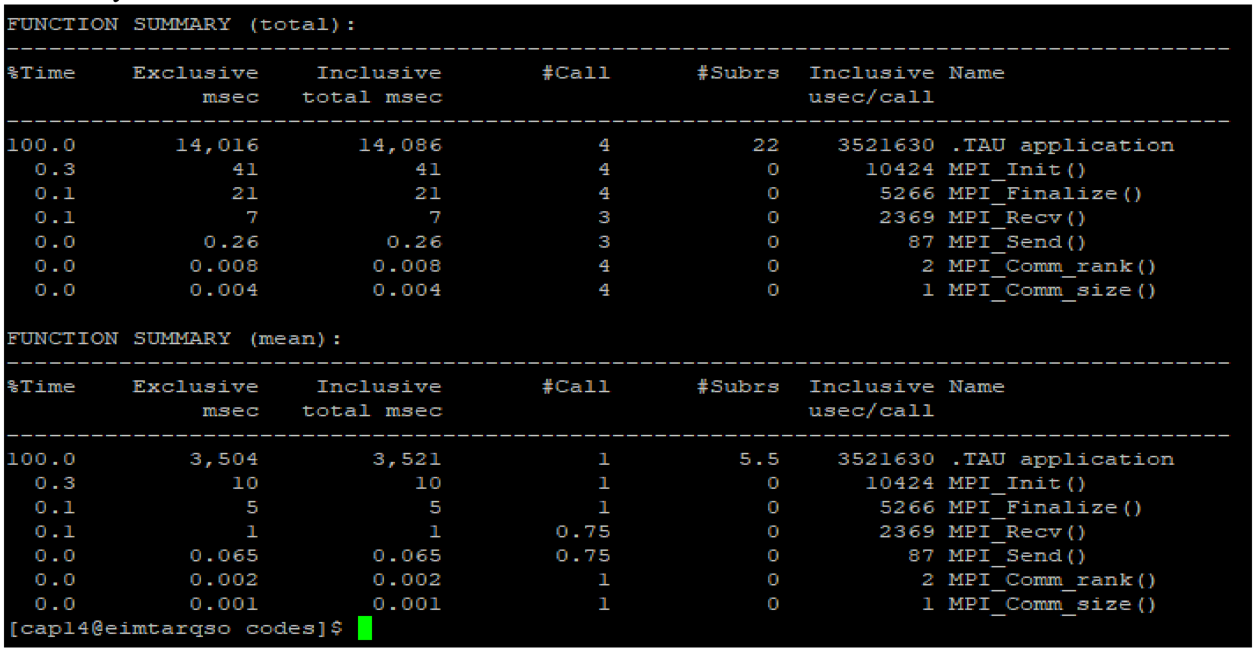
Node 2


```
NODE 2;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,506        3,521           1           5    3521179 .TAU application
  0.3           10           10           1           0      10455 MPI_Init()
  0.1            4            4           1           0       4310 MPI_Finalize()
  0.0        0.045        0.045           1           0         45 MPI_Send()
  0.0        0.002        0.002           1           0          2 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 2, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         0          0          0          0          0  Message size received from all nodes
         1          8          8          8          0  Message size sent to all nodes
---------------------------------------------------------------------------------------
```

Node 3

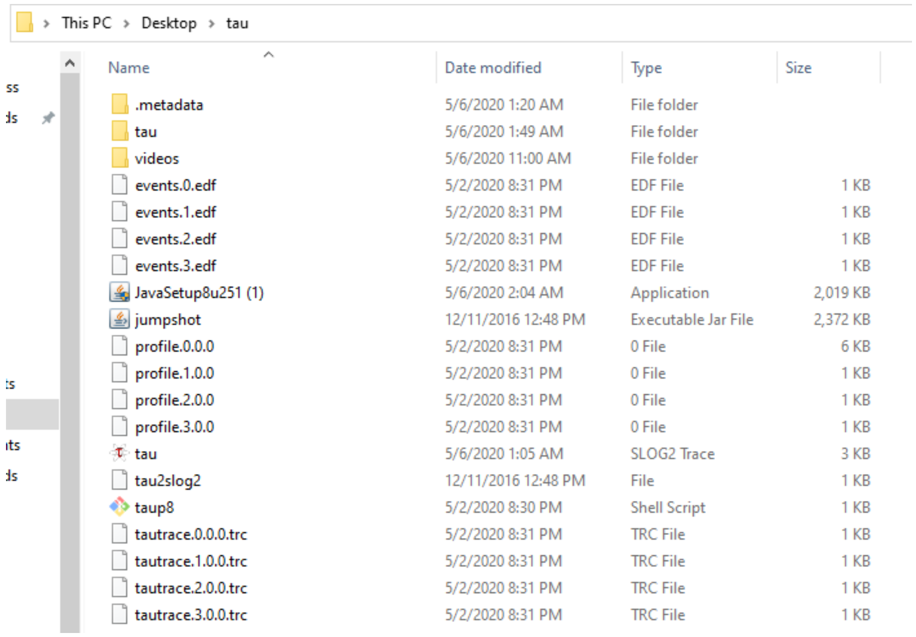

```
NODE 3;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,507        3,522           1           5    3522791 .TAU application
  0.3           10           10           1           0      10413 MPI_Init()
  0.1            4            4           1           0       4627 MPI_Finalize()
  0.0        0.152        0.152           1           0        152 MPI_Send()
  0.0        0.002        0.002           1           0          2 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 3, CONTEXT 0, THREAD 0
---------------------------------------------------------------------------------------
NumSamples   MaxValue   MinValue  MeanValue  Std. Dev.  Event Name
---------------------------------------------------------------------------------------
         0          0          0          0          0  Message size received from all nodes
         1          8          8          8          0  Message size sent to all nodes
---------------------------------------------------------------------------------------
```

