Programming the Intel® Xeon Phi™ Coprocessor

Tim Cramer

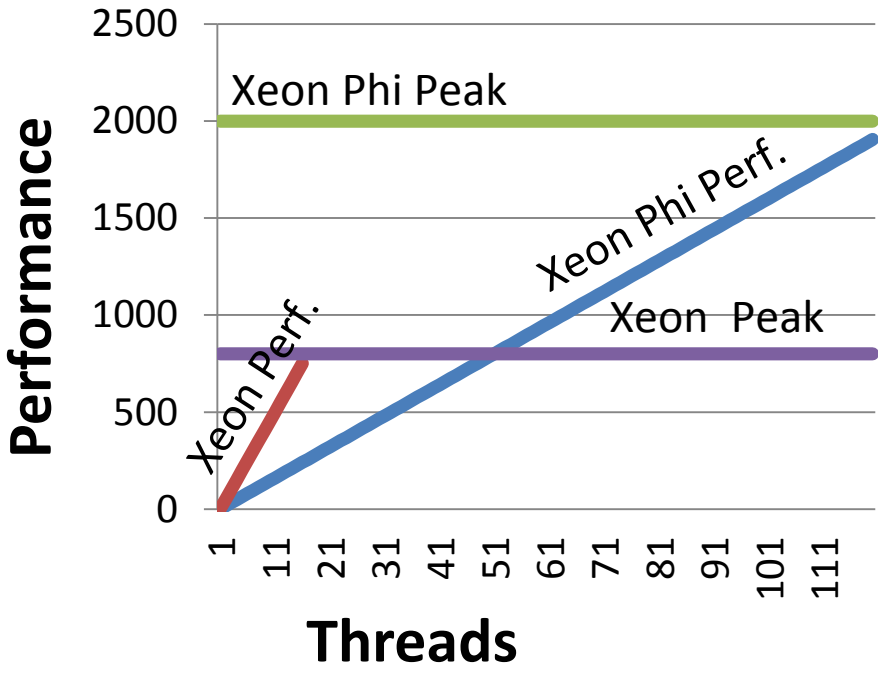
19.03.2015

Agenda

- **Motivation**
- **Many Integrated Core (MIC) Architecture**
- **RWTH MIC Environment**
- **Programming Models**
  - → Native
  - → Pragma-based programming
    - → ~~Language Extension for Offload (LEO)~~
    - → OpenMP 4 -> Christian Terboven will talk about that later
  - → ~~Symmetric (with Message Passing)~~
- **Performance Expectations**
- **Debugging**

Motivation

- Demand for more compute power
- Reach higher performance with more threads
- Power consumption: Better performance / watt ratio
- GPUs are one alternative, but: CUDA is hard to learn / program
- Intel Xeon Phi can be programmed with established programming paradigms like OpenMP, MPI, Pthreads



Source: James Reinders, Intel

What is the Intel Xeon Phi?

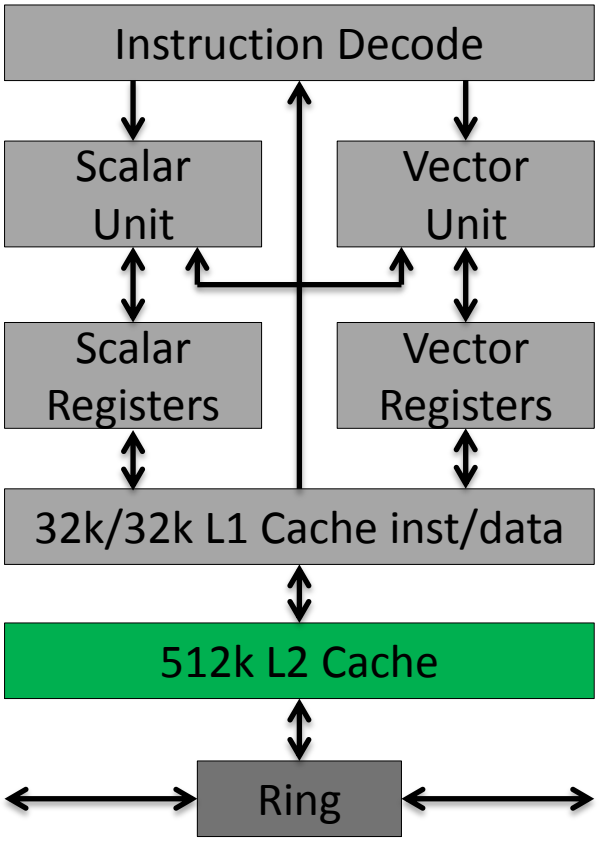
When to Use the Intel Xeon Phi?

