Events . Training

INTRODUCTION TO HYBRID PROGRAMMING IN HPC

דוגמה לאקטואליות הנושא

Enterprises & SME Research & Science

Introduction to Hybrid Programming in HPC

Date:	2018, Tuesday June 19
Organizer:	HLRS
Location:	HLRS, Room 0.438 / Rühle Saal, University of Stuttgart, Nobelstr. 19, D-70569 Stuttgart, Germany
Tags:	Parallel Programming (PAR) MPI OpenMP Training English

OVERVIEW

Most HPC systems are clusters of shared memory nodes. Such SMP nodes can be small multi-core CPUs up to large many-core CPUs. Parallel programming may combine the distributed memory parallelization on the node interconnect (e.g., with MPI) with the shared memory parallelization inside of each node (e.g., with OpenMP or MPI-3.0 shared memory). This course analyzes the strengths and weaknesses of several parallel programming models on clusters of SMP nodes. Multi-socket-multi-core systems in highly parallel environments are given special consideration. MPI-3.0 has introduced a new shared memory programming interface, which can be combined with inter-node MPI communication. It can be used for direct neighbor accesses similar to OpenMP or for direct halo copies, and enables new hybrid programming models. These models are compared with various hybrid MPI+OpenMP approaches and pure MPI. Numerous case studies and micro-benchmarks demonstrate the performance-related aspects of hybrid programming.

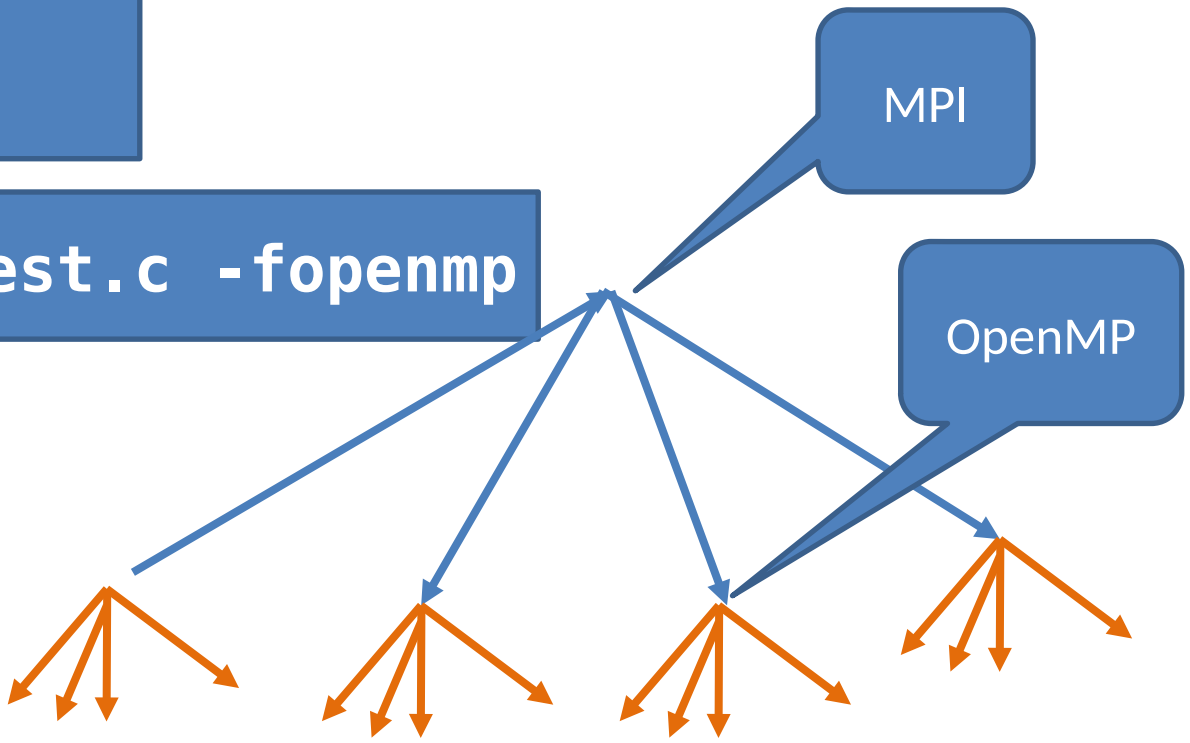
Tools for hybrid programming such as thread/process placement support and performance analysis are presented in a "how-to" section. This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves.

Hybrid MPI + OpenMP Demo

Machine File:
Node1
Node2
Node3
node4

Each node has 8 cores

```
mpicc -o mpi_out mpi_test.c -fopenmp
```



Demo: cd /home/telzur/Documents/Teaching/PP2XXXXX/lectures/10/code
program name: hybridpi.c

```
mpicc -o mpi_exe mpi_test.c -fopenmp
```

```
export OMP_NUM_THREADS=8 (bash)
setenv OMP_NUM_THREADS 8 (csh)
```