Summary



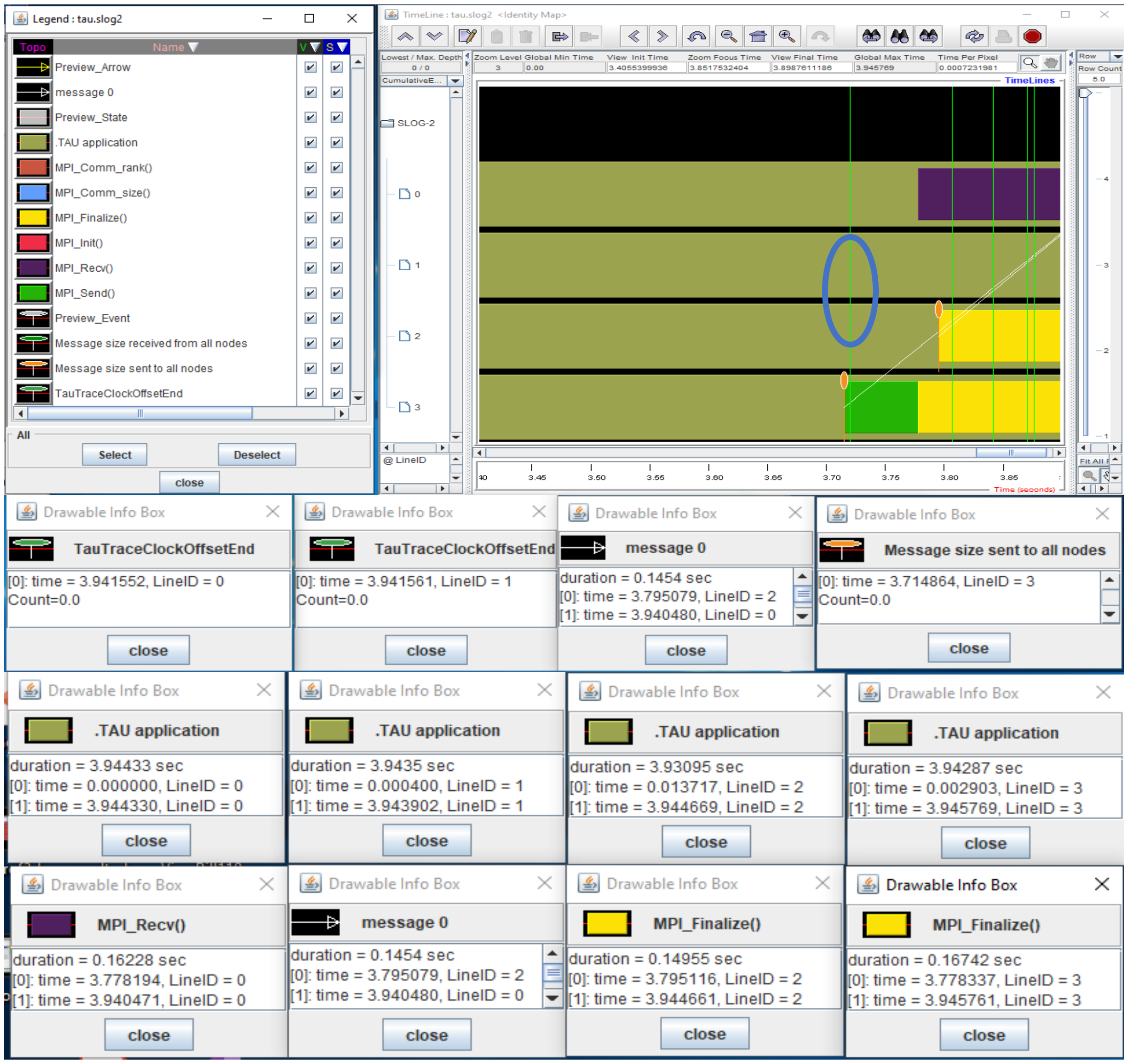
```
FUNCTION SUMMARY (total):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0       14,016       14,086           4          22    3521630 .TAU application
  0.3           41           41           4           0      10424 MPI_Init()
  0.1           21           21           4           0       5266 MPI_Finalize()
  0.1            7            7           3           0       2369 MPI_Recv()
  0.0         0.26         0.26           3           0         87 MPI_Send()
  0.0        0.008        0.008           4           0          2 MPI_Comm_rank()
  0.0        0.004        0.004           4           0          1 MPI_Comm_size()

FUNCTION SUMMARY (mean):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        3,504        3,521           1         5.5    3521630 .TAU application
  0.3           10           10           1           0      10424 MPI_Init()
  0.1            5            5           1           0       5266 MPI_Finalize()
  0.1            1            1        0.75           0       2369 MPI_Recv()
  0.0        0.065        0.065        0.75           0         87 MPI_Send()
  0.0        0.002        0.002           1           0          2 MPI_Comm_rank()
  0.0        0.001        0.001           1           0          1 MPI_Comm_size()
[cap14@eimtarqso codes]$
```