- **Xeon Phi is not intended to replace Xeon -> choose the right vehicle**

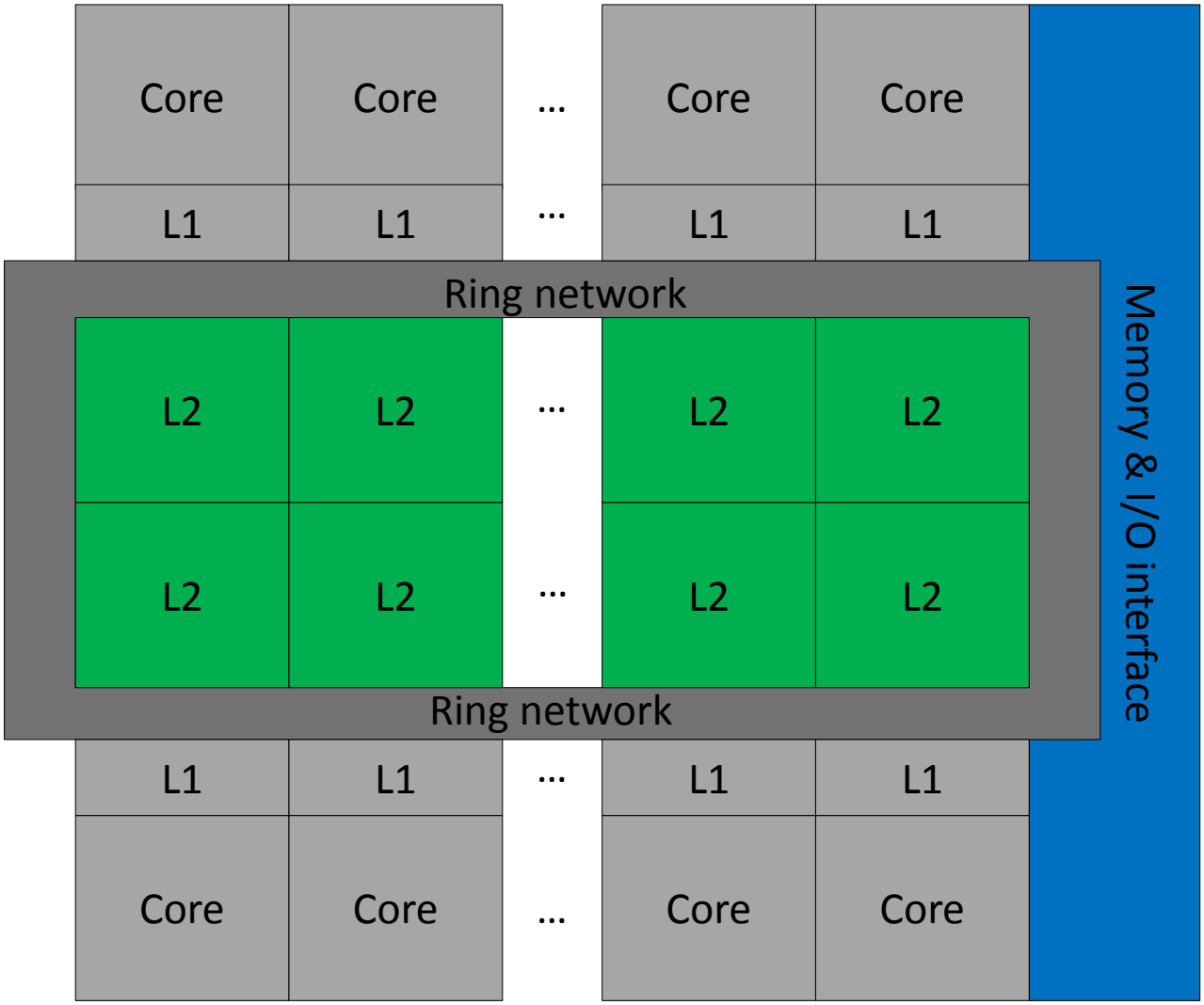
Architecture (1/2)

