JÜLICH
FORSCHUNGSZENTRUM

Optimisation and Benchmarking

Part 1 – LWM2, a useful tool

28. Nov 2014|

Alan O'Cais
a.ocais@fz-juelich.de

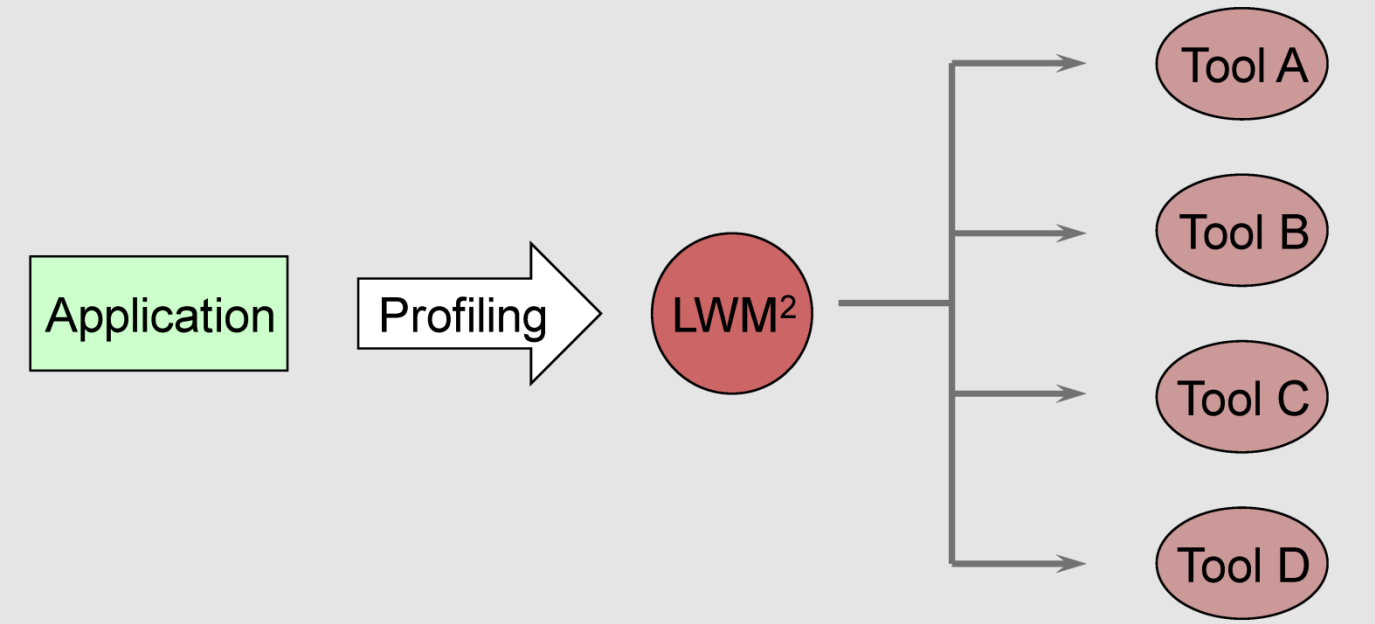
Live notes:
http://supercomputing.cyi.ac.cy/index.php/live

LWM² Introduction

- Light-Weight Measurement Module
  - Light-weight profiler
  - Low learning curve for usage
    - No recompilation / relinking
    - Simple and useful performance information
  - Light usage of resources
    - Low overhead during profiling
  - Geared towards cluster-based systems
  - **Does not enable detailed performance analysis**

Springboard for Tools

- LWM2 as springboard for performance tools
  - Low usage-curve for profiling
  - Simple output provides a guidance to use performance tool



Sample Output

- Time spent in various sections

Time spent:	Average	Minimum	Maximum
Time spent in MPI:	86.24%	81.90%	91.38%
Time spent in MPI P2P:	9.46%	0.85%	20.69%
Time spent in MPI Coll:	75.70%	60.34%	87.07%
Time spent in MPI I/O:	0.00%	0.00%	0.00%

- Multithreading Performance

Multithreading performance:	Average	Minimum	Maximum
OMP effective threads:	1.98	1.98	1.98
Max. thread count:	2	2	2

