

# VAMPIR & VAMPIRTRACE INTRODUCTION AND OVERVIEW

Jens Doleschal, Andreas Knüpfer ZIH, TU Dresden jens.doleschal@tu-dresden.de September 2009

NESSEE UT









#### Overview



- Introduction
- Event Trace Visualization
- Vampir & VampirServer
- The Vampir Displays
  - Timeline
  - Process Timeline with Performance Counters
  - Summary Display
  - Message Statistics
- VampirTrace
  - Instrumentation & Run-Time Measurement
- Conclusions



## Why bother with performance analysis?

- well, why are you here after all?
- efficient usage of expensive and limited resources
- scalability to achieve next bigger simulation

## **Profiling and Tracing**

- have an optimization phase
  - just like testing and debugging phase
- use tools!
- avoid *do-it-yourself-with-printf* solutions, really!

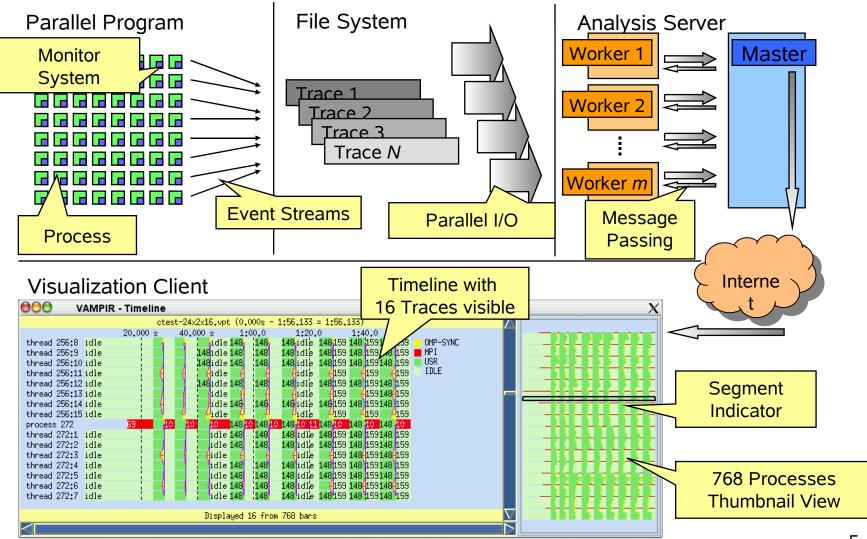


## **Trace Visualization**

- alternative and supplement to automatic analysis
- show dynamic run-time behavior graphically
- provide statistics and performance metrics
  - Global timeline for parallel processes/threads
  - Process timeline plus performance counters
  - Statistics summary display
  - Message statistics
  - more
- interactive browsing, zooming, selecting
  - adapt statistics to zoom level (time interval)
  - also for very large and highly parallel traces

#### **VampirServer Architecture**







The main displays of Vampir:

- Global Timeline
- Process Timeline w/o Counters
- Statistic Summary
- Summary Timeline
- Message Statistics
- Collective Operation Statistics
- Counter Timeline
- Call Tree

## Vampir Global Timeline Display



🗢 🤇 💻 Vampir - Timeline	<@a01> )	<b>a</b>   2
į	smg_big_ctr.otf (2.633 s - 2.634 s = 0.445 ms)	
2.633 s 2.633		
Process 0 hypre_StructInnerProd	S 2,004 S Application	
Process 1 hypre_StructInnerProd	Mineduce 308 MPI	
Process 2 hypre_StructInnerProd	308	
Process 3 hypre_StructInnerProd	308	
Process 4 hypre_StructInnerProd	171_Allroduco	
Process 5 hypre_StructInnerProd	308	
Process 6 hypre_StructInnerProd	308	
Process 7 hypre_StructInnerProd	308	
Process 8 hypre_StructInnerProd	308	
Process 9 hypre_StructInnerProd	Mile advoc	
Process 10 hypre_StructInnerProd	dP1_n11roduop	
Process 11 hypre_StructInnerProd	308	
Process 12 hypre_StructInnerProd	N/2 11/1 550.55	
Process 13 hypre_StructInnerProd	MRI_All.educo	
Process 14 hypre_StructInnerProd	308	
Process 15 hypre_StructInnerProd		
Process 16 hypre_StructInnerProd	SUB SUB SUB SUB	
Process 17 hypre_StructInnerProd	1921-011-03030 308 1921-011-03030 308	
Process 18 hypre_StructInnerProd Process 19 hypre_StructInnerProd		
Process 20 295 Willing course Process 21 hypre_StructInnerProd 1	300 308	
Process 22 hypre_StructInnerProd Process 22 hypre_StructInnerProd	300 300 300 300 300 300 300 300 300 300	
Process 23 hypre_StructInnerProd	TPL-01 hodupo	
Process 24 hypre_StructInnerProd	dP1_111_reduce	
Process 25 hypre_StructInnerProd	308	
Process 26 hypre_StructInnerProd	308	
Process 27 hypre_StructInnerProd	VPin (k) reduce	
Process 28 hypre_StructInnerProd	191_Alleduce 308	- 1
4	Γ	Ī
	Displayed 29 from 64 bars	

