Porting the parallel Nek5000 application to GPU accelerators with OpenMP4.5

Alistair Hart
(Cray UK Ltd.)

EPiGRAM

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

EPiGRAM Project

3 years to Oct.16

6 partners

www.epigram-project.eu

EPiGRAM

€3M

KTH (SE)

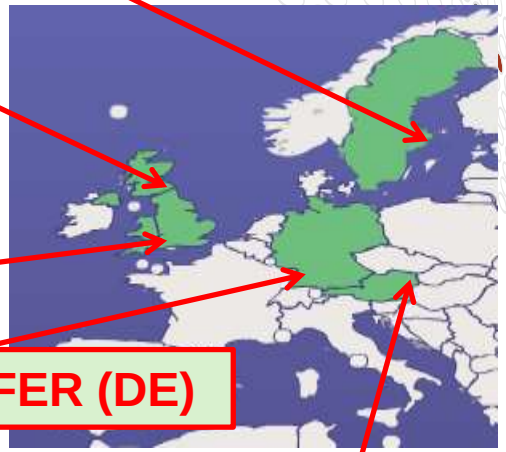
EPCC (UK)

CRAY (UK)

FRAUNHOFER (DE)

TUW (AT)

UIUC (USA)



CRAY

CREST

Contents of the talk

- A brief introduction to accelerator directives
- Porting the NekBone mini-app to OpenMP
- Porting the Nek5000 code from OpenACC to OpenMP

