UL HPC School 2017 PS6: Debugging, profiling and performance analysis



V. Plugaru

University of Luxembourg (UL), Luxembourg http://hpc.uni.lu







UNIVERSITÉ DU LUXEMBOURG

Latest versions available on Github:



UL HPC tutorials:

https://github.com/ULHPC/tutorials

http://hpc.uni.lu/hpc-school/

UL HPC School:

PS6 tutorial sources:

https://github.com/ULHPC/tutorials/tree/devel/advanced/debugging_profiling





V. Plugaru & UL HPC Team (University of Luxembourg)

UL HPC School 2017/ PS6

2 / 30



Summary



2 Debugging and profiling tools

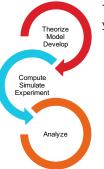




V. Plugaru & UL HPC Team (University of Luxembourg)



Main Objectives of this Session

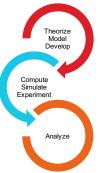


This session is meant to show you some of the various tools you have at your disposal on the UL HPC platform to: understand + solve development & runtime problems





Main Objectives of this Session



This session is meant to show you some of the various tools you have at your disposal on the UL HPC platform to:

understand + solve development & runtime problems

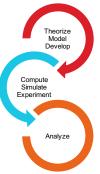
During the session we will:

- discuss what happens when an application runs out of memory and how to discover how much memory it actually requires.
- see debugging tools that help you understand why your code is crashing.
- see profiling tools that show the bottlenecks of your code - and how to improve it.





Main Objectives of this Session



This session is meant to show you some of the various tools you have at your disposal on the UL HPC platform to:

understand + solve development & runtime problems

During the session we will:

- discuss what happens when an application runs out of memory and how to discover how much memory it actually requires.
- see debugging tools that help you understand why your code is crashing.
- see profiling tools that show the bottlenecks of your code - and how to improve it.

Knowing what to do when you experience a problem is half the battle.





Summary









V. Plugaru & UL HPC Team (University of Luxembourg)



Tools at your disposal (I)

Common tools used to understand problems

• Do you know what time it is?

 $\,\hookrightarrow\,$ /usr/bin/time -v is just magic sometimes

- Don't remember where you put things?
 - $\,\hookrightarrow\,$ Valgrind can help with your memory issues
- Is your application firing on all cylinders?
 - \hookrightarrow with **htop** green means go! (red is bad)
- Got stuck?
 - $\,\hookrightarrow\,$ strace can tell you where you are and how you got there

Some times simple tools help you solve big issues.





Tools at your disposal (II)

HPC specific tools - Allinea

- Allinea DDT (part of Allinea Forge)
 - $\,\hookrightarrow\,$ Visual debugger for C, C++ and Fortran threaded and // code
- Allinea MAP (part of Allinea Forge)
 - $\,\hookrightarrow\,$ Visual C/C++/Fortran profiler for high performance Linux code
- Allinea Performance Reports
 - $\,\hookrightarrow\,$ Application characterization tool





Tools at your disposal (II)

HPC specific tools - Allinea

- Allinea DDT (part of Allinea Forge)
 - $\,\hookrightarrow\,$ Visual debugger for C, C++ and Fortran threaded and // code
- Allinea MAP (part of Allinea Forge)
 - $\,\hookrightarrow\,$ Visual C/C++/Fortran profiler for high performance Linux code
- Allinea Performance Reports
 - $\,\hookrightarrow\,$ Application characterization tool

Allinea tools are licensed

Make sure enough tokens available to profile/debug your code in the requested configuration (#cores)!

