Performance Analysis on Energy Efficient High-Performance Architectures

Roman Iakymchuk and François Trahay

Institut Mines-Télécom – Télécom SudParis
roman.iakymchuk@telecom-sudparis.eu

Cluster Computing Workshop
Lviv, Ukraine
May 3rd, 2013

Outline

Is Energy Consumption the Issue? (I)

K supercomputer at the RIKEN AICS (#1 in 2011)

Source: http://www.telegraph.co.uk/

- LINPACK reported 8.162 PFLOPS

Is Energy Consumption the Issue? (I)

K supercomputer at the RIKEN AICS (#1 in 2011)

Source: http://www.telegraph.co.uk/

- LINPACK reported 8.162 PFLOPS
- Energy consumption: 9.89 MW == 10 K houses

Is Energy Consumption the Issue? (I)

K supercomputer at the RIKEN AICS (#1 in 2011)

Source: http://www.telegraph.co.uk/

- LINPACK reported 8.162 PFLOPS
- Energy consumption: 9.89 MW == 10 K houses
- Annual support cost: $\$10$ m

Is Energy Consumption the Issue? (II)

Titan - Cray XK7 at Oak Ridge National Lab (#1 in 2012)

Source: http://en.wikipedia.org/wiki/Titan_(supercomputer)

- 16-core AMD Opteron 6200 and NVIDIA K20 Tesla

Is Energy Consumption the Issue? (II)

Titan - Cray XK7 at Oak Ridge National Lab (#1 in 2012)

Source: http://en.wikipedia.org/wiki/Titan_(supercomputer)

- 16-core AMD Opteron 6200 and NVIDIA K20 Tesla
- LINPACK reported 17.59 PFLOPS (**+115 %**)

Is Energy Consumption the Issue? (II)

Titan - Cray XK7 at Oak Ridge National Lab (#1 in 2012)

Source: `http://en.wikipedia.org/wiki/Titan_(supercomputer)`

- 16-core AMD Opteron 6200 and NVIDIA K20 Tesla
- LINPACK reported 17.59 PFLOPS (**+115 %**)
- Energy consumption: 8.2 MW (**-17 %**)

Is Energy Consumption the Issue? (II)

Titan - Cray XK7 at Oak Ridge National Lab (#1 in 2012)

Source: http://en.wikipedia.org/wiki/Titan_(supercomputer)

- 16-core AMD Opteron 6200 and NVIDIA K20 Tesla
- LINPACK reported 17.59 PFLOPS (**+115 %**)
- Energy consumption: 8.2 MW (**-17 %**)
- Annual support cost: $\$9$ m (**-10 %**)

Is Energy Consumption the Issue? (II)

Titan - Cray XK7 at Oak Ridge National Lab (#1 in 2012)

Source: http://en.wikipedia.org/wiki/Titan_(supercomputer)

- 16-core AMD Opteron 6200 and NVIDIA K20 Tesla
- LINPACK reported 17.59 PFLOPS (**+115 %**)
- Energy consumption: 8.2 MW (**-17 %**)
- Annual support cost: \$$9$ m (**-10 %**)
- Annual support cost of Titan without GPUs – \$$30$ m!

Use **alternatives** to the standard HPC architectures

Accelerators

- NVIDIA GPUs

Alternative Solutions

Use **alternatives** to the standard HPC architectures:

Accelerators

- NVIDIA GPUs

Low-power CPUs

- ARM CPUs

Sources: `www.green500.org`, `www.top500.org`, `www.montblanc-project.eu`

New Alternatives – New Challenges

- **Programmability.** Efficiently combining different programming models is not an easy task
- **Portability.** Adjusting applications to the new HPC systems may require major changes to applications
- **Efficiency.** Some HPC applications run better on standard HPC architectures

New Alternatives – New Challenges

- **Programmability.** Efficiently combining different programming models is not an easy task
- **Portability.** Adjusting applications to the new HPC systems may require major changes to applications
- **Efficiency.** Some HPC applications run better on standard HPC architectures

Solution: Thorough **performance analysis** of applications

Performance Analysis

- **Understanding** the application behavior is **necessary** to debug and optimize it
- Two approaches
  - Debug at the code level (gdb, Valgrind, Totalview)
  - Debug at the algorithm level (traces, gprof)

