always_inline performance

Benchmark using Intel Compiler Version 15.0.2.164 Build 20150121

Calebe de Paula Bianchini
IPCC/UNESP

The Problem

- Some issues were detected using Intel Compiler with *always_inline*
  - 10x slower to compile VecGeom
    - Using *-j 1*: ± 80 minutes
    - Using *-j 12*: ± 18 minutes
  - 2x bigger lib file libvecgeom.a
    - ± 70 MB
- What happen in GCC ?
  - gcc version 4.8.4

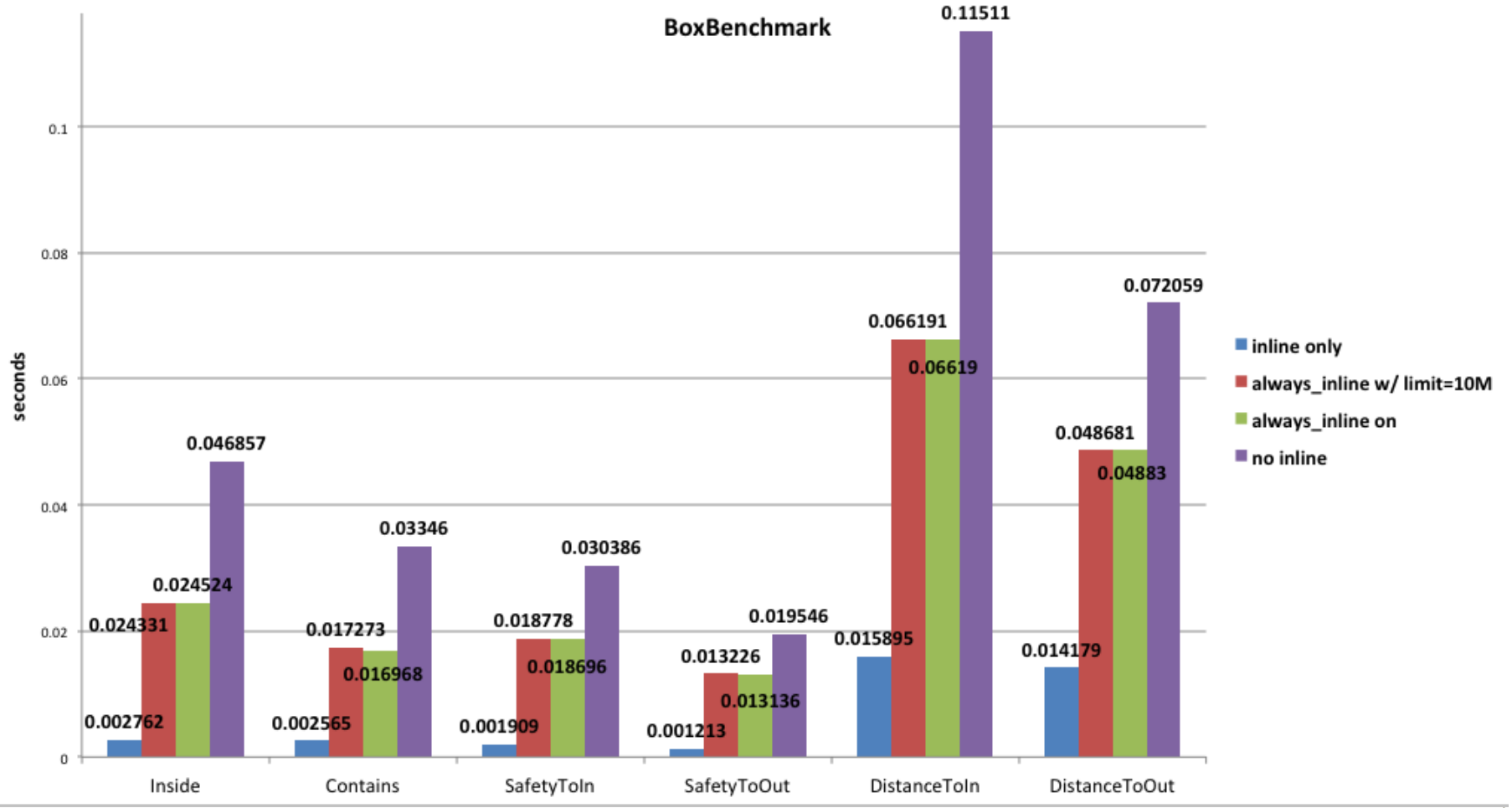
The Benchmark

- Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
  - 6 x 32 KB L1 cache (data/instruction)
  - 6 x 256 KB L2 cache
  - 15 MB L3 shared cache
  - 32 GB of RAM
  - Vc lib & AVX enabled
- Script *shape_benchmark.sh*
  - NPOINTS = 1024
  - NREP = 1024
  - JOBS = 10
- CMS Shapes: Box, Tube, Trapezoid, Cone, Polycone, Polyhedron

