Exercise 1: Detecting the focus of analysis (FOA)

a) Set the environment variables in `trace.sh`:

```
#!/bin/bash
source <path_to_extrae>/etc/extrae.sh
export EXTRAE_CONFIG_FILE=extrae.xml
export NX_ARGS="${NX_ARGS} --instrumentation=extrae"
export LD_PRELOAD=${EXTRAE_HOME}/lib/libompitrace.so
$*
```

b) Execute `OMP_NUM_THREADS=1 mpirun -np 4 ./trace.sh ./bt-mz_W.4` to generate a trace

c) Execute `wxparaver bt-mz_W.4.prv` to open the results with *Paraver*. Choose *Hints* → *MPI* → *MPI profile* and click on *MPI call*.

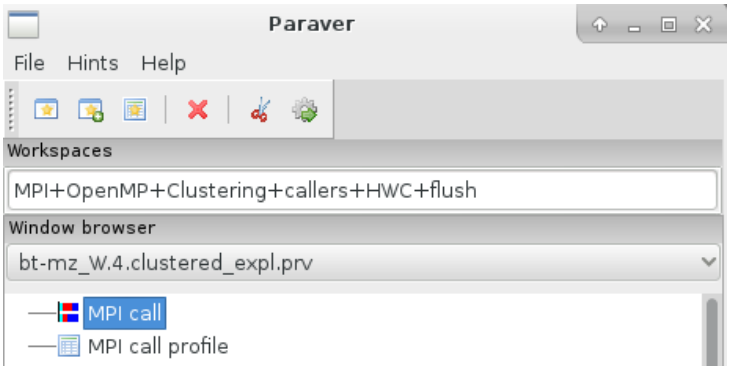

Figure 1: Paraver GUI

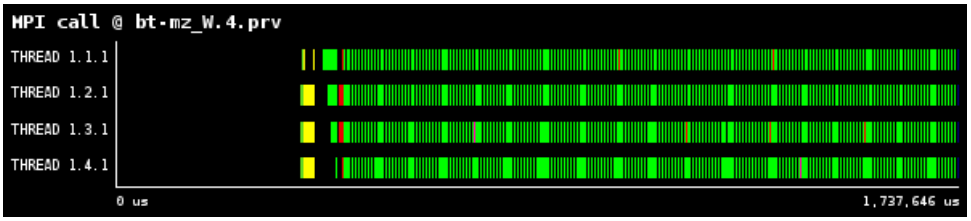

Figure 2: Full time line of the application execution (Paraver)

Cutting off of the initialization phase and the data distribution by zooming gives the time line of the Focus of Analysis (FOA).

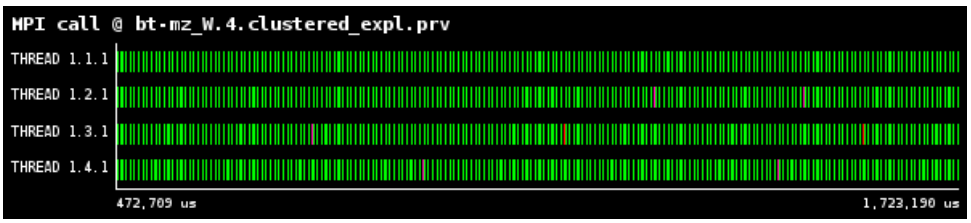

Figure 3: FOA of the application (Paraver)

Exercise 2: Load Balance

a) Choose the MPI call time line and zoom in the FOA. Press the right mouse button and choose *Copy*. Then press the right mouse button on the MPI call profile and choose *Paste* → *Time*.

The Load Balance of computations time outside of MPI among processes:
LD = Avg/Max = 97%

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Waitall
THREAD 1.1.1	98.83 %	0.32 %	0.22 %	0.63 %
THREAD 1.2.1	96.60 %	0.25 %	0.30 %	2.86 %
THREAD 1.3.1	94.74 %	0.33 %	0.21 %	4.72 %
THREAD 1.4.1	92.26 %	0.26 %	0.25 %	7.23 %
Total	382.43 %	1.16 %	0.98 %	15.44 %
Average	95.61 %	0.29 %	0.24 %	3.86 %
Maximum	98.83 %	0.33 %	0.30 %	7.23 %
Minimum	92.26 %	0.25 %	0.21 %	0.63 %
StDev	2.41 %	0.04 %	0.03 %	2.42 %
Avg/Max	0.97	0.87	0.82	0.53

Figure 4: MPI call profile with percentage of execution time in FOA (Paraver)

b) Choose *Hints* → *HWC* → *Histogram of useful instructions* and *Histogram of duration correlated with IPC*. Here are the histograms for useful duration and instructions in the FOA. They indicate how well the computation time and number of executed instructions are balanced among the processes. The duration histogram shows vertical lines, it indicates the computation time is well balanced among the processes. However the number of executed instructions by the processes is not so well balanced among the processes and can be investigated deeper by Clustering of the application.

