Enhancing C++ Application Performance with Intel Code Modernization

STAC 2016

First Derivatives plc

DATA | TRADING | RISK

First Derivatives

“Combining Domain Knowledge and Technical Expertise”

- **Domain Knowledge** – Capital Markets
- **Technical Expertise** – Vector Programming

What is Code Modernization?

Adapting and tuning previously written code libraries in order to take full advantage of advances in areas such as:

- Instruction set architecture
- Increased core & thread count
- Compilers and processors
- Optimised support libraries

How is this done?

There is a well defined and concise framework for approaching the code base. Broadly speaking:

- Performance Analysis with VTune
- Code Refactoring
- Vectorization
- Parallelization

Code Restructuring

Assessing current code layout and making adaptations as necessary with regards to:

- Minimizing loops and logical flow control statements
- Replacing standard math library functions (exp, log etc.) with Intel's optimized math library (MKL)
- Aligning vectors with memory size so that calls are made to cache as often as possible
- Remove dependencies between loops

Parallelize Outer Loops

Used in cases where there is at least one nested loop.

Simple example: square each entry in a table

We would normally run over each row and then each element of that row.

If however we have, say, 60 threads available then we would run over 60 rows simultaneously.

= **60 times speed up**

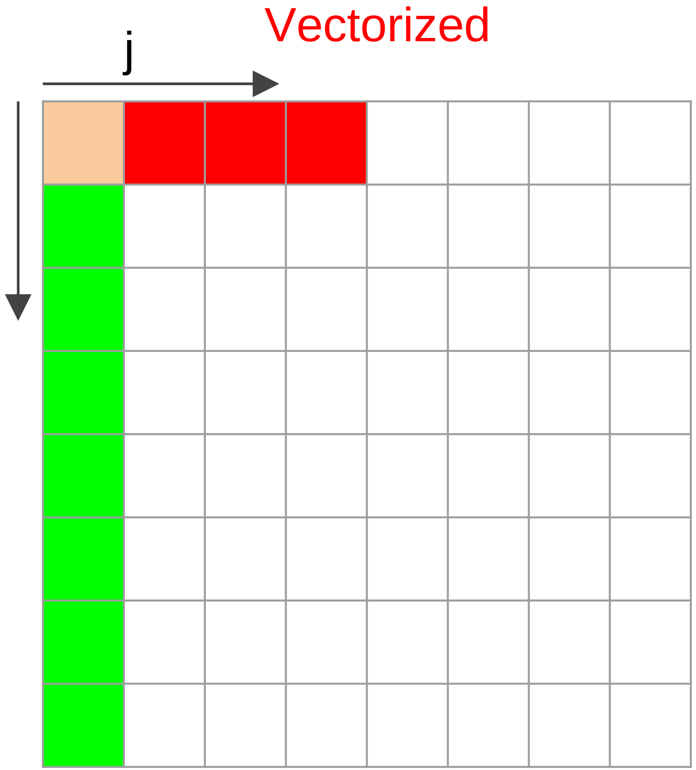
Vectorize Inner Loops

- Applied on a sequential loop iteration i.e. running over values in a row
- AVX (Advanced Vector Extensions) allows each thread to process up to 4 identical instructions at the same time
- Using previous example, in each row, the thread will calculate the square of the first four elements at once, then the next four and so on

= 4 times speed up

Putting It All Together

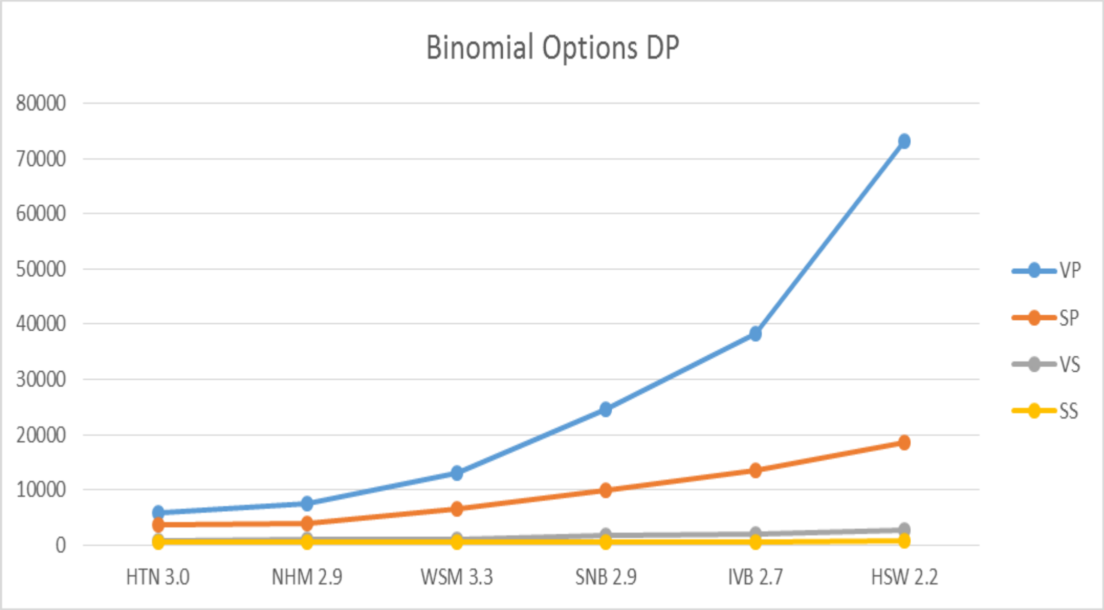
```
#pragma omp parallel num_threads(60)
#pragma omp for

for(int i=0; i<N,i++)
        {
                Matrix[ i ][ : ] *= Matrix[ i ][ : ];
        }
```



Vector notation replaces j loop.

- **4 x 60 = <u>240 times</u> speed up**

Modernization Matters



2007 Intel Xeon Processor X5472 (formerly codenamed Harpertown)	2009 Intel Xeon Processor X5570 (formerly codenamed Nehalem)	2010 Intel Xeon Processor X5680 (formerly codenamed Westmere)	2012 Intel Xeon Processor 2600 E5 family (formerly codenamed Sandy Bridge)	2013 Intel Xeon Processor E5 2600 v2 family (formerly codenamed Ivy Bridge)	2014 Future Intel Xeon Processor Codenamed Haswell
4C	4C	6C	8C	12C	14C

VP = Vectorized & Parallelized
SP = Scalar & Parallelized
VS = Vectorized & Single-Threaded
SS = Scalar & Single-Threaded

Not STAC Benchmarks

Thank You