ICHEC
Irish Centre for High-End Computing

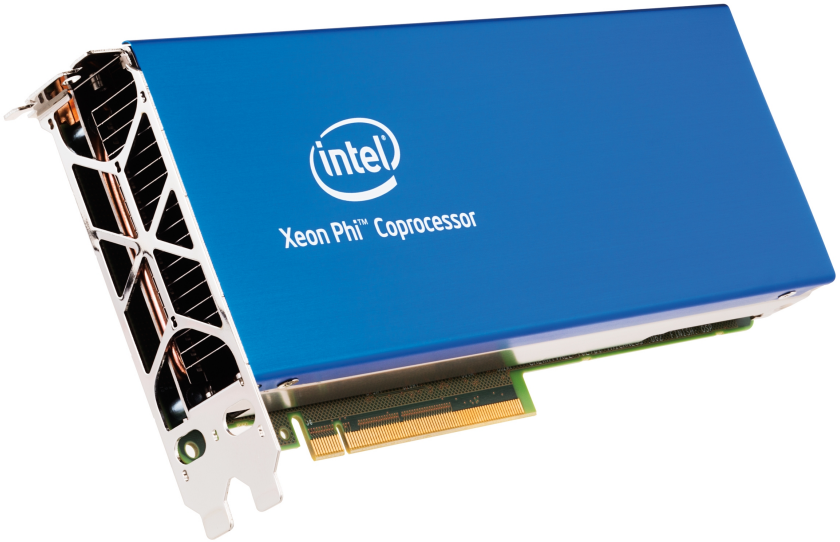
Running HARMONIE on Xeon Phi Coprocessors

Enda O'Brien
Irish Centre for High-End Computing

Science Foundation Ireland
Ireland's EU Structural Funds Programmes 2007 - 2013
Co-funded by the Irish Government and the European Union
EUROPEAN REGIONAL DEVELOPMENT FUND
AN ROINN OIDEACHAIS AGUS EOLAIOCHTA | DEPARTMENT OF EDUCATION AND SCIENCE
HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

Disclosure

Intel is funding ICHEC to port & optimize some applications, including HARMONIE, to Xeon Phi coprocessors.



Motivating Questions

Hypothetical:

- How much (human) effort is worth investing to obtain a **10 x** performance speedup, *if available*, from hardware accelerators?
- How about **2 x** speedup?
- Or **20%** speedup?

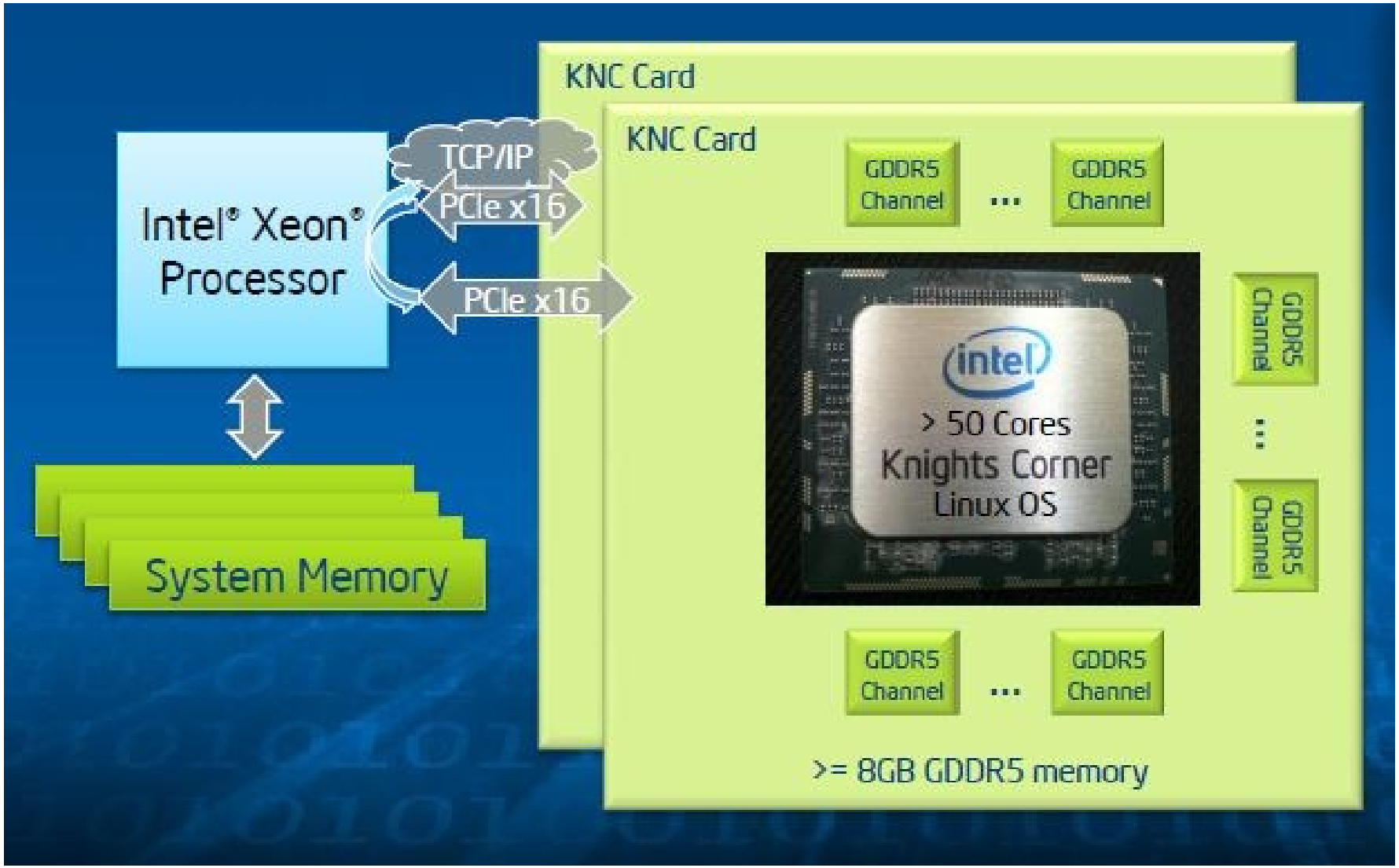
Practical:

- Which provides more value: an extra compute node, or an accelerator?

