

VAMPIR & VAMPIRTRACE DETAILS AND HANDS-ON

Matthias Weber, Jens Doleschal, Andreas Knüpfer ZIH, TU Dresden matthias.weber@tu-dresden.de November 2010

NIVERSITY

TECHNISCHE UNIVERSITÄT





INIVERSITAT



- Event Tracing in General
- Hands-on: NPB 3.3 BT-MPI
- Finding Performance Bottlenecks



VAMPIR & VAMPIRTRACE Event Tracing in General

Usage order of the Vampir Performance Analysis Toolset



- 1. Instrument your application with VampirTrace
- 2. Run your application with an appropriate test set
- 3. Analyze your trace file with Vampir
 - Small trace files can be analyzed on your local workstation
 - 1. Start your local Vampir
 - 2. Load trace file from your local disk
 - Large trace files should be stored on the cluster file system
 - 1. Start VampirServer on your analysis cluster
 - 2. Start your local Vampir
 - 3. Connect local Vampir with the VampirServer on the analysis cluster
 - 4. Load trace file from the cluster file system

Profiling and Tracing



- Tracing Advantages
 - Preserve temporal and spatial relationships
 - Allow reconstruction of dynamic behavior on any required abstraction level
 - Profiles can be calculated from traces
- Tracing Disadvantages
 - Traces can become very large
 - May cause perturbation
 - Instrumentation and tracing is complicated
 - Event buffering, clock synchronization, ...

Common Event Types



- Enter/leave of function/routine/region
 - time stamp, process/thread, function ID
- Send/receive of P2P message (MPI)
 - time stamp, sender, receiver, length, tag, communicator
- Collective communication (MPI)
 - time stamp, process, root, communicator, # bytes
- Hardware performance counter values
 - time stamp, process, counter ID, value
- etc.

Open Trace Format (OTF)



- Open source trace file format
- Available at http://www.tu-dresden.de/zih/otf
- Includes powerful libotf for reading/parsing/writing in custom applications
- Multi-level API:
 - High level interface for analysis tools
 - Low level interface for trace libraries
- Actively developed by TU Dresden in cooperation with the University of Oregon and the Lawrence Livermore National Laboratory



- Instrumentation: Process of modifying programs to detect and report events
- There are various ways of instrumentation:
 - Manually
 - Large effort, error prone
 - Difficult to manage
 - Automatically
 - Via source to source translation
 - Via compiler instrumentation
 - Program Database Toolkit (PDT)
 - OpenMP Pragma And Region Instrumenter (Opari)





manually or automatically



- Instrumentation with VampirTrace
 - Hide instrumentation in compiler wrapper
 - Use underlying compiler, add appropriate options

CC=icc	CC=vtcc
CXX=icpc	CXX=vtcxx
F90=ifc	F90=vtf90
MPICC=mpicc	MPICC=vtcc -vt:cc mpicc

- Re-compile & re-link
- Trace run
 - User representative test input
 - Set parameters, environment variables, etc.
 - Perform trace run
- Get Trace



VAMPIR & VAMPIRTRACE HANDS-ON: NPB 3.3 BT-MPI

Hands-on: NPB 3.3 BT-MPI

. . .



• Move into tutorial directory in your home directory

% cd tutorial

Select the VampirTrace compiler wrappers

```
% gedit config/make.def
    -> comment out line 32, resulting in:
              32: #MPIF77 = mpif77
    -> remove the comment from line 38, resulting in:
              38: MPIF77 = vtf77 –vt:f77 mpif77
    -> comment out line 89, resulting in:
              89: #MPICC = mpicc
    -> remove the comment from line 95, resulting in:
              95: MPICC = vtcc -vt:cc mpicc
```



• Build benchmark

% make clean; make suite

Launch as MPI application

```
% cd bin.vampir; export VT_FILE_PREFIX=bt_1_initial
% mpiexec –np 16 bt_W.16
NAS Parallel Benchmarks 3.3 -- BT Benchmark
```

```
Size: 24x 24x 24
Iterations: 200 dt: 0.0008000
Number of active processes: 16
```

```
Time step 1
```

```
•••
```

