Preparation of IFS physics for future architectures

Sami Saarinen (CSC – IT Center for Science Ltd, Finland)
Deborah Salmond & Richard Forbes (ECMWF)
Oct 27-31, 2014 for ECMWF HPC workshop

Outline

- Adaptation of IFS physics cloud scheme (CLOUDSC) to new architectures as part of ECMWF scalability programme
- Emphasis on GPU-migration by use of OpenACC directives
- Comparisons against Intel XeonPhi (MIC) and Intel Xeon/Haswell

CLOUDSC problem setup

- **Given single MPI-task's worth of grid point columns (NGPTOT ~ 40,000) @ T2047 L 137 (~10km) Forecast (~ 10% of total wall clock time)**
  - Divided into column blocks (max block size = NPROMA)
  - Each grid point column is independent of each other
    - Only vertical dependency counts
  - Sweep over column blocks, each calling CLOUDSC
    - Lots of natural multi- & manycore-parallelism with OpenMP
- **Aiming at a single source code for CLOUDSC on CPUs/MICs (OpenMP) and on GPUs (OpenACC)**
  - Performance check against original/old CLOUDSC

Driver code for CLOUDSC with OpenMP

```
!$OMP PARALLEL PRIVATE(JKGLO,IBL,ICEND)
!$OMP DO SCHEDULE(DYNAMIC,1)

    DO JKGLO=1,NGPTOT,NPROMA   ! So called NPROMA-loop
       IBL=(JKGLO-1)/NPROMA+1   ! Current block number
       ICEND=MIN(NPROMA,NGPTOT-JKGLO+1) ! Block length <= NPROMA

       CALL CLOUDSC ( 1, ICEND, NPROMA, KLEV, &
                    & array(1,1,IBL), & ! ~ 65 arrays like this
                    )
    END DO

!$OMP END DO
!$OMP END PARALLEL
```

NGPTOT per MPI-task ~ 40,000 on T2047 L 137 ~10km

Typical values for NPROMA in OpenMP implementation: 24 – 64

Development of OpenACC/GPU-version

- **The driver-code with OpenMP-loop kept ~ intact**
  – OpenACC (in most cases) can co-exist with OpenMP
- **CLOUDSC (~3,500 lines of Fortran2004) was pre-processed through ”`acc_insert`” Perl-script →**
  – Automatic creation of **ACC KERNELS** and **ACC DATA PRESENT / CREATE** clauses to CLOUDSC
- **With an effort of “one long working day” and use of profiling tool ”`nvprof`” the GPU-compute time came down from original 40s to 0,24s on a single nVIDIA K40 (using PGI 14.7 compiler)**