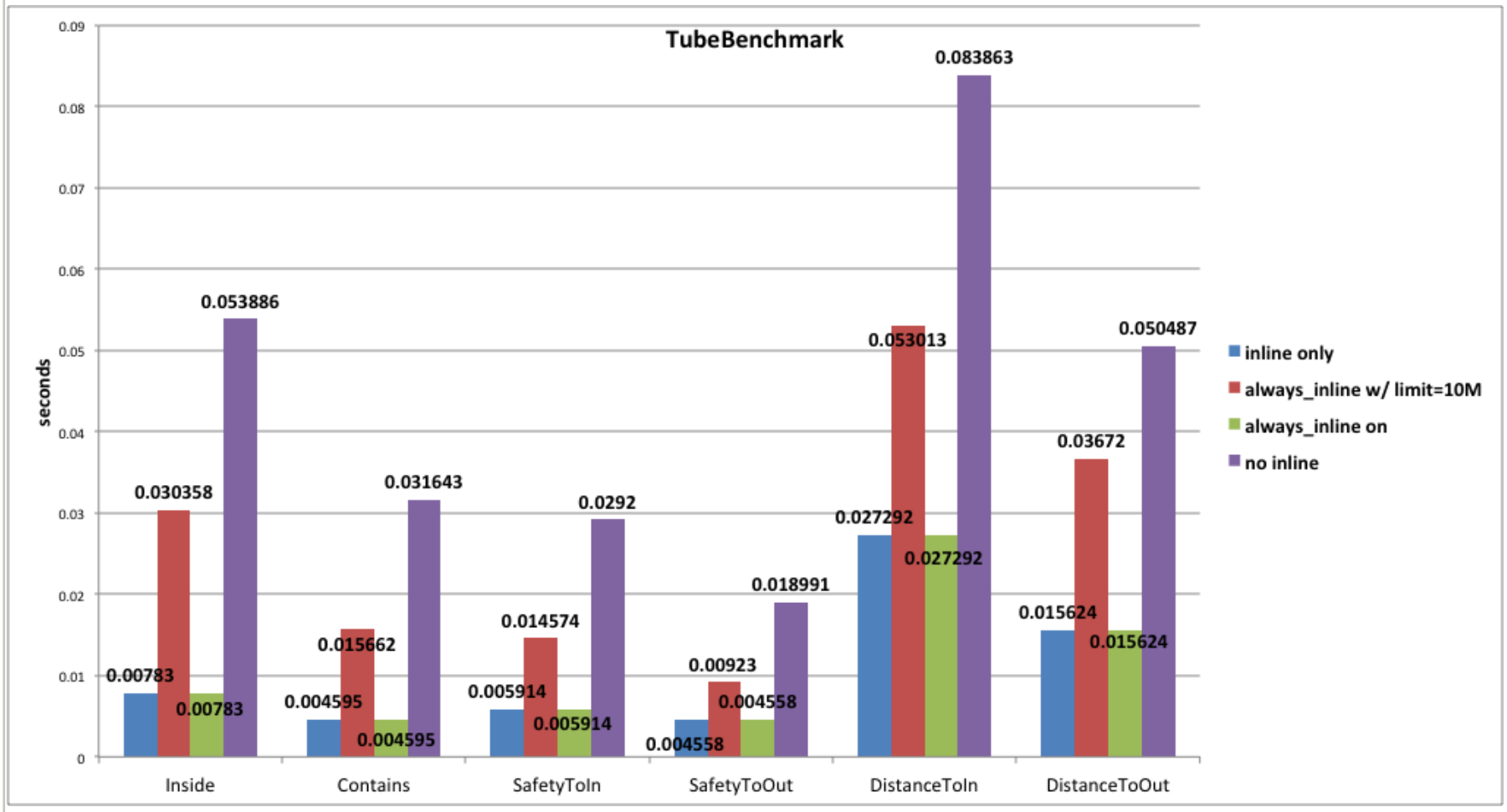
Intel Compiler Benchmark

- inline only
  - *VECGEOM_INLINE = inline*
  - Regular compilation for icc (*-O3 -Wall -fPIC -diag-disable 3438 -fno-alias -xAVX*)
- always_inline w/ limite=10M
  - *VECGEOM_INLINE = inline __attribute__((always_inline))*
  - Regular compilation + *-finline-limit=10000000*
- always_inline
  - *VECGEOM_INLINE = inline __attribute__((always_inline))*
  - Regular compilation
- no line
  - *VECGEOM_INLINE = inline*
  - Regular compilation + *-fno-inline -inline-level=0 –Winline*

Intel Compiler Results



Intel Compiler Results



Intel Compiler Results

- Compiler time (using *-j 12*)
  - inline only: ± 13,0 minutes
  - always_inline w/ 10M: ± 17 minutes
  - always_inline: ± 17 minutes
  - no inline: ± 2,0 minutes
- Lib size
  - inline only: 32 MB
  - always_inline w/ 10M: 66 MB
  - always_inline: 66 MB
  - no inline: 53 MB

