German Research School
for Simulation Sciences

Light-Weight Measurement Module

Introduction and Usage

Aamer Shah
GRS Aachen

JÜLICH
FORSCHUNGSZENTRUM

RWTH AACHEN
UNIVERSITY

Agenda

- Introduction
- LWM^2 Usage
- Profiling Output
  - Time-based
  - Multithreading
  - MPI
  - Interpreting Output
- Running on GraphIT

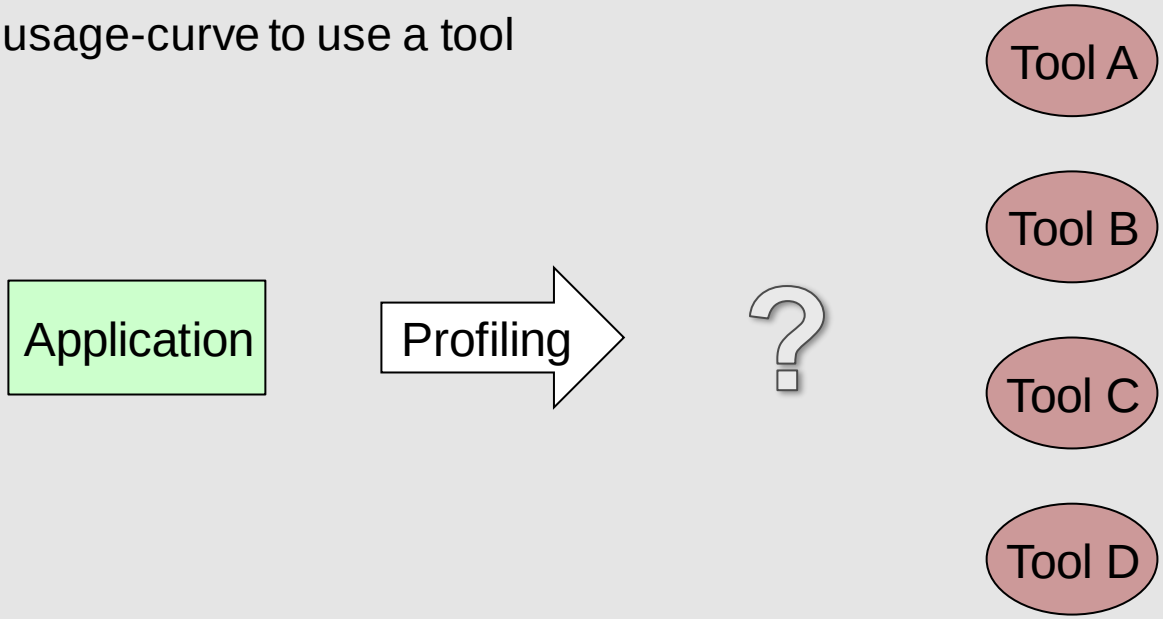
Introduction

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
  - Does not enable detailed performance analysis