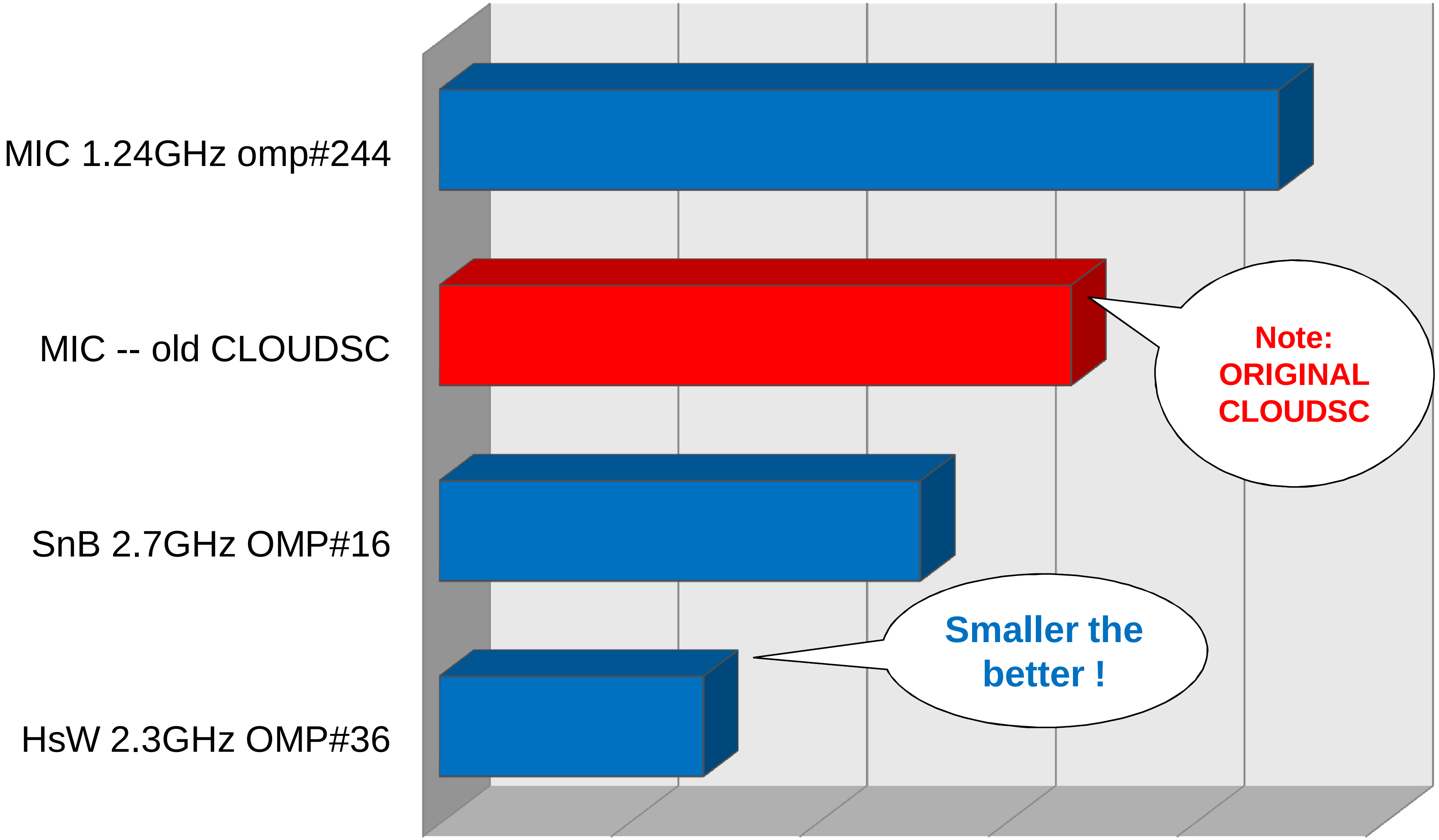
Driving CLOUDSC with OpenACC

```
!$OMP PARALLEL PRIVATE(JKGLO,IBL,ICEND) &
!$OMP& PRIVATE(tid, idgpu) num_threads(NumGPUs)
tid = omp_get_thread_num() ! OpenMP thread number
idgpu = mod(tid, NumGPUs)  ! Effective GPU# for this thread
CALL acc_set_device_num(idgpu, acc_get_device_type())
!$OMP DO SCHEDULE(STATIC)
    DO JKGLO=1,NGPTOT,NPROMA  ! NPROMA-loop
       IBL=(JKGLO-1)/NPROMA+1  ! Current block number
       ICEND=MIN(NPROMA,NGPTOT-JKGLO+1) ! Block length <= NPROMA
       !$acc data copyout(array(:,:,IBL), ...) & ! ~22 : GPU to Host
       !$acc& copyin(array(:,:,IBL))             ! ~43 : Host to GPU

       CALL CLOUDSC (... array(1,1,IBL) ...) ! Runs on GPU#<idgpu>

       !$acc end data
    END DO
!$OMP END DO
!$OMP END PARALLEL
```

