EZTRACE - ADT EZPerf

easy performance trace analyser

HASTARAN Matias - SYLVAND Guillaume - RUE François

INTRODUCTION

Modern HPC applications are complex:

- Complex hardware: NUMA architecture, hierarchical caches, accelerators;
- Hybrid programming models: MPI/OpenMP/Pthread.
  - Understand the performance of those application is diffucult.

Generating program behavior trace with a low overhead

- Code instrumentation

INTRODUCTION

EZPerf environment

- Project based on EZTrace performance tool
- EPI HiePACS

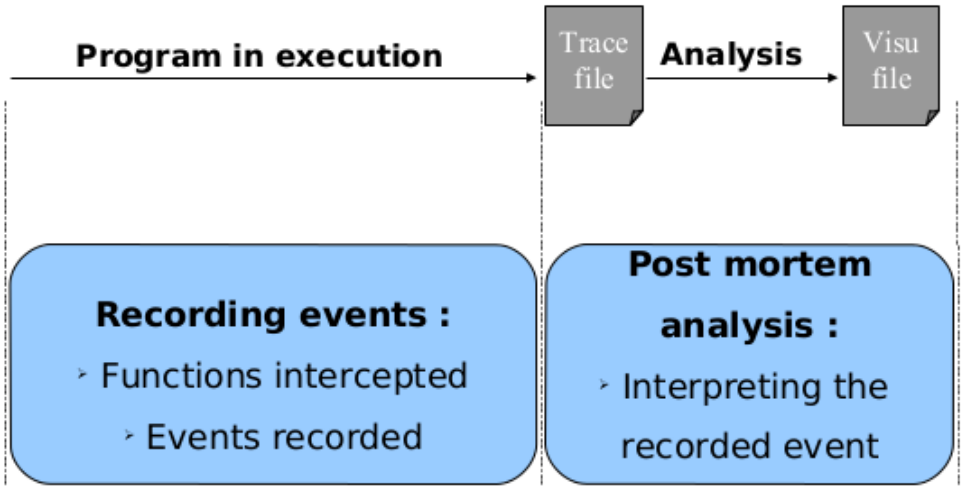
EZPerf Team

- Guillaume SYLVAND
  Scientist project manager (INRIA)
- François Trahay
  EZTrace's architect (Télécom SudParis)
- François Rué
  Technical project manager (INRIA)
- Matias HASTARAN
  Engineer(INRIA)

1

EZTRACE

Execution in two phases



- mpirun -np 16 eztrace myProg
- eztrace_convert /tmp/<username>_eztrace_log_rank_*

User module

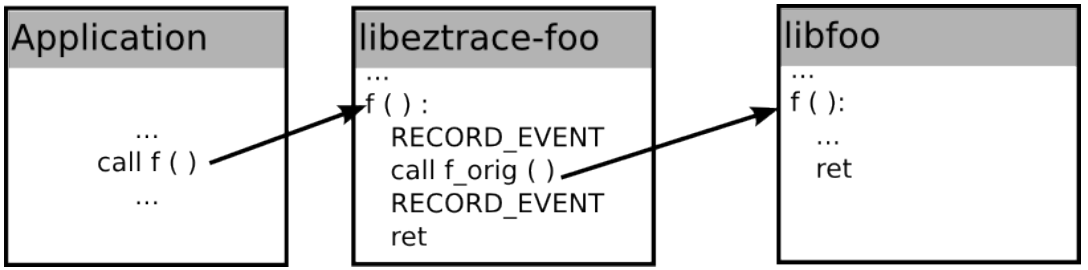
```
BEGIN_MODULE
NAME myModule
DESC "description for my module"
ID 42
void do_something()
END_MODULE
```

- eztrace_create_plugin plugin.tpl
- export EZTRACE_LIBRARY_PATH=module_path
- Run eztrace

2

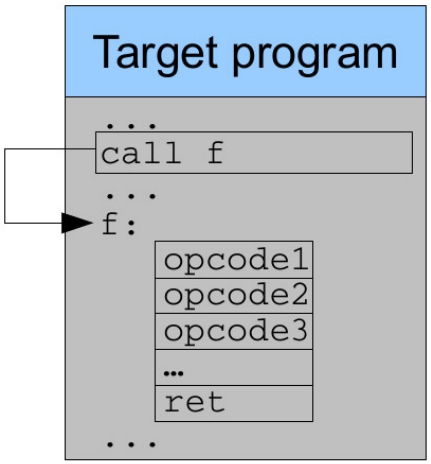
Interception mechanism in eztrace

LD_PRELOAD mechanism



- Pros:
  - Easy to use
  - Easy to run
  - Easy to do
- Cons:
  - Only C functions
  - Only in dynamic libraries

EZTrace Instrumentation

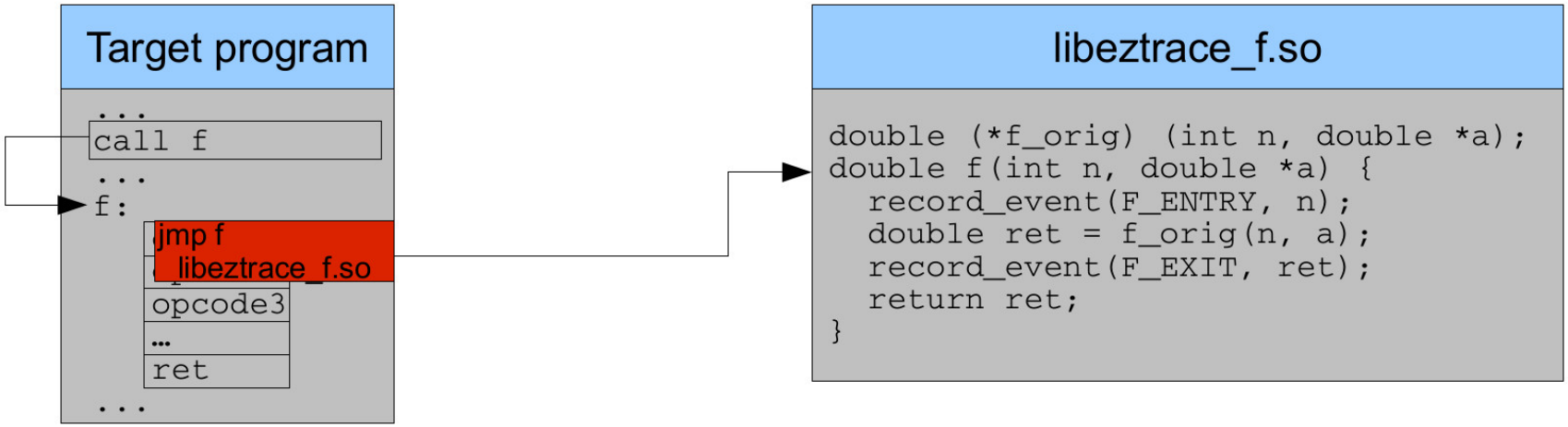


libeztrace_f.so

```
double (*f_orig) (int n, double *a);
double f(int n, double *a) {
  record_event(F_ENTRY, n);
  double ret = f_orig(n, a);
  record_event(F_EXIT, ret);
  return ret;
}
```