- Hardware counters

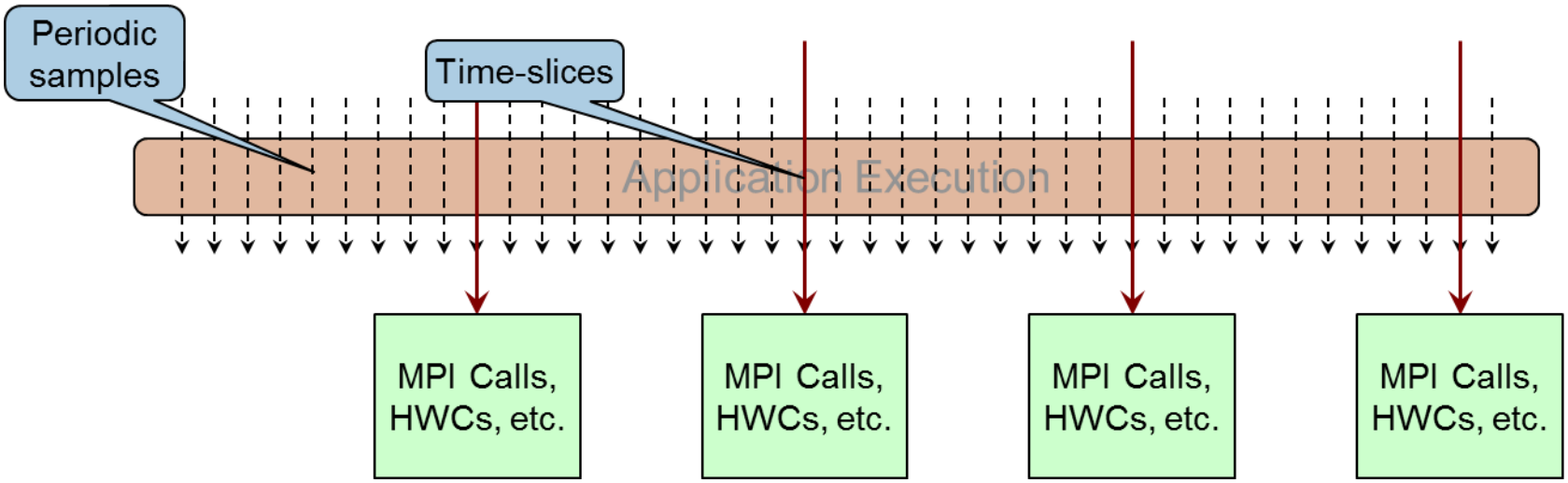
Sequential performance:	Average	Minimum	Maximum
Load/Store hit ratio:	98.34%	98.31%	98.37%

Usage

- No recompilation / relinking required when using LWM^2
- Setting proper environment variables allows profiling with LWM^2
- For MPI and hybrid applications:
  - mpiexec –x –E LD_PRELOAD=<LWM2_library> …
  - The format of passing on the value to LD_PRELOAD may change for different MPI implementations
- For non-MPI applications
  - LD_PRELOAD=< LWM^2 _library> <executable>

What does LWM^2 output?

- Aggregation of data for
  - Every time slice
  - Whole execution (execution digest)



- Can read time slice information with *l2freader* utility
- We will focus on summary information (outputted to console)

Console Summary

- Summary divided into many parts
- Output changes for the type of application profiled
- First part provides overview of the application
- Includes, besides others
  - Job id
  - Wall clock time
  - Number of processes

```
*************************************
                          Job Digest
*************************************
                Job id:   1060
  Wall clock time [s]:    128.72
     Nr. of Processes:    4
```

- Rest of the sections contain profiling metrics
  - For MPI applications, metrics are presented with
    - Average, minimum and maximum values across processes

