A study on compiler flags and performance events

Mirela Botezatu

Supervisor: Andrzej Nowak - CERN

February 26, 2013

A study on compiler flags and performance events
Mirela Botezatu

Objective

Prerequisites
- Compiler flags
- Performance events
- Benchmarks

Data retrieval
- Collection environment
- Parallel Setup
- Results

Data analysis
- Statistics on compiler flags
- Coefficient of variation
- Correlations of performance events
- PCA and varimax rotation

Conclusions

Motivation

The quality of the compiler directly impacts the quality of the binary.

- Compilers are the bridge between software and hardware, and the role they play in satisfying real-time and performance constraints is crucial.

We can help the compiler make the appropriate optimizations through the compiler flags.

- Compilers have a pletoria of new capabilities, optimization techniques based on a vast amount of academic research. Fully exploiting them is not an easy task.

We can use performance events to understand the performance issues in the code.

- We can use the PMU to identify performance issues and monitor performance response.

A study on compiler flags and performance events

Mirela Botezatu

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Goals

- ✓ Achieve a deeper understanding of the potential of new compilers and to see to what extent we can predict their performance.
- ✓ Exploratory analysis on performance events for an insightful benchmark characterization.
- ✓ Can we predict compiler optimizations from performance events counts?

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Numbers and figures triggering our motivation

A comparison of compiled *binary speed* of ICC and GCC on a syntthetic set of benchmarks from Adobe.

- ✓ **ICC 13.1.0**
- ✓ **GCC 4.7.2**

Benchmark	**GCC -O2**	**ICC -O2**	**ICC gain**
Functionobjects.cpp	157.2	154.5	**1.01x**
Loop_unroll.cpp	634.4	115.2	**5.51x**
Simple_types_constant_folding.cpp	81.7	103.1	**0.79x**
Simple_types_loop_invariant.cpp	281.6	197.1	**1.42x**
Stepanov_abstraction.cpp	184.1	140.3	**1.31x**
Stepanov_vector.cpp	200.5	143.8	**1.39x**

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Numbers and figures triggering our motivation

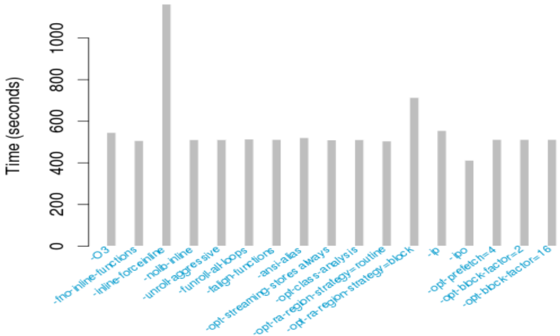
A comparison of compiled *binary speed* of ICC and GCC on ROOT.

- ✓ **ICC 13.1.0**
- ✓ **GCC 4.7.2**
- ✓ **ROOT 5.34**

Benchmark	**GCC -O2**	**ICC -O2**	**ICC gain**
bench	38.66	39.4	**0.97x**
stress	16.28	15.01	**1.08x**
stressShapes	1.04	0.89	**1.16x**
stressLinear	6.94	6.21	**1.11x**
stressSpectrum	3.66	4.38	**0.83x**
stressFit	2.46	2.17	**1.13x**

Numbers and figures triggering our motivation

Influence of compiler flags over the *compile time* - HEPSPEC06



Some differences between ICC and GCC

Some exampes of differences between the O2 and O3 optimization levels for the two compilers ICC and GCC

- ✓ ICC enables inlining at O2 whereas GCC enables it at O3.
- ✓ ICC at O2 optimization level has inlining and other interprocedural optimizations within a source file, vectorization. Vectorization and most inlining is enabled in GCC only at the O3 optimization level.
- ✓ GCC enables ”-fstrict-aliasing” (enforces strict aliasing rules) starting from O2 whereas ICC doesn’t enable it not even at O3.
- ✓ Loop unrolling is enabled starting from O2 with ICC whereas in GCC at O2 there is the flag ”frerun-loop-opt”, which also enables some loop optimizations, but no loop unrolling.
- ✓ ICC has optimized math library functions by default.

