

# VampirTrace Cheat Sheet

## Commandline arguments

-vt:help	Show the help message.
-vt:cc <cmd>	Set the underlying compiler command.
-vt:inst <type>	Set the instrumentation type.
compinst	fully-automatic by compiler
manual	manual by using VampirTrace's API
dyninst	binary by using Dyninst (www.dyninst.org)
-vt:opari <args>	Set options for OPARI command.
-vt:<par-type>	Force application's parallelization type.
seq	sequential
mpi	parallel (uses MPI)
mt	arallel (uses OpenMP/POSIX threads)
hyb	hybrid parallel (MPI + Threads)
-vt:verbose	Enable verbose mode.
-vt:show	dry-run, shows call to compiler.

## Compiler Wrappers

environment variables affecting the compiler wrappers:

VT_INST	Instrumentation type (equivalent to -vt:inst)
VT_CC	C compiler command (equivalent to -vt:cc)
VT_CFLAGS	C compiler flags
VT_LDFLAGS	Linker flags
VT_LIBS	Libraries to pass to the linker

## Serial programs

original: gfortran hello.f90 -o hello  
vampir: vt.f90 hello.f90 -o hello

## MPI parallel programs

original: mpicc hello.c -o hello  
vampir: vtcc -vt:cc mpicc hello.c -o hello

## Threaded parallel programs

original: ifort <-openmp|-pthread> hello.f90 -o hi  
vampir: vt.f90 <-openmp|-pthread> hello.f90 -o hi

## Hybrid MPI/Threaded parallel programs

original: mpif90 <-openmp|-pthread> hello.F90  
-o hello  
vampir: vt.f90 -vt:f90 mpif90 <-openmp|-pthread>  
hello.F90 -o hello

## Unification of Local Traces

Set OMP\_NUM\_THREADS if needed

Threaded: vtunify <prefix>

MPI: mpirun -np <nrank> vtunify-mpi <prefix>

## Instrumentation

### Manual Instrumentation

C:	Fortran:
#include "vt_user.h"	#include "vt_user.inc"
VT_USER_START("name");	VT_USER_START('name')
...	...
VT_USER_END("name");	VT_USER_END('name')

Combined with automatic compiler instrumentation:

```
vtcc -DVTRACE hello.c -o hello
```

Without compiler instrumentation:

```
vtcc -vt:inst manual -DVTRACE hello.c -o hello
```

## Tracing Calls to 3rd-Party Libraries

Create a instrumented version of shared library:

```
vtlibwrapgen -g SDL -o wrap.c /.../SDL/*.h  
vtlibwrapgen --build --shared -o libwrap wrap.c  
LD_PRELOAD=$PWD/libwrap.so <executable>
```

## Switching tracing on/off

VT\_ON/VT\_OFF start and stop the recording of events  
check whether tracing is enabled: VT\_IS\_ON

## Trace buffer rewind

Decide dynamically *after* a section wether to record events  
do step=1,number\_of\_time\_steps  
    VT\_SET\_REWIND\_MARK()  
    call compute\_time\_step(step)  
    if(finished\_as\_expected) VT\_REWIND()  
end do

## Intermediate buffer flush

Force flushing of buffers: VT\_BUFFER\_FLUSH

## Intermediate time synchronisation

Flush the buffer at *!synchronized!* point:(eg. barrier): VT\_TIMESYNC Don't forget to export VT\_ETIMESYNC=yes

## Intermediate counter update

For counter values at any given point: VT\_UPDATE\_COUNTER

## User defined counter

Allow recording of program variable values or any other numerical quantity. It is identified by its name, the counter group, the type of its value (int or float) and the unit(e.g. "GFlop/sec").

C:	Fortran:
#include "vt_user.h"	#include "vt_user.inc"
unsigned int id, gid;	integer :: id, gid
gid = VT_COUNT_GROUP_DEF	VT_COUNT_GROUP_DEF
("name");	('name', gid)
id = VT_COUNT_DEF("name",	VT_COUNT_DEF('name',
"unit", type, gid);	'unit', type, gid, id)!

All the uppercase entries start with **VT\_COUNT\_** that has been omitted to fit on the cheatsheet.