Time step 60 [0]VampirTrace: Maximum number of buffer flushes reached \ (VT_MAX_FLUSHES=1)

```
[0]VampirTrace: Tracing switched off permanently
```

```
Time step 200
```



• Resulting trace files

```
% Is -alh
4,1M bt_1_initial.16
3,6K bt_1_initial.16.0.def.z
3.8M bt_1_initial.16.0.marker.z
3.8M bt_1_initial.16.10.events.z
3.8M bt_1_initial.16.1.events.z
3.8M bt_1_initial.16.2.events.z
3.8M bt_1_initial.16.2.events.z
...
3.8M bt_1_initial.16.c.events.z
3.8M bt_1_initial.16.c.events.z
3.8M bt_1_initial.16.d.events.z
3.8M bt_1_initial.16.d.events.z
3.8M bt_1_initial.16.e.events.z
3.8M bt_1_initial.16.f.events.z
66 bt_1_initial.16.otf
```

• Visualization with Vampir7

% vampir bt_1_initial.16.otf



VI-HPS



V					V	ampir - [T	race Vi	ew - bt_V	V.16.otf]				_ + X
🖌 File	View	Help												OOX
View	Chart	Filter												
	-	-			5 (1 84)	A								
		<u> </u>			+ 🐶	🙂 🍟	<i>»</i>							
All Drov		Accumulate	d Evolucivo	Time per	Function		Funct	ion Summan	y					28
AILFIO	600 s	ACCUMUALE		11me per 1450 s	A00 c	350 s	300 s	250 s	200 s	150 s	100 s	50 s () c	
653.6	35754	s	500 5	4503	400 3	550 5	500 3	2303	200 3	150 5	100 5	503 0	MPI Wait	
						383.610785	s						MAIN	
					-			278.77477	s				MPI Waitall	
									215.019	456 s			flush	=
											24.7	/63068 s	MPI_Init	
												4.109706 s	sync time	
												2.678737 s	binvcrhs_	
								~				2.313013 s	matmul_sub_	
												1.998098 s	x_solve_cell_	
												1.959289 s	matvec_sub_	
												1.912154 s	y_solve_cell_	
												1.833275 s	z_solve_cell_	
												1.718103 s	initialize_	
												1.675548 s	exact_solution_	
												1.279341 s	MPI_Bcast	
												<1 s	exact_rhs_	
												<1 s	MPI_Barrier	
												<1 s	lhsinit_	
												<1 s	compute_rhs_	
												<1 s	copy_faces_	
												<1 s	MPI_Comm_split	
												<1 s	MPI_lsend	
												<1 s	x_backsubstitute	e_
												<1 s	MPI_Comm_dup	
													i	
244 s												On Liver	(n)	ing in V

On LiveDVD (running in VM)



Issue:

Tracing was switched off because the internal trace buffer was too small

Result:

- 1. Asynchronous behavior of the application due to buffer flush of the measurement system
- 2. No tracing information available after flush operation

Solutions:

1. Increase trace buffer size

- 2. Increase number of allowed buffer flushes (not recommended)
- 3. Use filter mechanisms to reduce the number of recorded events
- Switch tracing on/off if your application works in an iterative manner to reduce the number of recorded events (see the VampirTrace manual for more details)



 Decrease number of buffer flushes by increasing the buffer size

% export VT_MAX_FLUSHES=1 VT_BUFFER_SIZE=120M

• Set a new file prefix

% export VT_FILE_PREFIX=bt_2_buffer_120M

Launch as MPI application

% mpiexec -np 16 bt_W.16

Remove the old trace first !

• Visualization with Vampir 7

% vampir bt_2_buffer.16.otf

Only for laptops with at least 2GB main memory !



