Third Party Tools for Titan

Richard Graham, Oscar Hernandez, Christos Kartsaklis, Joshua Ladd, Jens Domke (Oak Ridge National Laboratory)

Jean-Charles Vasnier, Stephane Bihan, and Georges-Emmanuel Moulard (CAPS Enterprise)

Presented by: Joshua S. Ladd





Goals

- Provide OLCF-3 users with high productivity programming languages & compilers.
- Provide a programming environment with tools to support the porting of codes.
- Work with vendors to provide compiler, performance, and debugger capabilities needed to port applications with GPUs :
 - CAPS enterprise (OpenHMPP Directives)
 - The Portland Group (Accelerator Directives)
 - Cray (OpenMP for Accelerators)
 - NVIDIA
 - TU-Dresden (Vampir)
 - Allinea (DDT)
- Joint Standardization Efforts: OpenACC, OpenMP ARB



Improve Productivity and Portability

- The Directive based approach provides:
 - Incremental porting/development
 - Fast Prototyping
 - The programmer can quickly produce code that runs in the accelerator
 - Increases Productivity
 - Few code modifications to produce accelerated code
 - Retargetable to different architectures (CPU, GPUs, FPGAs)
 - Tools can assist the user generate the directives, debug them, and do performance analysis
- Leading technologies with accelerator directives:
 - OpenACC directives (Cray, PGI, CAPS, NVIDIA)
 - CAPS OpenHMPP directives
 - PGI accelerator directives



Compilers Available for OLCF-3

Compiler Vendor	C/C++	Fortran	CUDA C / OpenCL	CUDA Fortran	OpenHMPP	PGI Acc Dir	OpenACC	OpenMP CPU
Cray	Х	Х					Х	Х
PGI	Х	Х	Р	Х		Х		Х
CAPS HMPP	Х	Х			Х		Х	Х
NVIDIA			Х					
Pathscale	Х	Х			Р		Р	Х
Intel	Х	Х						Х
GNU	Х	Х						Х

X = Supported P = In Progress

• Cray, CAPS, and NVIDIA are directly involved with the OLCF-3 Effort



Current Work

- We are currently working with Vendors to provide a set of tools that target the application needs for OLCF-3
- We are also building a tool environment to support the applications:



Programming Models

- System supports: C/C++/Fortran, OpenMP, MPI, SHMEM
- GPU accelerator directives: OpenACC and OpenHMPP directives
- We identified features needed for programmability, performance improvements or bugs.
 - C++ support [Feature]
 - Fortran module support [Feature]
 - Inlining support [Feature]
 - Need to allocate data directly in the GPU [Performance]
 - 3D scheduling support for thread blocks [Performance]
 - Support for libraries [Feature]
 - Codelet functions [Feature]
 - Worksharing between devices in nodes. [Feature]



HMPP Overview

- C/Fortran OpenHMPP and OpenACC directive-based programming model (API-based for C++)
 - Offload functions and regions onto accelerators
 - Manage CPU-GPU data movements
 - Finely tune loop optimizations and use of hardware specific features
 - Complementary to OpenMP and MPI
- A source-to-source HMPP compiler
 - Generates CUDA/OpenCL/MiC kernels
 - Works with standard compilers
- A runtime library
 - Dispatch computations on available GPUs
 - Scale to multi-GPU systems



HMPP Enhanced Features: Driven by OLCF Application Needs

- MADNESS:
 - API for C++ applications
- LAMMPS:
 - Multi-GPU and CPU data distribution
- LSMS:
 - Seamless integration of accelerated libraries



HMPP for C++ Applications

 Runtime API close to OpenCL with one directive to let HMPP generate the CUDA version of a function

- Templates not supported by OpenACC
- MADNESS heavily dependent on templates

9 Managed by UT-Battelle for the U.S. Department of Energy



HMPP Data Distribution

- Directives for C and Fortran
 - Define a multi-GPU data distribution scheme and let HMPP execute over multiple devices

```
float vin1[NB][SIZE][SIZE], vin2[NB][SIZE][SIZE], vout[NB][SIZE][SIZE];
#pragma hmpp parallel, device="i%2"
                                                                3D
  for( i=0;i<NB;i++) {</pre>
   //Allocate the mirrors for vin1, vin2 and vout
#pragma hmpp allocate, data["vin1[i]", ...], size={size, size}U 0
                                                                      2D
    //Transfer data to the GPU from the mirrors
#pragma hmpp advancedload, data["vin1[i]","vin2[i]","vout[i]"]
#pragma hmpp sgemm callsite
    sgemm( vin1[i], vin2[i], vout[i] );
//Get back the result
#pragma hmpp delegatedstore, data["vout[i]"]
                                                          GPU1
```



HMPP Accelerated Library Integration

- Use accelerated libraries as an alternative to the original library
 - Keep one single source code with CPU and GPU libraries
 - Uses hmppalt directives with a proxy mechanism
- Compatible with CUBLAS, MAGMA, CULA libraries
- Access of data allocated by library with directives





C/CUDA/...

