ICHEC
Irish Centre for High-End Computing

# Accelerating Harmonie with GPUs (or MICs)

(A view from the starting-point)

Enda O'Brien, Adam Ralph
Irish Centre for High-End Computing

sfi 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 RÍOGHACHT OIDEACHAIS AGUS EOLAÍOCHTA | DEPARTMENT OF EDUCATION AND SCIENCE

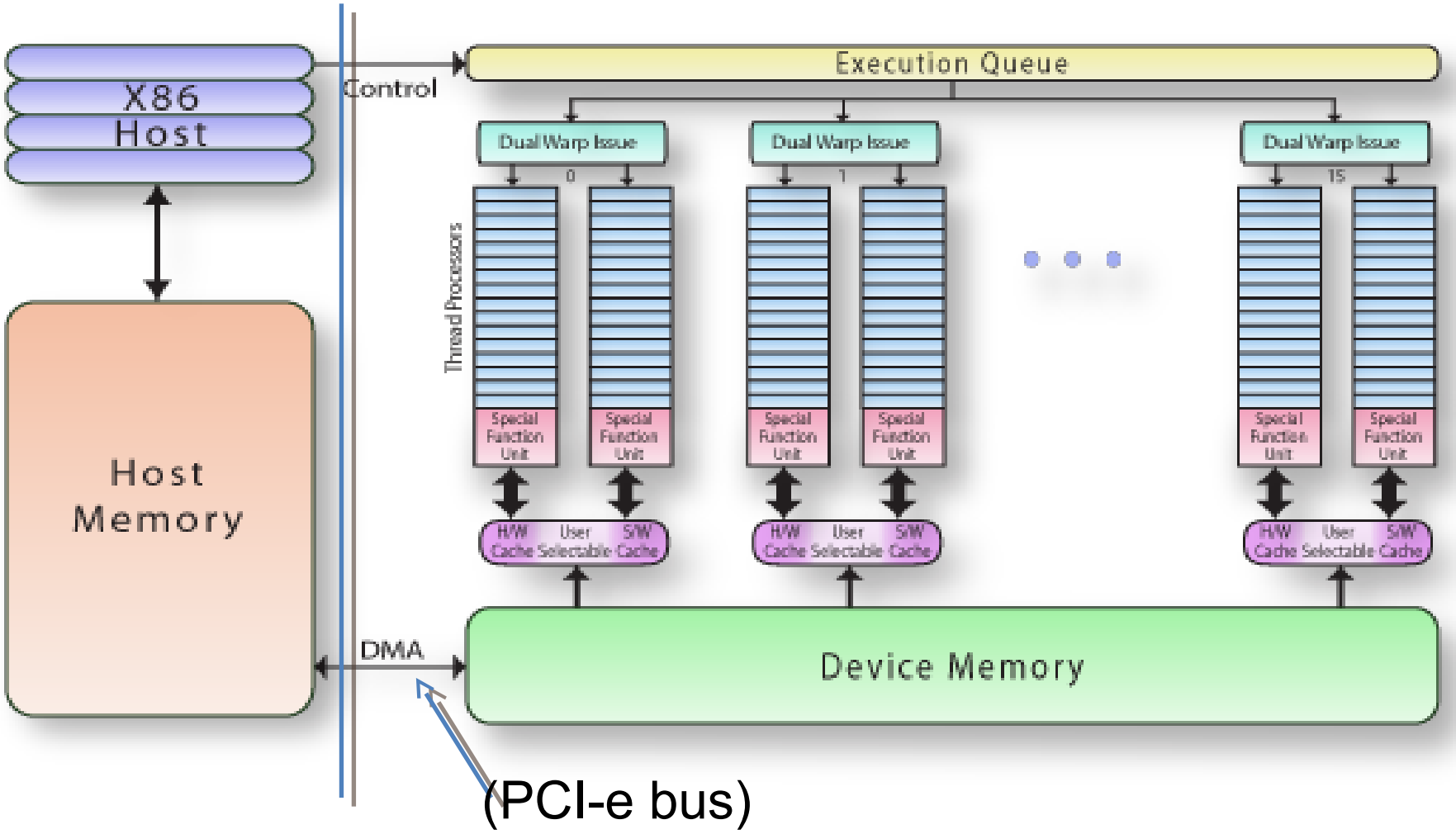
HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

