ICHEC
Irish Centre for High-End Computing

Can "Accelerators" Really Accelerate Harmonie?

Enda O'Brien, Adam Ralph
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

Motivation

- There is constant demand for more performance
- Conventional compute cores not getting any more powerful
  - Sequential jobs already at performance limit
- Only way to more performance is with more parallelism.
  - This will impose new algorithmic constraints (e.g., MPI_Alltoall will become impractical)
- New devices offer massive hardware parallelism:
  - General-Purpose Graphical Processing Units (GP-GPUs)
  - Many Integrated Core (MIC), i.e. Intel Xeon Phi co-processor

Exploiting this parallelism is a software challenge

ICHEC
Irish Centre for High-End Computing

Intel Xeon Phi Coprocessor

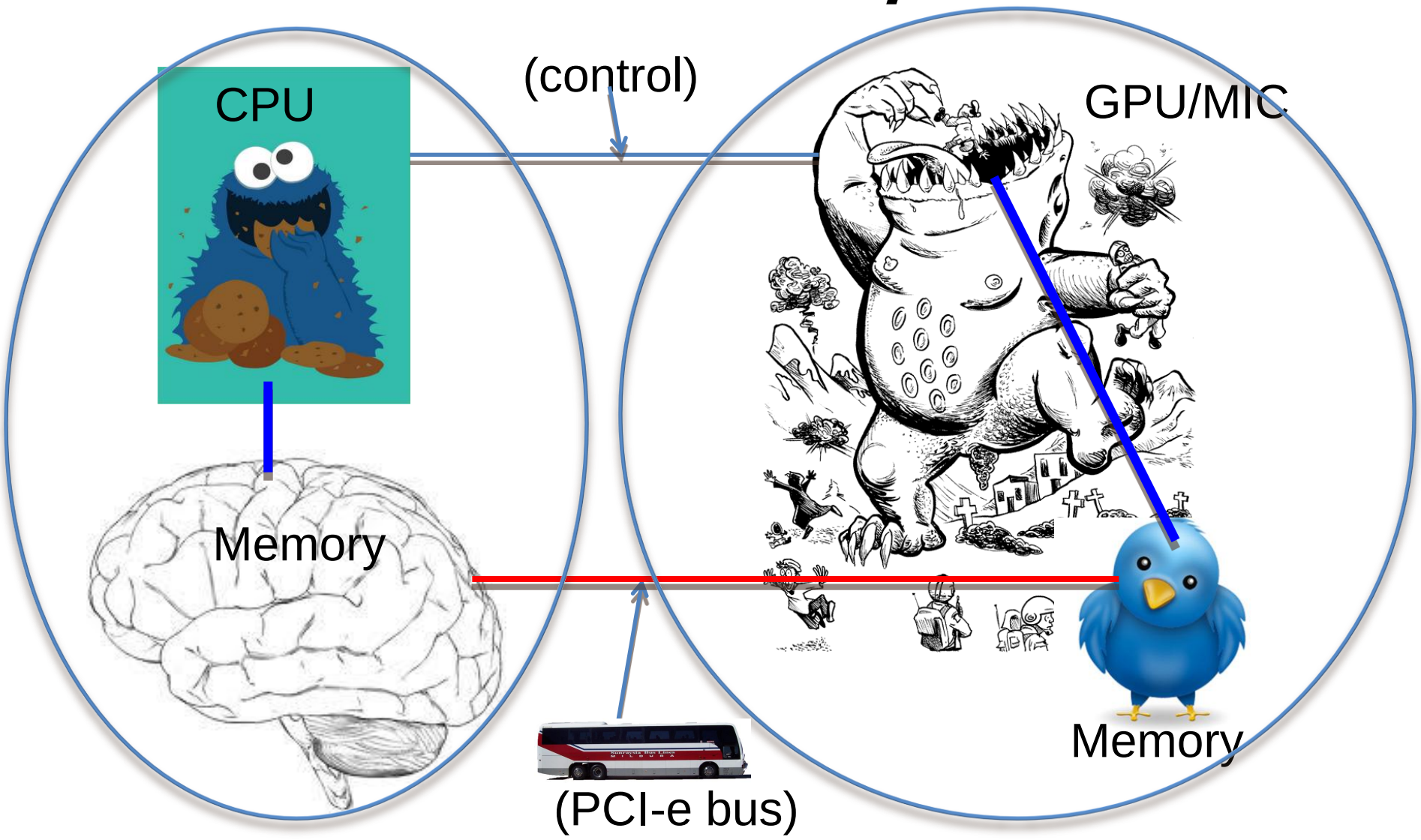
- Host:
- 4-core SandyBridge
- 2 "Hyperthreads"/core
- 8 logical cores
- 32 GB memory