Typical values for NPROMA in OpenACC implementation: > 10,000

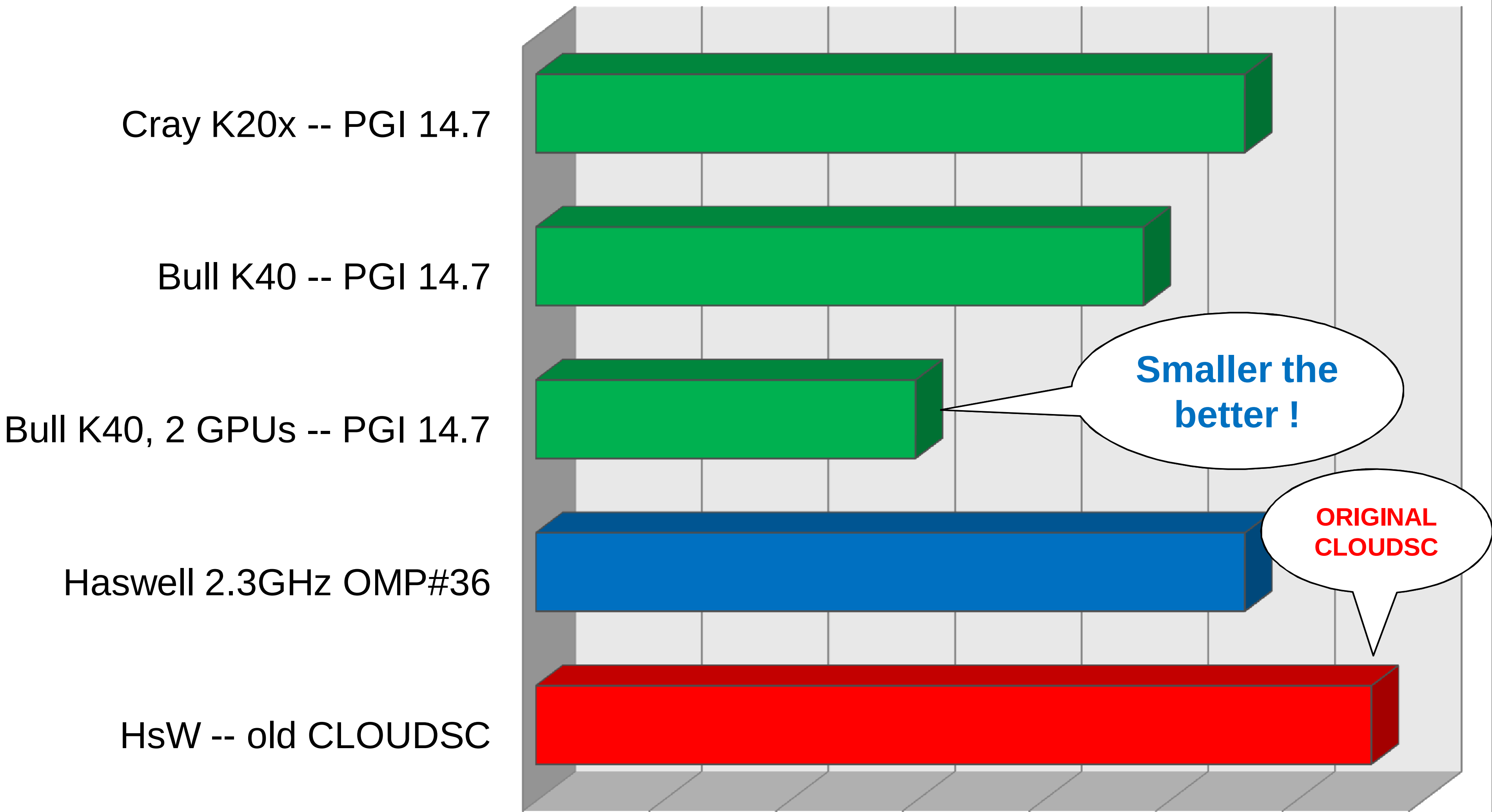
Some initial results

CLOUDSC: Xeon & XeonPhi (MIC) – Intel compilers



	HsW 2.3GHz OMP#36	SnB 2.7GHz OMP#16	MIC -- old CLOUDSC	MIC 1.24GHz omp#244
Time (s)	0,28	0,51	0,67	0,89