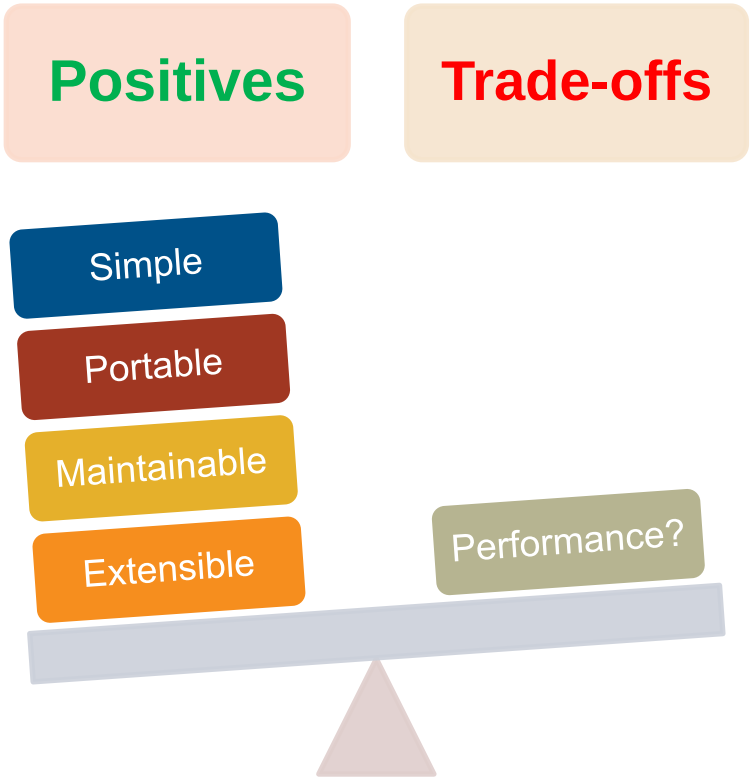
Directive based programming

Add directives to code

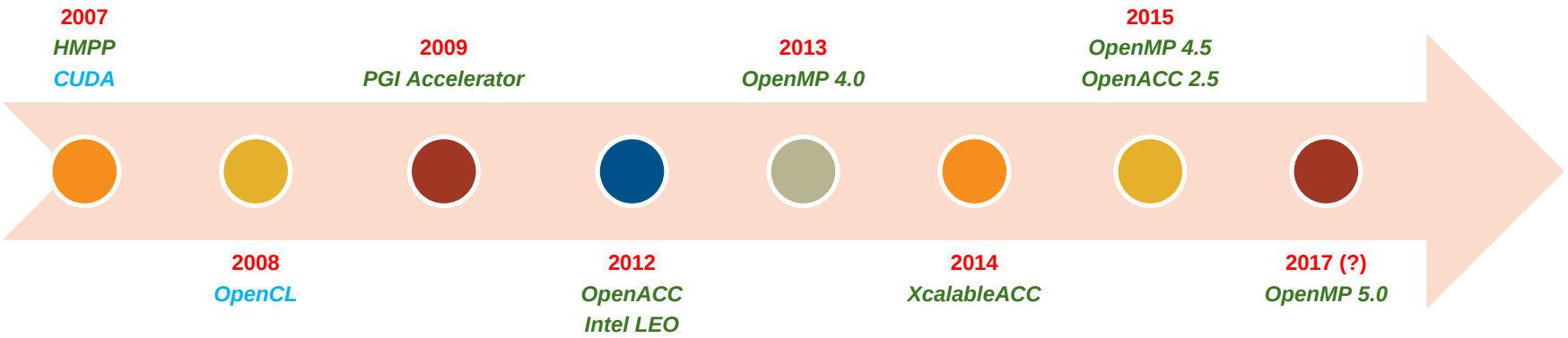
Compile for CPU as usual

Or compile for accelerator

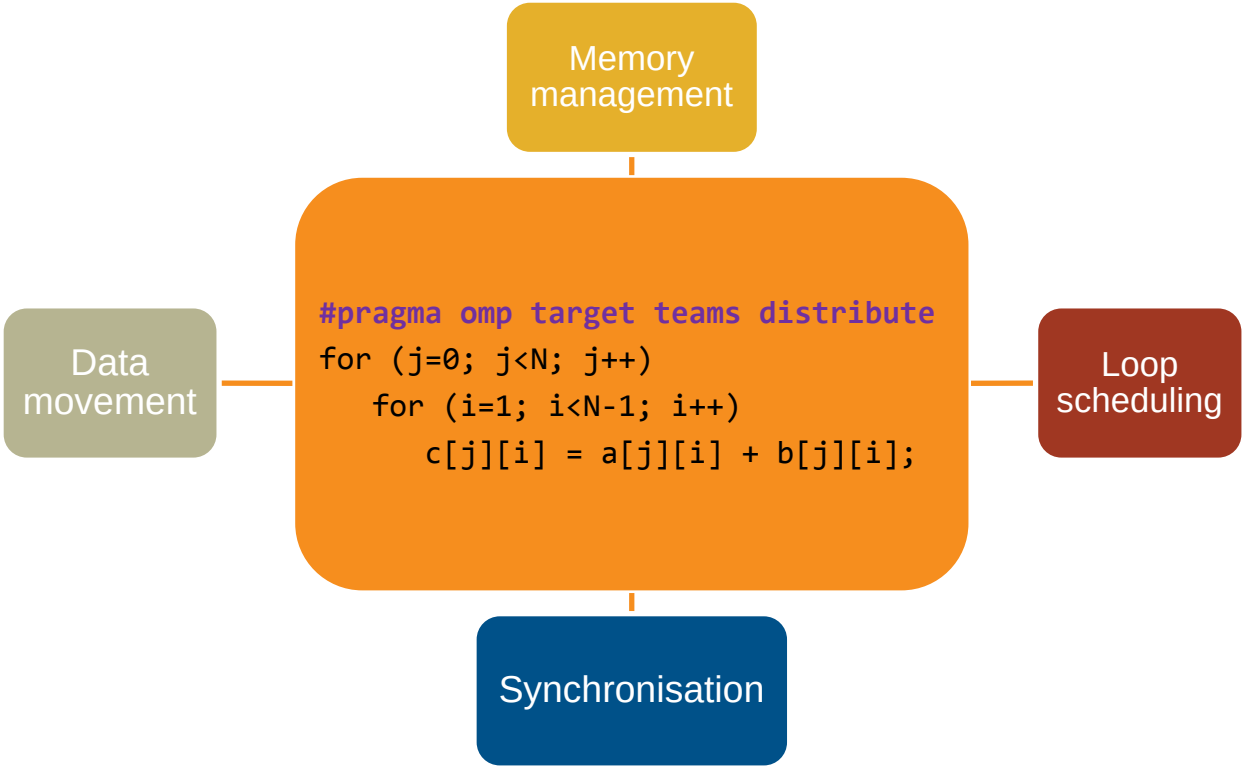
Why use accelerator directive models?



Accelerator directives are not new



A simple example



target teams distribute

- **target creates an offload kernel**
- **teams creates a "league" of "threadteams"**
  - Similar to a parallel region, but can't synchronise
  - Compiler chooses number of teams and threads per team
  - Can over-ride this with optional clauses
- **distribute partitions loop iterations over threads**
  - Multiple loops can be partitioned (unlike host OpenMP)