- Device:
- 61 (physical) cores
- 4 threads/core
- 244 logical cores
- 8 GB (total) memory

(PCI-e bus)

Host Node

GPU/MIC Device



Accelerator Programming Principles

- Send massed ranks of data through GPU in lock-step.
- Avoid dependencies, conditionalities
- Programmer should know (and control) what data is on "host" and what is on "accelerator", and how data moves between them.

- Data movement should be minimized.
  - Ideal would be to run *entirely* on accelerator.
- CPU and GPU operations can run *asynchronously*

Weather Models on GPUs

- JMA ASUCA model (Takayuki Aoki)
  - Translated to C, then to CUDA
  - Entire model runs on (~4,000) GPUs
- Dynamical core of NIM (NOAA/ESRL)
  - Uses F2C (Fortran to C) converter
  - Then HMPP directives from CAPS
- COSMO (CSCS/Meteo Swiss)
  - OpenACC directives for physics
  - Re-write in C++ (-> CUDA) for dynamics
- WRF (?)
  - Some parts translated to C/CUDA
  - Directives…?

Options for "Accelerating" Harmonie

- Translating to **CUDA** (or C) - not a practical option.
- **OpenACC** directives could work (similar to OpenMP).
  - www.openacc.org
- Intel **MIC** directives could also work (even more similar to OpenMP).

OpenACC/MIC Restrictions

- Can't transfer pointers (only data that is pointed to)
- Data arrays passed to GPU must be contiguous
  - E.g. for array(L,M,N), these won't work:

```
call sub1(array(1:L,1,1:N),...)
call sub2(array(2:L-1,M,2:N-1),...)
```

Test Code (Fortran)

```
 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
```

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
```

Wrap in a subroutine

```
do icount=0,maxcount


call mainloop(nx,ny,nz,wght1,wght2,arr_in,arr_out)


call mainloop(nx,ny,nz,wght1,wght2,arr_out,arr_in)
  if(icount.lt.10 .or. mod(icount,iofreq).eq.0) then