Jumpshot



This PC › Desktop › tau

Name	Date modified	Type	Size
.metadata	5/6/2020 1:20 AM	File folder	
tau	5/6/2020 1:49 AM	File folder	
videos	5/6/2020 11:00 AM	File folder	
events.0.edf	5/2/2020 8:31 PM	EDF File	1 KB
events.1.edf	5/2/2020 8:31 PM	EDF File	1 KB
events.2.edf	5/2/2020 8:31 PM	EDF File	1 KB
events.3.edf	5/2/2020 8:31 PM	EDF File	1 KB
JavaSetup8u251 (1)	5/6/2020 2:04 AM	Application	2,019 KB
jumpshot	12/11/2016 12:48 PM	Executable Jar File	2,372 KB
profile.0.0.0	5/2/2020 8:31 PM	0 File	6 KB
profile.1.0.0	5/2/2020 8:31 PM	0 File	1 KB
profile.2.0.0	5/2/2020 8:31 PM	0 File	1 KB
profile.3.0.0	5/2/2020 8:31 PM	0 File	1 KB
tau	5/6/2020 1:05 AM	SLOG2 Trace	3 KB
tau2slog2	12/11/2016 12:48 PM	File	1 KB
taup8	5/2/2020 8:30 PM	Shell Script	1 KB
tautrace.0.0.0.trc	5/2/2020 8:31 PM	TRC File	1 KB
tautrace.1.0.0.trc	5/2/2020 8:31 PM	TRC File	1 KB
tautrace.2.0.0.trc	5/2/2020 8:31 PM	TRC File	1 KB
tautrace.3.0.0.trc	5/2/2020 8:31 PM	TRC File	1 KB


Legend : tau.slog2

Topo	Name	V	S
	Preview_Arrow	✓	✓
	message 0	✓	✓
	Preview_State	✓	✓
	.TAU application	✓	✓
	MPI_Comm_rank()	✓	✓
	MPI_Comm_size()	✓	✓
	MPI_Finalize()	✓	✓
	MPI_Init()	✓	✓
	MPI_Recv()	✓	✓
	MPI_Send()	✓	✓
	Preview_Event	✓	✓
	Message size received from all nodes	✓	✓
	Message size sent to all nodes	✓	✓
	TauTraceClockOffsetEnd	✓	✓

TimeLine : tau.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel	Row Count
0 / 0	3	0.00	3.4055399936	3.8517532404	3.8987611186	3.945769	0.0007231981	5.0

Drawable Info Box — TauTraceClockOffsetEnd
[0]: time = 3.941552, LineID = 0
Count=0.0

Drawable Info Box — TauTraceClockOffsetEnd
[0]: time = 3.941561, LineID = 1
Count=0.0

Drawable Info Box — message 0
duration = 0.1454 sec
[0]: time = 3.795079, LineID = 2
[1]: time = 3.940480, LineID = 0

Drawable Info Box — Message size sent to all nodes
[0]: time = 3.714864, LineID = 3
Count=0.0

Drawable Info Box — .TAU application
duration = 3.94433 sec
[0]: time = 0.000000, LineID = 0
[1]: time = 3.944330, LineID = 0

Drawable Info Box — .TAU application
duration = 3.9435 sec
[0]: time = 0.000400, LineID = 1
[1]: time = 3.943902, LineID = 1

Drawable Info Box — .TAU application
duration = 3.93095 sec
[0]: time = 0.013717, LineID = 2
[1]: time = 3.944669, LineID = 2

Drawable Info Box — .TAU application
duration = 3.94287 sec
[0]: time = 0.002903, LineID = 3
[1]: time = 3.945769, LineID = 3

