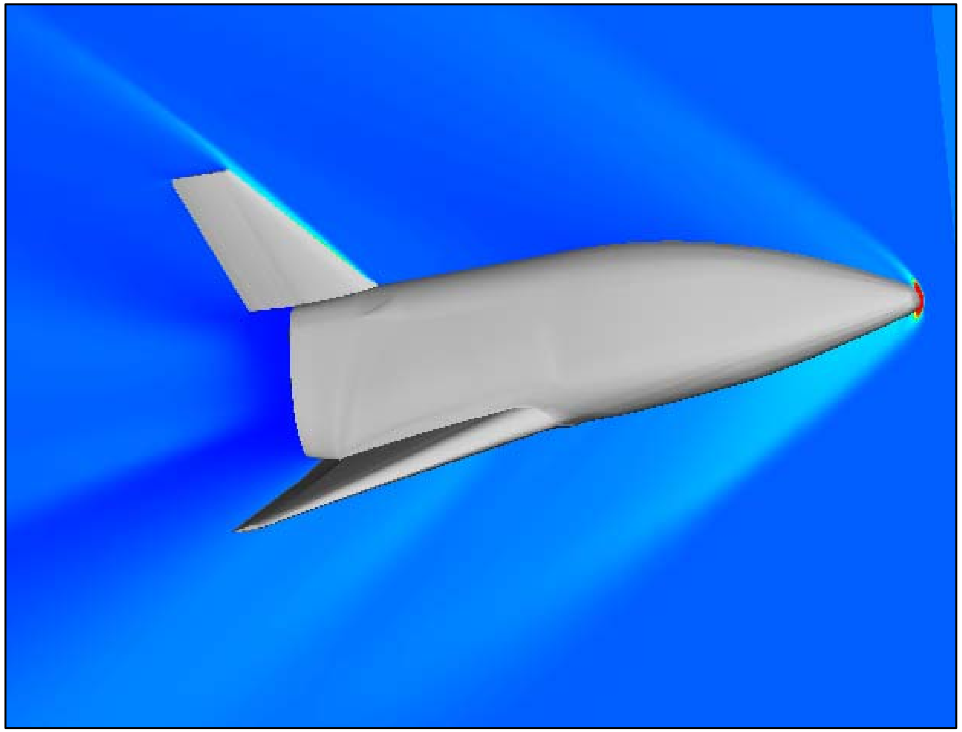
Case Study: FLOWer – Combining MPI and Autoparallelization