  write(*,100) (arr_in(2,2,k),k=2,nz-1,nz/7)
  endif
enddo
```

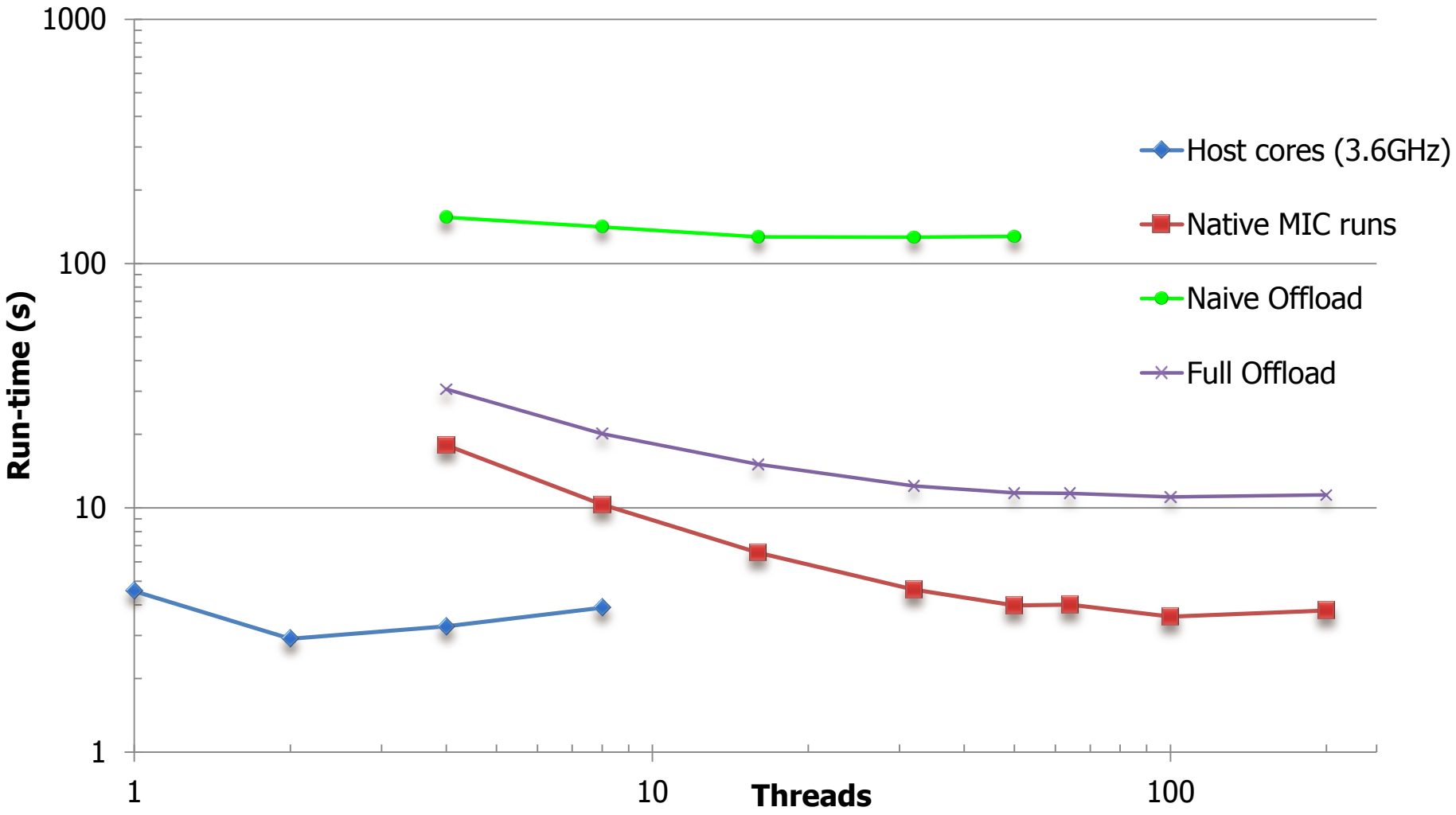
Naively add some directives…

```
!dir$ attributes offload:mic  ::mainloop

      do icount=0,maxcount
!dir$ offload target(mic:0) in(arr_in) out(arr_out)

      call mainloop(nx,ny,nz,wght1,wght2,arr_in,arr_out)
!dir$ offload target(mic:0) in(arr_out) out(arr_in) &

      call mainloop(nx,ny,nz,wght1,wght2,arr_out,arr_in)
        if(icount.lt.10 .or. mod(icount,iofreq).eq.0) then

