Intel software tools



Intel Parallel Studio

In the following slides we will give an overview of the tools avalaible from the Intel Parallel Studio.

You can use these tools on the Cineca HPC machines using

[faffinit@r000u18l02 faffinit]\$ module load intel/pe-xe-2016--binary [faffinit@r000u18l02 faffinit]\$ source \$INTEL_HOME/parallel_studio_xe_2016.3.067/bin/psxevars.sh mpsvars.sh: Warning: Hardware events collection is disabled by default. To enable it, run mpsvars.sh with --vtune (recommended) or --papi option. Copyright (C) 2009-2016 Intel Corporation. All rights reserved. Intel(R) Inspector XE 2016 (build 460803) Copyright (C) 2009-2016 Intel Corporation. All rights reserved. Intel(R) VTune(TM) Amplifier XE 2016 (build 463186) Copyright (C) 2009-2016 Intel Corporation. All rights reserved.



Preliminarly...

Load compiler and Intel MPI:

module load intel
module load intelmpi
(if needed) module load mkl

SuperComputing Applications and Innovation

Poisson

Check the Makefile:

[faffinit@r000u18l02 Poisson-C-0.6.1.4]\$ cat Makefile VERSION=0.6.1.4 #DEBUG=-g -debug inline-debug-info DEBUG=-gOMP=-gopenmp #SIMD=-DSIMD #ISA=-xAVX #ISA=-xCORE-AVX2 #ISA=-xHOST #ISA=-xMIC-AVX512 MPI=-DUSE MPI #REPORT=-gopt-report5 ADD FLAGS= DEFINES=-DDOUBLE -DUSE MPI



SuperComputing Applications and Innovation

Level-0 MPI profiling

```
export I_MPI_STATS=ipm
mpirun -np 36 ./poisson.x
```

check stats.ipm

```
export I_MPI_STATS=1
mpirun -np 36 ./poisson.x
```

check stats.txt



I_MPI_DEBUG

[0]	MPI	<pre>startup():</pre>	Rank	Pid	Node name	Pin	сри
[0]	MPI	<pre>startup():</pre>	0	11530	r045c01s03	0	
[0]	MPI	<pre>startup():</pre>	1	11531	r045c01s03	1	
[0]	MPI	<pre>startup():</pre>	2	11532	r045c01s03	2	
[0]	MPI	<pre>startup():</pre>	3	11533	r045c01s03	3	
[0]	MPI	<pre>startup():</pre>	4	11534	r045c01s03	4	
[0]	MPI	<pre>startup():</pre>	5	11535	r045c01s03	5	
[0]	MPI	<pre>startup():</pre>	6	11536	r045c01s03	6	
[0]	MPI	<pre>startup():</pre>	7	11537	r045c01s03	7	
[0]	MPI	<pre>startup():</pre>	8	11538	r045c01s03	8	
[0]	MPI	<pre>startup():</pre>	9	11539	r045c01s03	9	
[0]	MPI	<pre>startup():</pre>	10	11540	r045c01s03	10	
[0]	MPI	<pre>startup():</pre>	11	11541	r045c01s03	11	
[0]	MPI	<pre>startup():</pre>	12	11542	r045c01s03	12	
[0]	MPI	<pre>startup():</pre>	13	11543	r045c01s03	13	
[0]	MPI	<pre>startup():</pre>	14	11544	r045c01s03	14	
[0]	MPI	<pre>startup():</pre>	15	11545	r045c01s03	15	
[0]	MPI	<pre>startup():</pre>	16	11546	r045c01s03	16	
[0]	MPI	<pre>startup():</pre>	17	11547	r045c01s03	17	
[0]	MPI	<pre>startup():</pre>	18	11548	r045c01s03	18	
[0]	MPI	<pre>startup():</pre>	19	11549	r045c01s03	19	
[0]	MPI	<pre>startup():</pre>	20	11550	r045c01s03	20	
[0]	MPI	<pre>startup():</pre>	21	11551	r045c01s03	21	
[0]	MPI	<pre>startup():</pre>	22	11552	r045c01s03	22	
[0]	MPI	<pre>startup():</pre>	23	11553	r045c01s03	23	
	MDT	atastus/1.	24	44664	-045-01-03	24	

export I_MPI_DEBUG=4



SuperComputing Applications and Innovation

Analysis of the Poisson code

Objectives:

- demonstrate features of Intel Trace Analyzer and Collector (ITAC) and VTune Amplifier
- find the root cause for suboptimal scaling
- show ways of tuning where the tools indicate suboptimal performance



Simple scaling analysis

Just give a look to the code running with different number of tasks/threads (up to 2 nodes, for example) Speedup S is defined as S[p]=T[1]/T[p]

Efficiency E is defined as E[p]=S[p]/p

In the ideal case: S[p]=p and E[p]=1



Before running..

Don'f forget:

- to load the intel modules (and intelmpi)
- to source the psxe.vars
- check the resources you're running on with cpuinfo



cpuinfo

[faffinit@r000u17l01 ~]\$ cpuinfo Intel(R) processor family information utility, Version 5.1.3 Build 20160120 (build id: 14053) Copyright (C) 2005-2016 Intel Corporation. All rights reserved. ===== Processor composition ===== Processor name : Intel(R) Xeon(R) E5-2695 v3 Packages(sockets) : 2 Cores : 28 Processors(CPUs) : 56 Cores per package : 14 Threads per core : 2 ===== Processor identification ===== Processor Thread Id. Core Id. Package Id.

 It's a standard problem (e.g. heat equation).

- We will investigate a square 3600x3600 computational grid. It's on the edge of the bandwith limitations.
- Grid points will be distributed to MPI ranks to a 2D process grid. The cartesian process grid is a feature of the Poisson solver. Other programs can have different data distribution.



Optimal grid?

CINECA

Which is the optimal choice for the grid partitioning?