Some Jargon

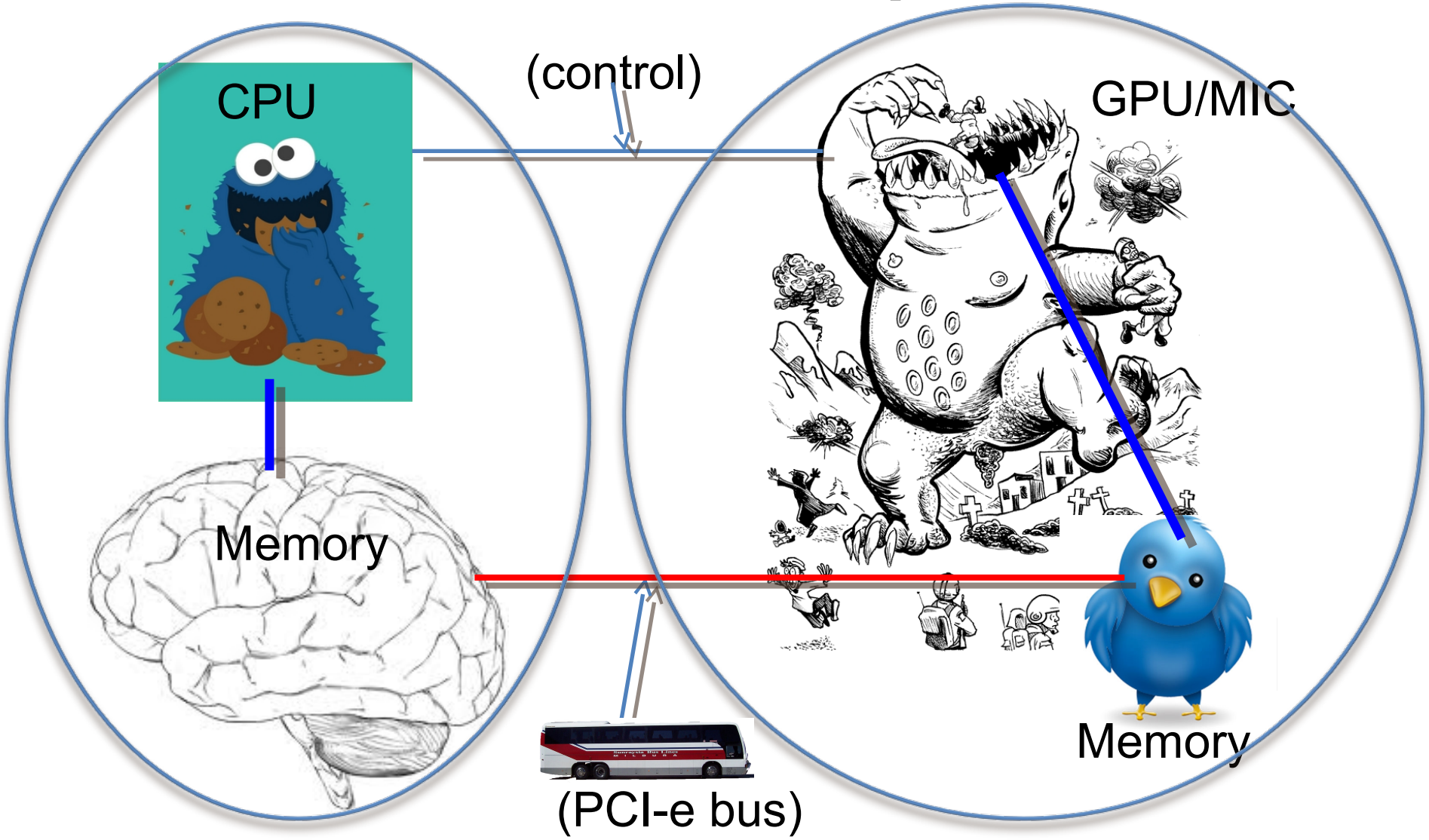
- Hardware "**accelerators**" are "devices" attached to "host" nodes.
  - Include MICs and GPUs.
- **MIC** is "Many Integrated Core"
  - Intel Xeon Phi coprocessors are one kind of MIC processor.
  - MIC cores may be *heterogeneous* (different cores perform different functions), unlike (standard) "multi-core" processors, which are *homogeneous* (all cores the same).
- **GP-GPU** is "General-Purpose Graphical Processing Unit.

Xeon Phi Coprocessor Overview



Host Node

GPU/MIC Device



Ways to use Accelerators

	Xeon Phi	GPU
Offload mode: -Uses directives in source -Many programming constraints -All processes run on hosts, with parallel sections offloaded to accelerator	✔ (possible, but hard)	✔ (possible, but hard)
Native mode: -no source changes required -Cluster of MIC nodes	✔ (easy)	✗
Symmetric mode: -No source changes required -MICs & hosts each a separate node in a cluster	✔ (should be easy, but isn't)	✗

Offload of Main OpenMP loop Fails

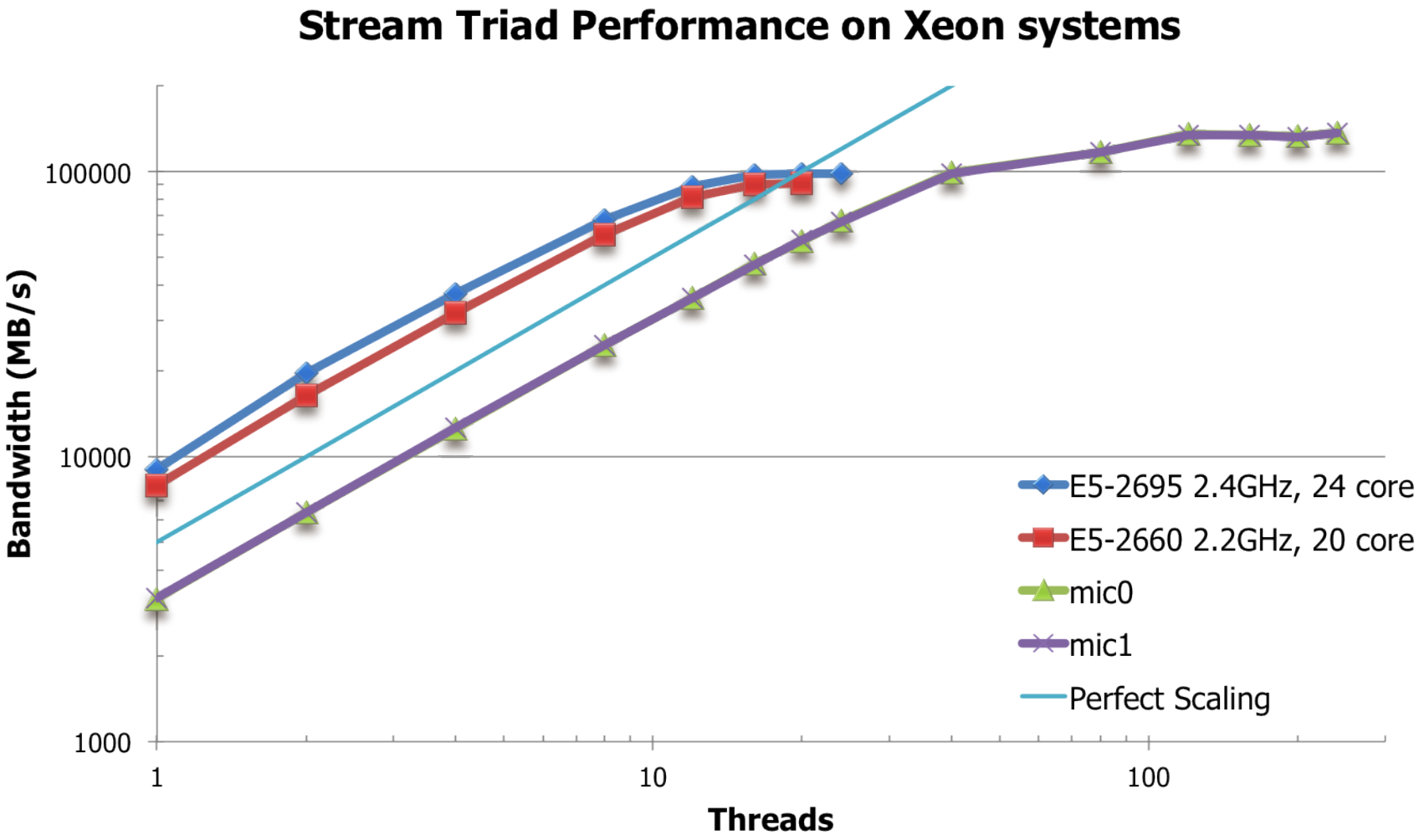
```
cpg.F90(570): error #8545:
A variable used in an OFFLOAD region must not be of
derived type with pointer or allocatable components.
[YDSL]
!dir$ omp offload target(mic)
in(ydsl,CDCONF,LDRETCFOU,LDWRTCFOU0,LDCPG_SPLIT)
```