#### **Process Timeline Display**

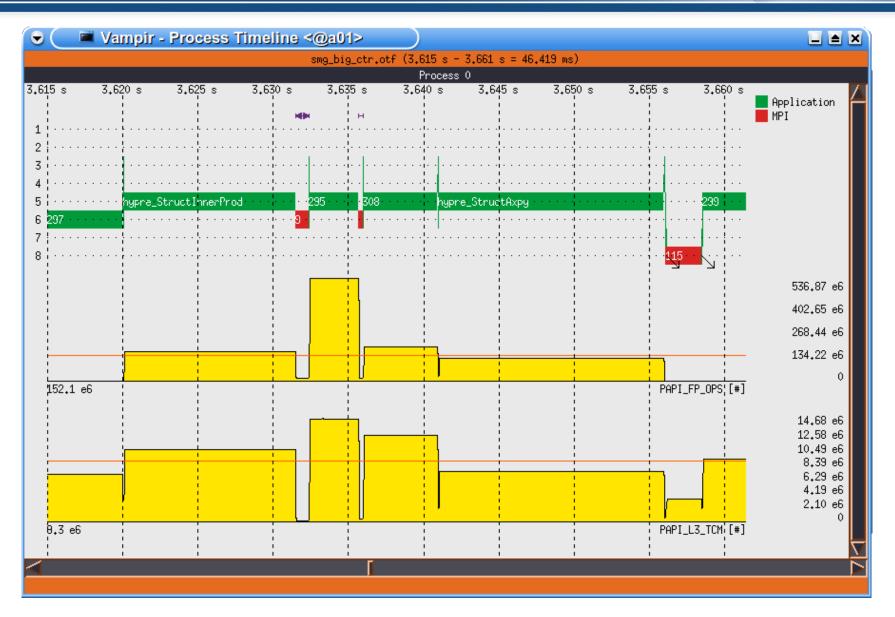


VI-H

#### 🕏 🤍 💻 Vampir - Process Timeline - < 2 > <@a01>



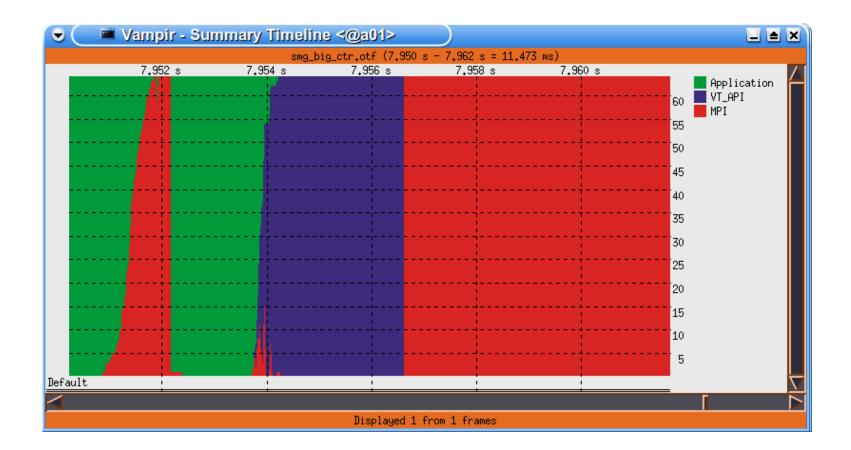




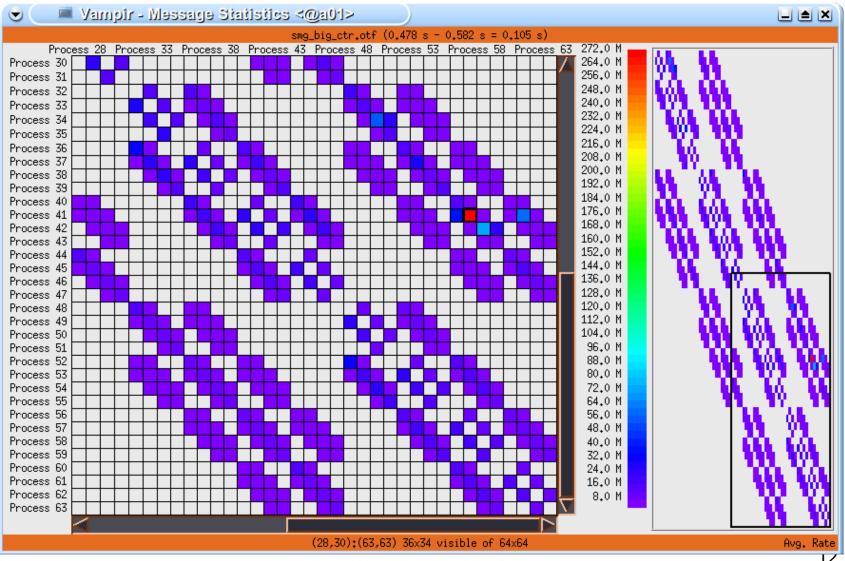


Name	Token	Value	
nypre_StructMatvecCompute	[299]	2:15,199	
nypre_StructAxpy	[ 306 ]	2:10.744	
nypre_StructInnerProd	[295]	1:14.087	
1PI_Finalize	[82]	1:04.179	
nypre_StructCopy	[297]	51.516 s	
1PI_Waitall	[ 163 ]	20.135 s	
nypre_StructVectorSetConstan	tVa[ 303 ]	20.124 s	
nypre_StructScale	[ 308 ]	15.580 s	
1PI_Allreduce	[9]	13,283 s	
1PI_Isend	[ 115 ]	9.010 s	
nypre_StructMatrixSetBoxValu	es [ 229 ]	8.455 s	
sync	[2]	5.654 s	
nain	[ 184 ]	4.661 s	
nypre_CAlloc	[ 186 ]	2.050 s	
nypre_StructVectorSetBoxValu	es [ 260 ]	1.827 s	
nypre_StructMatrixInitialize	Dat[ 224 ]	0.738 s	