Grid and performances



SuperComputing Applications and Innovation

Measure MPI times with ITAC

[faffinit@r000u17l01 ~]\$ qsub -I -A cin_priorit -l select=1:ncpus=36:mpiprocs=36 -l walltime=15:00 qsub: waiting for job 63431.r000u17l01 to start qsub: job 63431.r000u17l01 ready

[faffinit@r040c03s01 ~]\$ module load intel intelmpi [faffinit@r040c03s01 ~]\$ source /cineca/prod/opt/compilers/intel/pe-xe-2016/binary/itac/9.1.2.024/bin/itacvars.sh mpsvars.sh: Warning: Hardware events collection is disabled by default. To enable it, run mpsvars.sh with --vtune (recommended) or --papi option. [faffinit@r040c03s01 ~]\$ cd /marconi_scratch/userinternal/faffinit/Poisson-C-0.6.1.4 [faffinit@r040c03s01 Poisson-C-0.6.1.4]\$ mpirun -trace n 16 ./poisson.x

```
Residuum after 751 iterations = 0.004992
Compute time = 1.543324 [sec]
Perf = 3.406284 [GFlops]
[0] Intel(R) Trace Collector INFO: Writing tracefile poisson.x.stf in /marconi_scratch/userinternal/faffinit/Poisso
n-C-0.6.1.4
```





- finding the right path in your case can be troublesome: if you can, use the which or the tab autocompletion...
- use OMP_NUM_THREADS=1
- by default, ITAC creates a .stf file for each process. It results in a high number of files. To overcome this problem use export VT_LOGFILE_FORMAT=STFSINGLE
- after the data collection is completed, you will need a graphical interface to visualize with ITAC (-> use RCM)









Flat Profile Load B	alance Call Tr	ee Call Gr	aph						
Jame	TSelf	TSelf	TTotal	#Calls	TSelf /Call	TTotal /Call	#Procs	TSelf /Proc	TTotal /Proc
- MPI_Comm_rank MPI_Finalize MPI_Isend MPI_Irecv MPI_Waitane MPI_Waitane MPI_Barrier MPI_Barrier MPI_Allreduce	264e-6 s 19.994e-3 s 483.588e-3 s 430.316e-3 s 409e-6 s 651.898e-3 s 33.078e-3 s 722.051e-3 s	1	264e-6 s 19.994e-3 s 483.588e-3 s 430.316e-3 s 409e-6 s 651.898e-3 s 33.078e-3 s 722.051e-3 s	16 16 84280 84280 32 48160 16 12032	16.5e-6 s 1.24962e-3 s 5.73787e-6 s 5.10579e-6 s 12.7812e-6 s 13.5361e-6 s 2.06737e-3 s 60.0109e-6 s	16.5e-6 s 1.24962e-3 s 5.73787e-6 s 5.10579e-6 s 12.7812e-6 s 13.5361e-6 s 2.06737e-3 s 60.0109e-6 s	16 16 16 16 16 16 16	16.5e-6 s 1.24962e-3 s 30.2242e-3 s 26.8947e-3 s 25.5625e-6 s 40.7436e-3 s 2.06737e-3 s 45.1282e-3 s	16.5e-6 1.24962e-3 30.2242e-3 26.8947e-3 25.5625e-6 40.7436e-3 2.06737e-3 45.1282e-3

SuperC

MPI times

The total wall clock run time T[p], is given by the time spent in MPI plus time spent in computation:

T[p]=T_comp[p]+T_mpi[p]

Speedup and efficiency can be calculated for the compute time separately:

S_comp[p]=T_comp[1]/T_comp[p]=T[1]/T_comp[p]





Source of the MPI time

ITAC shows timing of all MPI routines used by an application The timing of MPI routines may be due to network transfer times caused by interconnect bandwith limitations (latencies) The other possibility are waiting times caused by the algorithm: load imbalance or dependencies



A network model

- Latency L is defined as the transfer time for a 0 byte message
- Bandwidth BW is defined as the transfer rate for asymptotically large messages
- Message volume V is the data amount sent

```
The transfer time is:
T_trans[V] = L + (1/BW)*V
```

uting Applications and Innovation

ITAC ideal network simulator

It is extremely complicated to simulate a realistic network!

- An extreme case the ideal network may be simulated by setting all transfer times to 0. This would mean L=0 and BW=∞ for the simple model
- ITAC offers an ideal network simulation with transfer times set to zero. Compute times (non MPI) will stay the same An existing real trace file is used as basis for the simulation



ITAC ideal network simulator

With a perfectly balanced algorithm the total MPI time will be vanishing in the ideal case

- In most real cases the MPI time will just shrink but not vanish
- The remaining part is due to waiting time e.g. when the receiver is starting to receive before the sender is ready to send

Start simulator with:

Advanced -> Idealization



Intel® Trace Analyzer 🕑								
<u>F</u> ile <u>O</u> ptions <u>P</u> roject <u>W</u> indows <u>H</u> elp								
<mark>™</mark> 1:/galileo/home/userinto View <u>C</u> harts <u>N</u> avigate	ernal/faffinit/poisson.x.single.stf <u>A</u> dvanced Layout			and the second				<u> </u>
📰 <u>ï</u> 💷 📩 0.000	 Tagging Filtering Reset Tagging/Filtering 	- 2	All_Proces	sses 🔀 MPI e>	kpanded in (Major	Function (Groups) 🔗 🍸	🗼 🕕 🖄 »
Flat Profile Load B	Command line for VTune Amplifier Command line for Advisor							
Name	🐱 Idealization	I	#Calls	TSelf /Call	TTotal /Call	#Procs	TSelf /Proc	TTotal /Proc
 All_Processes Group Application MPI_Comm_size MPI_Comm_rank MPI_Finalize MPI_Isend MPI_Isend MPI_Irecv MPI_Wtime MPI_Waitall MPI_Barrier MPI_Allreduce 	 Imbalance Diagram Seek and Jump (ITF only) Process Aggregation Function Aggregation Default Aggregation Show Process Group 'Other' Show Function Group 'Other' 722.051e-3 s 	.8.7541 s 226e-6 s 264e-6 s .994e-3 s .588e-3 s .316e-3 s 409e-6 s .898e-3 s .078e-3 s 2.051e-3 s	16 16 16 84280 84280 84280 84280 84280 84280 16 16 12032	1.02576 s 14.125e-6 s 16.5e-6 s 1.24962e-3 s 5.73787e-6 s 5.10579e-6 s 12.7812e-6 s 13.5361e-6 s 2.06737e-3 s 60.0109e-6 s	1.17213 s 14.125e-6 s 16.5e-6 s 1.24962e-3 s 5.73787e-6 s 5.10579e-6 s 12.7812e-6 s 13.5361e-6 s 2.06737e-3 s 60.0109e-6 s	16 16 16 16 16 16 16 16	1.02576 s 14.125e-6 s 16.5e-6 s 1.24962e-3 s 30.2242e-3 s 26.8947e-3 s 25.5625e-6 s 40.7436e-3 s 2.06737e-3 s 45.1282e-3 s	1.17213 s 14.125e-6 s 16.5e-6 s 1.24962e-3 s 30.2242e-3 s 26.8947e-3 s 25.5625e-6 s 40.7436e-3 s 2.06737e-3 s 45.1282e-3 s

×

≡ Þ.



