home assignment #1

The work was written by

Shahaf Zohar 205978000

1. Our code:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "mpi.h"

#include <math.h>

int main(int argc, char \*argv[])

{

int myid,numprocs,i,namelen,n,flag=0;

int totalPoints = 0;

int pointsInsideCircle = 0;

double x,y,pi,mypi,Stime,pi\_estimate;

char processor\_name[MPI\_MAX\_PROCESSOR\_NAME];

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numprocs);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);

MPI\_Get\_processor\_name(processor\_name, &namelen);

while(!flag){

if(myid ==0){

if (totalPoints == 0) totalPoints=10000000; else totalPoints=0;

Stime = MPI\_Wtime();

}

MPI\_Bcast(&totalPoints, 1, MPI\_INT,0, MPI\_COMM\_WORLD);

if (totalPoints == 0) flag = 1;

else{

for(i = myid+1; i < totalPoints; i+=numprocs){

x = (double)rand() / RAND\_MAX; // Random number between 0 and 1

y = (double)rand() / RAND\_MAX; // Random number between 0 and 1

// Check if the point (x, y) is inside the unit circle

if(x\*x + y\*y <= 1){

pointsInsideCircle++;

}

}

// Estimate pi using the Monte carlo method

pi\_estimate = 4.0 \* pointsInsideCircle/totalPoints;

MPI\_Reduce(&pi\_estimate,&pi,1,MPI\_DOUBLE,MPI\_SUM,0,MPI\_COMM\_WORLD);

if(myid == 0){

printf("Estimated value of pi: %.6f\nThe clock time %f.\n",pi ,MPI\_Wtime()-Stime);

fflush(stdout);

}

}

}

MPI\_Finalize();

return 0;

}

1. A brief explanation of how to run the code:

Compilation of the program:

mpicc hw1.c -o a

Running the program after compiling and selecting several processes:

mpirun -np <number\_of\_processes> ./a

Using Scalasca

A performance analysis tool for parallel programs:

scalasca -instrument mpicc -o cpi\_scalasca ./hw1.c

Running the program with Scalasca and several processes can be added:

scalasca -analyze mpirun -np <number\_of\_processes> ./cpi\_scalasca

After running the previous command, a Scalasca file is created. This command runs the Scalasca file:

scalasca -examine ./scorep\_cpi\_scalasca\_<number\_of\_processes>\_sum

Using jumpshot:

Compilation of the program:

tau\_cc.sh -o test\_tau ./test.c

Running the program after compiling and selecting several processes:

