Intel Compiler Options and Optimizations

SNUG TechTalk

SciNet
www.scinet.utoronto.ca
University of Toronto
Toronto, Canada

December 14, 2011

SciNet

Outline

SciNet

Overview

Intel® Composer XE 2011 aka (v12.1)

- C (icc)
- C++ (icpc)
- FORTRAN (ifort)
- Threaded Building Blocks (TBB)
- Integrated Performance Primitives (IPP)
- Math Kernel Libraries (MKL)

SciNet

Optimizations

Optimization Levels

- **-O0** disable optimization
- **-O1** optimizes for code size
- **-O2** optimizes for speed (default)
- **-O3** **-O2** plus more aggressive optimizations

SciNet

Optimizations

Optimization Levels

- **-O0** disable optimization
- **-O1** optimizes for code size
- **-O2** optimizes for speed (default)
- **-O3** **-O2** plus more aggressive optimizations

From the Manual

"The **-O3** option is particularly recommended for applications that have loops that do many floating-point calculations or process large data sets."

SciNet

Optimizations

-O2 Optimizations

- intrinsic inlining
- inlining
- constant propagation
- forward substitution
- routine attribute propagation
- variable address-taken analysis
- dead static function elimination
- removal of unreferenced variables
- constant propagation
- copy propagation
- dead-code elimination
- global register allocation
- global instruction scheduling and control speculation
- loop unrolling
- optimized code selection
- partial redundancy elimination
- strength reduction/induction variable simplification
- variable renaming
- exception handling optimizations
- tail recursions
- peephole optimizations
- structure assignment lowering and optimizations
- dead store elimination

Optimization Terminology
Inlining

Inlining

Replaces the function call with the actual functions code.

SciNet

Optimization Terminology
Inlining

Inlining

Replaces the function call with the actual functions code.

Original

```
int func(int &x,int &y) { return 4*x+3*y; }

int main(){
  int x=4, y=3;
  int b=fun(x,y)
}
```

SciNet

Optimization Terminology
Inlining

Inlining

Replaces the function call with the actual functions code.

Original

```
int func(int &x,int &y) { return 4*x+3*y; }

int main(){
  int x=4, y=3;
  int b=fun(x,y)
}
```

Inlined

```
int main(){
  int x=4,y=3;
  int b= 4*x+3*y;
}
```

et

Optimization Terminology
Branch Elimination

Original

```
if ( x < x1 ) {
  a = a0 + a1;
} else if ( x < x2 ) {
  a = a0 - a1;
} else if ( x < x3 ) {
  a = a0 * a1;
} else if ( x < x4 ) {
  a = a0 / a1;
} else   {
  a = a0;
}
```

Optimizer Approaches

- static branch elimination
- compute all cases and conditions, then pick the correct one
- replace with switch statements, jump tables
- branch re-alignment

et

Optimizations

-O3 Additional Optimizations

- Loop Blocking for cache
- Loop Permutation or Interchange
- Loop Distribution
- Loop Fusion
- Loop Unrolling
- Unroll and Jam
- Loop Blocking or Tiling
- Loop Reversal
- Loop Peeling
- Loop Rerolling
- Profile-Guided Loop Unrolling
- Code Replication to eliminate branches
- Memory-access optimizations
- Data Prefetching
- Scalar Replacement
- Partial-Sum Optimization
- Predicate Optimization
- Data Transformation: Malloc Combining and Memset Combining
- Memset and Memcpy Recognition
- Statement Sinking for Creating Perfect Loopnests

SciNet

Optimization Terminology
Loop Unrolling

Original

```
for (int x=0; x < 100; x++)
{
  func(x);
}
```

SciNet

Optimization Terminology
Loop Unrolling

Original

```
for (int x=0; x < 100; x++)
{
  func(x);
}
```

Optimized

```
for (int x = 0; x < 100; x+=5)
{
  func(x);
  func(x+1);
  func(x+2);
  func(x+3);
  func(x+4);
}
```

SciNet

Optimization Terminology
Loop Collapsing

Original

```
int a[100][300];
for (int i = 0; i < 300; i++)
  for (int j = 0; j < 100; j++)
     a[j][i] = 0;
```

SciNet

Optimization Terminology
Loop Collapsing

Original

```
int a[100][300];
for (int i = 0; i < 300; i++)
  for (int j = 0; j < 100; j++)
     a[j][i] = 0;
```

Optimized

```
int a[100][300];
int *p = &a[0][0];

for (int i = 0; i < 30000; i++)
  *p++ = 0;
```

SciNet

Optimization Terminology
Loop Fusion

Original

```
int x[100], y[100];
for (int i = 0; i < 100; i++)
   x[i] = 1;
for (int i = 0; i < 100; i++)
   y[i] = 2;
```

SciNet

Optimization Terminology
Loop Fusion

Original

```
int x[100], y[100];
for (int i = 0; i < 100; i++)
   x[i] = 1;
for (int i = 0; i < 100; i++)
   y[i] = 2;
```

Optimized

```
int  x[100], y[100];
for (int i = 0; i < 100; i++)
{
  x[i] = 1;
  y[i] = 2;
}
```