4

Waiting time due to dependencies



CINECA

The simulated MPI time for the ideal network may be regarded as the waiting time T_wait due to imbalance and dependencies: T_mpi = T_transfer + T_wait

After generation of an ideal trace file the result can be displayed in the Imbalance Diagram:

Advanced Application -> Imbalance Diagram





Imbalance diagram

The imbalance diagram displays the relation of transfer to wait time. Due to the result we can decide how to proceed with tuning:

- Transfer time (Interconnect) dominates: the algorithm is balanced but we have to improve the network performance by e.g. different process placement or new network hardware
- Waiting time (Imbalance) dominates: the algorithm has to be revisited e.g. better load balancing. New network hardware or better process placement will not help!



Testing different grids







Compute time almost equal for 2D 24x16 and 1D 384x1 process
 grid. Row vectors are long enough. 3600/16 = 225 for 24x16
 process grid and 3600 grid points for 384x1 process grid

SuperComputing Applications and Innovation



Compute time for 1x384 is almost 3X longer probably because of short vector length 3600/384 < 10



Imbalance time best for 2D because process grid fits perfectly: local grid = (3600/24) x (3600/16) = 150 x 225 grid points

SuperComputing Applications and Innovation



Imbalance time for 384x1 slightly worse because number of local grid point rows will vary between 10 and 9 (3600/384 = 9.375). See next slide(s) for a discussion about the measurement of imperfect data distribution



Imbalance time for 1x384 is even larger because of longer compute time. The imbalance stretches with compute time

Global load imbalance

A portion of the waiting time is normally due to Global Load Imbalance. The Global Load Imbalance is measured by determining the maximum per rank and average compute time over all processes:

T_load = T_compute_max - T_compute

= T_mpi - T_mpi_min

T_load is the time we may win by achieving a perfect load balance. It should be lower than the previously calculated MPI time for an ideal network (= T_wait = Imbalance/p)


Load imbalance: MPI for 1D 384x1

Intel® Trace Analyz	er - [1: C:/Users/hbo roject Windows	ckhor/Projects/ClusterToolsTrai Help	ninson_resul	ts/Trainin	gPXT/02_ITAC/1_D_nx1/p	oissonITC.x_16_PPN24_P384_V1.stf]	- # x	
	0.030 399 - 0.056 90	9 : 0.026 510 Seconds	- 🖳 AL	Processes	5 Major Function Grou	os 🤌 🍸 🗼 🚯 🏠 🔟 🛛	2.96	
The Day Res								
Hat Profile Loa	d Balance	e Cali Graph						
Children of All_Processes	•						Show Pies	
Name	▲ TSelf	TSelf TTota	sl #(Calls	TSelf/Call		-	
Process 121	11.83e-3 s	11	.83e-3 s	1300	9.1e-6 s			
Process 122	11.738e-3 s	11.	738e-3 s	1300	9.02923e-6 s			
Process 123	11.712e-3 s	11.	712e-3 s	1300	9.00923e-6 s			
Process 124	11.568e-3 s	11.	568e-3 s	1300	8.89846e-6 s			
Process 125	11.521e-3 s	11.	521e-3 s	1300	8.86231e-6 s			
Process 126	11.579e-3 s	11.	579e-3 #	1300	8.90693e-6 #	Deples 0 1	10.1	0
Process 127	11.514e-3 s	11.	514e-3 s	1300	8.856938-6 #	Ranks 0-1	43: I	UX3600 boints
Process 128	11.776e-3 s	11.	776e-3 s	1300	9.058460-6 #			
Process 129	11.90/6-3 5		387e-3 5	1300	9.22077e=6 8	Danke 11/	1 202	· 0v2600 pain
Process 130	11.700e-3 s		300e-3 s	1200	3.221098-0 5	Raliks 144	+-202	. 925000 poin
Process 131	11 7000-2 4		7000-2 -	1200	9.075290-5 -			
Process 132	11 9240-2 4		9240-2 =	1200	9 190-6 -	(10*144+9)	3*740	1 = 3600
Process 134	11 7440-3	11	744-3 -	1300	9 033850-6 .		<i>2</i> 2 1 0	7 - 3000
Process 135	11 7598-3	11	759=-3 =	1300	9.045398-6 #			
Process 136	11.624e-3	11	624e-3 #	1300	8.941548-6 8			
Process 137	11.4740-3	11.	474-3 =	1300	8.826164-6 #			
Process 138	11.6250-3	11.	625e-3 #	1300	8,942314-6 8			
Process 139	11.6330-3 :	11.	633e-3 #	1300	8.9484			
Process 140	11.844e-3 s	11.	844e-3 #	1300	119-6 s			
Process 141	11.916e-3 s	11.	916e-3 a	1000	9.16616e-6 s			
Process 142	12.051e-3 s	12.	0518 3 8	1300	9.27e-6 s			
Process 143	11.9e-3 s	1	1.9e-3 s	1300	9.15385e-6 #			
Process 144	12.778e-3 #	12.	778e-3 #	1300	9.82923e-6 #	The second s	10000000000	CONTRACTOR AND A DESCRIPTION OF A DESCRI
Process 145	13.268e-3 #	13.	268e-3 #	1300	10.2062e-6 s	Minimal	MDI	can be found b
Process 146	13.027e-3 s	13.	027e-3 =	1300	10.0208e-6 s	1-mmaa		
Process 147	13.156e-3 s	13.	156e-3 s	1300	10.12e-6 s	- 12 - 1 - K		TOULOU
Process 148	12.92e-3 s	12	.92e-3 #	1300	9.93846e-6 s	CLICKIN	ig on	i Self (Column
Process 149	13.05e-3 s	13	.05e-3 #	1300	10.0385e-6 #		0	
Process 150	13.03e-3 a	13	.03e-3 #	1300	10.0231e-6 #		anda	r) - cort
Process 151	13.009e-3 s	13.	009e-3 #	1300	10.00690-6 #		reaue	Sort
Process 152	13.122e-3 =	13.	122e-3 =	1300	10.0938e-6 s	1610		
Process 153	13.358e-3 s	13.	358e-3 s	1300	10.2754e-6 s			
Process 154	13.357e-3 #	13.	357e-3 #	1300	10.27468-6 #	100 m		
Process 155	13.272e-3 s	13.	272e-3 #	1300	10.20928-6 #			
Process 156	13.0480-3 1	13.	0486-3 #	1300	10.03698-6 #			
Process 157	13.3050-3 1	13.	3056-3 #	1300	10.23468-6 8			
Frocess 158	13,104e-3 s	13.	1048-3 3	1300	10.086-6 5		_	