        write(*,100) (arr_in(2,2,k),k=2,nz-1,nz/7)
        endif
      enddo
```

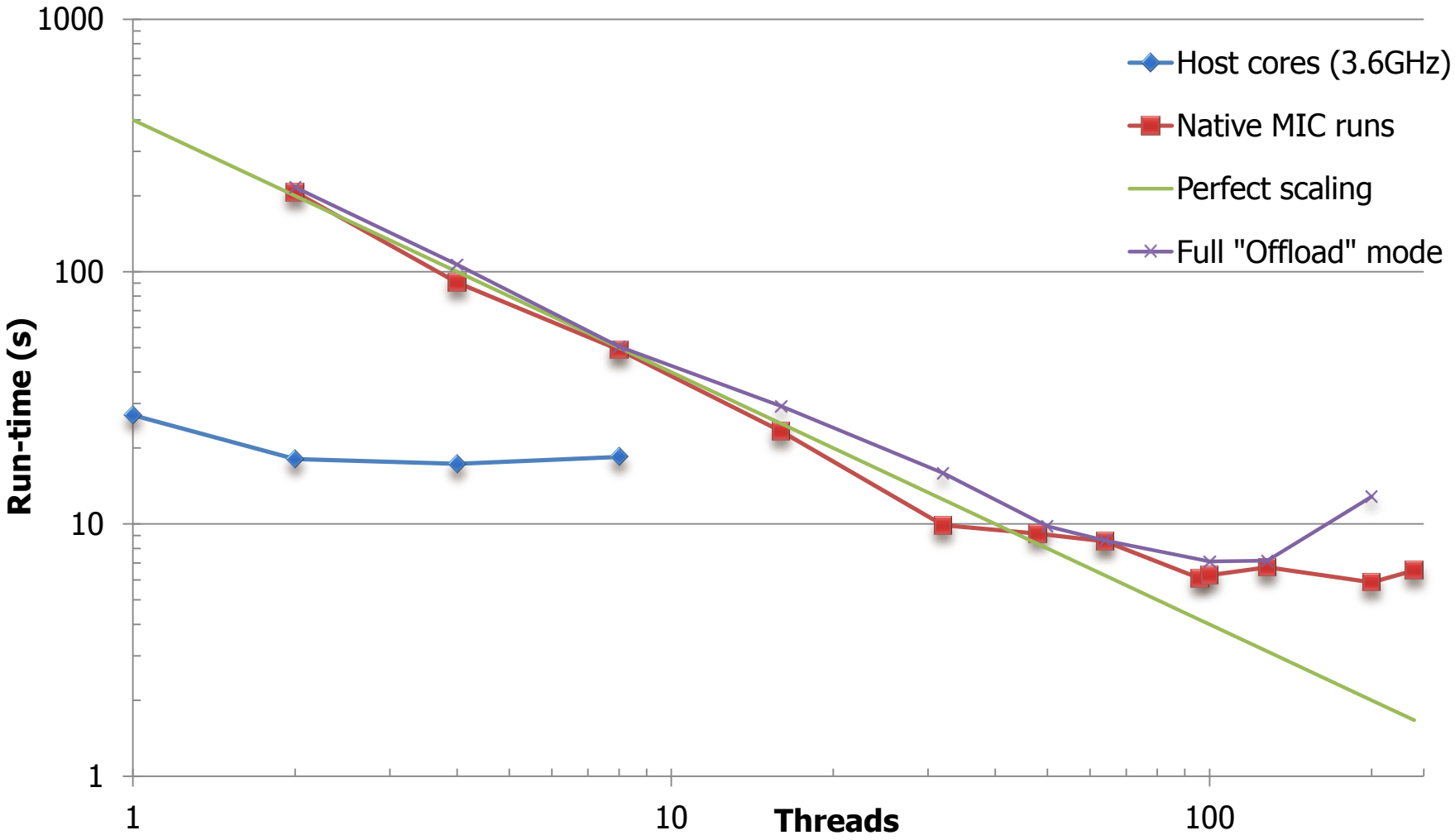
Minimize Data Movement

```
!dir$ attributes offload:mic  ::mainloop
!dir$ offload_transfer target(mic:0) in(arr_in : alloc_if(.true.)  &
       free_if(.false.))
!dir$ offload_transfer target(mic:0) nocopy(arr_out : alloc_if(.true.) &
       free_if(.false.))
!dir$ offload_transfer target(mic:0) in(nx,ny,nz,wght1,wght2 )