**Fortran:**

Type	Count call	Data type
TYPE_INTEGER	INTEGER_VAL	integer (4 byte)
TYPE_INTEGER8	INTEGER8_VAL	integer (8 byte)
TYPE_REAL	REAL_VAL	real
TYPE_DOUBLE	DOUBLE_VAL	double precision

**C/C++:**

Type	Count call	Data type
TYPE_SIGNED	SIGNED_VAL	signed int (max. 64-bit)
TYPE_UNSIGNED	UNSIGNED_VAL	unsigned int (max. 64-bit)
TYPE_FLOAT	FLOAT_VAL	float
TYPE_DOUBLE	DOUBLE_VAL	double

## Environment Variables

	Global Settings	
VT_APPPATH	Path to the application executable.	-
VT_BUFFER_SIZE	Internal event trace buffer.	32M
VT_CLEAN	Remove temporary trace files?	yes
VT_COMPRESSION	Write compressed trace files?	yes
VT_FILE_PREFIX	Prefix used for trace filenames.	
VT_FILE_UNIQUE	Unique naming? no, yes, [id]	no
VT_MAX_FLUSHES	Maximum number of buffer flushes.	1
VT_MAX_THREADS	Threads VT reserves resources for	65536
VT_PFORM_GDIR	Global directory to store final trace.	./
VT_PFORM_LDIR	Node-local temp directory	/tmp/
VT_UNIFY	Unify local trace files afterwards?	yes
VT_VERBOSE	Quiet (0), Critical (1), Information (2)	1
	<b>Optional Features</b>	
VT_CPUIDTRACE	Tracing of core ID of a CPU?	no
VT_ETIMESYNC	Enhanced timer synchronization?	no
VT_ETIMESYNC_INTV	Interval between two phases(seconds).	120
VT_IOLIB_PATHNAME	Library to use for LIBC I/O calls.	-
VT_IOTRACE	Tracing of application I/O calls?	no
VT_LIBCTRACE	Tracing of fork/system/exec calls?	yes
VT_MEMTRACE	Memory allocation counter?	no
VT_MODE	TRACE, STAT, TRACE:STAT (both).	TRACE
VT_MPICHECK	Correctness checking via UniMCI?	no
VT_MPICHECK_ERREXIT	Force trace write on error	no
VT_MPITRACE	Tracing of MPI events?	yes
VT_OMPTRACE	Tracing of OpenMP events	yes
VT_PTHREAD_REUSE	Reuse IDs of terminated Pthreads?	yes
VT_STAT_INV	Interval of profiling records	0
VT_STAT_PROPS	FUNC, MSG, COLLOP, ALL	ALL
VT_SYNC_FLUSH	Synchronized buffer flush?	no
VT_SYNC_FLUSH_LEVEL	Minimum buffer fill level in percent.	80
	<b>Counters</b>	
VT_METRICS	list of counter.	-
VT_METRICS_SEP	Separator string in VT_METRICS.	:
VT_RUSAGE	Colon-separated list of rusage counters-	
VT_RUSAGE_INTV	Sample interval in ms.	100
	<b>Filtering, Grouping</b>	
VT_DYN_BLACKLIST	Blacklist file for Dyninst	-
VT_DYN_SHLIBS	Colon-separated list of shared libraries-	
VT_FILTER_SPEC	Name of function/region filter file.	-
VT_GROUPS_SPEC	Name of function grouping file.	-
VT_JAVA_FILTER_SPEC	Name of Java specific filter file.	-
VT_GROUP_CLASSES	Create Java class group automatically?	yes
VT_MAX_STACK_DEPTH	Maximum stack level (0 = unlimited)	0
	<b>Symbol List</b>	
VT_GNU_NM	Command for symbols of object files	nm
VT_GNU_NMFIL	File with symbol list information.	-

## Synchronized Buffer Flush

Have Vampir flush buffers when 1 buffer reaches VT\_SYNC\_FLUSH\_LEVEL at the next MPI collective that is using MPI\_COMM\_WORLD.  
export VT\_SYNC\_FLUSH=yes

## Enhanced Timer Synchronization

This scheme inserts additional synchronization phases at appropriate points in the program flow. Currently, VampirTrace makes use of all MPI collective functions associated with MPI\_COMM\_WORLD. A LAPACK library with C wrapper support has to be provided.

```
export VT_ETIMESYNC=yes
```

Additionally, set the interval in seconds between the synchronization phases.

```
export VT_ETIMESYNC_INTV=600
```

LAPACK libraries providing a C-LAPACK API that can be used by

VampirTrace:

- CLAPACK ([www.netlib.org/clapack](http://www.netlib.org/clapack))

- AMD ACML

- IBM ESSL

- Intel MKL

- SUN Performance Library

Note though that this is not needed for systems with a global timer.

