

Event Tracing with VAMPIR

Ventsislav Petkov Technische Universität München

petkovve@in.tum.de















- Introduction
- Vampir Displays
- Advanced topics
 - GPU Support
 - Scalability
- Finding Performance Bottlenecks
- Conclusion & Outlook



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 per process/thread

Trace Recording

- Individual event records
- Precise time stamp, process/thread ID
- Event specific information



Trace Visualization

- Alternative and supplement to automatic analysis
- Show dynamic run-time behavior visually
- Provide statistics and performance metrics
 - global timeline for parallel processes/threads
 - process timeline plus performance counters
 - statistic summary display
 - communication statistics, more ...
- Interactive browsing, zooming, selecting
 - adapt statistics to zoom level (time interval)
 - also for very large and highly parallel traces



Scalable parallel Vampir visualization architecture







- Introduction
- Vampir Displays
- Advanced topics
 - GPU Support
 - Scalability
- Finding Performance Bottlenecks
- Conclusion & Outlook



The main displays of Vampir:

- Master Timeline
- Process and Counter Timeline
- Function Summary
- Message Summary
- Process Summary
- Communication Matrix
- Call Tree

Vampir 7 Display Overview









	Vampir - [Trace View - /home/dolescha/tr	acefiles/feature	-traces/wrf-p64-io-men	m-rusage/wrf.1h.otf]	×
🍟 File 🛛 View	Help				_ 8 ×
<u>V</u> iew <u>C</u> hart	<u>F</u> ilter				
S 1 1	4. 👟 🤜 🔚 🎬 🖏 🐁 🕓 📿 🖈 📕				
+ ==	Timeline			Function Lege	nd
				Application	26343
2	6.860 s 26.865 s 26.870 s 26.875 s	26.880 s 26.8	85 s 26.890 s	DYN	
Process 0	MPI_Wait	MPI_Wait	solve_em_		
Process 1	MPI Wait	MRI_Wait	solve_em_	MEM	
Process 2	MPI_Wait		solve_em_		
Process 3	MPI_Wait	MPI_Wait	solve_em_		
Process 4	module_em_mp_rk_step_prep_	MPI_Wait	solve_em_	WRF	
Process 5	MPLWait		solve_em_		
Process 6	module_em_mp_rk_step_prep_		solve_em_		
Process 7	MPI_Wait	solv	/e_em_		
Process 8	module_em_mp_rk_step_prepMRL_Wait		solve_em_	Context Viev	v
Process 9	MPI_Wait		solve_em_	🔚 Master Timeline 🖾	+
Process 10	MPLWait		solve_em_	Property Value	
Process 11	MPL_wait		solve_em_	Display Master Ti	meline
Process 12	MPI_Wait		solve_em_	Type Function	
Process 13	MPLWait		solve_em_	Function Name MPI_wait	
Process 14	module_em_mp_rk_step_prep_		solve_em_	Interval Begin 26.87226	4 s
Process 15	module_em_mp_rk_step_prep_		solve_em_	Interval End 26.88523	6 s
Process 16	module_em_mp_rk_step_prep_		solve_em_	Duration 0.012972	S
Process 17	module_em_mp_rk_step_prep_		solve_em_	Source Line	
Process 18	MPI_Wait	Wait	solve_em_	Source Line	
Process 19	module_em_mp_rk_step_prep_		solve_em_		
Process 20	module_em_mp_rk_step_prep_		solve_em_		
Process 21	MPI_Wait	Wajt	solve_em_		
Process 22	MPI_Wait		solve_em		
Process 23	module_em_mp_rk_step_prep	Wait	solve_em_		
Process 24	MPI_Wait		MPI_Wait		
•					
		_			

















Vampir - [Trace View - /home/dolescha/trace	files/feature-trac	ces/wrf-p64-io-mem-i	rusage/wrf.1h.otf]
¥ File ∨iew Help			
⊻iew <u>C</u> hart <u>F</u> ilter			
1 🚍 😽 💹 🌑 🎘 🔚 🛗 🖏 🚱 💽 🛷 🛛 🗖	ير استراد الات		NALES OF THE PARTY PARTY OF THE PARTY OF T
Message Summary			Context View
7500 6000 4500 3000 1500	0	📑 Message Summar	y 🛛]
8593	15.039062 KiB 📥	Property	Value
4928	10.664062 KiB	Display	Message Summary
4911	10.9375 KiB	Message Size	105.820312 KiB
4900	23.203125 KiB	Message Transfer Rate	561.904092 MiB/s
2800	16.875 KiB —		Min: 21.282674 MiB/s, Max: 1.872321 GiB
2800	16.453125 KiB		
2254	48.398438 KiB		
2156	145.12 KiB		
2156	75.839844 KiB		
2071	60.800781 KiB		
1470	218.45 KiB		
1288	36.09375 KIB		
1288	35.2/3438 KIB	ĺ.	
Message Summary	-		
1600 MiB/s 1280 MiB/s 960 MiB/s 640 MiB/s 320 MiB/s 0 M	iB/s		
577.638488 MiB/s	212.68 KiB 📥		
562.663161 MiB/s	217.59375 KiB		
561.904092 MiB/s	105.82 KiB		
543.888495 MiB/s	108.28125 KiB		
493.213465 MiB/s	296.71875 KiB		
477.257369 MiB/s	159.25 KiB		
469.472582 MiB/s	162.9375 KiB		
467.421276 MiB/s	35.273438 KiB		
461.590886 MIB/s	36.09375 KiB		
457,004100 MIB/S	325.875 KiB		
431.100737 MIB/S	318.46875 KIB		
432.708147 MIB/s	147.65625 KIB		
404.034303 MIB/S	48.398438 KIB		