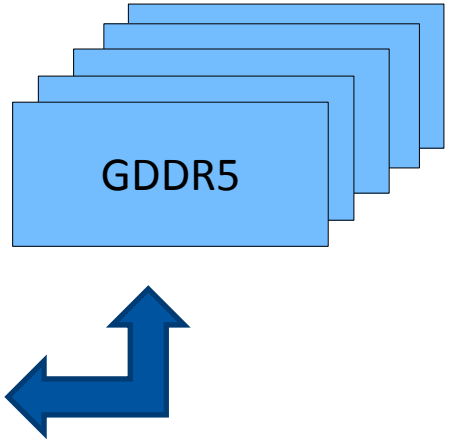
Source: Intel

Intel Xeon Phi Coprocessor
- 1 x Intel Xeon Phi @ 1090 MHz
- 60 Cores (in-order)
- ~ 1 TFLOPS DP Peak
- 4 hardware threads per core (SMT)
- 8 GB GDDR5 memory
- 512-bit SIMD vectors (32 registers)
- Fully-coherent L1 and L2 caches
- Plugged into PCI Express bus

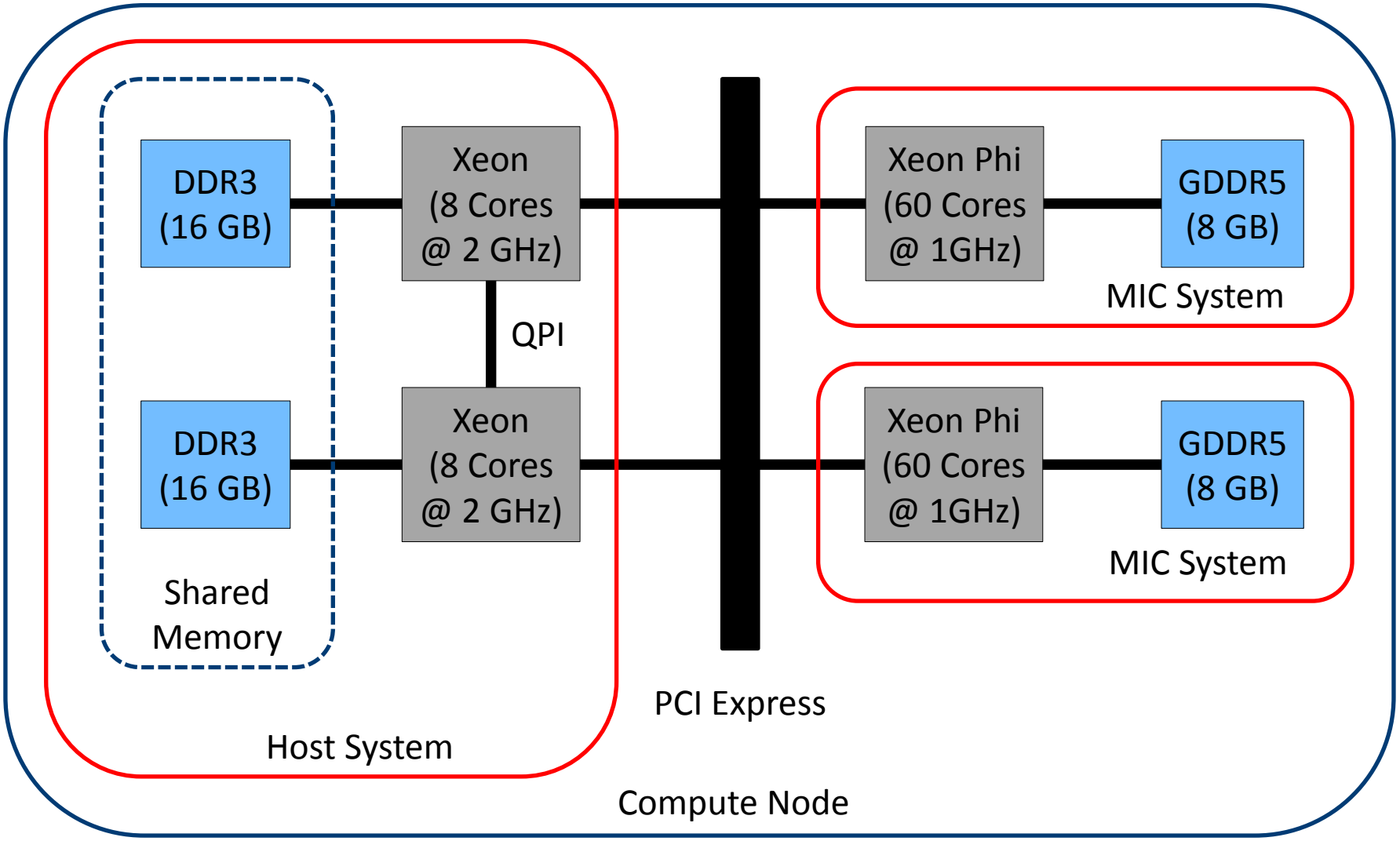


Architecture (2/2)





