Maximizing parallelizaton of BLAST: Output Formatting Section (OFS)

Albert Golembiowski, Ph.D.

Intel® Corporation

Objectives

- get the last serial section (OFS) in runtime blast(n/p) to parallelize
- Improve throughput performance of the heterogeneous load sharing Xeon® – XeonPhi ® model, for multiple concatenated queries and multiple db volumes (model #4).

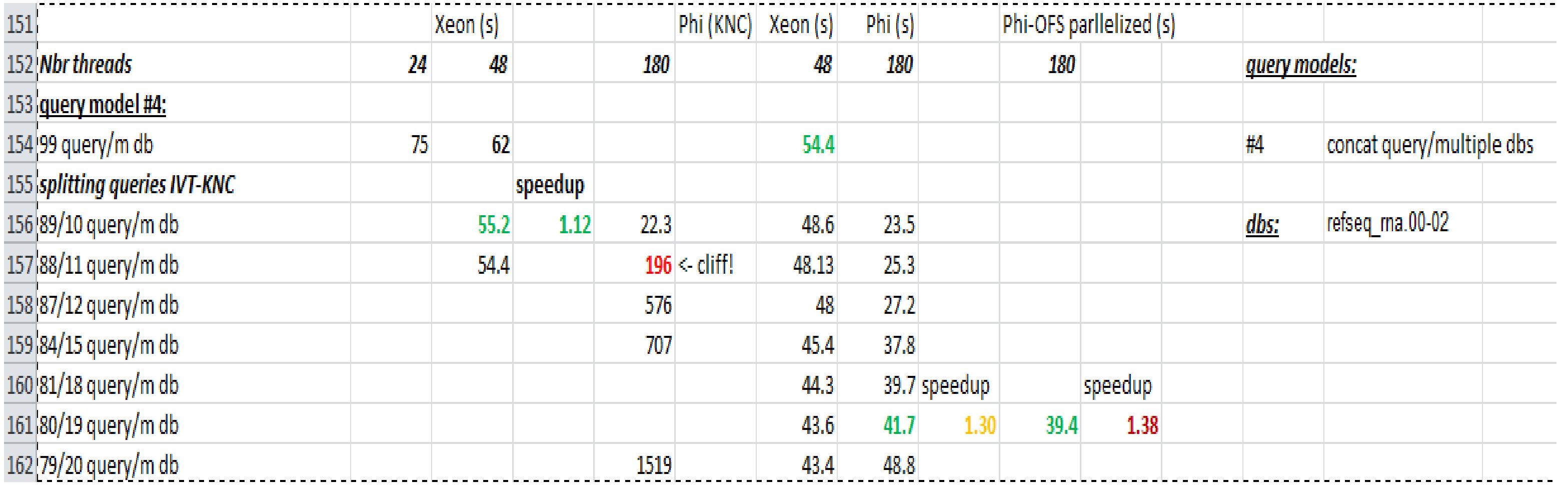
Benchmark configurations

- Intel ® Xeon ® Two 12-Core E5-2697 v2, 2.7GHz (IVT)
- Intel ® Xeon Phi® 7120A (KNC)
- BLAST version 2.2.29, code base retrieved from NCBI website.
- NCBI provided 100 BLASTn benchmarks.
- Multiple query/multiple db command line options are:

```
> blastn_mic –task blastn –db ‘db/refseq_rna.00 db/refseq_rna.01
db/refseq_rna.02’ –query queries/blastn/NM_5_concat –num_threads 180
```