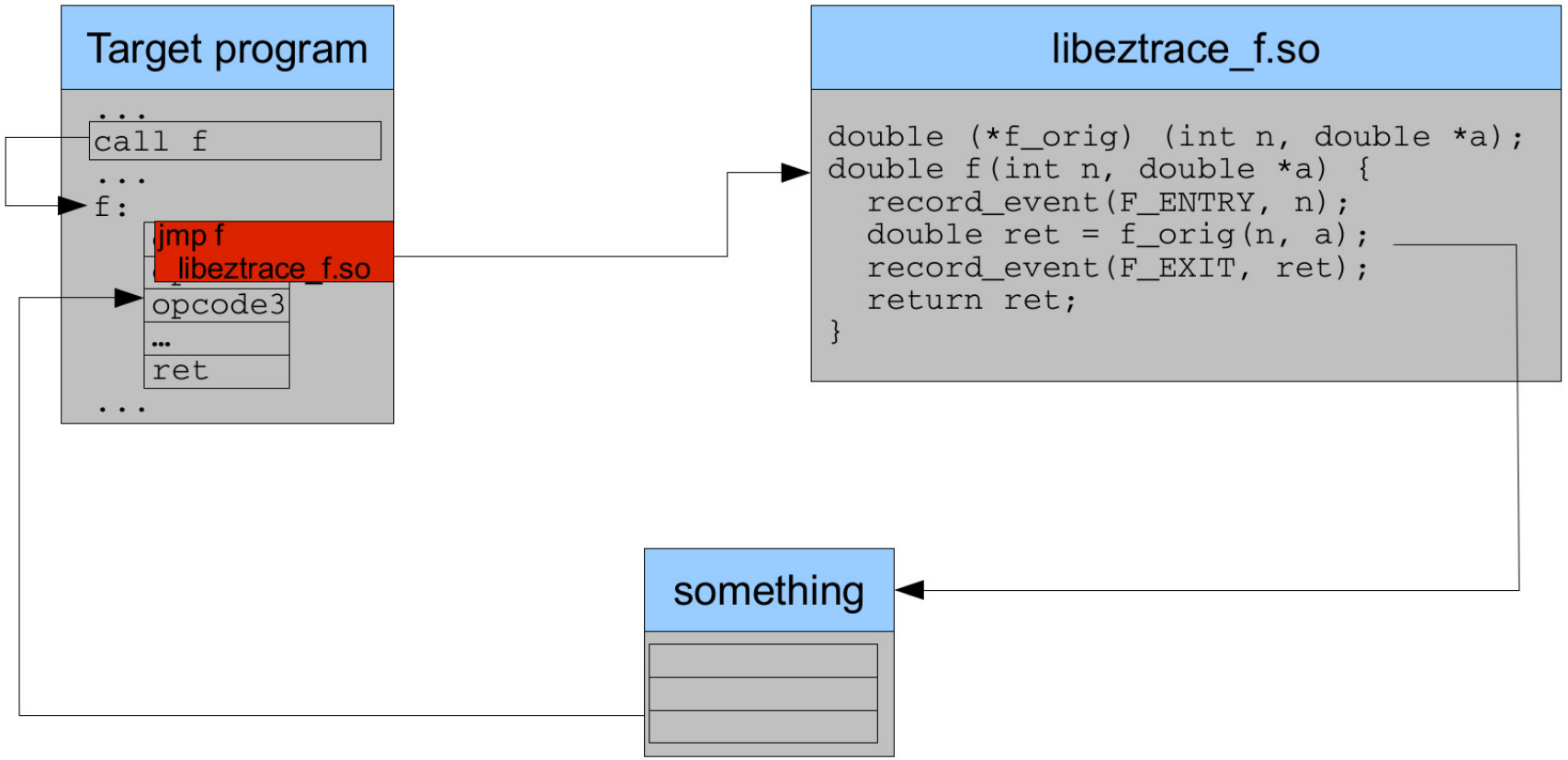
- Coarse-grain instrumentation
- Interception code is compiled into a shared library
- Mechanism similar to LD_PRELOAD

EZTrace Instrumentation



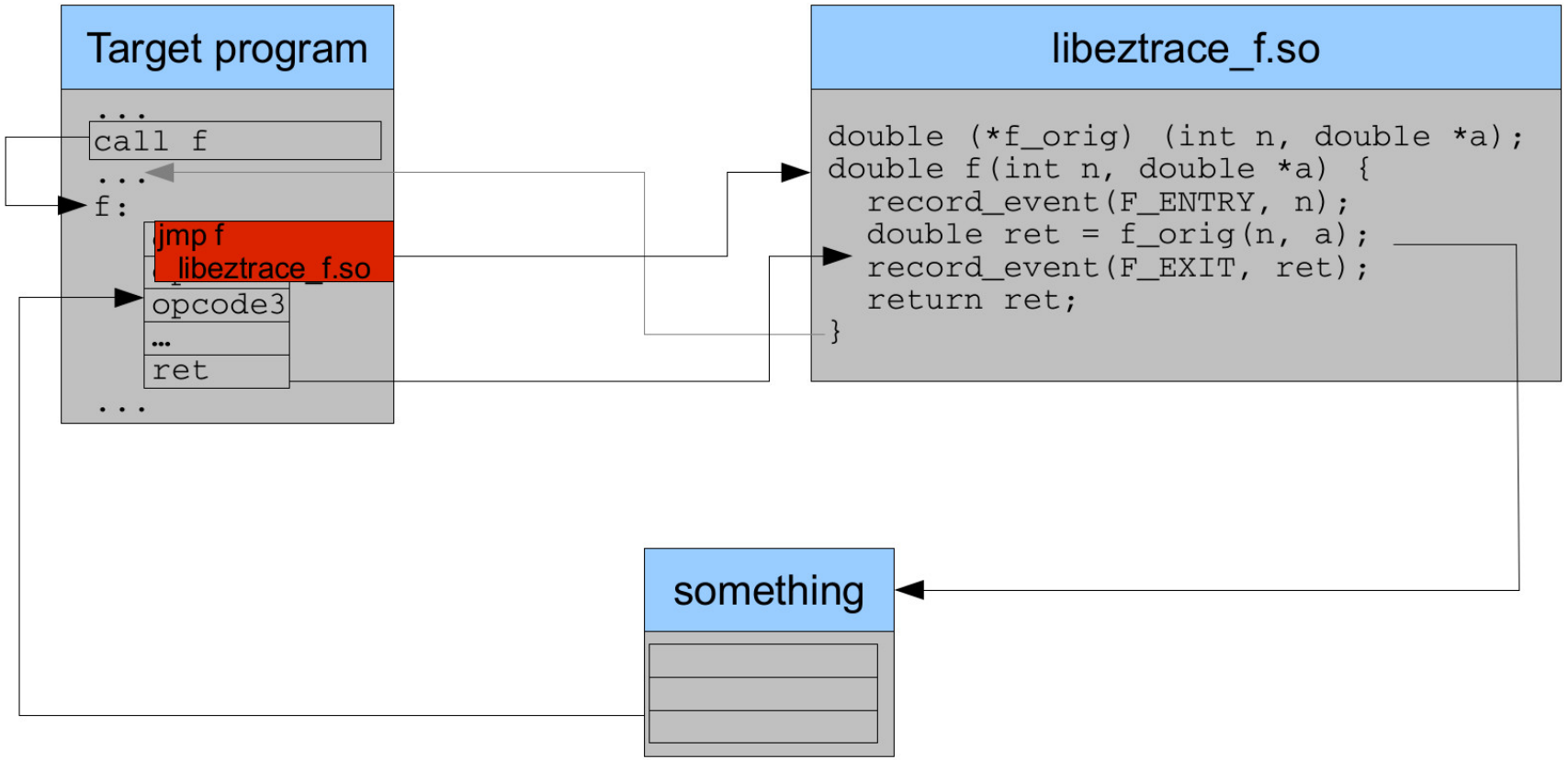
- f_orig contains the address of the original function f
- First f function opcodes are overwritten by a jump

EZTrace Instrumentation



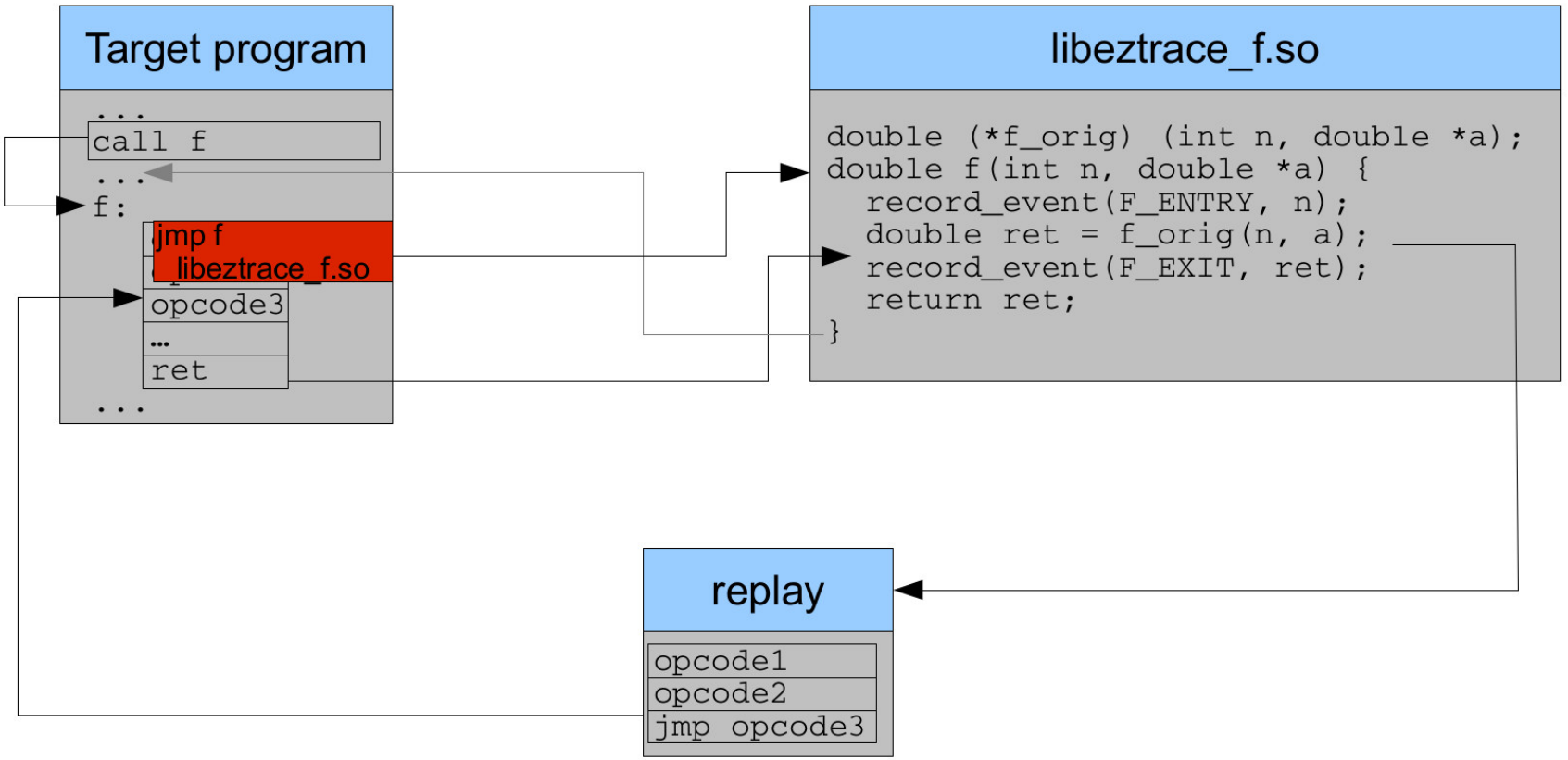
- A trampoline is created somewhere in memory
- f_orig will jump to this trampoline
- The trampoline will jump to the next opcode