- \hookrightarrow license check can be integrated in common RJMS (is in SLURM)
- $\,\hookrightarrow\,$... so your jobs are able to wait for tokens to be available



Tools at your disposal (III)

HPC specific tools - Intel

- Intel Advisor
 - $\,\hookrightarrow\,$ Vectorization + threading advisor: check blockers and opport.
- Intel Inspector
 - $\,\hookrightarrow\,$ Memory and thread debugger: check leaks/corrupt., data races
- Intel Trace Analyzer and Collector
 - $\,\hookrightarrow\,$ MPI communications profiler and analyzer: evaluate patterns
- Intel VTune Amplifier
 - $\,\hookrightarrow\,$ Performance profiler: CPU/FPU data, mem. + storage accesses





Tools at your disposal (III)

HPC specific tools - Intel

- Intel Advisor
 - $\,\hookrightarrow\,$ Vectorization + threading advisor: check blockers and opport.
- Intel Inspector
 - $\,\hookrightarrow\,$ Memory and thread debugger: check leaks/corrupt., data races
- Intel Trace Analyzer and Collector
 - $\,\hookrightarrow\,$ MPI communications profiler and analyzer: evaluate patterns
- Intel VTune Amplifier
 - $\,\hookrightarrow\,$ Performance profiler: CPU/FPU data, mem. + storage accesses

Intel tools are licensed

All come as part of Intel Parallel Studio XE - Cluster edition!





Tools at your disposal (IV)

HPC specific tools - Scalasca & friends

Scalasca

 $\,\hookrightarrow\,$ Study behavior of // apps. & identify optimization opport.

• Score-P

 $\,\hookrightarrow\,$ Instrumentation tool for profiling, event tracing, online analysis.

• Extra-P

 $\,\hookrightarrow\,$ Automatic performance modeling tool for // apps.





Tools at your disposal (IV)

HPC specific tools - Scalasca & friends

Scalasca

 $\,\hookrightarrow\,$ Study behavior of // apps. & identify optimization opport.

• Score-P

 $\,\hookrightarrow\,$ Instrumentation tool for profiling, event tracing, online analysis.

• Extra-P

 $\,\hookrightarrow\,$ Automatic performance modeling tool for // apps.

Free and Open Source! See other awesome tools at http://www.vi-hps.org/tools





Allinea DDT - highlights

DDT features

- Parallel debugger: threads, OpenMP, MPI support
- Controls processes and threads
 - $\,\hookrightarrow\,$ step code, stop on var. changes, errors, breakpoints
- Deep memory debugging
 - $\,\hookrightarrow\,$ find memory leaks, dangling pointers, beyond-bounds access
- C++ debugging including STL
- Fortran including F90/F95/F2008 features
- See vars/arrays across multiple processes
- Integrated editing, building and VCS integration
- Offline mode for non-interactive debugging
 - $\,\hookrightarrow\,$ record application behavior and state



.



Allinea DDT - on ULHPC

Modules

- On all clusters: module load tools/AllineaForge
- Caution! May behave differently between:
 - \hookrightarrow Debian+OAR (Gaia, Chaos) and CentOS+SLURM (Iris)

Debugging with DDT

- Load toolchain, e.g. (for Intel C/C++/Fortran, MPI, MKL): \hookrightarrow module load toolchain/intel
- Compile your code, e.g. mpiicc \$code.c -o \$app 8 Run your code through DDT (GUI version)
 - \hookrightarrow iris: ddt srun ./\$app

- Gaia/chaos: ddt mpirun -hostfile \$OAR_NODEFILE ./\$app
- Q Run DDT in batch mode (no GUI, just report):
 - \hookrightarrow ddt --offline -o report.html --mem-debug=thorough