That is a show-stopper.

Xeon vs. Xeon Phi: Vital Stats

	E5-2660 2.2 GHz	Xeon Phi 5110P
Cores (pre node)	20	61
Threads (per node)	40	240
Clock Freq.	2.2 GHz	1.053 GHz
Memory	64 GB/node	8 GB x 2 cards = 16 GB
Max. Stream Triad	91 GB/s	137 GB/s
Linpack	320 Gflop/s	720 Gflop/s
IMB PingPong latency	< 2 usec	5 - 12 usec
IMB PingPong Bandwidth	> 4 GB/s	0.22 - 4 GB/s

*Phi performance is contingent on using **all** cores or threads!*


Stream Triad Performance on Xeon systems
Bandwidth (MB/s)
100000
10000
1000
1
10
100
Threads
E5-2695 2.4GHz, 24 core
E5-2660 2.2GHz, 20 core
mic0
mic1
Perfect Scaling

IMB Ping-Pong 0-byte Message Latency (usec)

t[usec]	host0	host0-mic0	host0-mic1	host1	host1-mic0	host1-mic1
host0	0.36	5.24	6.40	1.96	6.43	7.05
host0-mic0	5.24	2.28	9.08	6.43	8.96	9.71
host0-mic1	6.40	9.08	2.37	7.05	9.71	10.99

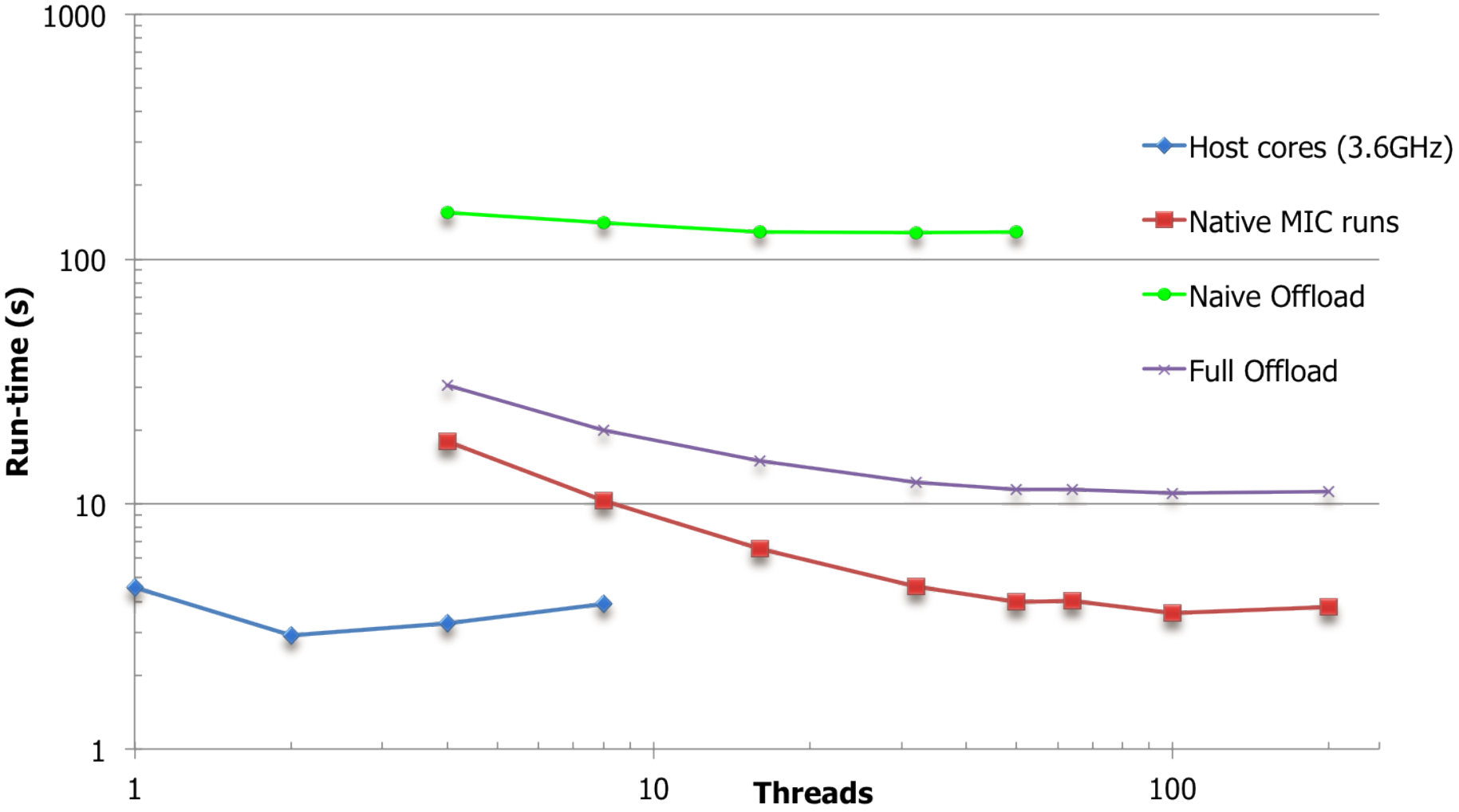
IMB Ping-Pong 4-MB Message Bandwidth (MB/s)