Dieter an Mey

Center for Computing and Communication
Aachen University

anmey@rz.rwth-aachen.de

Case Study: FLOWer

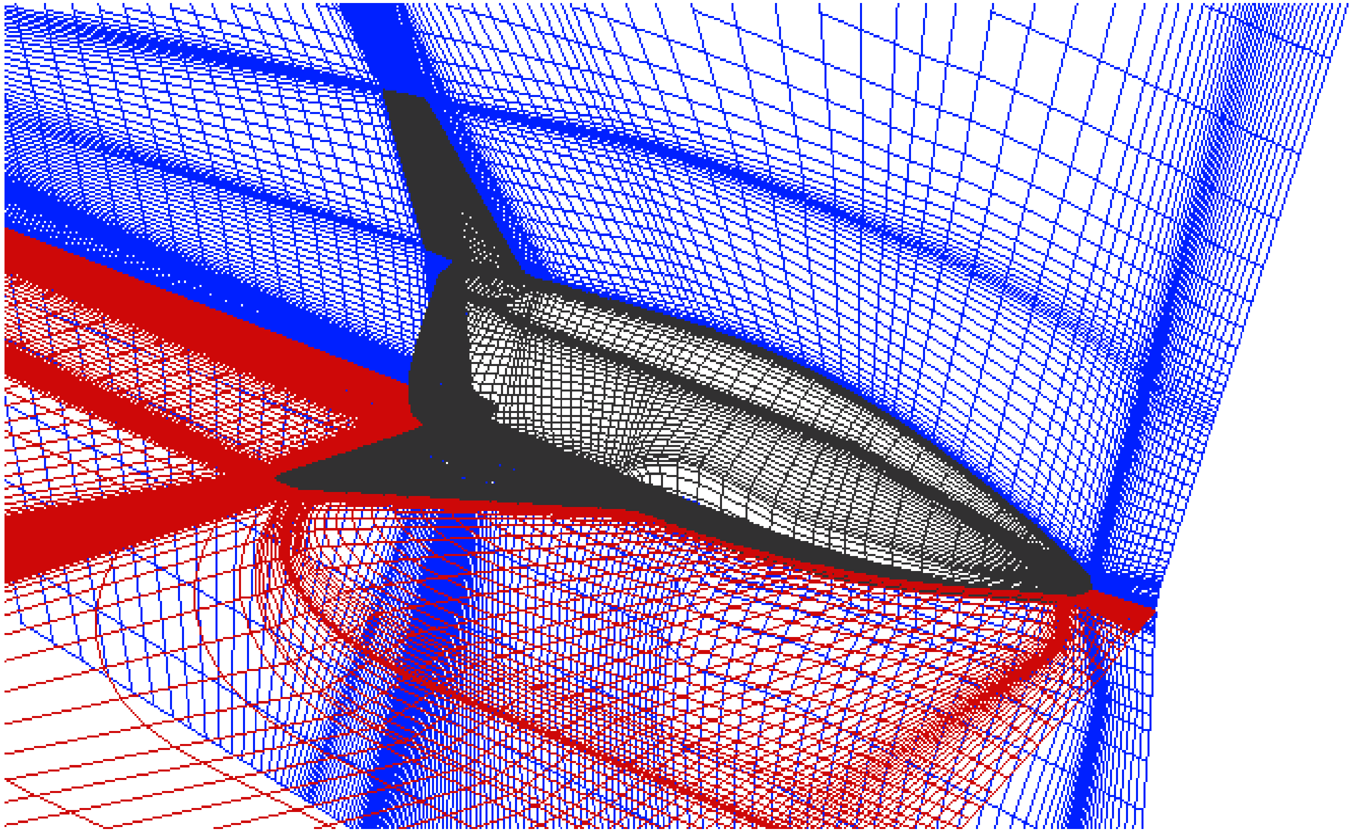


Case Study: FLOWer

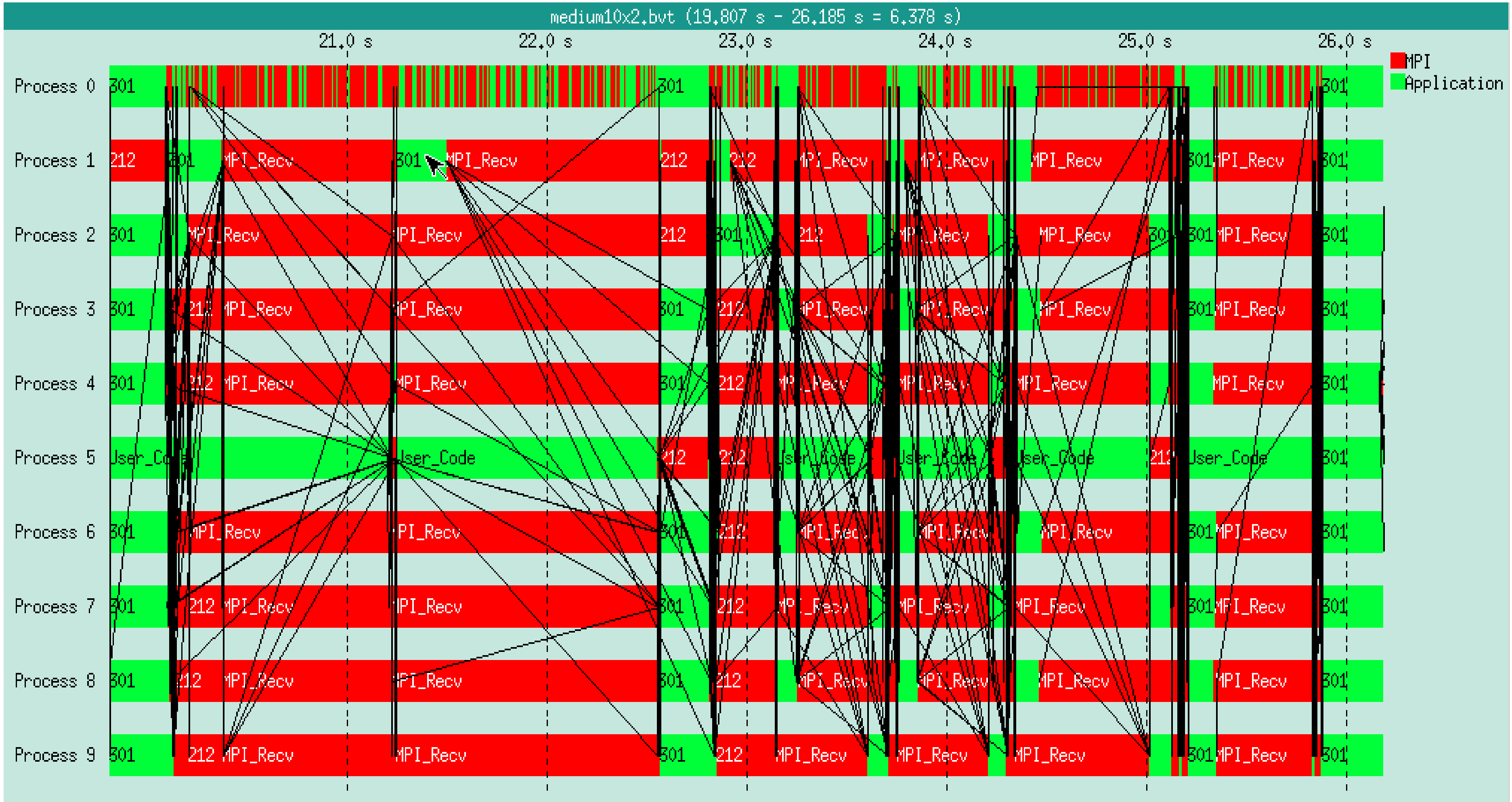
- Sponsered by the German Research Council (DFG), Birgit Reinartz and Michael Hesse in the Lehr- und Forschungsgebiet für Mechanik, RWTH Aachen University simulated PHOENIX, a small scale prototype of the **Space Hopper**, of a **space launch vehicle** designed to take off horizontally and glide back to earth after placing its cargo in orbit.
- Solution of the Navier-Stokes-Equations with the **FLOWer** flow solver, developped at the German Airospace Center (DLR)
- Parallelized with the **CLIC-3D** communication library which handles all **MPI** communication.
- Block-oriented information exchange => the number of blocks is limits the number of MPI tasks.
- Underneath to the coarse-grained parallelization with CLIC-3D/MPI, on a lower level, most of the compute intense loops can be **parallelized automatically** by the Fortran compiler => **Hybrid Parallelization**