Allinea DDT - interface

Ele Edt Yew Control Tools Window					e 7.0.3 <@iris-053>				<u> </u>
		rocess C Thread E Step Thr							_
Al 0 1	2 3 4 5	6 7 8 9 10 11 1	2 13 14 15 16 17 1	8 19 20 21 22 23	24 25 26 27				
Project Ries	📧 cstartmpi.c 🖂						Loope La com	e(s) Current Stack	
Search (Ctrl+K)							Lacole	ecs) current stack	e x
	83	_Status status; /* Retu				-	Variable Name	Value	-
funclD: void	A 84 t2	= malloc(sizeof(typeThre	e]];				-argc ⊕ argv	1 	_
 func2(): int func3(): void 	85 for	(p=0;p<100;p++) bigArray[p]=80000+p;					-being/Watched	0	
 main(int argc, char** 	A 87 88						bigArray dest	-134225592	
 e print_arglconst char typeOne 	89 P2 90 P2	_Init(Garge, Gargv); Comm.cack[MPI_COMM_WORL	D. Sow rank):				dynamicArray		
 typeThree typeTap 	91 MP1 92	_Comm_nankIMPI_COMM_WORL _Comm_sizeIMPI_COMM_WORL	D, Spl;				environ	0x7mmaf38 32767	
🐵 🔳 External Code	93 dvr	namicArray = malloc(sizeo	f(int)*100000);				-message -my rank		
8- 1 / Headers	95	<pre>in = malloc(sizeof(int) +</pre>	p);				-P		
B Sources	96 8 for 97 f	(x=0;x=10000;x++)					÷s +sdm		
- M multe3.c	98 99 1	dynamicArray[x] = x%10;					- source	-1	
- 10 _powitf2.c	100						# t2	0x603010	
- 🖉 abortf.c	101 pri 102	intf('ny rank is %d\n', m	y_rank);				⊕ tables -tag		
 — III accumulate.c III accumulate.cdesc.c 	103 E for 104 {	(x=0;x<12;x++)					test		
- 🕷 accumulateř.c - 🕷 ad_aggregate.c	105	y = 0; while $[y = 12]$					e troopa		
 If ad aggregate new c 	106 E 107	{					-y	0	
- 10 ad close.c	A 108 0 109	<pre>tables[x][y] = (x+1 y += my rank + 1;</pre>)*[y+1];						
- 10 ad coll exch new.c	110)	Add breakpoint for All						
- 10 ad delete.c	112		Add tracepoint for All (y.	my_rank)					
- 🕷 ad_done.c	114 {	largc > 1 66 my_rank == 0	 Delete breakpoint for All 	1					
- If ad end c	115 116	printf("Rank %d has %d printf("They are:\n"):	ar 🜔 Edit breakpoint for All						
A and freed or	112	forfunds universities!	Pun to here			<u>×</u>	Type: none selected		
	ooints Stacks Th	acepoints Tracepoint Output	1 Undo	Ctri+Z			valuate		8×
Watchpoints			Bado	Ctri+ Shift+ Z		8 ×	Expression Value		
Processes Scope Expres				Ctri+X					
All #0 main dynam	nicArray write only	hardware	Copy	Ctri+ C					
			Paste Delete						
1			Select Al						
			Open in external editor						
1			Close	Ctrl+W					



V. Plugaru & UL HPC Team (University of Luxembourg)

.



Allinea MAP - highlights

MAP features

- Meant to show developers where&why code is losing perf.
- Parallel profiler, especially made for MPI applications
- Effortless profiling
 - $\,\hookrightarrow\,$ no code modifications needed, may not even need to recompile
- Clear view of bottlenecks
 - $\,\hookrightarrow\,$ in I/O, compute, thread or multi-process activity
- Deep insight in CPU instructions affecting perf.
 → vectorization and memory bandwidth
- Memory usage over time see changes in memory footprint
- Integrated editing and building as for DDT

Full details at allinea.com/products/map/features





Allinea MAP - on ULHPC

Modules

- On all clusters: module load tools/AllineaForge
- Caution! May behave differently between:
 - $\,\hookrightarrow\,$ Debian+OAR (Gaia, Chaos) and CentOS+SLURM (Iris)

