

# PROFILING YOUR APPLICATION WITH INTEL<sup>®</sup> VTUNE<sup>™</sup> AMPLIFIER AND INTEL<sup>®</sup> ADVISOR Carlos Rosales-Fernandez

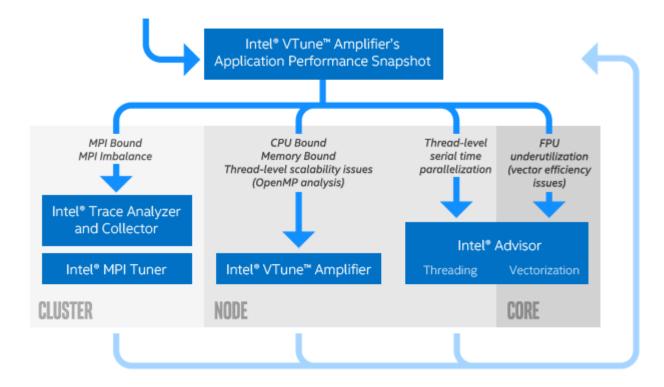
## Tuning at Multiple Hardware Levels

Exploiting all features of modern processors requires good use of the available resources

- Core
  - Vectorization is critical with 512bit FMA vector units (32 DP ops/cycle)
  - Targeting the current ISA is fundamental to fully exploit vectorization
- Socket
  - Using all cores in a processor requires parallelization (MPI, OMP, ... )
  - Up to 64 Physical cores and 256 logical processors per socket on Theta!
- Node
  - Minimize remote memory access (control memory affinity)
  - Minimize resource sharing (tune local memory access, disk IO and network traffic)



## **Tuning Workflow**





## VTune<sup>™</sup> Amplifier's Application Performance Snapshot

High-level overview of application performance

- Identify primary optimization areas
- Recommend next steps in analysis
- Extremely easy to use
- Informative, actionable data in clean HTML report
- Detailed reports available via command line
- Low overhead, high scalability



### Usage on Theta

Launch all profiling jobs from **/projects** rather than **/home** 

No module available, so setup the environment manually:

- \$ source /opt/intel/vtune\_amplifier/apsvars.sh
- \$ export PMI\_NO\_FORK=1

Launch your job in interactive or batch mode:

\$ aprun -N <ppn> -n <totRanks> [affinity opts] aps ./exe

Produce text and html reports:



#### **APS HTML Report**

Application	Performance Sn	apshot	
Application: heart_demo Report creation date: 2017-08-01 12:08:44 Number of ranks: 144 Ranks per node: 18 OpenMP threads per rank: 2 HW Platform: Intel(X seon(R) Processor oc Logical Core Count per node: 72		Your application is MPI bot This may be caused by high busy wait ti optimal communication schema or MPI like Intel® Trace Analyzer and Collector	ne inside the library (imbalance), non- library settings. Use <u>MPI profiling tools</u>
121.39s		Current.run         Target           MPI.Time         53.74% €         <10%           OpenMP.Imbalance         0.43%         <10%           Memory.Stalls         14.70%         <20%           FPU Utilization         0.30% №         >50%	Reita
50.98 0.68		I/O Bound 0.00% <10%	
SP FLOPS CPI	1, <u>MIN</u> 0.65)		
MPL Time 53.74% of Elapsed Time (65.23s)	OpenMP Imbalance 0.43% of Elapsed Time (0.52s)	Memory Stalls 14.70% of pipeline slots	FPU Utilization 0.30%▶
MPLImbalance 11.03% of Elapsed Time	Maria and Frankrick	Cache Stalls 12.84% of cycles	SP FLOPs per Cycle 0.08 Out of 32.00
(13.39s) TOP 5 MPI Functions %	Memory Footprint Resident:	DRAM Stalls 0.18% of cycles	Vector Capacity Usage 25.84%
Waitall 37.	Per node: 15 <u>Peak:</u> 786.96 MB	NUMA	FP Instruction Mix
Isend 6.4	Average: 687.49 MB Per rank:	31.79% of remote accesses	% of <u>Packed FP Instr.</u> : 3.54% % of 128-bit: 3.54%
Barrier 5.5.	Peak: 127.62 MB		% of <u>256-bit</u> : 0.00%
Irecv 3.7			% of <u>Scalar FP Instr.</u> : 96.46%
Scatterv 0.0	Per node:		FP Arith/Mem Rd Instr. Ratio 0.07
L/O Bound 0.00% (AVG 0.00, <u>PEAK</u> 0.00)	<u>Peak:</u> 9173.34 MB Average: 9064.92 MB Per rank: <u>Peak:</u> 566.52 MB Average: 503.61 MB		FP. Arith/Mem. Wr. Instr. Ratio 0.30 N

Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.

(int

# INTEL<sup>®</sup> ADVISOR

Vectorization and Threading

### Intel<sup>®</sup> Advisor

Modern HPC processors explore different level of parallelism:

- between the cores: multi-threading (Theta: 64 cores, 256 threads)
- within a core: vectorization (Theta: 8 DP elements, 16 SP elements)

Adapting applications to take advantage of such high parallelism is quite demanding and requires code modernization

The Intel® Advisor is a software tool for vectorization and thread prototyping

The tool guides the software developer to resolve issues during the vectorization process



## Typical Vectorization Optimization Workflow

There is no need to recompile or relink the application, but the use of -g is recommended.

- 1. Collect survey and tripcounts data
  - Investigate application place within roofline model
  - Determine vectorization efficiency and opportunities for improvement
- 2. Collect memory access pattern data
  - Determine data structure optimization needs
- 3. Collect dependencies
  - Differentiate between real and assumed issues blocking vectorization



## Using Intel<sup>®</sup> Advisor on Theta

Two options to setup collections: GUI (advixe-gui) or command line (advixe-cl).

I will focus on the command line since it is better suited for batch execution, but the GUI provides the same capabilities in a user-friendly interface.

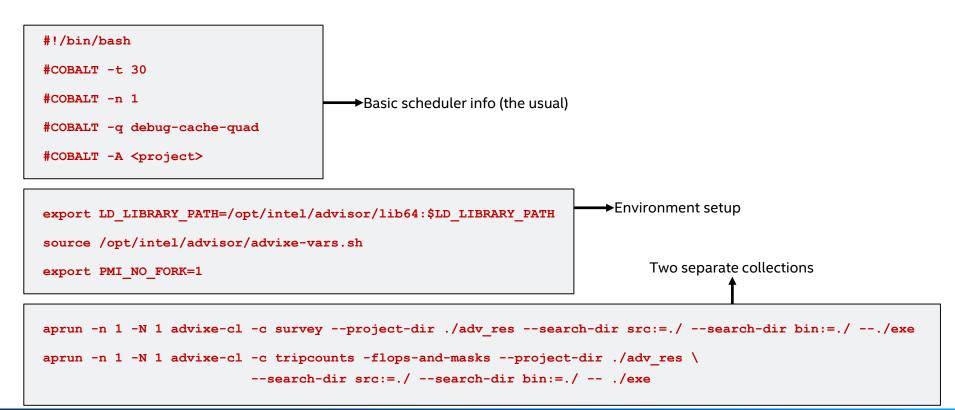
I recommend taking a snapshot of the results and analyzing in a local machine (Linux, Windows, Mac) to avoid issues with lag.

