

An Update on the Cray Tools Activities for Extreme Scale Computing

Luiz DeRose and Heidi Poxon Cray Inc.

Moving from X86 to Hybrid Multi-core Systems



- Running MPI only on a node will not work well
 - Too much memory used, even if on-node shared communication is available
 - As the number of MPI ranks increases, more off-node communication can result, creating a network injection issue
- Focus on where MPI starts leveling off
- Address by adding additional levels of parallelism, reducing MPI ranks per node
 - MPI -> MPI + OpenMP
 - MPI + OpenMP -> MPI + OpenMP GPU extensions

Steps to Porting to Hybrid Multi-core Systems



- Maximize on-node communication if MPI point-to-point communication is dominant in the program
 - Auto-grid detection and placement suggestions
- Determine where to add additional levels of parallelism
 - Find top time consuming loops with enough work for GPU
 - Loop statistics
- Do parallel analysis and restructuring on targeted high level loops
 - Scoping assistance

Steps to Porting to Hybrid Multi-core Systems (2)



Add parallel directives and acceleration extensions

- OpenMP extensions
- Run on X86 + GPU and get performance feedback
- Optimize for data locality and copies to the GPU
- Optimize kernel on GPU
 - Cray performance tools statistics

Automatic Communication Grid Detection



- Analyze runtime performance data to identify grids in a program to maximize on-node communication
 - Example: nearest neighbor exchange in 2 dimensions
 - Sweep3d uses a 2-D grid for communication
- Determine whether or not a custom MPI rank order will produce a significant performance benefit
- Grid detection is helpful for programs with significant point-topoint communication
- Produce a custom rank order if it's beneficial based on grid size, grid order and cost metric



Example summary for sweep3d (pat_report table Notes)

This application appears to use point-to-point MPI communication at least partly organized into a 8 X 6 grid pattern. Time spent in MPI routines accounted for over 63.1% of the execution time. A portion of this time could potentially be saved by utilizing a rank order that maximizes the fraction of communication that is between ranks on the same node. The following table estimates this fraction for several rank orders.

An MPICH_RANK_ORDER file was generated along with this report and contains the Custom rank order from the following table. This file also contains usage instructions and a table of alternative rank orders.

Automatic Grid Detection Example Table



Table 4: Sent Message Stats for Selected MPI Rank Orders

Rank	On-Node	On-Node Options for grid_order utility
Order	Bytes/PE	Bytes/PE%
		of Total
		Bytes/PE
Custom	1.30e+07	50.00% -R -P -m 48 -n 4 -g 8,6 -c 2,1
SMP	8.10e+06	31.25%
Fold	6.75e+05	2.60%
RoundRobin	0.00e+00	0.00%

MPICH_RANK_ORDER File Example



The 'Custom' rank order in this file targets nodes with multi-core
processors, based on Sent Msg Total Bytes collected for:
#

Program: /lus/nid00030/heidi/sweep3d/mod/sweep3d.mpi

- # Ap2 File: sweep3d.mpi+pat+27054-89t.ap2
- # Number PEs: 48

```
# Max PEs/Node: 4
```

```
#
```

. . .

To use this file, set the environment variable

MPICH_RANK_REORDER_METHOD to 3 prior to executing the program.
#

The following table lists rank order alternatives and the grid_order# command-line options that can be used to generate a new order.

Loop Statistics



- Helps identify loops to move to GPU:
 - Loop timings approximate how much work exists within a loop
 - Trip counts can be used to help carve up loop on GPU
- Enabled with CCE –h profile_generate option
- Loop statistics reported by default in pat_report table

Example Loop Stats



Notes for table 2:

Table option:

-0 loops

The Function value for each data item is the avg of the PE values. (To specify different aggregations, see: pat help report options s1)

This table shows only lines with Loop Incl Time / Total > 0.0095. (To set thresholds to zero, specify: -T)

Loop data version: L.12.2:B.3.1

Loop instrumentation can interfere with optimizations, so time reported here may not reflect time in a fully optimized program.