EZTrace Instrumentation



- The return of original f funtion will jump to the next event of new f function
- The return of the new f funtion will jump to the next event

EZTrace Instrumentation



- The trampoline is fill with the overwritten opcodes
- During the execution it will play the start of the original f function

EZTrace instrumentation

- Function are instrumented using plugins at the application startup
- Functions are instrumented by modifying the target program in memory
- Pros:
  - Efficient and simple
  - Easy to use
  - Easy to extends: little architecture-dependent code
- Cons:
  - Limited to functions entries and exits

Evaluation

Interception method	no instrumentation	PIN	DynInst	EZTrace
Statically linked library	4.7 ns	1287 ns	1294 ns	245.2 ns
Shared library	5.3 ns	1293 ns	-	227.4 ns

Table: Function duration when recording events during a function call

Interception method	no instrumentation	PIN	DynInst	EZTrace
Execution time (s)	0.45	3.16	3.28	2.26
Overhead (ns / iteration)	+0	+341	+413	+227

Table: Execution time of a molecular dynamics simulation using 4 OpenMP threads

Reference paper Europar 2012 : `http://hal.archives-ouvertes.fr/docs/00/86/30/37/PDF/version_finale.pdf`

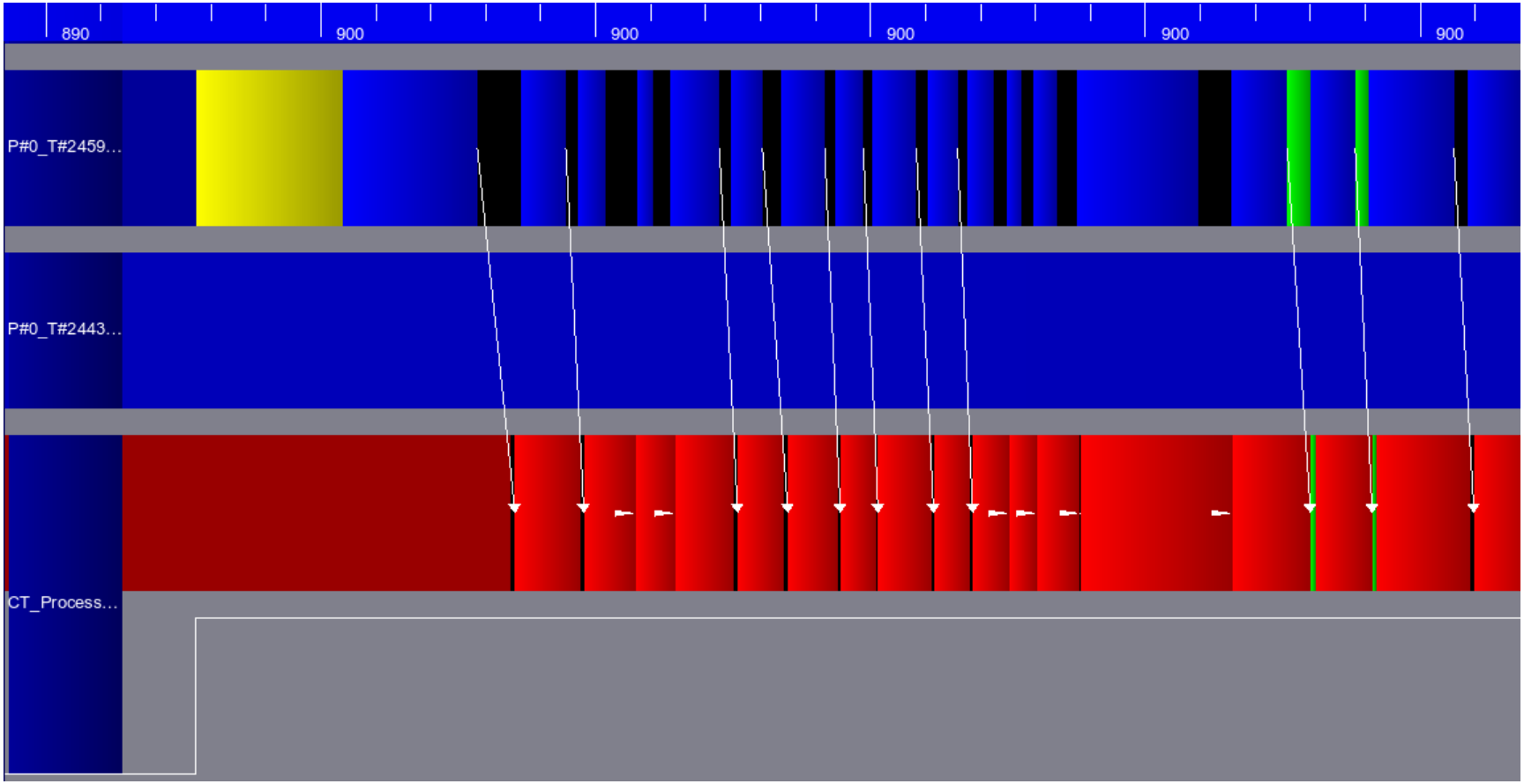
3

Recent Developments

Module

- MPI Module: Stabilization and new testsuite
- OpenMP : Module and eztrace_cc stabilization
- CUDA : New module !!

Cuda



Library replacement

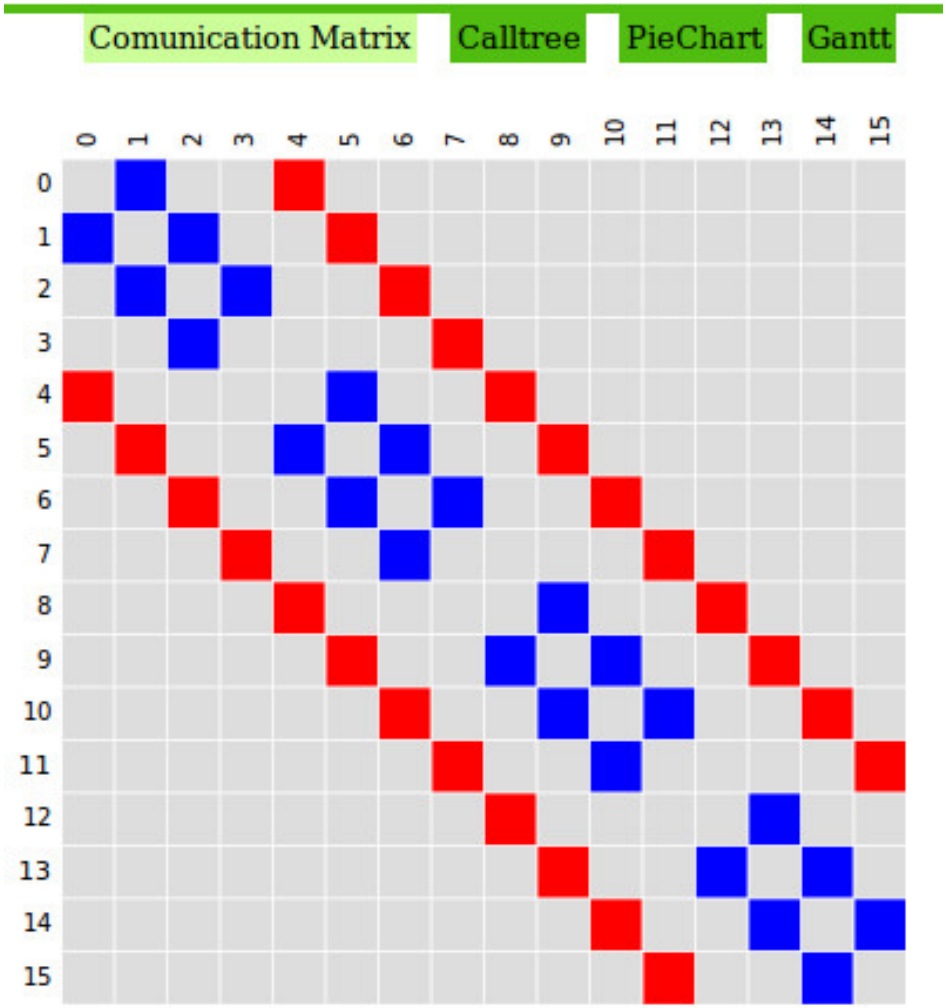
Goodbye FxT. Welcome LiTL.