Some things of note:

- Use /projects rather than /home for profiling jobs
- Set your environment:
  - \$ source /opt/intel/advisor/advixe-vars.sh
  - \$ export LD\_LIBRARY\_PATH=/opt/intel/advisor/lib64:\$LD\_LIBRARY\_PATH
  - \$ export PMI\_NO\_FORK=1



### Sample Script



## Cache-Aware Roofline

**FLOPS** 

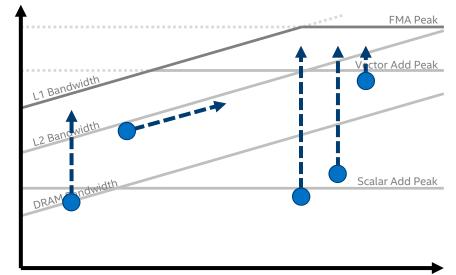
#### Next Steps

## If under or near a memory roof...

- Try a MAP analysis. Make any appropriate cache optimizations.
- If cache optimization is impossible, try reworking the algorithm to have a higher Al.

#### If Under the Vector Add Peak

Check "Traits" in the Survey to see if FMAs are used. If not, try altering your code or compiler flags to **induce FMA usage.** 



#### Arithmetic Intensity

If just above the Scalar Add Peak

Check **vectorization efficiency** in the Survey. Follow the recommendations to improve it if it's low.

#### If under the Scalar Add Peak...

Check the Survey Report to see if the loop vectorized. If not, try to **get it to vectorize** if possible. This may involve running Dependencies to see if it's safe to force it.

#### Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.



# **NBODY DEMONSTRATION**

The naïve code that could

## Nbody gravity simulation

https://github.com/fbaru-dev/nbody-demo (Dr. Fabio Baruffa)

Let's consider a distribution of point masses m\_1,...,m\_n located at r\_1,...,r\_n.

We want to calculate the position of the particles after a certain time interval using the Newton law of gravity.

```
struct Particle
                                              for (i = 0; i < n; i++) {
                                                                              // update acceleration
                                                 for (j = 0; j < n; j++) {
 public:
                                                   real type distance, dx, dy, dz;
                                                   real type distanceSqr = 0.0;
   Particle() { init();}
  void init()
                                                   real type distanceInv = 0.0;
    pos[0] = 0.; pos[1] = 0.; pos[2] = 0.;
                                                   dx = particles[j].pos[0] - particles[i].pos[0];
     vel[0] = 0.; vel[1] = 0.; vel[2] = 0.;
                                                   ...
     acc[0] = 0.; acc[1] = 0.; acc[2] = 0.;
     mass = 0.:
                                                   distanceSqr = dx*dx + dy*dy + dz*dz + softeningSquared;
                                                   distanceInv = 1.0 / sgrt(distanceSgr);
   real type pos[3];
   real type vel[3];
                                                   particles[i].acc[0] += dx * G * particles[j].mass *
   real type acc[3];
                                                                      distanceInv * distanceInv * distanceInv:
   real type mass;
                                                   particles[i].acc[1] += ...
};
                                                   particles[i].acc[2] += ...
```

### **Collect Roofline Data**

Starting with version 2 of the code we collect both survey and tripcounts data:

export LD\_LIBRARY\_PATH=/opt/intel/advisor/lib64:\$LD\_LIBRARY\_PATH

source /opt/intel/advisor/advixe-vars.sh

export PMI\_NO\_FORK=1

aprun -n 1 -N 1 advixe-cl --collect survey --project-dir ./adv\_res --search-dir src:=./ \
--search-dir bin:=./ -- ./nbody.x

aprun -n 1 -N 1 advixe-cl --collect tripcounts -flops-and-masks --project-dir ./adv\_res \
--search-dir src:=./ --search-dir bin:=./ -- ./nbody.x

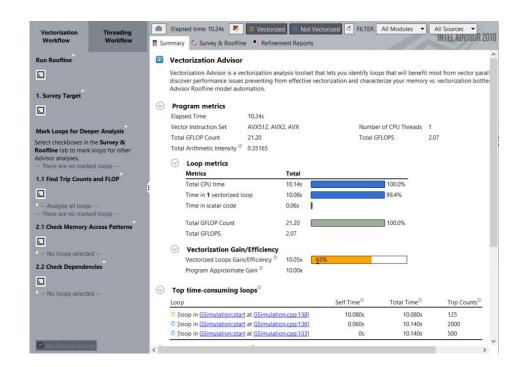
And generate a portable snapshot to analyze anywhere:

advixe-cl --snapshot --project-dir ./adv\_res --pack --cache-sources \

--cache-binaries --search-dir src:=./ --search-dir bin:=./ -- nbody\_naive

If finalization is too slow on compute add -no-auto-finalize to collection line.

### **Summary Report**



GUI left panel provides access to further tests

Summary provides overall performance characteristics

- Lists instruction set(s) used
- Top time consuming loops are listed individually
- Loops are annotated as vectorized and non-vectorized
- Vectorization efficiency is based on used ISA, in this case Intel<sup>®</sup> Advanced Vector Extensions 512 (AVX512)

#### Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.



