Performance Tools on OpenPOWER
The Score-P Tool Ecosystem

Bernd Mohr, Michael Knobloch
Jülich Supercomputing Centre
scalasca

Scalable, Integrated, Versatile, Maintained

PARALLEL PERFORMANCE TOOLS
Vampir

- Interactive event trace analysis
  - Alternative & supplement to automatic trace analysis
  - **Visual presentation of dynamic runtime behaviour**
    - Event timeline chart for states & interactions of processes/threads
    - Communication statistics, summaries & more
  - **Interactive browsing, zooming, selecting**
    - Linked displays & statistics adapt to selected time interval
    - Scalable server runs in parallel to handle larger traces
- Handles OTF (VampirTrace), OTF2 (Score-P), and EPIK (Scalasca V1) trace formats
- Developed by TU Dresden ZIH
  - Vampir Server & GUI have a commercial license

http://www.vampir.eu/
Vampir GUI
Scalasca

- **Scalable Analysis of Large Scale Applications**

- **Approach**
  - **Instrument** C, C++, and Fortran parallel applications
    - Based on Score-P framework
  - **Option 1:** scalable call-path profiling
  - **Option 2:** scalable event trace analysis
    - **Collect** event traces
    - **Process** trace
      - Wait-state analysis
      - Delay and root-cause analysis
      - Critical path analysis
  - **Categorize and rank** results

http://www.scalasca.org/
Scalasca Example: CESM Sea Ice Module
Late Sender Analysis
Scalasca Example: CESM Sea Ice Module
Late Sender Analysis + Application Topology
Scalasca Example: CESM Sea Ice Module
Direct Wait Time Analysis
Scalasca Example: CESM Sea Ice Module
Indirect Wait Time Analysis
Scalasca Example: CESM Sea Ice Module Delay Costs Analysis
SCORE-P TOOL ECOSYSTEM
Integration

• Need integrated tool (environment) for all levels of parallelization
  ▪ Inter-node (MPI, PGAS, SHMEM)
  ▪ Intra-node (OpenMP, multi-threading, multi-tasking)
  ▪ Accelerators (OpenACC, CUDA, OpenCL)

• Integration with performance modeling and prediction

• No tool fits all requirements
  ▪ Interoperability of tools
  ▪ Integration via open interfaces
Score-P Functionality

- Provide typical functionality for HPC performance tools
- **Instrumentation** (various methods)
  - Multi-process paradigms (MPI, SHMEM)
  - Thread-parallel paradigms (OpenMP, POSIX threads)
  - Accelerator-based paradigms (OpenACC, CUDA, OpenCL)
  - In any combination!
- Flexible **measurement** without re-compilation:
  - Basic and advanced **profile** generation
  - Event **trace** recording
  - Online access to profiling data
- Highly scalable I/O functionality
- Support all fundamental concepts of partner’s tools
Non-functional Requirements

- **Portability**: support all major HPC platforms
  - IBM Blue Gene, Cray X*, Fujitsu K/FX10/FX100, Tianhe-1A+2
  - x86, x86_64, PPC, Sparc, ARM clusters (Linux, AIX)

- **Scalability**
  - Petascale, supporting platforms with more than 100K cores

- **Low measurement overhead**
  - Typically less than 5%

- **Robustness and QA**
  - Nightly Builds, Continuous Integration Testing Framework

- Easy and uniform installation through **EasyBuild**
- Open Source: New BSD License
Score-P Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH, Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA
The Score-P Tool Ecosystem

Periscope

Online interface

Score-P

Instrumented target application

PAPI

TAU ParaProf

CUBE

Remote Guidance

Vampir

OTF2 traces

Scalasca

wait-state analysis

CUBE4 report

TAU PerfExplorer

CUBE4 report

TAU 2015 JSC 18

JULICH FORSCHUNGSZENTRUM
Connect to Vampir Trace Browser

To investigate most severe pattern instances, connect to a trace browser...