Xeon Phi Nodes at RWTH Aachen



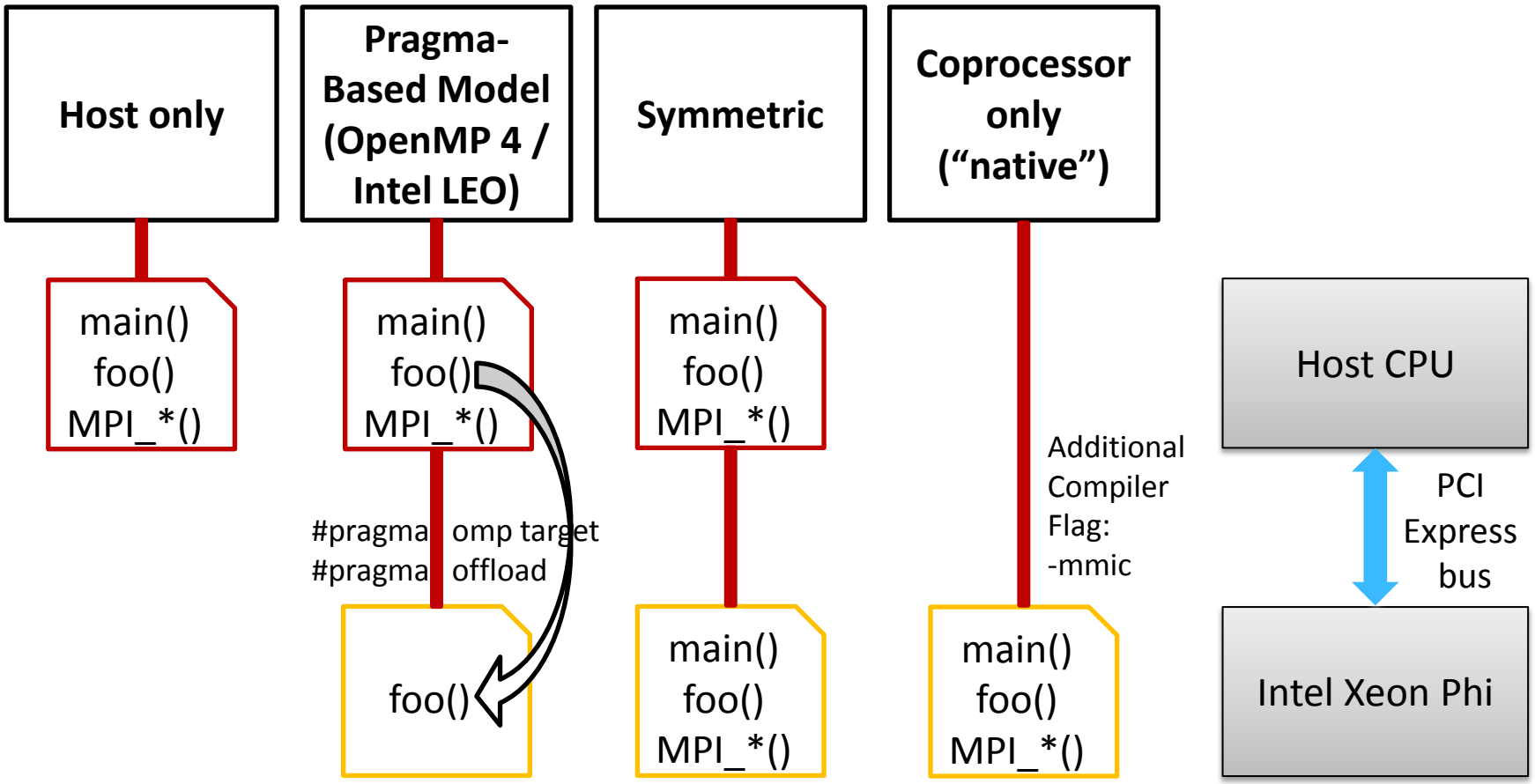
RWTH MIC Environment

- **Hardware**
  - → 9 nodes with two CPU and two Coprocessors each
- **Access**
  - → Your account has to be activated (contact **servicedesk@itc.rwth-aachen.de**)
- **File system**
  - → `HOME` and `WORK` mounted `in /{home,work}/<timid>`
  - → `HPCWORK` not available
- **Native usage**
  - → Linux is running on the device, but many features missing
  - → No modules available on the device
  - → Only one compiler version and one MPI version supported
- **MPI**
  - → Special module `intelmpi/5.0mic` sets up specific environment

Frontend

- **Interactive usage (frontend)**
  - → Login to “normal” frontend, e.g.,
    ```
    $ ssh cluster.rz.rwth-aachen.de
    ```
  - → Login to the host system
    ```
    $ ssh cluster-phi.rz.rwth-aachen.de
    ```
  - → Login to the coprocessor
    ```
    $ ssh cluster-phi-mic0.rz.rwth-aachen.de
    $ ssh cluster-phi-mic1.rz.rwth-aachen.de
    ```
  - → Login to the coprocessor only possible from the host (local ssh keys)
  - → Use the two MICs for development / job preparation
  - → No production runs allowed
  - → Restart of the system at 4 am every morning.
  - → Load balancing with MPIEXEC wrapper (refer to “Symmetric Execution”)
- **Documentation**
