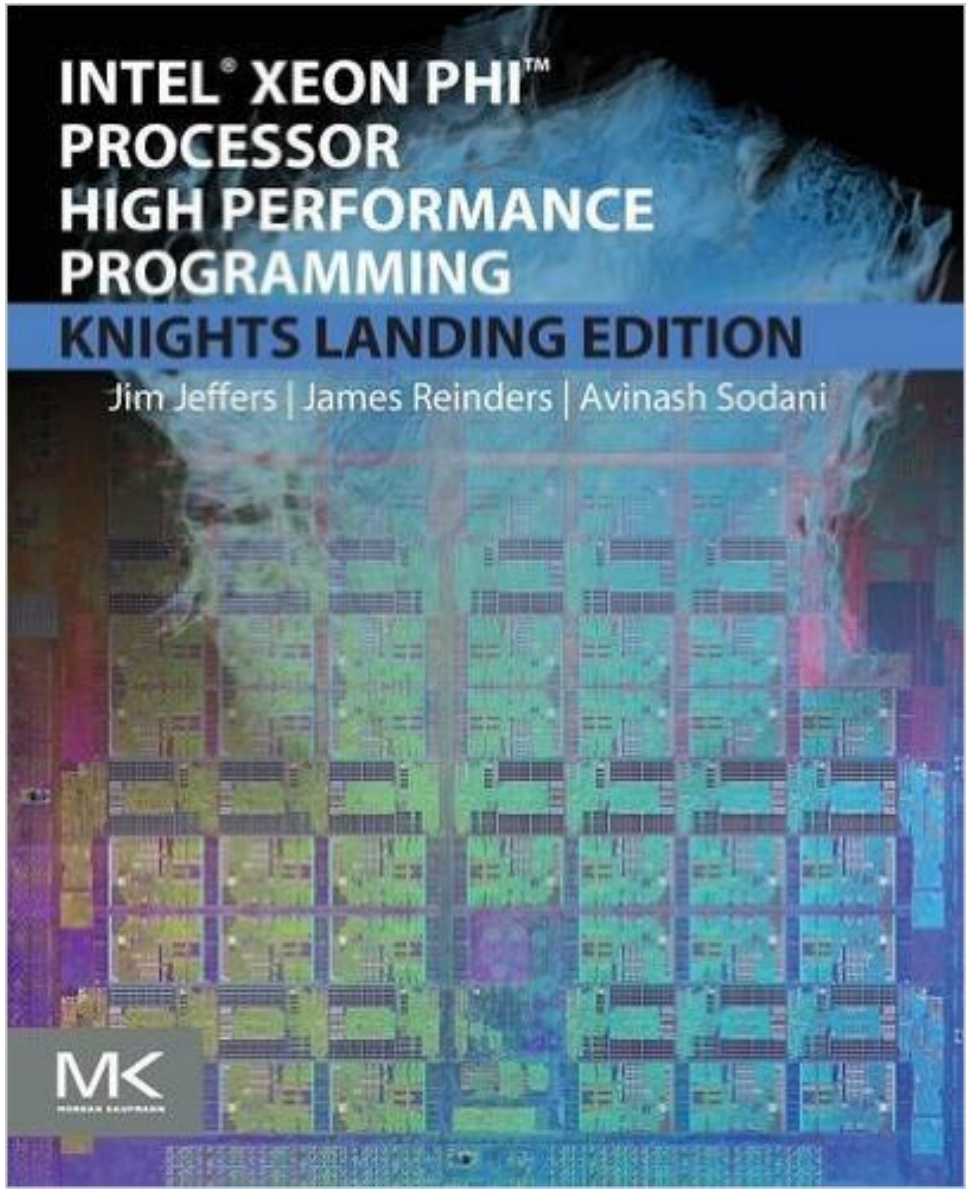
Tips and Tricks for FORTRAN on the Intel KNL

Dane Skow
16 March 2017
Research Computing Meetup

What is KNL ?

- “Knight’s Landing” is the code name for the newly released Intel Xeon Phi Processor
- It has up to 86 cores with 16GB onboard fast memory (MCDRAM).
- It is able to run all x86 code natively
  - NOT a coprocessor (unlike its predecessor)
- It has 512-bit wide registers for increased vectorization possibilities

A Door Prize !!