mpirun -np <number\_of\_processes> ./test2\_tau

Run paraprof

paraprof

close and run this command to prompt jumpshot

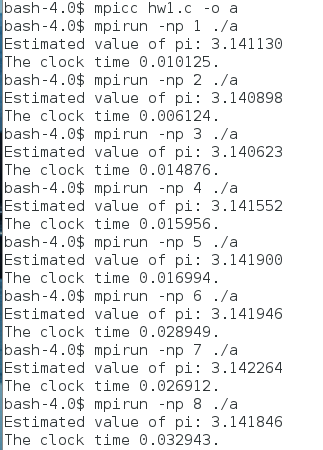
tau\_treemerge.pl

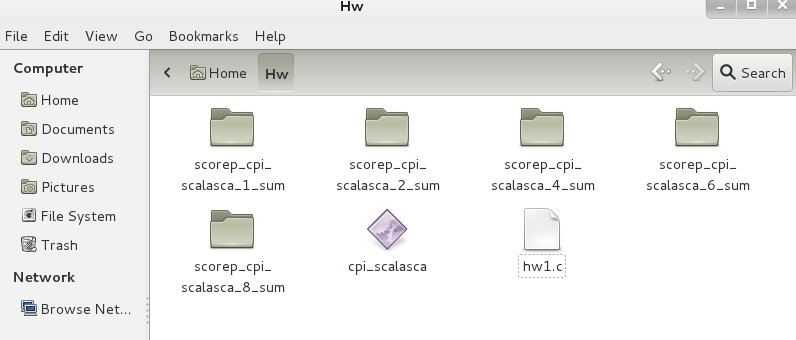
tau2slog2 tau.trc tau.edf

jumpshot ./tau.slog2

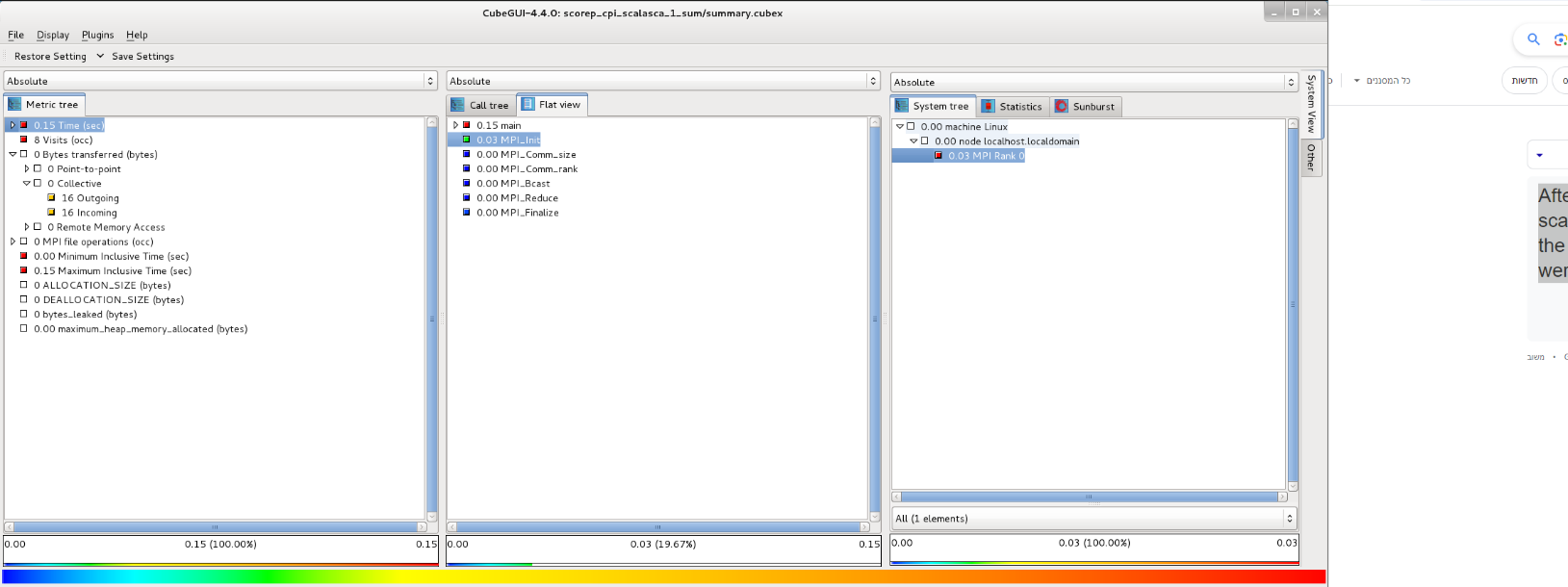
1. The plots that are requested in item D above:

Plot without Scalasca

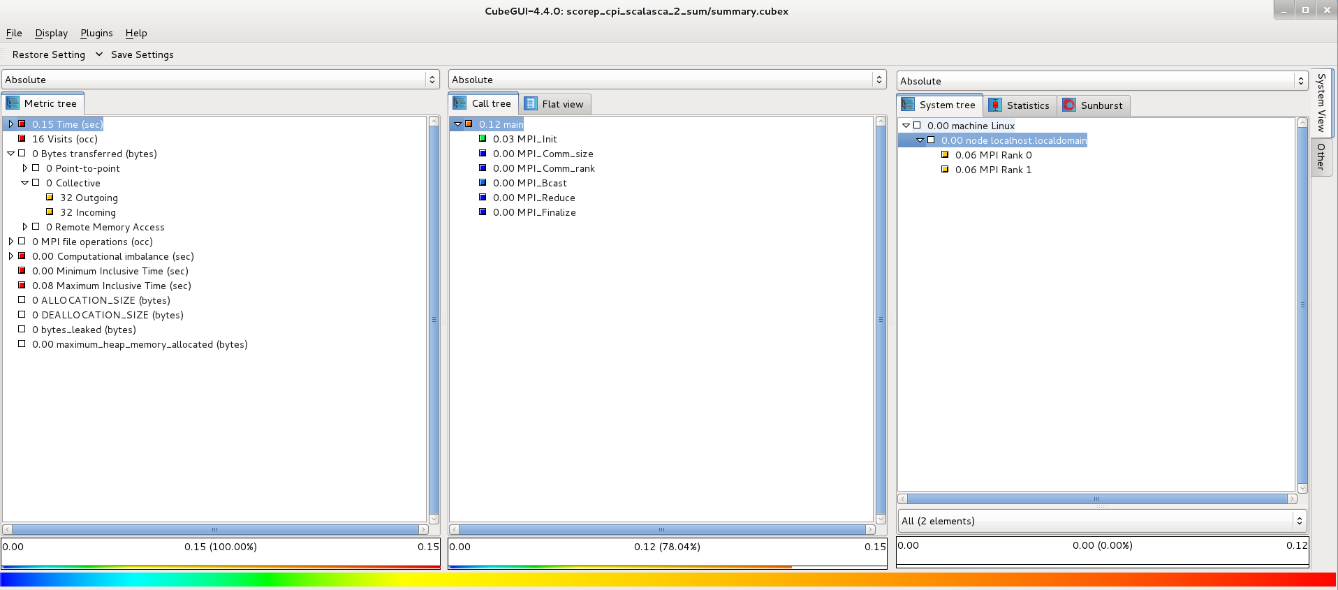


After running the Scalasca commands the following files were created:

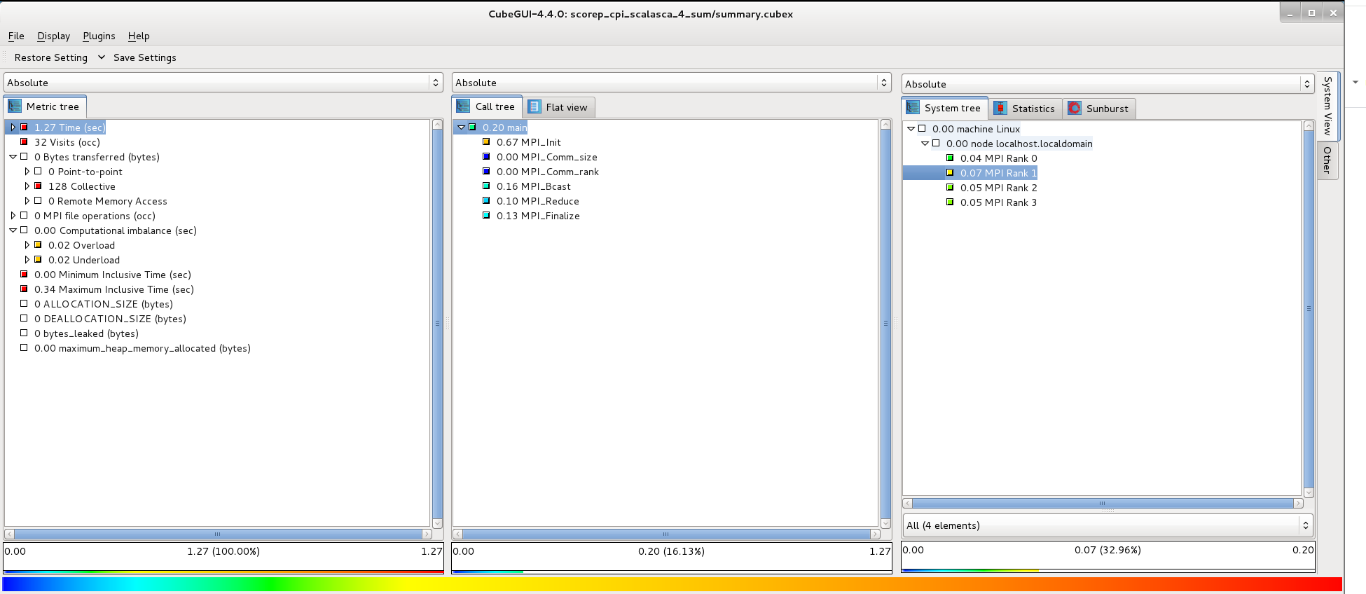
For one process:



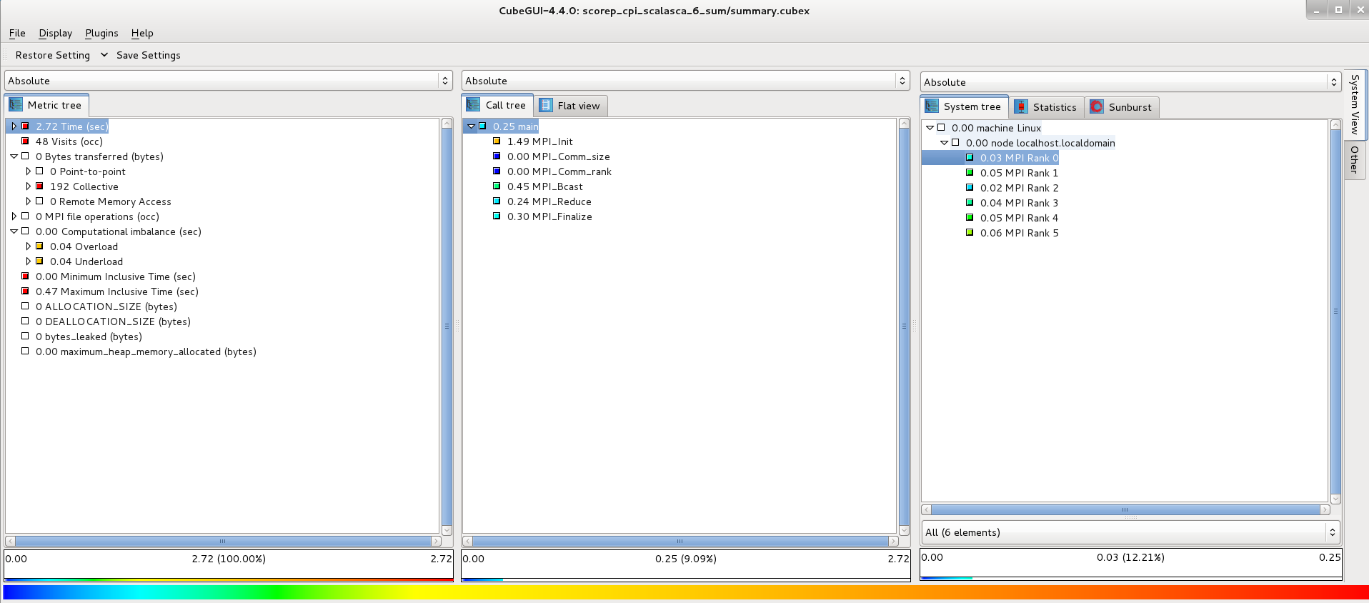
For two processes:



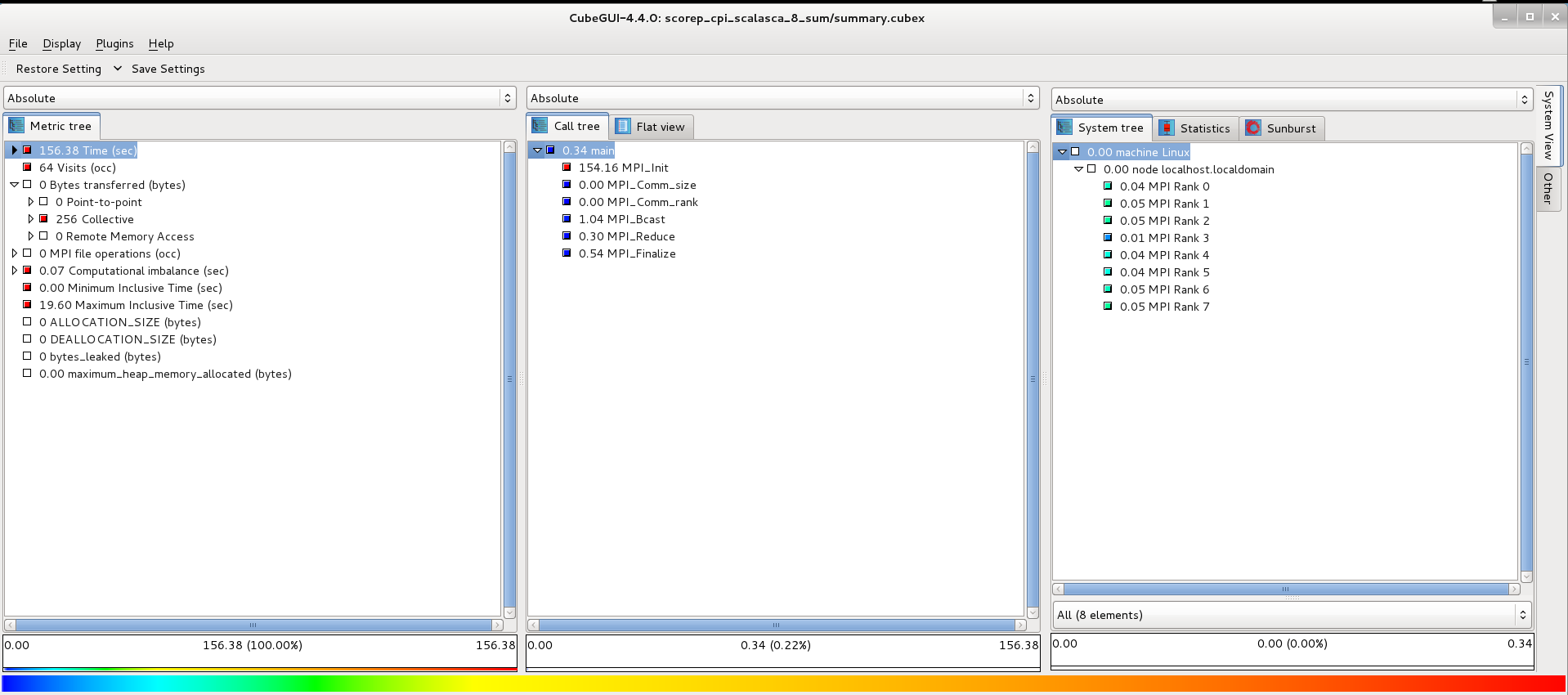
for four processes:



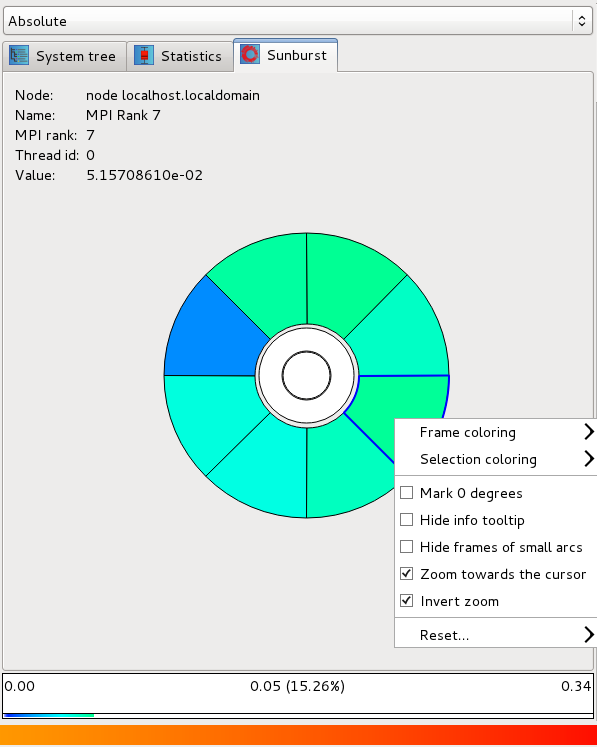
For six processes:

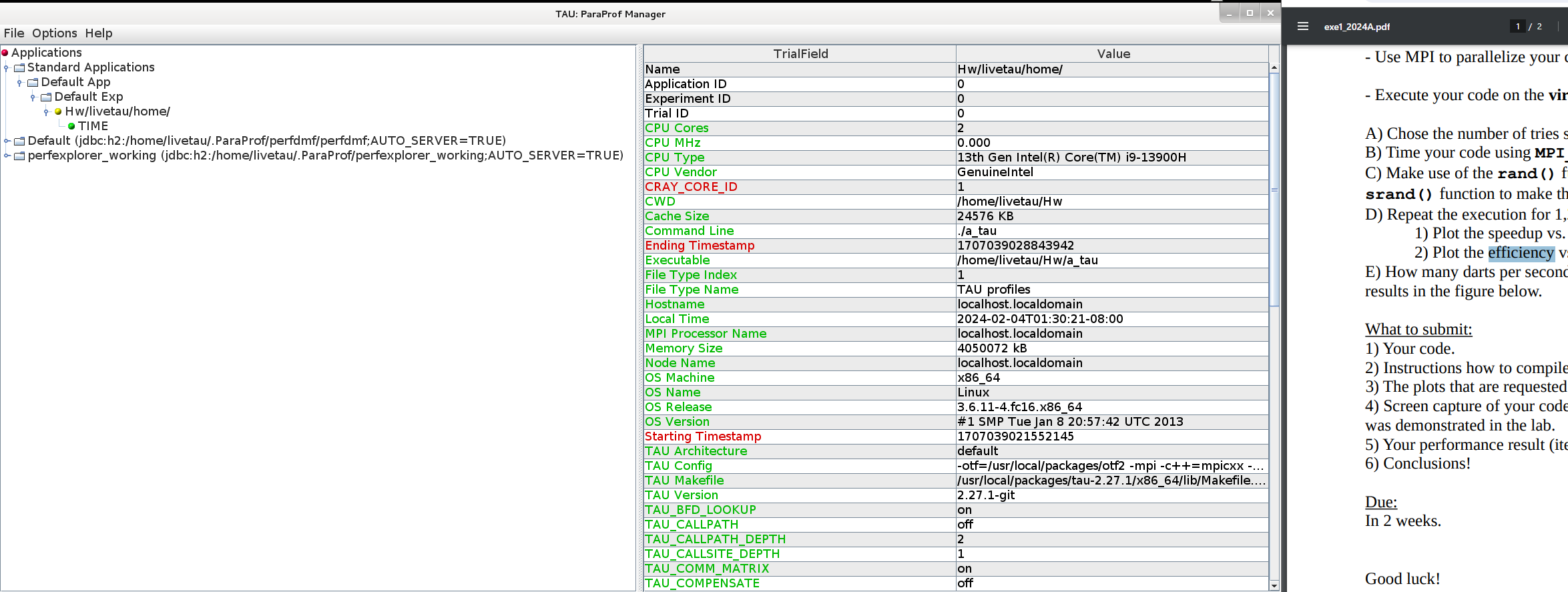


For eight processes:



An example that can be given for each of the processors:



1. Data about the system

A blue and white striped object

Description automatically generatedAnalysis on the system for 8 example processes:

A graph with a line

Description automatically generated

A screenshot of a computer

Description automatically generatedAdditional disconnections on the system with the help of jumpshot