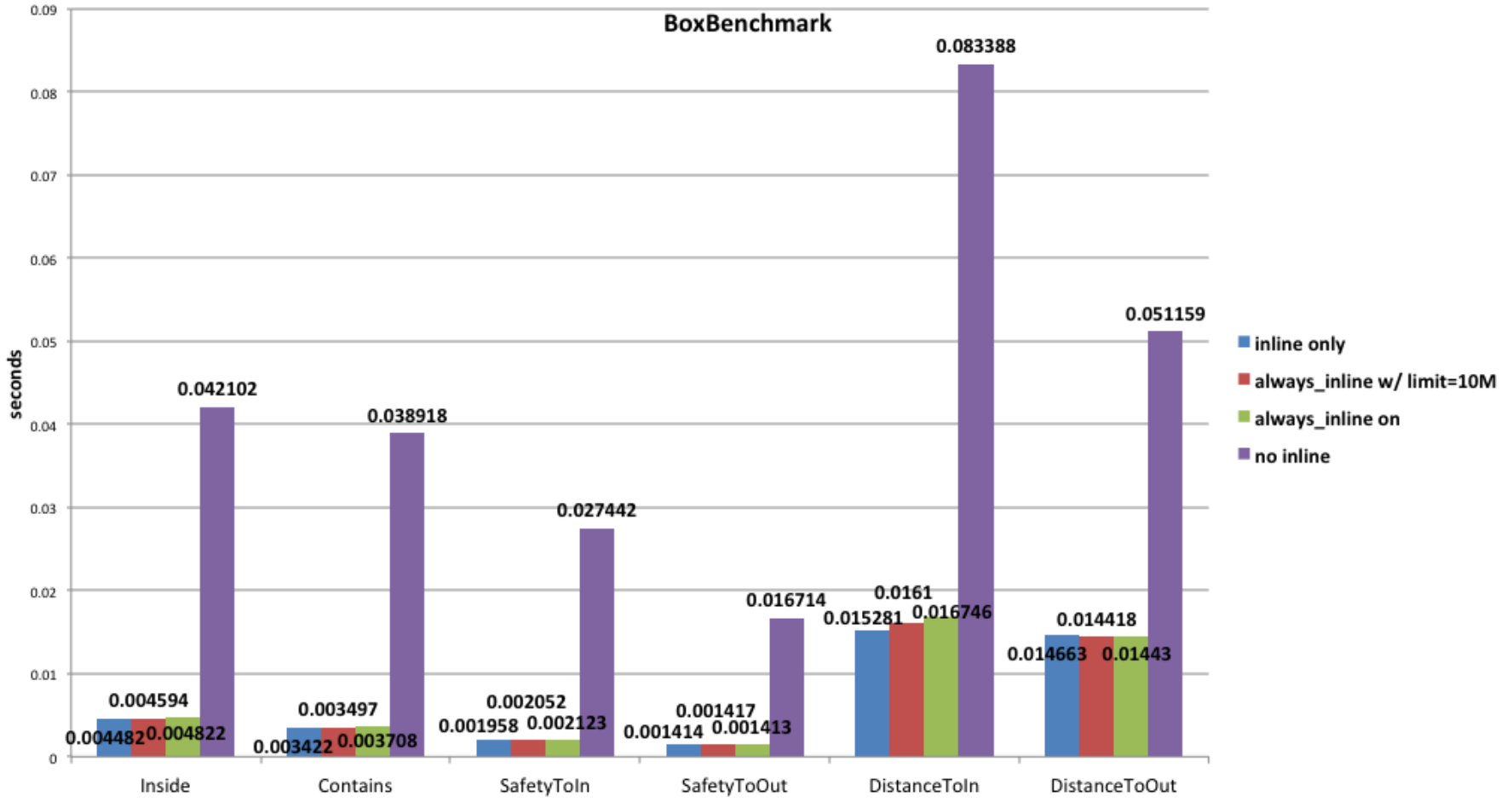
Intel Compiler Results

- *inline* is faster than others modifiers (or combination)
- On the worst case, *inline* is similar to *always_inline*
- Only one case that it really loose:
  Cone::DistanceToIn()