nypre_StructKrylovAxpy	[ 305 ]	0.668 s	
1PI_Init	[ 108 ]	0.436 s	
nypre_StructKrylovCopyVector	[296]	0,221 s	
nypre_StructKrylovMatvec	[298]	0.215 s	
nypre_PCGSolve	[ 293 ]	0.212 s	
1PI_Irecv	[ 113 ]	0.190 s	
nypre_BoxGetSize	[227]	0.182 s	
nypre_Free	[ 187 ]	0 <b>.1</b> 69 s	
nypre_InitializeCommunication	n [250]	0.160 s	







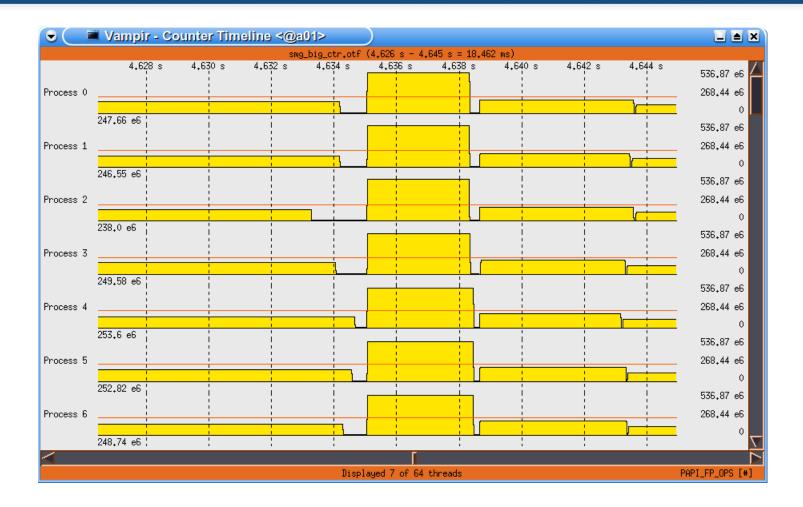


### **Collective Operation Statistics**



🔍 🗖 Va	mpir - Collective Op. Statistics <@a01>				
Busses A	smg_big_ctr.otf (0.587 s - 0.658 s = 71.398 ms) Process OProcess 4 Process 8 Process 13 Process 18 Process 23 Process 33				
Send	Nucless 4 Holess 13 Holes				
Receive	4.2 H 3.9 M 3.6 M 3.3 M 3.0 M 2.7 M 2.4 M 2.1 M 1.8 M 1.5 M 1.2 M 900.0 K 600.0 K 300.0 K				
	(0,0):(37,1) 38x2 visible of 64x2 Avg. Rate				