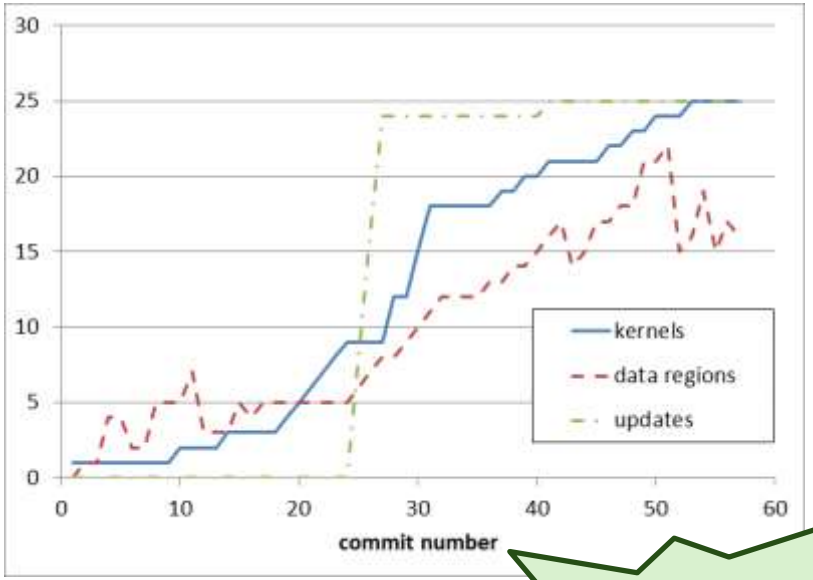
NekBone and Nek5000

Nek5000 and NekBone

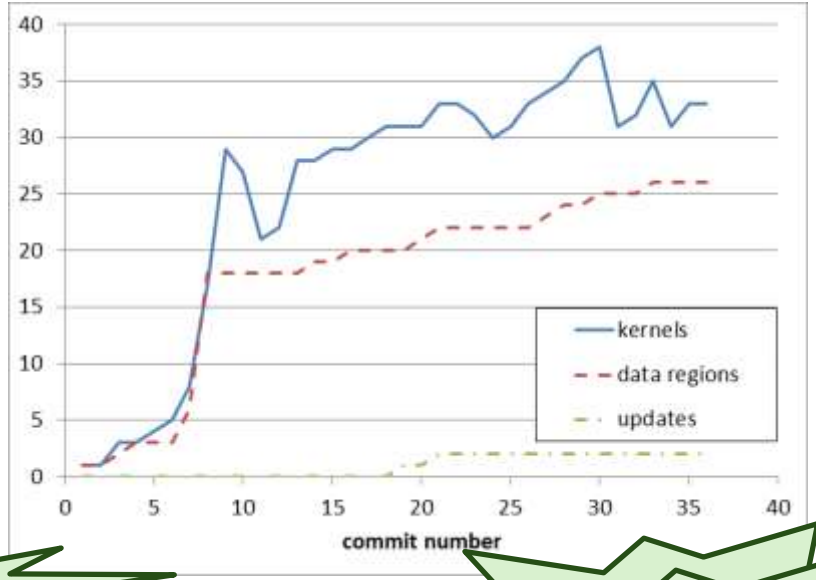
- **Nek5000: CFD code**
  - simulates incompressible fluids.
  - solves Navier-Stokes equations
  - semi-spectral element method.
- **~70k lines of code:**
  - 90% in Fortran 77
  - 10% in C (comms).
  - Parallelised with MPI
- **NekBone mini-app**
  - captures this in 11k lines

How many directives?

OpenMP



OpenACC



420 Mflops

G2G MPI

NekBone kernel performance (OpenMP4.0)

local_oz

```
G----------< !$omp target teams distribute
G            !$omp&  private(e_back,e_front)
G g---------< do e_z =1,mz-1
G g C-------<  do e_y = 1,my
G g C C-----<   do e_x = 1,mx
```

```
G----------< !$omp target teams distribute collapse(3)
G            !$omp&  private(e_back,e_front)
G C---------< do e_z =1,mz-1
G C C-------<  do e_y = 1,my
G C C g-----<   do e_x = 1,mx
G C C g          e_back = e_x + mx*(e_y-1)+mx*my*(e_z-1)
G C C g          e_front = e_back+mx*my
G C C g r2--<    do i=1,n_shared
G C C g r2        w(l_face_z_back(i),e_back)= &
G C C g r2           w(l_face_z_back(i),e_back)+ &
G C C g r2           w(l_face_z_front(i),e_front)
G C C g r2        w(l_face_z_front(i),e_front) = &
G C C g r2           w(l_face_z_back(i),e_back)
G C C g r2-->    enddo
G C C g----->   enddo
G C C------->  enddo
G C--------->enddo
G----------->
```

863µs→142µs

```
G----------< !$acc parallel loop

G g---------< do e_z =1,mz-1
G g 3-------<  do e_y = 1,my
G g 3 4-----<   do e_x = 1,mx
```

```
G----------< !$acc parallel loop collapse(3)

G C---------< do e_z =1,mz-1
G C C-------<  do e_y = 1,my
G C C g-----<   do e_x = 1,mx
G C C g          e_back = e_x + mx*(e_y-1)+mx*my*(e_z-1)
G C C g          e_front = e_back+mx*my
G C C g r2-<     do i=1,n_shared
G C C g r2        w(l_face_z_back(i),e_back)= &
G C C g r2           w(l_face_z_back(i),e_back)+ &
G C C g r2           w(l_face_z_front(i),e_front)
G C C g r2        w(l_face_z_front(i),e_front) = &
G C C g r2           w(l_face_z_back(i),e_back)
G C C g r2->     enddo
G C C g----->   enddo
G C C------->  enddo
G C--------->enddo
G----------->
```

