

ATPESC 2016

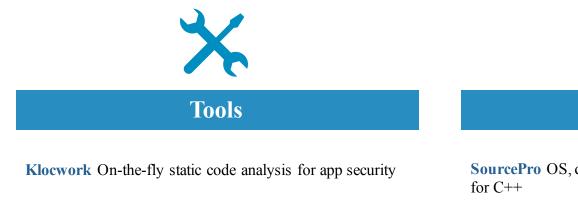
TotalView: Debugging from Desktop to Supercomputer

Peter Thompson Principal Software Support Engineer August 10, 2016



Accelerating Great Code

Our products and services



CodeDynamics Commercial dynamic analysis

OpenLogic Support Enterprise-grade SLA support

OpenLogic Audits Detailed open source license and security risk guidance

TotalView for HPC Scalable debugging

Zend Server Enterprise PHP app server

Zend Studio PHP IDE

Zend Guard PHP encoding and obfuscation

SourcePro OS, database, network, and analysis abstraction for C++

Libraries

Visualization Real-time data visualization at scale

PV-WAVE Visual data analysis

IMSL Numerical Libraries Scalable math and statistics algorithms

HydraExpress SOA/C++ modernization framework

HostAccess Terminal emulation for Windows

Stingray MFC GUI components



How does Rogue Wave help in HPC?

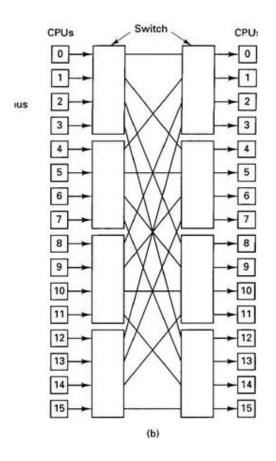
TotalView for HPC

- Source code debugger for C/C++/Fortran
 - Visibility into applications
 - Control over applications
- Scalability
- Usability
- Support for HPC platforms and languages



TotalView Overview

TotalView Origins



Mid-1980's Bolt, Berenak, and Newman (BBN) Butterfly Machine An early 'Massively Parallel' computer

*E*RogueWave

How do you debug a Butterfly?

- TotalView project was developed as a solution for this environment
 - Able to debug multiple processes and threads
 - Point and click interface
 - Multiple and Mixed Language Support
- Core development group has been there from the beginning and have been/are involved in defining MPI interfaces, DWARF, and lately OMPD (Open MP debugging interface)

Other capabilities added

- Support for most types of MPI
- Linux
- Lightweight Memory Debugging
- Type transformations STL and user containers
- Memscript and tvscript
- Reverse Debugging only on Linux x86-64
- Remote Display Client
- GPU debugging
- Intel Xeon Phi Including KNL
- Currently looking at ARM64 port

Key features of TotalView

- Interactive Debugging
- Interactive Memory Debugging
- Reverse Debugging
- Unattended Debugging

Serial, Parallel and Accelerated applications



Memory Analysis

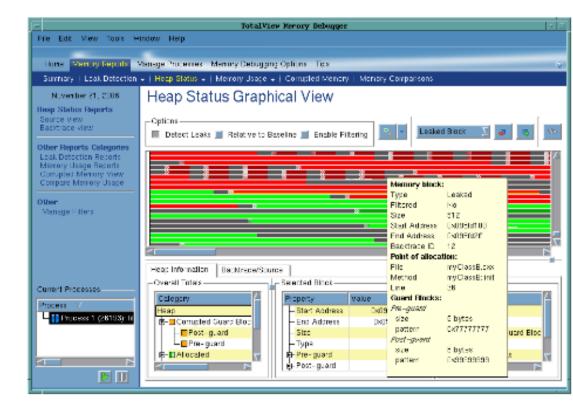
How do you identify buffer overflows?

