ADT EZPerf

EZPerf easy performance trace analyser

SYLVAND Guillaume - HASTARAN Matias

Contents

1. ADT EZPerf: general presentation
2. EZTrace tool
3. C2S@Exa ecosystem: EZPerf main goals

1

ADT EZPerf: general presentation

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)

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)

Collaborators

TELECOM
SudParis

INNOVATIVE
COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE UT

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Goals of the ADT

- Improvement and extension of the prototype
- Realisation of high-level interfaces
- Validate EZTrace on research and industrial application cases
- Enhance the technical and user complete documentation

Goals of the ADT

- Improvement and extension of the prototype

Task list

✍ Add some libs support (MPF, ...)

✍ Add some runtime support (StarPU, XKaapi, ...)

✍ New functionalities (Calltree, ...)

✍ Add a system to automaticaly bench and test (based on CI)

- Realisation of high-level interface
- Validate EZTrace on research and industrial application cases
- Add a complete technical and user complete documentation

Goals of the ADT

- Improvement and extension of the prototype
- Realisation of high-level interface

Task list

✍ Optimisation of post-mortem treatment module

✍ Think about new way to represent information

- Validate EZTrace on research and industrial application cases
- Add a complete technical and user complete documentation

Goals of the ADT

- Improvement and extension of the prototype
- Realisation of high-level interface
- Validate EZTrace on research and industrial application cases

Task list

✍ Work in collaboration with industrial and research team

- Add a complete technical and user complete documentation

Goals of the ADT

- Improvement and extension of the prototype
- Realisation of high-level interface
- Validate EZTrace on research and industrial application cases
- Add a complete technical and user complete documentation

Task list

✍ Add a documentation system (wiki, doxygen, ...)

2

EZTrace tool

EZTrace

Easy to use trace generator

EZTrace is a tool that aims at generating automatically execution trace from HPC (High Performance Computing) programs.

Capabilities

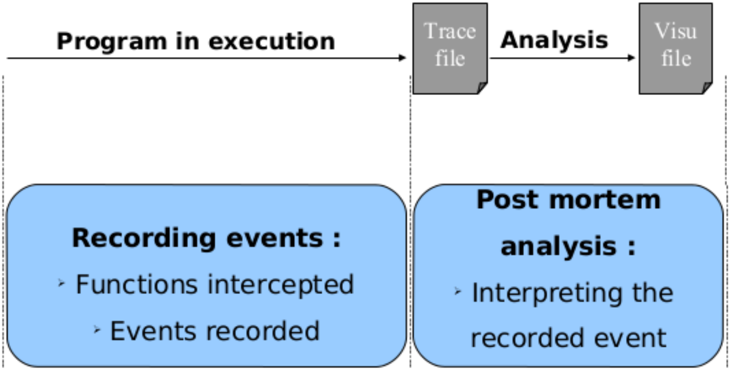
Especialy design for parallel applications, EZTrace is done in two levels:

- Pre - defined plugins
- User - defined plugins

Eztrace work in two phases :

- Interception phase without any instrumentation needed
- Post mortem analysis

Two phase mechanism



Recording events :

- Functions intercepted
- Events recorded

Post mortem analysis :

- Interpreting the recorded event

EZTrace

Easy to use trace generator

Pre - defined plugins:

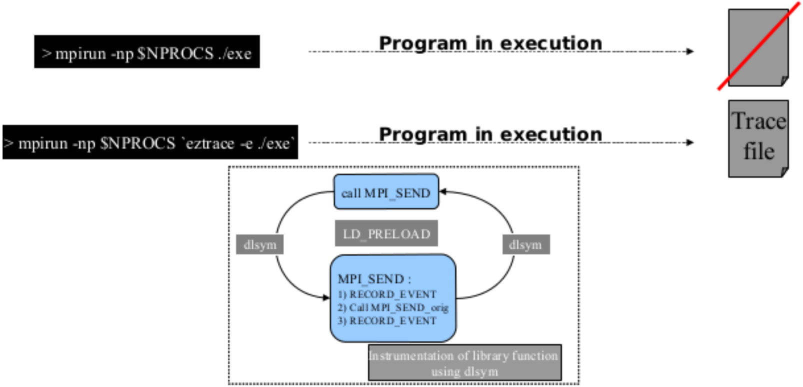
- mpi
- openmp
- pthread
- memory
- stdio
- CUDA

Capabilities

User - defined plugins:

- Own functions inner user programs
- Extern dynamic libraries

Interception mechanism



```
> mpirun -np $NPROCS ./exe
> mpirun -np $NPROCS `eztrace -e ./exe`
```