MB/s	host0	host0-mic0	host0-mic1	host1	host1-mic0	host1-mic1
host0	4067	4923	5193	5870	4156	505
host0-mic0	4923	2020	1269	4156	3539	494
host0-mic1	5193	1269	1951	505	494	266

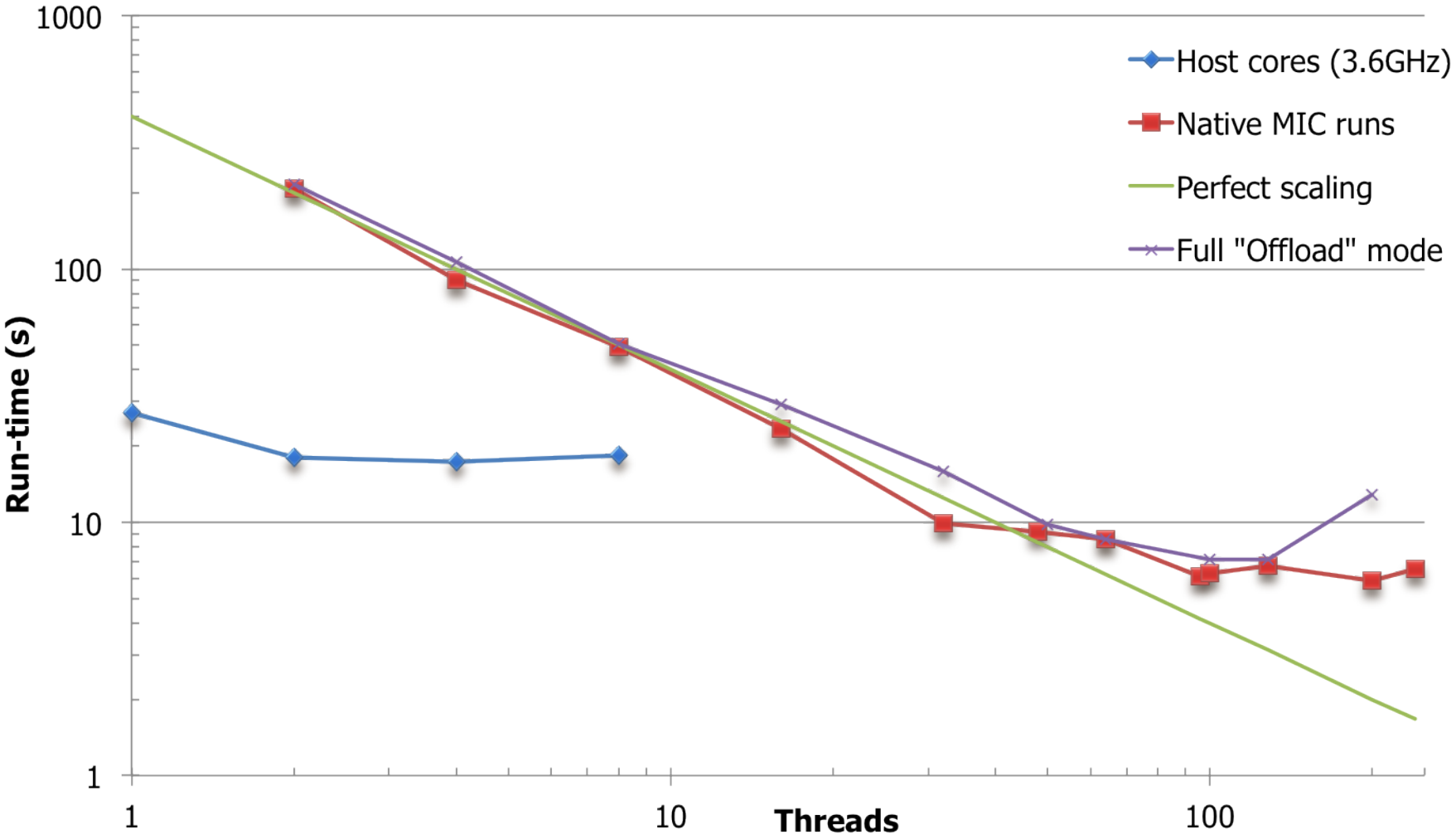
Test Code (Fortran)

```
!$OMP PARALLEL DO PRIVATE(i,j,k)
      do k=2,nz-1
        do j=2,ny-1
          do i=2,nx-1
            arr_out(i,j,k) = wght1*arr_in(i,j,k) + wght2*(
     &          arr_in(i-1,j,k) + arr_in(i+1,j,k) +
     &          arr_in(i,j-1,k) + arr_in(i,j+1,k) +
     &          arr_in(i,j,k-1) + arr_in(i,j,k+1) )
          enddo
        enddo
      enddo
!$OMP END PARALLEL DO
```

"Stencil test" OpenMP Performance on Phi, 1GB case



"Stencil test" OpenMP Performance on Phi, 6GB case



Best case: 3x speedup on Phi

HARMONIE on Xeon Phi

- HARMONIE builds ~cleanly with "-openmp -mmic", runs natively on Phi
  - No source code changes (in principle)
  - Must use Intel compilers, Intel MPI
  - Must re-build zlib, hdf5, netcdf, & grib_api with "-mmic"
    - Really a "cross-compile", but configure files don't recognize MIC archictecture.
    - Configure for host, then edit config.status and Makefiles before running "make".
    - Edit LD_LIBRARY_PATH etc. to pick up "mic" instead of "intel64" libraries.
- 8 Builds completed:
  - HARMONIE cycle37h1.1 and cycle38h1.1;
    - MPI-only and MPI/OpenMP
      - Host and Phi.
- Main executable from "Phi" build copied to "standard" installation
  - for use in "Forecast" phase only.
- Test case, IRELAND55: 300 x 300 x 65-point domain, 5.5 km resolution: *Memory needed: ~20GB minimum* (depends on run-time config.)