SuperComputing Applications and milliovation

Split of timings

We have now all components of our split of timings:

- = T_compute + T_mpi
 - = T_compute + T_trans + T_wait
 - = T_compute + T_trans + T_load + T_depend

The Imbalance diagram shows only the second line but we might additionally compute T_load and T_depend for a deeper analysis. T_depend is called Dependency time. This is just the rest of the imbalance time T_wait that is not due to the Global Load Imbalance.



Т

Refined imbalance diagram





Message passing profile

Message passing profile displays various characteristics of message passing in a sender/receiver Matrix

Charts -> Message Profile

The Matrix element N,M corresponds to the message passing characteristics from rank N to rank M. Change these attributes by:

Right Click -> Attribute to show

Characteristics are: total message volume, message passing time, max, min, average rate and count







Process aggregation

For 16 nodes (384 ranks on IVB) the total Message Passing profile is not very handy

- We may fuse the communication to compute node level. In this case 384 ranks are fused to 16 compute nodes:
- Advanced Process -> Aggregation
- This will pop up a new window: check All_Nodes and apply



Total volume 2D vs 1D distribution



SuperComputing Applications and Innovation

Average rate: 2D vs 1D distribution



Number of messages: 2D vs 1D dist



SuperComputing Applications and Innovation

Message profile: observations

Inter node communication has about the same volume in the 2D case but 16x more messages are sent

- There is just a single inter node message per boundary exchange in the 1D case (3 exchanges per iteration times 100 iterations == 300 messages)
- Communication rate drops so much in the quadratic 2D case that the total transfer time (Imbalance diagram) is almost equal for both configurations



Optimization hints

- A compromise between quadratic and 1D processor grid may be more appropriate here like 48x8 or 96x4. This will reduce the number of inter node messages and raise the bandwidth for each message
- The default rank to node mapping is just linear. This leads to alternating communication patterns (see following slides)
- A better mapping can be achieved by putting all ranks of a rectangular sub process grid onto a single node. The following slides explain the ranks to node mapping



Default mapping for 24x16 grid





SuperComputing Applications and Innovation

Optimal mapping

0	***	3	4	 7	8		11	12	 15
16		19	20	 23	24		27	28	 31
32	2	35	36	 39	40		43	44	47
48		51	52	 55	56		59	60	48
64		67	68	 71	72		75	76	79
80		83	84	 87	88	•••	91	92	95
96		99	100						

Node #0 Node #1 Node #2 Node #3

This 6x4 pattern can be repeated for all nodes. The number of processor boundary lines between nodes are: 4 (vertical) and 6 (horizontal)



Impact of mapping





Detailed visualization of MPI programs

After some global evaluations we may dive now into the MPI algorithm by showing the temporal evolution with ITAC

- Most programs consist of recurring patterns like iterations or different phases: initialization, computation and I/O
- Quantitative timeline shows nicely coarse patterns:
- Charts -> Quantitative Timeline



Quantitative timeline



SuperComputing Applications and Innovation

SuperComputing Applications and Innovation

After identification of basic patterns we may now change to the more detailed Event Timeline

- Event timeline is the most important Chart in ITAC
- Temporal development reveals root causes of dependencies due to suboptimal implementations
- Charts -> Event Timeline