Runtime Memory Analysis : Eliminate Memory Errors

- Detects memory leaks *before* they are a problem
- Explore heap memory usage

Features

- Detects
 - Malloc API misuse
 - Memory leaks
 - Buffer overflows
- Low runtime overhead
- Easy to use
 - Works with vendor libraries
 - No recompilation
 - No instrumentation





Reverse debugging

- How do you isolate an intermittent failure?
 - Without TotalView,
 - Set a breakpoint in code
 - Realize you ran past the problem
 - Re-load
 - Set breakpoint earlier
 - Hope it fails
 - Keep repeating
 - With TotalView
 - Start recording
 - Set a breakpoint
 - See failure
 - Run backwards/forwards in context of failing execution
 - Reverse Debugging

FRogueWave

- Re-creates the context when going backwards
- Focus down to a specific problem area easily
- Saves days in recreating a failure

Group (Control) 🔅	► II ■		1 1 E ReplayEngine I I E I I I S
Processes & Threads	× P #T ▼ 1 1 1	Members p1 p1 p1.1 p1.1	Image: Start Page x combined.cxx x 23 int do_parallel = 1; 25 int main(int argc, char **argv) 27 { 29 str = (char *) malloc(100); 30 strcpy(str, "Hello World"); 31 // General features 33 arrays(); 34 diveinal(); 35 printf("%s\n", str); 36 long arr_longs[7] = {1,2,3,4,5,16,32};
			<pre>39 pthreads_loop(); 40 41 // C++ features 42 derived_class(); 43 stl_viev(); 44 user_templates(); 45 trap_exception(); 46 47 exit(1); 48 } 50</pre>

memscript and tvscript

- Command line invocation to run TotalView and Memoryscape unattended
- tvscript can be used to set breakpoints, take actions at those breakpoints and have the results logged to a file. It can also do memory debugging
 - tvscript -create_actionpoint "method1=>display_backtrace show_arguments" \ create_actionpoint "method.c#342=>print x" myprog -a dataset 1
- memscript can be used to run memory debugging on processes and display data when a memory event takes place. Exit is ALWAYS an event

Memscrip -event_action \setminus

"alloc_null=list_allocations,any_event=check_guard_blocks" $\$

-guard_blocks -maxruntime "00:30:00" -display_specifiers $\$

"noshow_pc,noshow_block_address,show_image"

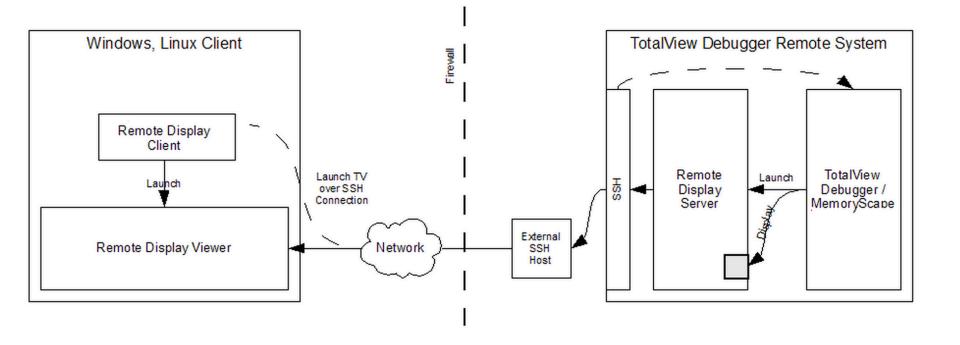
myProgram -a myProgramArg1

• Memscript data can be saved in html, memory debug file, text heap status file

Remote Display Client (RDC)

• Push X11 bits and events across wide networks can be painful. The RDC can help







The RDC setup

		RogueŴave		
Session Profiles:	1. Enter the Remote Host to run your del	oug session:		
8 8 8	Remote Host: vesta.alcf.anl.gov	User Name 🗘	: thompson	Advanced Options
perseid vesta	2. As needed, enter hosts in access orde	r to reach the Remote Host:		
	Host	Access By	Access Value	Commands
		User Name 🗘		
	2	User Name 🗘		
	3. Enter settings for the debug session o			
		TotalView MemoryS	cape	
	Path to TotalView on Remote Host:	(a sfel da burgana (basala inu (bia (
	Arguments for TotalView:	/soft/debuggers/totaiview/bin/t	totaiview	
	Your Executable (path & name):	runiob		
	Arguments for Your Executable:	-	LT_PARTNAME} : ALLc2	
	Submit Job to Batch Queueing System:	Custom		\$
	4. Enter batch submission settings fo	w the Domote Liest .		
	4. Enter batch submission settings fo Submit Command:			
	Script to execute via Submit Command:			
			mode script 0100	
	Additional Submit Command Options:	-4 ATPESC2015 -1 00 -1 512	-mode script -0 LOG	
		Launch Debug Ses	sion	