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Prerequisites

Software

- ✓ Compiler flags
- ✓ Performance events
- ✓ Benchmarks

Physical

- ✓ The data collection environment (a mini-cluster)

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Compiler flags:

Notes:

- ✓ We did not include flags that disregard strict standards compliance
- ✓ We did not include flags that are enabled by default
- ✓ We did not include ”tune for this architecture” switches

A study on compiler flags and performance events

Mirela Botezatu

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Compiler flags:

- ✓ -O3
- ✓ -fno-inline-functions
- ✓ -inline-forceinline
- ✓ -nolib-inline
- ✓ -unroll-aggressive
- ✓ -funroll-all-loops
- ✓ -falign-functions
- ✓ -ansi-alias
- ✓ -opt-streaming-stores always
- ✓ -opt-class-analysis
- ✓ -opt-ra-region-strategy=routine
- ✓ -opt-ra-region-strategy=block
- ✓ -ip
- ✓ -ipo
- ✓ -opt-prefetch=4
- ✓ -opt-block-factor=2
- ✓ -opt-block-factor=16

A study on compiler flags and performance events
Mirela Botezatu
Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Performance events

Classification:

- General processor characterization
  - General metrics
  - Microarchitectural efficiency and resource utilization
- On-core memory access
- Off core memory access

Performance events:

- ✓ UNHALTED_CORE_CYCLES
- ✓ INSTRUCTION_RETIRED
- ✓ UOPS_RETIRED:ANY
- ✓ UOPS_ISSUED:ANY
- ✓ ARITH:CYCLES_DIV_BUSY
- ✓ ARITH:DIV
- ✓ RESOURCE_STALLS:ANY
- ✓ BR_INST_EXEC:ANY
- ✓ BR_MISP_RETIRED:ALL_BRANCHES
- ✓ BACLEAR:CLEAR
- ✓ L2_RQSTS_:IFETCH:HIT
- ✓ L2_RQSTS_IFETCH:MISS
- ✓ ITLB_MISSES
- ✓ DTLB_LOAD_MISSES
- ✓ MEM_LOAD_RETIRED:L1D_HIT
- ✓ MEM_LOAD_RETIRED:L2_HIT
- ✓ MEM_LOAD_RETIRED:LLC_UNSHARED_ HIT
- ✓ MEM_LOAD_RETIRED:OTHER_CORE_L2_ HIT_HITM
- ✓ MEM_UNCORE_RETIRED:LOCAL_HITM
- ✓ MEM_UNCORE_RETIRED:LOCAL_DRAM_ AND_REMOTE_CACHE_HIT

Benchmarks:

- ✓ High Energy Physics (HEP) benchmarks - a set of benchmarks developed in openlab, which consists of representative snippets for evaluating the code used at CERN.
- ✓ Root benchmarks - benchmarks for stressing the functionality of ROOT
- ✓ Gooda I/O intensive benchmarks
- ✓ Adobe C++ Benchmarks - a set of C++ benchmarks typically used to quantify how well top compiler vendors implement various C++ operations and language features.
- ✓ FFT - a Fast Fourier Transform implementation

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Collection environment

25 x Westmere EP machines Intel(R) Xeon(R) CPU X5650

- ✓ 24 cores (each)
- ✓ Frequency: 2713 MHz
- ✓ Nr sockets: 2
- ✓ Hyper-Threading on
- ✓ Cache size: 12288KB, RAM size: 47 GB
- ✓ Linux Kernel 3.6
- ✓ Perf Tool 3.6