### Survey Report (Source)

C [loop in GSi G [loop in GSi G [start G [main G [Gop in GSi G [start G [start	imulation::start at GS n::start imulation::start at GS n Code Analytics	GSimulation.cpp:138] Simulation.cpp:136] Simulation.cpp:133]	Performance Issues     2 Inefficient gather/sc     1 Opportunity for outer 1.     1 Data type conversions .      <                  1 Data type conversions .                  • 1 Data type conversions .	0.060s1 0.000s1 0.000s1 0.000s1 0.000s1 0.000s1		Scalar Function Function Function	Why No Vectorization?	Vector. AVX5.	zed Loop Efficien 63%	cy Gain E	VL (Ve	FLOPS Self GFLOP 2.093
Construction of the second sec	Simulation::start at imulation::start at GS n::start imulation::start at GS n Code Analytics	GSimulation.cpp:138] Simulation.cpp:136] Simulation.cpp:133]	© 2 Inefficient gather/sc       © 1 Opportunity for outer I.       © 1 Data type conversions .       > <	10.080s     0.060s1     0.000s1     0.000s1     0.000s1     0.000s1     0.000s1     0.000s1	10.080s 10.140s 10.140s 10.140s 10.140s	Vectorized (Body) Scalar Function Function Function	E inner loop was alread	dy v	_	,		2.093
Construction of the second sec	imulation::start at GS n::start imulation::start at GS n Code Analytics	Simulation.cpp:136] Simulation.cpp:133] Assembly Recom	P 1 Opportunity for outer 1.     P 1 Data type conversions .     C     C     Memory Conversions .     C     Memory Conversions .	0.060s1 0.000s1 0.000s1 0.000s1 0.000s1 0.000s1	10.140s 10.140s 10.140s 10.140s	Scalar Function Function Function		dy v	. 63%	10.05x	16	
C start G f main G f main G f GSimulation G f G Simulation G f G f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f G f f f G f f f G f f f f G f	n::start imulation::start at G n Code Analytics	Simulation.cpp:133]	P 1 Data type conversions .     < <ul> <li>&lt; </li> <li>Mhy No Vector</li> </ul>	0.000sl 0.000sl 0.000sl 0.000sl	10.140s 10.140s 10.140s	Function Function Function						1.700
G f main G f GSimulation G Gloop in GSin C Top Down Ite cache_cc70d097 ine G for fint G for fin	imulation::start at G	Assembly @ Recom	> < mendations 🛱 Why No Vector	0.000s1 0.000s1 0.000s1	10.140s	Function Function	B inner loop was alread	dy v				
<pre>Source Top Down III: cache.ce70d097 III:</pre>	imulation::start at G	Assembly @ Recom	> < mendations 🛱 Why No Vector	0.000s1 0.000s1	10.140s	Function	E inner loop was alread	dy v				
<pre>C [loop in GSin C C C C C C C C C C C C C C C C C C C</pre>	imulation::start at G	Assembly @ Recom	> < mendations 🛱 Why No Vector	orization?			inner loop was alread	dy v				
Cource Top Down Ile: cache ec706097 ine const double 33	n Code Analytics	Assembly @ Recom	> < mendations 🛱 Why No Vector	orization?	10.140s	Scalar	inner loop was alread	dy v				
Top Down           ile: cache_ce70d097           ime           32         const acc           33         If for (int           34         {           35         ts0 +=           36         for (int           37         {           38         for (i           0         for (i           Vector         Vector		Assembly Secom	mmendations 🛱 Why No Vector									
0 [loop in Vector	uble to = time. s=1; s<=get_ns time.start();	.start();	Source					Total Time	% Lo	op/Function Ti	me %	Traits
0 [loop in Vector	(j = 0; j < n;	2						0.400		40.4		
	in GSimulation:	start at GSimulat 512; AVX512F_512	tion.cpp:138] loop processes Float32	; Int32; U	Int32 data	type(s) and incl	udes 2-Source Perm	0.100s		10.0	)80s	
9 {												
-		iz;										
	_type dx, dy, d											
-	_type distances											
3								0.100s				

Inline information regarding loop characteristics

#### ISA used

...

- Types processed
  - Compiler transformations applied
- Vector length used



## Survey Report (Code Analytics)

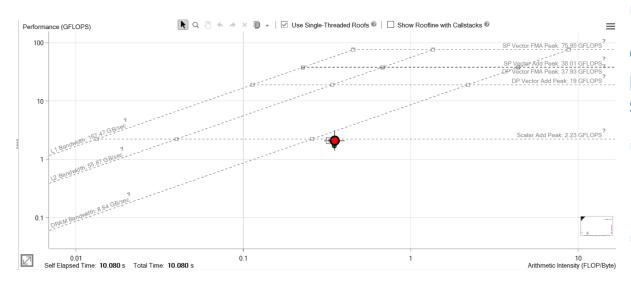
#### Detailed loop information

- Instruction mix
- ISA used, including subgroups
- Loop traits
  - FMA
  - Square root
  - Gathers / Blends point to memory issues and vector inefficiencies

Function Call Sites and Loops	@ Performance	alf Times	Total Time	Turne	Martine Al	o Vectorization?	Vectoriz	ed Loops		$\sum$	FLOF
Function Call Sites and Loops	Issues	en nme 🕈	Total Time	Туре	why is	vo vectorization?	Vector	Efficiency	Gain E	VL (Ve_	. Self
O [loop in GSimulation::start at GSimulation.cpp:138]	© 2 Inefficient gat 1	0.080s 🥅	10.080s 🥅	Vectorized (Body)			AVX5	63%	10.05x	16	2.09
Iloop in GSimulation::start at GSimulation.cpp:136	© 1 Opportunity for	0.060s1	10.140s	Scalar	inne	er loop was already v					1.70
∍ ∮_start		0.000s1		Function							
∋ f main		0.000s1		Function							
GSimulation::start		0.000s1		Function							
Iloop in GSimulation::start at GSimulation.cpp:133]	© 1 Data type conv	0.000s1	10.140s	Scalar	inne	er loop was already v					
Loop in GSimulation:start at GSimulation.cpp:138	Average Trip	o Counts:	125	C		GFLOPS: 2.0932 AVX-512 Mask U		7		$\odot$	
Vectorized (Body) Total time AVX512ER_512; 10.080s AVX512F_512 Self time Instruction Set	Sugges	stion: See R	lecommendatio	S May Decrease Perforence S May Decrease S May Decrease Perforence S M	r I Ir	Static Instruction Memory: 22 Compu- 12 Number of Vecto	ite: 21 A		Other:	0	
Dynamic Instruction Mix Summary®	FMA 2-Source Perm	stion: See R nutes	lecommendatio								



### **CARM** Analysis



Using single threaded roof

Code vectorized, but performance on par with scalar add peak?

- Irregular memory access patterns force gather operations.
- Overhead of setting up vector operations reduces efficiency.