Loop stats can safely be used in the compiler directives:

!PGO\$ loop_info est_trips(Avg) min_trips(Min) max_trips(Max)
#pragma pgo loop_info est_trips(Avg) min_trips(Min) max_trips(Max)

Explanation of Loop Notes (P=1 is highest priority, P=0 is lowest): novec (P=0.5): Loop not vectorized (see compiler messages for reason). sunwind (P=1): Loop could be vectorized and unwound. vector (P=0.1): Already a vector loop.

Example Loop Stats (2)



Table 2: Loop Stats from -hprofile_generate

Loop Loop Incl Loop Inc	cl Loop Loop	Loop Function=/.LOOP\.
Incl Time Time	/ Hit Trips	Notes PE='HIDE'
Time / Hi	it Avg	
Total		
24.6% 0.057045 0.0005	570 100 64.1	novec calc2LOOP.0.li.614
24.0% 0.055725 0.0000	009 6413 512.0	<pre>vector calc2LOOP.1.li.615</pre>
18.9% 0.043875 0.0004	439 100 64.1	novec calc1LOOP.0.li.442
18.3% 0.042549 0.0000	007 6413 512.0	vector calc1LOOP.1.li.443
17.1% 0.039822 0.0004	406 98 64.1	novec calc3LOOP.0.li.787
16.7% 0.038883 0.0000	006 6284 512.0	vector calc3LOOP.1.li.788
9.7% 0.022493 0.0002	230 98 512.0	vector calc3LOOP.2.li.805
4.2% 0.009837 0.0000	098 100 512.0	vector calc2LOOP.2.li.640

Cray Inc.

Source Code – Loopmark



		1000	1000	0	6.6	DO 300 I-1 M
~	32.33% calc2.F		Γ.	⊘	66	
	32.33% CALC2		- 1		67	DO 200 J=js,je
	Loop@66				68	UNEW(I+1,J) = UOLD(I+1,J)+
	Loop@67	1			69	1 TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)+CV
	Loop@89				70	2 +CV(I+1,J))-TDTSDX*(H(I+1,J)-H(I,J))
⊳	17.34% calc1.F				71	if(j.gt.1)then
⊳	0.21% swim.F				72	VNEW(I,J) = VOLD(I,J)-TDTS8*(Z(I+1,J)+Z(I,J))
					73	1 *(CU(I+1,J)+CU(I,J)+CU(I,J-1)+CU(I+1,J-1))
					74	2 -TDTSDY*(H(I,J)-H(I,J-1))
					75	endif
		1			76	if(j.eq.n)then
					77	VNEW(I,J+1) = VOLD(I,J+1)-TDTS8*(Z(I+1,J+1)+Z(I))
					78	1 *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
					79	2 -TDTSDY*(H(I,J+1)-H(I,J))
					80	endif
					81	PNEW(I,J) = POLD(I,J)-TDTSDX*(CU(I+1,J)-CU(I,J))
					82	1 -TDTSDY*(CV(I,J+1)-CV(I,J))
Info			L		83	200 CONTINUE
Line 66:		1			84	
	rolled 2 times. erchanged with loop				85	СМЕ
at line 6	-				86	

Cray Inc.