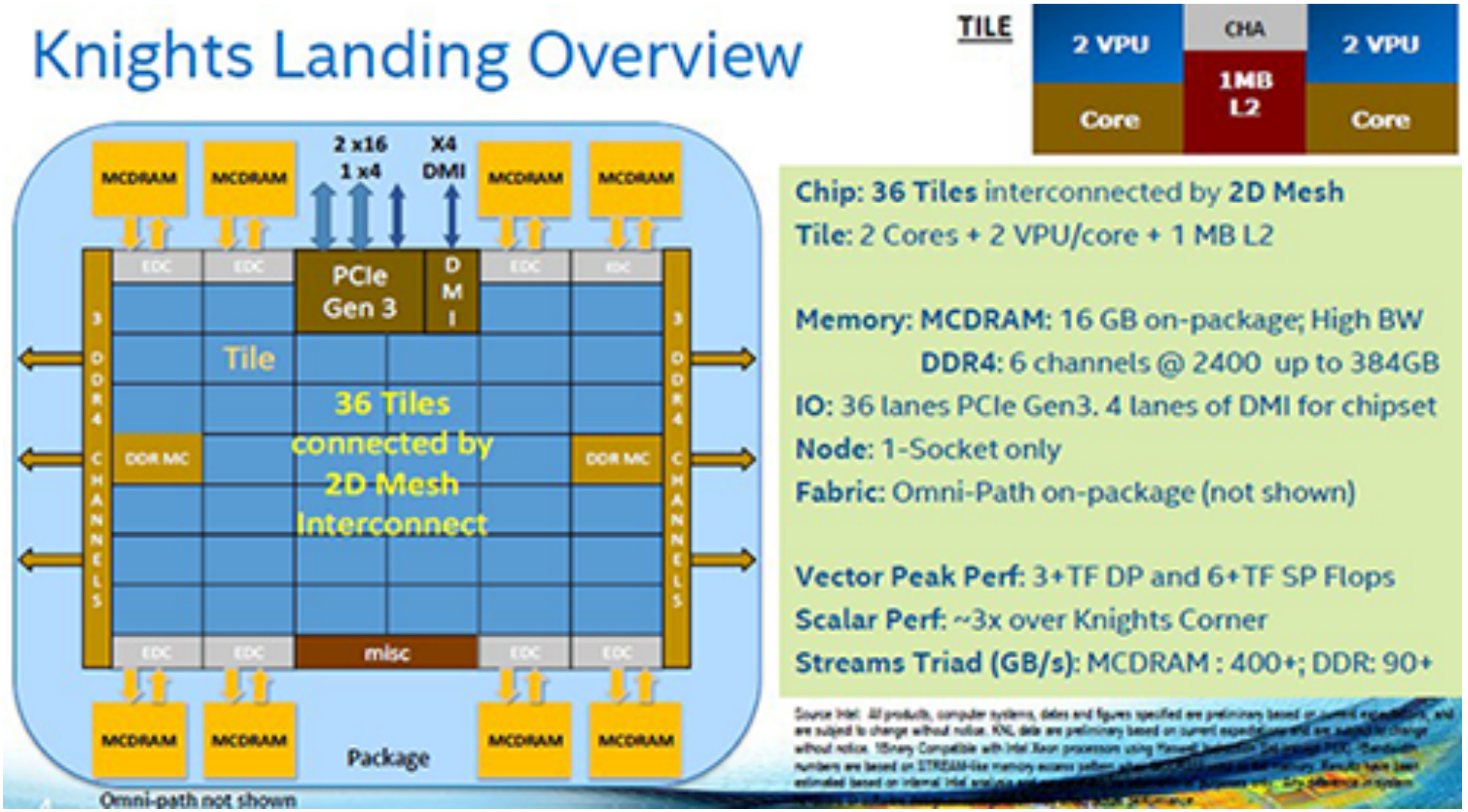
Useful book covering parallelization, vectorization, optimization techniques in general and for the KNL.

Everyone attending the meetup may have one. Pick it up before you leave.

KNL on Summit

- Summit will have 20 KNL nodes operational by June 1, 2017
  - 1 KNL 7250 processor per node (86 cores)
  - 96 GB of RAM
  - 100+ GB local SSD disk
  - 1 OPA low latency interconnect (100 Gbps)
  - 10 Gbps Ethernet connection

KNL Architecture



MCDRAM

- 3 modes of operation to choose from at boot time
  - Cache
    - MCDRAM is cache for DRAM
  - Flat
    - MCDRAM is treated like standard memory
  - Hybrid
    - 50% cache and 50% flat
    - 75% cache and 25% flat