Small Testcase

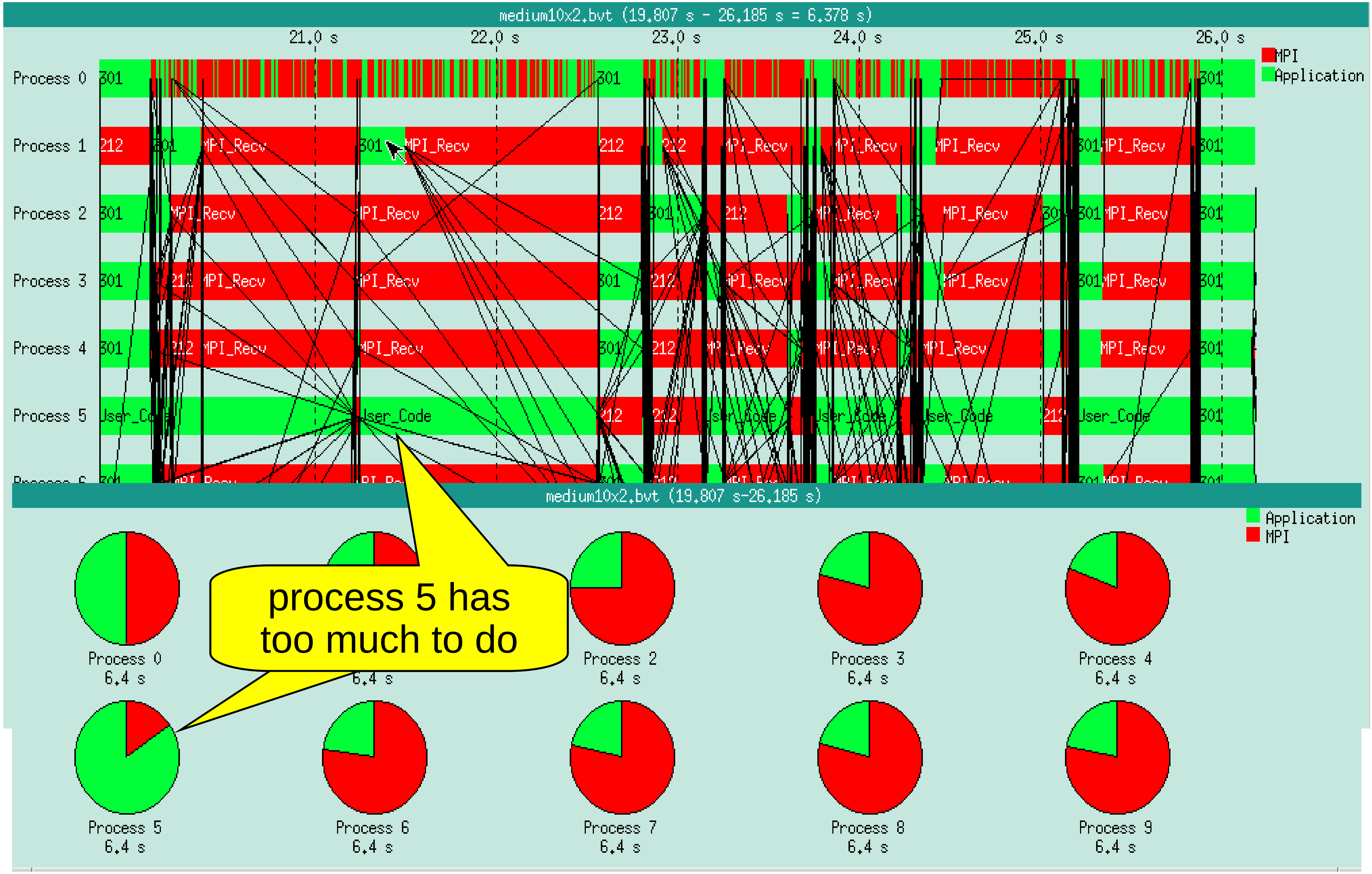
- Euler - Solver
- 1 Mio grid point
- inviscid flow
- 20 blocks