Query model and Xeon® – XeonPhi ® experiment

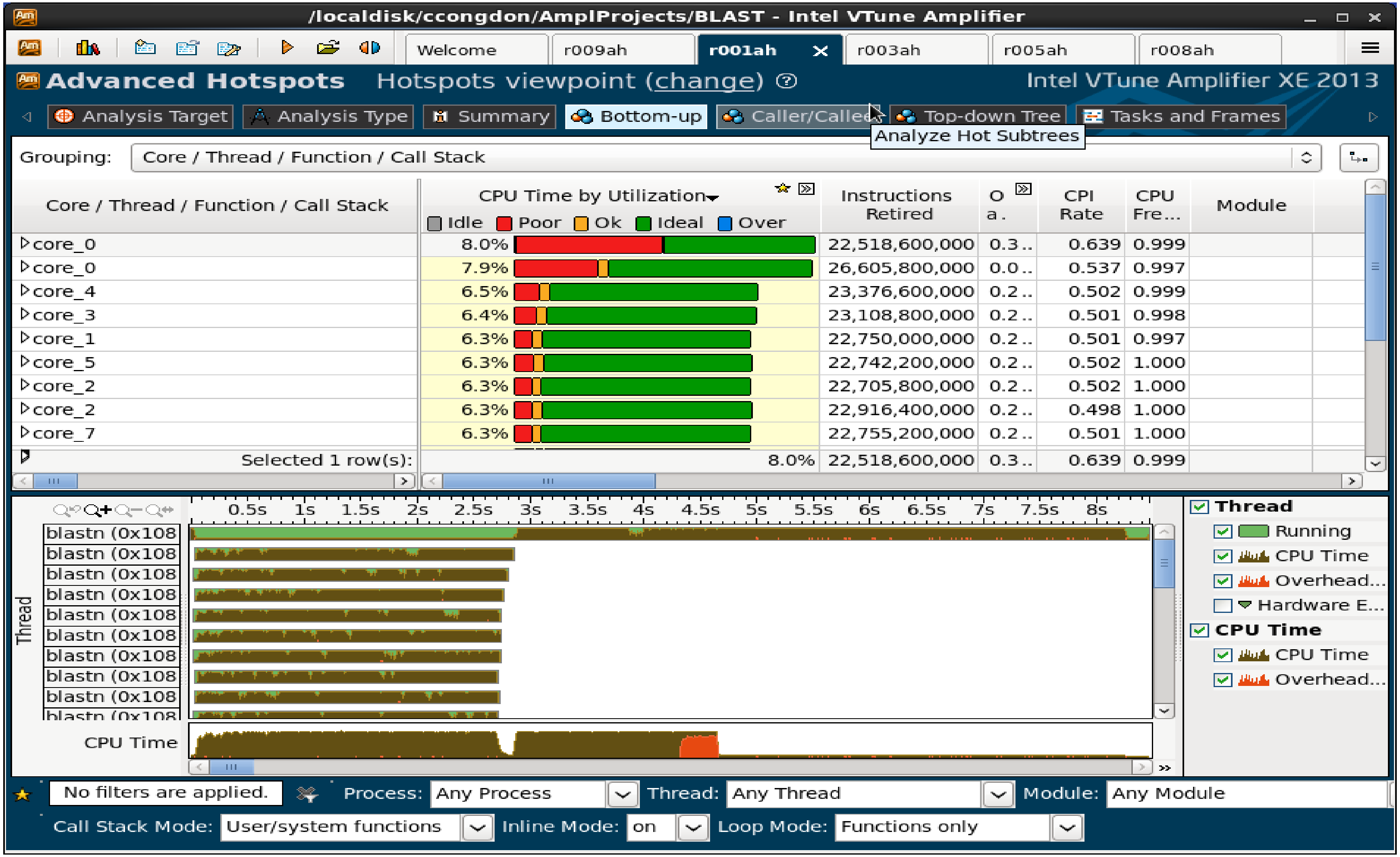
•



151			Xeon (s)			Phi (KNC)	Xeon (s)	Phi (s)		Phi-OFS parllelized (s)				
152	***Nbr threads***	***24***	***48***		***180***		***48***	***180***		***180***			***query models:***	
153	**query model #4:**													
154	99 query/m db	75	62				54.4						#4	concat query/multiple dbs
155	***splitting queries IVT-KNC***			**speedup**										
156	89/10 query/m db		55.2	1.12	22.3		48.6	23.5					***dbs:***	refseq_rna.00-02
157	88/11 query/m db		54.4		196	<- cliff!	48.13	25.3						
158	87/12 query/m db				576		48	27.2						
159	84/15 query/m db				707		45.4	37.8						
160	81/18 query/m db						44.3	39.7	speedup		speedup			
161	80/19 query/m db						43.6	41.7	1.30	39.4	1.38			
162	79/20 query/m db				1519		43.4	48.8						

The 80 concatenated queries on Xeon (IVT) and 19 queries on KNC, in heterogeneous load sharing model (#4) run concurrently in 41.7s and 39.4s respectively, as compared to all 99 concatenated queries on Xeon 54.4s. Here the speedup is 1.3x and ~1.4x for OFS parallelized