A study on compiler flags and performance events

Mirela Botezatu

Objective

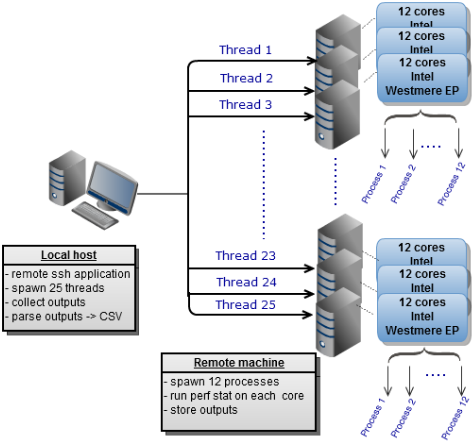
Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Parallel Setup



Results

A serial environment would have cost us 98 days (37 benchmarks x 786 configurations x 5 minutes)

- ✓ 2h20’ for running all the CPU intensive benchmarks
- ✓ 6 hours for running all the I/O intensive benchmarks
- ✓ Performance data for 29082 test runs.

A study on compiler flags and performance events

Mirela Botezatu

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Data analysis

1. Statistics on compiler flags
2. Coefficient of variation of performance events
3. Correlations of performance events
4. Principal Component Analysis

A study on compiler flags and performance events

Mirela Botezatu

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Flags bringing a performance gain, ordered by frequency

Compiler flag	Counts	Compiler flag	Counts
O3	963	Opt-streaming-stores-always	694
Ipo	951	Ansi-alias	686
Opt-ra-region-strategy=routine	821	Opt-prefetch=4	674
Ip	761	Faling-functions	657
Opt-ra-region-strategy=block	760	Unroll-aggressive	652
Funroll-all-loops	753	fno-inline-functions	628
Nolib-inline	740	ipo	694
Inline-forceinline	738	Opt-block-factor=16	616
Opt-class-analysis	700	Opt-block-factor=2	608

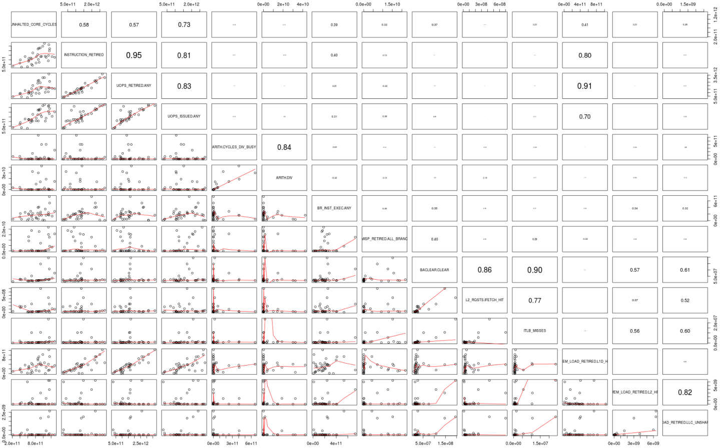
Flags causing performance degradation ordered by frequency

Compiler flag	Counts	Compiler flag	Counts
Opt-streaming-stores-always	1071	Ansi-alias	686
Nolib-inline	1004	Opt-prefetch=4	675
O3	838	Funroll-all-loops	673
Ipo	822	Inline-forceinline	665
Opt-ra-region-strategy=block	818	Unroll-aggressive	656
fno-inline-functions	773	Opt-class-analysis	647
Opt-ra-region-strategy=routine	757	Opt-block-factor=16	590
Ip	710	Opt-block-factor=2	586