...and select trace file from the experiment directory
Show most severe pattern Instances

Select “Max severity in trace browser” from context menu of call paths marked with a red frame.
Investigate most severe Instance in Vampir

Vampir will automatically zoom to the worst instance in multiple steps (i.e., undo zoom provides more context)
Integration of Measurement and Modelling

- Example: DFG SPPEXA Catwalk Project

```c
main() {
    foo()
    bar()
    compute()
}
```

**Performance measurements (profiles)**

- All functions

```
p_1 = 128
p_2 = 256
p_3 = 512
p_4 = 1,024
p_5 = 2,048
p_6 = 4,096
```

**Instrumentation**

**Automated modeling**

**Rank | Function | Model [s]**
--- | --- | ---
1 | bar() | $4.0 \times p + 0.1 \times \log(p)$
2 | compute() | $0.5 \times \log(p)$
3 | foo() | 65.7
Catwalk: Result Visualization
USE CASES
Performance Tool Scaling: Scalasca

• Latest test case
  ▪ Granular Dynamics Simulation
  ▪ Based on Physics Engine (PE) Framework (Erlangen)
  ▪ PRACE @ ISC Award winner
  ▪ MPI only

• Scalasca 1.x Experiments on JUQUEEN
  ▪ Full machine experiment: 28,672 nodes x 32 MPI ranks
    ▪ 917,504 processes [Limit: Memory / System metadata]
    ▪ Largest no. of threads: 20,480 nodes x 64 MPI ranks
      ▪ 1,310,720 processes [Limit: Memory / System metadata]

• Scalasca 2.x / Score-P 1.4.1 NAS BT-MZ on JUQUEEN
  ▪ Profiles: 16,384 x 64 = 1,048,576 threads [Limit: BT-MZ]
  ▪ Traces: 10,240 x 64 = 655,360 thread [Limit: OTF2]
Scalasca: 1,310,720 Process Test Case
Success Story: TerrSysMP

• Scale-consistent highly modular integrated multi-physics sub-surface/surface hydrology-vegetation atmosphere modelling system

• Fully-coupled MPMD simulation consisting of
  - COSMO (Weather prediction)
  - CLM (Community Land Model)
  - ParFlow (Parallel Watershed Flow)
  - OASIS coupler
Success Story: TerrSysMP

- Identified several sub-components bottlenecks:
  - Inefficient communication patterns
  - Unnecessary/inefficient code blocks
  - Inefficient data structures
- Performance of sub-components improved by factor of 2!
- Scaling improved from 512 to 32768 cores!
SCORE-P ON POWER8
CUDA AND CPI STACK ANALYSIS
Porting Score-P to POWER8

• Surprisingly smooth
  ▪ Only update of config.sub and config.guess necessary

• Support for XL, GNU and Clang compilers
  ▪ No chance to test PGI compilers yet

• Support for GPU via CUDA and OpenCL
  ▪ OpenACC support coming soon

• Support for hardware counters via PAPI
  ▪ >1100 different counters

• Scalasca trace analysis tools also ported to POWER8
Measurement of CUDA events

• Record CUDA events using the CUPTI interface

```
% export SCOREP_CUDA_ENABLE=gpu,kernel,idle
```

• Important record types:
  - runtime: CUDA runtime API
  - driver: CUDA driver API
  - gpu: GPU activities
  - kernel: CUDA kernels
  - Idle: GPU compute idle time
  - memcpy: CUDA memory copy

• For all record types consult the Score-P user guide
POWER8 CPI Stack Analysis

• Cycles-per-instruction (CPI) gives a good evaluation of applications performance on a processing architecture
• POWER8 features a Completion-based CPI Stack accounting model
• Provides information to narrow down bottlenecks in the processor pipeline
• Helps to identify code regions with tuning potential
• Provides hints on what to optimize