BLASTN: single query/single db (model #1)



/localdisk/ccongdon/AmplProjects/BLAST - Intel VTune Amplifier

Welcome | r009ah | r001ah | r003ah | r005ah | r008ah

Advanced Hotspots Hotspots viewpoint (change) Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

Analyze Hot Subtrees

Grouping: Core / Thread / Function / Call Stack

Core / Thread / Function / Call Stack	CPU Time by Utilization ■ Idle 🟥 Poor 🟧 Ok 🟩 Ideal 🟦 Over	Instructions Retired	O a.	CPI Rate	CPU Fre...	Module
▷core_0	8.0%	22,518,600,000	0.3..	0.639	0.999	
▷core_0	7.9%	26,605,800,000	0.0..	0.537	0.997	
▷core_4	6.5%	23,376,600,000	0.2..	0.502	0.999	
▷core_3	6.4%	23,108,800,000	0.2..	0.501	0.998	
▷core_1	6.3%	22,750,000,000	0.2..	0.501	0.997	
▷core_5	6.3%	22,742,200,000	0.2..	0.502	1.000	
▷core_2	6.3%	22,705,800,000	0.2..	0.502	1.000	
▷core_2	6.3%	22,916,400,000	0.2..	0.498	1.000	
▷core_7	6.3%	22,755,200,000	0.2..	0.501	1.000	
Selected 1 row(s):	8.0%	22,518,600,000	0.3..	0.639	0.999	

☑ Thread
☑ Running
☑ CPU Time
☑ Overhead...
□ Hardware E...
☑ CPU Time
☑ CPU Time
☑ Overhead...

No filters are applied. Process: Any Process Thread: Any Thread Module: Any Module