  - → **https://doc.itc.rwth-aachen.de/display/CC/Intel+Xeon+Phi+cluster**

Programming Models



Coprocessor only (“native”)

- **Cross-compile for the coprocessor**
  - → instruction set on the CPU and the coprocessor is similar, but identical
  - → easy (just add “-mmic”, login with ssh and execute)
  - → OpenMP, posix threads, OpenCL, MPI usable
  - → analyze benefit for hotspots
  - → very slow IO
  - → poor single thread performance
  - → host CPU will be bored
  - → only suitable for highly parallelized / scalable codes

OpenMP 4: Covered in Christian's presentation after this talk

Performance Expectations (1/2)

- **“WOW, 240 hardware threads on a single chip! My application will just rock!”**
  - → You really believe that?
  - → Remember the limitations!
    - →In-order cores
    - →limited hardware prefetching
    - →Running with 1 GHz only
    - →Small Caches (2 levels)
    - →Poor single thread performance
    - →Small main memory
    - →PCIe as bottleneck + offload overhead

Performance Expectations (2/2)

- **Comparison with our MPI nodes (2 x Intel Westmere CPUs)**

	Xeon Phi 5110P	2 x Xeon X5675
Frequency [GHz]	3.07	1.09
Cores	60	12
Peak Performance [GFLOPS]	144	1171
Peak Performance (no SIMD) [GFLOPS]	72	146
Memory Bandwidth (STREAM) [GB/s]	40	160
Memory Bandwidth (single core) [GB/s]	12	5