TotalView for the NVIDIA[®] GPU Accelerator

배 💮 saxpy_cuda			\odot \otimes \otimes
File Edit View Group Process Thread Action P	oint Debug	Tools Window	Help
Group (Control)	J 🗊 🀐	• • • •	
Physical Device: 0 2 SM: 0 2 Warp: 0 2 L	ane: 0		
Process 1 (20343): saxp			
Stack Trace	_UDH@.saxpy_i	Stack Frame	
C++ saxpy_parallel, FP=fffca0	Device: SM/WP/L n: a: x: y: Block "\$b i:	N: 0/0/0 of 1- 0x00040000 2 0x20020000 0x20030000 1": 0x00001000	4/48/32 (262144) 0 -> 0 0 -> -1
	Registers	for the frame:	¥
Function saxpy_paralle	l in saxpy₊c	u	
<pre>26 printf("Test FAILED\n"); \ 27 exit(-1); } 28 29 30 // GPU kernel function</pre>	File Edit Vie	ew Tools Window	Help
31] 11		J 👂 🔓 🔣 🖌
33 unsigned int i = blockIdx.x*blockDim.x +	Expression: A	Address:	0×00000010
35 if $(i < n)$ {	Type: @p	parameter const Matrix	
<pre>y[i] = a*x[i] + y[i]; 37 }</pre>	Field	Туре	Value 🚺
38 } 39 40 41 41 42 int main(int argc, char* argv[]) 43 { 44 // For time measurements	- width - height - stride - elements	int int float @global *	Dx0000003c (60) Dx00000014 (20) Dx0000003c (60) Dx00110000 -> 0
45 double tiStart1, tiStop1, tiStart2, tiStop2 46 double timeCPU. timeGPU. timeGPU kernel:			
Action Points Processes Threads			P- P+ T- T+
1 saxpy.cu#36 saxpy_parallel+0x88			

- NVIDIA CUDA 6.5, 7.0, 7.5, (8.0)
- Features and capabilities include
 - Support for dynamic parallelism
 - Support for MPI based clusters and multi-card configurations
 - Flexible Display and Navigation on the CUDA device
 - Physical (device, SM, Warp, Lane)
 - Logical (Grid, Block) tuples
 - CUDA device window reveals what is running where
 - Support for CUDA Core debugging
 - Leverages CUDA memcheck
 - Support for OpenACC



TotalView for the Intel[®] Xeon Phi[™] coprocessor

Supports All Major Intel Xeon Phi Coprocessor Configurations

- Native Mode
 - With or without MPI
- Offload Directives
 - Incremental adoption, similar to GPU
- Symmetric Mode
 - Host and Coprocessor
- Multi-device, Multi-node
- Clusters
- KNL Support Just works like a normal node
 - AVX2 support being added
- User Interface
 - MPI Debugging Features
 - Process Control, View Across, Shared Breakpoints
 - Heterogeneous Debugging
 - Debug Both Xeon and Intel Xeon Phi Processes