Single iteration

0	Application	APPENApplication ME	I RecMPApplication	MPL Recy MApplication	MPI_Alreduce		as used before in
1	MEApplication	A Application	PL Ret Application	MPI_Recv MPI_Applicate	MPI_Allreduce		as used before in
	MPApplication		PI R MApplication	MPI_Recv VM Applicati	MPI_Allreduce		the manning
	MPApplication	MPI 00 Application	PERCAN Application	MPI Recy Applicati	MPI_Alreduce		the mapping
e.	MPApplication	And Application	PL RC G- Application	MPI Recy MApplicatio	n MPI_Alizeduce		section: 24x16
	MEApplication	22 No Application	PL R CAP Application	MPI_Recv Applicatio	MPI_Alreduce		Section. 24x10
6	MEApplication	ME Replication	PI RATINA PI Recv Monto	ation MP 411 Recv	Application MPL Alt	reduce	Applic
	MEIApplication	M STAPL RecApplication	PI Recy AV MPI Re	Application API R	Application M	PI_Alreduce	Applic
	MEApplication	AP Inter Manager	PI Recy	Application A Market	Application M	PI_Alreduce	Applic
	MPApplication	Application	MP. LAN MARME	RecApplication PI Re	Application M	PI_Alreduce	Applic
0	MPApplication	R R AMApplication	MP Ger MAX	cation VELACITY A	Application MPLAIn	reduce	Applic
1	MP Application	Ret Application	MET TECH YEARY	Application M215 CV	PICApplication	educe	
12	MPIApplication	Market Research	BERGENER	Application	MPLA MPLA	lireduce	the second se
13	MPIApplication	M 2 Ro of Mark Application	MPLATICIES	Application (1994)	MPLA Mplication	dreduce	This is the default
4	MPIApplication	N 23 Ret MASSApplication	MELS SITUR	Application	MPL	Alkeduce	
15	MPI Application	2 Rite // Application	MPLEXICATION	Application Application	Application MPL	Alkeduce	mapping!
6	MPLApplication	STRATE And Cation	A API Recy MINING	on MF Recy WWW	MPI_Allecture_		
7	MPI Application	See A second cation	1/PL Recy 14 2 Contra	MF Recv N	MPL Alkeda	K6	
8	MPIApplication	Parallel Mitching Station	PI Recy CARL P	Kapon MPLRET	MPL All	reduce	Appl
9	MPI Application	Manual Contraction	MPI Receipt WAIM	cation MPI 2051	MPL All	keduce	Appl
0	MPLApplication	A State of the second second	PI Recy	MPI REC III	Application MPL All	reduce.	laa
1	MPI Application	Part And	MPI Recy III VI MIS	Net Station	MPI_All	reduce.	Appl
2	METApplication	A state of the second s	MPI Recviller	API Recy VI	Part Application MPL Alle	reduce	Appl
3	MPI Application		ication (1000000000000000000000000000000000000	Application	Mar Place Application	MPI_Alkeduce	Appl
4	MPT_Application	VII III Feer MMPL Re	Application	Application	ALANPA Lec # APL RApplication	MPI_Allreduce	Ad
ie i	MPL Application	ANTEIN COL	tion NUMBER OF	Application ////	Part Colling Application	MPI_Alreduce	and the second s



Boundary exchange in ideal networks

MPI times in the ideal network case are due to global load imbalances and dependencies

Dependencies are e.g. due to order of blocking sends and receives

The current naive implementation of the boundary exchange uses blocking sends and receives: MPI_Send, MPI_Recv The Ideal network simulation helps to clearly identify dependencies



Boundary exchange in ideal networks



SuperComputing Applications and Innovation

SuperComputing Applications and Innovation

Optimization hints

- Some of the dependencies may be resolved by using MPI_Isend and MPI_Irecv with an MPI_Waitall()in the end
- In a first step we may just exchange the blocking Sends/Recvs by the immediate routines and place a MPI_Waitall() at the end.
- Data copies of boundary arrays have to be done after the wait routine
- In a second step we may optimize the order of MPI routines and data copies. Some requests may be ended by a separate MPI_Wait()



Comparing ITAC traces

Compare before and after optimization e.g. compare boundary exchange with blocking Send/Receive to non blocking Send/Receive

Further potential comparison scenarios:

- Compare ideal to real trace
- Compare different number of ranks
- Compare different mappings

Comparing ITAC traces

Open tab: View → Compare

Compare	poissonITC.x	16	PPN24	P384_V	0.ideal.single.stf	ŧ
---------	--------------	----	-------	--------	--------------------	---

Select tracefile to compare the original one with, and click "OK". Resulting Comparison View will calculate the exact differences and speedups. Press F1 for more details.

Compare poissonITC.x_16_PPN24_P384_V0.ideal.single.stf with

O 1: C:/Users/hbockhor/Projects/ClusterToolsTraining/Poisson_results/TrainingPXT/02_ITAC/2_D/poissonITC.x_16_PPN24_P384_V0.ideal.single.stf

2: C:/Users/hbockhor/Projects/ClusterToolsTraining/Poisson_results/TrainingPXT/06_EXCHANGE1_ITAC/2_D/poissonITC.x_16_PPN24_P384_V0.ic

Choose another tracefile...

OK

Cancel

X

Open another file for a comparison



SuperComputing Applications and Innovation

View Charts Newigate Advanced Leyout Comparison Team A R R 10 1044 500 - 0.044 500 : 0.000 012 Seconds V AL Processes V Major Function Groups V X X 12 V X New Charts R 10 14 500 - 0.041 400 : 0.000 011 Seconds V X AL Processes V Major Function Groups V X X 12 V X Advisation Application Appl	Trace A: Send/Recv Trace B: Isend/Irecv/Waitall
44 Aukatoria periodian periodian 54 Sukatoria periodian periodian 55 Sukatoria periodian periodian 56 Sukatoria periodian periodian 57 Sukatoria periodian periodian 58 Sukatoria periodian periodian 59 Sukatoria periodian sukatoria 50 Sukatoria periodian sukatoria 50 Sukatoria periodian sukatoria 50 Sukatoria periodian sukatoria 51 Sukatoria periodian sukatoria 52 Sukatoria periodian sukatoria 53 Sukatoria periodian sukatoria 54 Sukatoria periodian periodian 55 Sukatoria periodian periodian 56 Sukatoria periodian periodian 57 Sukatoria periodian periodian 58 Sukatoria periodian periodian 59 Sukatoria periodian periodian 50 Sukatoria periodian periodian 50 Sukatoria <td>Wait Time on P6 has vanished! IPI Wait Time for this boundary exchange shrunk by a factor 0.689</td>	Wait Time on P6 has vanished! IPI Wait Time for this boundary exchange shrunk by a factor 0.689
Marholik Laad Sakes Cd Yee Al_Processes SrA A TSelf TSelf TIotal #C	

Instrumentation of user functions

So far, we only see MPI routines and Application time inside ITAC traces

Navigation becomes far more easy when adding user functions

For evaluation of the impact of optimization we may want to see the timing of the boundary exchange including all its MPI calls



ITAC compiler instrumentation

All source files or just the files of interest may be compiled with the *-tcollect* flag (Intel compiler only)

The executable has to be linked using this flag, as well

As an alternative (different compiler or code blocks that are not a function) we might consider to use the ITAC API functions for instrumentation.