Profiling with MAP







Allinea MAP - interface

Edit View Metrics Window		
	sses. 1 node Sampled from: Sun Jun 11 2017 21; 31:07 (UTC+02) for & Sa	Hide Netrics.
lain thread activity		
U floating-point 100		
0.16		
117 NB		_
0	of total); Main thread compute 0.0 %, MN 99.8 %, Skeping 0.2 %	Zoom 51 III
		Time spent on line 128
au_alkoallyc 🔯 🚶 🕺 osu_co	107 8 if(size > LARGE_MESSAGE_SIZE) (}	Breakdown of the 67.4% time
	111 112 disp =0;	Spent on this line: Executing instructions 0.0%
	113 # for (1 = 0; 1 < numprocs; 1++) ()	Calling other functions 100.0%
	121 122 MPI_Barrier(MPI_COMM_WORLD); 123	
	124 timer+0.0; 125 : for(i = 0; i < options.iterations + options.skip; i++) (
.45	128 t_start = MPI_Mtime(); 127	
	120 MPI_COM_ACRED); SWICCONTS_ISUISICS, MPI_CHAP, FROMONTS, FROMES, MPI_CHAP, COMP_ACRED); 130 MPI_COMP_ACRED);	
	131 t_stop = MD[_wtime(); 132	
.9\	133 H if(i=eptions.skip) ()	
	138 }	_
	<pre>140 latency = (double)(timer * 1ed) / options.iterations; 141</pre>	
	142 MPE_Peduce(&latency, Sain_time, 1, MPI_DOLLE, MPI_MIN, 0, 143 MPI_COMM_MORLD); 144 MPI_Peduce(&latency, Sain_time, 1, MPI_DOLLE, MPI_MIX, 0,	
	141 PPL/VPLCPE(starthofty, Data_Ellips, 1, PPL/DAta_L), PPL/PPL, 0	-
t/Output Project Riles M	Main Thread Stacks Functions	
hread Stacks	MPI function(s) on the	
	ê∕mên	
	67.4% MRLAltoalv 33.1% MRLaltoalv	
3% .	B 2 others	
1%	8 free buffer ⊜free buffer	
1%	Biminnap	
ng data from 1.176 samples t	s taken over 28 processes (42 per process)	Altona Forge 7.0.3 👔 Main Three
		UNIVERS



Allinea Perf. Reports - highlights

Performance Reports features

- Meant to answer How well do your apps. exploit your hw.?
- Easy to use, on unmodified applications
 - $\,\hookrightarrow\,$ outputs HTML, text, CSV, JSON reports
- One-glance view if application is:
 - $\hookrightarrow \ \textbf{well-optimized} \ \text{for the underlying hardware}$
 - $\,\hookrightarrow\,$ running optimally at the given scale
 - $\,\hookrightarrow\,$ affected by I/O, networking or threading bottlenecks
- Easy to integrate with continuous testing
 - $\,\hookrightarrow\,$ programatically improve performance by continuous profiling
- Energy metric integrated
 - \hookrightarrow using RAPL (CPU) for now on iris
 - $\,\hookrightarrow\,$ IPMI-based monitoring may be added later



Full details at allinea.com/products/allinea-performance-reports
V. Plugaru & UL HPC Team (University of Luxembourg)
UL HPC School 2017/ PS6



Allinea Perf. Reports - on ULHPC

Modules

- On all clusters: module load tools/AllineaReports
- Caution! May behave differently between:
 - $\,\hookrightarrow\,$ Debian+OAR (Gaia, Chaos) and CentOS+SLURM (Iris)
 - $\,\hookrightarrow\,$ Gaia: can collect GPU metrics
 - $\,\hookrightarrow\,$ Iris: can collect energy metrics

Using Performance Reports

- Load toolchain that you run your app. with, e.g.
 - \hookrightarrow module load toolchain/intel
- 2 Run your application through Perf. Reports
 - \hookrightarrow iris: perf-report srun ./\$app
 - Gaia/chaos: perf-report mpirun -hostfile \$OAR_NODEFILE
 ./\$app

Analysis by default in .html and .txt indicating also run config.