Display Scoping Information for Selected Loop



Ele	
Page 1 S LBM3D2P_d.690 ▼ LBM3D2P_d.190 0 1064 do k=0,lz<1 ▶ LBM3D2P_0 0 1065 do j=0,local_ly-1 ▶ INTALIZATION 1066 do i=0,local_lx-1 ▶ OIS 1067 if (cell(i,j,k/)=4) then ▶ SET_BOUNDARY_MACRO_PRESS2 1068 rho_tmp = 0.0d0 ▶ COLLISIONA 1070 rho_tmp = 0.0d0 ▶ COLUSIONB 1070 rho_tmp = 0.0d0 ▶ COLUSIONB 1070 rho_tmp = 0.0d0 ▶ COLUSIONB 1072 uy_tmp = 0.0d0 ▶ CAL_VELOCITY0 uz_tmp = 0.0d0 1071 ▶ CAL_VELOCITY0 1075 rho_tmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& ▶ COUG901066 1077 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& ▼ INJECTIONO 1078 + R(i,j,k, 3)+ R(i,j,k, 1)+ R(i,j,k, 1)& ↓ Loop@1152 1080 rho_ttmp = B(i,j,k, 0)+ B(i,j,k, 1)+ B(i,j,k, 2)& ↓ NJECTION 1081 + B(i,j,k, 3)+ B(i,j,k, 1)+ B(i,j,k, 2)& ↓ NJECTION 1081 + B(i,j,k, 1)+ B(i,j,k, 1)+ B(i,j,k, 1)& ↓ LBM_MMPROC 190 1083 + B(i,j,k, 1)+ B(i,j,k, 1)+ B(i,j,k, 1)+ ↓ LBM_MMPROC 1	H
Y LBMB22P_d.190	
► LBM3D2P_0 4190 ▶ INITALIZATION ▶ INITALIZATION ▶ OS > SET_BOUNDARY_MACRO_PRESS2 > SET_BOUNDARY_MACRO_PRESS > COLLISIONA > COLUSIONA > COLISIONA > COLUS	
▶ IBMB02P_0 ▶ INTALUZATION ▶ DIS > SET_BOUNDARY_MACRO_PRESS > SET_BOUNDARY_MCRO_PRESS > COLUSIONA > COLUSIONA > COLUSIONB > RECOLOR > STREAMNG > WALL_BOUNDARY > CAL_VELOCITYO C CAL_VELOCITYO	
▶ DIS 1067 if (cell(i,j,k)=4) then ▶ SET_BOUNDARY_MCRO_PRESS 1068 rho_tmp = 0.0d0 ▶ COLLISIONA 1069 rho_tmp = 0.0d0 ▶ COLLISIONA 1070 rho_btmp = 0.0d0 ▶ COLLISIONB 1071 ux_tmp = 0.0d0 ▶ RECOLOR 1072 uy_tmp = 0.0d0 ▶ STEAMNG 1073 uz_tmp = 0.0d0 ▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITYO 1074 1075 ▼ CAL_VELOCITYO 1076 + R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& ■ Loop@1065 1077 + R(i,j,k, 5)+ R(i,j,k, 5)& ■ Loop@1066 1077 + R(i,j,k, 0)+ R(i,j,k, 10)+ R(i,j,k, 11)& ▼ INJECTIONO 1078 + R(i,j,k, 0)+ R(i,j,k, 10)+ R(i,j,k, 11)& ■ Loop@1153 1080 rho_btmp = B(i,j,k, 0)+ B(i,j,k, 14) ■ INJECTION 1081 + B(i,j,k, 3)+ B(i,j,k, 10)+ B(i,j,k, 5)& ■ INJECTION 1082 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ INJECTION 1082 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ INJECTION 1083 + B(i,j,k, 12)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ INJECTION 1083 <td></td>	
> SET_BOUNDARY_MACRO_PRESS2 > SET_BOUNDARY_MCRO_PRESS > COLUSIONA > COLUSIONA > COLUSIONA > COLUSIONA > COLUSIONB > RECOLOR > STREAMNG > WALL_BOUNDARY > CAL_VELOCITY0 ▼ CAL_VELOCITY0 ▼ CAL_VELOCITY0 ▼ CAL_VELOCITY0 ▼ CAL_VELOCITY0 ▼ INJECTION0 Loop@1065 Loop@1152 Loop@1153 Loop@1154 > INJECTION SATURATION > LBM_MMPROC.190 ■ LBM_MMPROC.190	