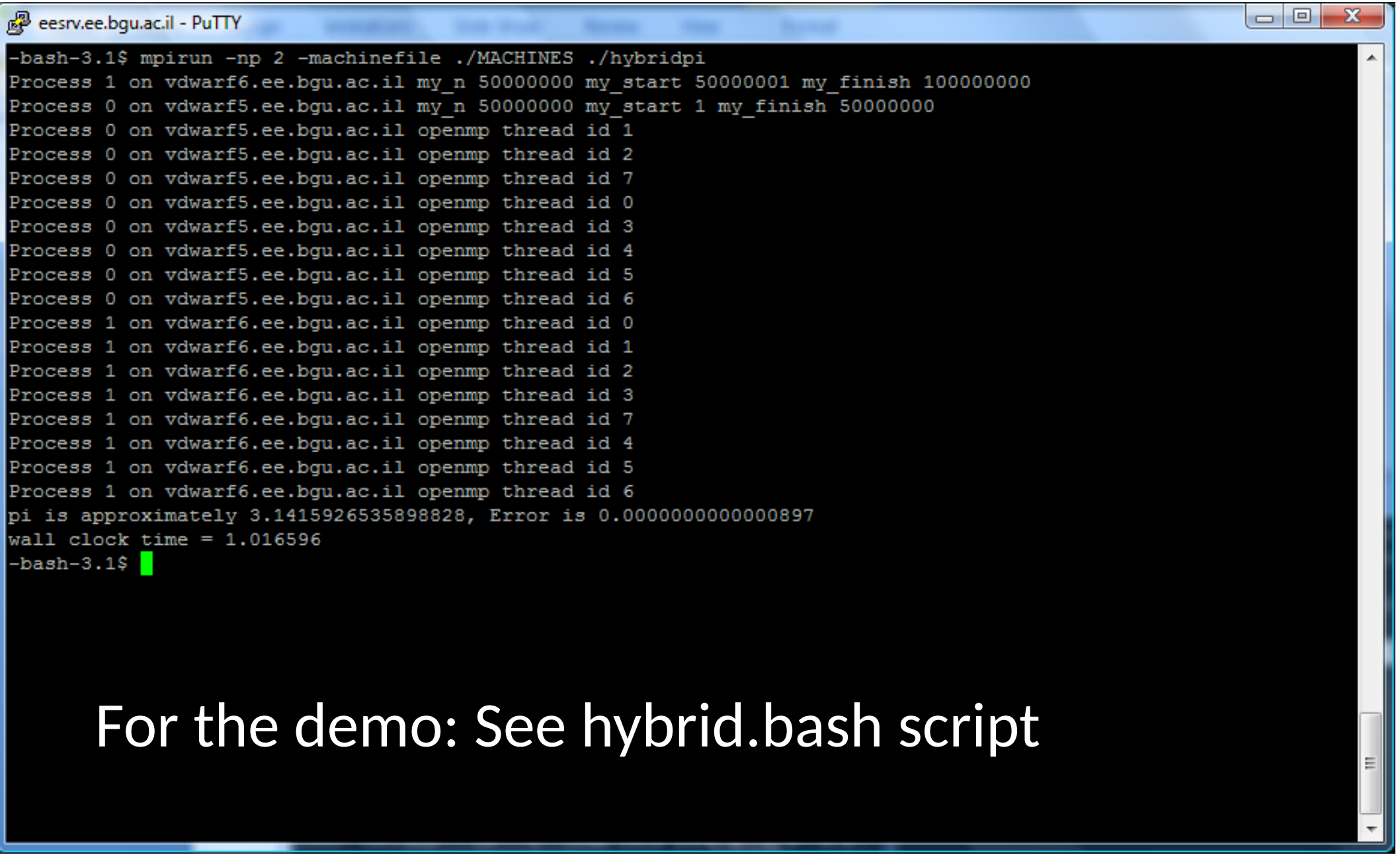
```
mpirun  -np  4  –machinefile ./machines mpi_exe
```

Hybrid Pi (MPI+OpenMP)

```
#include <stdio.h>
#include <mpi.h>
#include <omp.h>
#define NBIN 100000
#define MAX_THREADS 8

int main(int argc,char **argv) {
        int nbin,myid,nproc,nthreads,tid;
        double step,sum[MAX_THREADS]={0.0},pi=0.0,pig;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&myid);
        MPI_Comm_size(MPI_COMM_WORLD,&nproc);
        nbin = NBIN/nproc;
        step = 1.0/(nbin*nproc);
```

```
#pragma omp parallel private(tid)
    {
        int i;
        double x;
        nthreads = omp_get_num_threads();
        tid = omp_get_thread_num();
        for (i=nbin*myid+tid; i<nbin*(myid+1); i+=nthreads) {
            x = (i+0.5)*step;
            sum[tid] += 4.0/(1.0+x*x);
        }
        printf("rank tid sum = %d %d %e\n",myid,tid,sum[tid]);
    }
    for(tid=0; tid<nthreads; tid++)
        pi += sum[tid]*step;
    MPI_Allreduce(&pi,&pig,1,MPI_DOUBLE,MPI_SUM,MPI_COMM_WORLD);
    if (myid==0) printf("PI = %f\n",pig);
    MPI_Finalize();
    return 0;  }
```