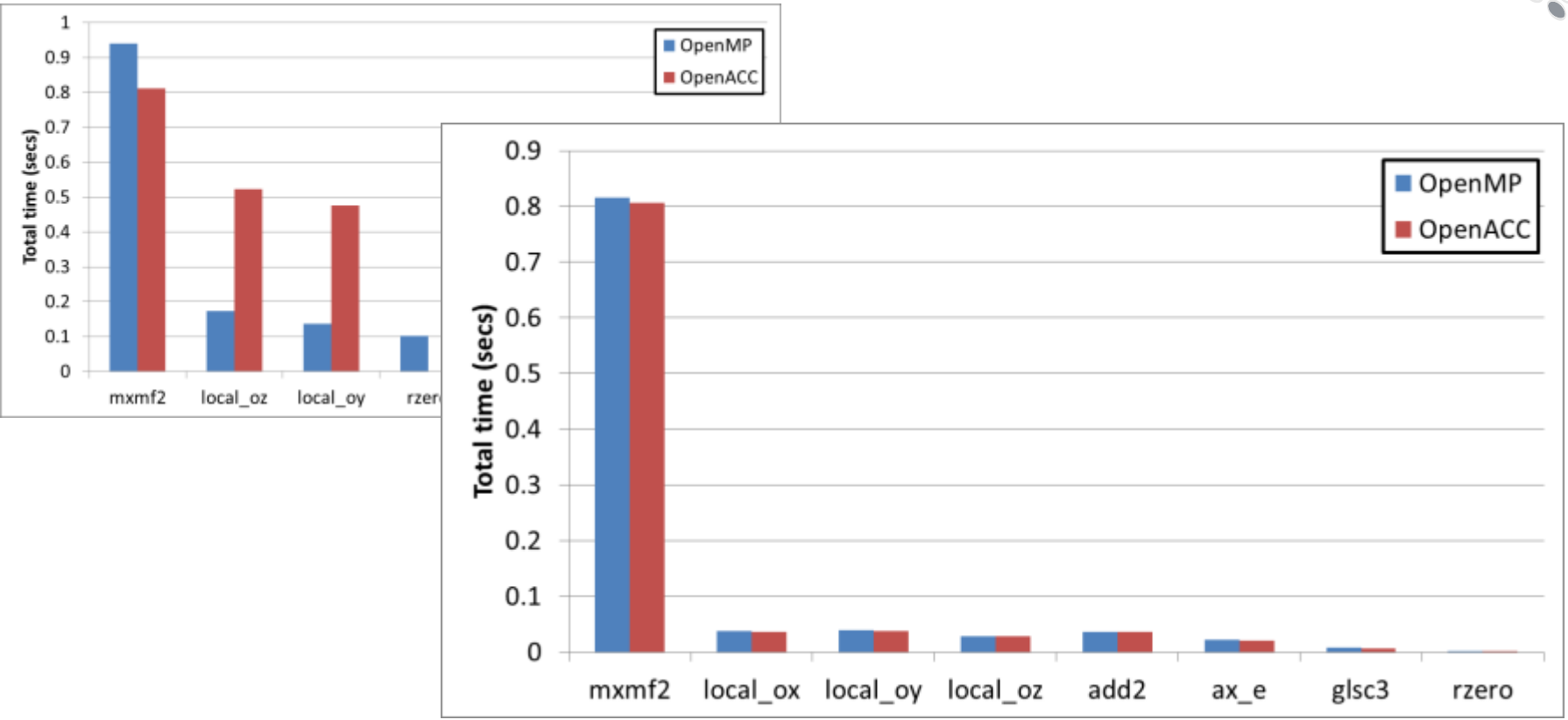
2592µs→141µs

mxmf2

```
G---------< !$omp target teams distribute
G g-------< DO j = 1,n3
G g g-----<  DO i = 1,n1
G g g         c(i,j) = 0
G g g r4--<   DO k = 1,n2
G g g r4       c(i,j) = c(i,j) + a(i,k)*b(k,j)
G g g r4-->   ENDDO
G g g----->  ENDDO
G g------> ENDDO
G---------> !$omp end target teams d
```

```
G---------< !$acc parallel loop
G g-------< DO j = 1,n3
G g g-----<  DO i = 1,n1
G g g         c(i,j) = 0
G g g r4--<   DO k = 1,n2
G g g r4       c(i,j) = c(i,j) + a(i,k)*b(k,j)
G g g r4-->   ENDDO
G g g----->  ENDDO
G g------> ENDDO
G---------> !$acc end parallel loop
```