LSMS case study

Already existing GPU version using cuBLAS and CULA libraries

```
$hmppalt myproxy declare, name="zgemm", fallback=true
                                     SUBROUTINE lsmszgemm(error,transa,...)
                                     END SUBROUTINE lsmszgemm
                                     !$hmppalt lsms declare,\ name="zgetrf", fallback=true
do iblk=nblk,2,-1
                                     SUBROUTINE lsmszgetrf(error, m, n, a, lda, ipiv,
                                     info)
    call cula device zgetrf(...)
    call cula device zgetrs(...)
                                     . . .
                                     END SUBROUTINE lsmszgetrf
    call cublas zgemm(...)
    call cublas_zgemm(...)
                                                                    Declare proxy interfaces
                                      . . .
                                     !$hmpp advancedload
enddo
                                     do iblk=nblk,2,-1
call cublas zgemm(...)
                                     !$hmppalt myproxy
call cublas get matrix(...)
                                     call zgetrf(...)
                                                                Call HMPP alternative library
                                     !$hmppalt myproxy //
                                     call zgemm(...)
                                                               while keep original call in place
                                     !$hmppalt myproxy
                                     call zgemm(...)
                                     enddo
```

Vampir Toolset

VampirTrace

- Application instrumentation
- Pre-run phase
- Via compiler wrapper, library wrapper and/or third-party software
- Measurement
- Event collection (functions calls, MPI, OpenMP, performance counter, memory usage, I/O, GPU)
- Timer synchronization
- Filtering and grouping of events

- Vampir (Client and Server)

 Trace visualization software
 - Alternative and supplement to automatic analysis
 - Show dynamic run-time behavior graphically
 - Provide statistics and performance metrics
 - Interactive browsing, zooming, selecting capabilities





Vampir Performance Analysis Tools

- Custom improvements for the OLCF-3 system
- Focused on three main areas
 - Scaling the Vampir tools to higher processor counts
 - Integrating GPU support for a comprehensive analysis of heterogeneous systems
 - Additional usability enhancements



Vampir toolset challenges

- Support for GPU tracing
- Scaling up to a Titan-sized HPC-system
- Overcome I/O challenges related to the huge amount of data generated by traces as well as the number of tracing streams
- General enhancements in terms of usability and analysis possibilities



Vampir GPU support for Titan

- Successive CUDA support based on latest CUDA releases
- How CUDA fits into Vampir's design
 - accelerator threads treated like OMP threads
 - CUDA memcpy treated like MPI communication
- Tracing of multiple CUDA kernels in one executable
- Tracing of asynchronous GPU events
- Tracing of performance counters on GPUs



Reuse of known metrics:

Thread = Kernel



Message = cudaMemCpy





Vampir Scalability Improvements

- Parallelization of VampirTrace tools: otfmerge, vtfilter, otfprofile
 - enable the processing of traces with >10k streams!!
- Enhanced output of otfprofile
 - Global message statistics
 - Time spent in MPI
 - Process clustering information
- Improved MPI behavior of VampirServer
 - Analyze traces with >10k analysis processes
 - Allows to analyze traces on the entire Titan system
- New displays
 - Performance radar (performance counter for all processes over time)
 - Highlight performance bottlenecks (related to counters)



I/O Challenge

- Online trace file compression (reduce i/o load)
 - Find pattern and record only one (aka rewind)
 - Find pattern irregularities: use marker to highlight pattern irregularities
- Visualization of those compressed traces via multiplexing the recorded pattern (only in the display, not in memory) to show the trace as a whole
- Use IOFSL to aggregate large number of streams in a small number of files (instead of saving each stream in one file) to reduce the load for the Lustre meta data server



S3D on Jaguar (200k+ cores)



19 Managed by UT-Battelle for the U.S. Department of Energy

Vampir Usability/Analysis Enhancements

- Vampir clients for all major OS platforms (Linux, Mac, Win)
- Automatic generation of filter files (based on previous runs) to trace only performance relevant parts
- Performance Radar
 - Derived counters
 - Highlighting of function anomalies based on user defined thresholds
- Enhanced filtering capabilities
 - functions, functions groups, processes inside of Vampir
- Compare View
 - Compare multiple traces directly, while showing them in one display



Vampir Usability/Analysis Enhancements

Performance Radar

Compare View





Allinea DDT Debugger

- Work with Allinea to improve the scalability of the DDT debugger
- Data Analysis
 - Parallel Watchpoints
 - Addition of "sparklines"
 - Scalable data analysis
 - Scalable breakpoints,
 stepping and program
 stack queries
- Major GPU enhancements





DDT Integration with Cray PE

 Support for Abnormal Process Termination (APT), allows to attach DDT to aborted process and review stack

Application 1110443 is crashing. ATP analysis proceeding...