CLOUDSC (acc kernels) : GPU compute time only



	HsW -- old CLOUDSC	Haswell 2.3GHz OMP#36	Bull K40, 2 GPUs -- PGI 14.7	Bull K40 -- PGI 14.7	Cray K20x -- PGI 14.7
Time (s)	0,33	0,28	0,15	0,24	0,28

Hybrid version : CPU-cores + GPU(s) ?

- **Since CPU/MIC versions favour rather small block size NPROMA and GPUs prefer it to be as large as possible → leads to some unexpected problems:**
  - A hybrid version, where all CPUs and GPUs on a node will be utilized, cannot realistically be run due to contradictory requirements for optimal choice of NPROMA
    - **For now : use MPI-tasks to separate CPU blocks from GPUs**
  - Large NPROMA requirement on GPUs also make memory reservation on the Host-side pretty high, e.g. at L 137 :
    - **Just CLOUDSC requires ~ NPROMA / 10,000 GBytes of STACK**

Obstacles with OpenACC

- **Only 2 compilers available at present**
  - PGI favours **ACC KERNELS**
  - CRAY/CCE favours **ACC PARALLEL**
- **Performance cross-difference can be > 10X !!**
- **Potential need to maintain 2 CLOUDSC versions**
  - Or **3** when considering the old CLOUDSC better on MICs
- **The 2 compilers also introduce different levels of overheads in ACC DATA mapping the i.e. the way to build Host & GPU data relationships**
  - Shouldn't these even out when these compilers mature ?

1% of CLOUDSC (acc kernels) [PGI]

```
!$ACC KERNELS LOOP COLLAPSE(2) PRIVATE(ZTMP_Q,ZTMP)
   DO JK=1,KLEV
      DO JL=KIDIA,KFDIA
         ztmp_q = 0.0_JPRB
         ztmp = 0.0_JPRB
         !$ACC LOOP PRIVATE(ZQADJ) REDUCTION(+:ZTMP_Q) REDUCTION(+:ZTMP)
         DO JM=1,NCLV-1
            IF (ZQX(JL,JK,JM)<RLMIN) THEN
               ZLNEG(JL,JK,JM) = ZLNEG(JL,JK,JM)+ZQX(JL,JK,JM)
               ZQADJ                = ZQX(JL,JK,JM)*ZQTMST
               ztmp_q = ztmp_q + ZQADJ
               ztmp = ztmp + ZQX(JL,JK,JM)
               ZQX(JL,JK,JM)        = 0.0_JPRB
            ENDIF
         ENDDO
         PSTATE_q_loc(JL,JK) = PSTATE_q_loc(JL,JK) + ztmp_q
         ZQX(JL,JK,NCLDQV)   = ZQX(JL,JK,NCLDQV) + ztmp
      ENDDO
   ENDDO
!$ACC END KERNELS LOOP
```

1% of CLOUDSC (acc parallel) [CCE]

```
!$ACC PARALLEL LOOP COLLAPSE(2) PRIVATE(ZQADJ,ZTMP_Q,ZTMP)
   DO JK=1,KLEV
      DO JL=KIDIA,KFDIA
         ztmp_q = 0.0_JPRB
         ztmp = 0.0_JPRB
 !        !$ACC LOOP PRIVATE(ZQADJ) REDUCTION(+:ZTMP_Q) REDUCTION(+:ZTMP)
         DO JM=1,NCLV-1
            IF (ZQX(JL,JK,JM)<RLMIN) THEN
               ZLNEG(JL,JK,JM) = ZLNEG(JL,JK,JM)+ZQX(JL,JK,JM)
               ZQADJ              = ZQX(JL,JK,JM)*ZQTMST
               ztmp_q = ztmp_q + ZQADJ
               ztmp = ztmp + ZQX(JL,JK,JM)
               ZQX(JL,JK,JM)       = 0.0_JPRB
            ENDIF
         ENDDO
         PSTATE_q_loc(JL,JK) = PSTATE_q_loc(JL,JK) + ztmp_q
         ZQX(JL,JK,NCLDQV)   = ZQX(JL,JK,NCLDQV) + ztmp
      ENDDO
   ENDDO
!$ACC END PARALLEL LOOP
```