- Easiest to install
- Trace file are lighter

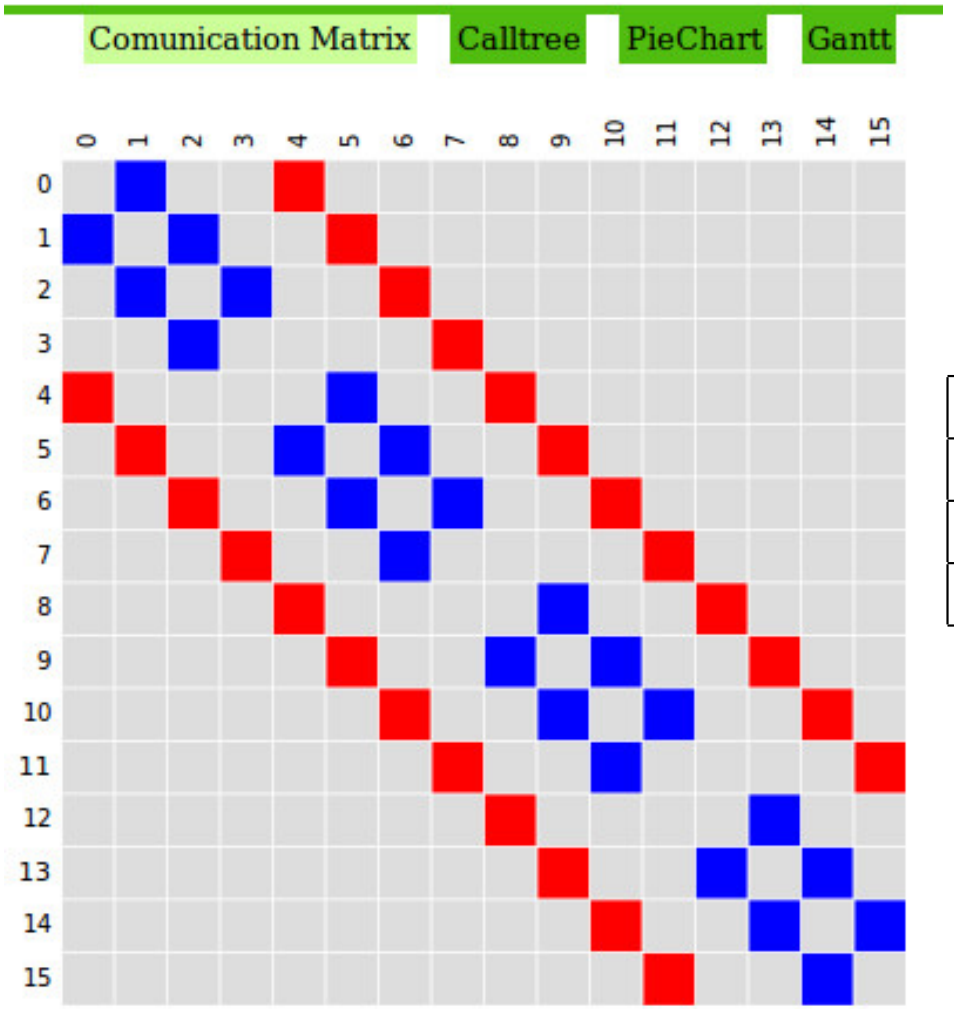
EZTrace statistic tool

```
MPI:
---
        MPI_SEND              :6720 calls
        MPI_IRECV             :6720 calls
        MPI_WAIT              :6720 calls
        MPI_BARRIER           :4 calls
        MPI_REDUCE            :4 calls
CT_Process #0 --        1680 messages sent
        Size of messages (byte):        min: 8  max: 56000      average: 27737  total: 46598912
        Time spent sending messages (ms):       min: 0.000908   max: 0.267776   average: 0.013695       total: 23.007082
        Time spent computing while sending messages (ms):       min: 0.000000   max: 0.000000   average: 0.000000       total: 0.00000
0
        Time spent in MPI_Send or waiting for a Isend to complete (ms): min: 0.000908   max: 0.267776   average: 0.013695       total:
 23.007082
        Time spent receiving messages (ms):     min: 0.003911   max: 0.315338   average: 0.019409       total: 32.606908
        Time spent computing while receiving messages (ms):     min: 0.002304   max: 0.313383   average: 0.013882       total: 23.3220
53
        Time spent in MPI_Recv or waiting for a Irecv to complete (ms): min: 0.000628   max: 0.088700   average: 0.005527       total:
 9.284855
CT_Process #1 --        1680 messages sent
        Size of messages (byte):        min: 8  max: 56000      average: 27737  total: 46598912
        Time spent sending messages (ms):       min: 0.000908   max: 0.578365   average: 0.014614       total: 24.551360
        Time spent computing while sending messages (ms):       min: 0.000000   max: 0.000000   average: 0.000000       total: 0.00000
0
        Time spent in MPI_Send or waiting for a Isend to complete (ms): min: 0.000908   max: 0.578365   average: 0.014614       total:
 24.551360
        Time spent receiving messages (ms):     min: 0.003492   max: 3.989680   average: 0.024444       total: 41.065636
        Time spent computing while receiving messages (ms):     min: 0.002235   max: 0.269941   average: 0.019668       total: 33.0427
60
        Time spent in MPI_Recv or waiting for a Irecv to complete (ms): min: 0.000628   max: 3.970682   average: 0.004776       total:
 8.022876
```