# Motivation

- There is constant, insatiable 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)
- General-Purpose Graphical Processing Units (GP-GPUs) offer massive (hardware) parallelism
  - Many Integrated Core (MIC) accelerators are a new alternative (e.g. Intel Xeon Phi co-processor)

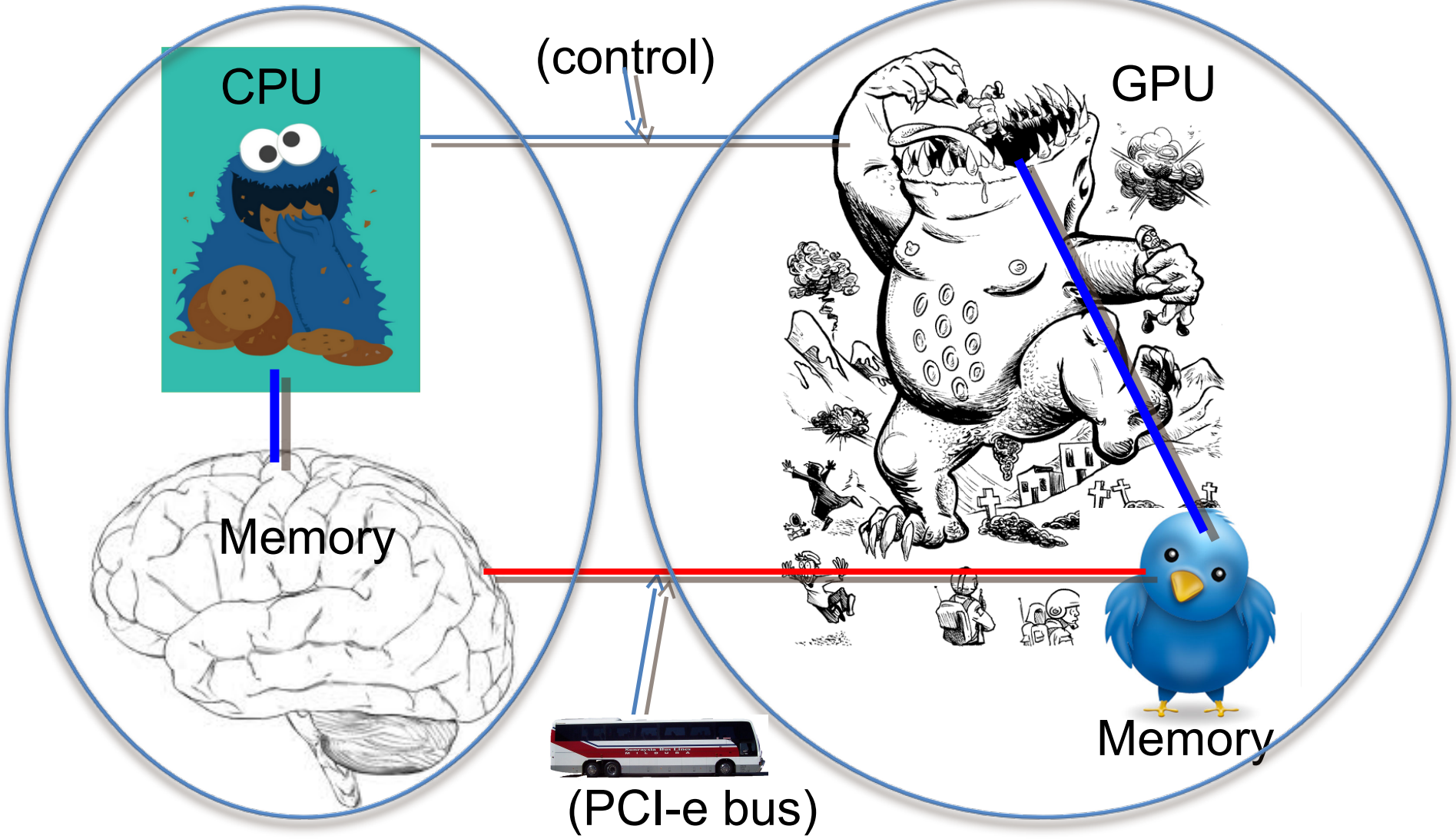
*Exploiting this parallelism is a software challenge*