Call Stack Mode: User/system functions Inline Mode: on Loop Mode: Functions only

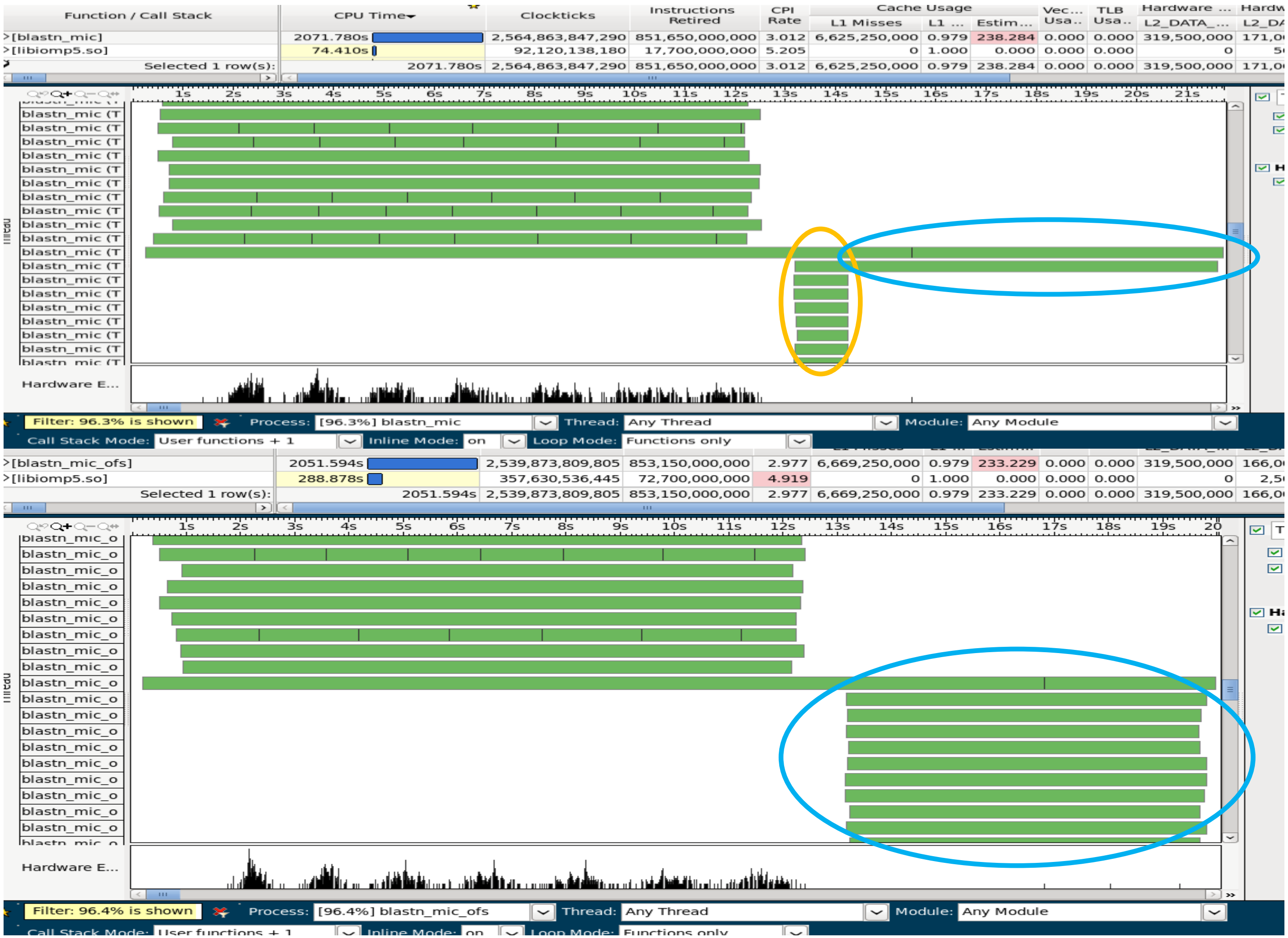
INTEL CONFIDENTIAL

Clearly are visible the 3 sections on "CPU Time". (1) preliminary search (pthread parallelized), (2) traceback search (omp parallelized), and (3) output formatting (serial)

Parallelizing OFS: strategy and implementation

- Identify top level loop and the one with most iterations (key loop)
- Clone/initialize the `display’ (CDisplaySeqalign::) object one clone per thread.
- Break up key loop into NT iteration chunks (equalize workload)
- Identify key loop global object changing state per iteration and make NT clones.
- Spin up clones to a correct initial state.
- Give each iteration chunk to one thread
- Eliminate thread data contentions (mutexes) for maximum speed
- Use LTS std::ostringstream object (STR_STREAM) per thread for output
- Reduce NT STR_STREAM objects into the original NcbiOstrStream output

BLASTN: 5 concat queries on KNC running 180T (model #4)



Function / Call Stack	CPU Time	Clockticks	Instructions Retired	CPI Rate	Cache Usage L1 Misses	L1 ...	Estim...	Vec... Usa..	TLB Usa..	Hardware ... L2_DATA_...	Hardw L2_DA
>[blastn_mic]	2071.780s	2,564,863,847,290	851,650,000,000	3.012	6,625,250,000	0.979	238.284	0.000	0.000	319,500,000	171,0
>[libiomp5.so]	74.410s	92,120,138,180	17,700,000,000	5.205	0	1.000	0.000	0.000	0.000	0	5
Selected 1 row(s):	2071.780s	2,564,863,847,290	851,650,000,000	3.012	6,625,250,000	0.979	238.284	0.000	0.000	319,500,000	171,0

Filter: 96.3% is shown Process: [96.3%] blastn_mic Thread: Any Thread Module: Any Module

Call Stack Mode: User functions + 1 Inline Mode: on Loop Mode: Functions only

>[blastn_mic_ofs]	2051.594s	2,539,873,809,805	853,150,000,000	2.977	6,669,250,000	0.979	233.229	0.000	0.000	319,500,000	166,0
>[libiomp5.so]	288.878s	357,630,536,445	72,700,000,000	4.919	0	1.000	0.000	0.000	0.000	0	2,5
Selected 1 row(s):	2051.594s	2,539,873,809,805	853,150,000,000	2.977	6,669,250,000	0.979	233.229	0.000	0.000	319,500,000	166,0