Web interface

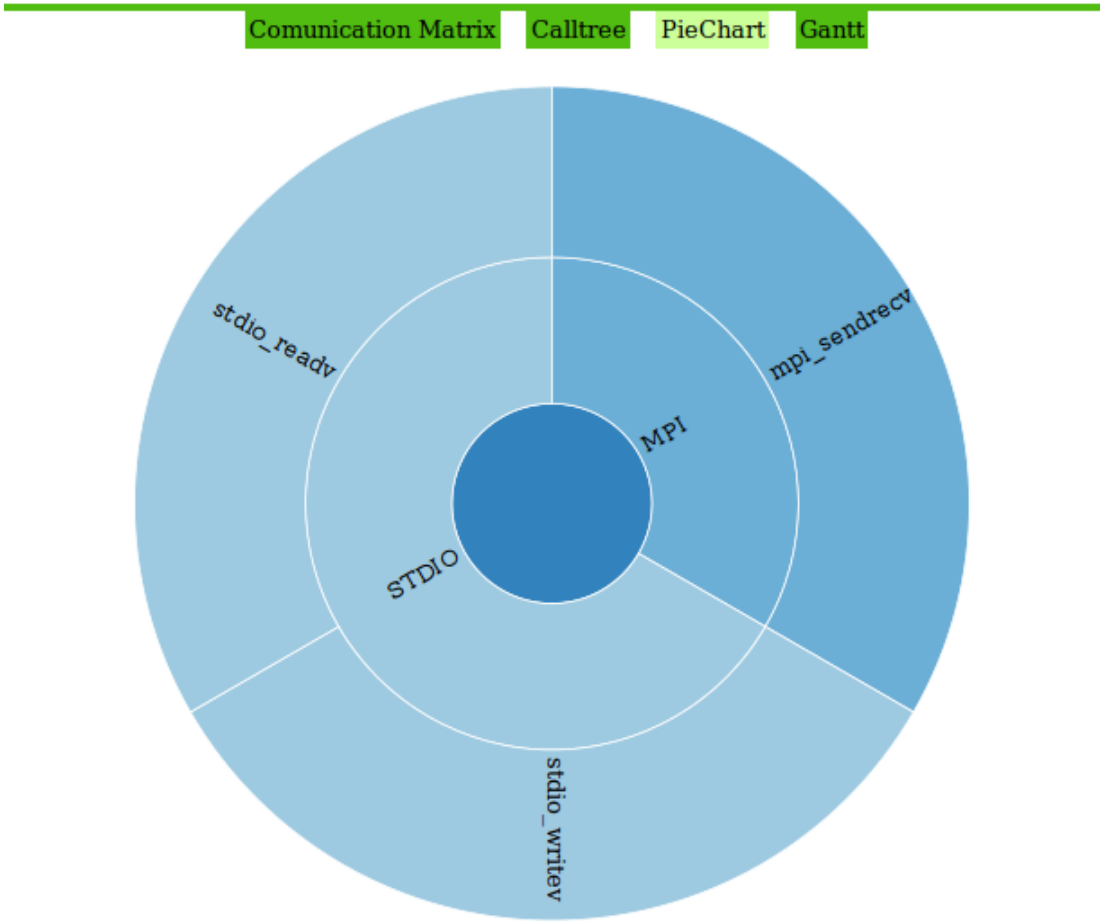


Web interface

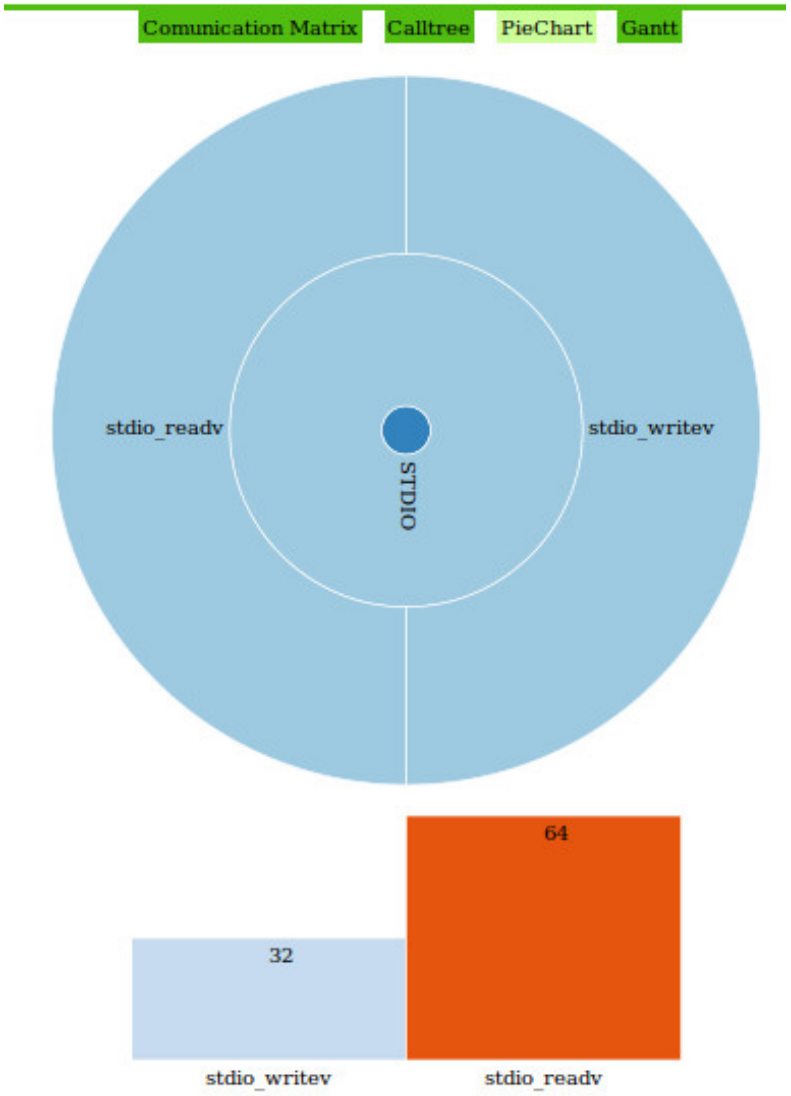


0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

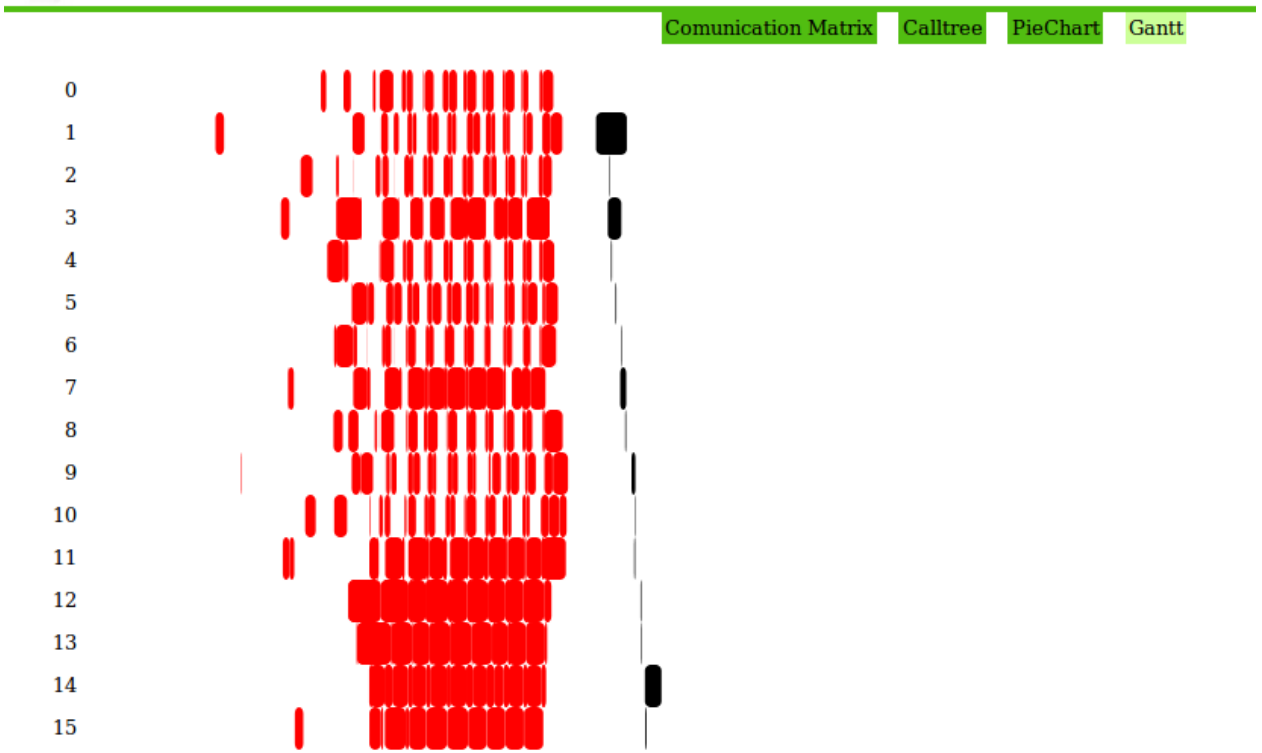
Web interface



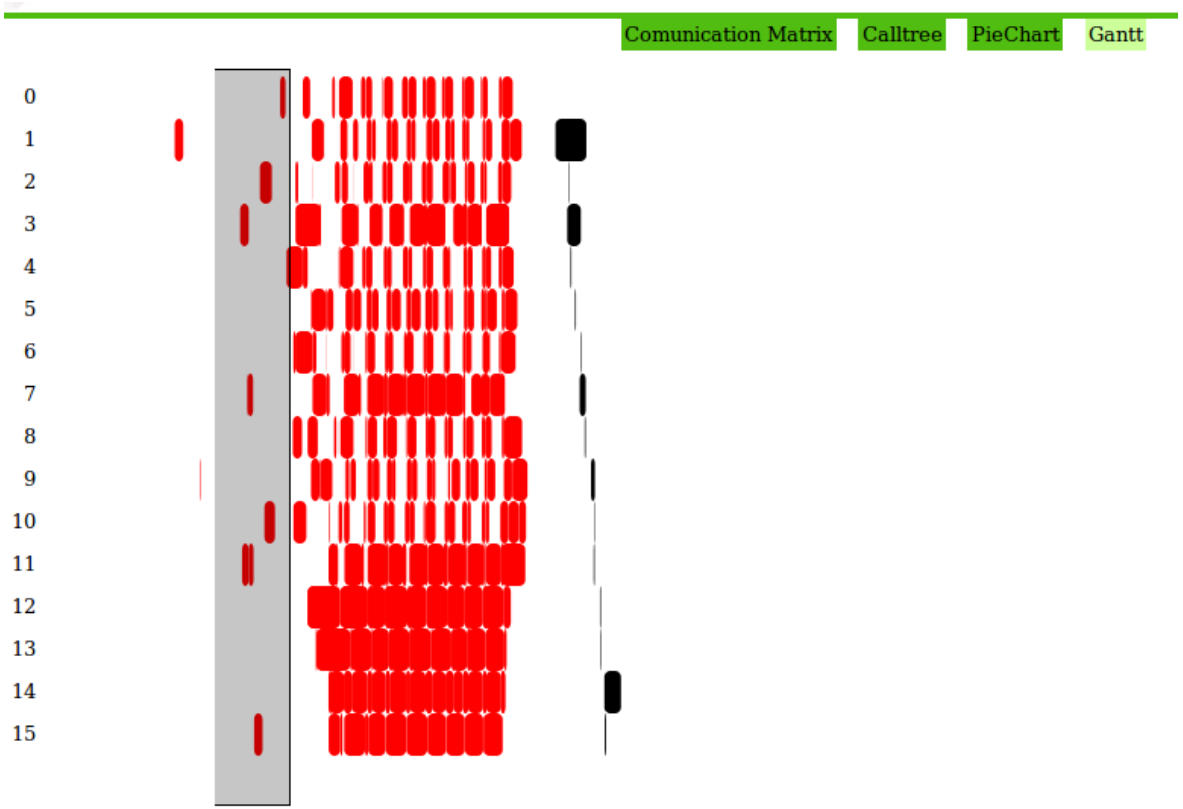
Web interface