VI-HPS



0		Vampir -	[Trace View ·	bt_2_buffer_1	20M.otf]		_ 0	×
🥁 File View Help							_ 8	×
View Chart Filter								
] 🗮 📷 🚧 🌕 🗃	1	🔯 🕓 💡 2	*	la Barthaa.	aradan dalambarahan kun k	kti di akti o nanki na na an kunadad adi		
All Processes Accumulated Ex	clusive Time per Fi	unction	Fun	ction Summary —				
14 s	12 s	10 s	8 5	65	4 5	25	0 5	
16.044812 s	12.5	10.5					MPI_Finalize	
		10.427354 s					sync time	
					4.390518 s		binvcrhs_	
					4.316299 s		matvec_sub_	
					4 192216 s		MPI_Wait	
					4.132115 s		matmul_sub_	
					4.122368 s		z_solve_cell_	
					4.074546 s		y_solve_cell_	
					4.002995 s		x_solve_cell_	
						586.20175 ms	copy_faces_	
						535.4764 ms	compute_rhs_	
						531.168 ms	MPI_Waitall	
						367.70525 ms	MPI_Irecv	
						358.02745 ms	MPI_Isend	
						281.89385 ms	z_send_solve_info_	
						251.46875 ms	y_send_solve_info_	
						190.4338 ms	binvrhs_	
						139.8223 ms	x_send_solve_info_	
						137.8881 ms	exact_solution_	
						96.4114 ms	x_solve_	
						95.21895 ms	y_solve_	
						95.11725 ms	z_solve_	
						92.7048 ms	x_backsubstitute_	
						92.6168 ms	MPI_Init	
						89.97525 ms	initialize_	
						75.1309 ms	z_backsubstitute_	
						71 5336		
						0	n an SGI Altix	470



Issue: Each function entry/exit, MPI event was recorded

Result:

Trace file becomes large even for short application runs and may not fit into the main memory

Solutions:

- 1. Use filter mechanisms to reduce the number of recorded events
- Switch tracing on/off if your application works in an iterative manner to reduce the number of recorded events (see the VampirTrace manual for more details)

Function Filtering



- Filtering is one of the ways to reduce trace size
- Environment variable VT_FILTER_SPEC

% export VT_FILTER_SPEC = /home/user/filter.spec

• Filter definition file contains a list of filters

```
my_*;test_* -- 1000
debug_* -- 0
calculate -- -1
* -- 1000000
```

- See also the vtfilter tool
 - can generate a customized filter file
 - can reduce the size of existing trace files

Switch Tracing On/Off



- Starting and stopping of tracing should be performed with care
- Tracing has to be activated on the same call stack level as it was switched off to ensure the consistency of the trace file
- Useful if your program behaves in an iterative manner or if you are only interested in some parts of your application

```
#include "vt_user.h"
...
VT_OFF();
for( i=1; i < 100; i++ ) { do something uninteresting };
VT_ON();
...</pre>
```

• Recompile your source code with the user macro "-DVTRACE"

% vtcc ... -DVTRACE source_code.c ...



• Generate your filter specification and set environment

% gedit filter.txt binvcrhs*; matvec_sub*; matmul_sub* -- 0

% export VT_FILTER_SPEC=filter.txt

• Set a new file prefix

% export VT_FILE_PREFIX=bt_3_filter

Launch as MPI application

% mpiexec -np 16 bt_W.16

Remove the old trace first !