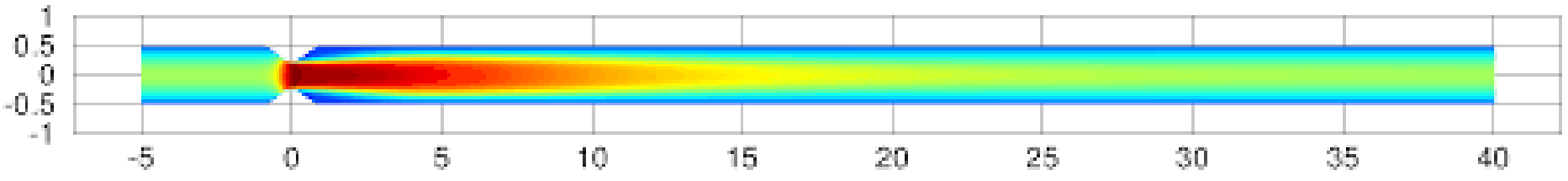
NekBone performance after tuning



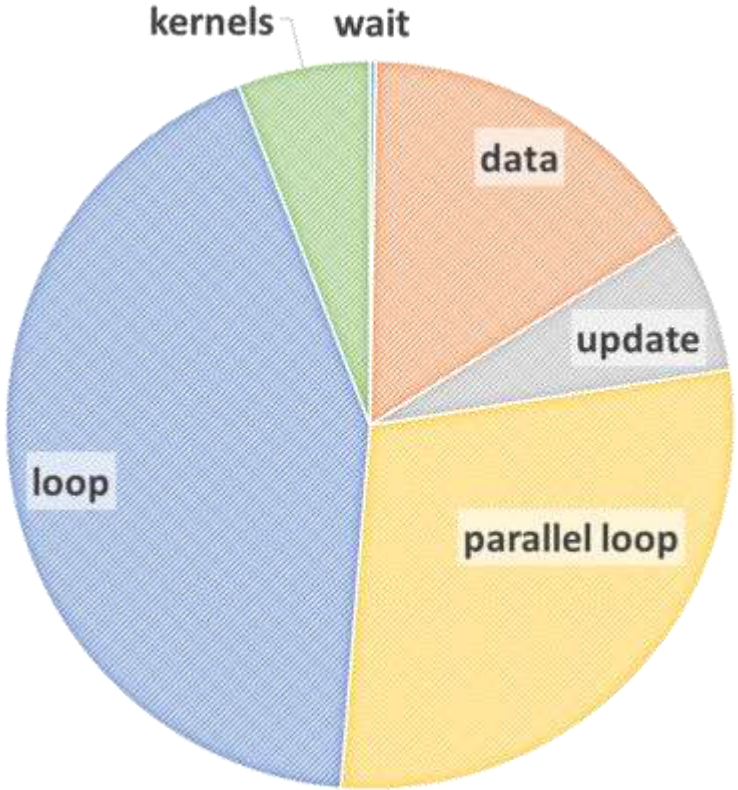
Nek5000

Nek5000

- **Already ported to OpenACC**
- **Migrating to OpenMP?**
  - Can we do this without losing performance?
  - What are the challenges in migrating?
  - What are the advantages of migration



OpenACC directive audit



- **74000 source lines**
  - 60000 lines of code
- **691 directive lines:**
  - 390 directives
  - 101 continuation lines
  - 200 end directives
- **155 loc/directive**
- **Traffic lights and alarm bells:**
  - data, update, parallel loop, wait
  - loop
  - kernels

Planning the migration with the calltree

- 13 source files
- Interoperability is key

```
Table 1:  Calltree View with Callsite Line Numbers

  Time% |      Time |     Calls |Calltree
        |           |           | PE=HIDE

 100.0% | 25.213824 |        -- |Total
|  54.8% | 13.811506 |         -- |nekton_:drive.f:line.50
||  54.8% | 13.811480 |          -- |nek_solve_:drive1.f:line.260
3||  40.3% | 10.164313 |           -- |nek_solve_.ACC_DATA_REGION@li.260:drive1.f:line.328
4|||  39.5% |  9.966259 |            -- |nek__multi_advance_:drive1.f:line.446
5|||       |           |               | nek_advance_:drive1.f:line.384
6|||       |           |               |  fluid_:drive2.f:line.758
7||||||  31.2% |  7.869682 |           -- |plan4_acc_:plan4.f:line.532
8|||||||  28.5% |  7.174353 |            -- |plan4_acc_.ACC_DATA_REGION@li.532:plan4.f:line.453
9|||||||  28.5% |  7.174286 |             -- |hsolve_acc_:navier4.f:line.952
10|||||||  28.5% |  7.174220 |             -- |hmholtz_acc_:hmholtz.f:line.1521
11|||||||||  28.4% |  7.163318 |             -- |hmholtz_acc_.ACC_DATA_REGION@li.1521:hmholtz.f:line.1468
12|||||||||  28.4% |  7.163282 |             -- |cggo_acc_:hmholtz.f:line.1581
13||||||||||   7.9% |  1.991531 |              -- |cggo_acc_.ACC_DATA_REGION@li.1581:hmholtz.f:line.1668
14||||||||||   7.9% |  1.981263 |              -- |fdm_h1_acc_:hmholtz.f:line.2161
15|||||||||||   5.0% |  1.249621 |               -- |fdm_h1_acc_.ACC_DATA_REGION@li.2161:hmholtz.f:line.2352
16|||||||||||   2.6% |  0.650272 |               -- |dssum_acc_:dssum.f:line.1002
17||||||||||||   2.1% |  0.519627 |   5,349.0 |dssum_acc_.ACC_DATA_REGION@li.1002:dssum.f:line.1002
18||||||||||||   0.0% |  0.007974 |   5,349.0 |dssum_acc_.ACC_COPY@li.1002:dssum.f:line.1002
```