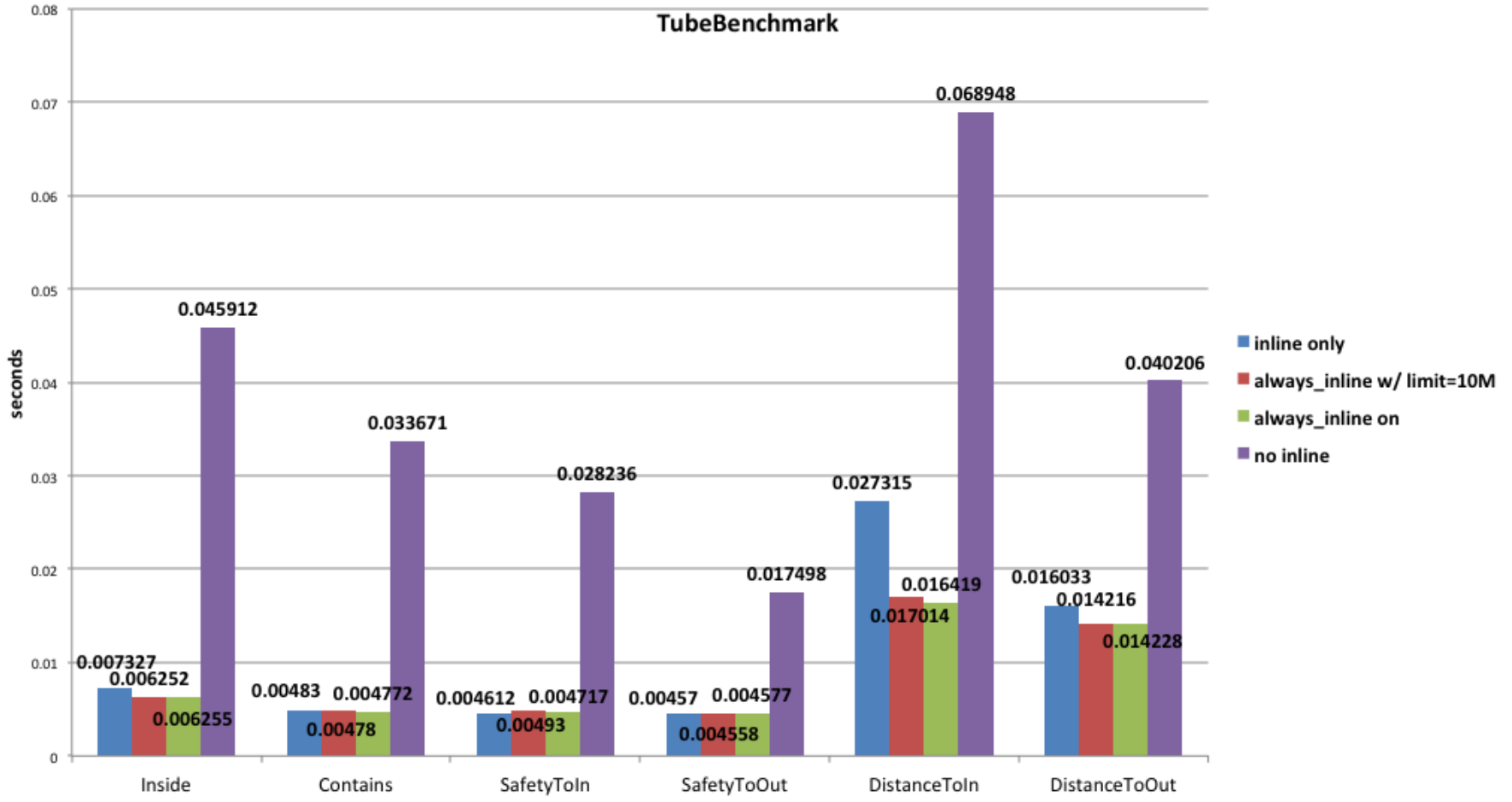
GCC Benchmark

- inline only
  - *VECGEOM_INLINE = inline*
  - Regular compilation for gcc without *-finline-limit=10000000*
- always_inline w/ limite=10M
  - *VECGEOM_INLINE = inline __attribute__((always_inline))*
  - Regular compilation (*-O2 -finline-limit=10000000 -ffast-math -ftree-vectorize -mavx -fabi-version=6 -Wall -fPIC*)
- always_inline
  - *VECGEOM_INLINE = inline __attribute__((always_inline))*
  - Regular compilation without *-finline-limit=10000000*
- no line
  - *VECGEOM_INLINE = inline*
  - Regular compilation + *-Winline -fno-inline*

GCC Results


BoxBenchmark
seconds
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.004482
0.004594
0.004822
0.042102
0.003422
0.003497
0.003708
0.038918
0.001958
0.002052
0.002123
0.027442
0.001414
0.001417
0.001413
0.016714
0.015281
0.0161
0.016746
0.083388
0.014663
0.014418
0.01443
0.051159
Inside
Contains
SafetyToIn
SafetyToOut
DistanceToIn
DistanceToOut
inline only
always_inline w/ limit=10M
always_inline on
no inline

GCC Results


TubeBenchmark
seconds
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.007327
0.006252
0.006255
0.045912
0.00483
0.00478
0.004772
0.033671
0.004612
0.00493
0.004717
0.028236
0.00457
0.004558
0.004577
0.017498
0.027315
0.017014
0.016419
0.068948
0.016033
0.014216
0.014228
0.040206
Inside
Contains
SafetyToIn
SafetyToOut
DistanceToIn
DistanceToOut
inline only
always_inline w/ limit=10M
always_inline on
no inline

GCC Results

- Time (using *-j 12*)
  - inline only: ± 1,0 minutes
  - always_inline w/ 10M: ± 2,0 minutes
  - always_inline: ± 2,0 minutes
  - no inline: ± 1,0 minutes
- Lib size
  - inline only: 22 MB
  - always_inline w/ 10M: 26 MB
  - always_inline: 26 MB
  - no inline: 44 MB