Performance Analysis

- **Understanding** the application behavior is **necessary** to debug and optimize it
- Two approaches
  - Debug at the code level (gdb, Valgrind, Totalview)
  - Debug at the algorithm level (traces, gprof)
- **Performance tracing** means to gather timestamped events that occur during the application execution
- Although traces may differ, the **pattern** remains the **same**

Performance Analysis

- **Understanding** the application behavior is **necessary** to debug and optimize it
- Two approaches
  - Debug at the code level (gdb, Valgrind, Totalview)
  - Debug at the algorithm level (traces, gprof)
- **Performance tracing** means to gather timestamped events that occur during the application execution
- Although traces may differ, the **pattern** remains the **same**

We aim at supporting **performance tracing** on both *GPU accelerators* and *low-power ARM CPUs*

Profiling Issues

- Profiling may require **source code** instrumentation
- That requires to **recompile** the application

An EZTrace Framework

Profiling Issues

- Profiling may require **source code** instrumentation
- That requires to **recompile** the application

EZTrace Features

- **No recompilation**
- Plugin-based – allows to profile custom functions

An EZTrace Framework

Profiling Issues

- Profiling may require **source code** instrumentation
- That requires to **recompile** the application

EZTrace Features

- **No recompilation**
- Plugin-based – allows to profile custom functions
- Performance tracing in two steps
  - **Trace collection**: Record raw events
  - **Post-mortem** trace **analysis**: Interpret events

EZTrace: Trace Collection

- Load appropriate plugins:

```
export EZTRACE_TRACE="mpi omp pthread memory stdio"
```

EZTrace: Trace Collection

- Load appropriate plugins:

```
export EZTRACE_TRACE="mpi omp pthread memory stdio"
```

- Execute the application with EZTrace

```
eztrace ./app param1 param2
mpiexec -np N eztrace ./mpiapp param0 param1
```

EZTrace: Trace Collection

- Load appropriate plugins:

```
export EZTRACE_TRACE="mpi omp pthread memory stdio"
```

- Execute the application with EZTrace

```
eztrace ./app param1 param2
mpiexec -np N eztrace ./mpiapp param0 param1
```

- Intercept functions of interest
  - Functions from **dynamic** libs are instrumented using LD_PRELOAD
  - Functions from **static** libs are instrumented by modifying the program in memory

EZTrace: Trace Collection

- Load appropriate plugins:

```
export EZTRACE_TRACE="mpi omp pthread memory stdio"
```

- Execute the application with EZTrace

```
eztrace ./app param1 param2
mpiexec -np N eztrace ./mpiapp param0 param1
```

- Intercept functions of interest
  - Functions from **dynamic** libs are instrumented using LD_PRELOAD
  - Functions from **static** libs are instrumented by modifying the program in memory
- Generate raw traces: one per process

EZTrace: Post-Mortem Analysis

- Add semantic to the trace file
  - For example, an event with #90010 corresponds to `MPI_START_SENDRECV`

EZTrace: Post-Mortem Analysis

- Add semantic to the trace file
  - For example, an event with #90010 corresponds to `MPI_START_SENDRECV`
- Load plugins possibly different from the recording phase

EZTrace: Post-Mortem Analysis

- Add semantic to the trace file
  - For example, an event with #90010 corresponds to `MPI_START_SENDRECV`
- Load plugins possibly different from the recording phase
- Interpret events
  - Compute statistics

```
eztrace_stats /tmp/<username>_eztrace_log*
```

  - Generate traces in various formats like Pajé and OTF

```
eztrace_convert -t PAJE -o paje.trace
    /tmp/<username>_eztrace_log*
```

EZTrace: Post-Mortem Analysis

- Add semantic to the trace file
  - For example, an event with #90010 corresponds to `MPI_START_SENDRECV`
- Load plugins possibly different from the recording phase
- Interpret events
  - Compute statistics

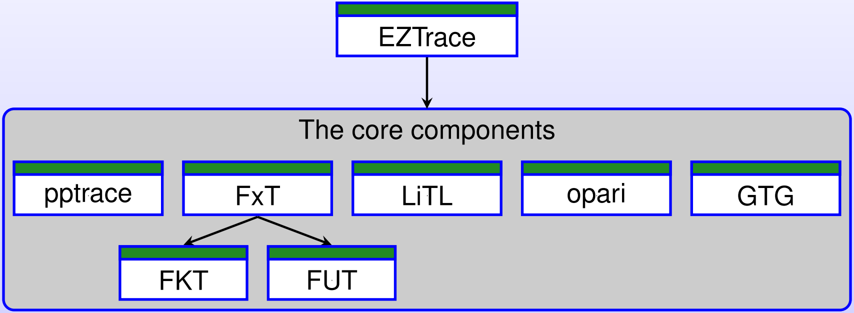
```
eztrace_stats /tmp/<username>_eztrace_log*
```