User functions

Str.	Profile Load Balance Co cesses T	Tree Cal Graph	TSelf/Call	And and an	det alred ve
max max <th>Profile Load Balance Ca cesses V</th> <th>Titee Call Graph</th> <th>TAN<u>as</u></th> <th>And a social sec</th> <th>Not Alredical</th>	Profile Load Balance Ca cesses V	Titee Call Graph	TAN <u>as</u>	And a social sec	Not Alredical
ray of a set	ry Fy Profile Load Balance Ca	Tree Cal Graph		A the second second	yder, Alfredurel Mer, Alfredurel
Style Availability Availabi				11 No. 1 Anna State Stat	MPL Alcolum
Style	94 94 101		- CAL	111 Boddum	MDT Alreduse
Style Mark					MPLAked
Style Max Max<		A ST ST WIRED		Price Discourses and	
Stry av av av av av av bv		1.1 MARY		TIM 2	PELAKETIF
Stry All All </td <td>and the second sec</td> <td></td> <td></td> <td></td> <td>PET Alreduse</td>	and the second sec				PET Alreduse
City Mail Mail Mail Mail Mail Mail Mail Mail Mail City Mail Mail <t< td=""><td></td><td>and the second second</td><td>A Bullet</td><td>and American</td><td>Met Alreduc</td></t<>		and the second second	A Bullet	and American	Met Alreduc
City Mail	12	A PA Street	ol2 Lav black	and the state side up	MPI_AI
Stry All All </td <td></td> <td>11 A States</td> <td>////www.bladk</td> <td>A Resolution</td> <td>MELA</td>		11 A States	////www.bladk	A Resolution	MELA
Start	4		RI Wolack	10 Million Residuum	MELAkeduc
Strip No. No. No. No. No. No. Strip Strip Strip Strip Strip Strip Strip Strip Strip Strip Strip Strip Strip Str	1		IN N	A DAVESOLUM	Ministration
Corp All Not All Not All Corp All All All All All All Corp All All All All All All All All Corp All All All All All All All All Corp All All All All All All All All Corp All All A					ALC ALCONTRACT
any of a set		and the second	Aller	and the state of the second	MEL shad of
any of a set		A syled	1 Isack	MER / siduum	PI_Airedure
Cary And	1	A Marshared	black	MA Sofresiduum	MPI_Albeduce
Stry Mark Mark Stry Mark	,	¥ 91 0 000	2011 (black	A Azveskuum	PI_Akeriure
Carry Araban Parlan Carry Araban Parla Carry Araban		Philosophed	M PMPplack	A Providence siduum	WPI_Alkeduce
arry and a set of a s			PH V DESC	The south	Met_alteduce.
		MIRENA.	ALL MARK	IIM WAY	
nany and the second second Second second		AND A DECEMBER OF	and the second		Allecture
appy and the second		Ward Reat Tred	MP 1 Al plack	as a star residuum	MPL Alterior
	1	and the second s	MP Istack	All 17 17 State residuum	MPI_Alteria
7 copy / 2014 The Arrest Aken Kit. Akenture	1	MRN RALL MANNED	A Skillack	esid.um	API_Akedure
397 A Martin Carlo			AND A		Alteraire
		ACIMIN			
		11 allowed	the black	El Martin Brench an	MPL alkedure
	(1 In the	diff black	19 Periduum	MPI Alteriur
topy Alexandrow Ale	1	Rept of P1 Wed	a black	Min Minesiduum	MPI_Alreduce
copy Contract Market Market		Contract from	Mark Bark	Aldenie residuani	Alreduce
1957 Ared P 29,453. Aredam 992, Aredam 992, Aredam		and the second s	a subject	internal and a second using	MPI_Alkeduie
			and the second s	eaction	CPT ADDRESP

Instrumented user functions like exchange0 can improve analysis of the MPI algorithm

Exchange routine is on bottom of the list. But the total time TTotal also contains all MPI functions. This time exceeds the Allreduce time

SuperComputing Applications and Innovation

SuperComputing Applications and Innovation

CINEC/

Intel VTune Amplifier XE

- We used ITAC for the analysis of the message passing algorithm
- We already saw that computation performance saturated on a single node
- With this tool we may have a closer look to the processor performance and program structure
- VTune Amplifier XE based analysis can be started and performed by its GUI. Together with MPI on a Cluster which probably prefers batch usage, we will use the command line interface



Hotspot analysis

This is the most basic analysis type to start an investigation

The analysis will present hotspots of the calculation for a chosen MPI rank. Timings go down to source lines or assembly code

The Call Stack provides information about how the function is called and how much time is due to this branch



Collecting data

This analysis may be conducted for each of all 384 ranks but probably we may concentrate on a single rank first:

mpirun	-n	1 amplxe-clresult-dir hotspots \
		collect hotspots poisson.x :\
	-n	383 poisson.x

Hotspot analysis is performed on rank 0 and results are stored in directory hotspots.0. All other ranks run poisson.x without analysis. More complex selection of ranks are possible building groups of ranks doing analysis or not



Collecting data

A new syntax:









3 different stacks for MPI_Waitall. Source line of call to exchange in poisson.c is shown. Exchange is called 3 times!

<no current="" project=""> - Intel VTune Amplifier Image: Ima</no>	hotspots_38 ×			1 2
Basic Hotspots Hotspots by CPU	Usage viewpoint (<u>change</u>) 🕐	Ir	ntel VTune Amplifier XE 2015	Click on first
4 🐵 Analysis Target 🚊 Analysis Type 🔛	Collection Log 🛛 🛍 Summary 🔹 Bottom-up 🔹 Caller/Cal	llee 🗳 Top-down Tree 🔣 Tasks and	d Frames 🔹 comm.c 🔹 poissor 🕨	Check off mist
Source Assembly	Assembly grouping: Address	•	Data Of Interest (CPU Metrics)	stack and
So. Li. Source 205 206 #ifdef USE_MPI 207 // boundary exchange 208 209 ≥ exchange(\$p, \$gr); 210 #endif 211 212 213 // RED update	CPU Time: Total by Utilization Idle Poor Ok Ideal Over 20.001ms	CPU Time: Self by Utili Idle Poor Ok Idea Oms Oms	☆ Viewing 《 1 of 3 ▶ selected stack(s) 40.0% (0.020s of 0.050s) iibmpi.so.12.0!PMPI_Waitall - waitall.c poissonAXE.s/exch367d - comm.c:526 poissonAXE.s/exch1960 - comm.c:307 poissonAXE.s/exch1960 - comm.c:307 poissonAXE.s/main+0xeb - poisson.c:219 libc-2.12.so!_libcUnknown]:{Unknown	poisson.c line: shows first cal to exchange at line 209
214 215 red(sp,sgr); 216 217 #ifdef USE_MPI 218 // boundary exchange 219 220 exchange(sp,sgr); 221 #endif 222				Second call at line 220 shows up by selecting
223 // BLACK update 224 225 black(sp, sgr); 226 227 #ifdef USE_MPI 228 // boundary exchange 229 230 exchange(sp, sgr); 231 #endif 233 Si	elected 1 row(s): 20.00			another stack
	+) P		

Advanced hotspots

Hotspot identification using directly the Performance monitoring Unit (PMU). Needs special drivers realized by kernel modules (root rights necessary for installation)

- Exchange **hotspots** by **advanced-hotspots** in previous command line
- Instructions retired is the basic indicator for processor utilization.