Debugging and profiling tools

Allinea Perf. Reports - output (I)



srun grmx, mpi mdrun -s bench_mase_cubic.tpr -nsteps 10000 1 node (28 physical, 28 logical cores per node) 126 Giß per node 28 processes, OM__NUM_THREADS was 0 tris-653 Sun Jun 11 2017 20:13:59 (UTC+02) 19 seconds mmt/risggs/saps/rself/data/production/v0.1-20170602/ default/software/bio/GROMACS/2016.3-intel-2017a-hybrid/ bin



Summary: gmx_mpi is Compute-bound in this configuration



Time spent running application code. High values are usually good. This is **average**; check the CPU performance section for advice Time spent in MPI calls. High values are usually bad.

This is average; check the MPI breakdown for advice on reducing it

Time spent in filesystem I/O. High values are usually bad. This is **negligible**; there's no need to investigate I/O performance

This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

CPU

A breakdown of the 54.6% CPU time:				
Single-core code	5.5%	1		
OpenMP regions	94.5%			
Scalar numeric ops	5.2%	1		
Vector numeric ops	44.2%			
Memory accesses	50.6%			

The per-core performance is memory-bound. Use a profiler to identify timeconsuming loops and check their cache performance.

V. Plugaru & UL HPC Team (University of Luxembourg)

MPI



Most of the time is spent in point-to-point calls with an average transfer rate. Using larger messages and overlapping communication and computation may increase the effective transfer rate.





Allinea Perf. Reports - output (II)

CPU

ne 54.6%	CPU time:
5.5%	1
94.5%	
5.2%	1
44.2%	
50.6%	
	5.5% 94.5% 5.2% 44.2%

The per-core performance is memory-bound. Use a profiler to identify timeconsuming loops and check their cache performance.

A breakdown of the 0.0% I/O time: Time in reads 0.0% Time in writes 0.0% Effective process read rate 0.00 bytes/s Effective process write rate 0.00 bytes/s

No time is spent in I/O operations. There's nothing to optimize here!

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	75.6 MiB	J.
Peak process memory usage	86.6 MiB	
Peak node memory usage	11.0%	I

The peak node memory usage is very low. Running with fewer MPI processes and more data on each process may be more efficient.

A breakdown of the 45.4% MPI time: Time in collective calls 33.5% Time in point-to-point calls 66.5%

Effective process collective rate	426 MB/s	
Effective process point-to-point rate	419 MB/s	

Most of the time is spent in point-to-point calls with an average transfer rate. Using larger messages and overlapping communication and computation may increase the effective transfer rate.

OpenMP

A breakdown of the 94.5% time in OpenMP regions:

Computation	99.5%	
Synchronization	0.5%	1
Physical core utilization	100.0%	
System load	101.9%	

OpenMP thread performance looks good. Check the CPU breakdown for advice on improving code efficiency.

Energy

A breakdown of how the 0.899 Wh was used:

CPU	100.0%	
System	not supported %	1
Mean node power	not supported W	1
Peak node power	not supported W	1

The whole system energy has been calculated using the CPU energy usage.

System power metrics: No Allinea IPMI Energy Agent config file found in (null). Did you start the Allinea IPMI Energy Agent?



19 / 30



Intel Advisor - highlights

Advisor features

- Vectorization Optimization and Thread Prototyping
- Analyze vectorization opportunities
 - $\,\hookrightarrow\,$ for code compiled either with Intel and GNU compilers
 - \hookrightarrow SIMD, AVX* (incl. AVX-512) instructions
- Multiple data collection possibilities
 - \hookrightarrow loop iteration statistics
 - \hookrightarrow data dependencies
 - \hookrightarrow memory access patterns
- Suitability report predict max. speed-up
 - $\,\hookrightarrow\,$ based on app. modeling

Full details at software.intel.com/en-us/intel-advisor-xe





Intel Advisor - on ULHPC

Modules

• On iris/gaia/chaos: module load perf/Advisor

Using Intel Advisor

Load toolchain: module load toolchain/intel Compile your code, e.g. mpiicc \$code.c -o \$app Collect data e.g. on gaia:

```
mpirun -n 1 -gtool "advixe-cl -collect survey \
-project-dir ./advisortest:0" ./$app
```

4 Visualise results with advixe-gui \$HOME/advisortest

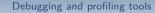




V.