User - defined plugin

To obtain user - defined plugin:

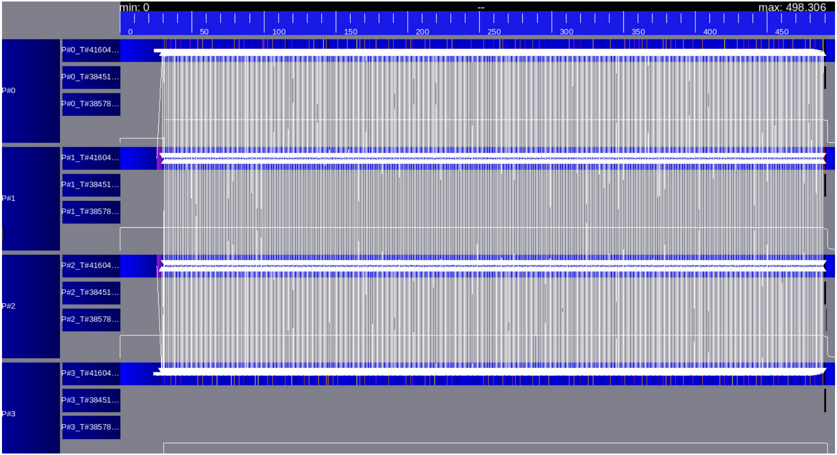
- Define functions of interest in a text file like this:

Module definition

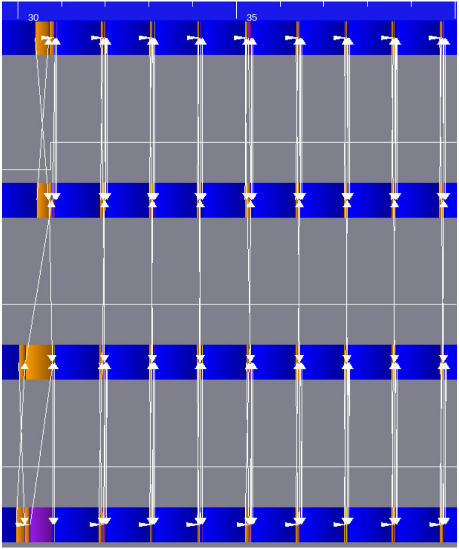
```
BEGIN_MODULE
NAME mxm
DESC ”module for mxm function ”
double** makmat( int m, int n )
void gendat(int lda,int m,int l,int n,double **a,double **b )
END_MODULE
```

- Apply: `eztrace_create_plugin module`
- Go to the output directory generated and type make
- Go to the prog directory and precise EZTRACE_TRACE and EZTRACE_LIBRARY_PATH env variables
- Run you prog

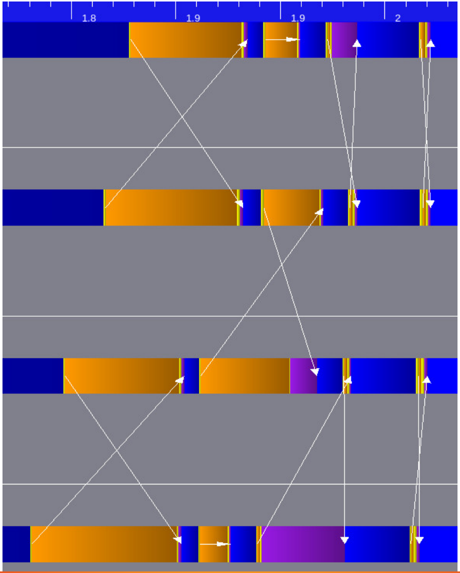
Trace


min: 0
--
max: 498.306
0
50
100
150
200
250
300
350
400
450
P#0
P#0_T#41604...
P#0_T#38451...
P#0_T#38578...
P#1
P#1_T#41604...
P#1_T#38451...
P#1_T#38578...
P#2
P#2_T#41604...
P#2_T#38451...
P#2_T#38578...
P#3
P#3_T#41604...
P#3_T#38451...
P#3_T#38578...