Vampir - [Trace View - /home/dolescha/tracefiles/feature-traces/wrf-p64-io-mem-rusage/wrf.1h.o	tf] 💶 💌
Y File View Help	_ @ ×
View Chart Eilter	
Process Summary	
0 s	22.0
0 s 5 s 10 s 15 s 20 s 25 s 30)s 35 s
12 MPI Brast Solve em modul iver MPI Wait mount mount open min	
2 MPI Boast solve em module river MPI Wait mourt moud moup open mu	
7 MPI Boast solve em moduleriver MPI Wait moder mod m Jopen m.	extse read
8 MPI Bcast solve em moduleriver MPI Wait moder mod m Lopen m	
5 MPI_Bcast Lsolve_em moduleriver_MPI_Wait Lmoder_mod m open m	
9 MPI_Bcast Loolve_em_ moduleriver_MPI_Wait Lmoder_ modld_m upen_m	
4 MPI_Bcast solve_emModuleriver_1MPI_Waitmodemodmopopen_1m	
3 MPI_Bcast solve_emmoduleriver_\MPI_Waitmoder_\modmm	
11 MPI_Bcast solve_em_ moduleriver_MPI_Waitmodermod_ m lopen_m	
	· •











Vampir - [Trace View - /home/dolescha/tracefiles/feature-traces/wr	f-p64-io-mem-rusage/wr	f. 1h. otf] 📃 💌
🖌 File View Help		_ B ×
View Chart Filter		
Call Tree	•	<u>//×</u>
All Processes		
Function	Min Inclusive Time	Max Inclusive Time 🛛 🗸 🔺
MPI_Wait	0.000000 s	711.716400 ms
🖶 🔜 module_radiation_driver_mp_radiation_driver_	0.000000 s	575.206150 ms
module_radiation_driver_mp_cal_cldfra_	0.000000 s	398.550000 μs
📄 🔜 module_radiation_driver_mp_radconst_	0.000000 s	211.300000 µs
malloc	0.000000 s	20.650000 μs
	0.000000 s	8.450000 μs
malloc	0.000000 s	2.450000 µs
tree tree	0.000000 s	1.950000 µs
module_microphysics_driver_mp_microphysics_driver_	0.000000 s	439.903/50 ms
module_em_mp_rk_tendency_	0.000000 s	398.592250 ms
module_em_mp_rk_step_prep_	0.000000 s	321.801500 ms
module_small_step_em_mb_advance_w_	0.000000 s	180.472900 ms
module_poi_anver_mp_poi_anver_	0.000000 s	158.015350 ms
module_awall_sten_em_mn_advance_uv	0.000000 s	146 178800 ms
module_small_step_em_mp_advance_uv_	0.000000 s	143.593150 ms
H module_em_mp_rk_scalar_tend	0.000000 s	120.663300 ms
MPL Isend	0.000000 s	95.921000 ms
module small step em mp calc p rho	0.000000 s	71.979550 ms
module small step em mp small step prep	0.000000 s	69.524800 ms
MPI Irecv	0.000000 s	68.071900 ms
module small step em mp sumflux	0.000000 s	67.906000 ms
modulo om mersk undete seeler	0.00000 e	51.027450 mc
module_radiation_driver_mp_radconst		×
Callers Callees		
module radiation driver mp radiation driver		





- Introduction
- Vampir Displays
- Advanced topics
 - GPU Support
 - Scalability
- Finding Performance Bottlenecks
- Conclusion & Outlook





- Full support for GPGPU computing, CUDA and OpenCL
- Shows Kernel invocations, idle time, and DMA transfers





- Support for hybrid modes: GPGPU + MPI + threads
 - Function invocations in host processes (Process X) and threads
 - Kernel invocations in CUDA threads (Thread x/y)
 - Host-GPU interactions via CUDA API (light arrows)
 - Host-Host interactions via MPI (bold arrows)