# Host-GPU Schematic (Fermi)



# Host Node

# GPU Device



ICHEC
Irish Centre for High-End Computing

# 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 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.

# OpenACC Limitations

- 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),...)
```

# PGI Idiosyncrasies…

In original obshor.F90:

```
PGF90-S-0038-Symbol, ngomgfl, has not been explicitly
declared (obshor.F90)
```

"Solution" is to replace:

```
USE GOMS_MIX
```

With:

```
USE GOMS_MIX, ONLY : NGOMGFL, YGOMUA5, YGOMS5, YGOMUA5_2D, &

 & YGOMS5_2D, YGOMUA, YGOMS, YGOMUA_2D, YGOMS_2D
```

Similarly, in Bator.F90:

```
 USE BATOR_MODULE
```

Doesn't work (though in theory, it should); instead use:

```
 USE BATOR_MODULE,    ONLY : TREF_FICOBS, CLSID, ZENTSUP
```

# Arrays Declared with Zero Size

- Compile with "mpif90 –Mchkptr …" (i.e., pgf90)
- Get warnings (*not* errors):

```
PGF90-W-0435-Array declared with zero size
   (acvppkf.F90: 145)
```

- Offending source, local variables:

```
REAL(KIND=JPRB) :: ZCH1   (KLON,KLEV,0)
REAL(KIND=JPRB) :: ZCH1TEN(KLON,KLEV,0)
```

- Okay perhaps, until:

```
CALL  CONVECTION_SHAL( KLON, KLEV, …, ZCH1,ZCH1TEN)
```

# More (explicit) Zero-size arrays

- `PGF90-W-0435-Array declared with zero size (Mandalay.F90: 364)`

```
REAL(KIND=JPRB) :: zinfo(0)
```

Then:

```
call getdb('MANDALAY',0,iret,info,0,zinfo,0,...)
...
call putdb('MANDALAY',0,iret,info,0,zinfo,0)
```

(at least, this didn't cause any trouble)

- `PGF90-W-0435-Array declared with zero size (prep.F90: 70)`

```
REAL, DIMENSION(0) :: ZZS
```

(Harmless, since ZZS not referenced further).

# Allocatable Arrays, set to Zero Size

- In mpa/turb/interface/aro_turb_mnh.h:

```
REAL(KIND=JPRB), DIMENSION(0,0,0), INTENT(IN) :: PEPSM
REAL(KIND=JPRB), DIMENSION(0,0,0) , INTENT(INOUT) ::PREPSS
```

- Those arrays not used anywhere in aro_turb_mnh
- aro_turb_mnh called from apl_arome, where:

```
ALLOCATE(ZEPSM (0,0,0))
ALLOCATE(ZEPSS (0,0,0))
```

- Relatively easy to live without these.
- Not the real problem (at least these are “allocated” ).

# Changes to CPG, MF_PHYS

- 12 x SP_* arrays declared & allocated in SURFACE_FIELDS_MIX with zero in at least one dimension:
  - `ALLOCATE(SP_SG(NPROMA,YSP_SGD%NDIM,NGPBLKS))`
  - The SP_* arrays are then "USE"d in CPG;
  - Passed in argument list to MF_PHYS, becoming lower-dimensional PSP_*
  - Used locally in MF_PHYS (and passed on to other subroutines)
- Avoid the zero-dimension declaration (and run-time failure), e.g.,
  - "USE" SP_* in MF_PHYS;
  - declare PSP_* as local vars in MF_PHYS, and ensure they have non-zero size