Basic example

- **Both directives**
- **private scalars**
- **thread_limit**

- **a7_dudxyz.f**

```
                #ifdef _ACCEL_OPENMP
G-----------< !$omp target teams distribute collapse(4)
G             !$omp& private(tmpx) thread_limit(VLENGTH)
G             #endif
G             #ifdef _ACCEL_OPENACC
D             !$acc parallel loop collapse(4)
D             !$acc& private(tmpx) vector_length(VLENGTH)
G             #endif
G C---------< DO E = 1,NEL
G C C-------<  DO K = 1,NZ1
G C C C-----<   DO J = 1,NY1
G C C C g---<    DO I = 1,NX1
G C C C g         TMPX = 0.0
G C C C g 6-<     DO L = 1,NX1
G C C C g 6        TMPX = TMPX + DXTM1(L,I)*U(L,J,K,E)
G C C C g 6->     ENDDO
G C C C g         DU(I,J,K,E) = TMPX*RM1(I,J,K,E)
G C C C g--->    ENDDO
G C C C----->   ENDDO
G C C------->  ENDDO
G C--------> ENDDO
```

Providing loop information

- collapse

```
G------------< !$omp target teams distribute
G g----------< DO IE=1,NEL
G g            !$omp simd collapse(3)
D              !$acc loop collapse(3)
G g C--------<  IZ=1,NZ1
G g C C------<  DO IY=1,NY1
G g C C g----<   DO IX=1,NX1
G g C C g         DPCM1(IX,IY,IZ,IE) = 0.0
G g C C g r4-<    DO IQ=1,NX1
G g C C g r4       DPCM1(IX,IY,IZ,IE) += ...
G g C C g r4->    ENDDO
G g C C g---->   ENDDO
G g C C------>  ENDDO
G g C-------->ENDDO
```

- a2_setprec.f

Providing loop information (2)

- Reductions on inner loops
- forced collapse

```
G-----------< !$omp target teams distribute private(h1b)
G g---------< DO ie=1,nel
G g           h1b = 0
G g          !$omp simd collapse(3) reduction(+:h1b)
D            !$acc loop reduction(+:h1b)
G g g-------<  DO i3=1,nz1
G g g C-----<   DO i2=1,ny1
G g g C Cr4-<    DO i1=1,nx1
G g g C Cr4       h1b = h1b + h1(i1,i2,i3,ie)
G g g C Cr4->    ENDDO
G g g C----->   ENDDO
G g g------->  ENDDO
```

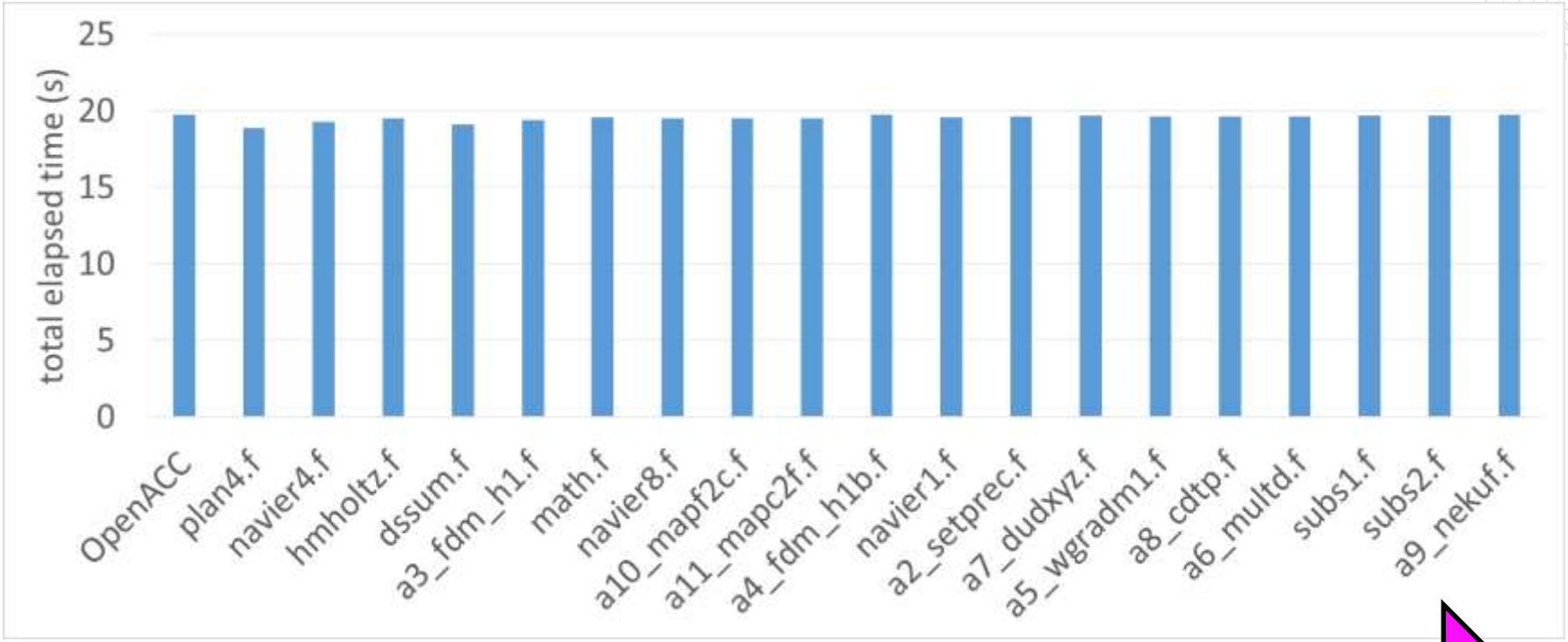
- a4_fdm_h1b.f

Providing loop information (3)