▶ SET_BOUNDARY_MCR0_PRESS 1069 rho_rtmp = 0.0d0 ▶ COLLISIONA 1070 rho_btmp = 0.0d0 ▶ COLLISIONB 1071 ux_tmp = 0.0d0 ▶ RECOLOR 1072 uy_tmp = 0.0d0 ▶ STREAMNG 1073 uz_tmp = 0.0d0 ▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITYO 1074 1074 ▼ CAL_VELOCITY 1075 rho_rtmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& Loop@1065 1077 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& Loop@1066 1077 + R(i,j,k, 0)+ R(i,j,k, 10)+ R(i,j,k, 11)& ↓ Loop@1152 1079 + R(i,j,k, 12)+ R(i,j,k, 13)+ R(i,j,k, 14) Loop@1153 1080 rho_btmp = B(i,j,k, 0)+ B(i,j,k, 1)+ B(i,j,k, 14) ↓ INJECTION 1081 + B(i,j,k, 3)+ B(i,j,k, 1)+ B(i,j,k, 11)& ↓ INJECTION 1082 + B(i,j,k, 6)+ B(i,j,k, 7)+ B(i,j,k, 10)+ B(i,j,k, 11)& ↓ INJECTION 1083 + B(i,j,k, 6)+ B(i,j,k, 10)+ B(i,j,k, 11)& ↓ LBM_MPIPROC.190 1084 + B(i,j,k, 12)+ B(i,j,k, 13)+ B(i,j,k, 14)	
▶ COLLISIONA 1070 rho_btmp = 0.0d0 ▶ COLLISIONB 1071 ux_tmp = 0.0d0 ▶ RECOLOR 1072 uy_tmp = 0.0d0 ▶ STREAMNG 1073 uz_tmp = 0.0d0 ▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITYO 1073 uz_tmp = 0.0d0 ▼ CAL_VELOCITYO 1074 1075 ▼ CAL_VELOCITY 1076 + R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& ■ Loop@1065 1077 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& ■ Loop@1066 1077 + R(i,j,k, 0)+ R(i,j,k, 10)+ R(i,j,k, 11)& ■ Loop@1152 1076 + R(i,j,k, 10)+ R(i,j,k, 10)+ R(i,j,k, 11)& ■ Loop@1153 1080 rho_btmp = B(i,j,k, 0)+ B(i,j,k, 1)+ R(i,j,k, 14) ■ Loop@1154 1081 + B(i,j,k, 3)+ B(i,j,k, 3)+ B(i,j,k, 3)& ■ INJECTION 1082 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ Loop@1154 1081 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ Loop@1154 1082 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ Loop@1154 1082 + B(i,j,k, 0)+ B(i,j,k, 10)+ B(i,j,k, 11)& ■ LBM_MPIPROC.190 1083 + B(i,j,k, 12)+ B(i,j,k, 13)+ B(i,j,k, 14	
▶ COLLISIONB 1070 Ind_dtmp = 0.0d0 ▶ RECOLOR 1071 $ux_tmp = 0.0d0$ ▶ STREAMNG 1072 $uy_tmp = 0.0d0$ ▶ WALL_BOUNDARY 1073 $uz_tmp = 0.0d0$ ▶ CAL_VELOCITY0 1074 ▼ CAL_VELOCITY 1075 rho_rtmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& Loop@1065 1076 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& Loop@11666 1077 + R(i,j,k, 6)+ R(i,j,k, 10)+ R(i,j,k, 11)& ↓ INJECTION0 1078 + R(i,j,k, 12)+ R(i,j,k, 13)+ R(i,j,k, 14) Loop@1152 1079 + R(i,j,k, 0)+ B(i,j,k, 1)+ B(i,j,k, 2)& ↓ INJECTION 1080 rho_btmp = B(i,j,k,0)+ B(i,j,k, 1)+ B(i,j,k, 2)& ↓ INJECTION 1081 + B(i,j,k, 6)+ B(i,j,k, 1)+ B(i,j,k, 3)& ↓ LBM_MPIPROC.f90 1083 + B(i,j,k, 12)+B(i,j,k, 13)+B(i,j,k, 14)	