	)				
smg_big_ctr.otf					
Call Tree					
Call Tree <u></u>	Counts	Time	Search		
🗄 🖉 Þ hypre_StructKrylovDestroyVector	3	28.800 us47.900 us			
⊕… ▶ hypre_StructKrylovFree	4	33.900 us45.550 us			
🚊 🕨 🕨 hypre_StructKrylovMatvecDestroy	1	20,250 us,.47,050 us	Advanced Search		
🖨 🕨 🕨 hypre_StructMatvecDestroy	1	27.800 us40.150 us			
🕀 🕨 🕨 hypre_ComputePkgDestroy	1	30.450 us60.250 us	Folding		
···· ■ hypre_Free	1	2.600 us16.500 us	Fold Level: 🔽 3 👗		
▪ hypre_StructMatrixDestroy	1	18.100 us19.600 us			
• hypre_StructVectorDestroy	1	17,550 us.,18,700 us	Fold All		
🖨 👻 HYPRE_StructStencilCreate	1	26.400 us84.200 us	Unfold All		
■ hypre_CAlloc	1	2.650 us9.650 us			
⊕… 🕨 hypre_StructStencilCreate	1	19,500 us.,24,250 us			
⊢… ▼ HYPRE_StructStencilDestroy	1	19,850 us.,22,800 us	M		
— Global Call Breakdown ( hypre_StructMatvecDestroy (3	17)) —				
Caller			Callee		
hypre_StructKrylovMatvecDestroy		hypre_Free hypre_StructMatrixDestroy hypre_StructVectorDestroy hypre_ComputePkgDestroy			
ocess Filter: off   Timeline Portion: off Exclusive Ti					



## **Program Instrumentation**

- detect run-time events (points of interest)
- pass information to run-time measurement library

## **Profile Recording**

- collect aggregated information (Time, Counts, ... )
- about program and system entities
  - functions, loops, basic blocks
  - application, processes, threads, ...

## Trace Recording

- save individual event records together with precise timestamp and process or thread ID
- plus event specific information



- What do you need to do for it?
   Use VampirTrace
- Instrumentation (automatic with compiler wrappers)

CC=icc	CC=vtcc
CXX=icpc	CXX=vtcxx
F90=ifc	F90=vtf90
MPICC=mpicc	MPICC=vtcc

- Re-compile & re-link
- Trace Run (run with appropriate test data set)
- more details later



## What does VampirTrace do in the background?

- Instrumentation:
  - via compiler wrappers
  - by underlying compiler with specific options
  - MPI instrumentation with replacement lib
  - OpenMP instrumentation with Opari
  - also binary instrumentation with Dyninst
  - partial manual instrumentation



## What does VampirTrace do in the background?

- Trace Run:
  - event data collection
  - precise time measurement
  - parallel timer synchronization
  - collecting parallel process/thread traces
  - collecting performance counters (from PAPI, memory usage, POSIX I/O calls and fork/system/exec calls, and more ... )
  - filtering and grouping of function calls

#### Summary



- Vampir & VampirServer
  - interactive trace visualization and analysis
  - intuitive browsing and zooming
  - scalable to large trace data sizes (100GByte)
  - scalable to high parallelism (2000 processes)
- Vampir for Windows in progress, beta available
- VampirTrace
  - convenient instrumentation and measurement
  - hides away complicated details
  - provides many options and switches for experts
- VampirTrace is part of Open MPI 1.3

#### **Partners with pictures**















Forschungszentrum Jülich

- Jülich Supercomputing Centre
- RWTH Aachen University
  - Center for Computing and Communication
- Technical University of Dresden
- Center for Information Services and High Performance Computing

#### University of Tennessee

Innovative Computing Laboratory

Technical University of München

Chair for Computer Architecture

#### University of Stuttgart

High Performance Computing Centre













**Partner logos** 











RNTHAACHEN

UNIVERSITY



Universität Stuttgart