MPI Communication

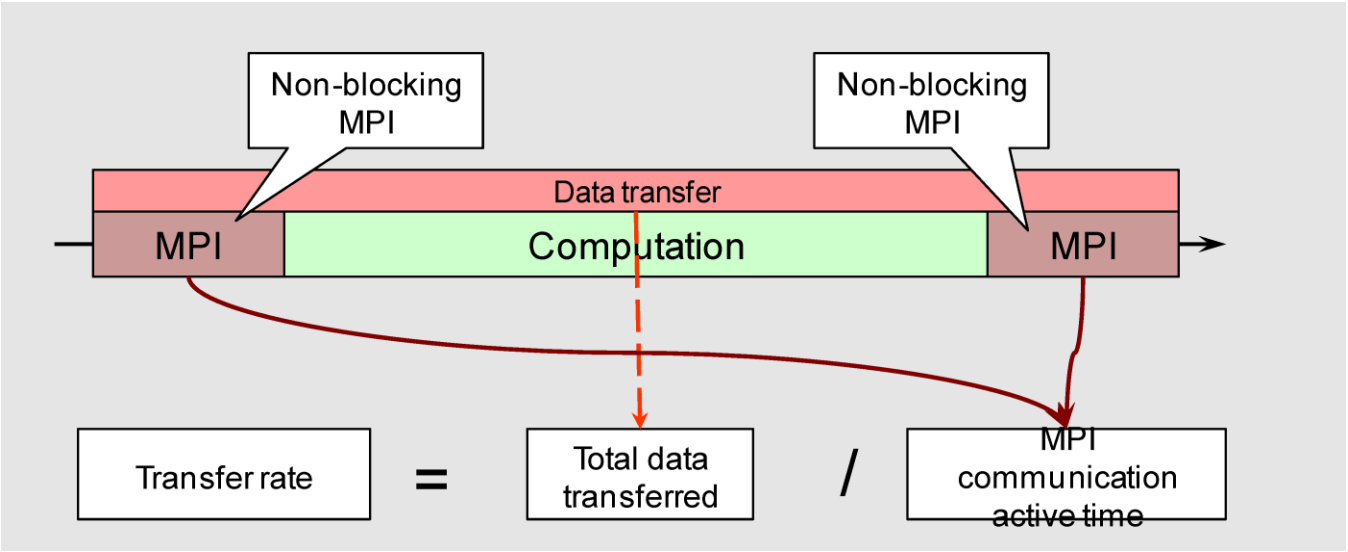
- Basic metrics about MPI communication

MPI Communication:	Average	Minimum	Maximum
Size of P2P messages [Bytes]:	1000000.00	1000000	1000000
Size of collective messages sent [Bytes]:	175001.60	0	1000000
Size of collective messages recv [Bytes]:	175002.20	0	7000000
P2P message frequency [/s]:	79.37	79.33	79.38
Collective invocation frequency [/s]:	396.84	396.67	396.90
P2P bytes transfer rate [/s]:	2542635658.91	1933962264.15	3360655737.70
Coll bytes transfer rate [/s]:	558480474.80	300151215.23	2204313671.27

- Frequency is calculated using both active and inactive application execution time

MPI Communication

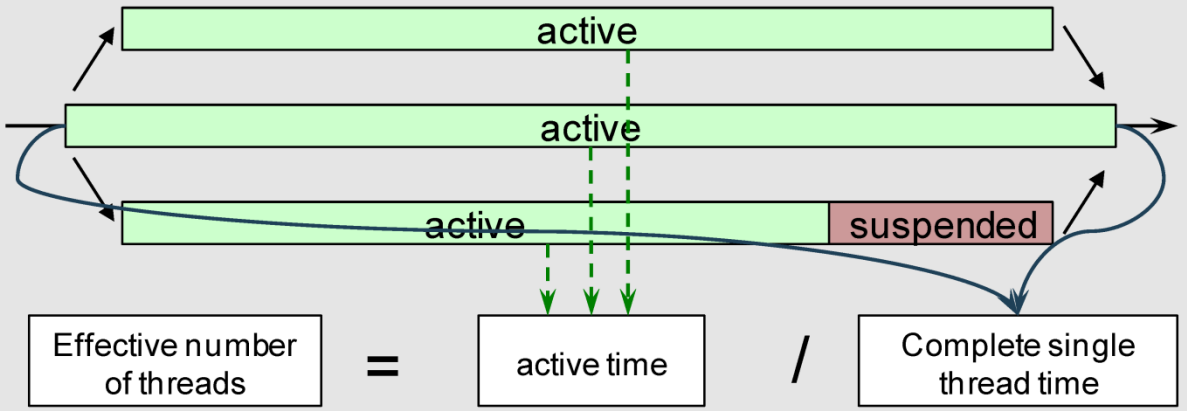
- Transfer rate values consider the active time in communication calls
  - Can result in high transfer rates for non-blocking communication
  - Indicates good overlap between communication and computation



Multithreading Performance

- The number of threads set for the application
- The effective number of threads active during execution

Multithreading performance:	Average	Minimum	Maximum
OMP effective threads:	1.98	1.98	1.98
Max. thread count:	2	2	2



Interpreting Output: Summary

Metrics	Performance tools
L1 hit ratio	ThreadSpotter, Paraver
Effective thread count	ThreadSpotter, Vampir, Paraver
Time based metric - MPI	Scalasca, Vampir, Paraver, Dimemas
Time based metric - CUDA	Vampir, Paraver
Low transfer rates in MPI	Scalasca, Paraver, Vampir, Dimemas

What will we use it for?

- Benchmarking requires
  - Performing timing measurements
  - Testing scalability
- We will look at
  - OpenMP performance
  - MPI performance
  - IO performance
- LWM^2 will give us access to (and record) information in all of these areas with low effort on our part

Special Thanks: Aamer Shah, GRS Aachen

- All slide content from one of his presentations
- See full presentation at:
  - http://www.vi-hps.org/upload/projects/hopsa/hopsa-nov12-lwm2.pdf
- Reference:

  A. Shah, F. Wolf, S. Zhumatiy, and V. Voevodin.

  Capturing inter-application interference on clusters.

  In 2013 IEEE International Conference on Cluster Computing (CLUSTER)