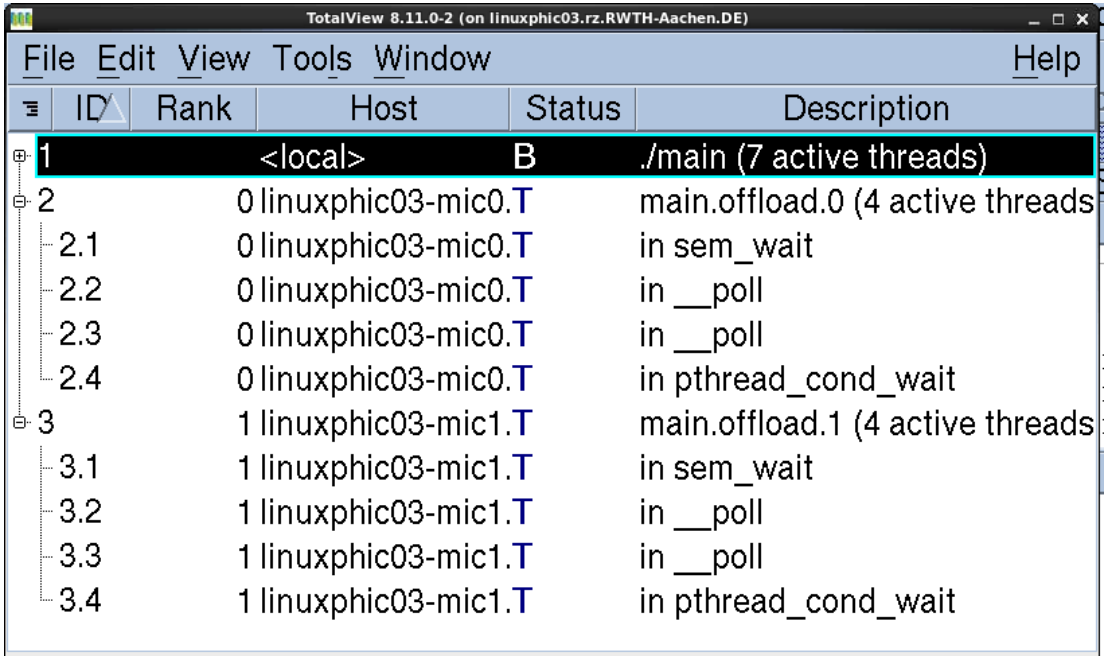
→ Only scalable, vectorizable codes benefit of the Xeon Phi at the moment

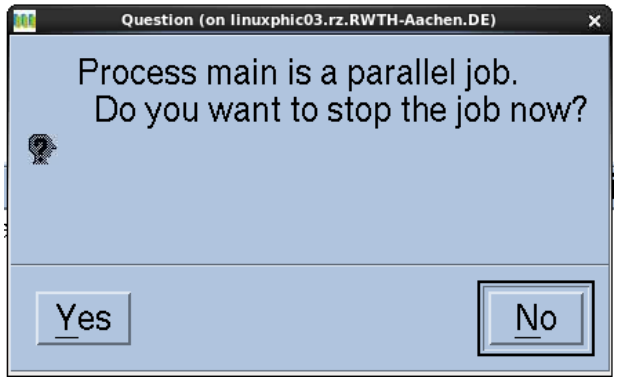
→ Performance is not for free, tuning will be necessary

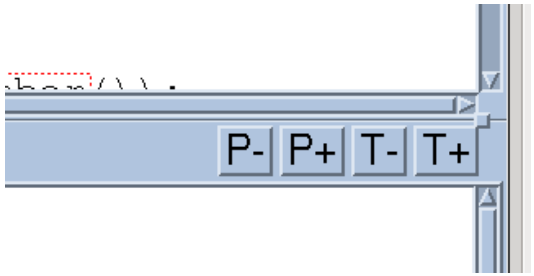
Debugging

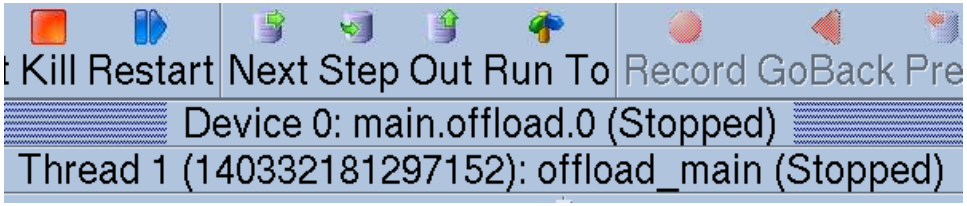
- **Latest and greatest TotalView works**

  → `icc -g -offload-option,mic,compiler,"-g" -o main main.c`









The End