Drawable Info Box — MPI_Recv()
duration = 0.16228 sec
[0]: time = 3.778194, LineID = 0
[1]: time = 3.940471, LineID = 0

Drawable Info Box — message 0
duration = 0.1454 sec
[0]: time = 3.795079, LineID = 2
[1]: time = 3.940480, LineID = 0

Drawable Info Box — MPI_Finalize()
duration = 0.14955 sec
[0]: time = 3.795116, LineID = 2
[1]: time = 3.944661, LineID = 2

Drawable Info Box — MPI_Finalize()
duration = 0.16742 sec
[0]: time = 3.778337, LineID = 3
[1]: time = 3.945761, LineID = 3

Comment: Two concepts to take into consideration are:
Profile: Is the Statistical summary of all metrics measured shows how much total time resources each call utilized, and
Trace: Is the timeline of runtime events took place, shows when each event happened and where.
Using **jumpshot** we can notice in our interval when the message size is sent to all nodes, as well we can observe when the MPI_send begin and end notice the MPI_Recv wind-up at the ending of MPI_send. The yellow interval represents the conclusion of the MPI section and starts of the sequential part of the code, the thin green line is a few amounts of messages beside the black block represent the message 0 the TAU application run for about 4 seconds accord with the data in the drawable infobox that give information about the time that each event take to complete. Looking to the TAU summary the application is the one that takes most of the time and the MPIcomm_size the one with less time consumed, also notice 4 nodes that represent the four core required the diagonal line show when each event happened and where this is located

10. Perform a brief comparative study of the NPB benchmarks introduced above (EP, BT, and CG) using Extrae/Paraver. You can provide a summary of statistics, screenshots, and the comments that you consider appropriate.
attached:ep_bt_cg

Original benchmark:

- EP kernel Embarrassingly Parallel:

is an Embarrassingly Parallel benchmark. It generates pairs of Gaussian random deviates according to a specific scheme. The goal is to establish the reference point for the peak performance of a given platform.

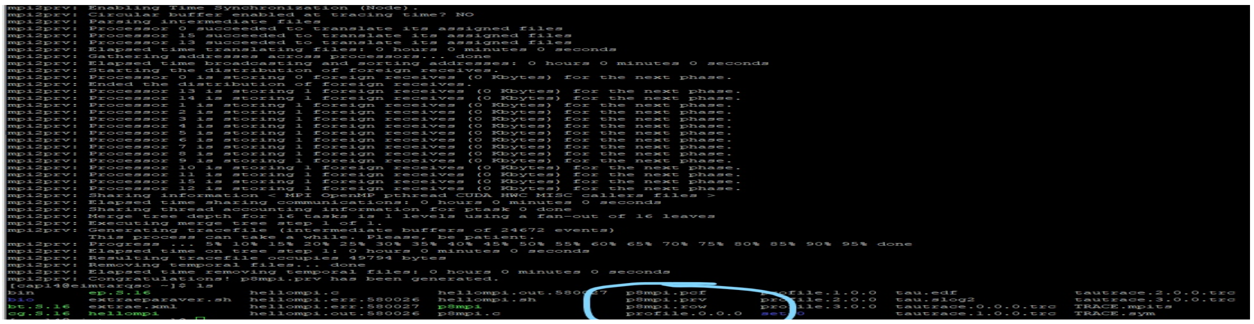
- BT pseudo application Block Tri-diagonal solver:

is a simulated CFD application that uses an implicit algorithm to solve 3-dimensional (3- D) compressible Navier-Stokes equations. The finite differences solution to the problem is based on an Alternating Direction Implicit (ADI) approximate factorization that decouples the x, y, and z dimensions. The resulting systems are Block-Tridiagonal of 5×5 blocks and are solved sequentially along each dimension.

- CG kernel that Conjugate Gradient, irregular memory access, and communication:

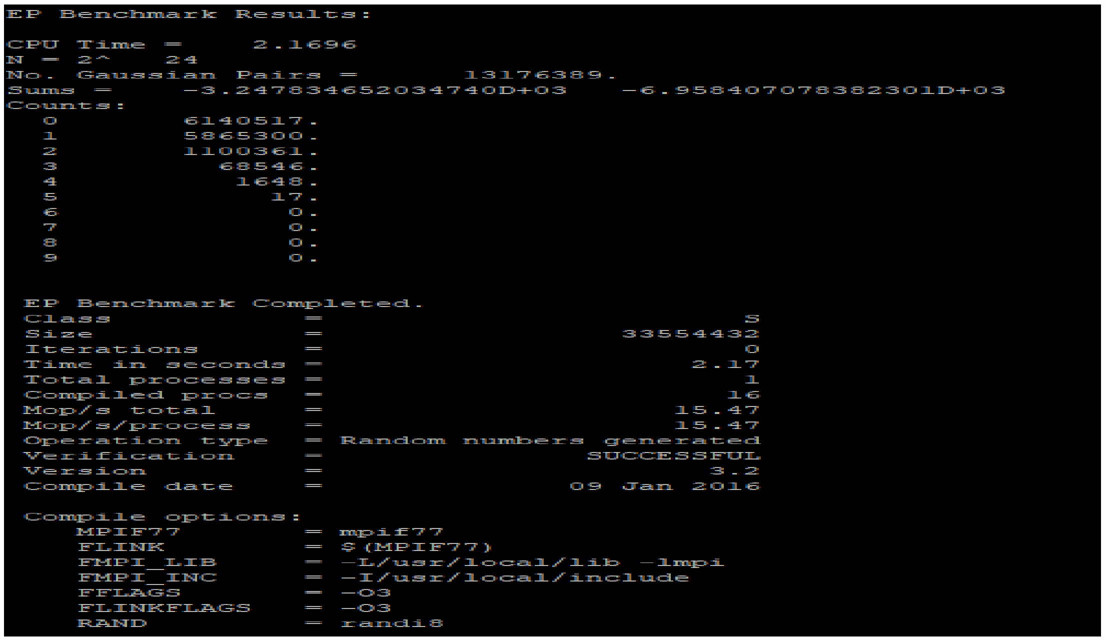
uses a Conjugate Gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, unstructured matrix. This kernel tests unstructured grid computations and communications by using a matrix with randomly generated locations of entries.

Benchmark class S: small for quick test purposes



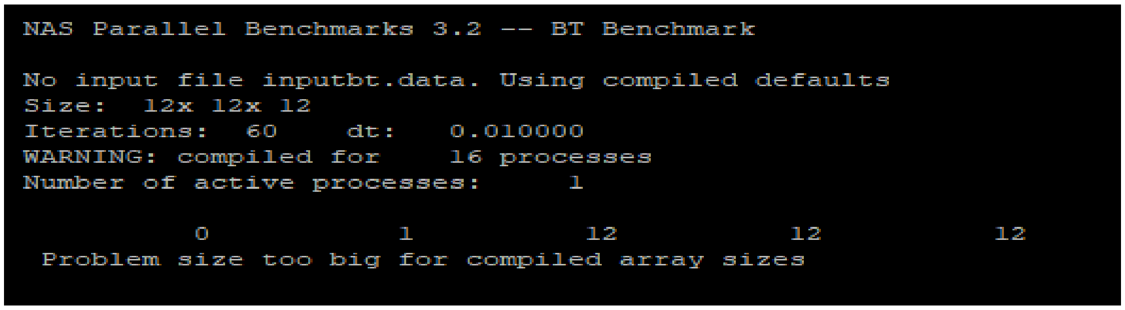
```
mpi2prv: Enabling Time Synchronization (Node).
mpi2prv: Circular buffer enabled at tracing time? NO
mpi2prv: Parsing intermediate files
mpi2prv: Processor 0 succeeded to translate its assigned files
mpi2prv: Processor 15 succeeded to translate its assigned files
mpi2prv: Processor 13 succeeded to translate its assigned files
mpi2prv: Elapsed time translating files: 0 hours 0 minutes 0 seconds
mpi2prv: Gathering addresses across processors... done
mpi2prv: Elapsed time broadcasting and sorting addresses: 0 hours 0 minutes 0 seconds
mpi2prv: Starting the distribution of foreign receives.
mpi2prv: Processor 0 is storing 0 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Ended the distribution of foreign receives.
mpi2prv: Processor 13 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 14 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 1 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 2 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 3 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 4 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 5 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 6 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 7 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 8 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 9 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 10 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 11 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 15 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Processor 12 is storing 1 foreign receives (0 Kbytes) for the next phase.
mpi2prv: Sharing information < MPI OpenMP pthread CUDA HWC MISC callers files >
mpi2prv: Elapsed time sharing communications: 0 hours 0 minutes 0 seconds
mpi2prv: Sharing thread accounting information for ptask 0 done
mpi2prv: Merge tree depth for 16 tasks is 1 levels using a fan-out of 16 leaves
mpi2prv: Executing merge tree step 1 of 1.
mpi2prv: Generating tracefile (intermediate buffers of 24672 events)
         This process can take a while. Please, be patient.
mpi2prv: Progress ... 5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% done
mpi2prv: Elapsed time on tree step 1: 0 hours 0 minutes 0 seconds
mpi2prv: Resulting tracefile occupies 49794 bytes
mpi2prv: Removing temporal files... done
mpi2prv: Elapsed time removing temporal files: 0 hours 0 minutes 0 seconds
mpi2prv: Congratulations! p8mpi.prv has been generated.
[capl4@eimtarqso ~]$ ls
bin       ep.S.16             hellompi.c             hellompi.out.580027  p8mpi.pcf        profile.1.0.0  tau.edf                  tautrace.2.0.0.trc
bio       extraeparaver.sh    hellompi.err.580026    hellompi.sh          p8mpi.prv        profile.2.0.0  tau.slog2                tautrace.3.0.0.trc
bt.S.16   extrae.xml          hellompi.err.580027    p8mpi                p8mpi.row        profile.3.0.0  tautrace.0.0.0.trc       TRACE.mpits
cg.S.16   hellompi            hellompi.out.580026    p8mpi.c              profile.0.0.0    set-0          tautrace.1.0.0.trc       TRACE.sym
```