#### Next step is clear: perform a Memory Access Pattern analysis



19

### Memory Access Pattern Analysis (Refinement)

#### 

P	Sumi	mary 👌	Surv	ey & Rooflir	ne ឡ	Refiner	ment Repo	orts 🔹 🛚	IAP Source: GS	imulation.	pp						INTEL ADVISUR 20
Site	e Loc	ation			l	Loop-Ca	arried Dep	endencies	Strides Dis	tribution	Access Pattern	Max. Site Footprint	Site Name	Recomme	endations		
1	[loo	p in start	t at GS	imulation.c	pp:1 N	No infor	mation av	ailable	33% / 3	3% / 33%	Mixed strides	5KB	loop_site_1	© 2 Ineffic	cient gather/	scatter instruct	ions present
M	emor	y Access	Patter	ns Report	Depe	ndencie	es Report	♀ Recom	mendations								
ID		Stri	de	Туре		5	Source		Nested Funct	ion Var	iable references		Max. Site	Footprint	Modules	Site Name	Access Type
= P	1	10;	40	Constant st	tride	0	GSimulatio	on.cpp:144		blo	ck 0x60a0b0 alloc	ated at GSimulation.cpp	:109 4KB		nbody.x	loop_site_1	Read
	142		rea	l_type di	istand	ceInv :	= 0.0£;										
	143																
	144 145								i].pos[0];	//1f							
	145								i].pos[1]; i].pos[2];	//1f							
P		8		Gather strid	and the second		The second second second second	on.cpp:144	and the second se		and the second	ated at GSimulation.cpg	109 5KB		nbody.x	loop site 1	Read
-	142	_		l type di				meppini		010		acca at opiniaran anapp			novajn	icop_site_i	
	143		100	r_oypo u	LUCUIIC		0.02,										
	144								i].pos[0];	//1f							
	145 146								i].pos[1]; i].pos[2];	//1f //1f							
		ī		Parallel site		-				//11	rop				nbodv.x	loop site 1	
-	142	_						n.cpp. 144							nbouy.x	loop_site_1	
	142		rea	l_type di	istand	ceinv :	= 0.0f;										
	144		dx	= particl	les[j]	.pos[	0] - pa:	rticles[	i].pos[0];	//1f	lop						
	145								i].pos[1];	//1f							
	146			-	-				i].pos[2];	//1f	lop				1		1
-	-	0		Uniform str	ride	0	GSimulatio	on.cpp:149					4B		nbody.x	loop_site_1	Read
	147										11000						
	148			tanceSqr tanceInv							//6flops div+1sqrt						
	150		415	cancerny	- 1.0	01 / 31	dres (ar	scanceaq	- / ,	//1	arvingto						
	151		par	ticles[i]	].acc	[0] +=	dx * G	* parti	cles[j].ma:	ss * dis	tanceInv * dis	stanceInv * distar	nceInv; /	/6flops			

Storage of particles is in an Array Of Structures (AOS) style

This leads to regular, but non-unit strides in memory access

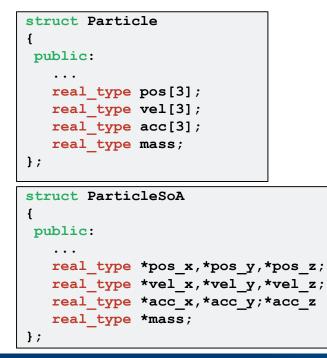
- 33% unit
- 33% uniform, non-unit
- 33% non-uniform

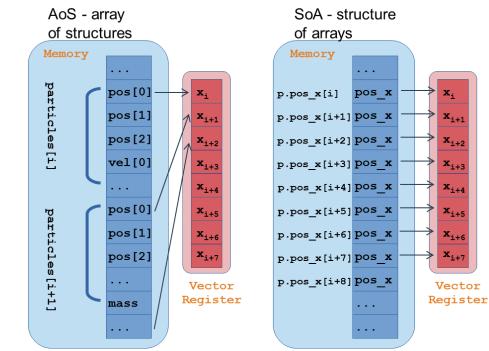
Re-structuring the code into a Structure Of Arrays (SOA) may lead to unit stride access and more effective vectorization



### Vectorization: gather/scatter operation

The compiler might generate gather/scatter instructions for loops automatically vectorized where memory locations are not contiguous





Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved

\*Other names and brands may be claimed as the property of others.

## Performance After Data Structure Change

In this new version (version 3 in github sample ) we introduce the following change:

 Change particle data structures from AOS to SOA

Note changes in report:

- Performance is lower
- Main loop is no longer vectorized
- Assumed vector dependence prevents automatic vectorization

Scalar No loc	nction Call Sites and Loops in GSimulation:start at GSimulation.cpp:151] r loop. Not vectorised: vector dep op transformations applied n GSimulation:start at GSimulation.cpp:171]	Scalar loop.		46.360s	Type Scalar		No Vectorization?	Vector	Gain E	VL (Ve	Self GFLOPS
Scalar No loo Iloop in f_start f main	r loop. Not vectorized: vector dep op transformations applied	Scalar loop.			Scalar	vec	tor dependence pre				
No loc of [loop in y f _start y f main	op transformations applied		Not vector				tor dependence pre				1.122
∍ ƒ_start ∍ ƒ main	n GSimulation::start at GSimulation.cpp:171]	No loop transformations appl			or dependend	ce prevent:	s vectorization				
∍ f main		♀ 1 Assumed depe	0.040s1	0.040s1	Scalar	vec	tor dependence preve				0.475 📖
			0.000s1	46.400s	Function						
🗉 🕇 GSimul			0.000s1	46.400s	Function						
		<	0.000s1	46.400s	Function						
	C 46.360s						AVX-512 Mask	usage:	100		
		Memory: 8		1 Mixed : 1	1 Other:	T	Code Optimizati Compiler: Intel(R) C++ running on Intel(R) 60 Version: 18 0.0 128 B	Intel(R)		er for appl	ications
5	Static Instruction Mix Summary <sup>®</sup> Memory 24% (8)  Compute 32% (11) Mixed <sup>®</sup> 32% (11) Other 12% (4)  Dynamic Instruction Mix Summary <sup>®</sup>	4 Numbe	r of Vector Re	gisters: 21			Version. 10.0.0.120 BC	10 20170			

#### Next step is clear: perform a Dependencies analysis



22

### **Dependencies Analysis (Refinement)**

#### aprun -n 1 -N 1 advixe-cl --collect dependencies --project-dir ./adv\_res \ --search-dir src:=./ --search-dir bin:=./ -- ./nbody.x