Hybrid MPI+OpenMP continued

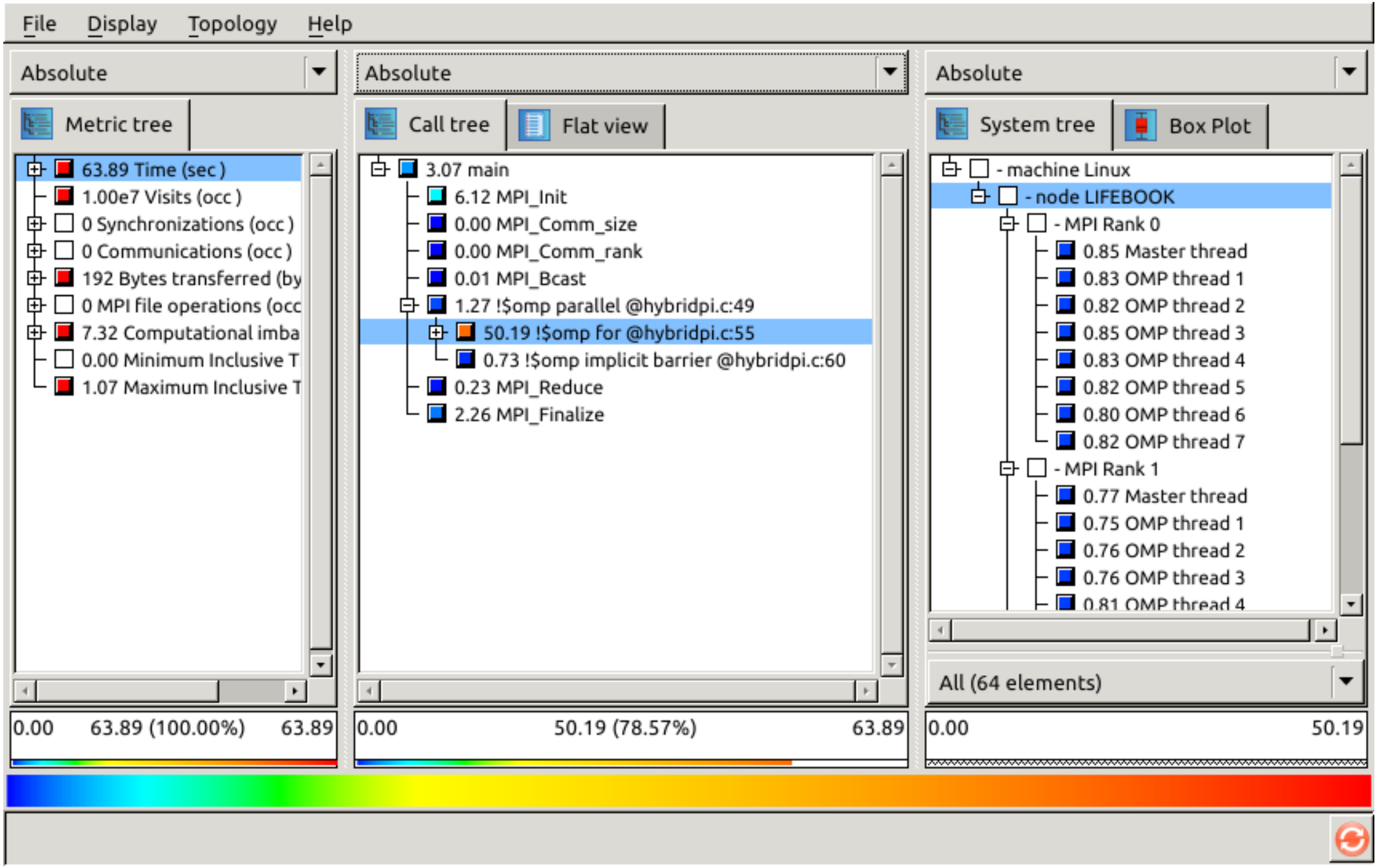


For the demo: See hybrid.bash script

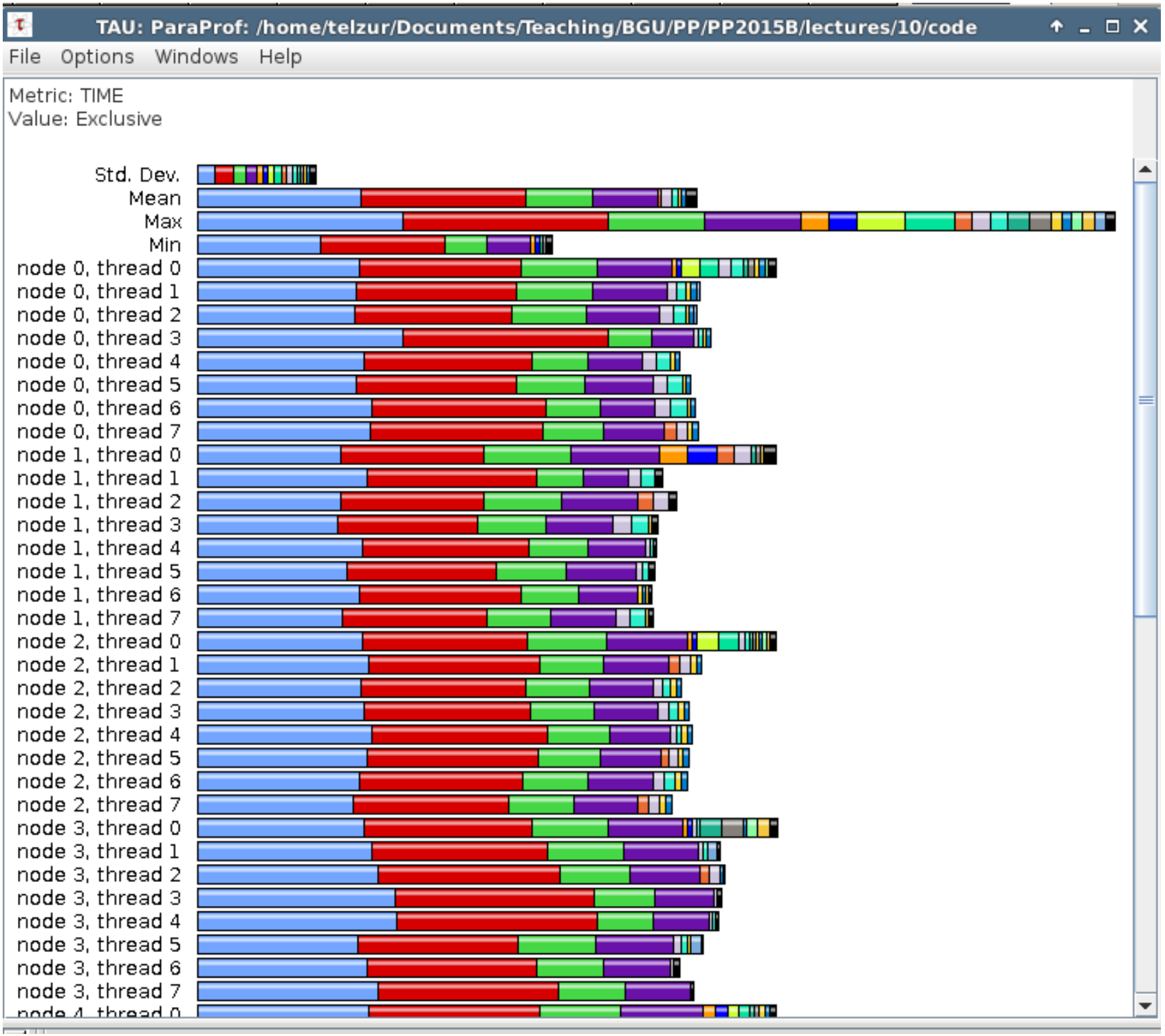
More demos: Scalasca and TAU