Maximum is 4 simultaneous instructions per clock-tick.

The output shows CPI: clock-ticks per Instruction. 4 simultaneous
<no current="" project=""> - Intel VTune A</no>	Amplifier	
	Welcome advanced-h X	E
Advanced Hotspots Hot	spots viewpoint (<u>change</u>)	
Analysis Target Analysis	Type a Collection Log 🐮 summary 💀 Bottom-up 🌆 Caller/Callee 🚺 In	op-down tree E Tasks and Frames we nomprort.s
 Elapsed Time: 1.37 	85 4	
CPU Time: Instructions Retired:	0.3515	Instructions Datiradi completed Instructions
CPI Rate:	0.686	instructions Retired: completed instructions
CPU Frequency Ratio:	0.999	CDI Data Clack Ticks par Instruction
Overhead Time:	0s	CPT Rate: Clock Ticks per Instruction
Spin Time:	0s	CDITERS AND DATES AT TURE REPORT
Ton Hotspots		CPU Frequency Ratio: >1 : Turbo boost!
This section lists the most act	ive functions in your application. Optimizing these hotspot functions typically result	
Function CI	PU Time [©]	
residuum	0.058s	
MPIDI CH31 Progress	0.055s	
intel_ssse3_rep_memcpy	0.048s	
black	0.047s	
[Others]	0.091s	
✓ CPU Usage Histogram	m 4a	Second routine comes from MPI – Progress
		Engine
Collection and Platfo	orm Info 🐁	Engine
This section provides informa	tion about this collection, including result set size and collection platform data.	Mana MDI internal functions about
Application Command Line:	prg.sh	More MPT Internal functions shown
Operating System:	2.6.32-358.6.2.el6.x86_64.crt1 Red Hat Enterprise Linux Server release 6.5 (Santiago)	
MPI Process Rank:	0	
Computer Name:	esg046	
Collection start time:	11:07:12 13/08/2014 UTC	
Collection stop time:	11:07:14 13/08/2014 UTC	
Name: Frequency:	Inter(K) Xeon(K) E3/E7 V2 processor 2.7 GHz	
Logical CPU Count:	48	

<no current="" project=""> - Intel VTune Amp</no>	olifier									
	Velcome advanced-h X							atol VTuna	Amo	lifer VE 201
Advanced Hotspots Hotsp	ots viewpoint (change)							inter v rune	Amp	ITTEL XE 201
🕘 Analysis Target 🛛 🛱 Analysis Typ	e 🔚 Collection Log 🖬 Summary 🚳	Bottom-up	🕯 Calle	r/Callee	🔹 Top-down	Tree 🔣 Tasks and	Frames 🔹 libr	npifort.s		
Grouping: Function / Call Stack										
Function / Call Stack	CPU Time by Utilization *	Instructio Retired	Ov. [™] an.	CPI Rate	CPU Frequency Ratio	Module	Fi	unction (Full)		C.
🗄 residuum	57.917ms	132,300,000	0ms	1.224	1.038	poissonAXE.x	residuum			
MPIDI_CH3I_Progress	54.947ms	321,300,000	0ms	0.504	1.095	libmpi.so.12.0				
⊞ red	53.462ms	234,900,000	0ms	0.586	0.955					
⊞_intel_ssse3_rep_memcpy	47.522ms	102,600,000	0ms	1.055						
1 black	46.779ms	270.44	VIIIS	0.400	1.029	poissonAXE.x	black			compute.c
MPID_nem_mpich_test_recv	25.246ms	121,500,000	0ms	0.756	1.350	libmpi.so.12.0	MPID_nem_mp	pich_test_recv		mpid_nem_ini
■ MPID_nem_dapl_rc_poll_dyn_opt_20	11.138ms	32,400,000	0ms	1.167	1.260	libmpi.so.12.0	MPID_nem_da	ID_nem_dapl_rc_poll_dyn_opt_20		dapl_poll_rc.c
MPID_nem_network_poll	9.653ms	0	0ms	0.000	0.935	libmpi.so.12.0	MPID nem network poll			mpid_nem_ne
MPIDL CH3_iSendy	8.168ms	5,400,000	0ms	3.000	0.736	libmpi.so.12.0	MPIDI_CH3_iSe	PIDI CH3 iSendy		ch3_isendv.c
MPIDU Sched progress	4.455ms	29,700.000	0ms	0.000	0.000	libmpi.so.12.0	MPIDU Sched progress		mpid sched.c	
MPIDU Sched are pending	3.713ms	48,600,000	0ms	0.111	0.540	libmpi.so.12.0	MPIDU Sched	are pending		mpid sched.c
I MPI intel ssse3 rep memcpy	2.970ms	5,400.000	0ms	2.500	1.687	libmpifort.so.12.0	I MPI intel	I MPI intel ssse3 rep memcpy		
Outside any known module1	2.970ms	0	0ms	0.000	0.675		(Outside any k	nown module]		
Il exchangel	1.485ms	10,800,000	0ms	0.500	1.350	poissonAXE.x	exchangel			comm.c
Fifunc@0x3032	1.485ms	2 700 000	0ms	2.000	1.350	libmb4-rdmav2.so	func@0x3032			
EMPIDI CH3 iStartMsg	1.485ms	0	Oms	0.000	1.350	libmni.so.12.0	MPIDI CH3 iStartMsg		ch3 istartmsn.c	
Selected 1 row(s):	25.246m	\$ 121,500,000	0ms	0.756	1.350		111 101 01 10 010	are read		cito_data introduc
4 🗔 🔰 🕴	e [h	100000000			-			
poissonAXE.x (TID: 7554	i 0.2s 0.3s 0.4s 0.3	s 0.6s		75	0.85 0.95	15 1.1	5 1.25	1.35	200	Thread Running Las CPU Time Catalog Overhead a
	paused							paused.		CPU Time
CPU Time	paused				_		_	paused		
e								1 2	•	
No filters are applied.	Process: Any Process Three Three	ad- Any Threa	d		Mo	dules Any Module		 Utilization: 4 	Any Litik	zation 💌