- Details about kernel invocations
- Derived counter values, e.g. about GPU execution rate



- Currently no scalability limits provided enough analysis processes and enough distributed memory
- Full-scale ORNL Jaguar trace opened with 20,000
 VampirServer analysis processes
- Vampir \rightarrow find ways to show that much data



- Compact timelines (Fit to chart height)
 - In master timeline and performance radar
 - Visualization of more processes than vertical pixels
- Clustering
 - Allows detection of groups with similar behavior and outliere
- Performance radar
 - Highlighting performance conditions in a global timeline

Scalability Feature I: Fit to chart height





Scalability feature II: Clustering





Scalability feature II: Clustering

VI-HPS



Scalability Feature III: Performance Radar



- Identification of relevant spots with a heat map
- Performance counters and properties derived from trace events
- Arithmetics on counter data and event properties



Scalability Feature III: Performance Radar



- Easily reveals where given functions occur, e.g. MPI_Barrier
- Shows density of function calls

00	🥁 Trace View – /Traces/64_pmp2/pmp.otf – Vampir							-		
0.s				07.10	2.0					97 s
				Timelir	10					2 X
	10 s	20 s	30 s	40 s	50 s	60 s	70 s	80 s	90 s	
Process 0		_		ŕ						
Process 4	÷	1								
Process 8					4					
Process 12		and the second second			_					
Process 16	-									
Process 20		2								
Process 24	1	<								
Process 28										
Process 32	1	-								
Process 36	į	_								
Process 40	-	-								
Process 44	4									
Process 48	1									
Process 52										
Process 56	-				Ę					
Process 60	1	_			_					
Occurrences of Functi	on "MPI Bai	rrier" over T	ime							
Process 0										
Process 4		_								11
Process 8		-	_							11
Process 12										ii 🗌
Process 16		-								11
Process 20		-	-							11
Process 24										
Process 28										11
Process 32		_								-
Process 36										11
Process 40										1
Process 44		_								11
Process 48										11
Process 52										1
Process 56		_								
Process 60		_	-2							1
	1			31	12					

Scalability Feature III: Performance Radar



 Allows global sum over all processes, ranks and threads





¥	Vampir - Trace View - Jaguar:3	0045:/lustre/widow1/scratch/tilsche/iofsl/traces/542425/merged/s3d.otf		_ + X
Y File Edit Chart Filter Window Help		287/3 5 241.3 5		
	Timeline	2180316 21 X	Function Summary	
Process 0		Filter Processes		× *
Process 6433 Process 13073 Process 1073		Z Include/Exclude All		
Process 26353	rocess Groups	Y Include/Exclude All		_rate_bounds_
Process 32634		✓ MPI Communicator 0 58/58 > ▲		lc_temp_
Process 52914	ammunicators	MPI Communicator 1 48/48 >		looptool_ .centh allpts
Process 59534 Process 6194	ommunicators	MPI Communicator 10 72/72 >		ae It
Process 72834 Process 79474 Process 706114		✓ MPI Commator 100 72/72 >		ge_rc_
Process 92754 P	rocess Hierarchy	✓ MPI Commtor 1000 72/72 >		_buff
Process 106035		✓ MPI Commor 10000 48/48 >		
Process 112675		✓ MPI Commor 10001 48/48 >		+
Process 125955		✓ MPI Commor 10002 48/48 >		
Process 145875		✓ MPI Commor 10003 48/48 >		
Process 152515		✓ MPI Commor 10004 48/48 >		
Process 172436		✓ MPI Commor 10005 48/48 >		
Process 19076		✓ MPI Comm. or 10006 48/48 >		
Process 192356		MPI Comm or 10007 48/48 >		
Process 6 PEE		MPI Comm or 10008 48/48 >		
Process 13730		MPI Comm or 10009 48/48 >		
Process 27459		MPI Comm. tor 1000 72/72 >		
Process 41188 Process 48053		MPI Comm. or 10010 49/49 >		
Process 54918 Process 61782		MPI Comm. or 10010 48/48 >		
Process 68647 Process 75512	Imber of processes	MPI Comm. or 10011 48/48 >		
Process 82376 Process 89241 200	0448	MPI Commor 10012 58/58 >		
Process 96106 Process 102970		V MPI Commor 10013 58/58 >		0 ⁰
Process 108350 Process 116760	lected processes	▼ MPI Commor 10014 58/58 >		Sum
Process 130429 200	0448	✓ MPI Commor 10015 58/58 > -		12 KiB 18 KiB
Process 144158 Process 151023				6iB 768 B
Process 157888 Process 164752	Reset		🖌 Apply 🔢 🔴 Cancel 📔 🖉 OK	
Process 171617 Process 178482	<u> </u>			
Process 185346 Process 192211				
0.000 0.025 0.050 0.075 0.100	0.125 0.150 0.175	0.200 0.225 0.250 0.275 0.300		
241.535 S				Connected: laguar