• Visualization with Vampir 7

% vampir bt_3_filter.16.otf



0	Vampir - [Trace View - bt_	3_filter.otf]
🥁 File View Help		<u>_ 문 ×</u>
<u>V</u> iew <u>C</u> hart <u>F</u> ilter		
] 🗮 📷 🛄 🌕 🖉] 🔠 🎬 🚳 🚯 🚱 🖓 🖈 🛛 🚛 💶 🗖	a teletara ettima a lukusta astada utarida del adacante a telese
	Timeline	Function Summary
0.5		All Processes, Accumulated Exclusive Time per Function Group
	15 25 35 45	25 S 20 S 15 S 10 S 5 S 0 S
Process 0 sync time	MPI_Finalize	21:197246 s
Process 1 sync time	MPI_Finalize	15.350851 s
Process 2 sync time	MPI_Finalize	
Process 3 sync time	MPI_Finalize	Function Legend
Process 4 sync time	MPI_Finalize	Application
Process 5	MPI_Finalize	
Process 6 sync ti	me • • • • • • • • • • • • • • • • • • •	
Process 7 sync time	e e e e e e e e e e e e e e e e e e e	Communication Matrix View
Process 8 sync tir	me • • • • • • • • • • • • • • • • • • •	Average Bandwidth
Process 9	The provide the providet the prov	00000000000000000000
Process 10 sync tir	ne 🛉 🛉 🙌 🗧 🕂 🗧 🚧 🗧 🚧 MPI_Finalize	
Process 11 sync time	MPI_Finalize	
Process 12 sync time	MPI_Finalize	
Process 13 sync time	MPI_Finalize	Process 0
Process 14 sync time	MPI_Finalize	Process 2
Process 15 sync time	MPI_Finalize	Process 4
		Process 6
		Process 8
		Process 10 100 M
		Process 12
		Process 14
<u> </u>		
		On an SGI Altix470



· · · · · · · · · · · · · · · · · · ·		Vam	pir - [Trace Vie	ew - bt_3_filte	er.otf]		X			
🤟 File View Help							_ & ×			
View Chart Filter										
n di succese de setembre de s										
All Processes Accumulated Exc	lusive Time per Fi	unction	Fund	tion Summary —						
14 c	12 e	10 c	8 6	6.5	4.5	2 6	0.5			
16.038443 s	12 5	10 5	03	03	43	2 3	MPI Finalize			
15.350851 s							sync time			
			7.625	073 s			z solve cell			
			7.588	402 s			y solve cell			
			7.50	9255 s			x solve cell			
					3.75982	?7 s	MPI_Wait			
						582.38795 ms	copy_faces_			
						539.38395 ms	compute_rhs_			
						500.0404 m	s MPI_Waitall			
						359.35745 n	ns MPI_Irecv			
						348.6544 n	ns MPI_Isend			
						278.206	ms z_send_solve_info_			
						247.75795	ms y_send_solve_info_			
						199.47945	ms binvrhs_			
						137.36535	5 ms exact_solution_			
						134.78645	5 ms x_send_solve_info_			
						101.5465	5 ms MPI_Init			
						96.1484	5 ms x_solve_			
						95.347	1 ms z_solve_			
						94.4999	5 ms y_solve_			
						90.1934	5 ms initialize_			
						87.824	5 ms x_backsubstitute_			
						71.5572	5 ms z_backsubstitute_			
						67.04	3 ms y_backsubstitute_			
						35.7870	5 ms exact_rhs_			
						35.3153	5 ms Ihsabinit_			
						<u> </u>	- 1			
				·			On an SGI Altix47			



Issue:

Runtime filtering will be called for every event

Result:

Runtime filtering may increases the runtime overhead

Solutions:

- 1. Use manual source instrumentation (high effort, not recommended)
 - 2. Only instrument interesting source files with VampirTrace
- Switch tracing on/off if your application works in an iterative manner to reduce the number of recorded events (see the VampirTrace manual for more details)

However, these trace files include no information about the computational performance of your application. Therefore, in the **next step**:

Recording of hardware performance counters





- PAPI counters can be included in traces
 - If VampirTrace was build with PAPI support
 - If PAPI is available on the platform
- VT_METRICS specifies a list of PAPI counters

% export VT_METRICS = PAPI_FP_OPS:PAPI_L2_TCM

 see also the PAPI commands papi_avail and papi_command_line



- Memory allocation counters can be recorded:
 - If VampirTrace build with memory allocation tracing support
 - If GNU glibc is used on the platform
- intercept glibc functions like "malloc" and "free"
- Environment variable VT_MEMTRACE

% export VT_MEMTRACE = yes

- I/O counters can be included in traces
 - If VampirTrace was build with I/O tracing support
- Standard I/O calls like "open" and "read" are recorded
- Environment variable VT_IOTRACE

% export VT_IOTRACE = yes



• Record PAPI hardware counters

% papi_avail % papi_event_chooser PRESET PAPI_FP_OPS % export VT_METRICS=PAPI_FP_OPS:PAPI_L2_TCM

• Set a new file prefix

% export VT_FILE_PREFIX=bt_4_papi

Launch as MPI application

% mpiexec -np 16 bt_W.16

Remove the old trace first !