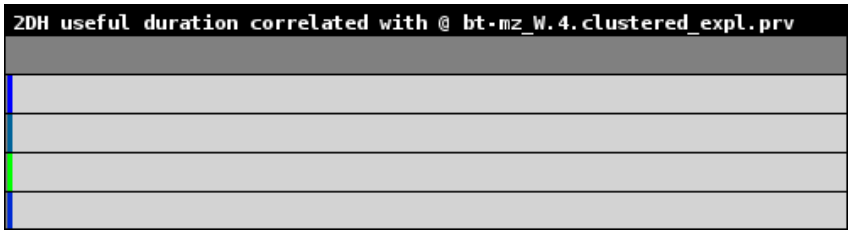

Figure 5: Histogram of useful duration in FOA (Paraver)

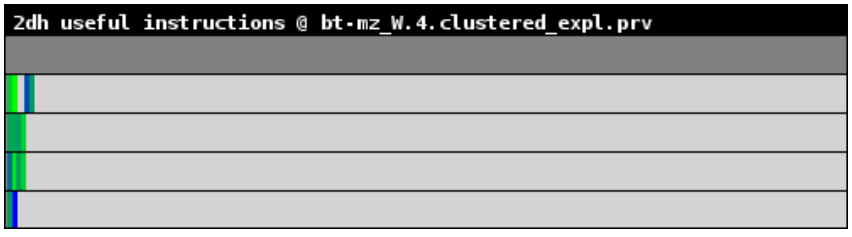

Figure 6: Histogram of useful instructions in FOA (Paraver)

c) To show the number of instructions per cycle change for MPI profile the *Statistic* parameter from *Time%* to *Average Value* and *Window* parameter from *MPI call* to *Instructions per cycle* (see Figure 7).

The average number of instructions per cycle is 2.02-2.08 by different processes. The load balance of IPC among processes is 98%.

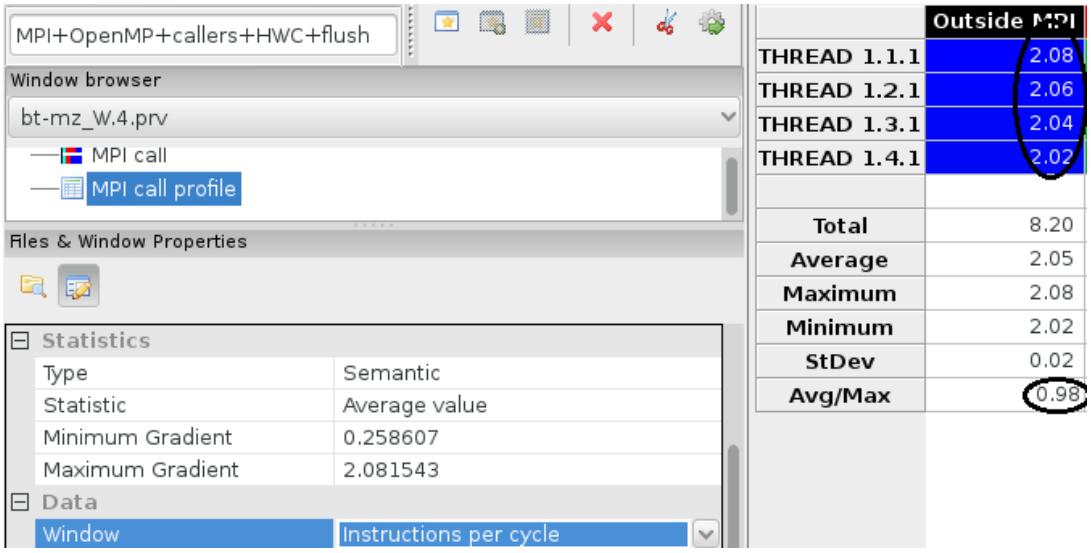

Figure 7: Average number of instructions per cycle executed by processes (Paraver)

Exercise 3: Efficiency

a) To simulate the program execution on an ideal network click right mouse button on the MPI call time line → *Run* → *Dimemas*. Choose the Dimemas Cfg file `IdealNetwork.4.1ppn.cfg`. Click *Run*.

b) You can get the Serialization Efficiency from the new MPI call profile (see Figure 8). The total execution wall clock time for calculation of Transfer Efficiency you can see by click the right mouse button on the MPI call time line and choosing *Info Panel* (see Figure 9).