Harmonie Run-time Script Changes

Consolidate all run-time changes in **scr/Forecast**

- Need pre-built Harmonie executable: **MASTERODB.MIC**
- Modify hostfile, set Phi-related environment variables, submit exe.:

```
convert ${PBS_NODEFILE} > hfile.mic

export I_MPI_FABRICS=shm:ofa   # (Optional…)
export I_MPI_MIC=enable
export I_MPI_MIC_POSTFIX=.MIC
export MIC_ENV_PREFIX=MIC_
Export KMP_STACKSIZE=200M
Export KMP_MONITOR_STACKSIZE=12MB
Export KMP_AFFINITY=”compact”
Export OMP_NUM_THREADS=240

export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MC_COMPILER_LIB}:$
{MIC_MKL_LIB}:${MIC_NETCDF_LIB}:${MIC_HDF5_LIB}

$MPPEXEC -f hfile.mic $BINDIR/$MODEL -maladin -v$VERSION -e$CNMEXP
-c$NCONF -t$TSTEP -fh$LL -a$ADVEC || exit
```

MPI vs. OpenMP on Host nodes

Host: 20 physical cores; 40 logical cores (with Hyperthreading)

	No HyperThreads			Using HyperThreads		
MPI Processes	*OpenMP Threads*	*Total Threads*	*Forecast Time (s)*	*OpenMP Threads*	*Total Threads*	*Forecast Time (s)*
2	10	20	**1570**	20	40	**940**
5	4	20	**1445**	8	40	**814**
10	2	20	**1384**	4	40	**727**
20	1	20	**769**	2	40	**687**

On Host: Use MPI processes in preference to OpenMP threads

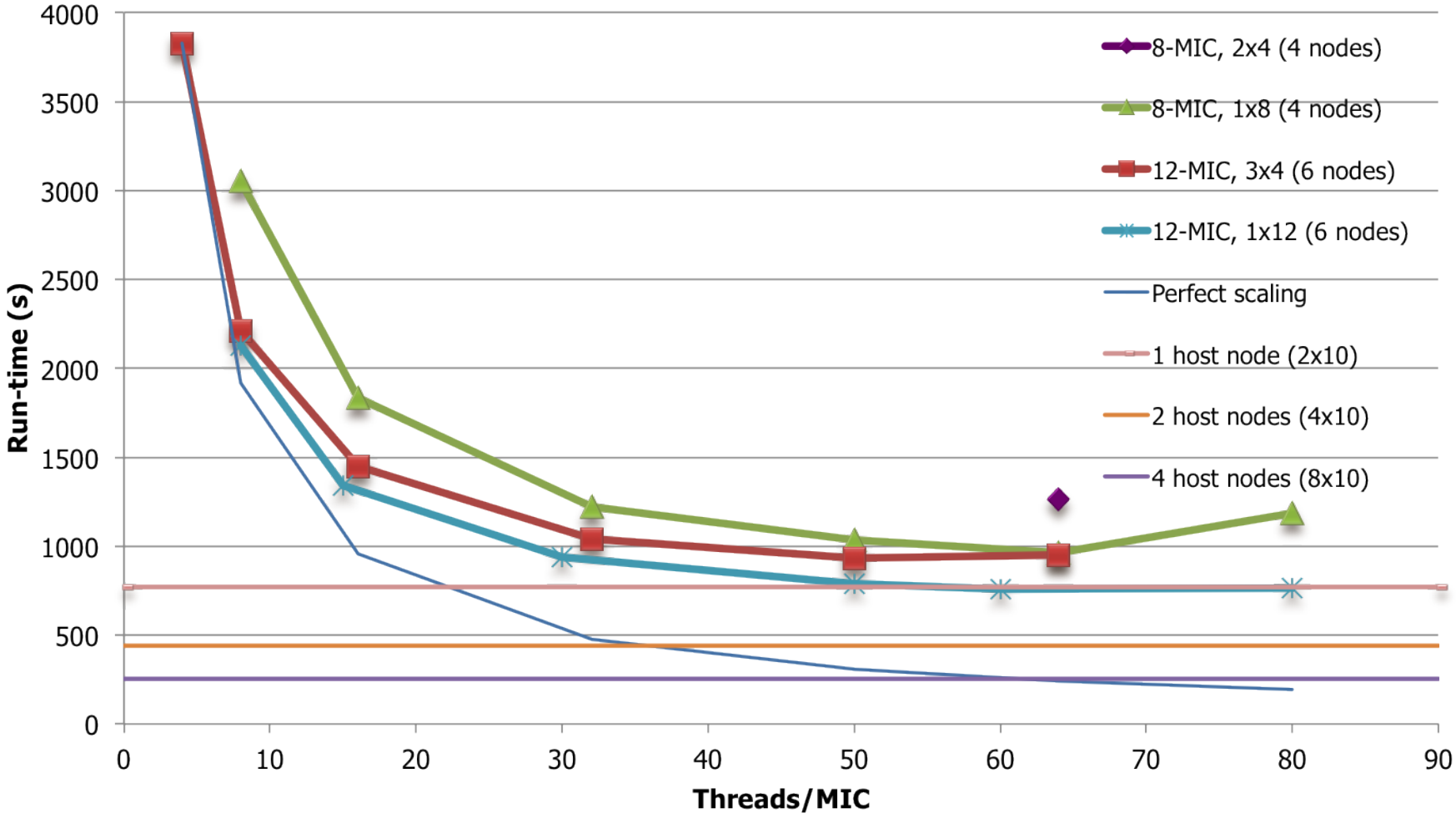
- (after using OpenMP to soak up the "HyperThreads" or "virtual cores")

MPI vs. OpenMP on MIC cards

MPI Only	192 MPI tasks (12 MICs, **16 MPI tasks**/MIC)	3779s
MPI/OpenMP	12 MPI tasks (12 MICs, **16 OMP threads**/task)	1448s
MPI/OpenMP	12 MPI tasks (12 MICs, 50 OMP threads/task	931s

On MICs: Use OpenMP threads in preference to MPI processes

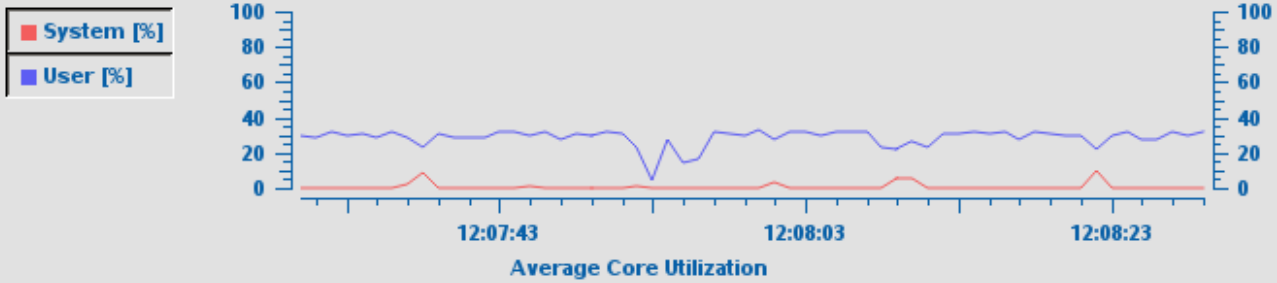
Harmonie Scalability (Ireland 5.5km, 6-hr Forecasts)
Using 8 or 12 MICs, 1 MPI task/MIC