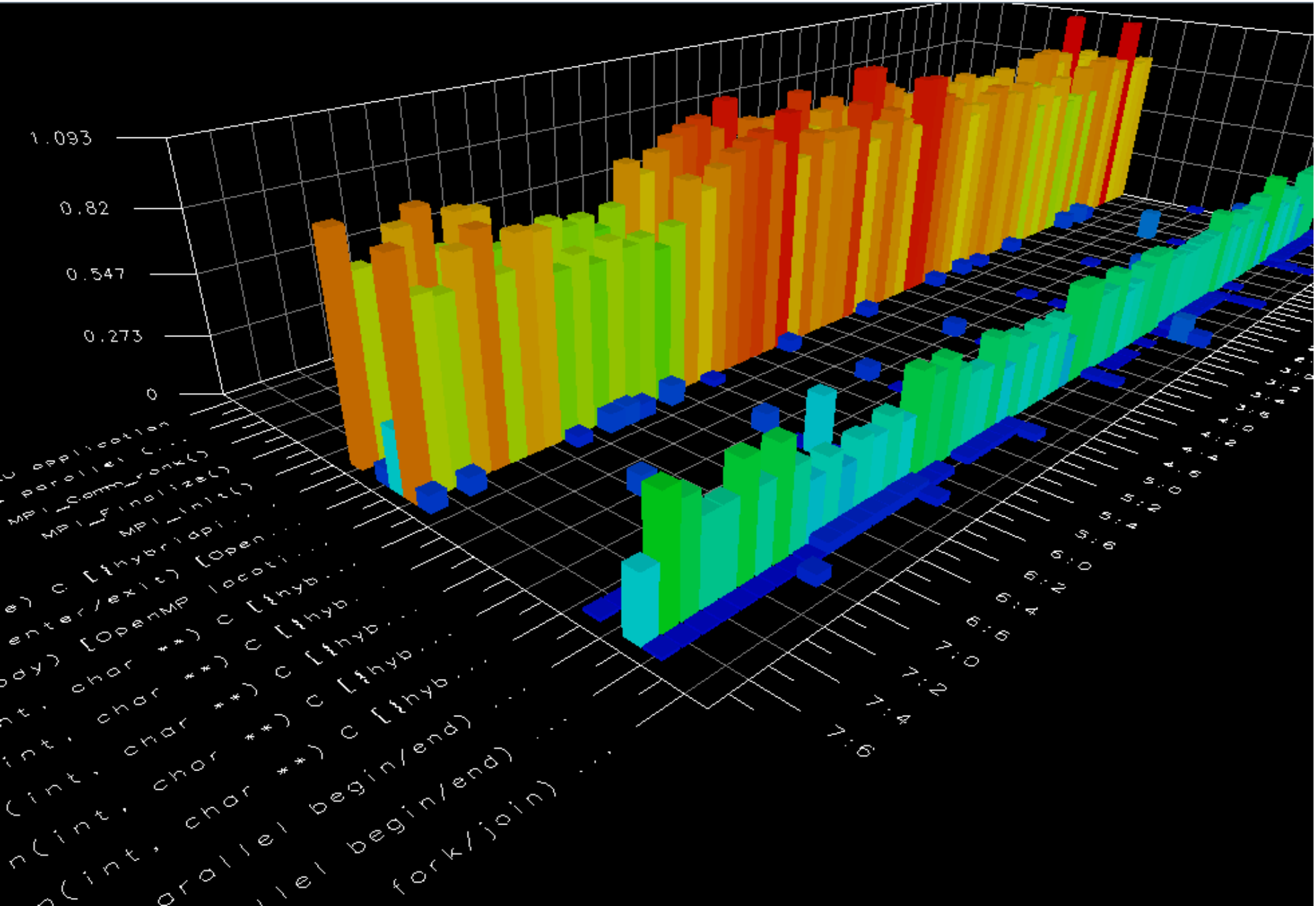
Demos on the Vi-HPS VM:
~/PP/10/...

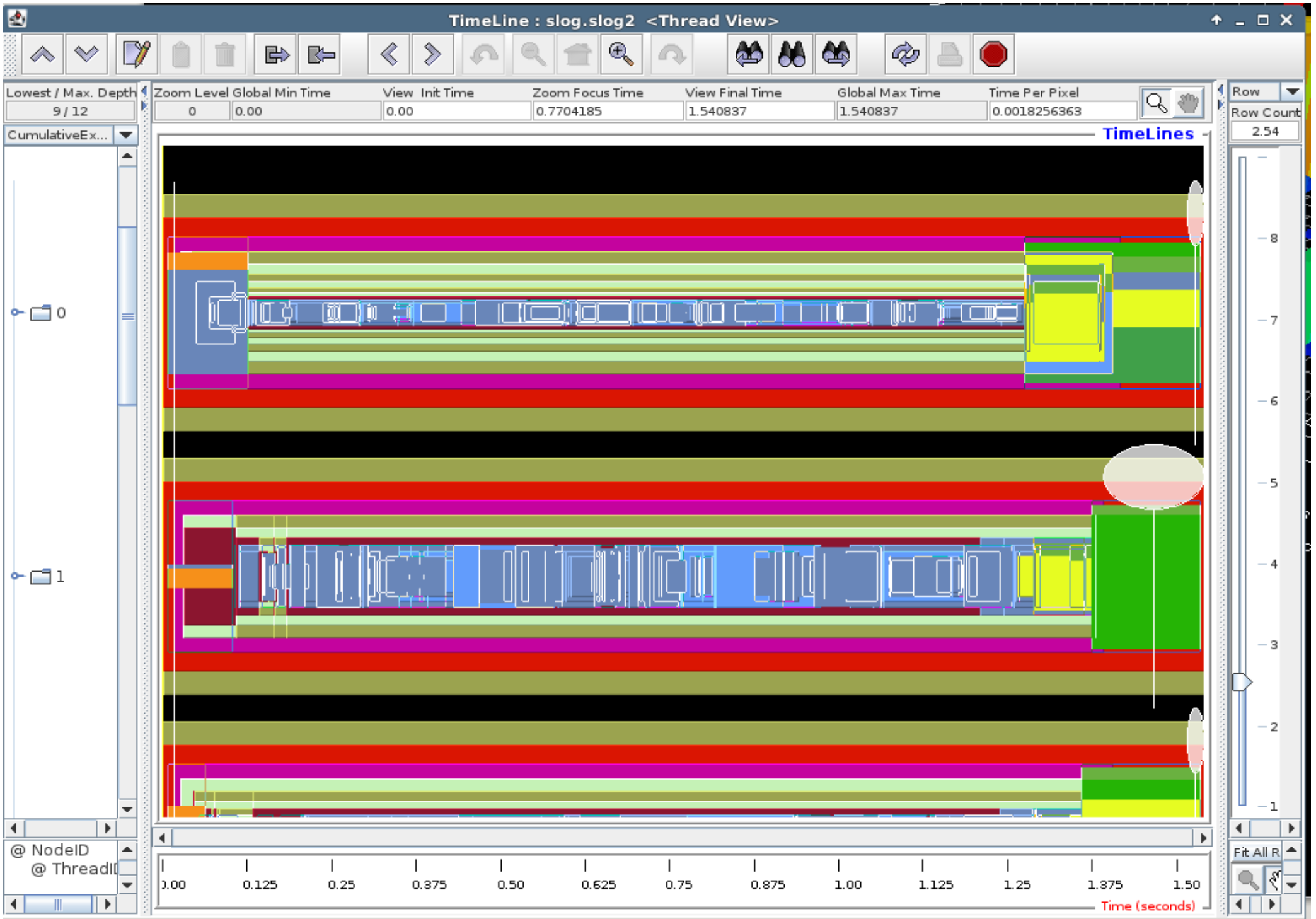
scalasca -examine ./scorep_hybridpi_scalasca_8x8_sum/