▶ RECOLOR 1071 ux_tmp = 0.0d0 ▶ STREAMING 1072 uy_tmp = 0.0d0 ▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITY0 1074 1075 ▼ CAL_VELOCITY 1075 rho_rtmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& Loop@1065 1076 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& Loop@1166 1077 + R(i,j,k, 6)+ R(i,j,k, 7)+ R(i,j,k, 8)& ▼ INJECTIONO 1078 + R(i,j,k, 10)+R(i,j,k, 11)& Loop@1152 1079 + R(i,j,k, 12)+R(i,j,k, 13)+R(i,j,k, 14) Loop@1153 1080 rho_btmp = B(i,j,k,0)+ B(i,j,k,7)+ B(i,j,k,3)& LOOD@1154 1081 + B(i,j,k,3)+ B(i,j,k,7)+ B(i,j,k,3)& ▶ INJECTION 1082 + B(i,j,k,0)+ B(i,j,k,1)+ B(i,j,k,1) LBM_MPIPROC.90 1083 + B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)	
▶ STREAMING 1072 uy_tmp = 0.0d0 ▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITY0 1074 ▼ CAL_VELOCITY 1075 rho_rtmp = R(i,j,k, 0)+ R(i,j,k, 1)+ R(i,j,k, 2)& Loop@1065 1076 + R(i,j,k, 3)+ R(i,j,k, 4)+ R(i,j,k, 5)& Loop@1066 1077 + R(i,j,k, 0)+ R(i,j,k, 10)+ R(i,j,k, 11)& ▼ INJECTION0 1078 + R(i,j,k, 9)+ R(i,j,k, 10)+ R(i,j,k, 11)& Loop@1152 1079 + R(i,j,k, 12)+ R(i,j,k, 13)+ R(i,j,k, 14) Loop@1153 1080 rho_btmp = B(i,j,k, 0)+ B(i,j,k, 1)+ B(i,j,k, 2)& Loop@1154 1081 + B(i,j,k, 3)+ B(i,j,k, 7)+ B(i,j,k, 8)& ▶ INJECTION 1082 + B(i,j,k, 6)+ B(i,j,k, 7)+ B(i,j,k, 8)& LBM_MPIPROC.f90 1083 + B(i,j,k, 9)+B(i,j,k, 10)+B(i,j,k, 14)	
▶ WALL_BOUNDARY 1073 uz_tmp = 0.0d0 ▶ CAL_VELOCITY0 1074 ▼ CAL_VELOCITY 1075 □ cor @1004 1076 □ cor @1065 1077 □ cor @1066 1078 ▼ INJECTION0 1079 □ cor @1152 1079 □ cor @1153 1079 □ cor @1154 1080 ▶ INJECTION 1081 ▶ LBM_MPIPROC.f90 1084	
▶ CAL_VELOCITY0 1074 ▼ CAL_VELOCITY 1075 Loop@1065 1076 Loop@1066 1077 ▼ INJECTION0 1079 Loop@1152 1079 Loop@1153 1079 Loop@1154 1080 ▶ INJECTION 1081 ▲ LBM_MPIPROC.f90 1083 + B(i,j,k, 12)+B(i,j,k, 10)+B(i,j,k, 11)& 1084 + B(i,j,k, 12)+B(i,j,k, 13)+B(i,j,k, 14)	
∇ CAL_VELOCITY 1075 rho_rtmp = R(i, j, k, 0) + R(i, j, k, 1) + R(i, j, k, 2)& Loop@1065 1076 + R(i, j, k, 3) + R(i, j, k, 4) + R(i, j, k, 5)& Loop@1066 1077 + R(i, j, k, 6) + R(i, j, k, 7) + R(i, j, k, 8)& ∇ INJECTION0 1078 + R(i, j, k, 10) + R(i, j, k, 10) + R(i, j, k, 11)& Loop@1152 1079 + R(i, j, k, 12) + R(i, j, k, 13) + R(i, j, k, 14) Loop@1153 1080 rho_btmp = B(i, j, k, 0) + B(i, j, k, 1) + B(i, j, k, 2)& Loop@1154 1081 + B(i, j, k, 3) + B(i, j, k, 3) + B(i, j, k, 5)& NJECTION 1082 + B(i, j, k, 0) + B(i, j, k, 7) + B(i, j, k, 8)& SATURATION 1083 + B(i, j, k, 12) + B(i, j, k, 10) + B(i, j, k, 11)& V LBM_MPIPROC.f90 1084 + B(i, j, k, 12) + B(i, j, k, 14)	
Loop@1065 1076 + R(i,j,k, 3) + R(i,j,k, 4) + R(i,j,k, 5)& Loop@1066 1077 + R(i,j,k, 6) + R(i,j,k, 7) + R(i,j,k, 8)& ▼ INJECTION0 1078 + R(i,j,k, 9) + R(i,j,k, 10) + R(i,j,k, 11)& Loop@1152 1079 + R(i,j,k, 12) + R(i,j,k, 13) + R(i,j,k, 14) Loop@1153 1079 + R(i,j,k, 3) + B(i,j,k, 1) + B(i,j,k, 2)& Loop@1154 1080 rho_btmp = B(i,j,k,0) + B(i,j,k,1) + B(i,j,k, 5)& ▶ INJECTION 1082 + B(i,j,k,6) + B(i,j,k,7) + B(i,j,k,1) & SATURATION 1083 + B(i,j,k,9) + B(i,j,k,10) + B(i,j,k,11)& ▶ LBM_MPIPROC.f90 1084 + B(i,j,k,12) + B(i,j,k,13) + B(i,j,k,14)	