• Requires measurement and analysis of >40 hardware counters
## POWER8 CPI Stack

### Completion Stalls

<table>
<thead>
<tr>
<th>Cycles</th>
<th>Stall due to BR or CR</th>
<th>Stall due to Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stall due to CR</td>
<td>Stall due to CR</td>
</tr>
<tr>
<td></td>
<td>Stall due to Fixed-Point</td>
<td>Stall due to Fixed-Point Long</td>
</tr>
<tr>
<td></td>
<td>Stall due to Fixed-Point (Other)</td>
<td>Stall due to Fixed-Point (Other)</td>
</tr>
<tr>
<td></td>
<td>Stall due to Vector</td>
<td>Stall due to Vector Long</td>
</tr>
<tr>
<td></td>
<td>Stall due to Vector (other)</td>
<td>Stall due to Vector (other)</td>
</tr>
<tr>
<td></td>
<td>Stall due to Scalar</td>
<td>Stall due to Scalar Long</td>
</tr>
<tr>
<td></td>
<td>Stall due to Scalar (other)</td>
<td>Stall due to Scalar (other)</td>
</tr>
<tr>
<td></td>
<td>Stall due to LSU</td>
<td>Stall due to LSU Reject</td>
</tr>
<tr>
<td></td>
<td>Stall due to Store Finish</td>
<td>Stall due to Store Finish</td>
</tr>
<tr>
<td></td>
<td>Stall due to Load Finish</td>
<td>Stall due to Load Finish</td>
</tr>
<tr>
<td></td>
<td>Stall due to Store Forward</td>
<td>Stall due to Store Forward</td>
</tr>
<tr>
<td></td>
<td>Stall due to Load/Store (other)</td>
<td>Stall due to Load/Store (other)</td>
</tr>
</tbody>
</table>

### Waiting to Complete

- Stall due to Next-to-Complete Flush

### Thread Blocked

- Blocked due to LWSync
- Blocked due to HWSync
- Blocked due to ECC Delay
- Blocked due to Flush
- Blocked due to COQ Full
- Thread Blocked (other)

### Completion Table Empty due to IC Miss

- Completion Table Empty due to IC L3 Miss
- Completion Table Empty due to IC Miss (other)

### Completion Table Empty due to Branch Mispredict

- Completion Table Empty due to Branch Mispredict + IC Miss

### Completion Table Empty

- Completion Table Empty – Dispatch Held
  - Dispatch Held due to Mapper
  - Dispatch Held due to Store Queue
  - Dispatch Held due to Issue Queue

- Completion Table Empty (Other)

### Completion Cycles
# POWER8 CPI Stack – LSU Details

<table>
<thead>
<tr>
<th>Stall due to Load/Store</th>
<th>Stall due to L2/L3 Hit</th>
<th>Stall due to L3 Miss</th>
<th>Stall due to L2/L3 Hit with conflict</th>
<th>Stall due to On-chip L2/L3</th>
<th>Stall due to On-chip Memory</th>
<th>Stall due to Off-chip L2/L3</th>
<th>Stall due to Off-chip Memory</th>
<th>Stall due to Off-node Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall due to Dcache Miss</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
<tr>
<td>Stall due to LSU Reject</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
<tr>
<td>Stall due to Store Finish</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
<tr>
<td>Stall due to Load Finish</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
<tr>
<td>Stall due to Store Forward</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
<tr>
<td>Stall due to Load/Store (other)</td>
<td>Stall due to L2/L3 Hit</td>
<td>Stall due to L3 Miss</td>
<td>Stall due to L2/L3 Hit with conflict</td>
<td>Stall due to On-chip L2/L3</td>
<td>Stall due to On-chip Memory</td>
<td>Stall due to Off-chip L2/L3</td>
<td>Stall due to Off-chip Memory</td>
<td>Stall due to Off-node Memory</td>
</tr>
</tbody>
</table>
### POWER8 CPI Stack – Hardware Counter