Stack walkback for Rank 23 starting: __start@start.S:113 __libc_start_main@libc-start.c:220 main@atploop.c:48 __kill@0x4b5be7 Stack walkback for Rank 23 done Process died with signal 11: 'Segmentation fault' View application merged backtrace tree file 'atpMergedBT.dot' with 'statview' You may need to 'module load stat'.

atpFrontend: Waiting 5 minutes for debugger to attach...

- Multiple core file support using xt_setup_core_handler()
- Open MPI (Cray XK/alps) version support



DDT GPU Support

- Cray compiler support
 - Stepping into OpenACC regions
 - Setting breakpoints within OpenACC regions
- Support for HMPP directives
 - stepping into HMPP codelets
- Multiwarp stepping in CUDA codes
- CUDA memory debugging capabilities
 - Identify memory leaks on the device
 - Visual display of memory allocated on host and device



DDT Debugging with HMPP directives

$\bigcirc \ominus \ominus \ominus$	X Allinea Distributed Debugging Tool v3.0.rc5						
<u>S</u> ession <u>C</u> ontrol Se <u>a</u> rch <u>V</u> iew <u>H</u> elp							
🗍 Current Group: All 📃 🔽 Focus on current: 🕥 Proc	ess 🔿 Thread 🦵 Step Threads Together						
Threads: 1							
Project Files Fortran Modules Project Files Project Files Search Search Image:	Image: state of the state						
<pre> debug_list.cc del_op.cc del_op.cc del_opv.cc del_opv.cc del_opv.cc del_opv.cc divergence_sphere5d_acc divergence_sphere5d_acc divergence_sphere5d_acc_tuned divergence_sphere5d_gpu_constant : attu divergence_sphere5d_gpu_s : attributes(gl divergence_sphere5d_gpu_s : attributes(gl divergence_sphere5d_gpu_s : attributes(und) divergence_sphere5d_launcher divergence_sphere5d_launcher divergence_sphere5d_launcher_constant divergence_sphere5d_launcher_shared divergence_sphere5d_oscar divergence_sphere5d_oscar </pre>	<pre>120 1\$hmpp <cudagroup> region, args[v5d, Dvv, rmetdetp, metdet, dinv, rdx, rdy, gsize_d, nlev_d, nv_d, gv, vvtemp 721 1\$hmpp <cudagroup> args[div4d].io=out, private=[dvdy00, dudx00, i, j, k, q, 1] 723 do g=1, qsize_d 724 do k=1, nlev_d 725 l convert to contra variant form and multiply by g 726 do j=1, nv_d 727 [\$acc do seq 728 do i=1, nv_d 729 gv(i, j, 1) = metdet(i, j)*(Dinv(1, 1, i, j)*v5d(i, j, k, q, 1) + & 730 Dinv(1, 2, i, j)*v5d(i, j, k, q, 2)) 731 gv(i, j, 2) = metdet(i, j)*(Dinv(2, 1, i, j)*v5d(i, j, k, q, 2)) 733 enddo 735 l compute d/dx and d/dy 736 do j=1, nv_d 737 do j=1, nv_d 738 do l=1, nv_d 739 dudx00=0.0d0 741 do i=1, nv_d 742 dudx00 = dudx00 + Dvv(i, 1) * gv(i, j, 1) 743 dvdy00 = dvdy00 + Dvv(i, 1) * gv(j, i, 2) * The second second</cudagroup></cudagroup></pre>						
Input/Output Breakpoints Watchpoints Stacks	s Tracepoints Evaluate						
qsize= 65 nlev= 60 nelem= Iteration 1 CPU time(ms) 8.30793380737305 Checksum= 0.138112301779722 Type here ('Enter' to send):	11 						