Memory Debugging

• Both native and symmetric mode

File Edit	View Too <u>l</u> s Wind	ow		Help	
ID 🛆 Ran	k Host	Status	Description	1 I	
	<local> R <local> B</local></local>		intel/composer	xe/Sample	
1.2	<local> R <local> R</local></local>	in ma in _			
1. 3	<local> R</local>	in _			
1.4	<local> R</local>		hread_cond_wai		
÷ 2.1	192.168.1.1(M		'coi_procs/1/58	56/offloa	
2.2	192.168.1.1(R 192.168.1.1(B		m_wait mpute07		
	192.168.1.1(R	in			
2.4	192.168.1.1(R	in pt	hread_cond_wai	t	
le <u>E</u> dit <u>V</u> iew <u>G</u> roup <u>F</u>	Process Thread Action	Point Debug Too	1s Window		Help
oup (Control)		sa 🕸 🦇 🗌			
	Halt Kill Restart Next :				Live
	Process 2 (58560192.1 Thread 2 (1399858	168.1.100): offlo	ad_main (Mixed)		
Stack		ці (не ы.	Stack Fra	ne	
] compute07,] L sample07 76 par re	FP=7f50fd4d24f0 gion1_2_39, FP=7f50fd4d2	Function "com out:	pute07":	754 -> 0x41400000 (:	109
offload_entry_sample	eC07_c_76sample07, FP=7f r7offloadE_PPvS0_tS0_t,	size: Local variabl	0x00000010	(16)	
COISinkPipe::RunFunc	tion, FP=7f50fd4d2dc0 Messages, FP=7f50fd4d2dc1	1:	0x00000010	(16)	
COISinkPipe::ThreadP	roc EP=7f50Ed/d2e20	Registers for	the frame:		
] start_thread, clone,	FP=7f50fd4d2f30 FP=7f50fd4d2f38	Xrax: 0	x7f50fd4d2754 (1399	85823803220>	
		Zedy: 0	x00000010 (16) x7f50fd4d2754 (1399		
	Punction annuuta	07 in sampleCO7.c		050070004403	_ <u>7</u>
0 for (1=0; 1 <s; 1<="" td=""><td></td><td>or in samplecor.c</td><td></td><td></td><td></td></s;>		or in samplecor.c			
1 { 2 array1[i] = ;					
11 { 12 array1[i] = ; 13 }	-6239				
5 #ifdefMIC					
7 =else					
8 retval = 0; 9 #endif					
0	rray initialization was				
2 return retval;	rag inicialización was	done on carged			
95attribute((targe) 96 {	t(mic))) void compute07((int* out, int siz	e)		
7 int i; 8					
<pre>for (i=0; i<size, 0="" pre="" {<=""></size,></pre>	; <u>i</u> ++)				
1 out[i] = arrs	ay1[i]*2;				- 11
3 3		•••••			7
3 3 4 //					
4 //					12
4 //	Th <u>r</u> eads			P- P+ T =	
4 // ion Points Processes ((139985896167360) R	in sem_wait			<u>P-</u> <u>P+</u> T-	
4 // ion Points Processes ((139985896167360) R (139985834444544) R	in sem_wait in compute07 inpoll			<u>P-</u> <u>P</u> + T -	
(139985896167360) R	in sem_wait in compute07	3		<u>p-</u> <u>p+</u> <u>T-</u>	
ion Points Processes ((139985896167360) R (18958382807822) MG (13998583444544) R	in sem_wait in compute07 inpoll			<u>p-</u> <u>p+</u> <u>T</u> =	
<pre>4 // ion Points Processes (139985896167360) R (139985834444544) R (1398583444544)</pre>	in sem_wait in compute07 inpoll			<u>P-</u> <u>P</u> + <u>T</u> -	

Knights Landing Memory

- KNL will have on-board High Bandwidth Memory (HBM) which can be accessed much faster than going out to main memory.
 - Cache
 - Explicitly managed for placement of frequently accessed data
- MemoryScape will be able to track allocations made both the standard heap and the on-chip HBM
- Optimization may include making sure that the right data structures are available to the processor in HBM
 - MemoryScape can show you data structure usage and placement
- KNL machines starting to come online

Linux OpenPower (LE) support

- Support for OpenPower (Linux power LE)
 - All major functionality
 - Support for CUDA Debugging on GPU Accelerators



TotalView's Memory Efficiency

- TotalView is lightweight in the back-end (server)
- Servers don't "steal" memory from the application
- Each server is a multi-process debugger agent
 - One server can debug thousands of processes
 - Not a conglomeration of single process debuggers
 - TotalView's architecture provides flexibility (e.g., P/SVR)
 - No artificial limits to accommodate the debugger (e.g., BG/Q 1-P/CN)
- Symbols are read, stored, and <u>shared</u> in the front-end (client)
- Example: LLNL APP ADB, 920 shlibs, Linux, 64 P, 4 CN, 16 P/CN, 1 SVR/CN

Process	VSZ (largest, MB)	RSS (largest, MB)
TV Client	4,469	3,998
MRNet CP	497	4
TV Server	304	53