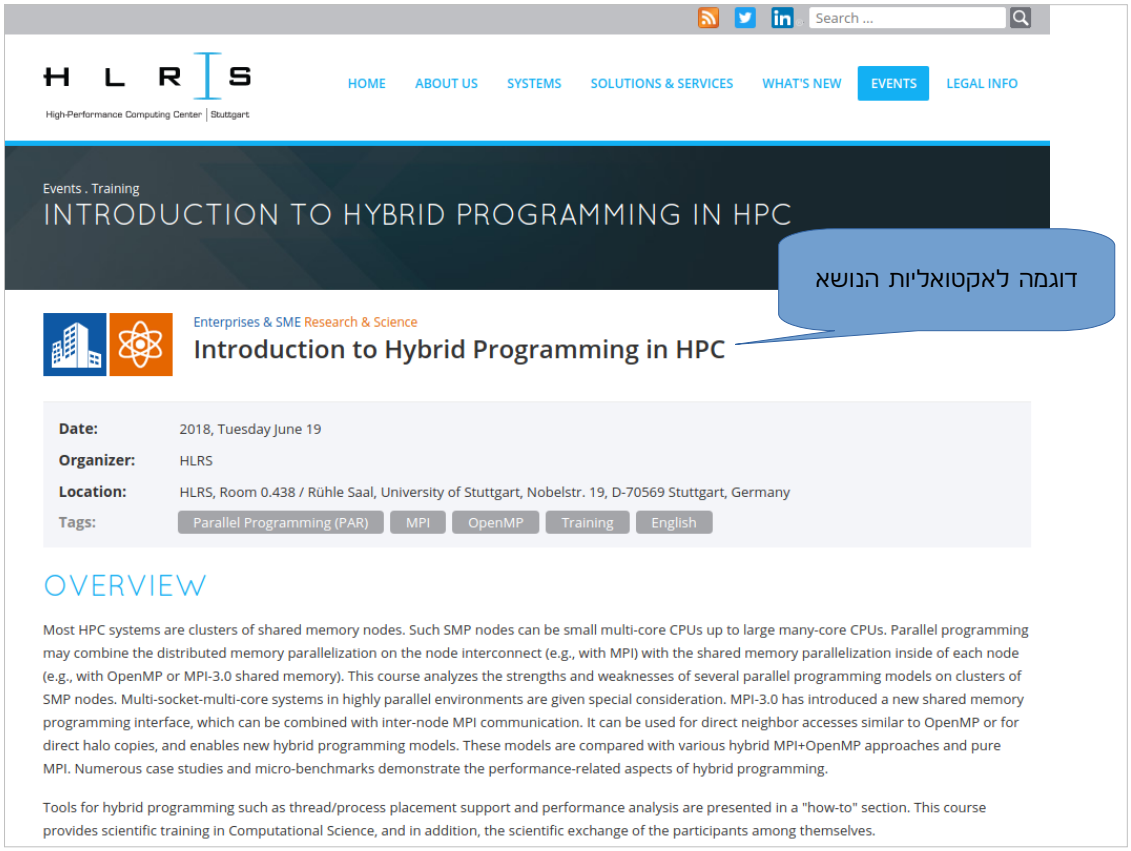


```
export OMP_NUM_TREADS=8

export \
TAU_MAKEFILE=/media/telzur/.../lib/Makefile.tau-
callpath-mpi-python-pdt-openmp-opari

export TAU_COMM_MATRIX=1

export TAU_TRACE=1

export TAU_PROFILE=1

tau_cc.sh  -g -o hybridpi_tau ./hybridpi.c

mpirun -np 8 ./hybrid_tau

tau_treemerge

tau2slog2  tau.trc tau.edf -o slog.slog2

jumpshot ./slog.slog2 &

paraprof &
```


TAU: ParaProf: /home/telzur/Documents/Teaching/BGU/PP/PP2015B/lectures/10/code
File Options Windows Help
Metric: TIME
Value: Exclusive
Std. Dev.
Mean
Max
Min
node 0, thread 0
node 0, thread 1
node 0, thread 2
node 0, thread 3
node 0, thread 4
node 0, thread 5
node 0, thread 6
node 0, thread 7
node 1, thread 0
node 1, thread 1
node 1, thread 2
node 1, thread 3
node 1, thread 4
node 1, thread 5
node 1, thread 6
node 1, thread 7
node 2, thread 0
node 2, thread 1
node 2, thread 2
node 2, thread 3
node 2, thread 4
node 2, thread 5
node 2, thread 6
node 2, thread 7
node 3, thread 0
node 3, thread 1
node 3, thread 2
node 3, thread 3
node 3, thread 4
node 3, thread 5
node 3, thread 6
node 3, thread 7


1.093
0.82
0.547
0.273
0


TimeLine : slog.slog2 <Thread View>
Lowest / Max. Depth
9 / 12
Zoom Level
0
Global Min Time
0.00
View Init Time
0.00
Zoom Focus Time
0.7704185
View Final Time
1.540837
Global Max Time
1.540837
Time Per Pixel
0.0018256363
Row
Row Count
2.54
CumulativeEx...
TimeLines
0
1
@ NodeID
@ ThreadID
0.00
0.125
0.25
0.375
0.50
0.625
0.75
0.875
1.00
1.125
1.25
1.375
1.50
Time (seconds)
Fit All R