DDT and OpenACC Directives

<u>Session</u> <u>Contr</u>	rol Se <u>a</u> rch <u>V</u> i	ew <u>H</u> elp		[🛛 Allinea DD1	HEAD			
Þ •	▶ <mark> </mark>								
Focus on curre	ent: 💽 Process	C Thread 🗖	Step Threa	ds Together Step CUDA	threads by: War	p (default) 💌			
Threads: 1 2 K1									
CUDA Threads Block 0 😴 0 😴 Thread 0 😴 0 😴 Go									
Pr Forti	r 📔 📧 bas	sic.f90 🔀 🛛 📧	acc_runtim	e.c 💌				Locals Curren	Curr GPU
Project Files	₽ × 30	inpu [.]	t(i) = i'	*4.0				▲ Locals	₽×
Search (Ctrl+K)	31	END DO						Variable Name	Value 🔺
🔳 Project Files	33	result =	-5					tesult	
🗄 💼 Source Tre	e 35	!\$omp acc	_region	_loop private (t) f	irstprivate	(a,var_int,var_o	double,var_co	or var bool	8 .TRUE.
🕀 🛑 Header Fil	es 36	DO i = 1	,n					var_bool	.TRUE.
Source File	es 37	t = :	input(1)	* 4.0				··var_complex	(8,9)
+ basic.19	30	resu	$\mathbf{t}(\mathbf{i}) = \mathbf{i}$	t+a-h INet	consequence	$e_{result(i)} = t$	2	var_complex	(8,9)
	40	i cou			consequence	5, 1050000(1) = (var_double	6
	41	IF ()	/ar bool) THEN				var_double	6
	42		/ar_doub	le = var_int + var	_real;			var_int	3
	6 43	,	/ar_comp	lex = CMPLX(a,t)					3
	44		Chanaa	tone/otnings not a	upperted ap	a a a a l a m a t a m a			4.5
	45		lvar etr	ipa – var string /	upported on / var char	accelerators			
•			var str.	rig – var string /	y var char		•	Type: none selected	
Input/Output	Breakpoints	Watchpoints	Stacks	Kernel Progress View	Tracepoints	Tracepoint Output	Ev	valuate	₽×
Stacks							đ× E	xpression Value	
Threads GF	PU Threac Fun	ction					<u> </u>		
1	0??								
1	0 🔤 🖂 🖻 ma	ain (basic.f90:51)							
1		ray_acc_sync (ac	c_runtime.c	:1019)					
1		cray_acc_new_s	sync_info_1	(acc_runtime.c:980)					
1		⊡cray_acc_m	v_synchron	ize (acc_nw_nviula.c.448)					
1		□ ? ? 古22							
1	0	: ⊨??							
1	0	<u> </u>							
1	0 0	่ בי בי							
1	0 0	L.??							
1 10	24 🗖 🖬 🖬	ain \$ck I35 1 (ba	asic.f90:51)				_		Ready

DDT and VisIT

- Integration with VisIT (Available with VisIT 2.5 summer 2012)
 - Coupling advanced application generated data visualization capabilities with scalable debugging

		DDT - Options	×		
	System	VisIt Visualization			
2 ⁵	Job Submission	VisIt is a free interactive parallel visualization and graphical analysis t scientific data. By setting 'Visualization breakpoints' DDT can feed in	ool for viewing ormation to VisIt.		
	Remote Launch	DDT can also work with programs you have already modified to displ (instrumented using VisIt's runtime library libsim).	ay data in VisIt		
		✓ Allow the use of VisIt with DDT	Session Control Search View Help	Allinea DDT HEAD - + ×	VISIT 2,4,0 – + × <u>File Controls Options Windows PlotAtts OpAtts Help</u> Global
Province of the second	Code Viewer	Visit launch command: /home/jbyrd/Work/visit/visit2.4.0/src/bin/v	isit	U 🔯 ! 🛍 v 😨 v 🍕	Active window 1 🗘 🗌 Auto apply
	Appearance	Custom arguments:	All 0 1 2 3 Create Group		💕 🚰 🐑 🗞 🕅 Open Close Reopen Replace Overlay
		 Launch Visit with small viewer (-small argument) 	Project Files Project Pro	Allinea DDT - Visualization Point × 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	Active source 001324225388.ftables-ddt.sim2
()	Vislt	✓ Enable visualisation breakpoints (preloads ddtsim if its not alreaded)	A B C D 001324225388.ftables-ddt.sim2 ddtmesh.domain 1	point and can only be resumed by Visit. Use Visit to examine the current variable(s) being visualized, bio can step to the next visualization point using Visit's animation controls.	
		Visit launch command on compute nodes :	Point: <21.8194, 9.35855> Zone: 141 Incident Nodes: 150 151 166 167 tables: <zonal> = 208</zonal>	To return control to DDT (and suspend Visit) use the "Release control to DDT button on the Visit viewer visualization point." To return control to DDT will also regain control if your program hits a breakpoint whilst running to the next al variables or the need to pause the	Plots
				This dialog will close automatically when Visit returns control to DDT	Pseudocolor - tables
			Max Tabs 8 + Save Picks as	unch Visit Request control from Visit	
			Float Format %g	Window 1 - + × 1 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Apply operators to all plots
			Don't clear this window Clear Picks		Apply subset selections to all plots
		L	Concise Output		
			Mesh Name Mesh Name Global nodes/zones Reference pick letter		
			For Nodes	200 - 200 -	
			Physical Coords Block-Logical Coords For Zones		
			Id Domain-Logical Coords Rick Logical Coords	** *	
27	Managed by UT-Battelle				
	for the U.S. Department	of Energy	Make default Apply Post Dismiss	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

Thank you for your attention!

• Questions and comments are most welcome