SE = Maximum comp time share on ideal network = 99.97%.
TE = Total time on ideal network / Real total time = 1223971/1267654 = 96.6%.
ComE = SE * TE = 96.5%.
PE = LB * ComE = 93.6%.

	Outside MPI	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.1.1	99.97 %	0.03 %	-	-
THREAD 1.2.1	96.49 %	3.47 %	0.00 %	0.04 %
THREAD 1.3.1	94.16 %	5.79 %	0.00 %	0.05 %
THREAD 1.4.1	91.62 %	8.31 %	0.00 %	0.07 %
Total	382.23 %	17.60 %	0.01 %	0.16 %
Average	95.56 %	4.40 %	0.00 %	0.05 %
Maximum	SE 99.97 %	8.31 %	0.00 %	0.07 %
Minimum	91.62 %	0.03 %	0.00 %	0.04 %
StDev	3.08 %	3.05 %	0.00 %	0.01 %
Avg/Max	0.96	0.53	1.00	0.79

Figure 8: MPI call profile of the execution simulated on ideal network (Paraver)

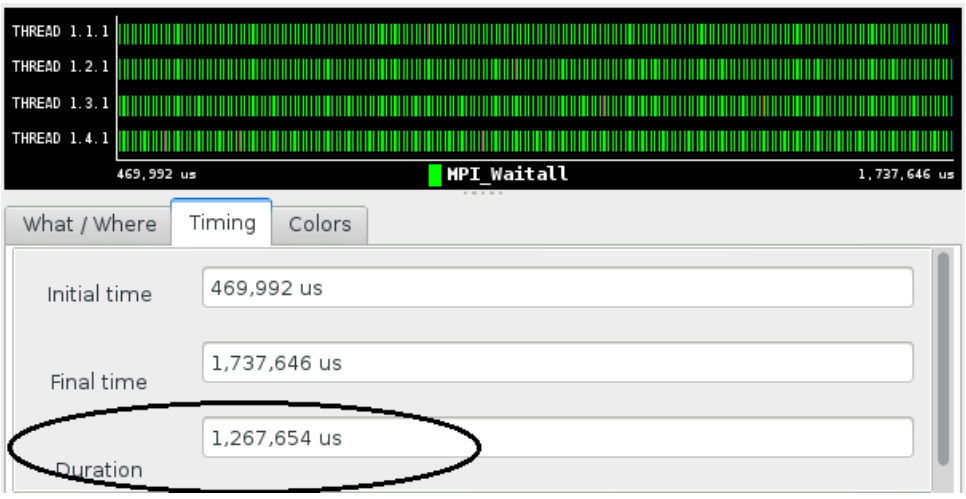

Figure 9: MPI call time line with Info Panel of real application execution (Paraver)

Exercise 4: Computing Performance

a) Execute the following call to generate a clustered trace of the application:

```
BurstClustering -p -d "cluster_explained.xml" -i "bt-mz_W.4.prv"
-o "bt-mz_W.4.clustered_expl.prv"
```

The created diagram (see Figure 10) shows a relationship of the number of instructions per cycle to the number of executed instructions for different clusters in the application. There are four cluster. The highest IPC is observed for Cluster 4, but the most number of instructions were executed for Cluster 2.

