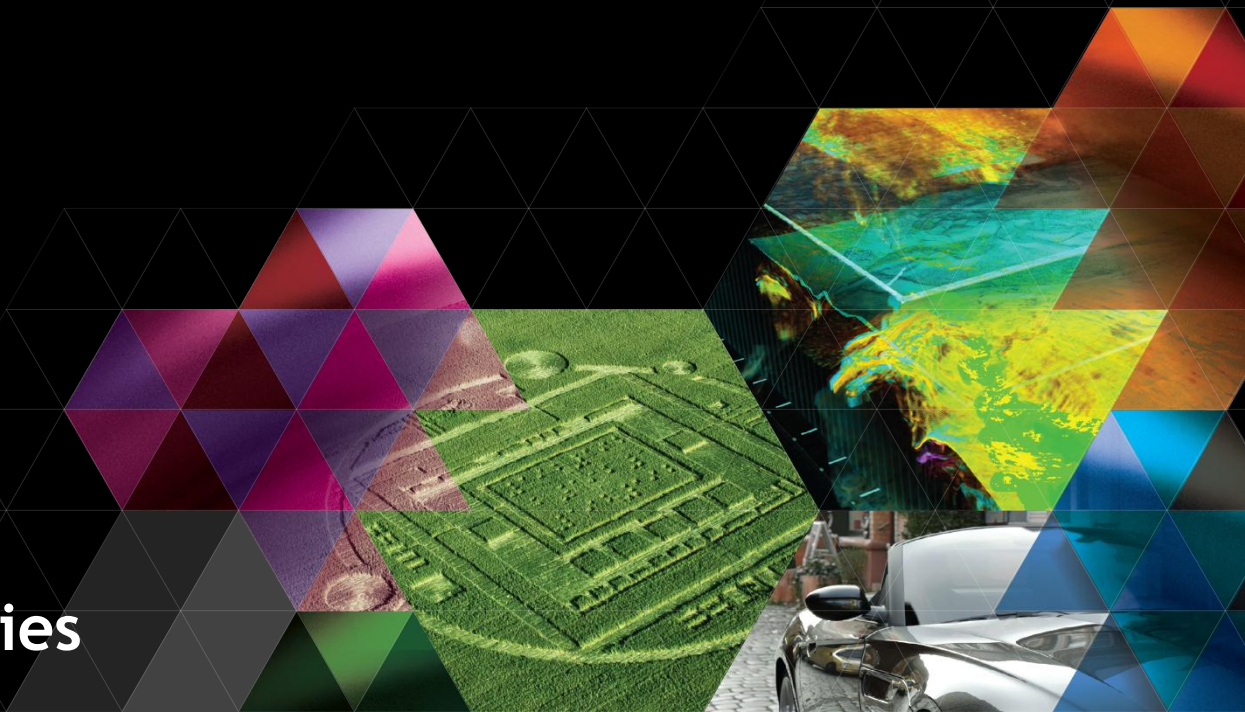
GPU TECHNOLOGY CONFERENCE

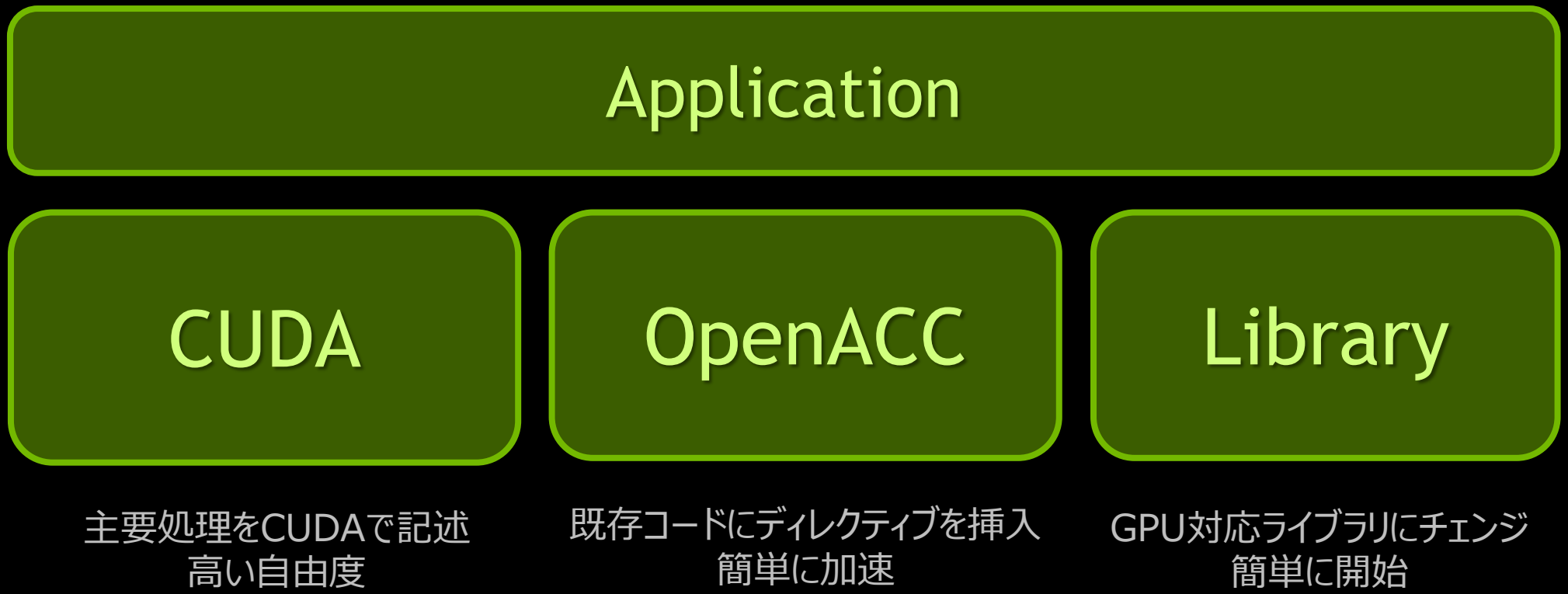
NVIDIA

OPENACCの現状



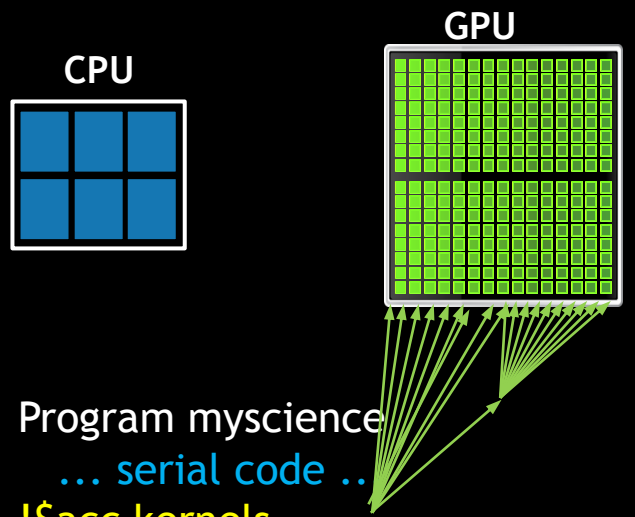
Akira Naruse
NVIDAI Developer Technologies

アプリをGPUで加速する方法



GPU TECHNOLOGY CONFERENCE

OPENACC




```
Program myscience
   ... serial code ..
!$acc kernels
   do k = 1,n1
      do i = 1,n2
         ... parallel code ...
      enddo
    enddo
!$acc end kernels
   ... serial code ...
End Program myscience
```