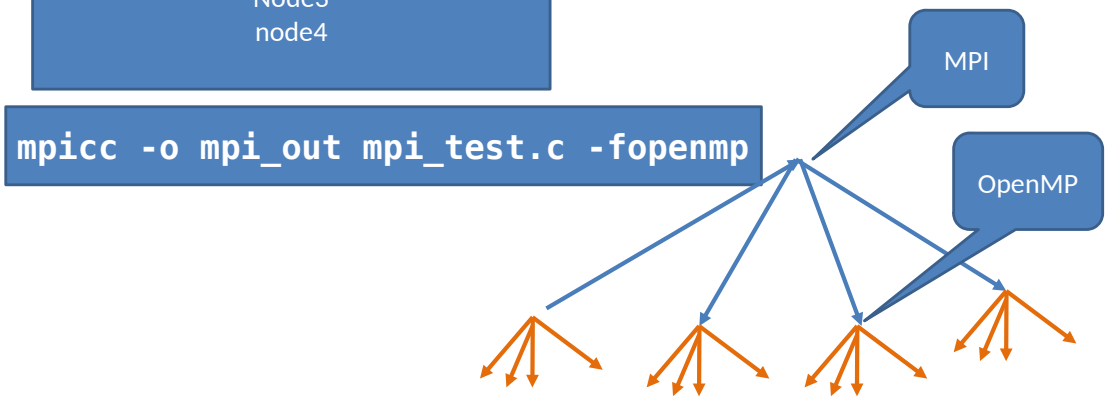

Search ...
HLRS
High-Performance Computing Center | Stuttgart
HOME
ABOUT US
SYSTEMS
SOLUTIONS & SERVICES
WHAT'S NEW
EVENTS
LEGAL INFO
Events . Training
INTRODUCTION TO HYBRID PROGRAMMING IN HPC
דוגמה לאקטואליות הנושא
Enterprises & SME Research & Science
Introduction to Hybrid Programming in HPC
Date: 2018, Tuesday June 19
Organizer: HLRS
Location: HLRS, Room 0.438 / Rühle Saal, University of Stuttgart, Nobelstr. 19, D-70569 Stuttgart, Germany
Tags: Parallel Programming (PAR) MPI OpenMP Training English
OVERVIEW
Most HPC systems are clusters of shared memory nodes. Such SMP nodes can be small multi-core CPUs up to large many-core CPUs. Parallel programming may combine the distributed memory parallelization on the node interconnect (e.g., with MPI) with the shared memory parallelization inside of each node (e.g., with OpenMP or MPI-3.0 shared memory). This course analyzes the strengths and weaknesses of several parallel programming models on clusters of SMP nodes. Multi-socket-multi-core systems in highly parallel environments are given special consideration. MPI-3.0 has introduced a new shared memory programming interface, which can be combined with inter-node MPI communication. It can be used for direct neighbor accesses similar to OpenMP or for direct halo copies, and enables new hybrid programming models. These models are compared with various hybrid MPI+OpenMP approaches and pure MPI. Numerous case studies and micro-benchmarks demonstrate the performance-related aspects of hybrid programming.
Tools for hybrid programming such as thread/process placement support and performance analysis are presented in a "how-to" section. This course provides scientific training in Computational Science, and in addition, the scientific exchange of the participants among themselves.

Hybrid MPI + OpenMP Demo

Machine File:
Node1
Node2
Node3
node4

Each node has 8 cores

```
mpicc -o mpi_out mpi_test.c -fopenmp
```



Demo: cd /home/telzur/Documents/Teaching/PP2XXXXX/lectures/10/code
program name: hybridpi.c

```
mpicc -o mpi_exe mpi_test.c -fopenmp
```

```
export OMP_NUM_THREADS=8 (bash)
setenv OMP_NUM_THREADS 8 (csh)
```

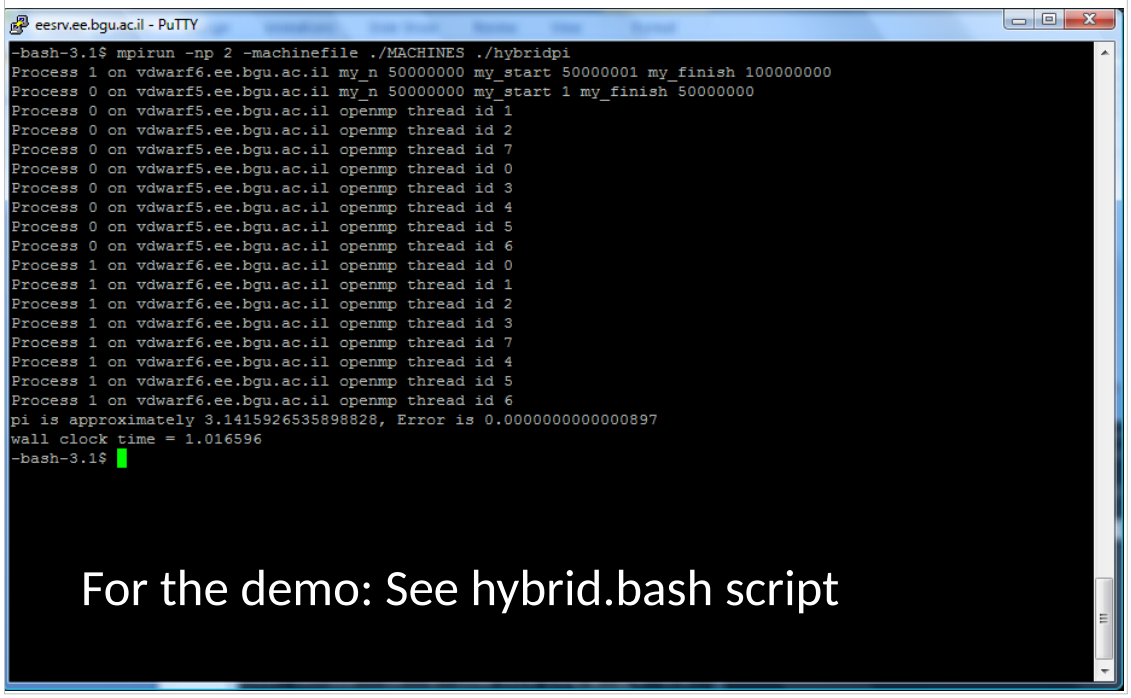
```
mpirun  -np  4  –machinefile ./machines mpi_exe
```

Hybrid Pi (MPI+OpenMP)

```
#include <stdio.h>
#include <mpi.h>
#include <omp.h>
#define NBIN 100000
#define MAX_THREADS 8

int main(int argc,char **argv) {
      int nbin,myid,nproc,nthreads,tid;
      double step,sum[MAX_THREADS]={0.0},pi=0.0,pig;
      MPI_Init(&argc,&argv);
      MPI_Comm_rank(MPI_COMM_WORLD,&myid);
      MPI_Comm_size(MPI_COMM_WORLD,&nproc);
      nbin = NBIN/nproc;
      step = 1.0/(nbin*nproc);
```

```
#pragma omp parallel private(tid)
    {
        int i;
        double x;
        nthreads = omp_get_num_threads();
        tid = omp_get_thread_num();
        for (i=nbin*myid+tid; i<nbin*(myid+1); i+=nthreads) {
            x = (i+0.5)*step;
            sum[tid] += 4.0/(1.0+x*x);
        }
        printf("rank tid sum = %d %d %e\n",myid,tid,sum[tid]);
    }
    for(tid=0; tid<nthreads; tid++)
        pi += sum[tid]*step;
    MPI_Allreduce(&pi,&pig,1,MPI_DOUBLE,MPI_SUM,MPI_COMM_WORLD);
    if (myid==0) printf("PI = %f\n",pig);
    MPI_Finalize();
    return 0;  }
```