- **No equivalent:**
  - Elimination process
  - Compiler directives
    - CCE: !dir$ novector
- **a10_mapf2c.f**

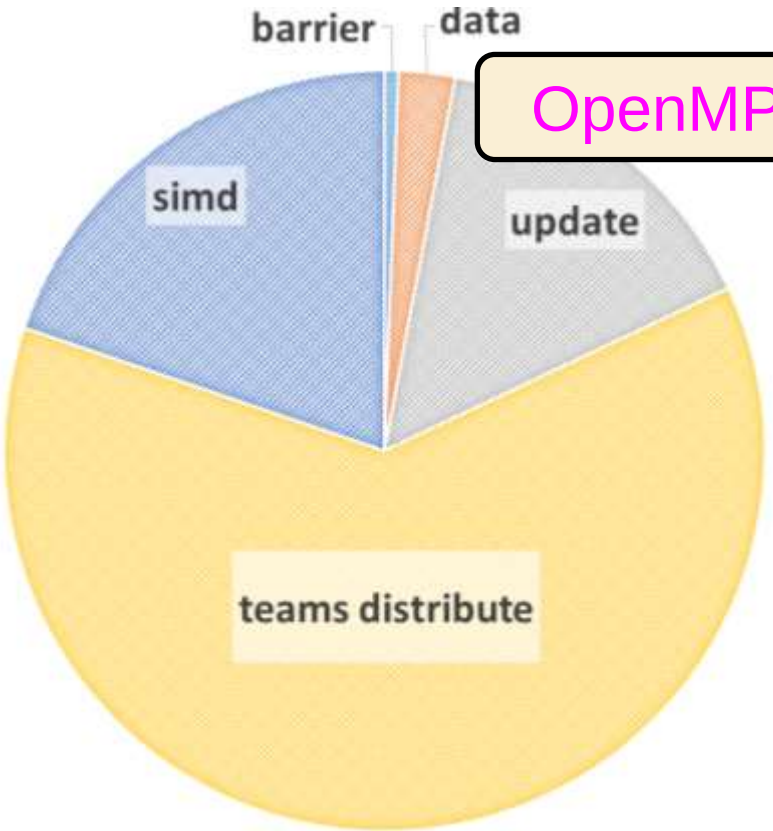
```
G------------< !$omp target teams distribute
G------------< !$omp& simd collapse(4)
G C----------< DO e = 1,nelv
G C C--------<  DO k = 1,nz1
G C C C------<   DO j = 1,ny1
G C C C g----<    DO i = 1,n2
G C C C g          tmpv = 0.0
D                !$acc loop seq
G C C C g r4-<     DO l = 1,nx1
G C C C g r4        tmpv = tmpv + ...
G C C C g r4->     ENDDO
G C C C g          v2(i,j,k,e) = tmpv
G C C C g---->    ENDDO
G C C C------>   ENDDO
G C C-------->  ENDDO
G C---------->  ENDDO
```