First Performance Experiments: 10x2 Processors



First Performance Experiments: 10x2 Processors



process 5 has too much to do

Hybrid Parallelization on a Shared Memory System

run10x1+5.ksh:

```
    #!/bin/ksh
    export OMP_NUM_THREADS=1;
    if [[ ${MP_RANK} = 0 ]]
    then
         export OMP_NUM_THREADS=
    fi;
    if [[ ${MP_RANK} = 5 ]]
    then
         export OMP_NUM_THREADS=5;
    fi;
    flower.exe
```

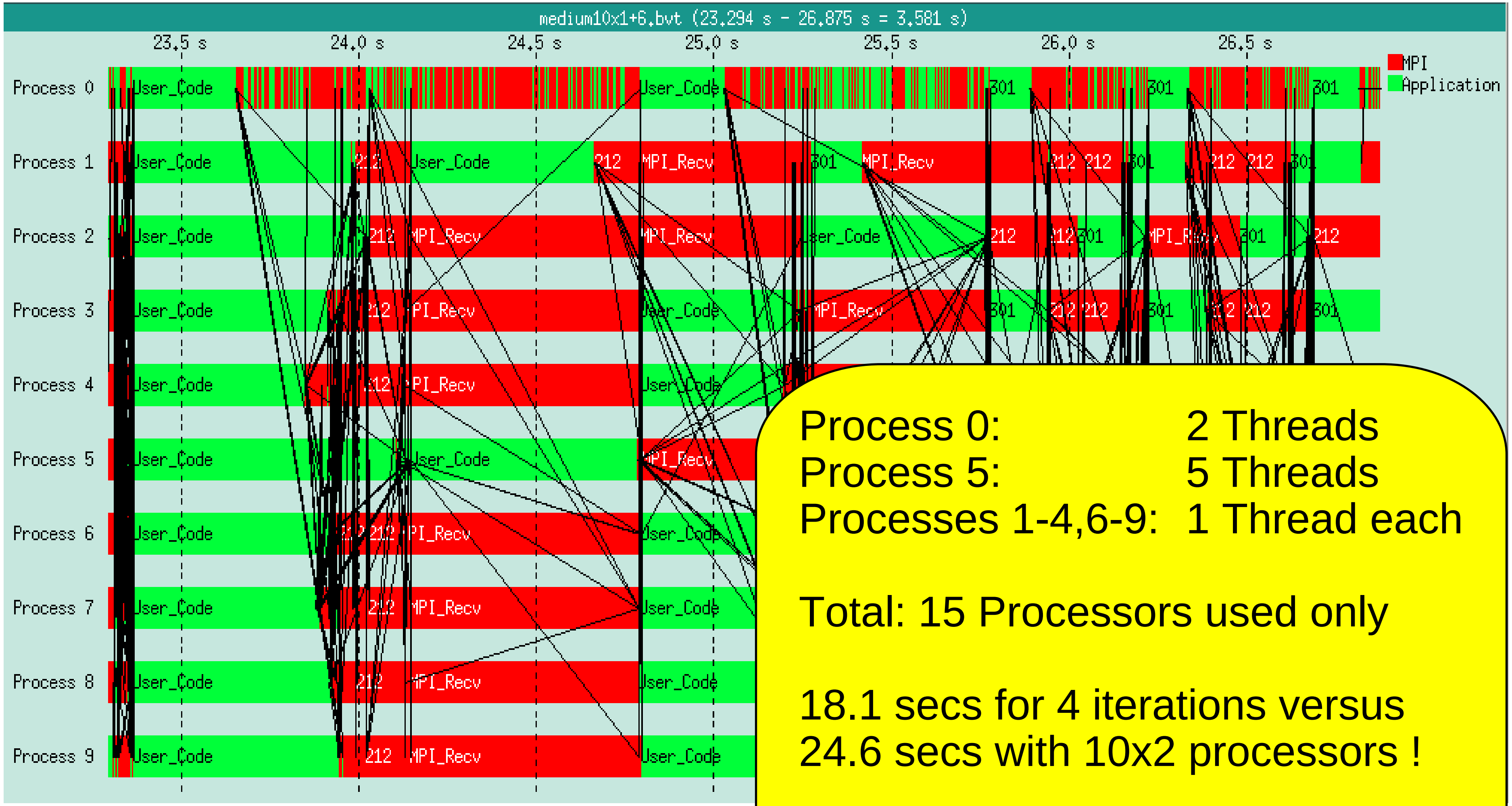
```
mprun -np 10 run10x1+5.ksh
```

Process 0:	2 Threads
Process 5:	5 Threads
Processes 1-4,6-9:	1 Thread each

Total: 15 Processors used only

Depends on MPI implementation

First Performance Experiments: 10x1+5 Processors



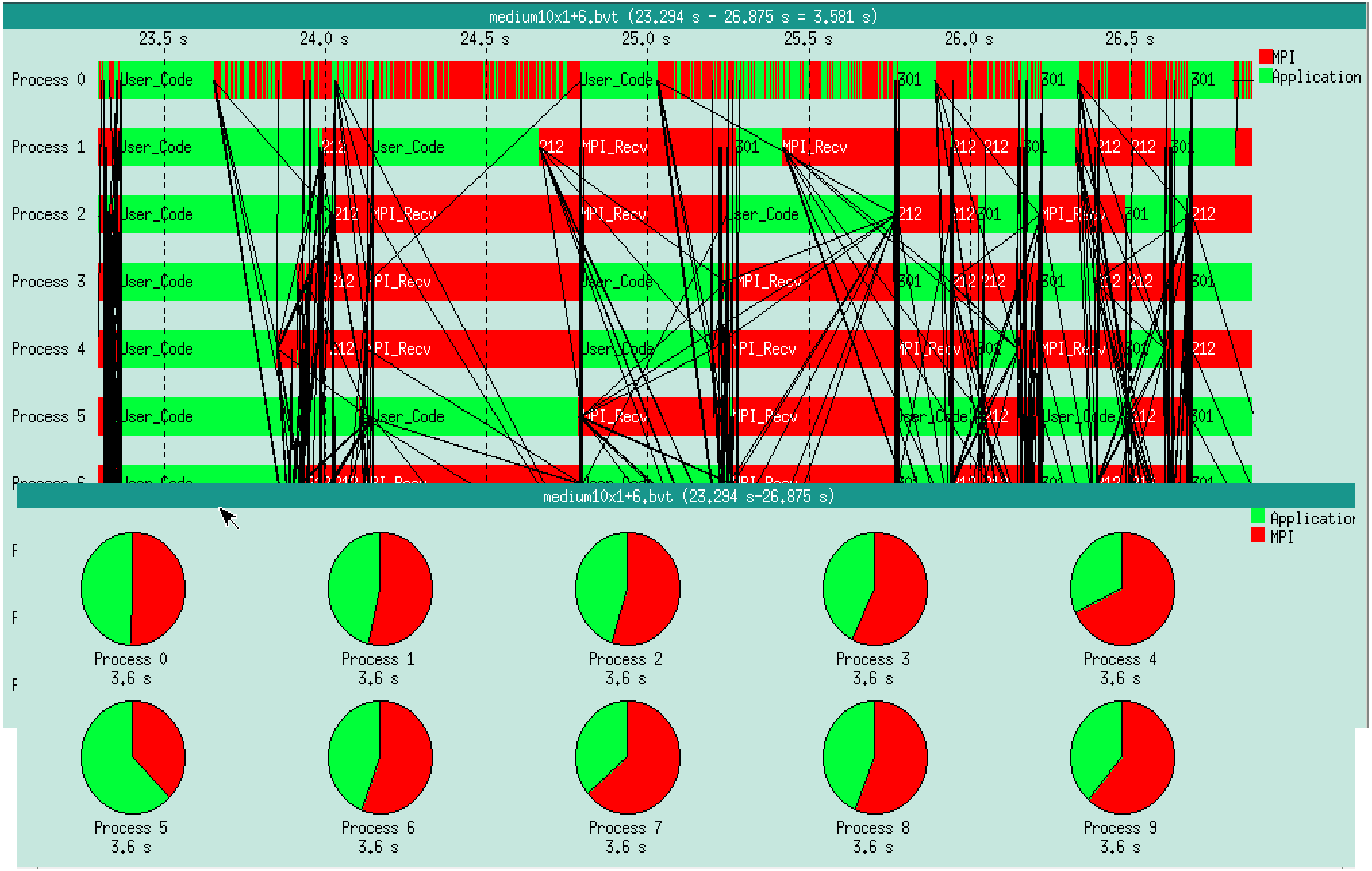
Process 0: 2 Threads
Process 5: 5 Threads
Processes 1-4,6-9: 1 Thread each