<table>
<thead>
<tr>
<th>Cycles</th>
<th>Breakdown 1</th>
<th>Breakdown 2</th>
<th>Breakdown 3</th>
<th>Breakdown 4</th>
<th>Breakdown 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_RUN_CYC</td>
<td>PM_CMIUSTALL</td>
<td>PM_CMIUSTALL_BRU_CRU</td>
<td>PM_CMIUSTALL_BRU_CRU</td>
<td>PM_CMIUSTALL_BRU_CRU</td>
<td>PM_CMIUSTALL_BRU_CRU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_CRU</td>
<td></td>
<td>PM_CMIUSTALL_CRU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_FXU_LONG</td>
<td></td>
<td>PM_CMIUSTALL_FXU_LONG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_FXU_OTHER</td>
<td></td>
<td>PM_CMIUSTALL_FXU_OTHER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_VECTOR</td>
<td></td>
<td>PM_CMIUSTALL_VECTOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_VECTOR_OTHER</td>
<td></td>
<td>PM_CMIUSTALL_VECTOR_OTHER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_SCALAR</td>
<td></td>
<td>PM_CMIUSTALL_SCALAR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_SCALAR_OTHER</td>
<td></td>
<td>PM_CMIUSTALL_SCALAR_OTHER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_VSU_OTHER</td>
<td></td>
<td>PM_CMIUSTALL_VSU_OTHER</td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_LSU</td>
<td>PM_CMIUSTALL_DCACHE_MISS</td>
<td>PM_CMIUSTALL_DMISS_L2L3</td>
<td>PM_CMIUSTALL_DMISS_L2L3_CONFLICT</td>
<td>PM_CMIUSTALL_DMISS_L2L3_NO_CONFLICT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_DMISS_L3MISS</td>
<td></td>
<td>PM_CMIUSTALL_DMISS_LMEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_DMISS_L1I3</td>
<td></td>
<td>PM_CMIUSTALL_DMISS_L1L3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_DMISS_REMOTE</td>
<td></td>
<td>PM_CMIUSTALL_DMISS_REMOTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PM_CMIUSTALL_DMISS_DISTANT</td>
<td></td>
<td>PM_CMIUSTALL_DMISS_DISTANT</td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_STORE</td>
<td>PM_CMIUSTALL_LOAD_FINISH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_ST_FWD</td>
<td>PM_CMIUSTALL_REJECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_LSU_OTHER</td>
<td>PM_CMIUSTALL_REJECT_LHS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PM_CMIUSTALL_NTCG_FLUSH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PM_CMIUSTALL_NTCG_OTHER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_NCTG_ALL_FIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_THRD</td>
<td>PM_CMIUSTALL_LWSYNC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_HWSYNC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_MEM_ECC_DELAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_FLUSH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_COQ_FULL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_CMIUSTALL_BLOCK_OTHER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_CYC</td>
<td>PM_GCT_NOSLOT_IC_MISS</td>
<td>PM_GCT_NOSLOT_IC_L3MISS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_IC_MISS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_IC_L3MISS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_IC_L2L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_IC_MRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_IC_MRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD_MAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD_SRQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GCT_NOSLOT_DISP_HLD_ISSQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_GRP_CMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTHER_CPI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cube – CPI Counter

Selected "PM_CMPLUSTALL"
Cube – CPI Stack Overview
Questions?

• Check out http://www.scalasca.org

• Or contact us at scalasca@fz-juelich.de
Performance Data Visualization with Vampir

Because “Seeing is Believing”

LZR Room E06
Tel. +49 351 - 463 – 39871

Guido Juckeland (guido.juckeland@tu-dresden.de)
Challenges for Performance Data Visualization

- Record and visualize all participating component activity
- Provide Guidance wrt. performance issues
## Score-P/OTF2 as a Common Acquisition Platform