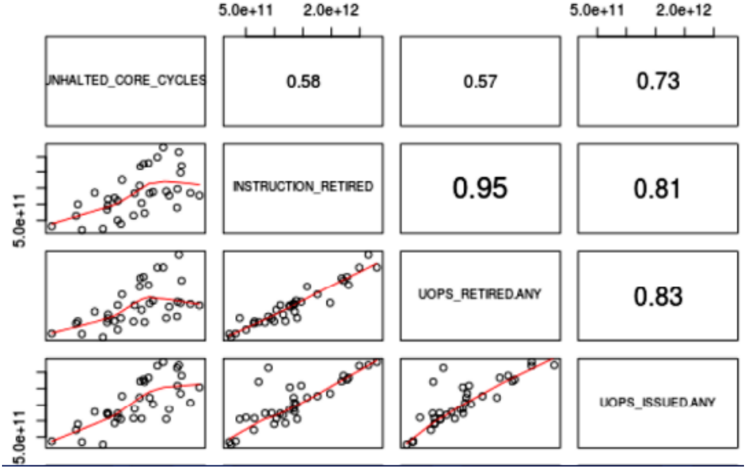
Coefficient of variation of performance events

Events	Coefficient of variation
UNHALTED_CORE_CYCLES	0.021
INSTRUCTION_RETIRED	0.004
UOPS_RETIRED.ANY	0.004
UOPS_ISSUED.ANY	0.005
ARITH.CYCLES_DIV_BUSY	0.12
ARITH.DIV	0.08
RESOURCE_STALLS.ANY	0.035
BR_INST_EXEC.ANY	0.006
BR_MISP_RETIRED.ALL_BRANCHES	0.01
BACLEAR.CLEAR	0.09
L2_RQSTS.IFETCH_HIT	0.081
L2_RQSTS.IFETCH_MISS	0.089
ITLB_MISSES	009
DTLB_LOAD_MISSES	0.094
MEM_LOAD_RETIRED.L1D_HIT	0.006
MEM_LOAD_RETIRED.L2_HIT	0.02
MEM_LOAD_RETIRED.LLC_UNSHARED_HIT	0.04
MEM_LOAD_RETIRED.OTHER_CORE_L2_ HIT_HITM	0.09
MEM_UNCORE_RETIRED.LOCAL_HITM	0.012
MEM_UNCORE_RETIRED.LOCAL_DRAM_ AND_REMOTE_CACHE_HIT	0.059
MEM_UNCORE_RETIRED.REMOTE_DRAM	0.29
MEM_UNCORE_RETIRED.REMOTE_HITM	0.78
MEM_UNCORE_RETIRED_OTHER_LLC.MISS	0.09

Correlations of performance events



Correlations of performance events



A study on compiler flags and performance events

Mirela Botezatu

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Principal component analysis

Principal Component Analysis is a technique for dimensionality reduction, data visualization and compression, latent concept discovery, and preprocessing data in general.

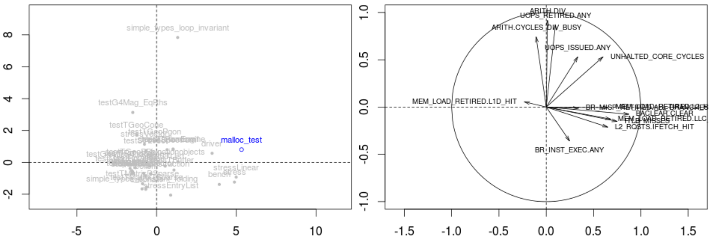
PCA is used to answer the following questions:

1. What are the main performance bottlenecks?
2. Can we identify the characteristics of a new benchmark?

What are the main performance bottlenecks?