Total: 15 Processors used only

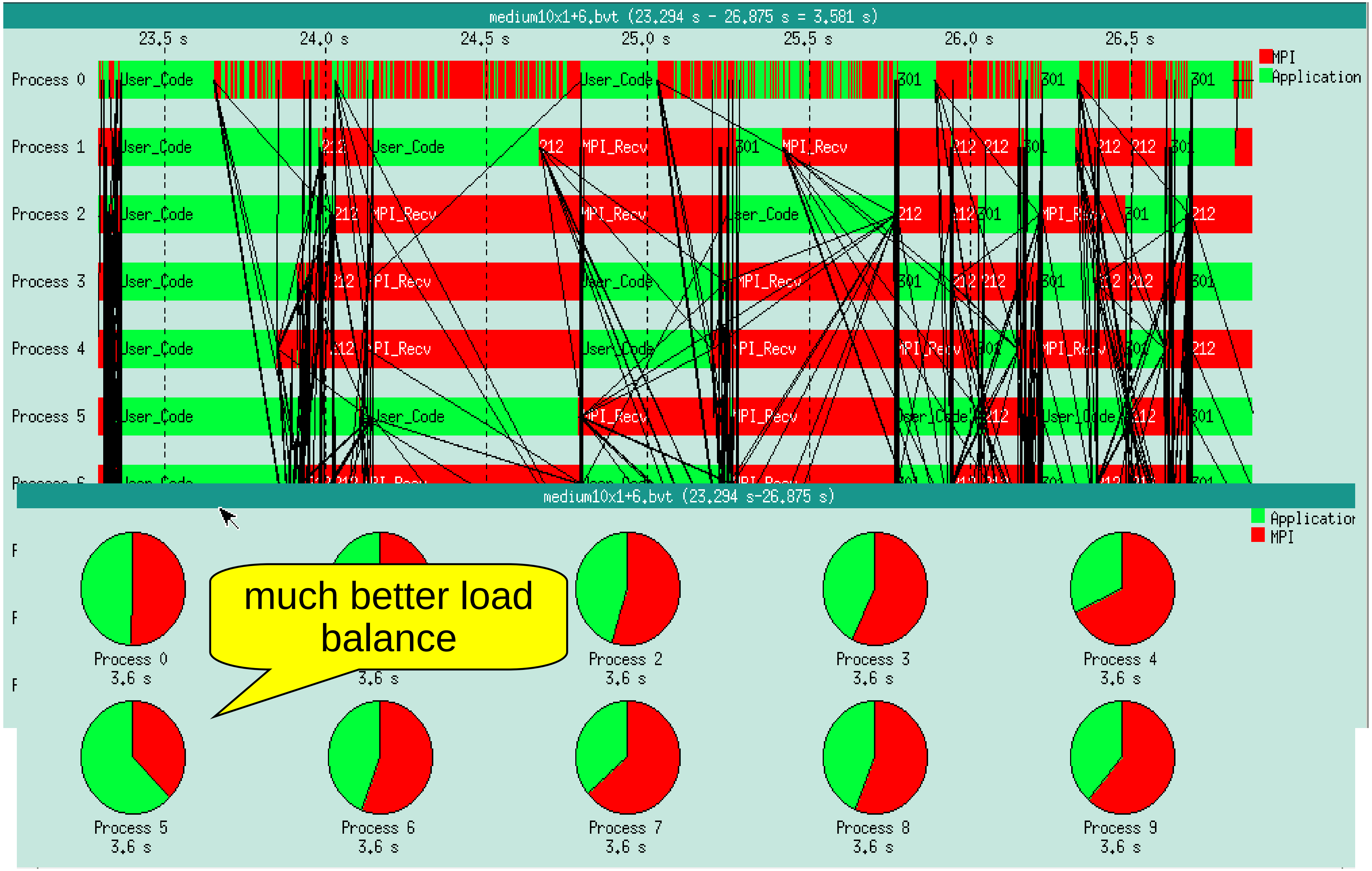
18.1 secs for 4 iterations versus
24.6 secs with 10x2 processors !