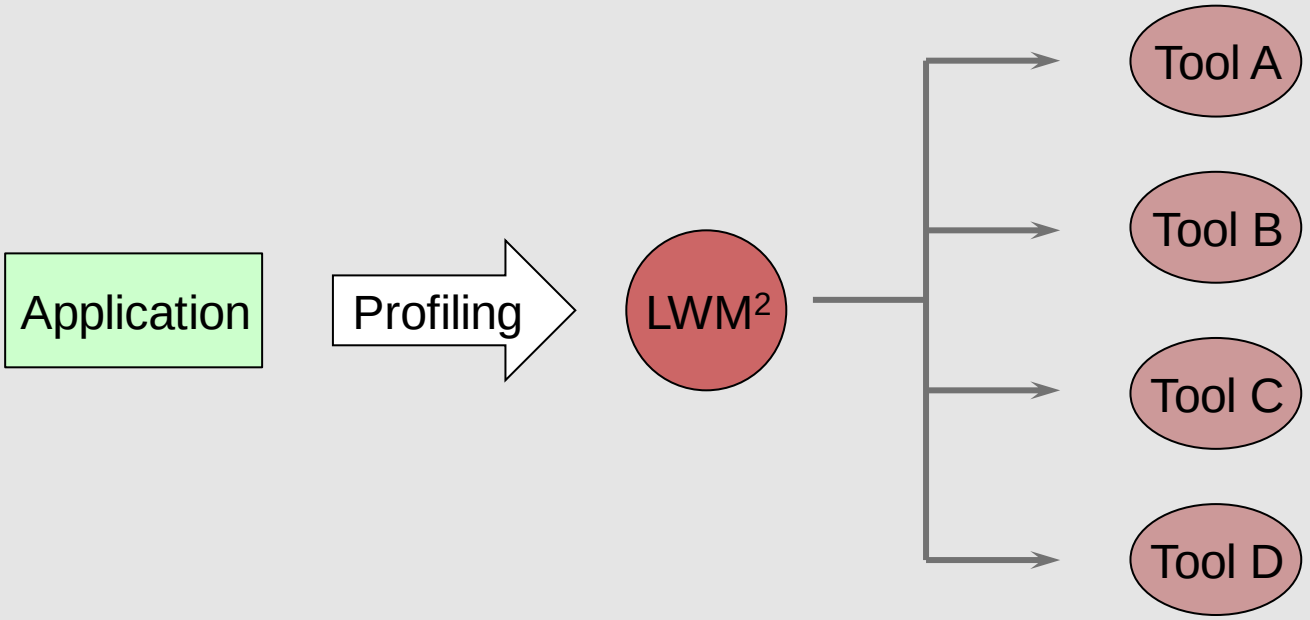
Role of LWM²: Performance Screening

- Which tool to use to profile application?
  - Every tool has its own strength
  - High usage-curve to use a tool

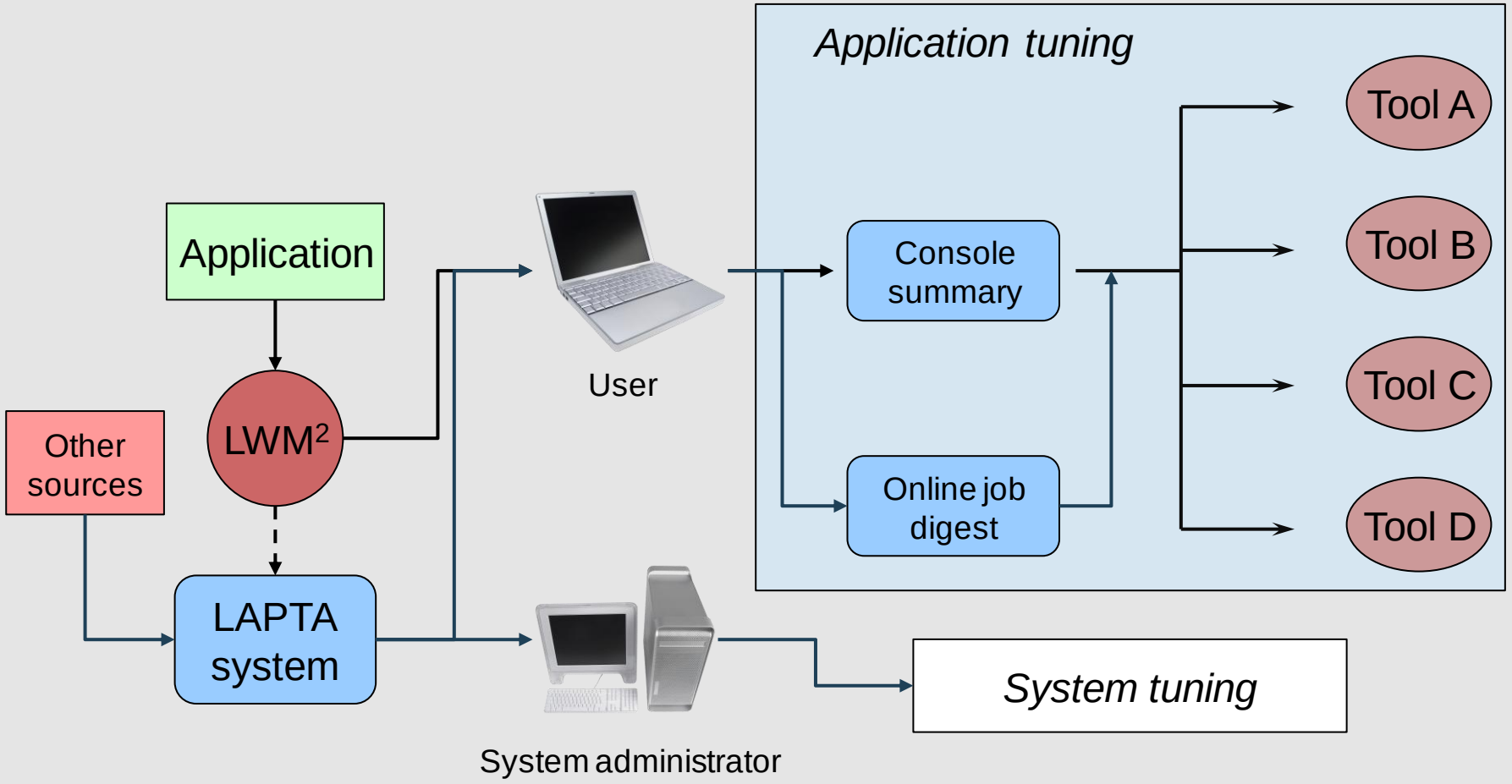


Springboard for Tools

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



HOPSA Tools Workflow



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%

LWM^2 Usage

LWM² Usage

- No recompilation / relinking required when using LWM²
- Setting proper environment variables allow profiling with LWM²
- 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=<LWM²_library> <executable>