EP Benchmark Result



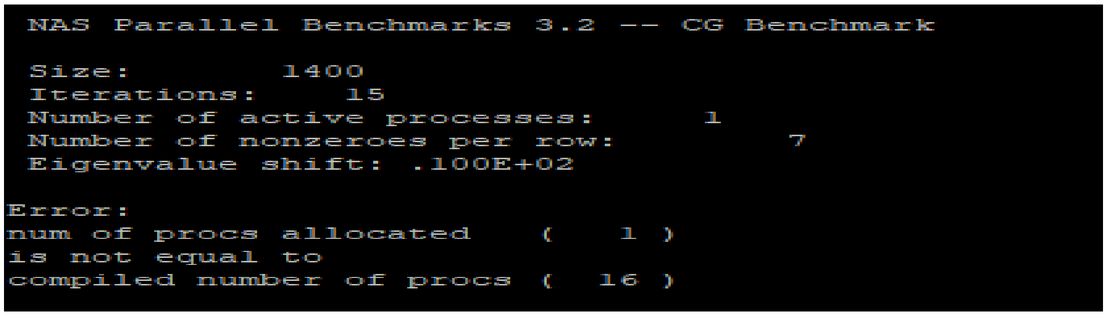
```
EP Benchmark Results:

CPU Time =      2.1696
N = 2^    24
No. Gaussian Pairs =        13176389.
Sums =      -3.247834652034740D+03    -6.958407078382301D+03
Counts:
  0         6140517.
  1         5865300.
  2         1100361.
  3           68546.
  4            1648.
  5              17.
  6               0.
  7               0.
  8               0.
  9               0.


 EP Benchmark Completed.
 Class           =                        S
 Size            =                 33554432
 Iterations      =                        0
 Time in seconds =                     2.17
 Total processes =                        1
 Compiled procs  =                       16
 Mop/s total     =                    15.47
 Mop/s/process   =                    15.47
 Operation type  = Random numbers generated
 Verification    =               SUCCESSFUL
 Version         =                      3.2
 Compile date    =              09 Jan 2016

 Compile options:
    MPIF77          = mpif77
    FLINK           = $(MPIF77)
    FMPI_LIB        = -L/usr/local/lib -lmpi
    FMPI_INC        = -I/usr/local/include
    FFLAGS          = -O3
    FLINKFLAGS      = -O3
    RAND            = randi8
```

BT Benchmark Result



```
 NAS Parallel Benchmarks 3.2 -- BT Benchmark

 No input file inputbt.data. Using compiled defaults
 Size:  12x 12x 12
 Iterations:  60    dt:   0.010000
 WARNING: compiled for    16 processes
 Number of active processes:     1

           0           1          12          12          12
  Problem size too big for compiled array sizes
```

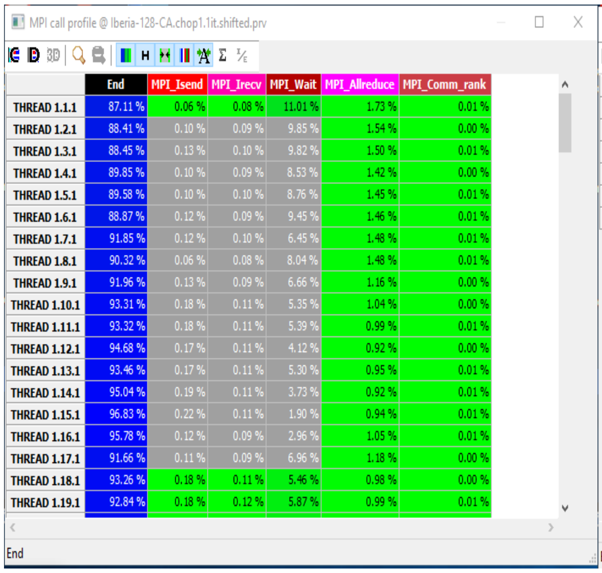
CG Benchmark Result



```
 NAS Parallel Benchmarks 3.2 -- CG Benchmark

 Size:        1400
 Iterations:     15
 Number of active processes:      1
 Number of nonzeroes per row:         7
 Eigenvalue shift: .100E+02

Error:
num of procs allocated    (    1 )
is not equal to
compiled number of procs (   16 )
```

