

UL HPC School 2017 PS4b: Debugging, profiling and performance analysis

UL High Performance Computing (HPC) Team

V. Plugaru

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



V. Plugaru (University of Luxembourg)

Latest versions available on Github:



UL HPC tutorials: UL HPC School:

https://github.com/ULHPC/tutorials

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

PS4b tutorial sources:

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





2 / 21

V. Plugaru (University of Luxembourg)



Introduction

Summary



2 Debugging and profiling tools





V. Plugaru (University of Luxembourg)

Introduction



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 problems

During the hands-on session you will:

- see what happens when an application runs out of memory and how to discover how much memory it actually requires.
- use debugging tools to understand why your code is crashing.
- use profiling tools to understand the (slow) performance of your code - and how to improve it.

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





Summary









V. Plugaru (University of Luxembourg



Tools at your disposal (I)

Common tools used to understand problems

- Do you know what time it is?

 → /usr/bin/time -v is just magic sometimes
- Don't remember where you put things?
 → 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 DDT (part of Allinea Forge)

 → Visual debugger for C, C++ and Fortran threaded and // code

 Allinea MAP (part of Allinea Forge)

 → Visual C/C++/Fortran profiler for high performance Linux code

 Allinea Performance Reports

 → Application characterization tool





Tools at your disposal (II)

HPC specific tools

- Allinea DDT (part of Allinea Forge)
 → 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 will be integrated in SLURM
- \hookrightarrow ... so your jobs will be able to wait for it to be available



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

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





Allinea DDT - on ULHPC

Modules

• On iris: module load tools/AllineaForge

- On gaia/chaos: module load Allinea/Forge
 - $\,\hookrightarrow\,$ we'll synchronize the software set to match iris soon

Debugging with DDT





LUXEMBOUR 9 / 21



Allinea DDT - interface

Fig. Edt View Control Tools Window	Halo		Allinea	DDT - Allinea Fo	rge 7.0.3 <@iris-053>			23 8
		m						
	Int & Group C	Property C Thread C Star Three						
			1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2 (52) (57) (57) (5	N [52] [52] [52] [53]			
Create Group			TO THE TO TO TA TO TO		0 24 20 20 27			
Project Bles #X	Contartenei e 🖸	ci -					Locals Dirrect line	(a)]. Current Stack]
Search (Chilak)	· Ustaturigent 🔼	1					Locals	ex conercacack
	82 1	Pl_Status status; /* Petur	n status for receive */			-	Variable Name	Value
func10 void	▲ 84 t	<pre>2 = malloc(sizeof(typeThree</pre>)];				-argc	1
- • func20 : int	. 05 f	or (p=0;p<100;p++)					-beingWatched	0
- e main(int argc, char++	89	bigArray[p]=80000+p;					⊕ bigArray	104005500
 print_arg[const char* print_arg[const char* 	89	Pl_Init(Garge, Gargv);	for and le				e dynamicArray	
 typeThree 	91	PI_Com_size[MPI_COMM_WORLD	, 6p1;				@ environ	0x700008f38
- Cyperwo	92 93 d	vnamicArray = mallocisizeof	(int)*100000):				-message	
a = 1	94 5	dim = malloc(sizeof(int) *	pl;				-my_rank	28
B I Sources	96 B f	or(x=0;x<10000;x++)					0.5	Cx.D
dwtc3.c	97 1	dynamicArray[x] = x510;					e source	
- M powtf2.c	99 }						🔅 status	
- (#) abort.c	101 p	rintf('my rank is Ad\n', my	_nank);				e tables	
- M accumulate.c	102 103 H f	or(x=0:x<12:x++)					-tag	
- I accumulate_cdesc.c	104 {	x = 0:					⊕ troopa	
- Ma ad_aggregate.c	106 8	while(y != 12)					CX .	°
- 10 ad_aggregate_new.c	107	tables[x][y] = (x+1)	*(v+1):				11	
- M ad_coll_build_req_new.c	0 169	y += my rank + 1;	177 and the second second second the			_		
- 10 ad_col_exch_new.c	111 }	,	Add tracepting for All figures of	ank)				
- I ad_delete.c	112	f(argc > 1 66 my rank == 0)						
- R ad_done_fake.c	114 {		Delete breakpoint for All					
- 10 ad end.c	116	printf("They are:\n");	e ot breakpoint for set			*1		
	119	forfuelt untract tak!	Pun to here				Type: none selected	
Input/Output Breakpoints Watchpoil	nts Stacks	Tracepoints Tracepoint Output	Undo			5	Valuate	8 ×
Watchpoints			Bado			- #×	Expression Value	
Processes Scope Express	on (Trigger (in impremented in						
e no main dynamic	erray write on	ry riardinarë	Copy					
			Dekte					
			Select All					
			Open in external editor					
1			Close	Ctrl+W				



UNIVERSITÉ DU

10 / 21

.