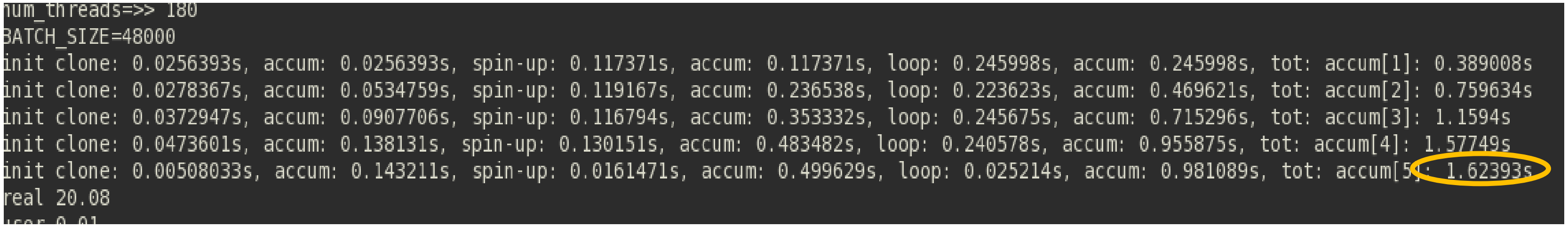
Filter: 96.4% is shown Process: [96.4%] blastn_mic_ofs Thread: Any Thread Module: Any Module

Call Stack Mode: User functions + 1 Inline Mode: on Loop Mode: Functions only

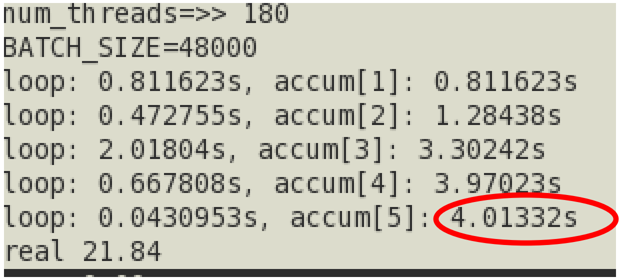
1s 2s 3s 4s 5s 6s 7s 8s 9s 10s 11s 12s 13s 14s 15s 16s 17s 18s 19s 20s 21s
blastn_mic (T
Hardware E...
1s 2s 3s 4s 5s 6s 7s 8s 9s 10s 11s 12s 13s 14s 15s 16s 17s 18s 19s 20
blastn_mic_o
Hardware E...

(1) GAT parallelized, (2) output formatting section (OFS) parallelized. i/o inhibited by writing output into /dev/null.

BLASTN: OFS parallelization re-arch



```
num_threads=>> 180
BATCH_SIZE=48000
init clone: 0.0256393s, accum: 0.0256393s, spin-up: 0.117371s, accum: 0.117371s, loop: 0.245998s, accum: 0.245998s, tot: accum[1]: 0.389008s
init clone: 0.0278367s, accum: 0.0534759s, spin-up: 0.119167s, accum: 0.236538s, loop: 0.223623s, accum: 0.469621s, tot: accum[2]: 0.759634s
init clone: 0.0372947s, accum: 0.0907706s, spin-up: 0.116794s, accum: 0.353332s, loop: 0.245675s, accum: 0.715296s, tot: accum[3]: 1.1594s
init clone: 0.0473601s, accum: 0.138131s, spin-up: 0.130151s, accum: 0.483482s, loop: 0.240578s, accum: 0.955875s, tot: accum[4]: 1.57749s
init clone: 0.00508033s, accum: 0.143211s, spin-up: 0.0161471s, accum: 0.499629s, loop: 0.025214s, accum: 0.981089s, tot: accum[5]: 1.62393s
real 20.08
user 0.01
```

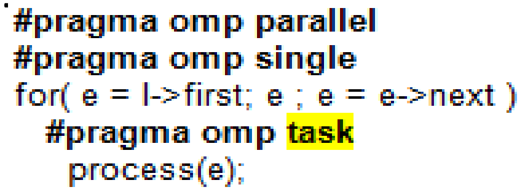


```
num_threads=>> 180
BATCH_SIZE=48000
loop: 0.811623s, accum[1]: 0.811623s
loop: 0.472755s, accum[2]: 1.28438s
loop: 2.01804s, accum[3]: 3.30242s
loop: 0.667808s, accum[4]: 3.97023s
loop: 0.0430953s, accum[5]: 4.01332s
real 21.84
```

(a) parallelized OFS: breakdown of time spent in key loop vs overhead introduced , (b) serial OFS: original loop timing

Things to try

- Resolve icc issue building blastp (workaround identified)
- Reduce OFS parallelization overhead
  - Current implementation of cloning maybe too heavy?
  - try omp



```
#pragma omp parallel
#pragma omp single
for( e = l->first; e ; e = e->next )
   #pragma omp task
     process(e);
```

- Parallelize GAT for blastp.

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.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

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