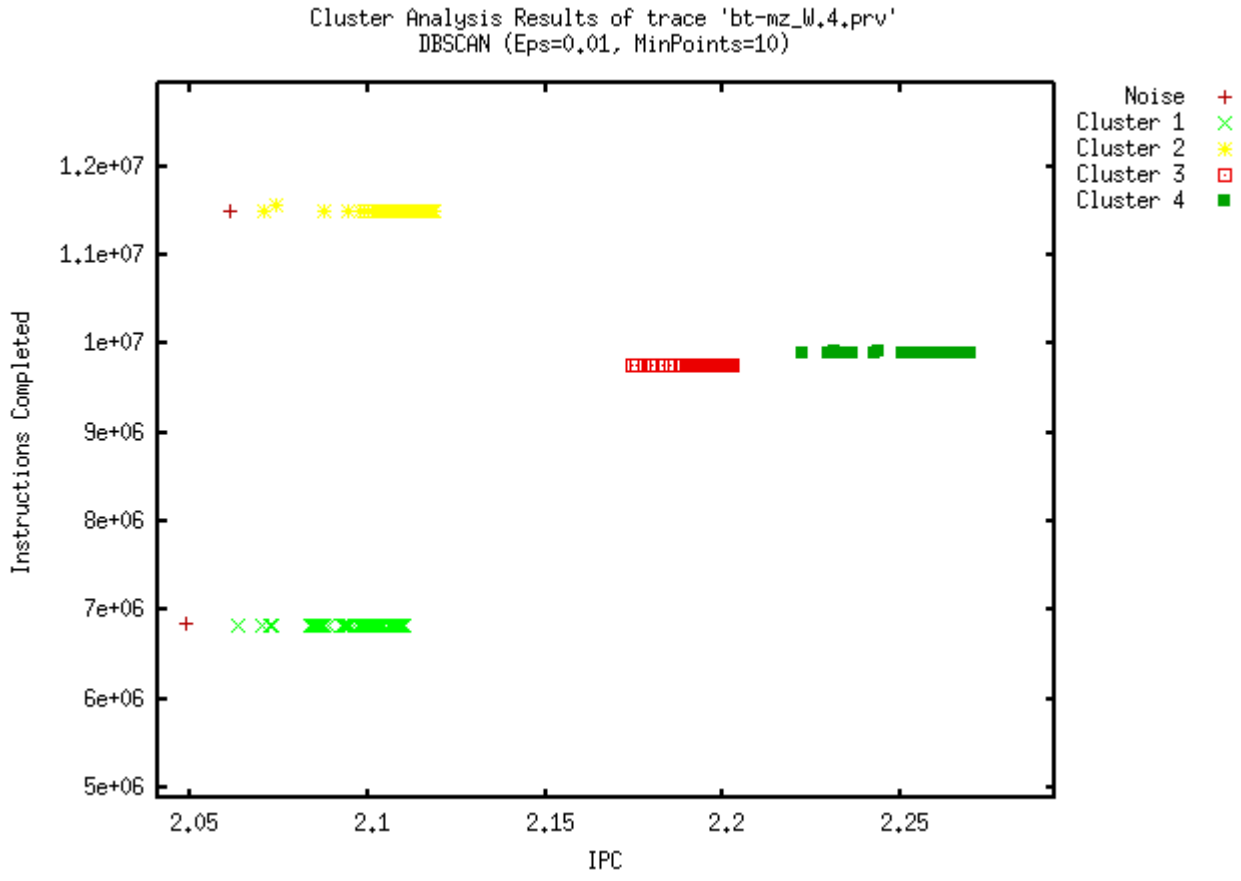

Figure 10: Relationship between IPC and number of executed instructions for different clusters

b) Choose *Hints* → *Clustering* → *Profile of clusters*. There is no cluster for process 1.4.1 recognized and processes 1.2.1 and 1.3.1 work only on Cluster 1. Cluster 1 needs also the most CPU time.

	End	Duration Filtered	Cluster 1	Cluster 2	Cluster 3	Cluster 4
THREAD 1.1.1	2.49 %	23.46 %	-	28.42 %	23.18 %	22.45 %
THREAD 1.2.1	5.42 %	77.62 %	16.96 %	-	-	-
THREAD 1.3.1	8.13 %	74.94 %	16.93 %	-	-	-
THREAD 1.4.1	10.41 %	89.59 %	-	-	-	-

Figure 11: Cluster ID profile with run time percentage for each cluster (Paraver)

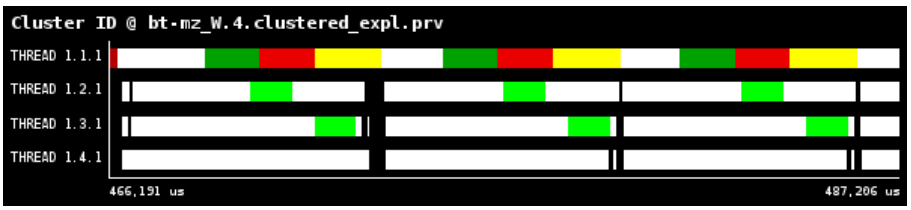

Figure 12: A cut from the zoomed Cluster ID time line (Paraver)

c) Also in Cluster 1 the smallest number of instructions with some lower IPC than in other Clusters is executed. Some reason for this can be a high cache miss ratio for this cluster.