- Introduction
- Vampir Displays
- Advanced topics
 - GPU Support
 - Scalability
- Finding Performance Bottlenecks
- Conclusion & Outlook



Vampir trace visualization

- several displays with many options
- identify essential parts of an application (initialization, main iteration, I/O, finalization)
- identify important components of the code (serial computation, MPI P2P, collective MPI, OpenMP)
- make a hypothesis about performance problems
- consider application's internal workings if known
- select the appropriate displays
- use statistic displays in conjunction with timelines



- Communication as such (dominating over computation)
- Late sender, late receiver
- Point-to-point messages instead of collective communication
- Unmatched messages
- Overcharge of MPI's buffers
- Bursts of large messages (bandwidth)
- Frequent short messages (latency)
- Unnecessary synchronization (barrier)
- The above usually result in a high MPI time share



Too much runtime share for MPI all-together





Long bursts of MPI_AllReduce operations



VI-HPS

Long bursts of MPI_AllReduce operations

Replace by NBC non-blocking allreduce + wait

 \rightarrow





Long bursts of MPI_AllReduce operations

Replace by NBC non-blocking allreduce + wait

(zoom)

 \rightarrow





Chains of MPI Sendrecv replace





Chains of MPI Sendrecv replace (zoom)

Propagate delays over successive MPI ranks

Chains of MPI Sendrecv_ replace

 \rightarrow

Replace by faster overlapping series of Send, Irecv, and Waitall

Chains of MPI Sendrecv_ replace

→ Replace by faster overlapping series of Send, Irecv, and Waitall (zoom)

Many MPI barriers are unneeded

Many MPI barriers are unneeded → Remove barriers if possible, following

operations

regular but

are faster.

start less

- Unbalanced computation (single late comer)
- Strictly serial parts of program (idle threads)
- Frequent tiny function calls, sparse loops
- Memory bound computation
 - Inefficient L1/L2/L3 cache usage, TLB misses
 - Detectable via HW performance counters
- I/O bound computation
 - Slow input/output, sequential I/O
 - I/O load imbalance
- Exception handling

¥ 💿

 \odot

Mostly idle OpenMP threads → Low parallel

efficiency

_ 8 × 🖌 File Edit Chart Filter Window Help 19.633 s Timeline Function Summary All Processes, Number of Invocations per Function 0 s 5 s 10 s 15 s 5 k 0 k Process 0 !\$omp ibarri...reading.c:21 9.080 Thread 1/0 9,080 !\$omp parall...reading.c:20 Process 1 4.544 hypre Free Thread 1/1 4.540 hypre NumThreads Process 2 4,540 parallel region Thread 1/2 4.532 hypre_BoxGetSize Process 3 hypre_CAlloc 2,416 Thread 1/3 1,952 hypre MAlloc Process 4 1.280 MPI Isend Thread 1/4 1.280 MPI Irecv Process 5 1,272 MPI Allreduce Thread 1/5 hypre StructInnerProd 1,224 Process 6 Context View Thread 1/6 Function Summary Process 7 Property Value Thread 1/7 Function Summary Display Function !\$omp parallel @threading.c:20 Function Group OMP (5) Number of Invocations 16 (20%) 41 4

Vampir <@neptun> <3>

Mostly idle OpenMP threads →

Low parallel efficiency

(zoom)

High rate of 🚟 😽 🔼 1 FPop/s with 3 s low rate of Process 0 1 main L3 cache 2 mandelbroi 3 iteration misses VS. low FPop/s 4.0 G 3.5 G rate due to 3.0 G 2.5 G a high L3 2.0 G 1.5 G 1.0 G miss rate. 0.5 G 0.0 G

(zoom)

- Introduction
- Vampir Displays
- Advanced topics
 - GPU Support
 - Scalability
- Finding Performance Bottlenecks
- Conclusion & Outlook

Matthias Jurenz, Andreas Knüpfer, Ronny Brendel, Matthias Lieber, Jens Doleschal, Jens Domke, Holger Mickler, Daniel Hackenberg, Michael Heyde, Thomas Ilsche, Guido Juckeland, Robert Dietrich, Frank Winkler, Michael Kluge, Matthias Müller, Holger Brunst, Ronald Geisler, Reinhard Neumann, Bert Wesarg, Rene Widera, Thomas Ilsche, Matthias Weber, Thomas William, Hartmut Mix,

and Wolfgang E. Nagel

Vampir and VampirTrace are available at http://www.vampir.eu and http://www.tu-dresden.de/zih/vampirtrace/, support via vampirsupport@zih.tu-dresden.de

IENNESSEE U

ÜLICH