Loop@1065 1077 + $R(i,j,k, 6)$ + $R(i,j,k, 7)$ + $R(i,j,k, 8)$ & Loop@1066 1078 + $R(i,j,k, 9)$ + $R(i,j,k, 10)$ + $R(i,j,k, 11)$ & Loop@1152 1079 + $R(i,j,k, 12)$ + $R(i,j,k, 13)$ + $R(i,j,k, 14)$ Loop@1153 1080 rho_btmp = $B(i,j,k, 0)$ + $B(i,j,k, 1)$ + $B(i,j,k, 2)$ & Loop@1154 1081 + $B(i,j,k, 3)$ + $B(i,j,k, 7)$ + $B(i,j,k, 3)$ & > INJECTION 1082 + $B(i,j,k, 6)$ + $B(i,j,k, 7)$ + $B(i,j,k, 8)$ & > LBM_MPIPROC.190 1083 + $B(i,j,k, 12)$ + $B(i,j,k, 13)$ + $B(i,j,k, 14)$	
Loop@1066 1078 + R(i,j,k, 9)+R(i,j,k, 10)+R(i,j,k, 11)& ↓ INJECTIONO 1079 + R(i,j,k, 12)+R(i,j,k, 13)+R(i,j,k, 14) ↓ Loop@1153 1079 + R(i,j,k, 12)+R(i,j,k, 13)+R(i,j,k, 14) ↓ Loop@1154 1080 rho_btmp = B(i,j,k,0)+ B(i,j,k,1)+ B(i,j,k,2)& ↓ INJECTION 1081 + B(i,j,k,3)+ B(i,j,k,4)+ B(i,j,k,5)& ↓ INJECTION 1082 + B(i,j,k,6)+ B(i,j,k,10)+B(i,j,k,11)& ↓ LBM_MPIPROC.f90 1084 + B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)	
Loop@1152 1079 +R(i,j,k, 12)+R(i,j,k, 13)+R(i,j,k, 14) Loop@1153 1080 rho_btmp = B(i,j,k,0)+ B(i,j,k,1)+ B(i,j,k,2)& Loop@1154 1081 + B(i,j,k,3)+ B(i,j,k,4)+ B(i,j,k,5)& > INJECTION 1082 + B(i,j,k,6)+ B(i,j,k,7)+ B(i,j,k,1)+ B(i,j,k,4) > LBM_MPIPROC.f90 1083 + B(i,j,k,0)+ B(i,j,k,10)+ B(i,j,k,11)&	
Loop@1153 1080 rho_btmp = B(i,j,k,0)+ B(i,j,k,1)+ B(i,j,k,2)& Loop@1154 1081 + B(i,j,k,3)+ B(i,j,k,4)+ B(i,j,k,5)& ▷ INJECTION 1082 + B(i,j,k,6)+ B(i,j,k,7)+ B(i,j,k,8)& SATURATION 1083 + B(i,j,k,9)+B(i,j,k,10)+B(i,j,k,11)& ▷ LBM_MPIPROC.f90 1084 + B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)	
Loop@1154 1081 + B(i,j,k,3)+ B(i,j,k,4)+ B(i,j,k,5)& ▷ INJECTION 1082 + B(i,j,k,6)+ B(i,j,k,7)+ B(i,j,k,8)& SATURATION 1083 + B(i,j,k,9)+B(i,j,k,10)+B(i,j,k,11)& ▷ LBM_MPIPROC.f90 1084 + B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)	
▶ INJECTION 1081 1082 + B(i,j,k,G)+ B(i,j,k,G)+ B(i,j,k,G), K,G) (K,G) (K	
SATURATION 1082 + B(i, j, k, 6)+ B(i, j, k, 7)+ B(i, j, k, 8)& LBM_MPIPROC.f90 1083 + B(i, j, k, 9)+B(i, j, k, 10)+B(i, j, k, 11)& 1084 + B(i, j, k, 12)+B(i, j, k, 13)+B(i, j, k, 14)	
SA IURA IION ▶ LBM_MPIPROC.f90 1083 + B(i,j,k,9)+B(i,j,k,10)+B(i,j,k,11)& 1084 +B(i,j,k,12)+B(i,j,k,13)+B(i,j,k,14)	
1084 +B(i,j, k, 12)+B(i,j, k, 13)+B(i,j, k, 14)	
1086 - R(i, j, k, 4)- B(i, j, k, 4)&	
1087 + R(i, j, k, 7)+ B(i, j, k, 7)&	
1088 - R(i, j, k, 8)- B(i, j, k, 8)&	
1089 + R(i, j, k, 9)+ B(i, j, k, 9)&	
1090 +R(i,j,k, 10)+B(i,j,k,10)&	