OpenACC compilers :

CSC

nVIDIA / PGI

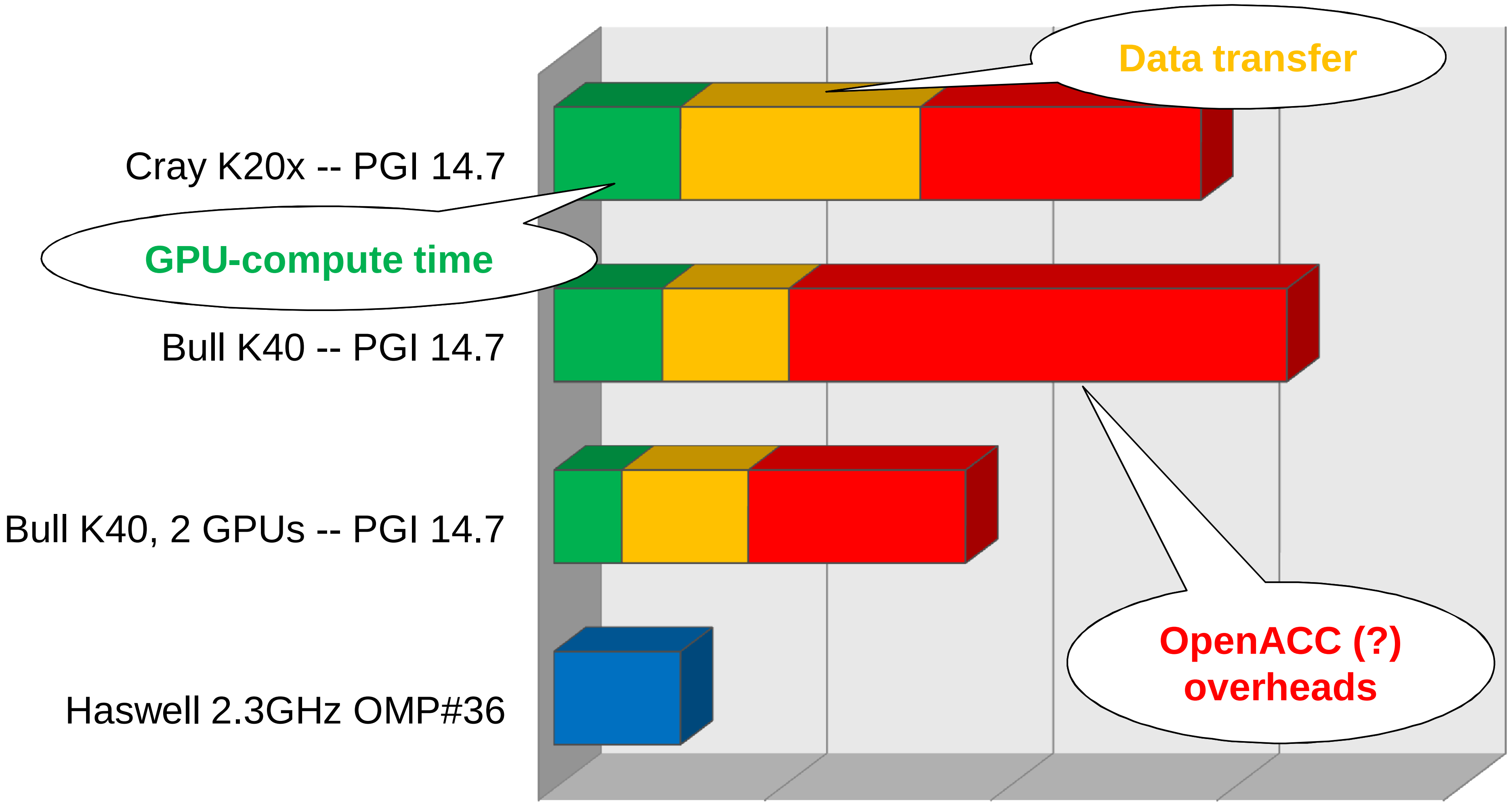
- Runs on all nVIDIA GPU-platforms, also Cray
- Often better performance with ACC KERNELS
- ACC DATA CREATE and ACC array PRESENT testing introduced relatively large overheads
- Host memory pinning for GPU transfers seemed to create large overheads