New UI Framework – aka CodeDynamics

						Rogue Wave						
Developer file	Window											
Group (Control)	: •	II 🖷 🕨 🛆 ð	9 13 *									
ocesses and Thre	arts X		StartPag	tx fork loop.cxx	X ty fork loop cyy X	select ×			Call S	itack #		
New1 View2			654	te - te-turk-seehoren	to the party team							
			665	if (do_segv && ne	== do_segv_index)			1	1	_select		
escription	¢₽	#T ♥ Members	666 667	{ struct timeval					G	wait_a_wi	hile	
Breakpoint	4	4 pl-4	668 669	<pre>long bad_addr; int *feo;</pre>					œ	snore		
□select	4	4 p2-3.1, p1.2,	670 671	int bar;	Phimas white					forker		
- 1.2	1	1 p1.2	672	<pre>wait_a_while { bad_addr = -3;</pre>					600	TOTKEF		
2.1	1	1 p2.1	673 674	foo = (int *)b bar = *foo;	ad_addr;				œ	fork_wrap	oper	
- 3.1	1	1 p3.1	675 676	*foo = bar + 1	a				œ	main		
4.3	1	1 p4.3	677	1						libc sta	rt main	
- wait a while	e 4	8 pl.1, p4.1, p2	678 679	for (::)					⊪—			
1.1	1	1 p1.1	680 681	struct timeval						_start		
- 1.3	1	1 pl.3	682	if (verbose)								
	1		683 684	printf ("Thr if (use mut)	ead %ld woke up in Sr	nore()\n*, (long)(pthr	<pre>mead_self()));</pre>		WAR	0		
2.2		1 p2.2	685 686	{ if (!pleas	a chut unl				Name	e	Туре	Value
- 2.3	1	1 p2.3	687	(▼ Arg	uments		
3.2	1	1 p3.2	688 689		65 !defined(_linux) ("Thread %d (posix =) = %ld): Trying for the	lock\n",		a	rg	void *	0x00000000
- 3.3	1	1 p3.3	690 691		<pre>int(pthread_t(pthre long(pthread_self())</pre>	end_self()]->sequence	e),		▼ Blo	ck		\$b1#\$b2
4.1	1	1 p4.1	692	Welif_AIX					1	meout	struct timeval	[struct timeval]
	_		693 694	#else	("Thread (ktid) %d:	Trying for the lock\n	<pre>", (int)(thread_self()));</pre>		▼ Blo	ck		\$b1
	bread attri	ibutes to group by:	695 696	printf	("Thread (posix) ID- long(pthread self()	-Ald: Trying for the l	ock\n",		1000	ne	int	0×00000000 (0)
Share Group			697	#endif								
Hostname			698	} /* if fflush (st	dout);					id_ticket	int	0xmmr (-1)
Process State			700	othread out	tex lock (Amutex):				- 10	icket	int	0x00000000 (0)
Thread State												
			Action Po	oints *				Command Li	-			
PC			ID♥	Туре	file		Line	Thread 1.2 val*)*	hit bre	sakpoint 1	at line 564 i	n "wait_a_while(tin
Source Line			1	com.roguewave.totalvie	w.breakpoint tx_fork_lo	op.cxx 564		Thread 3.2	hit bre	akpoint 1	at line 564 i	n "wait_a_while(tim
Function									hit bro	akpoint 1	at line 564 i	n "wait_a_while(tim
Action Point ID								val*)" Thread 2.3	hit bre	akpoint 1	at line 564 i	n 'wait a while(tim
Replay Mode								val*)*				n "wait a while(tim
MOVE UP	RESET							val*)*	nit pre	empoint I	at time 504 1	wart_g_wurte(flu
ocess: 2 (Stopper		ad: 2.1 (Stopped) Fran	ne: snore				EV WJB TVPDA/debugger/src/te			ine: 681		Line: 68

Using TotalView

For HPC we have two methods to start the debugger

The 'classic' method

- totalview –args mpiexec –np 512 ./myMPIprog myarg1 myarg2
- This will start up TotalView on the parallel starter (mpiexec, srun, runjob, etc) and when you hit 'Go' the job will start up and the processes will be automatically attached. At that point you will see your source and can set breakpoints.
- Some points to consider...
 - You don't see your source at first, since we're 'debugging' the mpi starter
 - Some MPI's don't support the process acquistion method (most do, but might be stripped of symbols we need when packaging)
 - In general more scalable than the next method...

