Interactive Performance Analysis with Vampir
UCAR Software Engineering Assembly
in Boulder/CO, 2013-04-03

Andreas Knüpfer, Thomas William
TU Dresden, Germany
Overview

• Introduction
• Vampir displays
• GPGPU support
• Scalability
• Performance Problems
• Conclusions & Preview on Tutorial
Why should I bother with performance analysis?
• Efficient usage of expensive and limited resources
• Scalability to achieve next bigger simulation
• Well, why are you here after all?

How should I do it?
• Optimization phase, just like the testing phase
• Use tools, avoid do-it-yourself-with-printf, really!

What methods are available?
• Profiling
• Event Tracing
Vampir Event Trace Visualization

• Based on recorded event traces
  – Sorted collection of run-time events per process
• Event types with time stamp and process:
  – Enter given function/subroutine
  – Send or receive message of N bytes to/from P
  – Hardware performance counter values
  – More …
• Visualize dynamic run-time behavior
  – Various timeline displays
  – Various statistics displays
• Interactive browsing/zooming/selection
  – Zoom in time and processes
  – Feasible for very large and highly parallel traces
Overview

• Introduction

• Vampir displays

• GPGPU support
• Scalability
• Performance Problems

• Conclusions & Preview on Tutorial
Vampir Display Overview
Process and Counter Timelines

![Timeline Diagram]

Process 0 Timeline:
- Values of Counter "MEM_APP_ALLOC" over Time
  - 50 M
  - 0 M

Process 63 Timeline:
- Values of Counter "ru_utime" over Time
  - 750 k
  - 0 k
Performance Radar
Performance Radar in Overlay Mode
Function Summary

Vampir - [Trace View - /home/dolescha/tracefiles/feature-traces/wrf-p64-io-mem-usage/wrf.1h.ofi]

Function Summary

All Processes, Accumulated Exclusive Time per Function Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>39.18573 s</td>
</tr>
<tr>
<td></td>
<td>33.298935 s</td>
</tr>
<tr>
<td></td>
<td>3.591342 s</td>
</tr>
<tr>
<td></td>
<td>2.295455 s</td>
</tr>
</tbody>
</table>

- MPI
- PHYS
- DYN

VI-HPS
Overview

• Introduction

• Vampir displays

• GPGPU support
• Scalability
• Performance Problems

• Conclusions & Preview on Tutorial
- Support for GPGPU computing with CUDA (OpenCL support in Score-P soon)
- Shows Kernel invocations and idle time (individual DMA transfers shown with Score-P soon)
Also for hybrid MPI+CUDA applications for large scale GPU computing beyond a single compute node
Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial
Vampir’s Scalability

Scalable parallel Vampir visualization architecture

![Diagram showing scalable parallel architecture](image-url)
Vampir’s Scalability: Full View
Vampir’s Scalability: Process Clustering
Identification of relevant spots with a heat map:

- Hardware performance counters
- Metrics derived from events, e.g. frequency of function calls, here MPI_Barrier
Identification of relevant spots with a heat map:

• Hardware performance counters

• Metrics derived from events, e.g. frequency of function calls, here MPI_Barrier
Scalability: 200,000+ Processes in Vampir
Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial
Finding Performance Problems

Types of performance problems:
• Serial vs. parallel performance
• Bottlenecks related to MPI, OpenMP, CUDA, …
• Computation vs. communication
• Program phases: initialization/finalization, iteration, I/O, checkpointing, …

General strategy:
• Find highest consumer(s) of run-time
• Start with hypothesis about most severe performance problem, then confirm or disprove

Some examples (more in tutorial parts):
Problems with MPI

Too much runtime in MPI all-togethere
Problems with MPI + OpenMP

Imbalanced computation causes MPI waiting time

OpenMP threads mostly idle
Problems with strange MPI patterns

Strange MPI patterns propagate delays over successive MPI ranks
Problems for serial performance

High FP rate, low L3 cache miss rate

vs.

low FP rate due to a high L3 cache miss rate
Conclusions:
• User-friendly and comprehensible visualization of complex and highly parallel run-time behavior
• Various aspects of parallel performance
• Highly scalable by itself

Tutorial on Thursday & Friday:
• Run-time measurement with Score-P + hands-on
• Analysis with Vampir, Scalasca, TAU + hands-on
• Combination of Vampir and Scalasca
• Work with your code and tool(s) of your choice
• GPGPU measurement and analysis (on request)

http://www.vampir.eu