  - Generate traces in various formats like Pajé and OTF

```
eztrace_convert -t PAJE -o paje.trace
    /tmp/<username>_eztrace_log*
```

- Post-mortem analysis does **not impact** on the application execution

EZTrace: Structure



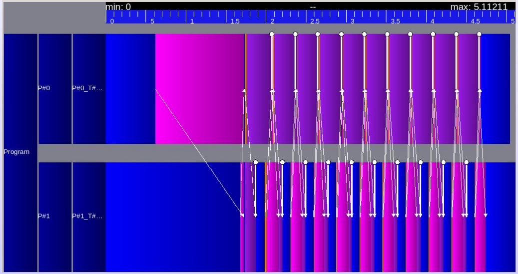
EZTrace: An Example

```
export EZTRACE_TRACE="mpi"
mpiexec -np 2 eztrace ./mpi_ring
```

EZTrace: An Example

```
export EZTRACE_TRACE="mpi"
mpiexec -np 2 eztrace ./mpi_ring
```

```
eztrace_convert -t PAJE -o paje.trace
    /tmp/iakymchuk__eztrace_log_*
```

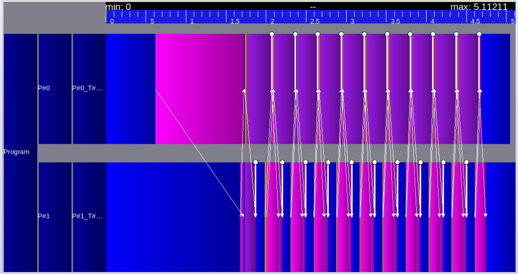
ViTE: 

EZTrace: An Example

```
export EZTRACE_TRACE="mpi"
mpiexec -np 2 eztrace ./mpi_ring
```

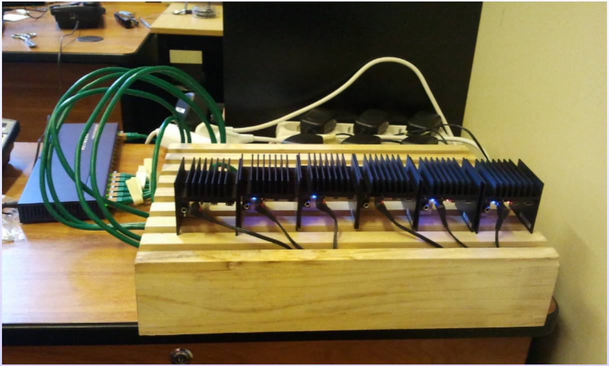
```
eztrace_convert -t PAJE -o paje.trace
    /tmp/iakymchuk__eztrace_log_*
```

ViTE:




```
eztrace_stats /tmp/iakymchuk_eztrace_log_*
  MPI_SEND               :20 calls
  MPI_RECV               :20 calls
  MPI_BARRIER            :22 calls
  Time spent sending (ns):  min: 7.717  max: 22.623
       average: 10.186  total: 101.863
```

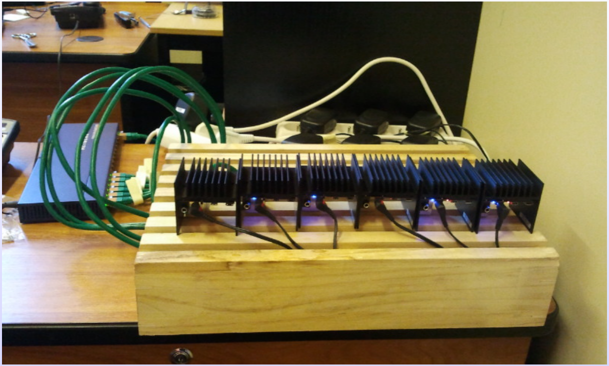
Experimental Setup



Source: http://www-public.it-sudparis.eu/~trahay_f/odroid_supercomputer/

Figure: Odroid-U2 cluster equipped with six quad-core ARM Cortex-A9 @1.7GHz

Experimental Setup



Source: http://www-public.it-sudparis.eu/~trahay_f/odroid_supercomputer/

Figure: Odroid-U2 cluster equipped with six quad-core ARM Cortex-A9 @1.7GHz

Energy consumption per node:

- Idle: 2 W
- Computing: 7 W

NAS Parallel Benchmarks (NPB)

Kernel	#Events	NPB (secs)	EZTrace (secs)	Overhead (%)
BT	3,645,137	295.97	297.74	0.60
MG	632,425	20.83	21.12	1.39
LU	8,003,726	324.27	310.41	−4.27
FT	12,678,743	283.01	284.00	0.35
IS	1,601,371	28.80	28.87	0.24
SP	4,438,161	483.02	484.10	0.22
CG	7,675,754	529.75	529.26	−0.09
EP	135,138	19.98	19.85	−0.65

Table: The results of running NAS Parallel Benchmarks on the 4-node of the odroid cluster. Settings: CLASS=B; NBPROCS=16

NAS Parallel Benchmarks (NPB)

Kernel	#Events	NPB (secs)	EZTrace (secs)	Overhead (%)
BT	3,645,137	295.97	297.74	0.60
MG	632,425	20.83	21.12	1.39
LU	8,003,726	324.27	310.41	−4.27
FT	12,678,743	283.01	284.00	0.35
IS	1,601,371	28.80	28.87	0.24
SP	4,438,161	483.02	484.10	0.22
CG	7,675,754	529.75	529.26	−0.09
EP	135,138	19.98	19.85	−0.65

Table: The results of running NAS Parallel Benchmarks on the 4-node of the odroid cluster. Settings: CLASS=B; NBPROCS=16

Execution	`min` (secs)	`max` (secs)
Raw LU	305.65	326.34
With EZTrace	304.63	327.77

Table: Variation in the time measurements

TELECOM SudParis

Conclusions and Future Work

Conclusions

- **EZTrace** – an open-source framework for performance analysis
- Predefined **plugins**: MPI, GNU OpenMP, Pthread synchronization functions, PLASMA, **CUDA**

Conclusions and Future Work

Conclusions

- **EZTrace** – an open-source framework for performance analysis
- Predefined **plugins**: MPI, GNU OpenMP, Pthread synchronization functions, PLASMA, **CUDA**
- **Customizable** with plugins
- Supports both the standard X86 and low-power **ARM CPUs**

Conclusions and Future Work

Conclusions

- **EZTrace** – an open-source framework for performance analysis
- Predefined **plugins**: MPI, GNU OpenMP, Pthread synchronization functions, PLASMA, **CUDA**
- **Customizable** with plugins
- Supports both the standard X86 and low-power **ARM CPUs**
- **Low** performance **overhead**

Conclusions and Future Work

Conclusions

- **EZTrace** – an open-source framework for performance analysis
- Predefined **plugins**: MPI, GNU OpenMP, Pthread synchronization functions, PLASMA, **CUDA**
- **Customizable** with plugins
- Supports both the standard X86 and low-power **ARM CPUs**
- **Low** performance **overhead**

Future Work

- **Enhance** the performance of the CUDA module
- Develop an **OpenCL** module
- Port EZTrace to Intel Phi and Kalray **MPPAs**

References

- **EZTrace: A Generic Framework for Performance Analysis**, François Trahay, François Rue, Mathieu Faverge, Yutaka Ishikawa, Raymond Namyst, and Jack Dongarra. In the Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). Newport Beach, CA, USA, May, 2011.
- **Runtime Function Instrumentation with EZTrace**, Charles Aulagnon, Damien Martin-Guillerez, François Rue, and François Trahay. In the Proceedings of the PROPER – 5th Workshop on Productivity and Performance. Rhodes, Greece, August, 2012.
- **An Open-Source Tool-Chain for Performance Analysis**, Kevin Coulomb, Augustin Degomme, Mathieu Faverge, and François Trahay. Tools for High Performance Computing, pp. 37–48, 2012.
- **LiTL: Lightweight Trace Library**, Roman Iakymchuk and François Trahay. The 4th Parallel Software Tools and Tool Infrastructures (PSTI) workshop at the 42nd International Conference on Parallel Processing (ICPP). Lyon, France, October 2nd, 2013. (submitted)

Questions?

roman.iakymchuk@telecom-sudparis.eu

www.aices.rwth-aachen.de:8080/~iakymchuk