LWM^2 Settings

- Some environment variables have to be set for proper LWM^2 profiling

- LWM2_CONSOLE_SUMMARY
  - Provide summary of profiling information on the console
  - VERBOSE provides additional information in summary

- LWM2_WRITE_FILE
  - NO means no output files are generated
  - TIME_SLICE generates multiple files with detailed profiling information
  - Any other value will generate output files with basic profiling information

LWM² Settings

- LWM2_HWC_CONFIG
  - Different settings to display different hardware counter values in the system
  - For GraphIT:
    i. LWM2_HWC_CPU_ONLY: Provides information about CPU related counters only
    ii. LWM2_HWC_MEM_ONLY: Provides information about Cache performance only
- Other variables
  - LWM2_IGNORE_LIST
  - LWM2_OUTPUT_DIR
  - LWM2_OUTPUT_FILENAME
  - LWM2_JOBID_VAR

Profiling Output

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

Time-based metrics

- Time spent in various parts of the application

Time spent:	Average	Minimum	Maximum
Time spent in MPI:	93.35%	92.23%	95.63%
Time spent in MPI P2P:	5.51%	0.00%	13.66%
Time spent in MPI Coll:	87.32%	78.54%	93.69%
Time spent in MPI I/O:	0.00%	0.00%	0.00%

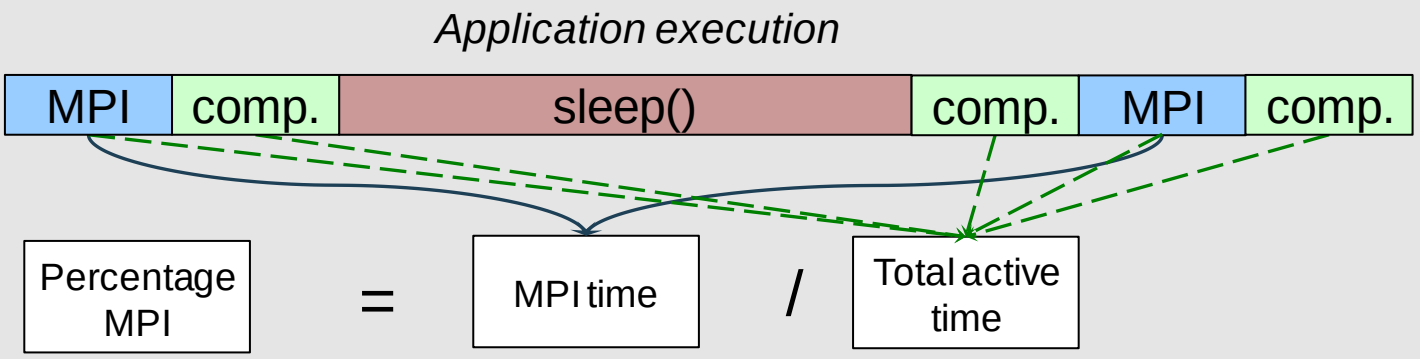
- Times for MPI subtypes are shown as a percentage of whole execution
  - Average values of subtypes should almost add up to the whole MPI time-spent value
  - Remaining time spent in other areas of MPI
    - MPI_Init, MPI_Finalize, etc

Time-based metrics

- Values for CUDA applications are also subdivided
  - Non-MPI application, so only one value

```
                    Time spent in CUDA:                 5.00%
Time spent in CUDA memcpy to device:                    0.00%
  Time spent in CUDA memcpy to host:                    2.50%
-------------------------------------------------------------
```