🖾 Su	mmary	🏷 Survey & R	oofline 📲	Refinement Rep	oorts 🚯 De	pendencies	Source: GSimul	ation.cp	p					INTELADVI	UDITLU
Site L	ocation			Loop-Carried	Dependenci	es Stride	s Distribution	A	cess Pattern		Max. Site Footprint	Site Name	Recomm	nendations	
🖻 (I	oop in s	tart at GSimulat	on.cpp:157]	RAW:4		No inf	ormation availa	able No	information	available	No information availated	ble loop_site_1	© 1 Prov	en (real) dependency prese	nt
Men	nory Acc	ess Patterns Rep	ort Depen	idencies Report	♀ Recomr	mendations	]								
		d Messages												Filter	
ID	•	Туре		Site Name	Sources		Modules	State						Severity	
P1		Parallel site info	mation	loop_site_1	GSimulation	.cpp	nbody.x	✓ Not	a problem					Error	4 item
P3	•	Read after write	dependency	loop_site_1	GSimulation	.cpp	nbody.x	R Nev	/					Information	1 iter
P4	•	Read after write	dependency	loop_site_1	GSimulation	.cpp; main.c	op nbody.x	Nev	/					Туре	
P5		Read after write			GSimulation		nbody.x	Nev						Parallel site information	1 ite
P6	•	Read after write	dependency	loop_site_1	GSimulation	.cpp	nbody.x	Nev	/					Read after write depend.	
Read	after w	rite dependenc	v: Code Loca	tions										Source	
ID	_	tion Address	Description			Function	Variable refere	ences	Module	State			^	GSimulation.cpp	5 iten
- X3	0x401c		Parallel site		ion.cpp:157	start			nbody.x	New				main.cpp	1 iter
	55	real_type	distance:	Inv = 0.0f;										Module	
	56 57	dy = part	i classino	s x[j] - par	ticles	on while	//1flop							nbody.x	5 iten
1	58 59	dy = part	icles->po	s_y[j] - par s_y[j] - par s_z[j] - par	cticles->p	os_y[i];	//1flop //1flop							State	
		cb8. 0x401d17			ion.cpp:164		register XMM1	1	nbody.x	New			_	New	4 iten
	62			f / sgrtf(di			//ldiv							Not a problem	1 iter
1	63				-			-							
1	64 65										<pre>distanceInv; //6f distanceInv; //6f</pre>				
	66										distanceInv; //6f				
	0x401c	d1e	Write	🗄 GSimulat	ion.cpp:164	start			nbody.x	Rew New					
_															
11	62 63	distance	Inv = 1.0	f / sqrtf(di	istanceSqr	:);	//ldiv	+lsqr	t					Sort By Item Name	3

Dependencies analysis has high overhead:

 Run on reduced workload

#### **Advisor Findings:**

- RAW dependency
- Multiple reduction-type dependencies

23

#### Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

#### Recommendations

Memory Access Patterns Report Dependencies Report & Recommendations

All Advisor-detectable issues: C++ | Fortran

#### Recommendation: Resolve dependency

The Dependencies analysis shows there is a real (proven) dependency in the loop. To fix: Do one of the following:

 If there is an anti-dependency, enable vectorization using the directive #pragma omp simd safelen(length), where length is smaller than the distance between dependent iterations in anti-dependency. For example:

ISSUE: PROVEN (REAL) DEPENDENCY PRESENT

The compiler assumed there is an anti-dependency (Write after read - WAR) or true dependency (Read after write - RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

**Q** Resolve dependency

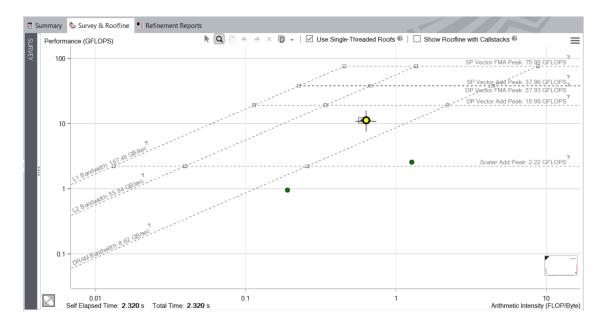
• If there is a reduction pattern dependency in the loop, enable vectorization using the directive #pragma omp simd reduction(operator:list). For example:

```
#pragma omp simd reduction(+:sumx)
for (k = 0;k < size2; k++)
{
    sumx += x[k]*b[k];
}</pre>
```

Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.



### **Performance After Resolved Dependencies**



#### New memory access pattern plus vectorization produces much improved performance!



### What next?

Performance per core may be improved, but it is capped at ~5x current value.

Let's explore threading with a suitability analysis.

- Recompile including annotation definitions
- Add headers to file
- Annotate suggested loops
- Run suitability collection

Vectorization Workflow	Threading Workflow		·	ed time: 2.43s	Vectorized Not Vect		All Modules	ort
			- /			- ·		
2		^			eading design and prototyping prmal development.	tool that lets you anal	yze, design, tune, and	d check threading de
. Annotate Source	es		i No se	ource files foun	d to scan for annotations	5.		
dd Intel Advisor an <u>Ientify</u> possible par			No ap	propriate source file	es were found in your project.			
heir enclosing para			Prog	ram metrics				
Steps to annotate	e		Elapse	ed Time	2.43s			
2.1 Specify the In	tel Advisor		Vecto	r Instruction Set	AVX512, AVX2, AVX	Numbe	er of CPU Threads 1	
include directory			Total	GFLOP Count	26.12	Total G	FLOPS 1	10.77
2.2 For Fortran m the library name			Total	Arithmetic Intensity	<sup>®</sup> 0.63431			
for linking annot	ation definitions.		>	Loop metrics				
2.3 Include the <u>an</u> <u>definitions</u> . 2.4 Insert <u>annota</u>			$\bigcirc$	Vectorization Ga	ain/Efficiency			
at least one para parallel task(s) w	llel site and the	H	😔 Тор	time-consuming	∣ loops <sup>®</sup>			
Use a code edito	or to <u>insert</u>		Consid	der adding parallel	site and task annotations around	I these time-consumin	ig loops found during	g Survey analysis.
annotations. 2.5 Rebuild using	Release build		Loop			Self Time <sup>®</sup>	Total Time <sup><math>\odot</math></sup>	Trip Counts <sup>®</sup>
settings and run			© [loc	op in GSimulation::s	tart at GSimulation.cpp:143]	0s	2.380s	500
3).		~	ٽ [اoo	op in GSimulation::st	tart at GSimulation.cpp:146]	0.040s	2.360s	2000
			o[[oc	op in GSimulation::s	tart at GSimulation.cpp:154]	2.320s	2.320s	125
	_		o[[oc	op in <u>GSimulation::st</u>	tart at GSimulation.cpp:177]	0.020s	0.020s	2000
G Re-finalize Surv	vev							



Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

**Optimization Notice** 

### Annotating the code

Add annotations as shown on the left sample

Complex sites may be analyzed in more detail using task sections if needed

- ANNOTATE\_SITE\_BEGIN / ANNOTATE\_SITE\_END
- ANNOTATE\_TASK\_BEGIN / ANNOTATE\_TASK\_END

Recompile including annotation definitions:

-I/opt/intel/advisor/include

Collect suitability data

```
#include "advisor-annotate.h"
. . .
ANNOTATE SITE BEGIN(steps)
for (int s=1; s<=get nsteps(); ++s)</pre>
{
  . . .
  ANNOTATE TASK BEGIN (particles)
  for (i = 0; i < n; i++)
    . . .
  ANNOTATE TASK END(particles)
ANNOTATE SITE END(steps)
```

Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

## Suitability report

Good speedup expected, but far from ideal (~56% efficiency ).