      do icount=0,maxcount
!dir$ offload target(mic:0) nocopy(arr_in , arr_out)  &
         nocopy(nx,ny,nz,wght1,wght2)
      call mainloop(nx,ny,nz,wght1,wght2,arr_in,arr_out)
!dir$ offload target(mic:0) nocopy(arr_out, arr_in) &
         nocopy(nx,ny,nz,wght1,wght2)
      call mainloop(nx,ny,nz,wght1,wght2,arr_out,arr_in)
        if(icount.lt.10 .or. mod(icount,iofreq).eq.0) then
!dir$   offload_transfer target(mic:0) out(arr_in : free_if(.false.) )
        write(*,100) (arr_in(2,2,k),k=2,nz-1,nz/7)
        endif
      enddo
```

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

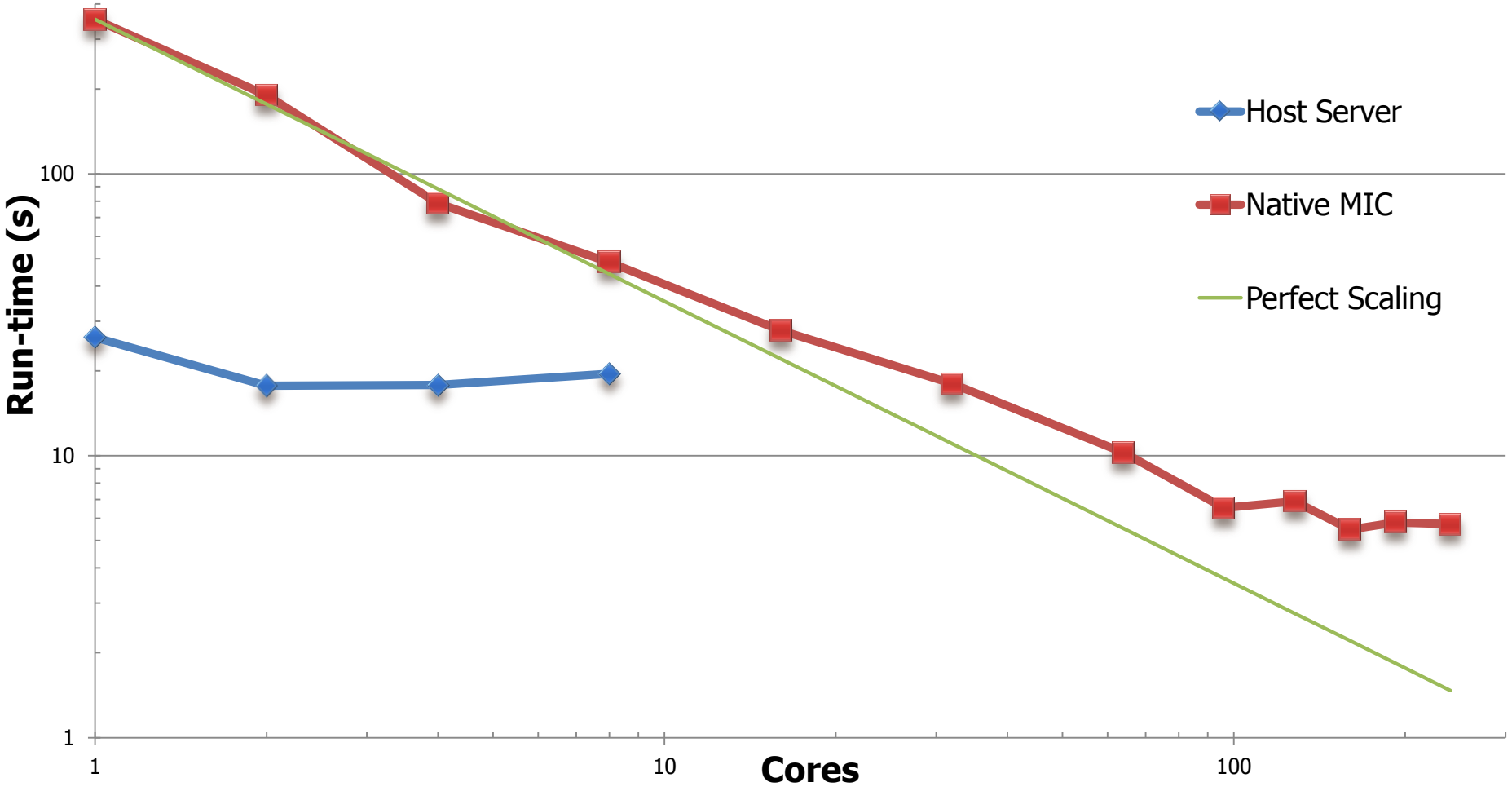


"Stencil test" OpenMP Performance on Phi, 6GB case



Best case: 3x speedup on Phi

Stencil Test, MPI Performance, 6GB Problem Size



Best case: 3x speedup on Phi

Harmonie on Phi

- Use cycle-38 "benchmark" kit
- Only the "small" case (50 x 50 x 65 points) fits in memory on either host or phi coprocessor
- Try "native" build of OpenMP/MPI hybrid:
  - Must use Intel MPI
  - Just compile with "-openmp –mmic"
  - No source changes (in principle)
  - Must re-build zlib, hdf5, netcdf, & grib_api with "-mmic"
    - Not trivial: really a "cross-compile", but configure files don't recognize the MIC archictecture.
    - Configure for host, then edit config.status and Makefiles before running "make".
    - Edit LD_LIBRARY_PATH and other environment vars. to pick up "mic" instead of "intel64" libraries.

Harmonie on Phi

- Harmonie builds & runs on host as usual
- At run-time on Phi, build using "-mmic" reports:

```
forrtl: severe (154): array index out of bounds
```

- Re-build with "-check bounds", to reveal:

```
forrtl: severe (408): fort: (3): Subscript #2 of array PGMVS has
value -99999999 which is less than the lower bound of 1
```

- Track this down (to arp/setup/sudim1.F90). No-brainer fix:

```
! Set default for index to be used for unused/undefined fields
NUNDEFLD=-99999999
!NUNDEFLD=1
```

- Next:

```
forrtl: severe (408): fort: (2): Subscript #2 of array ZGPSDT2D has
value 1 which is greater than the upper bound of 0
```

- Fix in arp/setup/suspsdt.F90; make YGPSDT%GP2D=1 (not 0):

```
!orig  ALLOCATE(IGRIB(0))
  ALLOCATE(IGRIB(1))
!orig   CALL ALLOCATE_GRID( YGPSDT, 0, 0, igrib )
  CALL ALLOCATE_GRID( YGPSDT, 0, 1, igrib )
  DEALLOCATE(IGRIB)
```

Harmonie now runs "natively" on Phi

…but very slowly (and unreliably):

Harmonie run-times (s) on host and (natively) on Phi				
	Threads			
MPI-Tasks (NX x NY)	1	2	4	8
1	342 (18034)	186 (9400)	113 (5155)	98 (3210)
1x2	220	128	104	
2x2	147	103.5 (3429)	(1916)	
2x4	102			
3x4	(2235)			

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. Other messages are more "useful" (& fixable):

```
cpg.F90(622): error #8537: A global variable used in an OFFLOAD region
must have the OFFLOAD:TARGET attribute.   [GFLTNDSL_DDH]
       & GMVTNDSL_DDH(1,1,1,IBL),GFLTNDSL_DDH(1,1,1,IBL), &
--------------------------------^
```

```
cpg.F90(893): error #8627: The dummy argument used in an OFFLOAD region
has the INTENT(IN) attribute; therefore it must be explicitly specified in
an IN clause.   [YDSL]
            PB1(YDSL%NSLCORE(JROF),JFLD)=ZSLBUF1AU(JROF-ISTGLO+1,JFLD,ILL)
---------------^
```

OpenACC Compiler Support

- CAPS/HMPP
- PGI
- Cray

Quote from Intel (June 2012): "OpenACC is a partial interim standard to cover only specific types of GPU. Intel is working on the committee to merge those facilities into future OpenMP"

(Intel has its own separate set of !DIR$ directives to support the Xeon Phi coprocessor).

Currently no OpenACC support from open-source compilers.

Harmonie/PGI issues 1: Un-allocated Arrays

In yoe_cuconvca.F90:

```
 REAL(KIND=JPRB)    ,ALLOCATABLE:: RCUCONVCA(:)
...