```
!     --- Moved from argument list
REAL(KIND=JPRB) :: PSP_SG(NPROMA,MAX(YSP_SGD%NDIM,1))
...
 IF (SIZE(SP_SG) .GT. 0) PSP_SG = SP_SG(:,:,KBL)
```

# Un-allocated arrays, only showing at run-time

In yoe_cuconvca.F90:

```
 REAL(KIND=JPRB)    ,ALLOCATABLE:: RCUCONVCA(:)
 REAL(KIND=JPRB)    ,ALLOCATABLE:: RNLCONVCA(:)
...
 IF (.NOT. LCUCONV_CA) THEN
    WRITE(NULOUT,*) 'convective CA not active!'
 ELSE
    ALLOCATE(RCUCONVCA(NGPTOT))
    RCUCONVCA=0.0
    ALLOCATE(RNLCONVCA(NGPTOT))
    RNLCONVCA=0.0 !NLIVES
 ENDIF
```

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

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

***Allocation is conditional, but transfer is unconditional...***

# (Un-allocated arrays, contd.)

In mf_phys.F90:

```
REAL(KIND=JPRB)    ,INTENT(INOUT)  :: PCUCONVCA(NPROMA)
REAL(KIND=JPRB)    ,INTENT(INOUT)  :: PNLCONVCA(NPROMA)
```

These arrays are passed further in call to APLPAR, and from there to ACCVUD:

```
REAL(KIND=JPRB)    ,INTENT(INOUT)  :: PCUCONVCA(KLON)
REAL(KIND=JPRB)    ,INTENT(INOUT)  :: PNLCONVCA(KLON)
```

Depending on `LCUCONV_CA`, these may never be referenced – but *declaring finite memory for unallocated arrays causes trouble for PGI with HARMONIE!*

---

- Trying to isolate the key bugs in simple "reproducer" programs, we found
  - Using zero-length arrays worked according to the standard, but
  - Using unassociated pointers caused segmentation faults.
  - See post #981 by Adam Ralph on "HARMONIE SYSTEM" in HIRLAM forum.

# OpenACC in acraneb.F90

```
669:  !$acc kernels
670:     IF ((LRPROX.OR.LRMIX).AND.(.NOT.(LRAUTOEV.OR.LRTDL))) THEN
671:        DO JLEV=KTDIA,KLEV
672:           DO JLON=KIDIA,KFDIA
673:              ZIRHOV=(PR(JLON,JLEV)*PT(JLON,JLEV))/PAPRSF(JLON,JLEV)
674:              ZQ=MAX(ZEPS2,PQ(JLON,JLEV))


809:     DO JN=1,IAUCR
810:  !$acc loop independent
811:         DO JLON=IIDIA(JN),IFDIA(JN)
812:            ZVOIGT=ZRHOZ0V(1)*ZVSH(JLON)/ZRSH(JLON)
813:            ZBZV=ZG4B(1)*ZNSH(JLON)/(ZRSH(JLON)/ZNSH(JLON))

4846:          PFRSO(JLON,JLEV)=PFRSO(JLON,KTDIA-1)
4847:          PFRTH(JLON,JLEV)=PFRTH(JLON,KTDIA-1)
4848:      ENDDO
4849:   ENDDO
4850:!$acc end kernels
```

# 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
```

# Using OpenACC with PGI

- Compile with:

```
FCFLAGS=… -Minfo=all -Mbackslash -acclibs
    -ta=nvidia,cuda4.1,fastmath,time
```

- "-Minfo" useful to show the "copyin" and "copyout" data.
- "-Mbackslash" needed by the write_cover_tex* files, e.g.:

```
WRITE(NTEX,*) '\medskip\'
```

- Link with:

```
LD_LANG02 = -L/opt/pgi/12.8/linux86-64/12.8/lib -lacc1mp
```

# 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 have potential to significantly accelerate Harmonie.
- With OpenACC, most problems are with "standard" compiler (CAPS or PGI)
  - OpenACC directives themselves are relatively straightforward, and seem to work as advertised.
- Compilers/directives will evolve (quickly)
  - May all converge in an "OpenMP" standard.
- "Human" contributions will always be needed.

# Future Compatibility?