Starting TotalView

The 'indirect' method

- Simply 'totalview' or 'totalview myMPIprog' and then you can choose a parallel system, number of tasks, nodes, and arguments to the program.
- With this method the program source is available immediately
- Less dependent on MPI starter symbols
- May not be as scalable as some 'indirect' methods launch a debug server per process

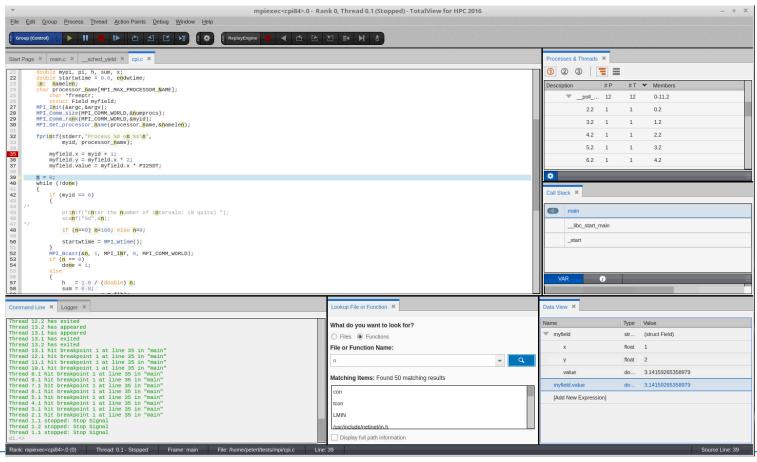
	X TotalView for HPC: Parallel Program Session	000
		PARALLEL DETAILS
7	Operation Mensory IE stars and a transfer of a star stars with E V .	PROGRAM DETAILS
	S Parallel System	
QUIRED	NMENT Name: BlueGeneQ-Cobalt 7	
	Tasks (np): [Enter the number of tasks]	
	Additional [Enter starter arguments as needed] Starter Arguments:	
incel		When you are ready, press Next to continue.
u	Parallel Settings Tasks (np): [Enter the number of tasks] Additional Starter Arguments: e ready,	When you are ready, press Next to



The New UI for HPC

MPI debugging with the new UI requires starting in 'classic' mode with the – newUI argument

totalview -newUI -args mpiexec -np 4 ./cpi



Rogue Wave

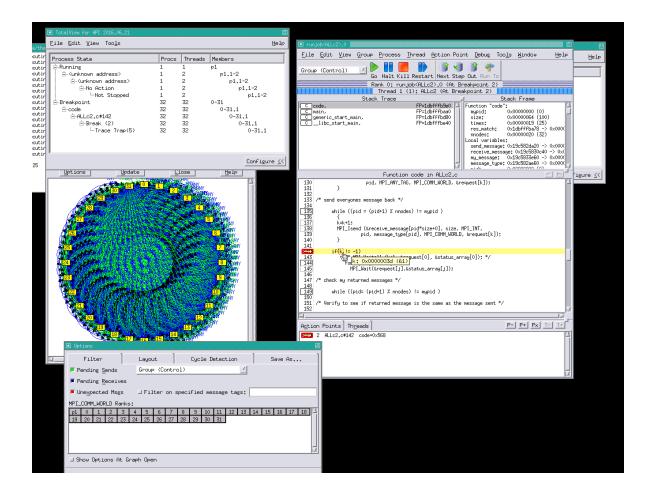
Starting MemoryScape and ReplayEngine

- MemoryScape must be enabled at the start of the program or your program can be linked against the Heap Interposition Agent (HIA) to have memory debugging always enabled. It can't be turned on when the program has been started.
- ReplayEngine can be enabled at the start, or at any point during debugging. Once started, it can't be disabled.