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



11 / 21



Allinea MAP - on ULHPC

Modules

- On iris: module load tools/AllineaForge
- On gaia/chaos: module load Allinea/Forge

Profiling with MAP







Allinea MAP - interface

N. Contraction of the second se	osu alitoaliv 28p 1n 2017-06+11 21+31.map - Allinea MAP - Allinea Forge 7.0.3 <@iris+0)53>	E 31 X
Ele Edit View Metrics W	ndow (Help Income Lands, Samplet from Sin An 11 2017 21:31:07.0 (France) for 9 Se		Lide Metrice
Main thread artikity			
Main Cireau accivicy			
CPU floating-point	10		
0.%			
Memory usage			
117 MB			
21:31:12-21:31:13 (0.832s,	9.8% of total): Main thread compute 0.0 %, MM 99.8%, Skeping 0.2%		Zoom 📲 🎟 💿
🗴 osu_alkoalko 🔯 🛛 🗴	osu_colic 🗉		Time spent on line 128 Ø ×
	107 H if(size > LAPGE_MESSAGE_SIZE) ()	×	Breakdown of the 67.4% time spent on this line:
	disp =0;		Executing instructions 0.0%
			Calling other functions 100.0%
	for(i = 0) i < options.iterations + options.skip; i++) (
(7) (A)			
07.49	120 PL ACCENTISATION AND ANTIONIS, PARADON, PL CHEN, PACHOL, PACHOL, PLANES, PL CHEN, 120 MPL COMP, MCR.D);		
	130 t_stop = MPI_wtime();		
	is if(imoptions.skip)		
1.35 .	137 PPE_Barrier(PPE_CDMP_WDPLD); 138)		
	<pre>140 latency = (double)(timer * la6) / options.iterations;</pre>		
	141 142 MPT_Reduce(&latency, Emin_time, 1, MPT_DOLBLE, MPT_MIN, 0,		
	143 MPT_Deduce(Statency, Emax_time, 1, MPT_DOLBLE, MPT_MAX, 0, 144 MPT_Beduce(Statency, Emax_time, 1, MPT_DOLBLE, MPT_MAX, 0,		
	145 PPT COMPARED;	<u>×</u>	
Input/Output Project File	s Main Thread Stacks Functions		
Nain Thread Stacks			8×
Total core time	V [MP] Function(s) on the		
67.4%	67.4% MethyAltoolay		
1.3%	1.3% MP[Barrier		
<0.1%	i fore buffer		
	Broundap		
<0.1%			
Showing data from 1.176 sa	mples taken over 28 processes (42 per process)		Allinea Force 7.0.3 2 Main Thread Vew
			uni.
			UNIVERSITÉ
			LUXEMBOU

13 / 21



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 well-optimized 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





UL HPC School 2017



Allinea Perf. Reports - on ULHPC

Modules

- On iris: module load tools/AllineaReports
- On gaia/chaos: module load Allinea/Reports

Using Performance Reports







Allinea Perf. Reports - output (I)



srun gmx, mpi mdrun -s bench_mase_cubic.tpr -nsteps 10000 1 node (28 physical, 28 logical cores per node) 126 GiB per node 28 processes, OME_NUM_THREADS was 0 1ris-053 Sun Jun 11 2017 20:13:59 (UTC+02) 19 seconds mmt/risgt/saps/rself/data/moduction/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 th	e 54.6%	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.

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)

A breakdown of	the 54.	.6% C	PU 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%	

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.

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%	i.
Effective process read rate	0.00 bytes/s	I.
Effective process write rate	0.00 bytes/s	T.

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 Peak process memory usage 86. Peak node memory usage

0 11110	
6 MiB	
1.0%	1

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

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?



UL HPC School 2017



Summary



2 Debugging and profiling tools





V. Plugaru (University of Luxembourg)

UL HPC School 2017

.



Now it's up to you

Easy right?



V. Plugaru (University of Luxembourg)



Now it's up to you

Easy right?

Well not exactly.



V. Plugaru (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 (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.



UL HPC School 2017

▲



Conclusion and Practical Session start

We've discussed

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

And now..

Short DEMO time!





Conclusion and Practical Session start

We've discussed

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

And now..

Short DEMO time!

Your Turn!



V. Plugaru (University of Luxembourg)



Thank you for your attention...

Questions?

http://hpc.uni.lu

The UL High Performance Computing (HPC) Team

University of Luxembourg, Belval Campus: Maison du Nombre, 4th floor 2, avenue de l'Université L-4365 Esch-sur-Alzette mail: hpc@uni.lu





2 Debugging and profiling tools





21 / 21

V. Plugaru (University of Luxembourg)