Profiling your application with Intel VTune at NERSC















VTune background and availability



- Focus: On-node performance analysis
 - Sampling and trace-based profiling
 - Performance counter integration
 - Memory bandwidth analysis
 - On-node parallelism: vectorization and threading
- Pre-defined analysis experiments
- GUI and command-line interface (good for headless collection and later analysis)
- NERSC availability (as the vtune module)
 - Edison (Dual 12-core Ivy Bridge)
 - Babbage (Dual 8-core Sandy Bridge + Dual Xeon Phi)







- Use the Cray ${\tt cc}~or~{\tt ftn}$ wrappers for the Intel compilers
- Suggested compiler flags:
 - –g : enable debugging symbols
 - - O2 : use production-realistic optimization levels (not O0)
- To use VTune on Edison, you have to:
 - Run within a <u>CCM job</u> (batch or interactive)
 - Use <u>dynamic linking</u> if profiling OpenMP code (-dynamic)
 - Use a working directory on a <u>Lustre \$SCRATCH filesystem</u>

```
edison09:BGW > ftn -dynamic -g -O2 -xAVX -openmp bgw.f90 -
o bgw.x
edison09:BGW > mkdir $SCRATCH/vtune-runs
edison09:BGW > cp bgw.x $SCRATCH/vtune-runs/
edison09:BGW > cd $SCRATCH/vtune-runs/
edison09:vtune-runs > qsub -I -q ccm_int -1 mppwidth=24
wait ...
```





- Once you're in a CCM job (either interactive or batch script)
 - cd to your submission directory
 - Launch VTune to profile your code <u>on a compute node</u> with aprun

```
CCM Start success, 1 of 1 responses
nid02433:~ > cd $PBS_O_WORKDIR
edison09:vtune-runs > module load vtune
nid02433:vtune-runs > aprun -n 1 amplxe-cl -collect
experiment_name -r result_dir -- ./bgw.x
```

- amplxe-cl is the VTune CLI
 - -collect : specifies the collection experiment to run
 -r : specifies an output directory to save results
 - -r: specifies an output directory to save results
- Set OMP_NUM_THREADS and associated aprun options (-d, -S, -cc depth, -cc numa_node) as needed
- Results can be analyzed by launching amplxe-gui and navigating to the result directory (preferably in NX)







• Available on Edison and Babbage (SNB + Xeon Phi)

nid02433:vtune-runs > aprun -n 1 amplxe-cl -collect gener
al-exploration -r ge_results -- ./bgw.x

- Detailed characterization of relevant performance metrics throughout your application
 - Default: low-level detail aggregated into summary metrics
 - Mouse-over for explanation of their significance
 - Can be used to characterize locality issues, poor vectorization, etc.
- Multiple "viewpoints" available:
 - Direct access to hardware event counters
 - Spin / sync overhead for OpenMP threaded regions





Experiments: General exploration



<no current="" project=""> - Intel VTune Amplifier <@edison05></no>	
🖆 😥 🖡 🖉 🕼 🗃 🕜 Welcome 🛛 ge 🛛 🗙	
eneral Exploration General Exploration viewpoint (<u>change</u>) ®	Intel VTune Ampli
Collection Log 😫 Analysis Target 🔺 Analysis Type 🔹 Summary 🔩 Bottom-up 🗟 Top-down Tree 🗮 Tasks and Frames	
Elapsed Time: [©] 60.181s 🖻	
<u>Clockticks:</u> 185,122,277,683	
Instructions Retired: 139,162,208,743	
<u>CPI Rate:</u> ⁽²⁾ 1.330 The CPI may be toollish. This cauled by issues such as memory stalls instruction starvation, branch misprediction or long	
The CF may be do may in this could be caused by issues such as menuty statis, instructions, Explore the other hardware-related metrics to identify what is causing high CPI.	
MUX Reliability: 0.989	
Paused Time: [®] Os	
Filled Pipeline Slots: ¹⁰	
⊗ <u>Retiring</u> : [©] 0.215	
Microcode Sequencer: 0.004	A whole lot of
Seneral Retirement: [™] 0.211	A WHOLE IOU OF
Unter: U.003 This metric represents a non-floating-noint (EP) yon fraction the CPU has executed. If your application has no EP	summary metrics!
operations, this is likely to be the biggest fraction.	•••••••
This metric represents an overall arithmetic floating-point (FP) uops fraction the CPU has executed.	
FP Scalar 0.000	
This metric represents an arithmetic floating-point (FP) vector uops fraction the CPU has executed. Make sure	
vector width is expected.	
⊗ Bad Speculation: [®] 0.002	
⊙ Unfilled Pipeline Slots (Stalls): [©]	
Back-End Bound: 0.774	
Identify slots where no uOps are delivered due to a lack of required resources for accepting more uOps in the back-end of the pipeline. Back-end metrics describe a portion of the pipeline where the out-of-order scheduler dispatches ready uOps into their respective execution units, and, once completed, these uOps get retired according to program order. Stalls due to data-cache misses or stalls due to the overloaded divider unit are examples of back-end bound issues.	
 memory bound to be a second sec	
where pipeline could be stalled due to demand load or store instructions. This accounts mainly for incomplete in-flight	
memory demand loads that coincide with execution starvation in addition to less common cases where stores could imply body processing on the pipeling	
● L3 Bound: [©] 0.101	
⊘ DRAM Bound: [©] 0.346	
This metric shows how often CPU was stalled on the main memory (DRAM). Caching typically improves the latency	
and increases performance.	
Memory Bandwidth: 0.262 This matric shows how often CPU could be stalled due to approaching bandwidth limits of the main memory	
(DRAM). Consider improving data locality in NUMA multi-socket systems.	
Memory Latency: [®] 0.603	
This metric shows how often CPU could be stalled due to the latency of the main memory (DRAM). Consider	
PARTMENT OF Office of	