- Y	Startup Parameters - Is	×
Debu	gging Options Arguments Standard I/O Para <u>l</u> lel	
	nable <u>ReplayEngine</u> ecord all program state while running. Roll back your program to any point in the past.	.
	nable memory debugging rack dynamic memory allocations. Catch common errors, leaks, and show reports.	
	Halt on memory errors	
	nable CUDA memory checking etect global memory addressing violations and misaligned global memory accesses.	
<u> </u>	ow Startup Parameters when TotalView starts	
	Changes take effect at process startup.	
	OK Cancel Help	



Running TotalView on Vesta



Rogue Wave

Memory Debugging on BG/Q

- In order to use memory debugging on the BG/Q, you must link against the agent. For instance, I first set up a environment variable
 - export TVLIB=/soft/debugger/totalview/linux-power/lib
 - mpixlc_r -g -o ALLc2.mem ALLc2.c -LTVLIB
 - –Wl,@\$TVLIB/tvheap_bgqs.ld

es	ses Memory Debugging Options Tips
	Memory Debugging Options
	Select your preferred level of debugging below or press Advanced Options for more control.
	Enable memory debugging
_	Levels of Debugging
	📕 Low
	Provides event notifications and leak detection. It allows the best perfomance for your process.
	Medium
	Adds corrupted memory detection by applying guard blocks. Performance may degrade slightly.
	📕 High
	Provides memory over run alerts by monitoring Red Zone violations. Your memory consumption will increase significantly.
	📕 Extreme
	Enables all options. There is a risk that performance may suffer and you will use more memory.



One of many reports available

Home Memory Reports Manage Proces	ses Memory Debugging Option	s Tips		
Gummary Leak Detection - Heap St			Mnarisons	
August 10, 2016	Heap Status Backtrace			
	- Data Source	Report	Options	
ave Data Save Report	Baca Source		operana	
xport Memory Data	🔷 Allocations 💠 Deallo	ocations 🔷 Hoard 🔷 Re	ed Zones 📗 🔳 Relative to Baseline 📕	Enable Filtering
eap Status Reports	Process	Total Bytes C	ount Function	Line # Source Information
raphical Report	±-213	576	24	
ource Report	<u>∔</u> -1179	840	21	
has Deserts Catagoritan	. 1126	400	10	
h er Reports Categories eak Detection Reports	÷- 190	80.00KB	10	
emory Usage Reports		80.00KB	10	
orrupted Memory Report	±- 184	1280,00KB	10	
ompare Memory Úsage		640.00KB	10	
ther Tasks	<u>⊨</u> -1176	178.00KB	8	
Manage Filters			wrap_posix_memalign	60 tv_heap_breakpoint.c
			PAMI::Memory::HeapMemoryManager:	119 HeapMemoryManager.h
			PAMI::MemoryAllocator<5680,16,4,	. 167 MemoryAllocator.h
			PAMI::Protocol::Send::Eager <pami.< td=""><td> 111 MemoryAllocator.h</td></pami.<>	111 MemoryAllocator.h
			PAMI_Send	380 Dispatch.h
			MPIDI_SendMsg_eager	129 mpidi_sendmsg.c
			MPIDI_Isend_handoff	462 mpidi_sendmsg.c
Process Selection			PMPI Isend	146 mpidi_macros.h
Process V			MPI Isend	102 mpiwrappers_c.c
- Prunjob <allc2.mem>.10</allc2.mem>			code	115 ALLc2.c
			main	225 ALLc2.c
- Prunjob <allc2.mem>.11</allc2.mem>			generic_start_main	226 libe-start.c
- Prunjob <allc2.mem>.12</allc2.mem>			libc_start_main	194 libc-start.c
👂 runjob <allc2.mem>.13</allc2.mem>				104 1100 3001 040
- 🦻 runjob <allc2.mem>.14</allc2.mem>	Source			
- 👂 runjob <allc2.mem>.15</allc2.mem>			/soft/debuggers/totalview-2016-06	-21/toolworks/totalview.2016.06.21/src/to
- Prunjob <allc2.mem>.16</allc2.mem>	57 */			
Prun, job (ALLc2, mem), 17	58			
<pre>/ run.job<allc2.mem .18<="" pre=""></allc2.mem></pre>	59 TV HEAP event = *eve			
	I IMM I V HEHP event = Xeve	nti		

Rogue Wave

More Information

• Product documentation on website:

http://www.roguewave.com/help-support/documentation/totalview

• Contact <u>sales@roguewave.com</u> with any inquires about our future plans with regard to TotalView product.



Thanks!

- Visit the website
 - <u>http://www.roguewave.com/products/totalview.aspx</u>
 - Documentation
 - Sign up for an evaluation
 - Contact customer support & post on the user forum