Cray / CCE

- Available only on Cray
- Favours ACC PARALLEL loops, thus potentially two OpenACC versions required (CCE not available on non-Cray GPU-platforms)
- ACC DATA CREATE and ACC PRESENT testing as well as memory pinning seemed not to cause big overheads compared PGI

Some results with GPU overheads

CLOUDSC (acc kernels) : GPU times with overheads



	Haswell 2.3GHz OMP#36	Bull K40, 2 GPUs -- PGI 14.7	Bull K40 -- PGI 14.7	Cray K20x -- PGI 14.7
Time (s)	0,28	0,15	0,24	0,28
Xfer (s)		0,28	0,28	0,53
Ovhd (s)		0,48	1,1	0,62

Next steps

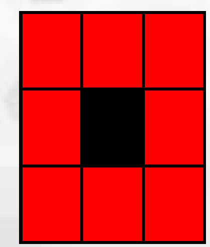
- Preliminary GPU migration of CLOUDSC has shown that code sharing with conventional CPUs is indeed possible with OpenACC – despite PGI & CCE compiler differences
- GPU migration can also discover more parallelism as some parts of the code gets looked into more thoroughly
  - Often with improved CPU performance, too
- Full ECMWF/IFS physics needs to be analyzed with most of the arrays residing **permanently** on GPUs, and with time stepping included
- Also remember: Intel KNL (Knights Landing) ~ 2016

Some conclusions

- IFS physics currently favours OpenMP way of coding and runs brilliantly on Intel Xeon (even on "MIC" type of systems, when MPI is not disturbing)
- OpenACC version on GPUs requires extraordinary large NPROMA, but then even a single K40 GPU "beats" a full node Intel Xeon/Haswell hands down (when not counting overheads & data transfers)
- OpenACC needs to mature : exactly the two available compilers (PGI & CCE) require exactly two different coding styles (dilemma as a result of ACC KERNELS vs. ACC PARALLEL approaches)

Dropped out from this presentation

- **OpenMP 4.0 accelerator directives**
  – Presently a major waste of HPC programmers' time ☹
- **On Cray/CCE CLOUDSC/OpenACC migration :**
  – Still need to understand a number of discrepancies over seemingly more robust looking PGI compiler
  – ACC PARALLEL vs. ACC KERNELS is haunting us
  – Note: Cray/CCE CPU-performance often superb ☺
- **Sub-columning technique, where each grid point column is replicated (say) by 9X :**
  – Effective NGPTOT ~ 360,000 fits an runs well on 32GB Intel Xeon CPU-servers
  – Runs OK on K20X / K40 GPU systems with OpenACC
  – Does NOT FIT into current generation XeonPhi MICs ☹



A special thanks to

- Dr. Peter Messmer (nVIDIA) for invaluable suggestions – and keeping PGI compiler developers busy
- Dr. Alistair Hart (Cray) for getting a version of CLOUDSC working with Cray/CCE OpenACC
- And Cray Inc. for providing very easy access to their Swan development resource in US