SciNet

Optimization Terminology
Loop Peeling

Original

```
int p = 10;
for (int i=0; i<10; ++i)
{
  y[i] = x[i] + x[p];
  p = i;
}
```

SciNet

Optimization Terminology
Loop Peeling

Original

```
int p = 10;
for (int i=0; i<10; ++i)
{
  y[i] = x[i] + x[p];
  p = i;
}
```

Optimized

```
y[0] = x[0] + x[10];
for (int i=1; i<10; ++i)
{
  y[i] = x[i] + x[i-1];
}
```

SciNet

Optimizations

System Specific

- **-march="cpu"** optimize for a specific cpu
- **-mtune="cpu"** produce code only for a specific cpu
- **-msse3,-msse4,-mavx, etc.** level of SIMD and vector instructions

SciNet

Optimizations

System Specific

- **-march="cpu"** optimize for a specific cpu
- **-mtune="cpu"** produce code only for a specific cpu
- **-msse3,-msse4,-mavx, etc.** level of SIMD and vector instructions

Use this instead!

-xHost optimize and tune for the compiling CPU

SciNet

Optimizations

System Specific

- **-march="cpu"** optimize for a specific cpu
- **-mtune="cpu"** produce code only for a specific cpu
- **-msse3,-msse4,-mavx, etc.** level of SIMD and vector instructions

Use this instead!

-xHost optimize and tune for the compiling CPU

GPC Recommendations

-xHost -O3

SciNet

Optimization Terminology
Vector Extensions

Intel x86_64 extensions

- Streaming SIMD Extensions (SEE1 - SSE4.2)
- AVX

Original x86

Add two single precision vectors requires four floating-point addition instructions.

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

SciNet

Optimization Terminology
Vector Extensions

Intel x86_64 extensions

- Streaming SIMD Extensions (SEE1 - SSE4.2)
- AVX

Original x86

Add two single precision vectors requires four floating-point addition instructions.

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

SSE

A single 128-bit 'packed-add' replaces four scalar addition instructions.

```
movaps xmm0, [v1]; xmm0 = v1.w | v1.z | v1.y | v1.x
addps xmm0, [v2];  xmm0 = v1.w+v2.w | v1.z+v2.z | v1.y+v2.y | v1.x+v2.x
movaps [vec_res], xmm0
```

et

Optimizations

Inter Procedural Optimizations (IPO)

- inlining
- constant propagation
- mod/ref analysis
- alias analysis
- forward substitution
- routine key-attribute propagation
- address-taken analysis
- partial dead call elimination
- symbol table data promotion
- common block variable coalescing
- dead function elimination
- unreferenced variable removal
- whole program analysis
- array dimension padding
- common block splitting
- stack frame alignment
- structure splitting and field reordering
- formal parameter alignment analysis
- C++ class hierarchy analysis
- indirect call conversion
- specialization
- Passing arguments in registers to optimize calls and register usage

SciNet

Optimizations

Inter Procedural Optimizations

- **-ip** single file ip optimization
- **-ipo** multiple file or whole program optimization

SciNet

Optimizations

Inter Procedural Optimizations

- **-ip** single file ip optimization
- **-ipo** multiple file or whole program optimization

Profile Guided Optimizations

- **-prof-gen** instrument code to generate profile
- **-prof-use** use profile to guide optimization

SciNet

Optimizations

Inter Procedural Optimizations

- **-ip** single file ip optimization
- **-ipo** multiple file or whole program optimization

Profile Guided Optimizations

- **-prof-gen** instrument code to generate profile
- **-prof-use** use profile to guide optimization

Flank Speed

-fast enables **-xHost -O3 -ipo -no-prec-div -static**

SciNet

Floating Point Math

-fpmodel

- **fast=1** default
- **fast=2** most aggressive
- **precise** value-safe optimizations on intermediate operations
- **except** strict floating point semantics
- **strict** disables all "fast-math" options

If Required

For floating point consistency and reproducibility use:
-fpmodel precise -fpmodel except

SciNet

Memory Model

Seen this error?

relocation truncated to fit: R_X86_64_PC32

SciNet

Memory Model

Seen this error?

relocation truncated to fit: R_X86_64_PC32

-mcmodel=

- **small** code and data restricted to the first 2GB of address space
- **medium** code restricted to the first 2GB of address space
- **large** no restrictions

SciNet

Intel Math Kernel Library

MKL Components

- BLAS
- LAPACK
- ScaLAPACK
- FFT
- PBLAS
- BLACS
- plus others

SciNet

Intel Math Kernel Library

Link Line - MKL 10.3 or less

-lmkl_intel_lp64 -lmkl_sequential -lmkl_core -lpthread -lm

Link Line - Composer XE 2011

- **-mkl=sequential** no-threaded versions (serial)
- **-mkl=parallel** threaded (openmp)
- **-mkl=cluster** for ScaLAPACK, FFT, BLACS

Link Line Advisor

http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

SciNet

Documentation

Intel Documentation

http://software.intel.com/en-us/articles/intel-parallel-studio-xe-for-linux-documentation/

Compiler Optimization flags

http://software.intel.com/sites/products/collateral/hpc/compilers/compiler_qrg12.pdf

White Paper on Floating Point

https://support.scinet.utoronto.ca/wiki/images/f/f2/FP_Consistency.pdf

SciNet