## Hardware Performance Counters

VampirTrace supports different hardware counter libraries and relies on those API's to gather the counter values. An exclamation mark "!" marks a counter to be measured as an absolute value.

### PAPI Hardware Performance Counters

VT\_METRICS=PAPI\_FP\_OPS:PAPI\_L2\_TCM:!CPU\_TEMP1  
CPU\_TEMP1 is provided by the lm-sensors component. See papi\_avail and papi\_native\_avail for available counter.

```
PAPI_L[1|2|3]_[D|I|T]C[M|H|A|R|W]
    Level 1/2/3 data/instruction/total cache
    misses/hits/accesses/reads/writes
PAPI_L[1|2|3]_[LD|ST|M]
    Level 1/2/3 load/store misses
PAPI_CA_SNP
    Requests for a snoop
PAPI_CA_SHR
    Req. for excl. access to shared cache line
PAPI_CA_CLN
    Req. for excl. access to clean cache line
PAPI_CA_INV
    Requests for cache line invalidation
PAPI_CA_ITV
    Requests for cache line intervention
PAPI_BRU_IDL
    Cycles branch units are idle
PAPI_FXU_IDL
    Cycles integer units are idle
PAPI_FPU_IDL
    Cycles floating point units are idle
PAPI_LSU_IDL
    Cycles load/store units are idle
PAPI_TLB_DM
    Data translation lookaside buffer misses
PAPI_TLB_IM
    Instruction transl. lookaside buffer misses
PAPI_TLB_TL
    Total translation lookaside buffer misses
PAPI_BTAC_M
    Branch target address cache misses
PAPI_PRF_DM
    Data prefetch cache misses
PAPI_TLB_SD
    Translation lookaside buffer shutdowns
PAPI_CSR_FAL
    Failed store conditional instructions
PAPI_CSR_SUC
    Successful store conditional instructions
PAPI_CSR_TOT
    Total store conditional instructions
PAPI_MEM_SCY
    Cycles Stalled Waiting for memory accesses
PAPI_MEM_RCY
    Cycles Stalled Waiting for memory Reads
PAPI_MEM_WCY
    Cycles Stalled Waiting for memory writes
PAPI_STL_ICY
    Cycles with no instruction issue
PAPI_FUL_ICY
    Cycles with maximum instruction issue
PAPI_STL_CCY
    Cycles with no instructions completed
PAPI_FUL_CCY
    Cycles with maximum instructions completed
PAPI_BR_UCN
    Unconditional branch instructions
PAPI_BR_CN
    Conditional branch instructions
PAPI_BR_TKN
    Conditional branch instructions taken
PAPI_BR_NTK
    Conditional branch instructions not taken
```

```
PAPI_BR_MSP
    Conditional branch inst. mispredicted
PAPI_BR_PRC
    Cond. branch inst. correctly predicted
PAPI_FMA_INS
    FMA instructions completed
PAPI_TOT_IIS
    Instructions issued
PAPI_TOT_INS
    Instructions completed
PAPI_INT_INS
    Integer instructions
PAPI_FP_INS
    Floating point instructions
PAPI_LD_INS
    Load instructions
PAPI_SR_INS
    Store instructions
PAPI_BR_INS
    Branch instructions
PAPI_VEC_INS
    Vector/SIMD instructions
PAPI_LST_INS
    Load/store instructions completed
PAPI_SYC_INS
    Synchronization instructions completed
PAPI_FML_INS
    Floating point multiply instructions
PAPI_FAD_INS
    Floating point add instructions
PAPI_FDV_INS
    Floating point divide instructions
PAPI_FSQ_INS
    Floating point square root instructions
PAPI_FNV_INS
    Floating point inverse instructions
PAPI_RES_STL
    Cycles stalled on any resource
PAPI_FP_STAL
    Cycles the FP unit(s) are stalled
PAPI_FP_OPS
    Floating point operations
PAPI_TOT_CYC
    Total cycles
PAPI_HW_INT
    Hardware interrupts
```

### CPC Hardware Performance Counters

Sun Solaris features the CPC performance counter library to query the hw-counter. See vtcpccavail for your options.