Intel® Xeon Phi™ Coprocessor Platform Status Panel

Cards | Advanced | Cards: 2 | Memory: 16 GB | Cores: 12

Utilization View (All Devices)

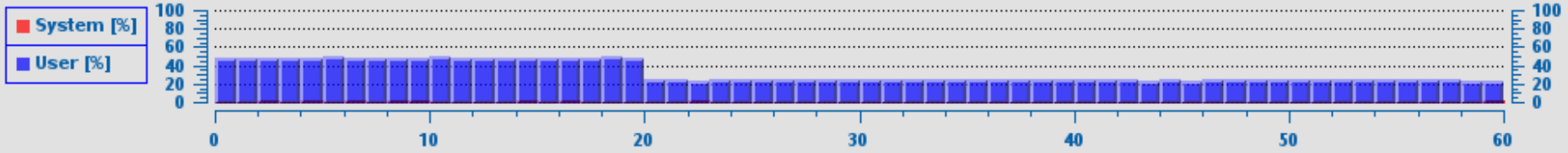


44
Average Core Temperature °C

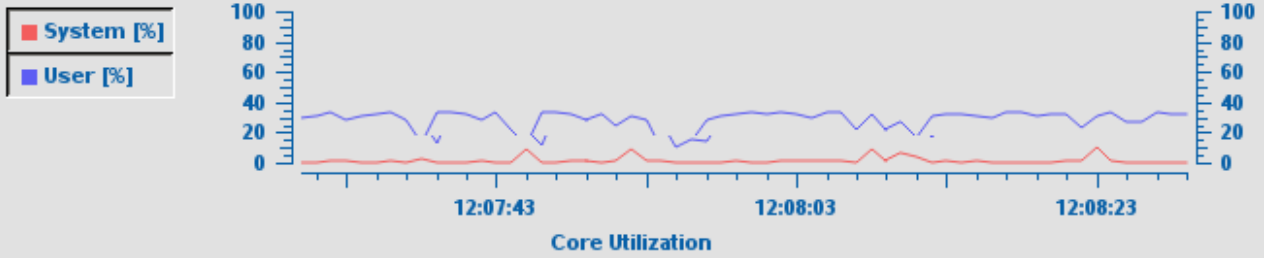
14.534 GB
Total Memory Usage

260 Watts
Total Power Usage

mic0: Core Histogram View



mic1: Utilization View

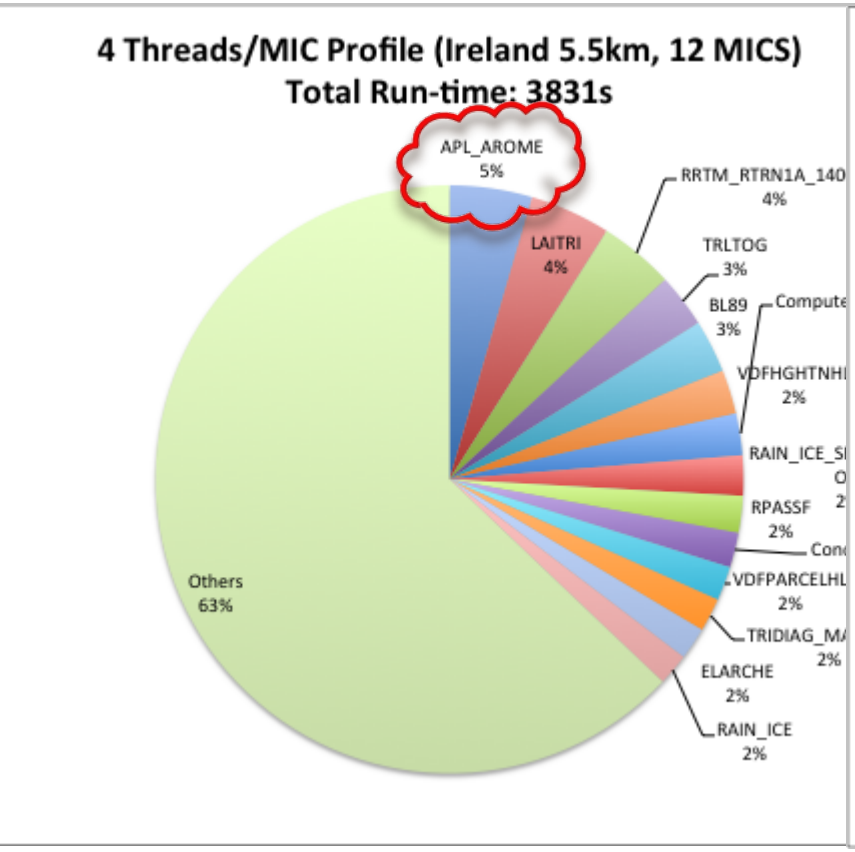


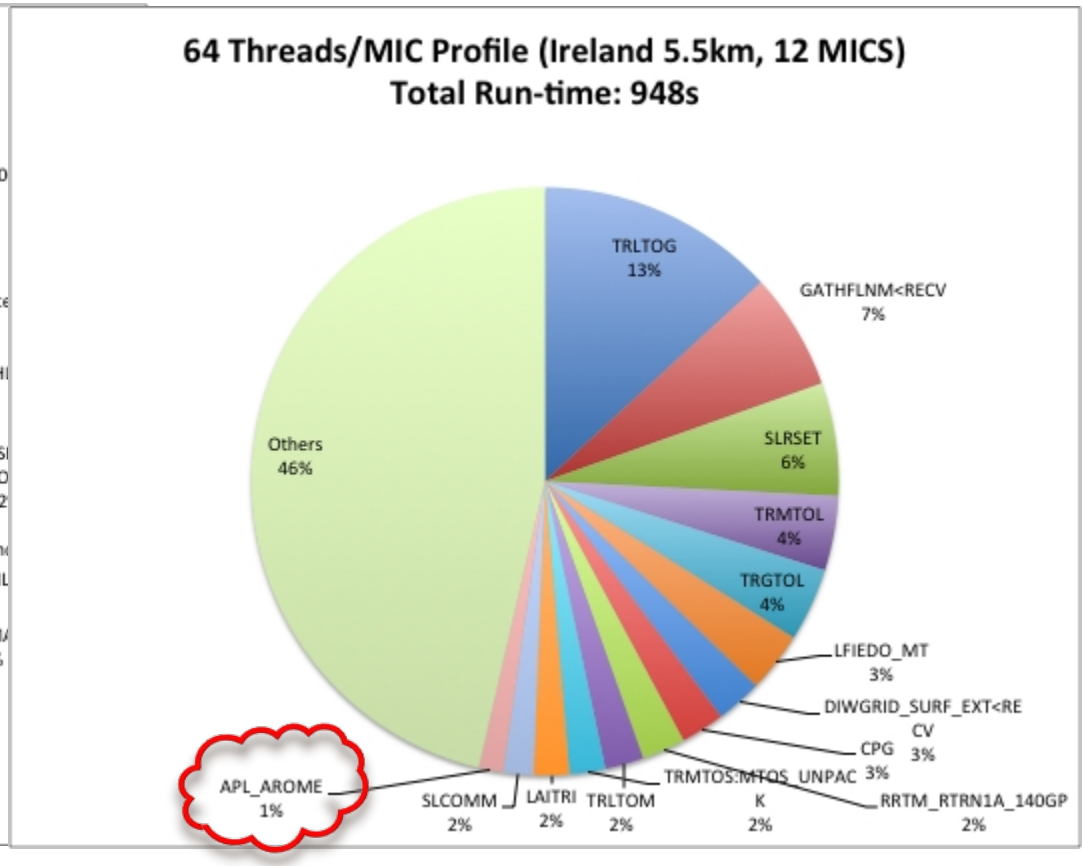
41
Processor Core Temperature °C

7212.2 MB
Memory Usage

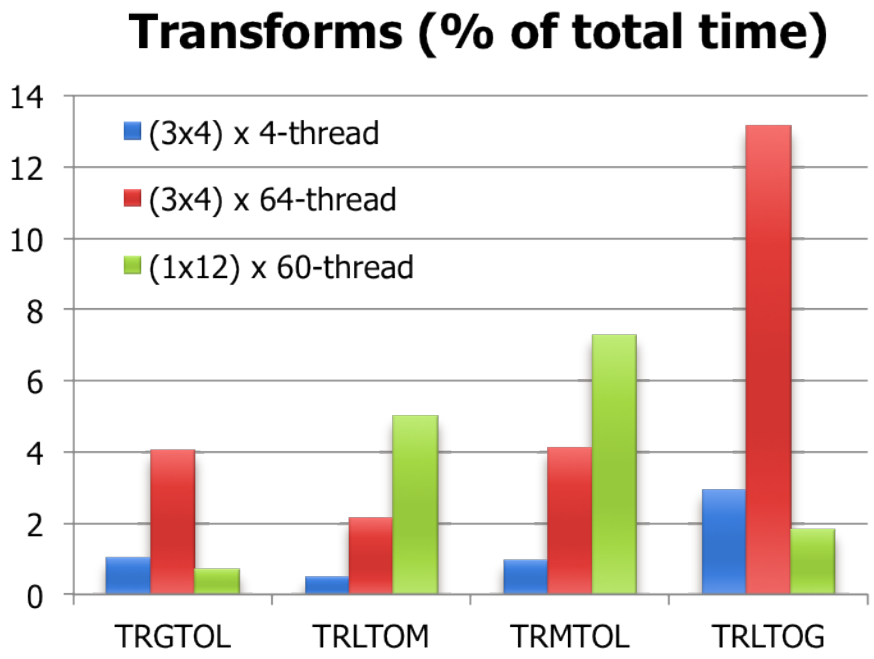
132 Watts
Power Usage

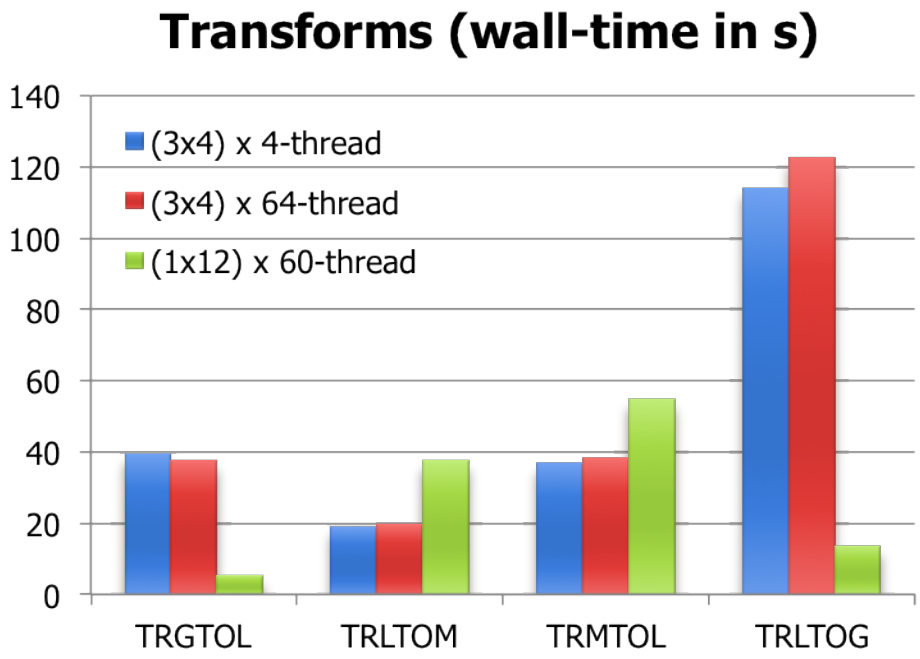
HARMONIE Profiles



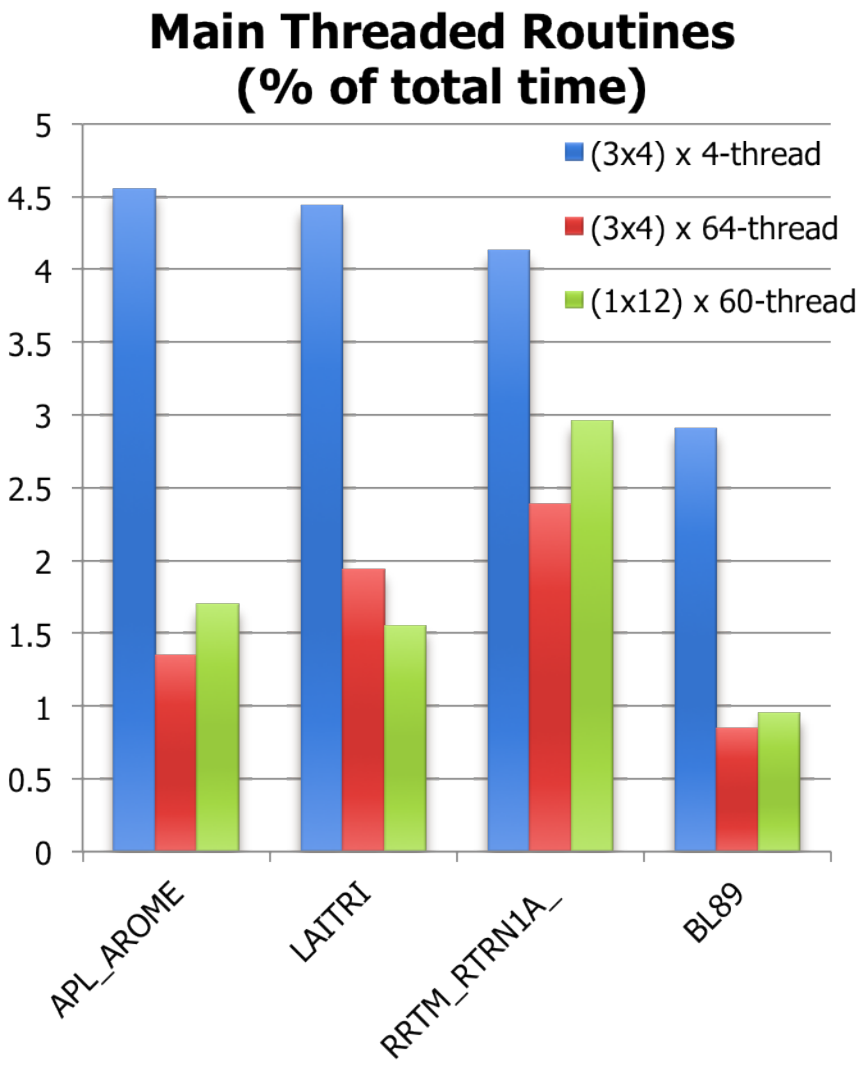


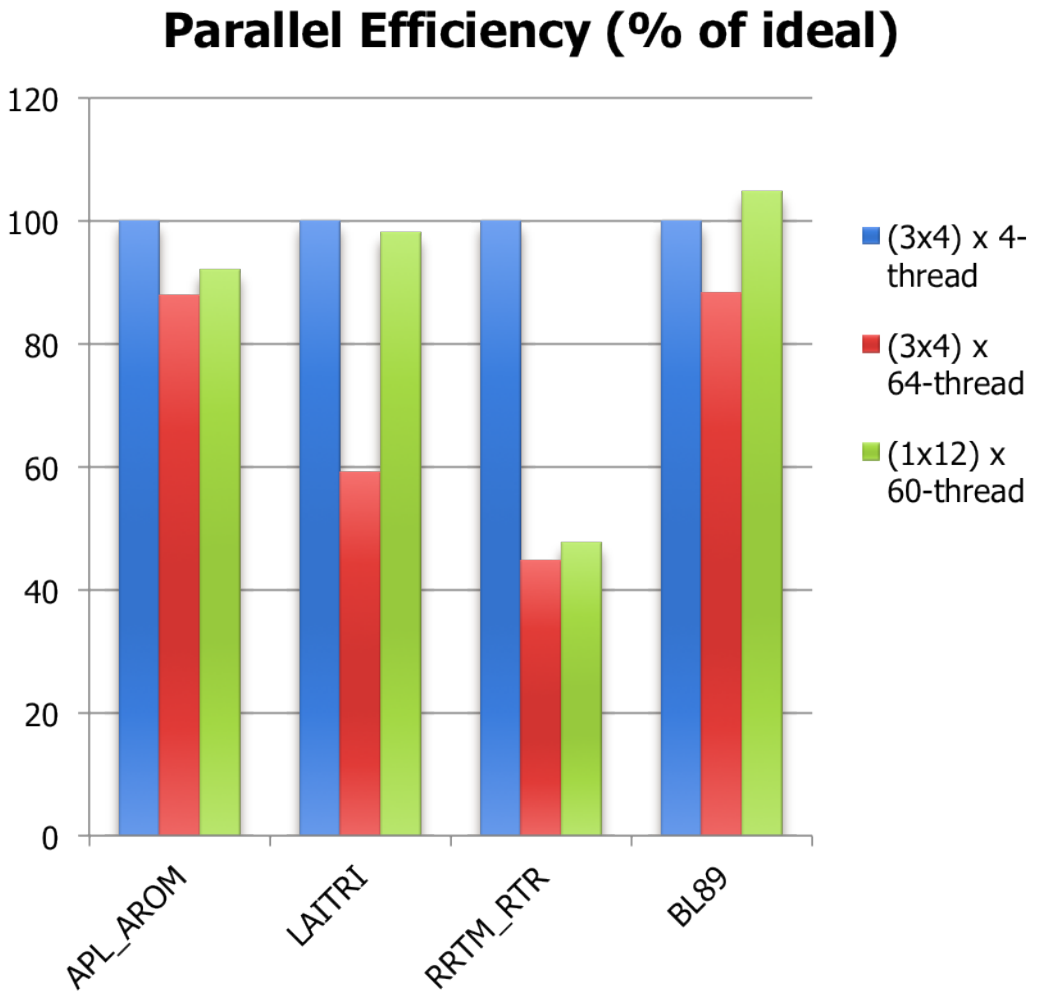