 IF (.NOT. LCUCONV_CA) THEN
    WRITE(NULOUT,*) 'convective CA not active!'
 ELSE
    ALLOCATE(RCUCONVCA(NGPTOT))
    RCUCONVCA=0.0
ENDIF
```

Then in cpg.F90, these arrays are passed to MF_PHYS, allocated or not:

```
USE YOE_CUCONVCA, ONLY : RCUCONVCA
CALL MF_PHYS &
    & (CDCONF,IBL,IGPCOMP,IST,IEND,IGL1,IGL2,IGL3,IGL4,...
    & ...,RCORI(IOFF),RCUCONVCA(IOFF), ...)
```

Allocation is conditional, but transfer is unconditional...

Harmonie/PGI issues 2: Using "SIZE" in declaration section

In coupling_surf_atmn.F90:

```
REAL, DIMENSION(KI), INTENT(IN) :: PTA   ! air temp. forcing
...
REAL, DIMENSION(SIZE(PTA))      :: ZPEW_A_COEF  ! implicit coeffs.
```

Instead, must declare ZPEW_A_COEF as:

```
REAL, DIMENSION(KI)             :: ZPEW_A_COEF  ! Implicit coeffs.
```

OpenACC in Laitri.F90

Just 2 extra lines to offload a parallel region to GPU:

```
! Scalar code
IF (LOPT_SCALAR) THEN
! 32-point interpolations
!$acc region
  DO JLEV=1,KFLEV
    ! interpolations in longitude
    DO JROF=KST,KPROF
      ! interpolations in longitude, stencil level 0
Z10(JROF)=PXSL(KL0(JROF,JLEV,1)+IV0L1)+PDLO(JROF,JLEV,1) &
 ...
    ENDDO
  ENDDO
!$acc end region
```

OpenACC in rrtm_rtrn1a_140GP

Some more explicit control over data movement:

```
!$acc data region local(iclddn_,z_surfemis,...,z_urad1__)
...
!$acc data region local(z_cldradu) copyout (p_totdfx,p_totdfc)
!$acc region
  DO JLON = KIDIA, KFDIA
    !-start JJM_000511
  ...
  ENDDO
!$acc end region
!$acc end data region
...
!$acc end data region
```

Modest but Real "Acceleration" ...

- Approx. 3x speedup of laitri and rrtm_rtrn1a_140GP, from DR_HOOK:

#	% Time (self)	Cumulated (sec)	Self (sec)	Self (1-CPU, CPU+GPU)	Total	# calls	Routine
1	15.24	108.87	108.87	(33.5,**10.1**)	108.89	59475	LAITRI
2	5.17	145.80	36.94		371.81	3965	APL_AROME
3	4.73	179.59	33.78		33.80	63440	TRIDIAG_MASSFLUX
4	3.80	206.77	27.18	(12.2,**3.7**)	27.18	325	RRTM_RTRN1A_140GP

(Times include data-transfer times, and are for a small problem size)

Current Assessment

- Both Intel MIC and OpenACC directives could potentially accelerate Harmonie.
- With OpenACC in Harmonie, most problems are with "standard" compiler (CAPS or PGI)
  - OpenACC directives themselves are relatively straightforward, and seem to work as advertised.
  - Even Intel compiler has trouble with Harmonie on MIC
- Compilers/directives will evolve (quickly)
  - May all converge in an "OpenMP" standard.
- "Human" contributions will always be needed.

Is the (potential) gain worth the (certain) pain?

If

- Acceleration of Harmonie by GPU/MIC is deemed worthwhile

Then

- A formal initiative is required to do it
- Source-code changes will be required
  (beyond insertion of directives)
- Will require broad acceptance

Else

- Carry on informally, or just leave it for now

Endif

Future Compatibility?