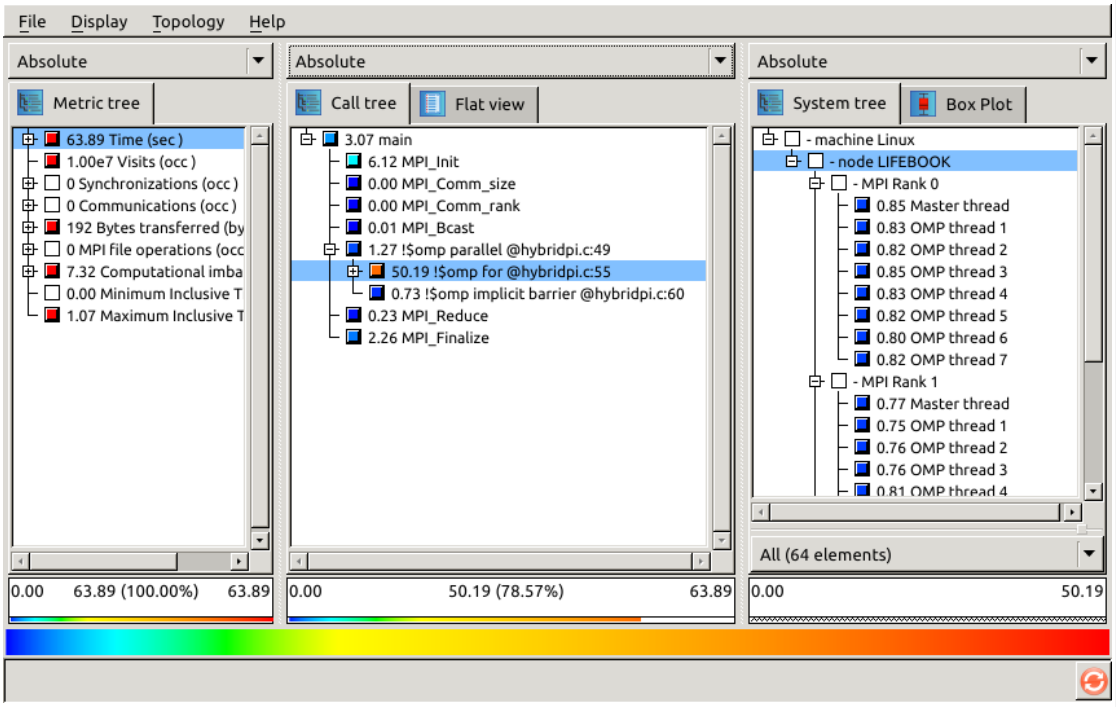
Hybrid MPI+OpenMP continued



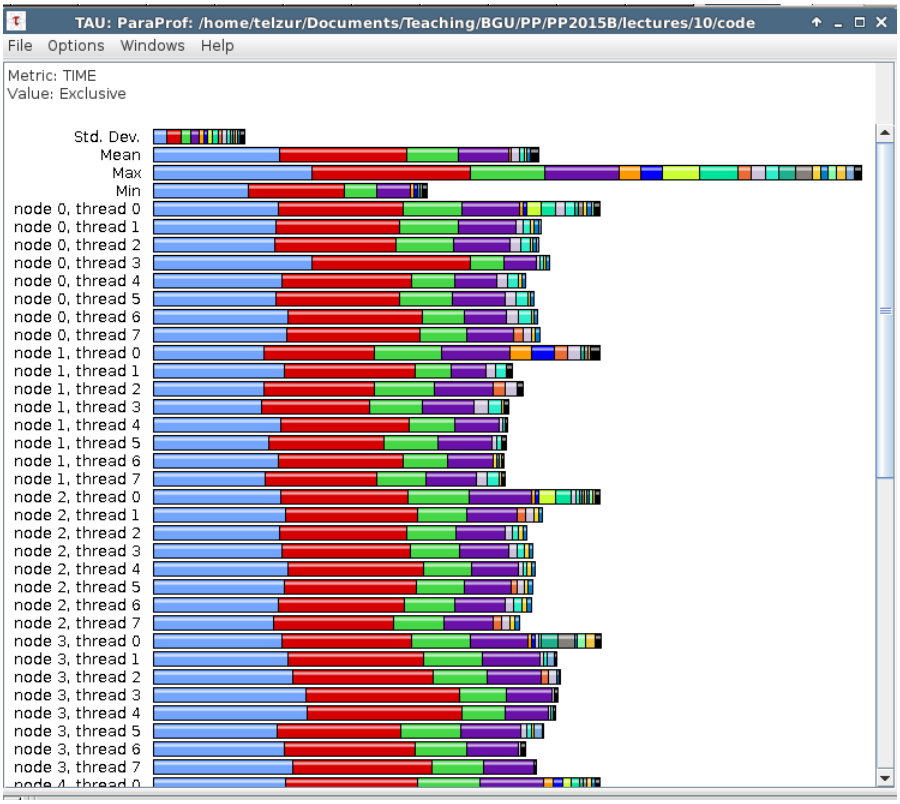
More demos: Scalasca and TAU

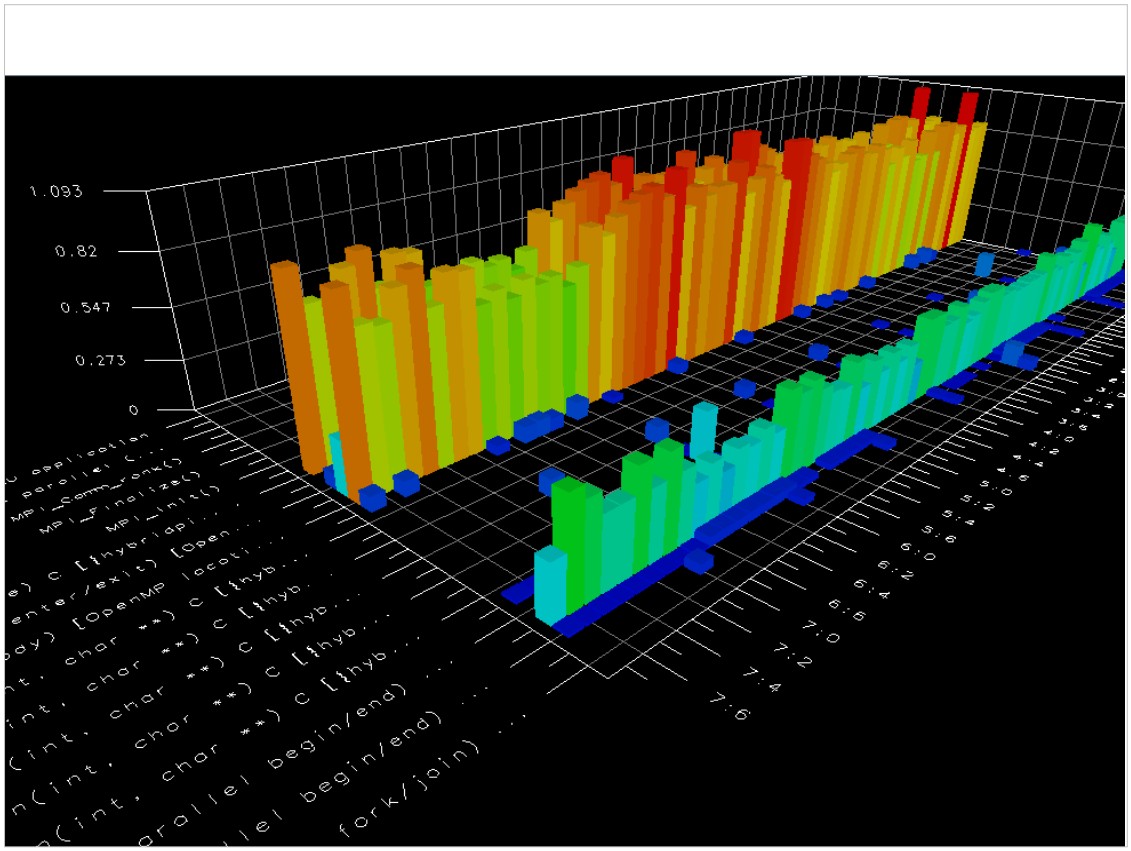
Demos on the Vi-HPS VM:
~/PP/10/...