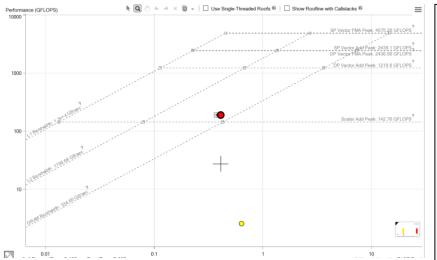
Modeling shows that increasing the task length would improve efficiency.

Next step: add omp parallel region to code and re-rest

Maximum Program	Target Syste	em: CPU	$\sim$	Threading Model: OpenMP	~	CPU Count: 64	4 ~
Gain For All Sites: 36.40x	Site Label	Source Location	Impact to Program (	Combined Site Metr	ics, All Instances		Site Instance Metrics,
	10s			Total Serial Time	Total Parallel Time	Site Gain	Parallel Time
Fredicted Faraller time. 0.0	steps	GSimulation.cpp:144	36.40x	2.192s	0.043s	51.32x	0.043s
Site Performance Scalability Sit	e Details						
Scalability of Maxi	mum Site Gain	Tasks Mo	deling	Runtime Modeling			
		Avg. Numbe	r Avg. Task	Type of Change	Gain Benefit if Enab	led	
64x-	Q	of Tasks:	Duration:	Reduce Site Overhead	+(	0.01x	
32x-	Ŷ	500	0.004s	Reduce Task Overhea	4 +1	1.90x	
M 16x- 8x- 4x- 2x- 0 1x-	Ŷ	0.008x 0.040x	0.008x 0.040x	Reduce Lock Overhea	d		
m 8x-		0.200x	0.200x	Reduce Lock Content	on		
n Site		1x (500) 5x	1x (0.004s) 5x	Enable Task Chunking		1.84x	
ິ <sub>ດ</sub> 2x- ງ		25x	25x				
□ 1x-		125x	125x				
			Apply				
2 4 8 CPU Coun	16 32 64 t						
	t	*					
CPU Coun	t 106s	* *					
CPU Coun 15.1% Load Imbalance: 0.0	t 106s						



### **Roofline for Threaded Version**



Now using regular roofline, instead of single-threaded

Still room for improvement, but at this point we need additional detail regarding shared resource utilization

#### for (int s=1; s<=get nsteps(); ++s)</pre> { ts0 += time.start(); #pragma omp parallel for for (i = 0; i < n; i++)// update acceleration real type ax i = particles->acc x[i]; real type ay i = particles->acc y[i]; real type az i = particles->acc z[i]; #pragma omp simd reduction(+:ax i,ay i,az i) for (j = 0; j < n; j++)real type dx, dy, dz; real type distanceSqr = 0.0f; real type distanceInv = 0.0f; dx = particles->pos x[j] - particles->pos x[i];

Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

# INTEL<sup>®</sup> VTUNE<sup>™</sup> AMPLIFIER

### Intel<sup>®</sup> VTune<sup>™</sup> Amplifier

VTune Amplifier is a full system profiler

- Accurate
- Low overhead
- Comprehensive (microarchitecture, memory, IO, treading, ... )
- Highly customizable interface
- Direct access to source code and assembly

Analyzing code access to shared resources is critical to achieve good performance on multicore and manycore systems

VTune Amplifier takes over where Intel® Advisor left



### **Predefined Collections**

#### Many available analysis types:

•	advanced-hotspots concurrency	Advanced Hotspots Concurrency	
	disk-io	Disk Input and Output	
	general-exploration	General microarchitecture exploration	
	gpu-hotspots	GPU Hotspots	
•	gpu-profiling	GPU In-kernel Profiling	
	hotspots	Basic Hotspots	
	hpc-performance	HPC Performance Characterization	
•	locksandwaits	Locks and Waits —	Python Support
•	memory-access	Memory Access	
•	memory-consumption	Memory Consumption	
	system-overview	System Overview	

...



## The HPC Performance Characterization Analysis

#### Threading: CPU Utilization

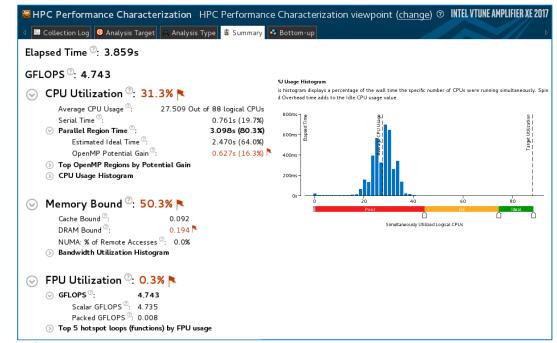
- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

#### Memory Access Efficiency

- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

#### Vectorization: FPU Utilization

- FLOPS<sup>†</sup> estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization



<sup>†</sup> For 3rd, 5th, 6th Generation Intel<sup>®</sup> Core<sup>™</sup> processors and second generation Intel<sup>®</sup> Xeon Phi<sup>™</sup> processor code named Knights Landing.

### Memory Access Analysis

#### Tune data structures for performance

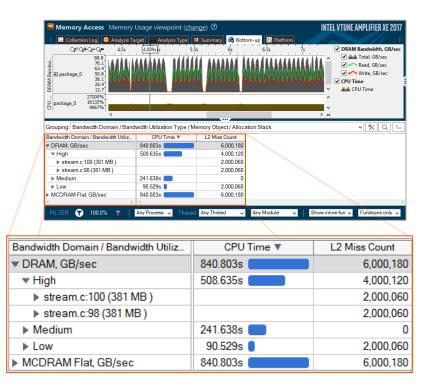
- Attribute cache misses to data structures (not just the code causing the miss)
- Support for custom memory allocators

#### **Optimize NUMA latency & scalability**

- True & false sharing optimization
- Auto detect max system bandwidth
- Easier tuning of inter-socket bandwidth

#### Easier install, Latest processors

- No special drivers required on Linux\*
- Intel<sup>®</sup> Xeon Phi<sup>™</sup> processor MCDRAM (high bandwidth memory) analysis



34

## Using Intel<sup>®</sup> VTune<sup>™</sup> Amplifier on Theta

Two options to setup collections: GUI (amplxe-gui) or command line (amplxe-cl).