### NEC SX Hardware Performance Counters

NEC SX machines uses special register calls to query the processor's hardware counters. See NEC SX manual.

```
SX_CTR_STM
    System timer reg
SX_CTR_USRCC
    User clock counter
SX_CTR_EX
    Execution counter
SX_CTR_VX
    Vector execution counter
SX_CTR_VE
    Vector element counter
SX_CTR_VECC
    Vector execution clock counter
SX_CTR_VAREC
    Vector arithmetic execution clock counter
SX_CTR_VLDEC
    Vector load execution clock counter
SX_CTR_FPEC
    Floating point data execution counter
SX_CTR_BCCC
    Bank conflict clock counter
SX_CTR_ICMCC
    Instruction cache miss clock counter
SX_CTR_OCMCC
    Operand cache miss clock counter
SX_CTR_IPHCC
    Instruction pipeline hold clock counter
SX_CTR_MNCCC
    Memory network conflict clock counter
SX_CTR_SRACC
    Shared resource access clock counter
SX_CTR_BREC
    Branch execution counter
SX_CTR_BPFC
    Branch prediction failure counter
```

## Resource Usage Counters

The Unix system call getrusage provides information about consumed resources and operating system events.

```
VT_RUSAGE=ru_stime:ru_majflt
```

The resource usage counters are not recorded at every event. Set intervall in ms:

```
export VT_RUSAGE_INTV=100 (most OS support min. of 10ms )
```

Counters are per process, not per thread.

Name	Unit	Linux	Description
ru_utime	ms	x	Total amount of user time used.
ru_stime	ms	x	Total amount of system time used.
ru_maxrss	kB		Maximum resident set size.
ru_ixrss	kB/s		Integral shared memory size (text segment).
ru_idrss	kB/s		Integral data segment memory used over runtime.
ru_isrss	kB/s		Integral stack memory used over the runtime.
ru_minflt	#	x	Number of soft page faults.
ru_majflt	#	x	Number of hard page faults.
ru_nswap	#		# times process was swapped out of phys. mem.
ru_inblock	#		Number of input operations via the file system.
ru_oublock	#		Number of output operations via the file system.
ru_msgsnd	#		Number of IPC messages sent.
ru_msgrcv	#		Number of IPC messages received.
ru_nsignals	#		Number of signals delivered.
ru_nvcsw	#	x	Number of voluntary context switches.
ru_nivcsw	#	x	Number of involuntary context switches.

## Function Filtering

To decrease trace-size specify filter directives before running.

```
# syntax: <regions> -- <limit>
#   regions semicolon-separated list of regions
#           (can be wildcards)
#   limit   assigned call limit
#           0 = region(s) denied
#           -1 = unlimited
add;sub;mul;div -- 1000
* -- 3000000
```

To activate set environment variable accordingly:

```
export VT_FILTER_SPEC=/path/to/filter.txt
```

Rank specific filtering is also possible:

```
@ 4 - 10, 20 - 29, 34
```

```
foo;bar -- 2000
```

```
* -- 0
```

## Function Grouping

All functions of a group are assigned one color in visualization.

Group name	Contained functions/regions
MPI	MPI functions
OMP	OpenMP API function calls
OMP_SYNC	OpenMP barriers
OMP_PREG	OpenMP parallel regions
Pthreads	Pthread API function calls
MEM	Memory allocation functions
I/O	I/O functions
LIBC	LIBC fork/system/exec functions )
Application	all the rest

Define you own groups in a text-file.

```
# syntax: <group>=<regions>
#   group   group name
#   regions semicolon-separated list of regions
#           (can be wildcards)
```

```
CALC=add;sub;mul;div
```

```
USER=app_*
```

To activate set environment variable accordingly:

```
export VT_GROUPS_SPEC=/path/to/groups.txt
```