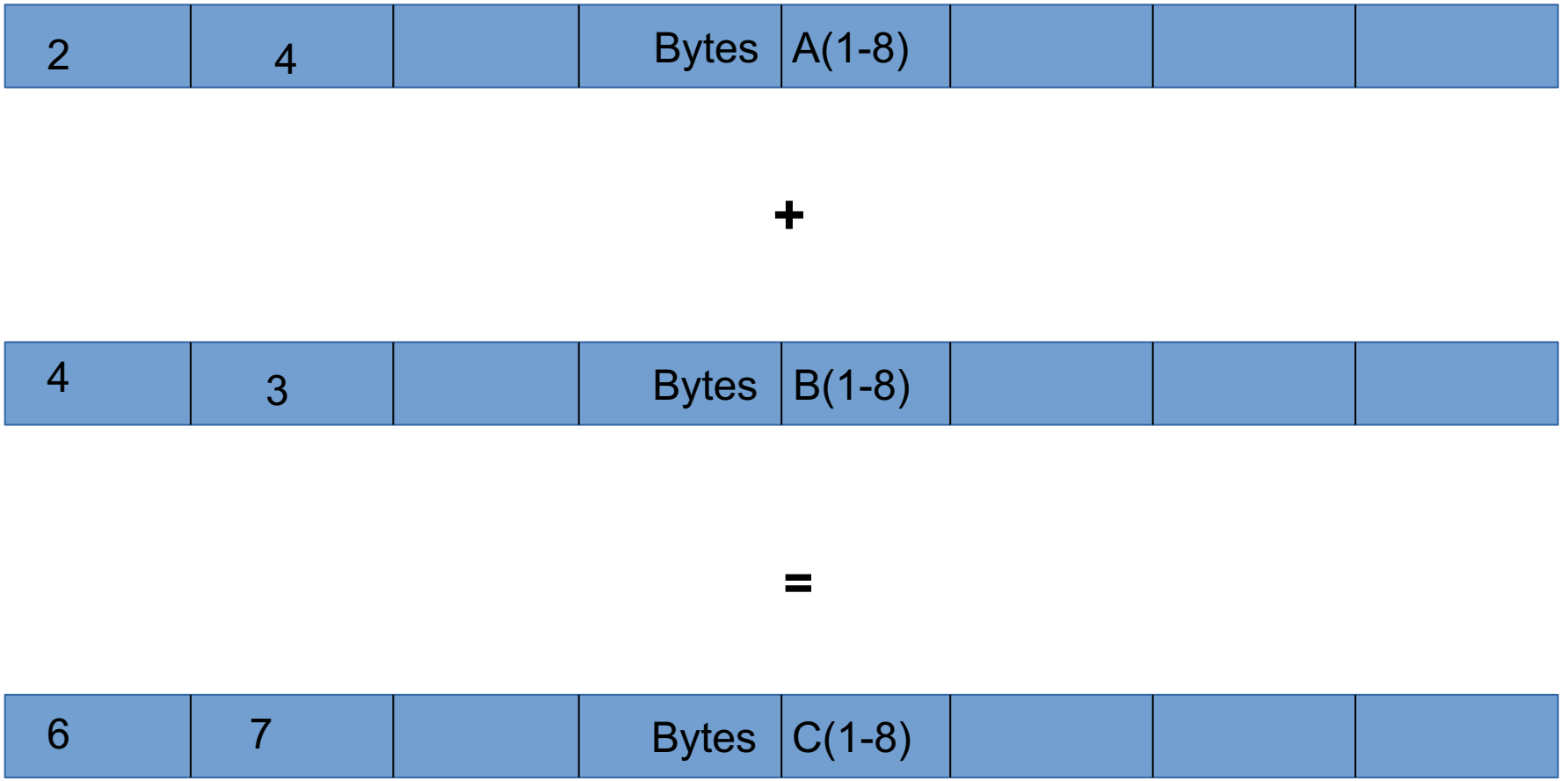
How to get started ?

- FORTRAN: add -xMIC_AVX2 to your FORTRAN Compiler flags.

What is Vectorization ?

- Vectorization is the application of a single operation to multiple data elements (the “vector”)
- Think of it as the “bite-size” of the math unit inside the CPU.
  - The processor takes a “bite” of data (64 bytes long in the case of the KNL) and operates on all that data at the same time.
  - This means that 8 normal math operations get done at the same time inside the math unit of the CPU.

Vector Math



Enemies of Compiler Vectorization

- Try to have the largest loop be the innermost loop.
- Try to avoid “IF/THEN” constructs
  - Multiplying by a mask is one alternate technique.
- Avoid data dependent exit conditions
- Avoid backward loop-carried dependencies
  - ie. statement 2 of iteration 1 should not require statement 1 of iteration 2 to be done after it.
- Try to use code that accesses contiguous memory locations
  - “Striding” in “bite-sized chunks” is most efficient.