Intel Advisor - interface

Bit Were Help Bit Were Help Witch With Star Witch Sta	2	/mnt/galagpfs/users/homedirs/vplugaru/scalasca-tests/hpcg-3.0/bin/advisortest - Intel Advisor <@gala-100.gala-cluster.unilux> 🗸 🔨 😢
witcom Track 0 dil Wetchick Wetchick Wetchick Wetchick Off Batch mode Wetchick Run Roothre Wetchick Collect Image: Survey & Burley & Bu	Eile View Help	
Vectorization Wetking Iterating Iterating Iteration Iteration <th>🛍 🖆 🖆 💿 🕑 📾 🕷 🕉 🏚 🔎</th> <th>8 49 1 📽 1 🔿</th>	🛍 🖆 🖆 💿 🕑 📾 🕷 🕉 🏚 🔎	8 49 1 📽 1 🔿
Workflow Workflow Summary Summary & Summ	welcome rank.0 🐹	•
Wetcontractions and subscription Address is a void winthy logs that will bendt most from vetor betteression betteression and subscriptions and subscriptions and subscriptions and characterize your memory vis. Wetcontractions and subscriptions and subscriptions and subscriptions and characterize your memory vis. Subscription Address is a void and subscriptions and subscriptions and characterize your memory vis. Wetcontractions betteression betteression and subscriptions and characterize your memory vis. Wetcontractions and PLOPS Collect Program metrics Program metrics		
 Collect C Coll	OFF Batch mode	Vectorization Advisor
Collect Constrained Sector Addition to define the sector Addition to the sector Additio	Run Roofline	 Vectorization Advisor is a vectorization analysis toolset that lets you identify loops that will benefit most from vector
 Leised Tris 247.468 Leised Tris 247.468 Vertration Set SES.SE2 Number of CPU Threads 1 Collect Collect Collec	Collect 🖿 🛌	parallelism, discover performance issues preventing from effective vectorization and characterize your memory vs. vectorization bottlenecks with Advisor Roofline model automation.
Li Hind Tip Controls and FLORS Li Hind Tip Controls and FLORS Collect I Collect II Collect III Collect IIII Collect IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	1. Survey Target	
Collection Collection <th>🕨 Collect 🍡 🖿 🛄</th> <th></th>	🕨 Collect 🍡 🖿 🛄	
Image: Control Contro Control Control Contrecontrol Control Control Control Control Con	1.1 Find Trip Counts and FLOPS	
Image: Source Sourc	Collect 🖿 📘	
Time in scalar coda 70.43 24.4% Mark Loops for Deeper Analysis state thaps in the Survey Report for the constraint of Survey Report for the consthe survey Report for the constreport for	Trip Counts	
Mark Loops for Deeper Analysis Select loops in the Survey Report Select loops in the Survey Report * The sea in the Survey Report	FLOPS	