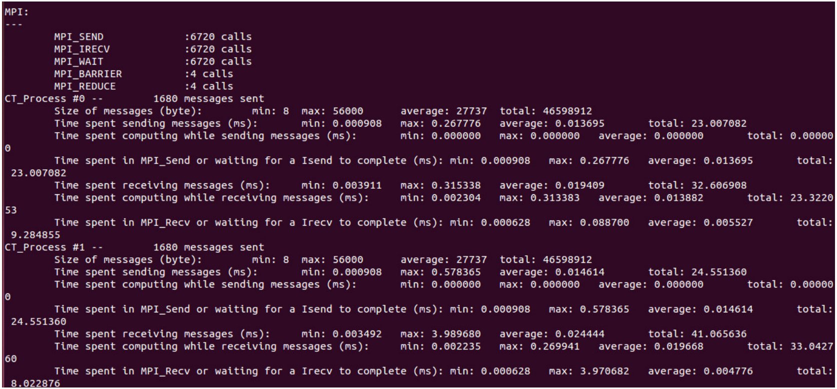
Trace



Trace



Eztrace statistics



```
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
```

3

C2S@Exa ecosystem: EZPerf main goals

Execution time : New features

- Add runtime treatment modules (StarPU, XKaapi...)
- Add some accelerator language support (CUDA, OpenCL...)
- Add Xeon Phi support (API : SCIF)
- Add possibility to adjust the level of detail
- Possibility to adjust the level of detail at specific moment
- Add a finest memory allocation support
- Possibility to use code instrumentation

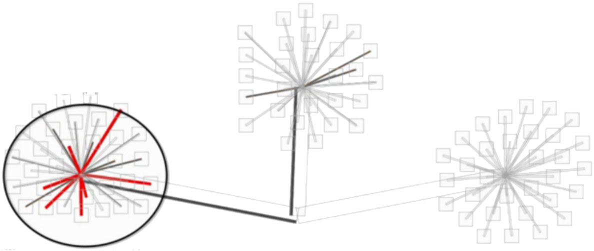
Post mortem treatement : New features

- Add new statistics information (execution time, mflops, network bandwidth, disk bandwidth)
- Optimisation of post-mortem analysis
- To be able to follow the progression simultaneously between the GanttChart and the DAG

New features

- Integrate tools to perform studies of scalability
- Works around representation module with other EPI expertise (like MESCAL EPI)
- Hide EZTrace call
- Detection of communication inconsistencies thanks to pattern recognition

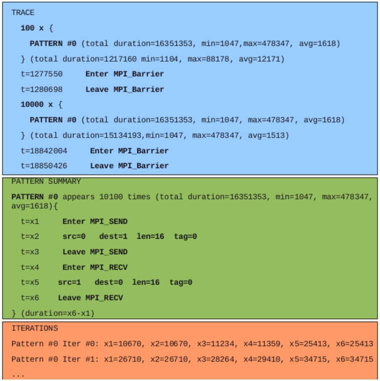
SimGrid



New features

- Integrate tools to perform studies of scalability
- Works around representation module with other EPI expertise (like MESCAL EPI)
- Hide EZTrace call
- Detection of communication inconsistencies thanks to pattern recognition

Pattern recognition format



```
TRACE
  100 x {
    PATTERN #0 (total duration=16351353, min=1047,max=478347, avg=1618)
  } (total duration=1217160 min=1104, max=88178, avg=12171)
  t=1277550     Enter MPI_Barrier
  t=1280698     Leave MPI_Barrier
  10000 x {
    PATTERN #0 (total duration=16351353, min=1047, max=478347, avg=1618)
  } (total duration=15134193,min=1047, max=478347, avg=1513)
  t=18842004     Enter MPI_Barrier
  t=18850426     Leave MPI_Barrier
PATTERN SUMMARY
PATTERN #0 appears 10100 times (total duration=16351353, min=1047, max=478347,
avg=1618){
  t=x1     Enter MPI_SEND
  t=x2     src=0   dest=1  len=16  tag=0
  t=x3     Leave MPI_SEND
  t=x4     Enter MPI_RECV
  t=x5    src=1   dest=0  len=16  tag=0
  t=x6    Leave MPI_RECV
} (duration=x6-x1)
ITERATIONS
Pattern #0 Iter #0: x1=10670, x2=10670, x3=11234, x4=11359, x5=25413, x6=25413
Pattern #0 Iter #1: x1=26710, x2=26710, x3=28264, x4=29410, x5=34715, x6=34715
...
```

- Loss-less trace compression
- No need to load the whole file to display the trace
- A specific iteration can be loaded on-demand

Interested people

- Avalon EPI - F.Suter

  Replace actual trace tool

- Cadarache CEA - G.Latu

  Insterested by memory consumption information

- Sinetics department (EDF R&D) - B.Lathuiliere

  Use a non intrusive tool

- EADS Innovation Works - G.Sylvand

  Use a tool wich offer a minimal impact on the performances

Contact

- EZTrace is freely available under the GPL-2 license
- Website : http://eztrace.gforge.inria.fr/
- Conact : eztrace-devel@lists.gforge.inria.fr
- APP :
  - Ref CNRS : 5309
  - Ref AQvalo : 2012-202

4

Question ?