• Visualization with Vampir 7

% vampir bt_4_papi.16.otf



VI-HPS

Vampir Trace Files: NPB 3.3 BT-MPI and further examples



• All NPB 3.3 BT-MPI trace files of a hands-on session are located at:

% cd \$HOME/workshop-vampirtrace/Examples/npb-bt-mpi/result_thinkpad

• All NPB 3.3 BT-MPI trace files created on a SGI-Altix are located at:

% cd \$HOME/workshop-vampirtrace/Examples/npb-bt-mpi/result_altix

• SMG 2000 trace files with various configurations are located at:

% cd \$HOME/workshop-vampirtrace/Examples/smg2000/

• Mandelbrot trace files can be found at:

% cd \$HOME/workshop-vampirtrace/Examples/mandelbrot



- Groups can be defined for related functions
 - Groups can be assigned different colors, highlighting different activities
- Environment variable VT_GROUPS_SPEC

% export VT_GROUPS_SPEC = /home/user/groups.spec

• Group file contains a list of associated entries

CALC=calculate MISC=my*;test UNKNOWN=*



- control options by environment variables:
 - VT PFORM GDIR – VT PFORM LDIR – VT FILE PREFIX – VT BUFFER SIZE – VT MAX FLUSHES - VT MEMTRACE – VT MPICHECK - VT IOTRACE - VT MPITRACE - VT FILTER_SPEC - VT GROUPS SPEC - VT METRICS

Directory for final trace files Directory for intermediate files Trace file name Internal trace buffer size Max number of buffer flushes Enable memory allocation tracing Enable MPI checking Enable I/O tracing Enable MPI tracing Name of filter definition file Name of grouping definition file PAPI counter selection





Thanks for your attention.



VAMPIR & VAMPIRTRACE Finding Performance Bottlenecks



- Trace Visualization
 - Vampir provides a number of display types
 - Each allows many different options
- Advice
 - 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

Finding Bottlenecks



- Communication
- Computation
- Memory, I/O, etc.
- Tracing itself



- Communications 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)