2.1 Check Dependencies" Logo Self Time® Total Time® • Collect Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Collect Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Collect Image: Self Time® Total Time® • Collect Image: Self Time® Total Time® • Collect Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Collect Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Ollog in Computation Time Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Ollog in Computation Time Image: Self Time® Total Time® • Moting to analyze Image: Self Time® Total Time® • Platform Information Image: Self Time® Total Tim	Select loops in the Survey Report for Dependencies and/or Memory Access Patterns analysis.	Vectorized Loops Gain/Efficiency® 1.81x 19095 Program Approximate Gain® 1.58x
► Collect □ loop in ComputeRMMC_inf & ComuteRMMC_infeps01 70.42% 70.63% ► Inform (to analyze □ loop in ComputeRMMC_infe & ComuteRMMC_infeps01 70.42% 70.63% ► Collect □ loop in ComputeRMMC_infe & ComuteRMMC_infeps01 70.42% 70.63% ► Collect □ loop in ComputeRMMC_infe & ComuteRMMC_infeps01 61.57% 56.75% ► Collect □ loop in ComputeRMMC_infe & ComuteRMMC_infeps01 20.331% 77.087% ● NetWing to analyze ● Peter MinInformation Win ref 0 0 0 0 Oper Information 0 0 0 0 Oper Information 0 0 0 0 Oper Information 0 0 0 0 Oper Informatin	2.1 Check Dependencies	
La betring to analyze 2.2 Check Memory Access Pattern 2.2 Check Memory Access Pattern C loop in ComputationSci_ord at Compactivities_Indep:23 56.756 C loop in Compactivities_Indep:24 56.756 C loop in Compactivities_Indep:25 56.756 C		
2.2 Check Manary Access Patterns Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.971 20.371s 7.2.204 () Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.972 20.371s 7.7.087s Collectin () Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.972 20.331s 77.087s () Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.972 20.331s 77.087s () Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.972 20.331s 77.087s () Ioop in ComputeSMMSG_ref at ComputeSMMSG_nef.spc.972 () Ioop in ComputeSMMSG_nef.spc.972 <li< th=""><th></th><th></th></li<>		
Collection details CPU Name Intel(R) Xeon(R) CPU X5670 @ 2.93GHz Fequency 2.93 GHz CPU Name Intel(R) Xeon(R) CPU X5670 @ 2.93GHz UNIVESTIE UNIVE	2.2 Check Memory Access Patterns	
Collection declars Platform information OP Information OPU Name Information CPU Name Information InformatioInformation Information Information Information	Collect	© [loop in <u>ComputeSYMGS_ref</u> at <u>ComputeSYMGS_ref.cpp:86</u>] 20.331s 77.087s
Mini rank 0 X5570 @ 2.93GHz Image: CPU X5570 @ 2.93GHz Image: C	O Nothing to analyze	S Collection details
Operating System Linux LUXEMBOU		MPI rank 0 CPU Name Intel(R) Xeon(R) CPU X5670 @ 2.93GHz Frequency 2.93 GHz Intel(R) Xeon(R) CPU Logical CPU Count 1.2 INIV/CPUTF
CRA-Hindafie Suivey PC Team (UMIVEENSITE/Wort Leaverneeting) under UL HPC School 2017/ PS6 22 / 3		Operating System Linux LUXEMBOL
	CRe-finatize Survey	UL HPC School 2017/ PS6