Paraver Implementation Results

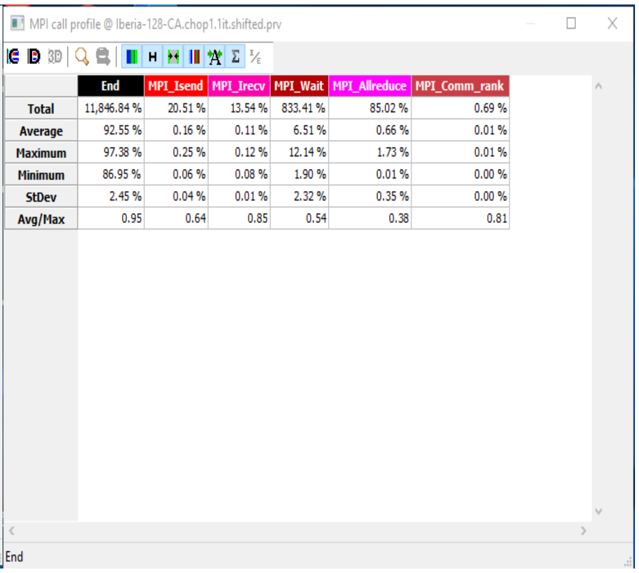
Parallel Efficiency MPI Call Profile Table 1



MPI call profile @ Iberia-128-CA.chop1.1it.shifted.prv

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Comm_rank
THREAD 1.1.1	87.11 %	0.06 %	0.08 %	11.01 %	1.73 %	0.01 %
THREAD 1.2.1	88.41 %	0.10 %	0.09 %	9.85 %	1.54 %	0.00 %
THREAD 1.3.1	88.45 %	0.13 %	0.10 %	9.82 %	1.50 %	0.01 %
THREAD 1.4.1	89.85 %	0.10 %	0.09 %	8.53 %	1.42 %	0.00 %
THREAD 1.5.1	89.58 %	0.10 %	0.10 %	8.76 %	1.45 %	0.01 %
THREAD 1.6.1	88.87 %	0.12 %	0.09 %	9.45 %	1.46 %	0.01 %
THREAD 1.7.1	91.85 %	0.12 %	0.10 %	6.45 %	1.48 %	0.01 %
THREAD 1.8.1	90.32 %	0.06 %	0.08 %	8.04 %	1.48 %	0.01 %
THREAD 1.9.1	91.96 %	0.13 %	0.09 %	6.66 %	1.16 %	0.00 %
THREAD 1.10.1	93.31 %	0.18 %	0.11 %	5.35 %	1.04 %	0.00 %
THREAD 1.11.1	93.32 %	0.18 %	0.11 %	5.39 %	0.99 %	0.01 %
THREAD 1.12.1	94.68 %	0.17 %	0.11 %	4.12 %	0.92 %	0.00 %
THREAD 1.13.1	93.46 %	0.17 %	0.11 %	5.30 %	0.95 %	0.01 %
THREAD 1.14.1	95.04 %	0.19 %	0.11 %	3.73 %	0.92 %	0.01 %
THREAD 1.15.1	96.83 %	0.22 %	0.11 %	1.90 %	0.94 %	0.01 %
THREAD 1.16.1	95.78 %	0.12 %	0.09 %	2.96 %	1.05 %	0.01 %
THREAD 1.17.1	91.66 %	0.11 %	0.09 %	6.96 %	1.18 %	0.00 %
THREAD 1.18.1	93.26 %	0.18 %	0.11 %	5.46 %	0.98 %	0.00 %
THREAD 1.19.1	92.84 %	0.18 %	0.12 %	5.87 %	0.99 %	0.01 %