<table>
<thead>
<tr>
<th>Level</th>
<th>Process Level Parallelism</th>
<th>Intra-Node Parallelism</th>
<th>Accelerator Offloading</th>
<th>Inter-Node Parallelism</th>
<th>Inter-connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Sampling</td>
<td>Directive based</td>
<td>native</td>
<td>Directive based</td>
<td>native</td>
</tr>
<tr>
<td>Paradigm</td>
<td>Resource Usage</td>
<td>OpenMP</td>
<td>Pthreads</td>
<td>CUDA</td>
<td>OpenMP Target</td>
</tr>
<tr>
<td>Tool Support</td>
<td>PAPI</td>
<td>CUPTI</td>
<td>OMPT</td>
<td>CUPTI</td>
<td>ACCT</td>
</tr>
<tr>
<td></td>
<td>OPARI</td>
<td>CUPTI</td>
<td>Custom Wrapping</td>
<td>Custom Wrapping</td>
<td>???</td>
</tr>
<tr>
<td></td>
<td>OMPT</td>
<td></td>
<td></td>
<td></td>
<td>PMPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSHMEM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CUPTI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

**Resource Usage**
- OpenMP
- Pthreads
- CUDA
- OpenCL
- OpenACC
- OpenMP Target
- MPI
- Open-SHMEM
- NVlink
- IB

**Inter-connect**
- Inter-Chip
- Inter-Node

**Score-P/OTF2**

Guido Juckeland 42
Program trace of a simple OpenACC reduction (CPU functions are instrumented)

Vampir Trace Visualization

- Region Stack
- Master Timeline
- Function Summary
- Context View
- Legend

Function Summary:
- Process Filter, Accumulated Exclusive Time per Function
  100 us
  0s
  194.37 us (98.8%)
  1.183 us (0.6%)
  1.169 us (0.59%)

Legend:
- OMP_PARALLEL
- OMP_LOOP
- OMP_SYNC
- OMP_API
- Application
- CUDA_API
- OPENACC
- CUDA_KERNEL
- Monitor

Guido Juckeland
Vampir Trace Visualization – ACCT

Instrumented: CPU functions + OpenACC (ACCT)

Diagram showing timeline and function summary with highlighted CPU functions:
- acc_data_construct
- acc_compute_construct
- acc_upload
- acc_wait
- acc_launch_kernel

Guido Juckeland
Advanced Analysis Techniques

- Analysis based on wait states:
  - **Critical path analysis** (based on CUPTI events)
CASITA as a Method for Guiding Programmers

- Score-P
  - Performance Measurement

- Application

- Trace
  - Dependency attributes

- CASITA Critical-Blame Analysis

- Optimization Rating
  #1 codelet_x
  #2 codelet_y
  #3 codelet_z
  #4 ...

- VAMPIR
  - Additional counter data
Blaming the Culprit

![Diagram showing event streams and time]

- `cuStreamSync` and `cuLaunch(A)`
- `Kernel X` and `Kernel A`
- Time line with event stream and kernels