1.8 times higher efficiency

First Performance Experiments: 10x1+5 Processors

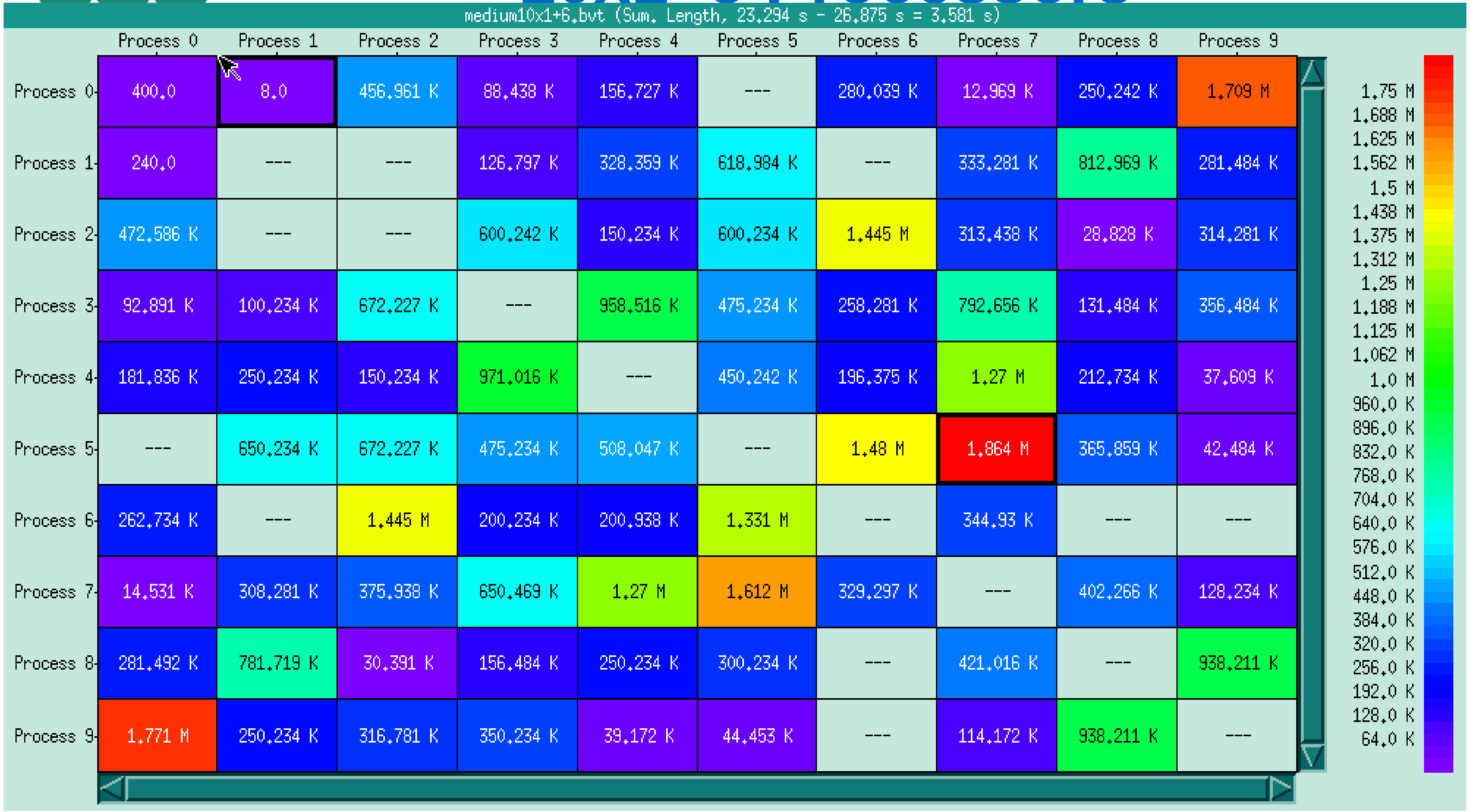


First Performance Experiments: 10x1+5 Processors



much better load balance

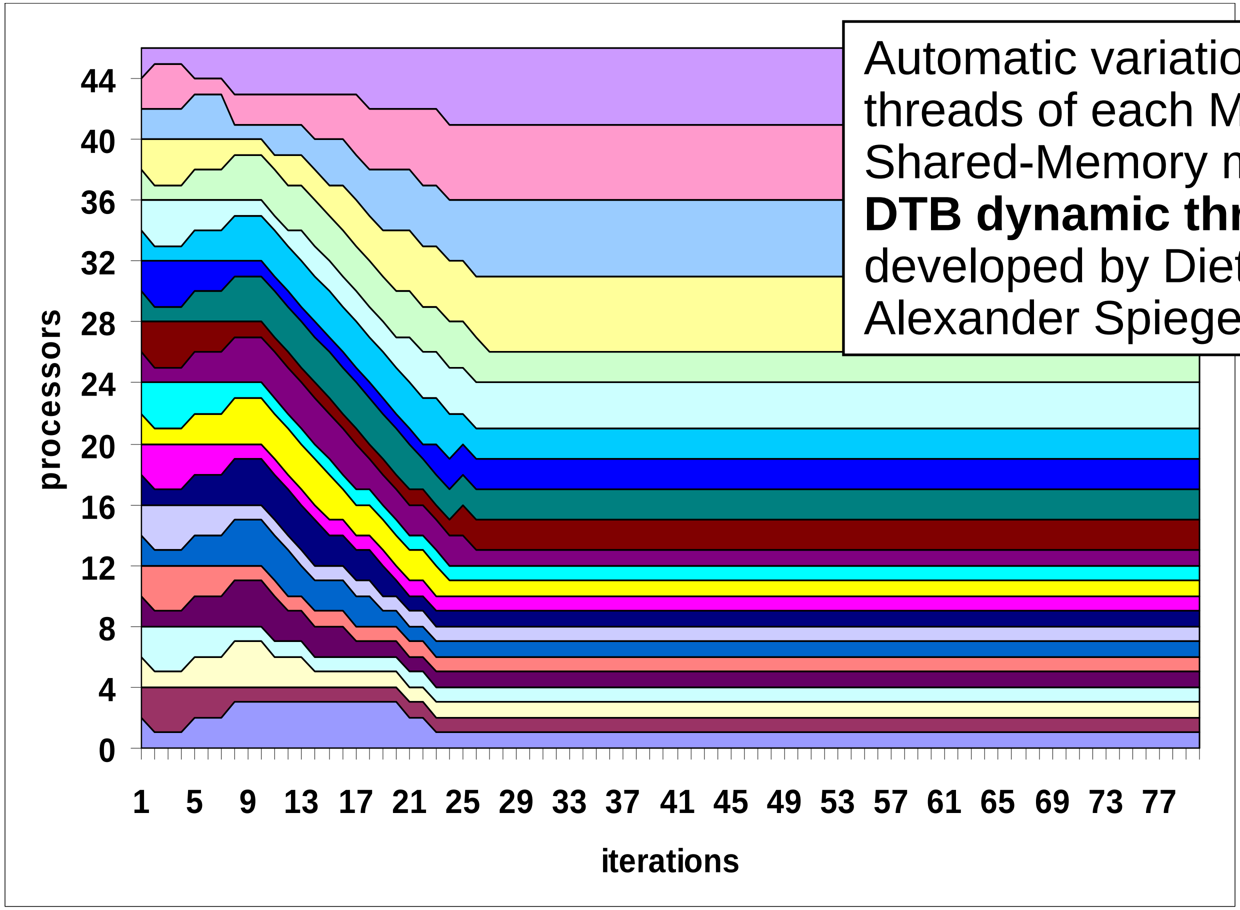
First Performance Experiments: 10x1+5 Processors



medium10x1+6.bvt (Sum. Length, 23.294 s - 26.875 s = 3.581 s)

	Process 0	Process 1	Process 2	Process 3	Process 4	Process 5	Process 6	Process 7	Process 8	Process 9
Process 0	400.0	8.0	456.961 K	88.438 K	156.727 K	---	280.039 K	12.969 K	250.242 K	1.709 M
Process 1	240.0	---	---	126.797 K	328.359 K	618.984 K	---	333.281 K	812.969 K	281.484 K
Process 2	472.586 K	---	---	600.242 K	150.234 K	600.234 K	1.445 M	313.438 K	28.828 K	314.281 K
Process 3	92.891 K	100.234 K	672.227 K	---	958.516 K	475.234 K	258.281 K	792.656 K	131.484 K	356.484 K
Process 4	181.836 K	250.234 K	150.234 K	971.016 K	---	450.242 K	196.375 K	1.27 M	212.734 K	37.609 K
Process 5	---	650.234 K	672.227 K	475.234 K	508.047 K	---	1.48 M	1.864 M	365.859 K	42.484 K
Process 6	262.734 K	---	1.445 M	200.234 K	200.938 K	1.331 M	---	344.93 K	---	---
Process 7	14.531 K	308.281 K	375.938 K	650.469 K	1.27 M	1.612 M	329.297 K	---	402.266 K	128.234 K
Process 8	281.492 K	781.719 K	30.391 K	156.484 K	250.234 K	300.234 K	---	421.016 K	---	938.211 K
Process 9	1.771 M	250.234 K	316.781 K	350.234 K	39.172 K	44.453 K	---	114.172 K	938.211 K	---

FLOWer on a Sun Fire 15K
23 MPI-Tasks x 2 OpenMP Threads



Automatic variation of the number of threads of each MPI process with in an Shared-Memory machine using the **DTB dynamic thread balancing library** developed by Dieter an Mey and Alexander Spiegel.

Summary

- **Tracing libraries** typically use the **MPI profiling interface**.
- It offers an easy-to-use collection of runtime information for MPI application after **re-linking**.
- The amount of **trace information** can be quite **large**,
  Tools for efficiently **controlling** and **filtering** trace data are essential.
- Trace data can be with the comfortable GUI s
- The performance of the **Flower CFD package** on a large **shared memory system** could be easily improved without any code changes:
  By **adjusting the number of threads per MPI task** manually, a much better **load balance** could be achieved.
- The **DTB** dynamic thread balancing library was developped to do this automatically