Order of Hotspot functions changes due to: 1. better time resolution 2. Internal MPI functions are displayed

🗑 Ad	vanced Hotspots Hotspots viewpoint (change)	0				Intel VTune Amp	lifier X	E 20
1 💿	Analysis Target 🕺 Analysis Type 📰 Collection Log 🖬 S	ummary 🔹 Bottom-up 崎 Ca	ller/Callee	Top-d	own Tree 🔣 Tasks	and Frames 🕞 libmpifort.s 🔂 comput	te.c 38	
Sour	ce Assembly 🗉 🖽 🗟 🥪 🧐 🐑	Assembly grouping: Address						
So. Li.▲	Source	CPU Time by Utilizatia *	Address 🔺	Sour Line	Assembly	CPU Time by Utilization * 🖻	Instruct Retired	ov. an.
228	int n = gr->lrow;		0x4058d8	244	addpd %xmm5, 1			
229	int m = gr->lcol;		0x4058.dc	244	addpd %xmm6, %			
230			0x4058e0	244	addpd %xmm7, %			
31	double** x_new = gr->x_new;		0x4058e4	241	add \$0x8, %r12			
32	double** x_cur = gr->x_cur;		0x4058e8	241	cmp %rbp, %r12	2		
33			0x4058eb	241	jb 0x405870 <8			
234	double resid = 0.0, resid_tmp;		0x4058ed		Block 17:			
235			0x4058ed	241	jmp 0x40594d <			
236	<pre>#pragma omp parallel for private(i,j) reduction</pre>		0x4058ef		Block 18:			
237	for(i=1; i< n+1; i++)		0x4058ef	241	nop			
238	#ifdef SIMD		0x4058f0		Block 19:			
239	<pre>#pragma simd reduction(+:resid)</pre>		0x4058f0	243	movapax 0x8(Oms	2,700,000	0
240	#endif		0x4058f6	243	movapax 0x18	2.970ms	8,100,000	0
241	for(j=1; j <m+1; j++)<="" td=""><td>3.713ms</td><td>0x4058fc</td><td>243</td><td>movapsx 0x28</td><td>3.713ms</td><td>5,400,000</td><td>0</td></m+1;>	3.713ms	0x4058fc	243	movapsx 0x28	3.713ms	5,400,000	0
242	{		0x405902	243	movapsx 0x38	2.970ms	10,800,	. Or
243	<pre>double diff = x_new[i][j] - x_cur[i][j];</pre>	33,414ms	0x405908	243	subpdx 0x8(%)	6.683ms	10,800,	. Or
244	resid += diff*diff;	20.791ms	0x40590f	243	subpdx 0x18(6.683ms	10,800,	. Or
245	3		0x405916	243	subpdx 0x28 (4.455ms	10,800,	. 01
246			0x40591d	243	subpdx 0x38(5.940ms	8,100,000	0
247	#ifdef USE_MPI		0x405924	244	mulpd %xmm4, %	11.880ms	18,900,	. 01
248	resid_tmp = resid;		0x405928	244	mulpd %xmm5, %	1.485ms 🔜	2,700,000	0
249	MPI_Allreduce(sresid_tmp,sresid,1,MPI_DOUB		0x40592c	244	mulpd %xmm6, %	2.970ms	8,100,000	0
250	#endif		0x405930	244	mulpd %xmm7, %	1.485ms	2,700,000	0
251			0x405934	244	addpd %xmm4, 4	2.228ms	8,100,000	0
252	#1fdef TIMING_SYS		0x405938	244	addpd %xmm5, %	Oms	5,400,000	0
253	timer_end(4);		0x40593c	244	addpd %xmm6, 4	Oms	5,400,000	0
254	#endif		0x405940	244	addpd %xmm7, 4	0.743ms	2,700,000	0
255			0x405944	241	add \$0x8, %r12	3.713ms	2,700,000	0
256	return resid;		0x405948	241	cmp %rbp, %r12	A State Stat		
257			0x40594b	241	jb 0x4058f0 <			
258	1	8	0x40594d		Block 20:			
			0x40594d	244	addpd %xmm2, 4			
	Selected 1 row(s):				Highlighted 43 ro	33.414ms	70,200,	01
	4 (m) +	4 m +			4	4		

.

Bandwith analysis

The speedup curve for a single node shows saturation for more than 12 ranks per node (24 cores per node in total)

Intel[®] VTune Amplifier XE provides a Bandwidth analysis for proving this assumption

We concentrate on total bandwidth which can be related to the bandwidth that is delivered by the STREAM benchmark (~80GB/s on IVB dual Socket)



BW: Bottom-up sequential run



CINECA SuperComputin

Efficiency vs BW



SuperComputing Applications and Innovation

CINECA SUBJECT SCAI

Optimization hints

Bandwidth can be reduced by combining copy and residuum routine. This is possible because residuum is at the end and copy at the beginning of a new iteration

Bandwidth reduction may only have and impact in the bandwidth limited regime that we observe for this grid size only for less than 4 nodes

Prefetching of data may also improve performance in the copy and reduction loop

A blocked loop structure for the iteration loop may also improve data

reuse



Summary

Some methodologies were presented for performing a MPI analysis

ITAC offers interesting new features like simulation of ideal traces and the computation of transfer and waiting time Intel® VTuneTM Amplifier XE analyzes the compute part of the application. Bandwidth analysis is useful for many HPC applications