CScADS, Aug 2011

Cray Inc.

13

Display Scoping Information for Selected Loop (2)



0 0		X OpenMP Construct
	LBM3	3D2P_d.f90: lines 1064 -> 1130
Name	Туре	Scope F L Info
b	Scalar	Shared
cell	Scalar	Shared
local_lx	Scalar	Shared
local_ly	Scalar	Shared
Iz	Scalar	Shared
r	Scalar	Shared
rho	Scalar	Shared
rho_btmp	Scalar	Private N N
rho_rtmp	Scalar	Private N N
rho_tmp	Scalar	Private N N
ux_tmp	Scalar	Private N N
uxyz	Scalar	Shared
uz_tmp	Scalar	Private N N
		<u> </u>

Example Performance Statistics



Table 1: Profile by Function Group and Function

Time% Time Imb. Imb. Calls Group	
Time Time% Function	
PE=HIDE	
Thread=HIDE	
100.0% 18.113521 6.0 Total	
100.0% 18.113443 5.0 USER	
90.6% 18.113000 0.000000 0.0% 1.0 acc_sampleACC_DATA_REGION@li.2	23
9.4% 0.000443 0.000000 0.0% 1.0 acc_sampleACC_REGION@li.24	
0.0% 0.000078 0.000000 0.0% 1.0 ETC	
0.0% 0.000078 0.000000 0.0% 1.0 exit	

Cray Inc.

Example Performance Statistics



Table 2: Time and Bytes Transferred for Accelerator Regions

Host	Host	ACC ACC Copy A	cc Copy Calls Calltree	
Time%	Time	Time In	Out	
		(MBytes) (MBytes)	
100.0%	18.113	18.112 209.808	209.808 4 Total	
100.0%	18.113	18.112 209.808	209.808 4 acc_sample_	
			acc_sample	ACC_DATA_REGION@li.23
3 90.6%	16.418		1 sync	
3 9.4%	1.695	1.695 209.808	209.808 2 transfer	
3 0.0%	0.000	16.418 0.000	0.000 1 acc_sample_	.ACC_REGION@li.24
4	1		async_kern	el
==========				

The Next Generation of Debuggers on Cray Systems





- Systems with hundreds of thousands of threads of execution need a new debugging paradigm
 - Innovative techniques for productivity and scalability
 - Scalable Solutions based on MRNet from University of Wisconsin Wisconsin STAT Stack Trace Analysis Tool
 - » Scalable generation of a single, merged, stack backtrace tree
 - running at 216K back-end processes
 - ATP Abnormal Termination Processing
 - » Scalable analysis of a sick application, delivering a STAT tree and a minimal, comprehensive, core file set.
 - Comparative debugging
 - A data-centric paradigm instead of the traditional control-centric paradigm
 - Collaboration with Monash University and University of Wisconsin for scalability
 - Fast Track Debugging
 - Debugging optimized applications
 - Added to Allinea's DDT 2.6 (June 2010)