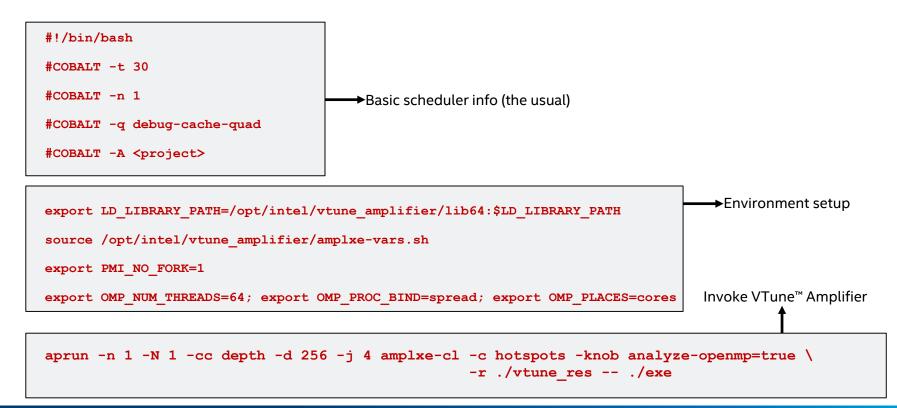
I will focus on the command line since it is better suited for batch execution, but the GUI provides the same capabilities in a user-friendly interface.

Some things of note:

- Use /projects rather than /home for profiling jobs
- Set your environment:
- \$ source /opt/intel/vtune\_amplifier/amplxe-vars.sh
- \$ export LD\_LIBRARY\_PATH=/opt/intel/vtune\_amplifier/lib64:\$LD\_LIBRARY\_PATH
- \$ export PMI\_NO\_FORK=1

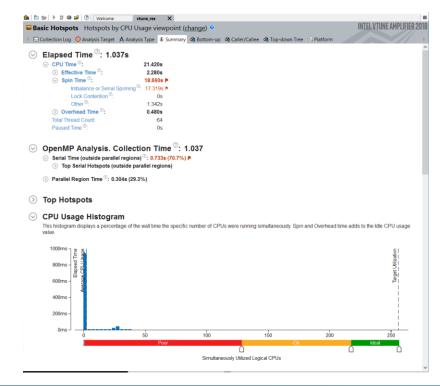


### Sample Script



Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

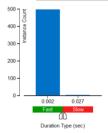
### Hotspots analysis for nbody demo (ver7: threaded)



#### • OpenMP Region Duration Histogram

This histogram shows the total number of region instances in your application executed with a specific duration. High number of slow instances may signal a performance bottleneck. Explore the data provided in the Bottom-up, Top-down Tree, and Timeline panes to identify code regions with the slow duration.

OpenMP Region: start\$omp\$parallel:64@unknown:146:182 ~



#### Lots of spin time indicate issues with load balance and synchronization

Given the short OpenMP region duration it is likely we do not have sufficient work per thread

Let's look a the timeline for each thread to understand things better...

#### **Optimization Notice**

Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.



### **Bottom-up view**

Grouping: Module / Function / Call Stack					~ 🛠 Q 🗞	CPU Time
	CPU Time	<b>v</b>	đ.			Viewing < 1 of 1 > selected stack
Module / Function / Call Stack	Effective Time by Utilization	Spin Time	Overhead Time	Module		100.0% (2.260s of 2.260s) nbody.xIGSimulation::start\$omp
libiomp5.so	0s	18.660s	0.320s			libiomp5.so![OpenMP dispatch
nbody.x	2.260s	0s	0.160s			libiomp5.sol[OpenMP fork]+0x
GSimulation::start\$omp\$parallel_fo	© 2.260s	0s	0s	nbody.x	GSimulation::start\$omp\$p	a nbody.xIGSimulation::start+0x6
GSimulation::start	Os	0s		nbody.x	GSimulation::start(void)	nbody.x!main+0x86 - main.cpp:
[Unknown]	0.020s	0s	0s			nbody.xl_start+0x28 - start.S:1
	<pre>&lt; </pre>					>
OMP Master Thread #0 (TID OMP Worker Thread #80 (TI OMP Worker Thread #56 (TI OMP Worker Thread #50 (TI						Running
OMP Worker Thread #56 (TI						to-Barrier Segment ✓ Thread ✓ ✓ ■ Running ✓ ➡ CPU Time ✓ ➡ Spin and Overhea ▼ CPU Sample
OMP Worker Thread #56 (Tl OMP Worker Thread #50 (Tl OMP Worker Thread #55 (Tl OMP Worker Thread #55 (Tl						to-Barrier Segment  Thread  Running  Charlen CPU Time  Spin and Overhea
OMP Worker Thread #56 (T OMP Worker Thread #50 (T OMP Worker Thread #55 (T OMP Worker Thread #54 (T OMP Worker Thread #49 (T						to-Barrier Segment ✓ Thread ✓ ✓ ■ Running ✓ ➡ CPU Time ✓ ➡ Spin and Overhea ▼ CPU Sample
OMP Worker Thread #56 (T OMP Worker Thread #50 (T OMP Worker Thread #55 (T OMP Worker Thread #54 (T OMP Worker Thread #49 (T OMP Worker Thread #58 (T						to-Barrier Segment ✓ Thread ✓ ✓ ■ Running ✓ ➡ CPU Time ✓ ➡ Spin and Overhea ▼ CPU Sample
OMP Worker Thread #56 (TL., OMP Worker Thread #50 (TL., OMP Worker Thread #55 (TL, OMP Worker Thread #54 (TL, OMP Worker Thread #54 (TL, OMP Worker Thread #56 (TL,						to-Barrier Segment ✓ Thread ✓ ✓ ■ Running ✓ ➡ CPU Time ✓ ➡ Spin and Overhea ▼ ⊂ CPU Sample
OMP Worker Thread #56 (TI OMP Worker Thread #50 (TI OMP Worker Thread #55 (TI OMP Worker Thread #54 (TI OMP Worker Thread #56 (TI OMP Worker Thread #56 (TI OMP Worker Thread #51 (TI						to-Barrier Segment
OMP Worker Thread #56 (TI OMP Worker Thread #56 (TI OMP Worker Thread #56 (TI OMP Worker Thread #49 (TI OMP Worker Thread #49 (TI OMP Worker Thread #58 (TI OMP Worker Thread #51 (TI OMP Worker Thread #52 (TI						to-Barrier Segment
OMP Worker Thread #56 (TI OMP Worker Thread #50 (TI OMP Worker Thread #56 (TI OMP Worker Thread #54 (TI OMP Worker Thread #58 (TI OMP Worker Thread #58 (TI OMP Worker Thread #58 (TI OMP Worker Thread #52 (TI OMP Worker Thread #51 (TI						to-Barrier Segment ✓ Thread ✓ ✓ ■ Running ✓ ➡ CPU Time ✓ ➡ Spin and Overhea ▼ ⊂ CPU Sample

There is not enough work per thread in this particular example.