Events	Factor1	Factor2	Factor3	Factor4
UNHALTED_CORE_CYCLES	0.646	0.260	-0.561	
INSTRUCTION_RETIRED	-0.686	0.601	-0.141	0.276
UOPS_RETIRED.ANY	-0.736	0.465	-0.238	0.343
UOPS_ISSUED.ANY	-0.670	0.583	-0.189	0.320
ARITH.CYCLES_DIV_BUSY	0.142		-0.903	
ARITH.DIV	0.146		-0.904	
RESOURCE_STALLS.ANY	0.476	-0.217	-0.215	-0.766
BR_INST_EXEC.ANY	-0.410	0.826		0.234
BR_MISP_RETIRED.ALL_BRANCHES	0.163	0.895	0.169	0.116
BACLEAR.CLEAR	-0.191	0.478		
L2_RQSTS.IFETCH_HIT	-0.694	0.214	0.332	0.322
L2_RQSTS.IFETCH_MISS	0.304	0.760	0.141	
ITLB_MISSES	-0.846		0.297	0.201
DTLB_LOAD_MISSES	0.311		0.386	-0.579
MEM_LOAD_RETIRED.L1D_HIT	-0.890		-0.155	0.227
MEM_LOAD_RETIRED.L2_HIT	0.421	0.178	0.394	-0.158
MEM_LOAD_RETIRED.LLC_UNSHARED_HIT	0.287		0.220	-0.898
MEM_LOAD_RETIRED.OTHER_CORE_L2_ HIT_HITM	-0.879	-0.168	0.205	0.171
MEM_UNCORE_RETIRED.LOCAL_HITM	-0.870	-0.170	0.206	0.165
MEM_UNCORE_RETIRED.LOCAL_DRAM_ AND_REMOTE_CACHE_HIT	0.479	-0.215	0.232	0.220
MEM_UNCORE_RETIRED.REMOTE_DRAM		-0.151	-0.846	

Can we identify the characteristics of a new benchmark?

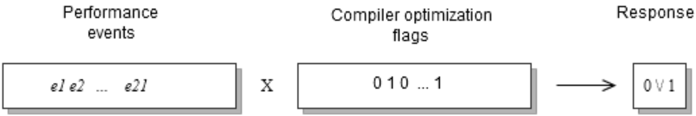


A study on compiler flags and performance events

Mirela Botezatu

Objective
Prerequisites
Compiler flags
Performance events
Benchmarks
Data retrieval
Collection environment
Parallel Setup
Results
Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation
Conclusions

Can we predict compiler flags given performance events counts?

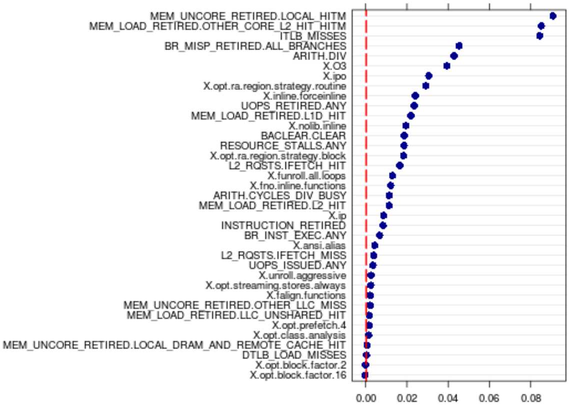


Random forests

Accuracy	Precision	mtry	ntree	mincriterion
80	82	5	1000	0

- ✓ tuned parameters (mtry, ntree)
- ✓ quality of the model asserted through OOB (out of bag error estimate)
- ✓ reported values were averaged on different seeds

Variable importance



Conclusions

- ✓ The selection of the beneficial compiler flags is subject to a judicious choice.
- ✓ Some optimization flags makes the compiler attempt to improve the binary speed and/or code size at the cost of compilation time.
- ✓ The projection matrix obtained after performing PCA can be used to identify the performance issues of a new benchmark.
- ✓ We have built a model that is able to associate performance bottlencks with the compiler optimization flags that are likely to attenuate them.

Objective

Prerequisites
Compiler flags
Performance events
Benchmarks

Data retrieval
Collection environment
Parallel Setup
Results

Data analysis
Statistics on compiler flags
Coefficient of variation
Correlations of performance events
PCA and varimax rotation

Conclusions

Thank you!

A study on compiler flags and performance events

Mirela Botezatu

Objective

Prerequisites
- Compiler flags
- Performance events
- Benchmarks

Data retrieval
- Collection environment
- Parallel Setup
- Results

Data analysis
- Statistics on compiler flags
- Coefficient of variation
- Correlations of performance events
- PCA and varimax rotation

Conclusions