Non-threaded routines dominate at large thread-counts





TRGTOL = TRansform Grid TO Latitude decomposition
TRLTOM = Transform Latitude TO M (zonal) decomposition
TRMTOL = TRansform M (zonal) TO Latitude decomposition
TRLTOG = TRansform Latitude decomposition TO Grid


Main Threaded Routines
(% of total time)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
(3x4) x 4-thread
(3x4) x 64-thread
(1x12) x 60-thread
APL_AROME
LAITRI
RRTM_RTRN1A_
BL89


Parallel Efficiency (% of ideal)
120
100
80
60
40
20
0
APL_AROM
LAITRI
RRTM_RTR
BL89
(3x4) x 4-thread
(3x4) x 64-thread
(1x12) x 60-thread

Issues

- Much performance (cores, threads) left unused because of memory limits.
- Could OMP_NUM_THREADS be increased without increasing memory usage?
  - Reduce number of "private" OMP variables?
  - Use more MPI tasks/MIC, fewer OMP-threads/MPI-task?
  - Find "optimal" KMP_STACKSIZE?
- To run Harmonie efficiently on the Xeon Phi coprocessors, need a problem size big enough to scale to ~100+ threads, yet small enough to fit in < 8GB memory.
  - Next-generation 7000-series MIC processors have 16 GB memory.

- **Symmetric mode** (HARMONIE running on both host and MIC processors simultaneously) currently "hangs" in first MPI collective.
  - ***Still, most promising prospect…***
- **Offload mode** has many issues with pointers in derived data-types, which will require many source-code changes.
  - Is that even worthwhile?