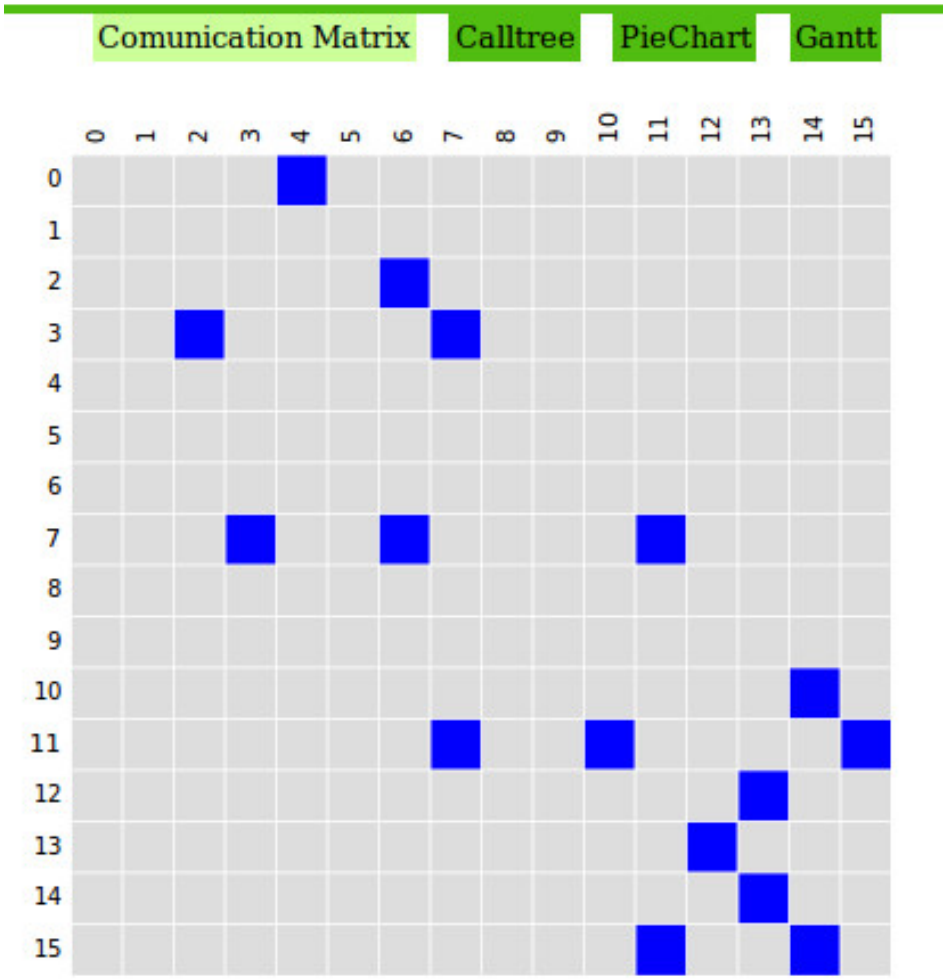
Web interface



Web interface



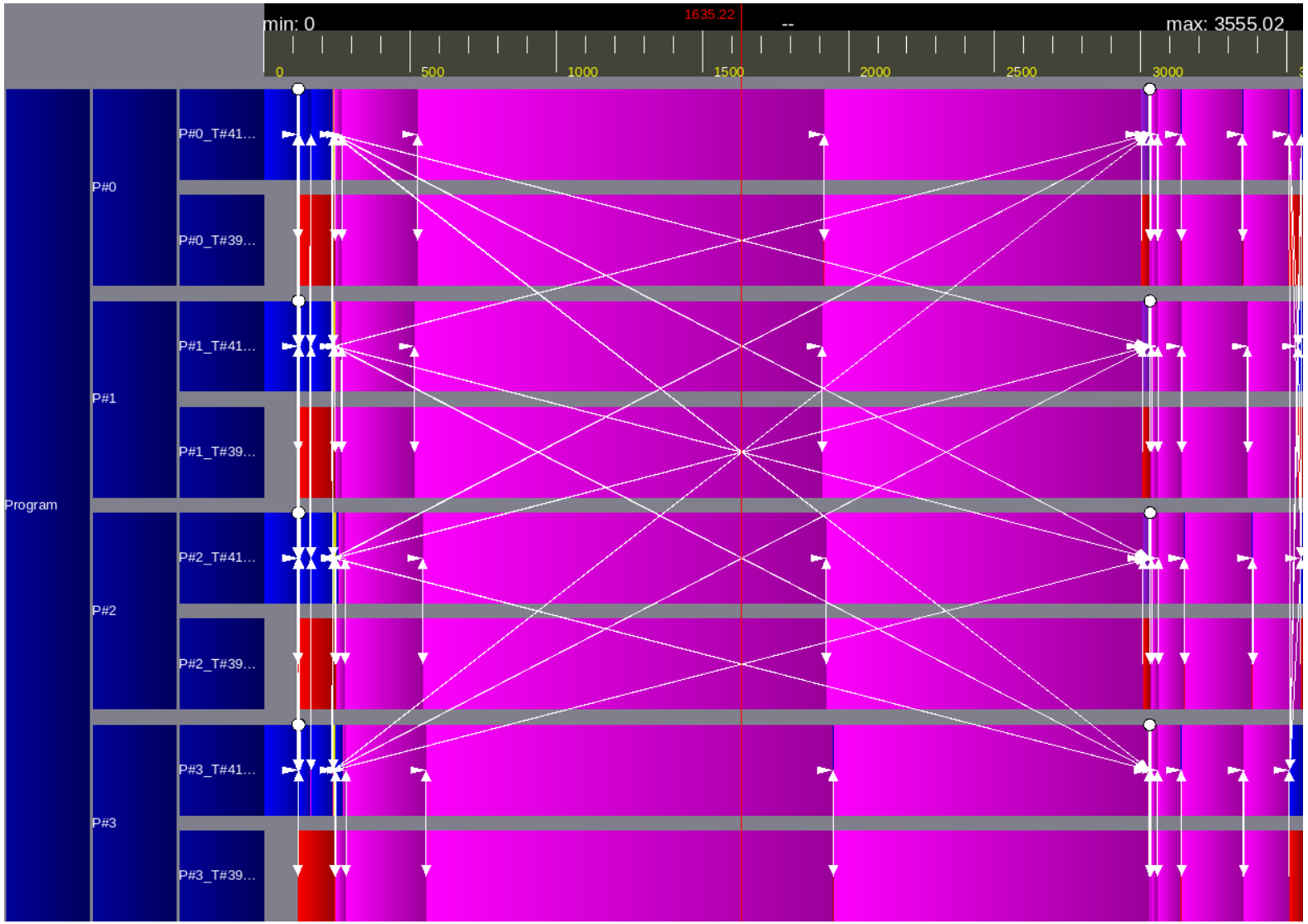
Web interface



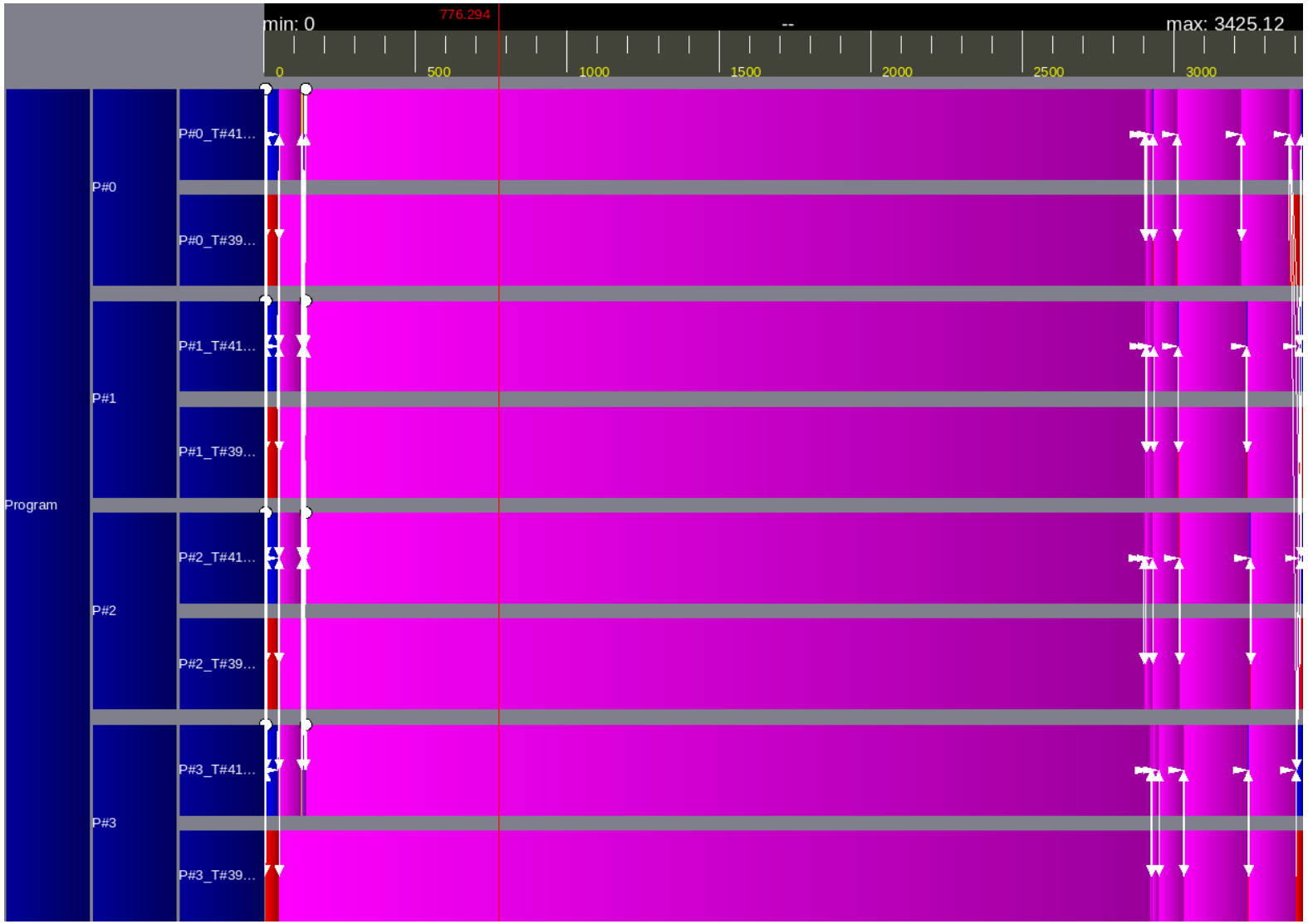
4

User Case : ScalFMM

ScalFMM


min: 0
1635.22
--
max: 3555.02
0
500
1000
1500
2000
2500
3000
Program
P#0
P#0_T#41...
P#0_T#39...
P#1
P#1_T#41...