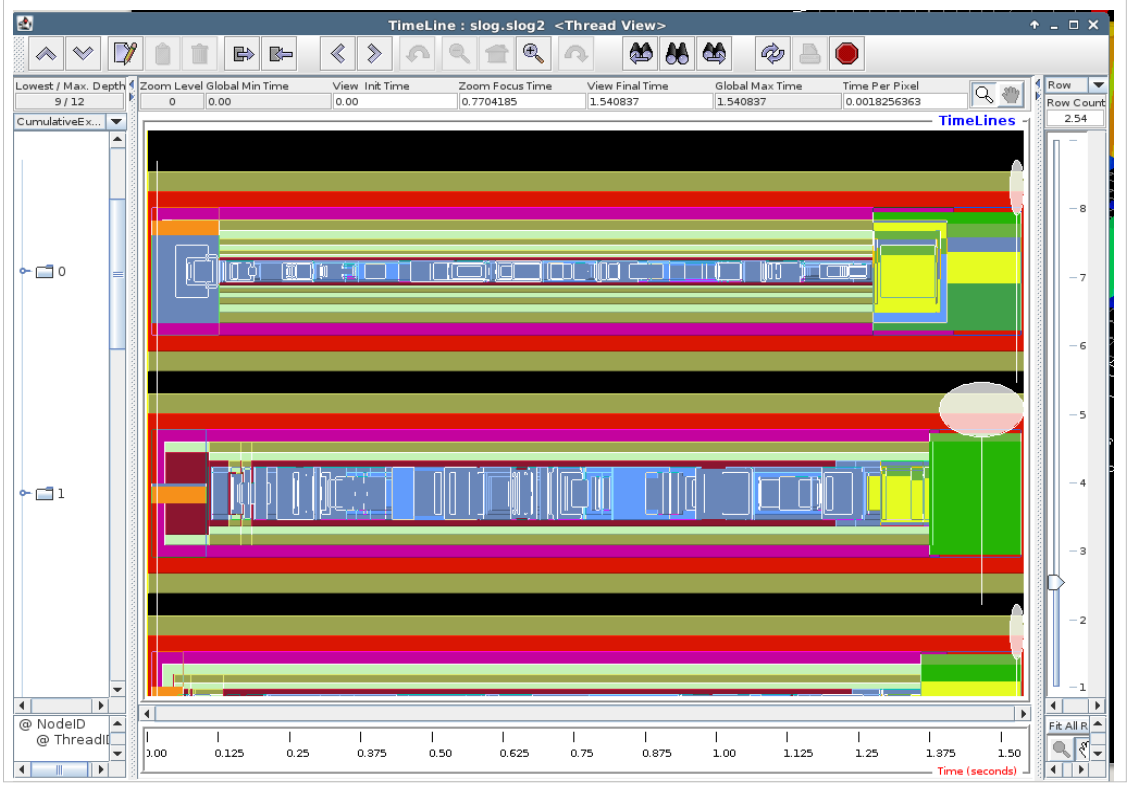
scalasca -examine ./scorep_hybridpi_scalasca_8x8_sum/



```
export OMP_NUM_TREADS=8

export \
TAU_MAKEFILE=/media/telzur/.../lib/Makefile.tau-
calTpath-mpi-python-pdt-openmp-opari

export TAU_COMM_MATRIX=1

export TAU_TRACE=1

export TAU_PROFILE=1

tau_cc.sh  -g -o hybridpi_tau ./hybridpi.c

mpirun -np 8 ./hybrid_tau

tau_treemerge

tau2slog2  tau.trc tau.edf -o slog.slog2

jumpshot ./slog.slog2 &

paraprof &
```


TAU: ParaProf: /home/telzur/Documents/Teaching/BGU/PP/PP2015B/lectures/10/code
File Options Windows Help
Metric: TIME
Value: Exclusive
Std. Dev.
Mean
Max
Min
node 0, thread 0
node 0, thread 1
node 0, thread 2
node 0, thread 3
node 0, thread 4
node 0, thread 5
node 0, thread 6
node 0, thread 7
node 1, thread 0
node 1, thread 1
node 1, thread 2
node 1, thread 3
node 1, thread 4
node 1, thread 5
node 1, thread 6
node 1, thread 7
node 2, thread 0
node 2, thread 1
node 2, thread 2
node 2, thread 3
node 2, thread 4
node 2, thread 5
node 2, thread 6
node 2, thread 7
node 3, thread 0
node 3, thread 1
node 3, thread 2
node 3, thread 3
node 3, thread 4
node 3, thread 5
node 3, thread 6
node 3, thread 7


1.093
0.82
0.547
0.273
0
fork/join)


TimeLine : slog.slog2 <Thread View>
Lowest / Max. Depth
9 / 12
Zoom Level
Global Min Time
View Init Time
Zoom Focus Time
View Final Time
Global Max Time
Time Per Pixel
0
0.00
0.00
0.7704185
1.540837
1.540837
0.0018256363
Row
Row Count
2.54
CumulativeEx...
TimeLines
0
1
@ NodeID
@ ThreadID
0.00
0.125
0.25
0.375
0.50
0.625
0.75
0.875
1.00
1.125
1.25
1.375
1.50
Time (seconds)
Fit All R