Double click on line to access source and assembly.

Notice the filtering options at the bottom, which allow customization of this view.

Next steps would include additional analysis to continue the optimization process.





Profiling Python is straightforward in VTune<sup>™</sup> Amplifier, as long as one does the following:

- The "application" should be the full path to the python interpreter used
- The python code should be passed as "arguments" to the "application"

In Theta this would look like this:



### Simple Python Example on Theta

#### aprun -n 1 -N 1 amplxe-cl -c hotspots -r vt\_pytest \ -- /usr/bin/python ./cov.py naive 100 1000

**INTEL VTUNE AMPLIFIER 2018** 

Basic Hotspots Hotspots by CPU Usage viewpoint (change)

🗉 🗔 Collection Log 📀 Analysis Target 🗍 Analysis Type 🔹 Summary 🐼 Bottom-up 🚸 Caller/Callee 🚸 Top-down Tree 🗈 Platform 💪 cov.py

#### Elapsed Time<sup>®</sup>: 209.598s

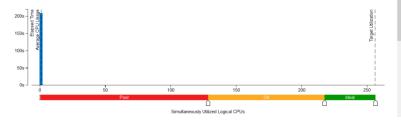
#### Solution → Solutio

This section lists the most active functions in your application. Optimizing these holspot functions typically results in improving overall application performance

Function	Module	CPU Time ®
naive	cov.py	113.533s
<genexpr></genexpr>	cov.py	91.587s
[Outside any known module]		1.460s
[Unknown stack frame(s)]		1.260s
<module></module>	cov.py	0.588s
[Others]		0.532s

#### ⊘ CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU usage value.



Naïve implementation of the calculation of a covariance matrix

Summary shows:

- Single thread execution
- Top function is "naive"

Click on top function to go to Bottomup view

#### Optimization Notice Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.



### **Bottom-up View and Source Code**

Grouping: Module / Function / Call Stack					~ 🛠 Q 4	CPU Time
	CPU Time 🔻		(K)			Viewing + 1 of 1 + selected stack(
Module / Function / Call Stack	Effective Time by Utilization	Spin Time	Overhead Time	Module		100.0% (112.473s of 112.473s) cov.py/naive - cov.py
▼ cov.py	203.728s	2.280s	0s			cov.py/main+0x42 - cov.py 200
▼ naive	111.873s	1.660s	0s	COV.DV	naive(fullArrav)	cov.pyl <module>+0x221 - cov.py</module>
▼ main	110.833s	1.660s	0s	cov.py	main()	python2.71 start+0x28 - [unknow.
▶ S <module> ← _start</module>	110.813s	1.660s	0s	cov.py	<module></module>	
▶ Tain ← <module> ← _star</module>	0.020s	0s	0s	cov.py	main()	
▶ S naive ← main ← <module> ←</module>	1.040s	0s	0s	cov.py	naive(fullArray)	1
sqenexpr>	90.967s	0.620s	0s	cov.py	naive@ <genexpr>i</genexpr>	
<module></module>	0.588s	0s	0s	cov.py	<module></module>	
▶ main	0.300s	0s	0s	cov.py	main()	
[Unknown]	2.720s	0s	0s			
libc-dynamic.so	0.132s	0s	0s			
python2.7	0.060s	0s	0s			
libpin3dwarf.so	0.020s	0s	0s			1
trankrianc on	1 0.020e	0e	0e		>	×
	50s	100s		150s	200s	Thread V
python (TID: 218893)						Running
						Spin and Overhead Ti…     CPU Sample

#### Inefficient array multiplication found quickly We could use numpy to improve on this

📟 Bas	sic Hotspots Hotspots by CPU Usage viewpoint ( <u>change</u> ) 🔮	
4 🖽 🕻	Collection Log	r/Callee 🛭 😪 Top-down Tree 🕒
Sourc	e Assembly 🔲 📃 💿 💿 🤣 🥸 💽 🔍 Assembly grouping: Function Range / Basi	c Block / Address
Sou Line	Source	CPU Time:
		Effective Time by Utili
Line		Idle Poor Ok Idea
59		
60	# calculate norm arrays and populate norm arrays dict	
61	for i in range(numCols):	
62	<pre>normArrays.append(np.zeros((numRows, 1), dtype=float))</pre>	
63	for j in range(numRows):	0.0%
64	<pre>normArrays[i][j]=fullArray[:, i][j]-np.mean(fullArray[:,</pre>	i 6.3%
65		
66		
67	# calculate covariance and populate results array	
68	for i in range(numCols):	
69	for j in range(numCols):	0.0%
70	result[i,j] = sum(p*q for p,q in zip(	
71	<pre>normArrays[i],normArrays[j]))/(numRows)</pre>	47.2%
72		
73	end = time.time()	
74	<pre>print('overall runtime = ' + str(end - start))</pre>	

Note that for mixed Python/C code a Top-Down view can often be helpful to drill down into the C kernels

**Optimization Notice** 

Copyright © 2018, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

### **Useful Options on Theta**

If finalization is slow you can use **-finalization-mode=deferred** and simply finalize on a login node or a differenet machine

If the collection stops because too much data has been collected you can override that with the -data-limit=0 option (unlimited) or to a number (in MB)

Use the **-trace-mpi** option to allow VTune Amplifier to assign execution to the correct task when not using the Intel<sup>®</sup> MPI Library.

Reduce results size by limiting your collection to a single node using an mpmd style execution:

```
aprun -n X1 -N Y amplxe-cl -c hpc-performance -r resdir -- ./exe : \
-n X2 -N Y ./exe
```



### **EMON Collection**

General Exploration analysis may be performed using EMON

- Reduced size of collected data
- Overall program data, no link to actual source (only summary)
- Useful for initial analysis of production and large scale runs
- Currently available as experimental feature

#### export AMPLXE EXPERIMENTAL=emon

aprun [...] amplxe-cl -c general-exploration -knob summary-mode=true[...]



#### Resources

#### **Product Pages**

- https://software.intel.com/sites/products/snapshots/application-snapshot
- https://software.intel.com/en-us/advisor
- https://software.intel.com/en-us/intel-vtune-amplifier-xe

**Detailed Articles** 

- https://software.intel.com/en-us/articles/intel-advisor-on-cray-systems
- https://software.intel.com/en-us/articles/using-intel-advisor-and-vtune-amplifier-with-mpi
- https://software.intel.com/en-us/articles/profiling-python-with-intel-vtune-amplifier-acovariance-demonstration



## Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit <u>www.intel.com/benchmarks</u>.

Copyright © 2018, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

#### **Optimization Notice**

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804



Software