Guido Juckeland
## Optimization Profile

<table>
<thead>
<tr>
<th>Activity Group</th>
<th>Instances</th>
<th>Time</th>
<th>Time on CP</th>
<th>Fraction CP</th>
<th>Fract. Gl.</th>
<th>Blame</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_Init</td>
<td>4</td>
<td>4.756601</td>
<td>1.189140</td>
<td>39.57%</td>
<td>0.00%</td>
<td>0.395750</td>
<td></td>
</tr>
<tr>
<td>Modify::Modify (LAMMPS*)</td>
<td>4</td>
<td>1.141863</td>
<td>0.285544</td>
<td>9.50%</td>
<td>13.35%</td>
<td>0.226491</td>
<td></td>
</tr>
<tr>
<td>AtomVecSphere::pack_border (int, int*, d)</td>
<td>4672</td>
<td>0.029269</td>
<td>0.008739</td>
<td>0.29%</td>
<td>8.61%</td>
<td>0.089040</td>
<td></td>
</tr>
<tr>
<td>AtomVec::ubuf::ubuf (int)</td>
<td>151085</td>
<td>0.027865</td>
<td>0.006450</td>
<td>0.28%</td>
<td>8.36%</td>
<td>0.086404</td>
<td></td>
</tr>
<tr>
<td>Force::Force (LAMMPS*)</td>
<td>4</td>
<td>0.383532</td>
<td>0.095886</td>
<td>3.19%</td>
<td>4.48%</td>
<td>0.076727</td>
<td></td>
</tr>
<tr>
<td>LAMMPS::LAMMPS (int, char**, ompi_commun)</td>
<td>4</td>
<td>0.313883</td>
<td>0.095291</td>
<td>3.17%</td>
<td>4.45%</td>
<td>0.076251</td>
<td></td>
</tr>
<tr>
<td>Input::Input (LAMMPS*, int, c)</td>
<td>4</td>
<td>0.436844</td>
<td>0.093091</td>
<td>3.10%</td>
<td>4.35%</td>
<td>0.074491</td>
<td></td>
</tr>
<tr>
<td>LAMMPS::create()</td>
<td>4</td>
<td>0.319075</td>
<td>0.079797</td>
<td>2.66%</td>
<td>3.73%</td>
<td>0.063853</td>
<td></td>
</tr>
<tr>
<td>AtomVecSphere::unpack_border (int, int,)</td>
<td>4672</td>
<td>0.031297</td>
<td>0.008349</td>
<td>0.28%</td>
<td>5.53%</td>
<td>0.058053</td>
<td></td>
</tr>
<tr>
<td>AtomVec::ubuf::ubuf (double)</td>
<td>151085</td>
<td>0.030222</td>
<td>0.006040</td>
<td>0.27%</td>
<td>5.33%</td>
<td>0.056003</td>
<td></td>
</tr>
<tr>
<td>CiteMe::add (char const*)</td>
<td>4</td>
<td>0.001769</td>
<td>0.001768</td>
<td>0.06%</td>
<td>4.11%</td>
<td>0.011639</td>
<td></td>
</tr>
<tr>
<td>Input::file()</td>
<td>4</td>
<td>0.000496</td>
<td>0.000165</td>
<td>0.01%</td>
<td>3.68%</td>
<td>0.036891</td>
<td></td>
</tr>
<tr>
<td>Input::execute_command()</td>
<td>96</td>
<td>0.396539</td>
<td>0.099180</td>
<td>3.30%</td>
<td>0.14%</td>
<td>0.034393</td>
<td></td>
</tr>
<tr>
<td>Lattice::Lattice (LAMMPS*, in)</td>
<td>8</td>
<td>0.250494</td>
<td>0.062657</td>
<td>2.09%</td>
<td>0.94%</td>
<td>0.030254</td>
<td></td>
</tr>
<tr>
<td>ComputePressure::virial_compute (int, in)</td>
<td>16016</td>
<td>0.012182</td>
<td>0.003137</td>
<td>0.10%</td>
<td>2.53%</td>
<td>0.026387</td>
<td></td>
</tr>
<tr>
<td>Atom::new_avec (char const*, int, int&amp;)</td>
<td>8</td>
<td>0.120604</td>
<td>0.030103</td>
<td>1.00%</td>
<td>1.41%</td>
<td>0.024080</td>
<td></td>
</tr>
<tr>
<td>main</td>
<td>4</td>
<td>0.106750</td>
<td>0.026689</td>
<td>0.89%</td>
<td>1.26%</td>
<td>0.021464</td>
<td></td>
</tr>
<tr>
<td>Thermo::parse_fields (char*)</td>
<td>8</td>
<td>0.106497</td>
<td>0.026605</td>
<td>0.89%</td>
<td>1.24%</td>
<td>0.021287</td>
<td></td>
</tr>
<tr>
<td>ComputeTemp::compute_scalar()</td>
<td>16020</td>
<td>0.014109</td>
<td>0.003634</td>
<td>0.12%</td>
<td>1.74%</td>
<td>0.018618</td>
<td></td>
</tr>
<tr>
<td>MPI_Wait</td>
<td>68002</td>
<td>0.193727</td>
<td>0.048585</td>
<td>1.62%</td>
<td>0.00%</td>
<td>0.016169</td>
<td></td>
</tr>
</tbody>
</table>
Live demo

Guido Juckeland
If you cannot see it,
you will never know it.
you will never understand it.
you will never improve it.