Experiments: General exploration

U.S. DEPARTMENT OF

Office of

Science



📮 🕥 <no current="" project=""> - Intel VTur</no>	ne Amplifier ·	<@edison05>												\odot) (X)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1															
General Exploration General Exploration viewpoint (change) @ Intel VTune Amplifier XE 2015															
🚳 🖾 Collection Log 🚇 Analysis Target 🗍 Analysis Type 🖿 Summary 🚱 Bottom-up 🗞 Top-down Tree 🖻 Tasks and Frames									Þ						
Grouping: Source Function Stack														\$	··· Q
		*							Filled Pipe	eline Slot	Filled Pip	eline Slot.	Unfilled F	ipeline Slo	ts (Stal
Source Function Stack	Clockticks: Total	Clockticks: Self	Instructions Retired: Total	Instructions Retired: Self	CPI Rate: Total	CPI Rate: Self	MUX Reliabili Total	MUX Reliabili Self	Retiring	Bad Spec	Retiring	Bad Spec	Back-Ei Mem Bound	nd Bound B D Core Bound	Eron Bou
マ y Total	100.0%	0	100.0%	0	1.302)							
▼ > hackakernel	98.7%	0	97.8%	0	1.315]							
▼ □ [Loop at line 127 in hackakernel]	98.1%	0	97.4%	0	1.312]							
≂ □ [Loop at line 168 in hackakernel]	94.3%	18,000,027	91.0%	0	1.349]							=
▼>[Loop at line 172 in hackakernel]	94.3%	134,000,201	91.0%	126,000,189	1.349	1.063									
⊐ [Loop at line 179 in hackakernel]	94.2%	164,920,247,380	90.9%	122,224,183,336	1.349	1.349									
▶ □ [Loop at line 194 in hackakernel]	3.7%	8,000,012	6.1%	4,000,006	0.774	2.000				1					
▶ > [Loop at line 140 in hackakernel]	0.2%	2,000,003	0.3%	2,000,003	0.953	1.000		_							
▶ > [Loop at line 252 in hackakernel]	0.0%	12,000,018	0.0%	24,000,036	0.500	0.500							_	_	
▶ > [Loop at line 71 in hackakernel]	0.3%	0	0.2%	0	2.161				<u> </u>	<u> </u>					
▶ >> [Loop at line 73 in hackakernel]	0.3%	0	0.2%	0	2.047										
▶ > [Loop at line 67 in hackakernel]	0.0%	0	0.0%	0											
▶ > [Loop at line 68 in hackakernel]	0.0%	0	0.0%	0	0.000				_		_				
▶ >svml_log4_e9	1.3%	2,224,003,336	2.2%	2,942,004,413	0.756	0.756									
Selected 1 row(s):	100.0%	0	100.0%	0	1.302		0.994	Ļ	0.221	0.00	2		0.48	3 0.40	5 1-
<	<				111			1							>
્ય ્- ્+્ 5s	10s	15s 20s	259	30s	35s	4()s	45s	50s		55s	60	✓ Thread		\checkmark
bgw.x (TID:															
													M C Rur	ining	
							_						Mardwa	are Event C	ount
													CP 📲	U_CLK_UN	HAL 🗸
		Filter	by pro	cess				Shov	v looj	ps as			🗹 Hardwar	e Events	
eac			، ، امممما ا					المبيد				=	🗹 Hardwa	are Event C	ount
<u>È</u>		and u	nread i					weii	asiu	ncuo	ns				нагы
							L								
									· · · ·						
			J.												
Hardware															
		4.00(1)	1		La an Ann at'		l talla a data				in and a second s	> »			18.
	.x 🚩 [9	4.0%] DGW.X (IID :	. [94.0%] [yw.x 🚩	ser junction	15+1		ode: jon		ops and fi	Inctions	<u> </u>			1