P#1_T#39...
P#2
P#2_T#41...
P#2_T#39...
P#3
P#3_T#41...
P#3_T#39...

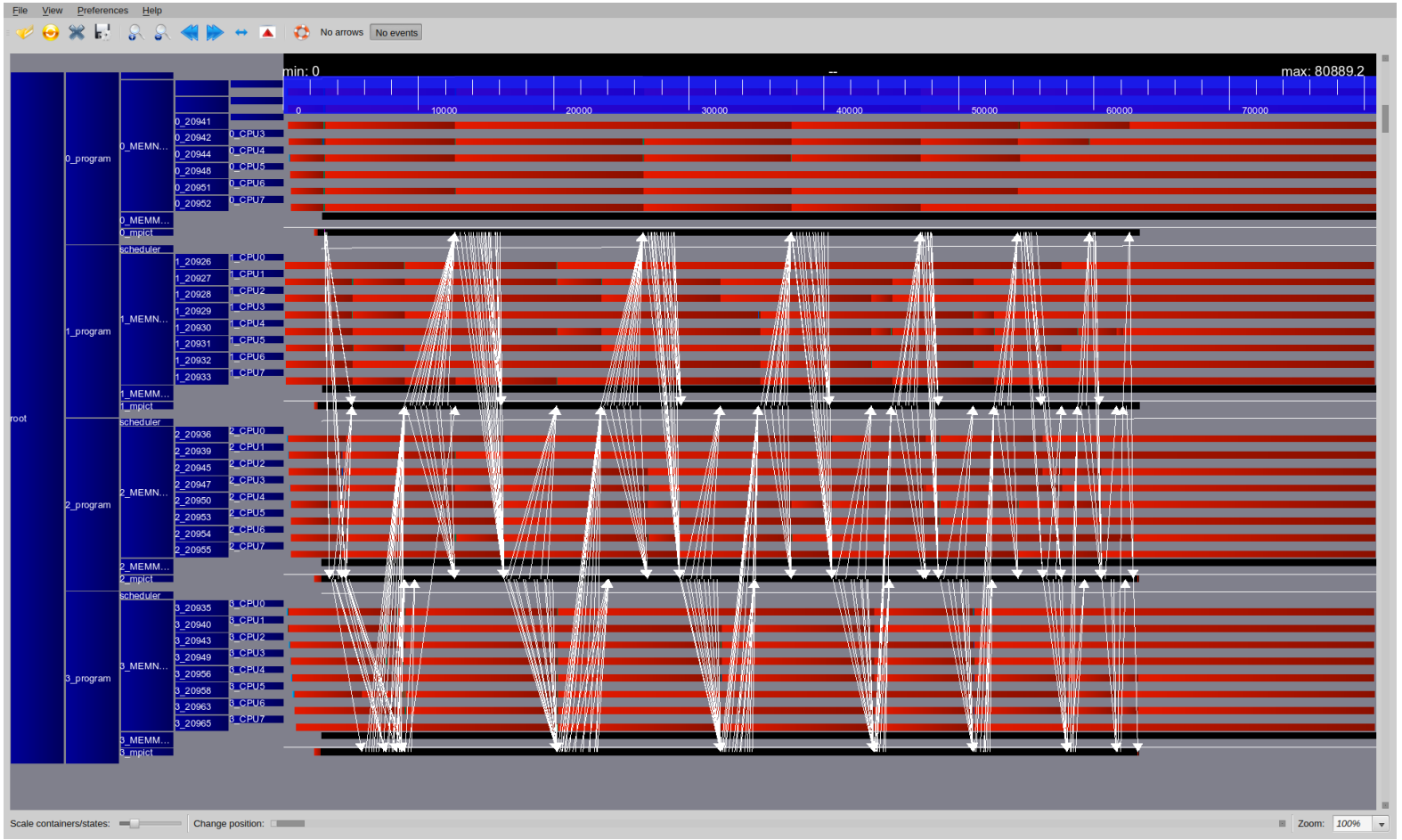
ScalFMM


min: 0
776.294
--
max: 3425.12
0
500
1000
1500
2000
2500
3000
Program
P#0
P#0_T#41...
P#0_T#39...
P#1
P#1_T#41...
P#1_T#39...
P#2
P#2_T#41...
P#2_T#39...
P#3
P#3_T#41...
P#3_T#39...