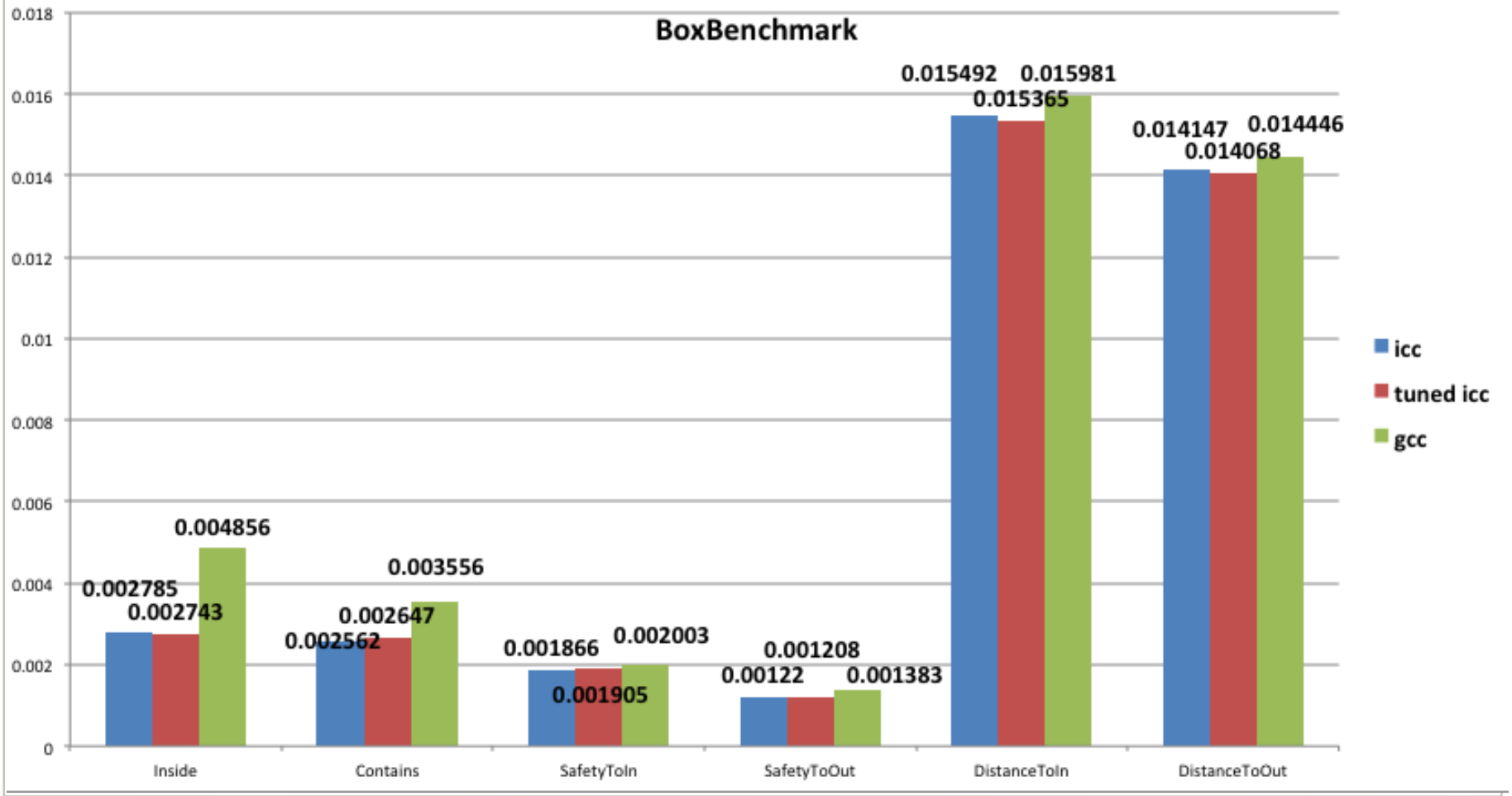
GCC Results

- *always_inline* with 10M is usually faster than other modifiers
- in worst case, *always_inline* is similar to *inline*
- *Box::DistanceToIn()* and *Trapezoid::DistanceToIn()* are exceptions (?)

Next steps…

- Build VecGeom with Profile Guided Optimization
  - There are some evidences that ICC will increase performance
- Compare the results
  - ICC & GCC with *inline* (only)
  - ICC & GCC with *always_inline w/ 10M*

Next steps…


BoxBenchmark
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0.002785
0.002743
0.004856
0.002562
0.002647
0.003556
0.001866
0.001905
0.002003
0.00122
0.001208
0.001383
0.015492
0.015365
0.015981
0.014147
0.014068
0.014446
Inside
Contains
SafetyToIn
SafetyToOut
DistanceToIn
DistanceToOut
icc
tuned icc
gcc