- Value represents the percentage time spent when the application was active



MPI Communication

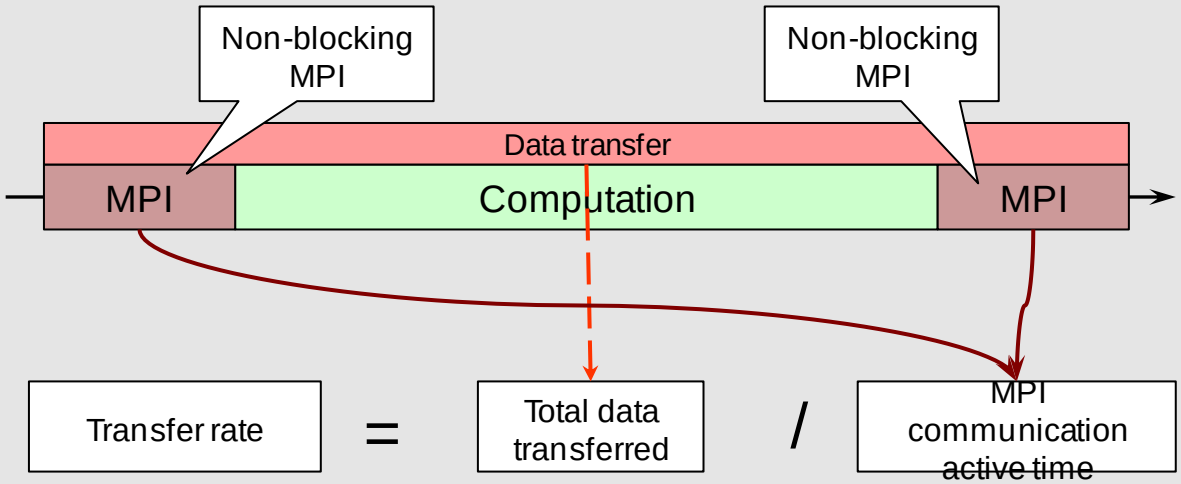
- Basic metrics about MPI communication

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



CUDA Memory Transfer

- CUDA memory transfer details

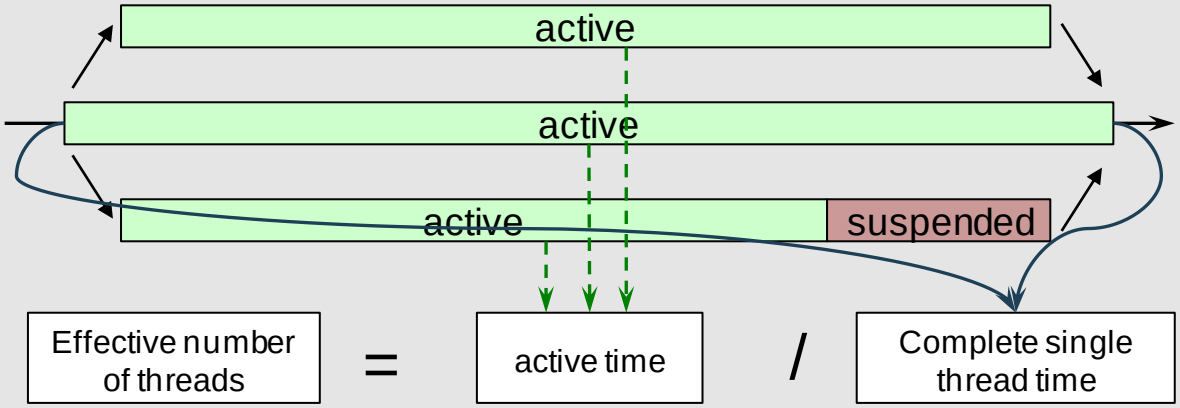
CUDA:	
Data transfered to device [Bytes]:	1228800
Data transfered to host [Bytes]:	819200
Data transfered to device [per msg]:	614400.00
Data transfered to host [per msg]:	819200.00
Frequency of Memcpy calls to device [/sec]:	0.40
Frequency of Memcpy calls to host [/sec]:	0.20

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

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



Hardware Counters

- Two configurations
  - LWM2_HWC_CONFIG=LWM2_HWC_CPU_ONLY

Sequential performance:	Average	Minimum	Maximum
CPI:	0.68	0.60	0.76
FLOPS:	4293439.50	1666735.00	6320490.00
FP Instructions:	25.46%	13.80%	32.72%

  - LWM2_HWC_CONFIG=LWM2_HWC_MEM_ONLY

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

Interpreting Output

Interpreting Output

- Figuring out application performance from output
- No set formula, as expected values change from application to application

- General bad values
  - Low L1 hit ratio
  - Check with ThreadSpotter or Paraver

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

- Low effective thread count
  - Check with ThreadSpotter, Vampir or Paraver

Interpreting Output

- Time-based values depends upon application
  - For time spend in MPI calls
    - Check with Scalasca, Paraver, Vampir or Dimemas
  - For time spend in CUDA calls
    - Check with Vampir or Paraver
- Low transfer rates for MPI
  - For non-blocking, bad overlap of communication and computation
  - For blocking, bad topology, network congestion, latency, etc
    - Check with Scalasca, Paraver, Dimemas or Vampir

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

Running on GraphIT

Running of GraphIT

- cleo system used to submit batch jobs
- Have to set the variables in a script to properly configure LWM^2
- The executable is called at the end in the script
- cleo-submit –np <processes> <script>
- -np <processes> specifies MPI processes
  - For hybrid applications, total processors allocated are <processes> x OMP_NUM_THREADS

Thank you