Experiments: General exploration

U.S. DEPARTMENT OF

Office of

Science



 ☐ ○ <no current="" project=""> - Intel VTune</no> ▲ △ <no current="" project=""> - Intel VTune</no> ▲ △ ○ Welcome ④ General Exploration Hardwa ▲ □ Collection Log ④ Analysis Target ▲ A Grouping: Call Stack 	ge × ge × are Event Counts viewpoint (change) Analysis Type M Summary Analysis Type M Summary	Change hardwa analysis	viewpoint to get re counters, hotsp , and more	to pot	© ● ● ● ■ ■ I VTune Amplifier XE 2015 ↓ ○ ••• ●	
Function Stack	SIMD FP 256.PACKED SINGLE	SIMD FP 256.PACK	ED DOUBLE	UOPS EXECUTED.THREAD		COMP OPS EXE.SSE S
∽ > Total	36,000,054	52,920,079,380		171,720,257,580	0	
≂ > hackakernel	0	52,056,078,084		161,568,242,352	0	
▼∍[Loop at line 127 in hackakernel]	0	52,056,078,084		160,704,241,056	0	
▼ □ [Loop at line 168 in hackakernel]	0	50,796,076,194		149,004,223,506	0	
▶ □ [Loop at line 172 in hackakernel]	0	50,796,076,194		149,004,223,506	0	=
▶ > [Loop at line 194 in hackakernel]	0	1,116,001,674		11,160,016,740	0	
▶ > [Loop at line 140 in hackakernel]	0	144,000,216		504,000,756	0	
▶ □ [Loop at line 252 in hackakernel]	0	0		36,000,054	0	
▶ > [Loop at line 131 in hackakernel]	0	0		0	0	
▶ > [Loop at line 71 in hackakernel]	0	0		504,000,756	0	
▶ □ [Loop at line 73 in hackakernel]	0	0		360,000,540	0	
▶ □[Loop at line 68 in hackakernel]	0	0		0	0	
▶ □ [Loop at line 67 in hackakernel]	0	0		0	0	
▶ > [vmlinux]	0	0		6 444 009 666	0	
▶ svml log4 e9	36 000 054	864 001 296		3 708 005 562	0	
▶ □ [pt/rpc]	0	0		0	0	
≥ svml log4	0	0		0	0	
≥strinlog i	0	0		0	0	
Selected 1 row(s):	0 36.000.05	4	52 920 079 380	171 720 25	7 580	
	S 30,000,000	T	52,520,075,50	171,720,23	,,500	>
	1ŷs <u>1</u> 5s <u>2</u> ŷs <u>2</u> 5	s 3ýs 3	5s 4ộs	, , 45s , 5φs , .	<u>55</u> 5	60:
Hardware ★ No filters are applied. ♀ Any Process	Any Thread	V User functions +		e: on V Functions only		₩₩₩ CPU_(▼





- Available on Edison and Babbage (Xeon Phi only)
 - Caveat: avoid Babbage SNB for now (node will lock up)

nid02433:vtune-runs > aprun -n 1 amplxe-cl _collect bandwi
dth _r bw_results -- ./bgw.x

- Gives DRAM read / write traffic as a function of time during program execution
- Useful to first calibrate with a well-understood code on the same platform (e.g. STREAM)
- Can help determine whether your code is at least partially (effectively) BW bound