To show the L3 cache miss ratio for each cluster first load the configuration file `L3_Total_miss_ratio.cfg` (see Figure 13). Change for Cluster ID profile the *Statistic* parameter from *Time%* to *Average Value* and *Window* parameter from *Cluster ID* to *L3 Total cache misses per 1000 instr.*

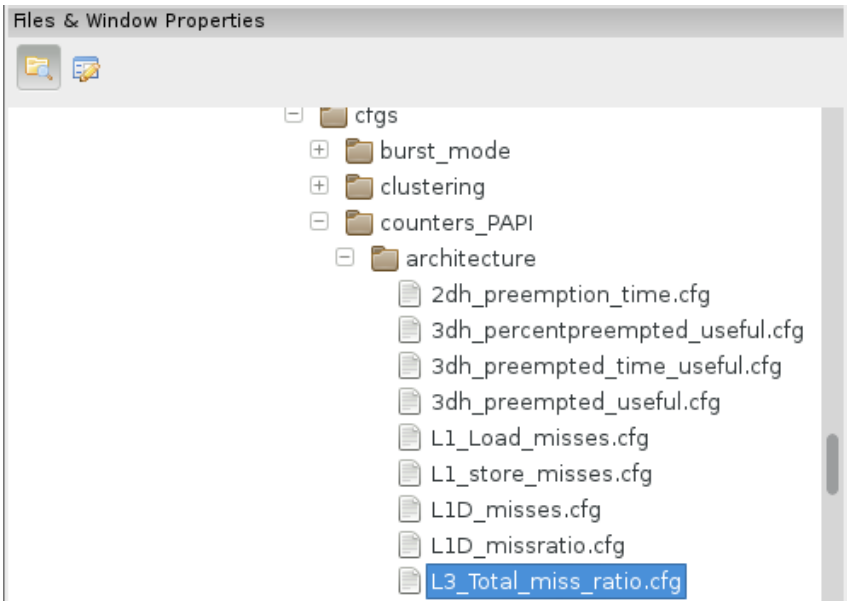

Figure 13: Configuration file loading window (Paraver)

How you can see in Figure 14, the highest cache miss ratio is observed for the Cluster 1 and the lowest cache miss ratio is by the cluster with the highest IPC.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
THREAD 1.1.1	-	1,544.37	1,720.20	98.31
THREAD 1.2.1	2,604.63	-	-	-
THREAD 1.3.1	2,605.49	-	-	-
THREAD 1.4.1	-	-	-	-

Figure 14: The average number of L3 total cache misses per 1000 instructions (Paraver)

Exercise 5: Communication

a) The most execution time in MPI operations is spent in MPI_Waitall (see Figure 4). Detailed communication behavior in one iteration you can see in Figure 15. The green parts are the MPI_Waitall calls. Only few time is spent for the data transfer and the most time in the communication the processes are waiting for each other. The processes seem to spend different time for their computations and the more fast processes wait for the data from the slower processes to continue their computation. Some reason for this can be the small load imbalance among processes. The first process spends only little time in MPI_Waitall, while the last one is over 7% of the full execution time in waiting.

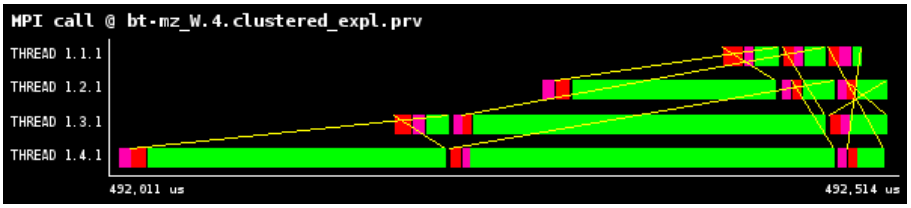

Figure 15: Cut of the zoomed MPI call time line (Paraver)

b) Choose *Hints* → *MPI* → *Point 2 point connectivity pattern*. Each process communicate with every process 201 times.

	THREAD 1.1.1	THREAD 1.2.1	THREAD 1.3.1	THREAD 1.4.1
THREAD 1.1.1	-	201	201	201
THREAD 1.2.1	201	-	201	201
THREAD 1.3.1	201	201	-	201
THREAD 1.4.1	201	201	201	-

Figure 16: Communication pattern of the application with number of communications between processes (Paraver)