Vectorization Tools

- Intel
  - Advisor
  - VTune Accelerator
- Allinea
- Valgrind
  - Useful for memory usage investigations
- HPC Toolkit
  - (http://www.hpctoolkit.org/)

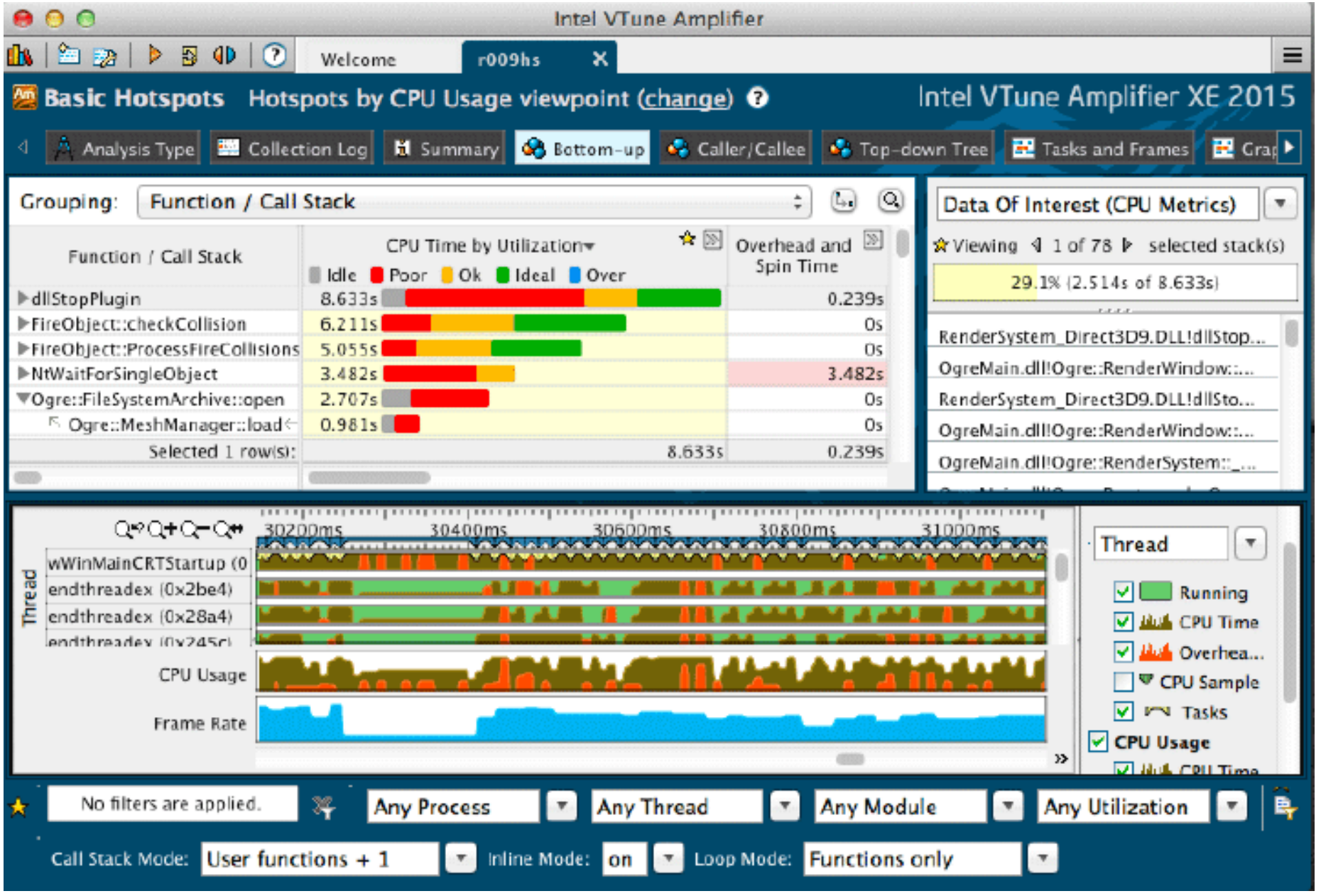
6 Step method to vectorizing your code

- Measure baseline performance
  - Use release code not debugging code
- Determine hotspots
  - Intel Vtune Amplifier a useful tool here
- Determine loop candidates
  - Intel compiler vec-report useful tool here

6 step method (cont'd)

- Get advice
  - Intel advisor can be used here
- Implement vectorization recommendations
  - One useful test is to see if executing the loop backwards would change the results.
- Repeat

Intel VTune Amplifier



Helpful URLs

- List of useful Environment Variables for Intel FORTRAN compiler
  - https://software.intel.com/en-us/node/680054
- Vectorization Toolkit
  - http://lotsofcores.com/intelvectkit
- HPC Toolkit
  - http://hpctoolkit.org/
- Stack Overflow
  - https://stackoverflow.com/

Research Computing Github Site

- Github.com/researchcomputing