Experiments: Bandwidth



<no current="" project=""> - Int</no>	tel VTune Amplifier <@edison05>			\odot \otimes (3
🛅 💿 🕨 🖉 🕐 🗃	Welcome bw.stream ×			
andwidth Bandwidth	viewpoint (change) 💿			Intel VTune Amplifier XE 2015
Collection Log 🙆 Analysis T	arget 🛆 Analysis Type 📓 Summary 🥵 B	ottom-up		
Elapsed Time: 73.3	320S			
CPU TIME: Instructions Retired: 41	827.7865 2 366 618 549			
CPI Rate: [®]	5.621			
The CPI may be too high.	This could be caused by issues such as me	mory stalls, instruction starvatio	n, branch misprediction or long	
Paused Time [®]	ne the other hardware-related metrics to ider	itity what is causing high CPI.		
r dabed mile.				
Average Bandwidth	<u>م</u>			
Package Bandwidth, GB/se	ec Read Bandwidth, GB/sec Write Bandv	idth, GB/sec	Average BW listed	
package 0 49.	288 29.850	19.439	hy CDU nackage	
ackage 1 0.	.088 0.046	0.042	ру СРО раскаде	
0.217155 Good				
Collection and Plat	form Info 🖻			
This section provides information of the section of	tion about this collection, including result se	size and collection platform da	ta.	
Application Command Line: User Name:	sfrench			
Operating System:	3.0.101-0.31.1_1.0502.8394-cray_ari_c VERSION = 11 PATCHLEVEL = 3	SUSE Linux Enterprise Server	11 (x86_64)	
	1			





BERKELEY LAB

Experiments: Bandwidth



O <no current="" project=""> - Intel VTune Amplifier <@ediso 1 2 0 Welcome bw.stream ×</no>	n05>		C)penMP reរ្	gions ⊗ ⊛ ⊗
Bandwidth Bandwidth viewpoint (change)					Intel VTune Amplifier XE 2015
🖪 🖾 Collection Log 🖶 Analysis Target 🗛 Analysis Type 🖬 Sum	mary 🔗 Bottom-up				
Q°Q+Q=Q+ 5s 10s 15s 20s	25s 34.555s [D	uration: 10.32slls 4	15s 50s	s 55s	60s 65s 70s Ruler Area
package_0	000000000000000000000000000000000000000	iopaoooooooioaaa	000000000	0000000000000	
package 1 =53.093					▲ Bandwidth, GB/sec
					Read Bandwidth, GB/
					Write Bandwidth, GB/
Peak BW		Zaara la an Calastian		Dee	d write
		Lindo Previous Zoom	Selection	Red	a, write,
		Reset Zoom	Selection	and	aggregate
package_1		Zeen to end Elber to	hu Calastian	BW	time series
		Eilter In by Selection	by Selection		
🚊 🛛 🛛 Click and drag	Filter Out by Selectio				
to zoom in for		Remove All Filters			
more detail		Change Band Height			
more detail		Show Time Scale As	2	>	
	III	Diamias Manu			>) >) >>
Grouping: Function / Call Stack		Distiliss Meriu			¢ Q ×
Function / Call Stack	CPU Time v 🖈	Instructions Retired	CPI Rate	Module	Function
▶main\$omp\$parallel_for@251	233.752s	96,746,145,119	6.763	stream_c.exe	main\$omp\$parallel_for@251
▶main\$omp\$parallel_for@261	232.648s	107,316,160,974	6.070	stream_c.exe	main\$omp\$parallel_for@261
▶main\$omp\$parallel_for@241	165.040s	75,140,112,710	6.147	stream_c.exe	main\$omp\$parallel_for@241
intel_ssse3_rep_memcpy	162.324s	61,828,092,742	7.348	libintlc.so.5	intel_ssse3_rep_memcpy
_kmp_wait_template <kmp_flag_64></kmp_flag_64>	16.935s	43,728,065,592	1.089	libiomp5.so	voidkmp_wait_template <kmp_flag_64>(kmp_info*, kmp_flag_64</kmp_flag_64>
P [vminux]	9.240s	10,352,015,528	2.525	vmlinux	[Vmiinux]
<pre>rkmp_wait_template<kmp_flag_b4> </kmp_flag_b4></pre>	2.9005	7,794,011,691	1.086	itrompo.so	voidkmp_wait_template <kmp_tiag_b4>(kmp_into*, kmp_tiag_b4</kmp_tiag_b4>
	1.60/5	642,000,963	7.879	sueam_c.exe	
Selected 1 row(s):	232.648s	107,316,160,974	6.070		
★ <mark>Filter: 1 is shown</mark> 🌾 Process: [100.0%] stream_c.exe 🔽	Thread: Any Thread	🔽 Module	Any Module		nline Mode: on 🔽 Loop Mode: Functions only







• At NERSC

- On our debugging and profiling tools pages: <u>http://www.nersc.gov/users/software/debugging-and-profiling/vtune/</u>
- More details on how to run your analysis on both the Edison compute nodes and the Babbage Xeon Phis
- Pointers to materials from previous NERSC trainings
- At Intel
 - Main documentation for 2015 version: <u>https://software.intel.com/en-us/node/529213</u>
 - Detailed descriptions of the various experiment types
 - Pointers to tutorials on specific topics or platforms