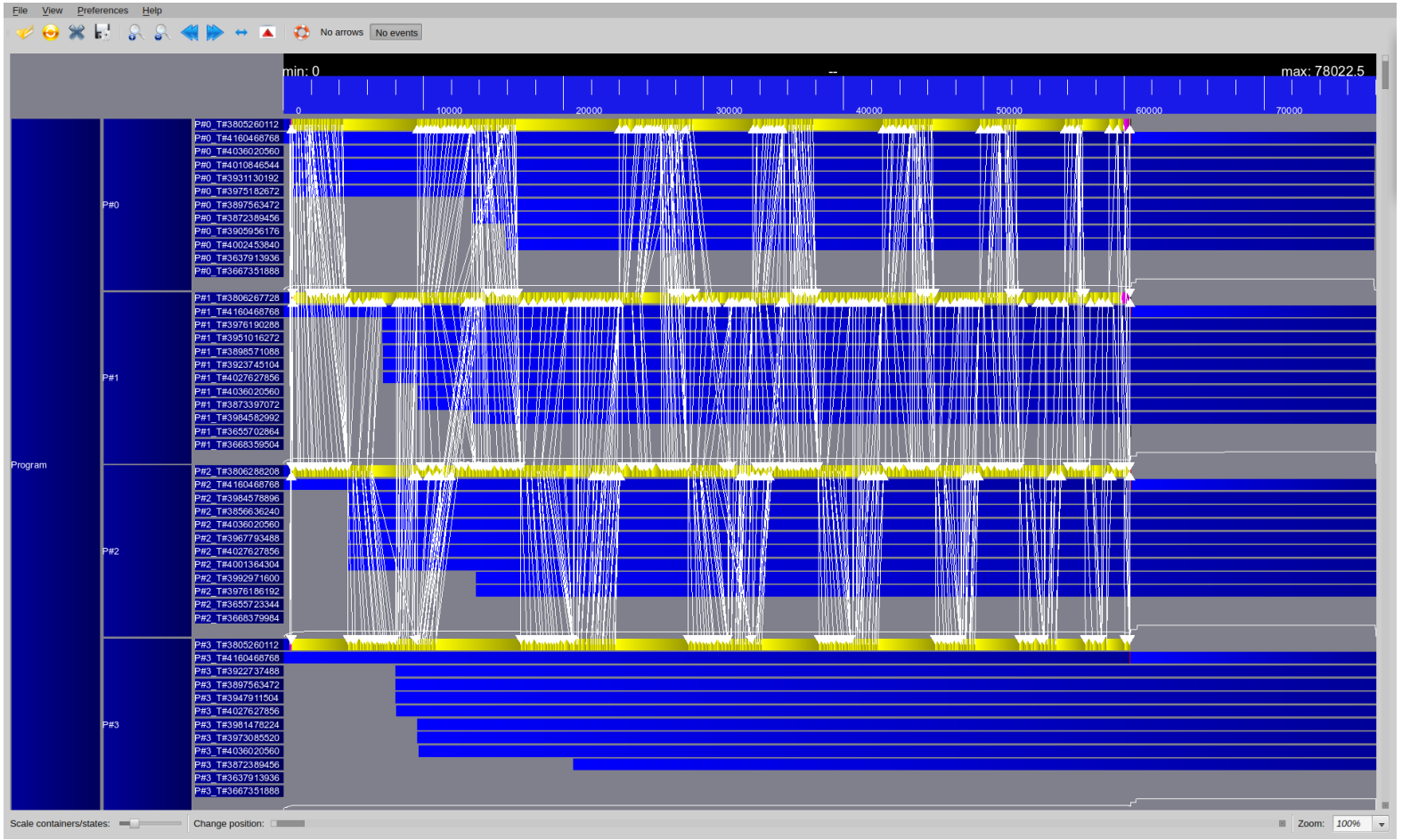
5

User Case : StarPU

StarPU


File View Preferences Help
No arrows
No events
min: 0
max: 80889.2
0
10000
20000
30000
40000
50000
60000
70000
root
0_program
1_program
2_program
3_program
0_MEMN...
1_MEMN...
2_MEMN...
3_MEMN...
scheduler
0_20941
0_20942
0_20944
0_20948
0_20951
0_20952
0_CPU3
0_CPU4
0_CPU5
0_CPU6
0_CPU7
1_20926
1_20927
1_20928
1_20929
1_20930
1_20931
1_20932
1_20933
1_CPU0
1_CPU1
1_CPU2
1_CPU3
1_CPU4
1_CPU5
1_CPU6
1_CPU7
2_20936
2_20939
2_20945
2_20947
2_20950
2_20953
2_20954
2_20955
2_CPU0
2_CPU1
2_CPU2
2_CPU3
2_CPU4
2_CPU5
2_CPU6
2_CPU7
3_20935
3_20940
3_20943
3_20949
3_20956
3_20958
3_20963
3_20965
3_CPU0
3_CPU1
3_CPU2
3_CPU3
3_CPU4
3_CPU5
3_CPU6
3_CPU7
Scale containers/states:
Change position:
Zoom:
100%

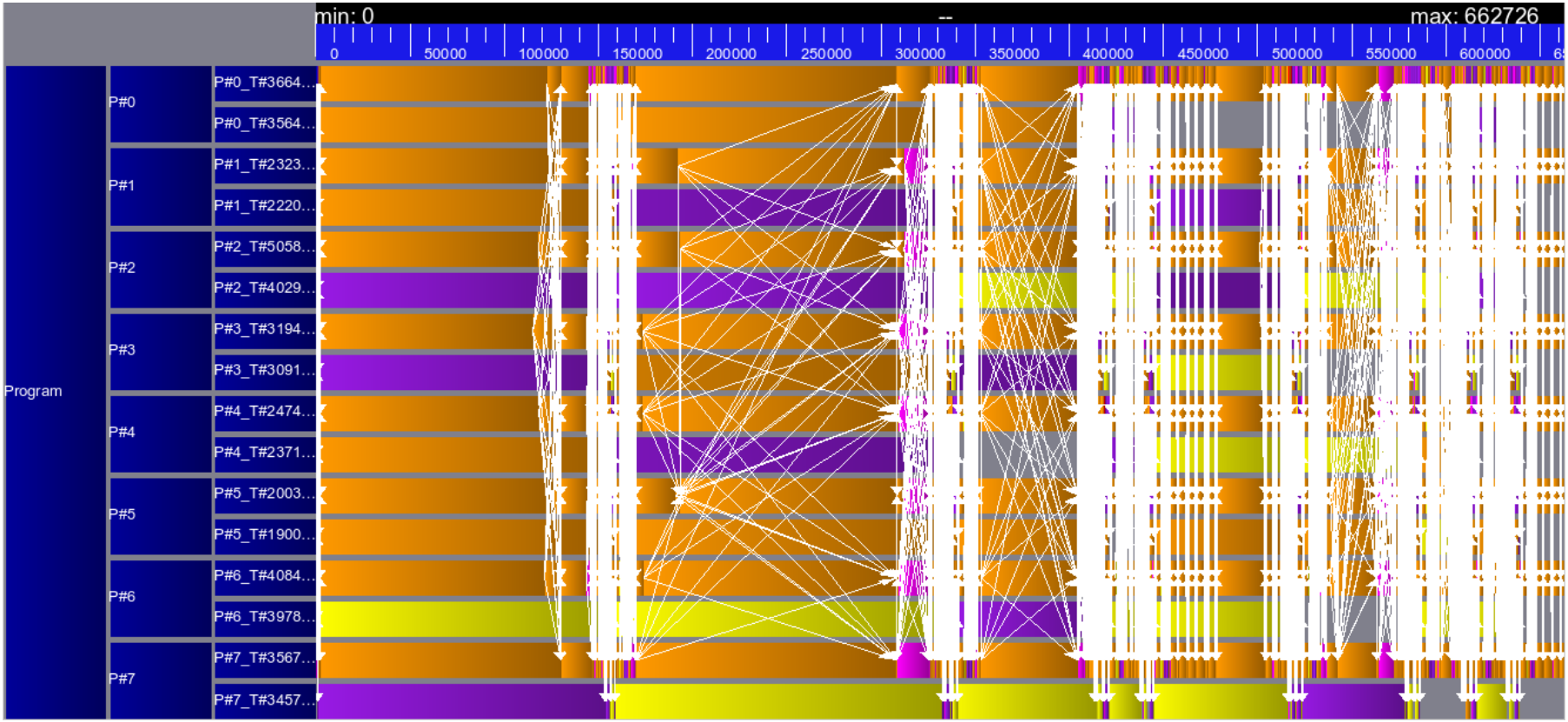
StarPU


File View Preferences Help
No arrows
No events
min: 0
max: 78022.5
0
10000
20000
30000
40000
50000
60000
70000
Program
P#0
P#0_T#3805260112
P#0_T#4160468768
P#0_T#4036020560
P#0_T#4010846544
P#0_T#3931130192
P#0_T#3975182672
P#0_T#3897563472
P#0_T#3872389456
P#0_T#3905956176
P#0_T#4002453840
P#0_T#3637913936
P#0_T#3667351888
P#1
P#1_T#3806267728
P#1_T#4160468768
P#1_T#3976190288
P#1_T#3951016272
P#1_T#3898571088
P#1_T#3923745104
P#1_T#4027627856
P#1_T#4036020560
P#1_T#3873397072
P#1_T#3984582992
P#1_T#3655702864
P#1_T#3668359504
P#2
P#2_T#3806288208
P#2_T#4160468768
P#2_T#3984578896
P#2_T#3856636240
P#2_T#4036020560
P#2_T#3967793488
P#2_T#4027627856
P#2_T#4001364304
P#2_T#3992971600
P#2_T#3976186192
P#2_T#3655723344
P#2_T#3668379984
P#3
P#3_T#3805260112
P#3_T#4160468768
P#3_T#3922737488
P#3_T#3897563472
P#3_T#3947911504
P#3_T#4027627856
P#3_T#3981478224
P#3_T#3973085520
P#3_T#4036020560
P#3_T#3872389456
P#3_T#3637913936
P#3_T#3667351888
Scale containers/states:
Change position:
Zoom:
100%

6

User Case : PaMPA

PaMPA


min: 0
max: 662726
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
500000
550000
600000
Program
P#0
P#0_T#3664...
P#0_T#3564...
P#1
P#1_T#2323...
P#1_T#2220...
P#2
P#2_T#5058...
P#2_T#4029...
P#3
P#3_T#3194...
P#3_T#3091...
P#4
P#4_T#2474...
P#4_T#2371...
P#5
P#5_T#2003...
P#5_T#1900...
P#6
P#6_T#4084...
P#6_T#3978...
P#7
P#7_T#3567...
P#7_T#3457...

7

Thank you!

Contact

- EZTrace is freely available under the CeCILL-B license
- Website : http://eztrace.gforge.inria.fr/
- Contact : eztrace-devel@lists.gforge.inria.fr