End

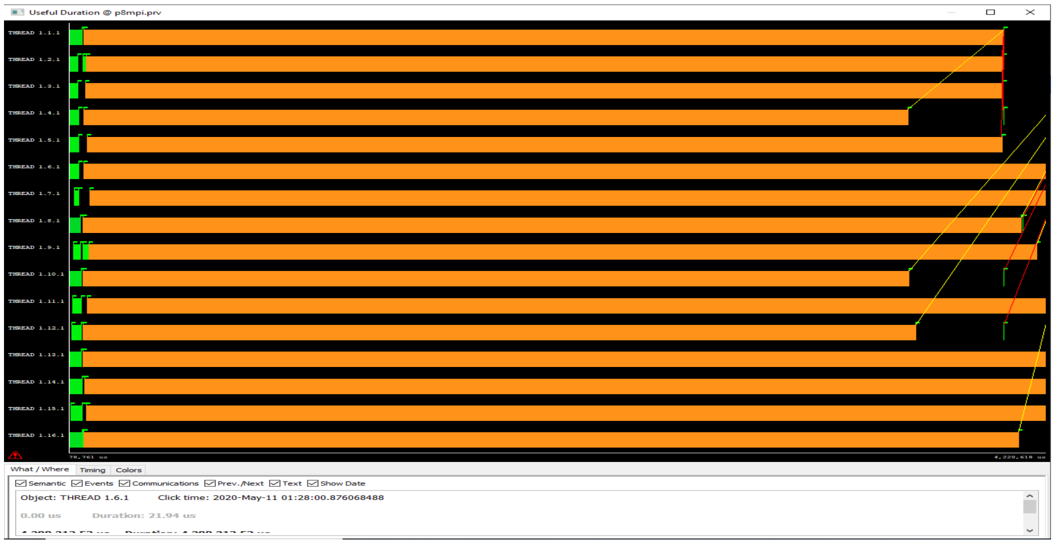


MPI call profile @ Iberia-128-CA.chop1.1it.shifted.prv

	End	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Allreduce	MPI_Comm_rank
Total	11,846.84 %	20.51 %	13.54 %	833.41 %	85.02 %	0.69 %
Average	92.55 %	0.16 %	0.11 %	6.51 %	0.66 %	0.01 %
Maximum	97.38 %	0.25 %	0.12 %	12.14 %	1.73 %	0.01 %
Minimum	86.95 %	0.06 %	0.08 %	1.90 %	0.01 %	0.00 %
StDev	2.45 %	0.04 %	0.01 %	2.32 %	0.35 %	0.00 %
Avg/Max	0.95	0.64	0.85	0.54	0.38	0.81

End

Computation Time distribution Histogram Fig 1

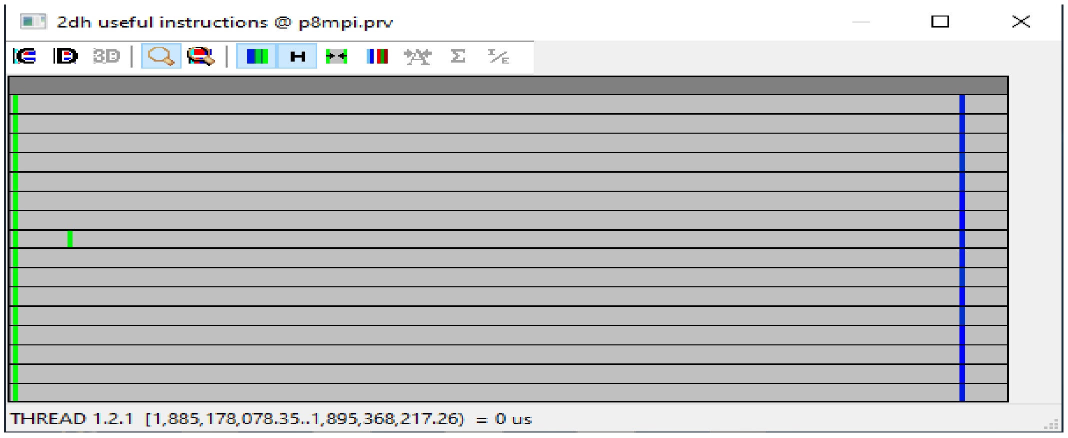


☑ Semantic ☑ Events ☑ Communications ☑ Prev./Next ☑ Text ☑ Show Date

Object: THREAD 1.6.1 Click time: 2020-May-11 01:28:00.876068488

0.00 us Duration: 21.94 us

Computational Load (instruction) distribution Histogram Fig 2



THREAD 1.2.1 [1,885,178,078.35..1,895,368,217.26) = 0 us

Comments:

- Table one (1) show us that the parallel efficiency global load balance and the communication efficiency are not lower than 85%
- Histogram fig 1 here we observe communication line and event flags as well the duration of the computation regions that are delimited by the exit from an MPI call to the entry to the next call this one looks balanced.
- Histogram fig 2 gives us a graph of the instruction for the computation regions, which are delimited by the exit from an MPI call and the entry to the next MPI call. This one looks balanced.
- For **CG** the eigenvalue is determined as part of the initial setup of each class in our case is 1.00E+04 this kernel is called by the CG benchmark 7 times to approximate a solution with the desire precision.
- For **EP** that measure the computer performance of the underlying node the generated gaussian pair is 13176389
- For **BT** is an application benchmark, which solves three sets of uncoupled systems of equations, first in the x dimension, then in the y dimension, and finally in the z dimension **we** use class S and the smallest size 12x12x12
- BT is the one with more interactions requires, therefore, more bandwidth
- **Outside MPI** efficiency = 92.55 % , Communication efficiency = 97.38 Load balance = 0.95

Bibliography and website used:

“Parallel programming author Peter Pacheco “
https://www.nas.nasa.gov/publications/npb.html
https://tools.bsc.es/extrae
https://tools.bsc.es/paraver.
https://www.cs.uoregon.edu/research/tau/home.php
visual studio 2019