Nek5000 performance during migration



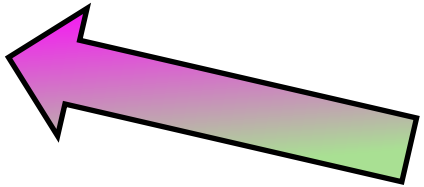
Porting time

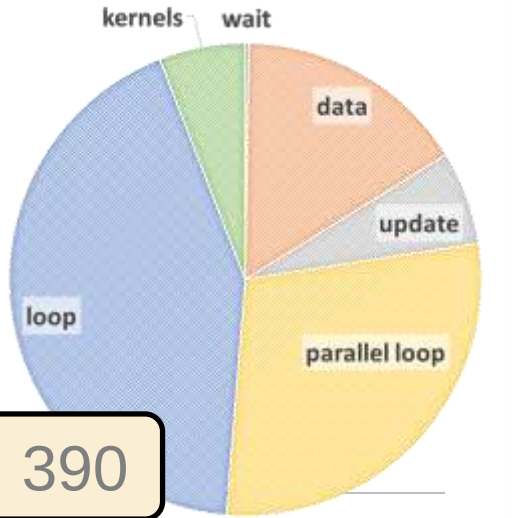
OpenMP directive audit



OpenMP: 167

- **Fewer:**
  - data regions
  - loop directives

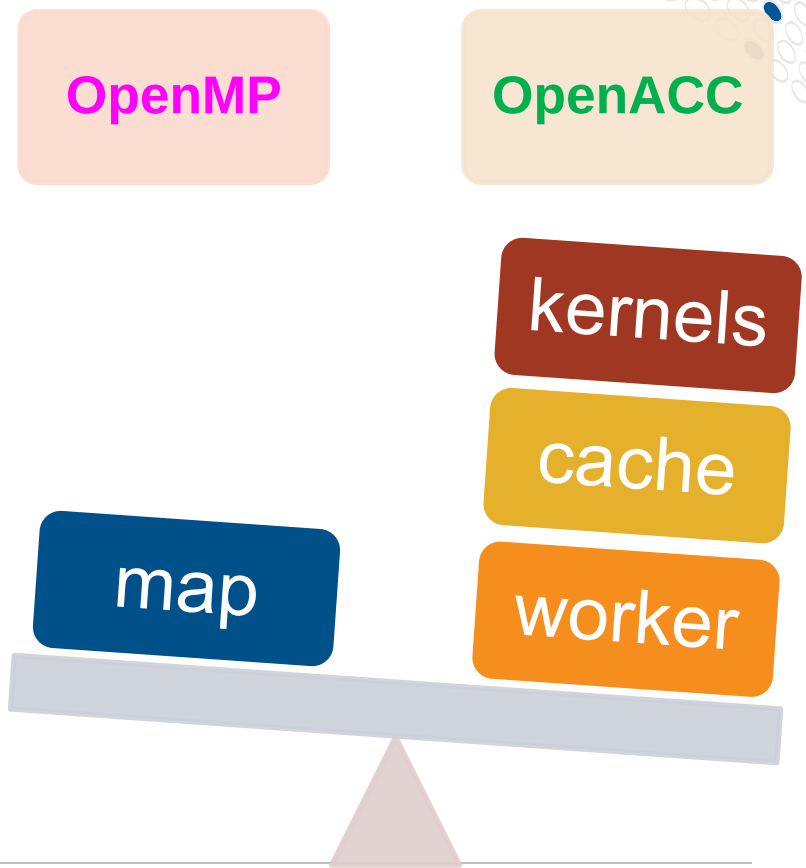




OpenACC: 390

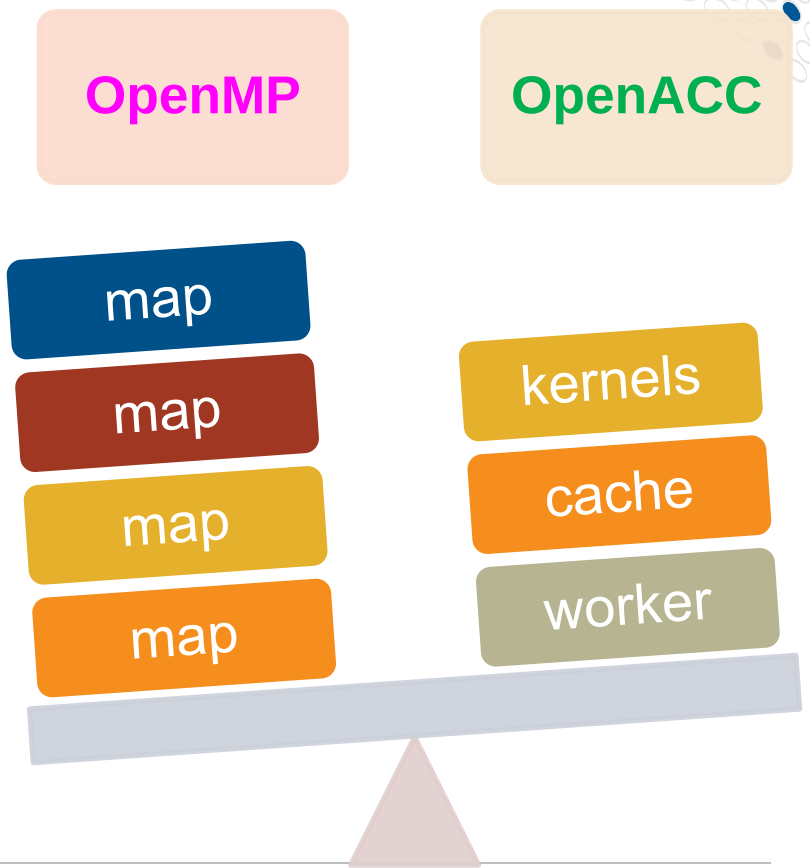
OpenMP4.5 versus OpenACC

- **OpenMP4.5 a close match**
  - added:
    - use_device_ptr
    - scalars firstprivate by default
    - unstructured data regions
    - better async control



OpenMP4.5 versus OpenACC

- **OpenMP4.5 a close match**
  - added:
    - use_device_ptr
    - scalars firstprivate by default
    - unstructured data regions
    - better async control



Conclusions

- **Directives offer productive performance-portability**
  - Nek5000/NekBone: 1 directive per 160 lines of code
- **OpenMP device constructs: mature programming model**
- **OpenMP device constructs can (should) be performant**
  - CCE: default comparable and often better than OpenACC
  - Fewer tuning clauses, but does not appear to be a problem
- **Straightforward migration path: OpenACC ⇔ OpenMP**
  - Interoperability is a great help for incremental porting
  - Subtleties but nothing deal-breaking
  - OpenMP tends to use fewer directives, more maintainable

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.