All of the above usually result in high MPI time share





Example: prevalent communication





prevalent communication: MPI_Allreduce





prevalent communication: timeline view



X				V	/ampir - Timelir	le			
			flash_modified_ne	w_00256.otf (4:00.	633 - 4:00.794 = (0.161 s)			
	4:00.6	64 4:00,68	6 4:00.68 4:00.	7 4:00,72	4:00.74 4	:00.76 4:0	0.78		
Process	0 user	MPI_Barrier	81 MPI_Ssend	MPI.	_Waitall u	iser	Appli Appli	ication 🔚 🚹	
Process :	1 user i	<u> 1PI_Barrier</u>	81 MPI_Ssend	MPI_Wa	aitall <mark>use</mark>	r	MPI		NA DA DA ANTA DI CINA DA DA BIBLIO DA
Process :	2 <u>user</u>	MPI_Barrier	 <u>81 MPI_Ssend</u> 		MPI_Waitall	user	NoSer	ndRecvReplace	
Process	3 <u>user</u>	MPI_Barrier	81 MPI_Sser	<u>d MPI_W</u>	Jaitall	user			AN A
Process	4 <u>user</u>	MPI_Bann	ier 81 MPI_Sse	nd	151	user			
Process	b user	MPI_Ba	nnien 81 MPI_Sse	nd	MPI_Waitall	user			
Process	6 user	MP1_Barrie	r 81 MPI_5send		10	1 user	+		
Process	7 User 0 Voor	MDI Dappio	annien <u>61 MPI_</u> bsei 91 MPI_Coord	na	MDT Use	user	1		
Process	o user 9 user	MPI Dannien	n oi nrijosenu 01 MDT Soond	MDT	Maitall	user	1		
Process -	5 <u>user</u> 10 user	MDT Dann	ion 01 MDT Coond		_waltail MDT_Usi+s11	user	1		
Process :	10 user	MPT Ra	ncier MPI Irecu MP	I Seend	In I_walcall	131 user			
Process	X				Vampir - T	imeline <2>			
Process					- nampii - i				
Process			flash_less	barrier_05_x1_0025	6.otf (4:01.584 -	4:01.744 = 0.1	61 s)		
Process		4:0	1.6 4:01.62 4:0	1.64 4:01.66	4:01.68	4:01.7 4	4:01.72 4:01.74	4	
Process	Process (user	81MP1_Ssend	MP1_Wa	aitall user			Hpplication	
Process	Process J	user	01 MDT Cound	rir1_wa	474 User				
	Process 2	user	01 NPT_Ssend	MDT Ustes11	151 user			NoSendRecvReplace	
	Process /	usen	01 NF1_3Senu 01 MPT Seend	nFI_waltall	171	LISON			
	Process P	USCI		nd	MPI Waitall	user			
	Process 6	user	81 MPI Ssend		131	user			
	Process 7	viser	81 MPT Ser	end	101	user			
	Process 8	user	81 MP1 Ssend	M	1PI Waitall	user			
	Process 9	Juser	81 MPI Ssend	MPI Wai	tall	user			
	Process 1	l0 user	81 MPI_Ssend		MPI_Waital	user			
	Process 1	1 user	81 <u>MP</u> 1	Ssend	131	user	user		
	Process 1	2 user	81 MPI_S	send	MPI_Waitall	user			
	Process 1	l3 user	81 MPI_S	end		user			
	Process 1	l4 user	81 MPI_Ssend		MPI_Waitall	user			
	Process 1	l5 user	81 MPI_Ssend			131. user			
	Process 1	l6 user	81 MPI_Ssend			131user			
	Process 1	7 user	81 MPI_Ssend		MPI_Waitall	user	user		
	\triangleleft			Í					
				Displayed	d 18 from 256 bars				

unnecessary MPI_Barriers



- unbalanced computation
 - single late comer
- strictly serial parts of program
 - idle processes/threads
- very frequent tiny function calls
- sparse loops

Bottlenecks in Computation



- 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 on single process
 - I/O load imbalance
- exception handling





low FP rate due to heavy cache misses

Bottlenecks in Computation





low FP rate due to heavy FP exceptions





irregular slow I/O operations

Effects due to Tracing



- measurement overhead
 - especially grave for tiny function calls
 - solve with selective instrumentation
- long/frequent/asynchronous trace buffer flushes
- too man concurrent counters
- heisenbugs





Trace buffer flushes are explicitly marked in the trace. It is rather harmless at the end of a trace as shown here.



- performance analysis very important in HPC
- use performance analysis tools for profiling and tracing
- do not spend effort in DIY solutions, e.g. like printf-debugging
- use tracing tools with some precautions
 - overhead
 - data volume
- let us know about problems and about feature wishes
- vampirsupport@zih.tu-dresden.de



- This work would have been impossible without the dedication of:
 - Matthias Lieber (Tracing & Analysis)
 - Matthias Jurenz (VampirTrace Software & Support)
 - Matthias Weber (Vampir Software & Support)
- The Vampir Team:

Matthias Jurenz, Andreas Knüpfer, Ronny Brendel, Matthias Lieber, Jens Doleschal, Holger Mickler, Daniel Hackenberg, Michael Heyde, Guido Juckeland, Dietrich Robert, Johannes Spazier, Michael Kluge, Matthias Müller, Holger Brunst, Ronald Geisler, Reinhard Neumann, Heide Rohling, Rene Widera, Thomas Ilsche, Matthias Weber, Bert Wesarg, Hartmut Mix, Thomas William, Wolfgang E. Nagel