Scalasca & friends - highlights

Scalasca features

- Scalable performance analysis toolset
 → for large scale // applications on 100.000s of cores
- Support for C/C++/Fortran code with MPI, OpenMP, hybrid
- 3 stage workflow: instrument, measure, analyze
 → at compile time, run time and resp. postmortem
- Score-P for instrumentation + measurement, Cube for vis. \hookrightarrow Score-P can also be used with Periscope, Vampir and Tau
- Facilities for measurement optimization to min. overhead
 - $\,\hookrightarrow\,$ by selective recording, runtime filtering

Full details at http://www.scalasca.org/about/about.html



UL HPC School 2017/



Scalasca - on ULHPC

Modules

• On iris/gaia/chaos:

module load perf/Scalasca perf/Score-P

Using Scalasca

Load toolchain: module load toolchain/foss Compile your code, e.g. scorep mpicc \$code.c -o \$app Collect data e.g. on gaia: scan -s mpirun -n 12 ./\$app Visualise results with square scorep_\$app_12_sum \hookrightarrow or generate text report: square -s scorep_\$app_12_sum \hookrightarrow ... and print it: cat scorep_\$app_12_sum/scorep.score



UL HPC School 2017/ PS6



Scalasca visualisation with Cube-P

Harder Setting	*	Cube-4.3.5: scorep_xhpcg_12_sum/summary.cubex <@gaia-100.gaia-duster.uni.lux>		(s
Observed Decoder Observed Decoder <th>e Display Plugins Help</th> <th></th> <th></th> <th></th>	e Display Plugins Help			
Mute Control System tree Syst		w Kheelida	* Absolute	
De Officie factor				BoxPlot
	Construction C	Image: State of the s	T → RATHER UND T → RAT	20.qala=cluster.t Rank 0 Rank 1 Rank 2 Rank 3 Rank 3 Rank 3 Rank 4 Rank 4 Rank 5 Rank 6 Rank 7 Rank 8 Rank 9 Rank 9 Rank 10
	ected "int ComputeMG_ref(const SparseMatrix&, const Vector&, Vec	(\$)*		
Cited 'In: ComputeMG_reliconst. SparseMatrix&, const. Vector&, Vec	ugaru & UL HPC Team (Univer	sity of Luxembourg) UL HPC Scho	ool 2017/ PS6	
ted 'int ComputeNG geliconst SpersonNational, const Vectoral," UNIVERSI UXEMBU UXIVERSI UXEMBU UXIVERSI UXEMBU		,		25 /



Summary







V. Plugaru & UL HPC Team (University of Luxembourg)





Now it's up to you

Easy right?



V. Plugaru & UL HPC Team (University of Luxembourg)





Now it's up to you

Easy right?

Well not exactly.



V. Plugaru & UL HPC Team (University of Luxembourg)



Now it's up to you

Easy right?

Well not exactly. Debugging always takes effort and real applications are never trivial.



V. Plugaru & UL HPC Team (University of Luxembourg)



Now it's up to you

Easy right?

Well not exactly. Debugging always takes effort and real applications are never trivial.

But we do guarantee it'll be /easier/ with these tools.



V. Plugaru & UL HPC Team (University of Luxembourg)



Conclusion and Practical Session start

We've discussed

A couple of small utilities that can be of big help
HPC oriented tools available for you on UL HPC

And now ..

Short DEMO time!



V. Plugaru & UL HPC Team (University of Luxembourg)



Conclusion and Practical Session start

We've discussed

A couple of small utilities that can be of big help
HPC oriented tools available for you on UL HPC





V. Plugaru & UL HPC Team (University of Luxembourg)

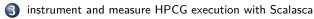


Hands-on start

• We will first start with running HPCG (unmodified) as per:

http://ulhpc-tutorials.rtfd.io/en/latest/advanced/HPCG/

- ... your tasks:
 - perform a timed first run using unmodified HPCG v3.0 (MPI only)
 - ✓ use /usr/bin/time -v to get details
 - $\checkmark~$ single node, use \geq 80 $\,$ 80 for input params (hpcg.dat) $\,$
 - vun HPCG (timed) through Allinea Perf. Report
 - \checkmark use perf-report (bonus points if using iris to get energy metrics)



- Remember: pre-existing reservations for the workshop:
 - \hookrightarrow 'hpschool': Iris cluster resv. (use --reservationname=hpcschool)
 - \hookrightarrow 4248619: Gaia cluster regular nodes (use -t inner=4248619)
 - $\hookrightarrow~$ 4248620: Gaia cluster GPU nodes
 - \hookrightarrow 1614176: Chaos cluster





Thank you for your attention...

Questions?

http://hpc.uni.lu

High Performance Computing @ UL Prof. Pascal Bouvry Dr. Sebastien Varrette & the UL HPC Team (V. Plugaru, S. Peter, H. Cartiaux & C. Parisot) University of Luxembourg, Belval Campus Maison du Nombre, 4th floor 2, avenue de l'Université L-4365 Esch-sur-Alzette mail: hpc@uni.lu





Debugging and profiling tools





30 / 30

V. Plugaru & UL HPC Team (University of Luxembourg)