ヒントの追加

既存のC/Fortranコード

簡単: 既存のコードに
コンパイラへのヒントを追加

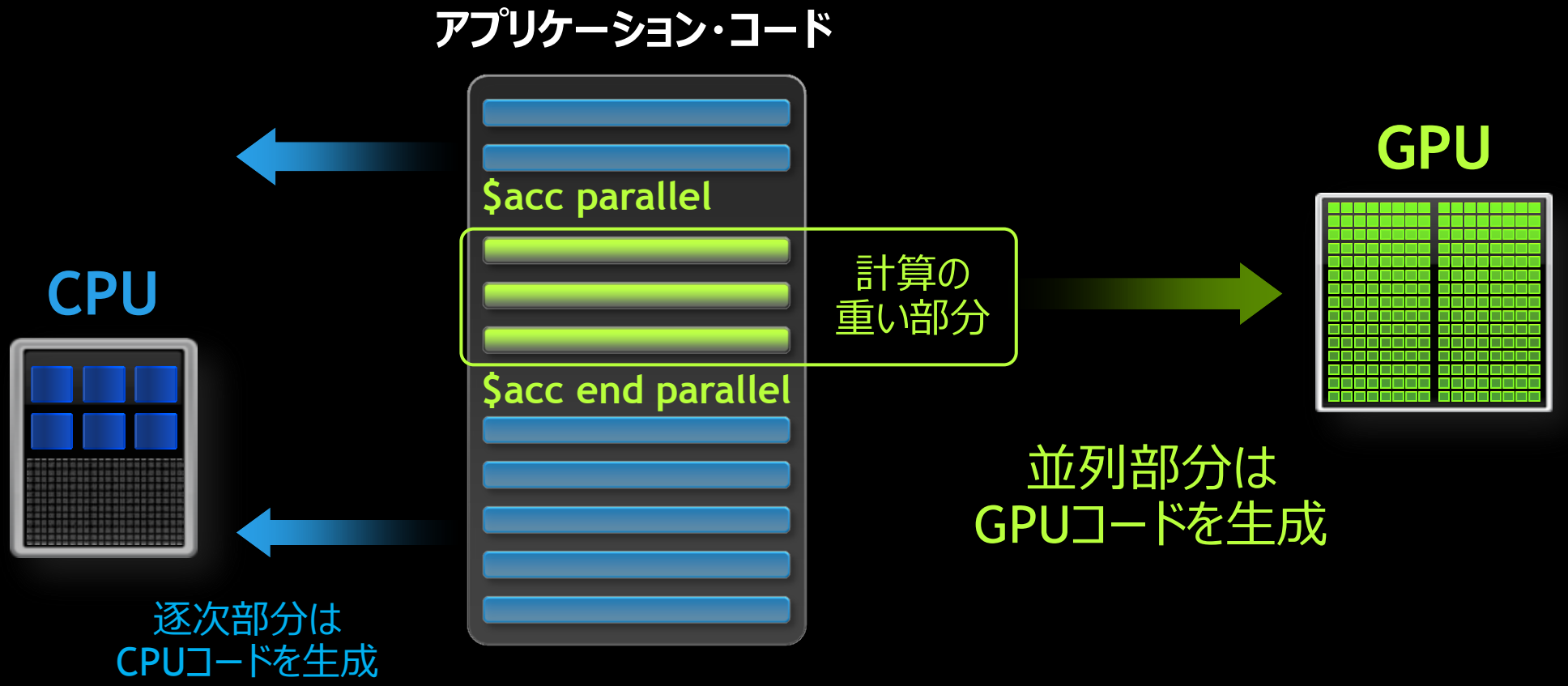
強力: そこそこの労力で、コンパイラが
コードを自動で並列化

オープン: 複数コンパイラベンダが、
複数アクセラレータをサポート
NVIDIA, AMD, Intel(予定)

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

実行モデル



SAXPY (Y=A*X+Y, C/C++)

OpenMP

```
void saxpy(int n,
           float a,
           float *x,
           float *restrict y)
{
#pragma omp parallel for
  for (int i = 0; i < n; ++i)
    y[i] += a*x[i];
}

...
saxpy(N, 3.0, x, y);
...
```

OpenACC

```
void saxpy(int n,
           float a,
           float *x,
           float *restrict y)
{
#pragma acc parallel copy(y[:n]) copyin(x[:n])
  for (int i = 0; i < n; ++i)
    y[i] += a*x[i];
}

...
saxpy(N, 3.0, x, y);
...
```

GPU TECHNOLOGY CONFERENCE

SAXPY (Y=A*X+Y, FORTRAN)

OpenMP

```
subroutine saxpy(n, a, X, Y)
  real :: a, X(:), Y(:)
  integer :: n, i

  !$omp parallel do
  do i=1,n
    Y(i) = a*X(i)+Y(i)
  enddo
  !$omp end parallel do
end subroutine saxpy


...
call saxpy(N, 3.0, x, y)
...
```

OpenACC

```
subroutine saxpy(n, a, X, Y)
  real :: a, Y(:), Y(:)
  integer :: n, i

  !$acc parallel copy(Y(:)) copyin(X(:))
  do i=1,n
    Y(i) = a*X(i)+Y(i)
  enddo
  !$acc end parallel
end subroutine saxpy


...
call saxpy(N, 3.0, x, y)
...
```

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

OPENMPとの併用

OpenMP / OpenACC

```
void saxpy(int n, float a,
           float *x,
           float *restrict y)
{
#pragma acc parallel copy(y[:n]) copyin(x[:n])
#pragma omp parallel for
  for (int i = 0; i < n; ++i)
    y[i] += a*x[i];
}

...
saxpy(N, 3.0, x, y);
...
```

簡単にコンパイル

OpenMP / OpenACC

```
void saxpy(int n, float a,
           float *x,
           float *restrict y)
```

```
$ pgcc -Minfo -acc saxpy.c
saxpy:
     16, Generating present_or_copy(y[:n])
         Generating present_or_copyin(x[:n])
         Generating Tesla code
     19, Loop is parallelizable
         Accelerator kernel generated
         19, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

```
...
```

GPU TECHNOLOGY CONFERENCE

簡単に実行

OpenMP / OpenACC

```
void saxpy(int n, float a,
           float *x,
           float *restrict y)
...
```

```
$ nvprof ./a.out
==10302== NVPROF is profiling process 10302, command: ./a.out
==10302== Profiling application: ./a.out
==10302== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 62.95%  3.0358ms         2  1.5179ms  1.5172ms  1.5186ms  [CUDA memcpy HtoD]
 31.48%  1.5181ms         1  1.5181ms  1.5181ms  1.5181ms  [CUDA memcpy DtoH]
  5.56%  268.31us         1  268.31us  268.31us  268.31us  saxpy_19_gpu
```

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

簡単に高速

Automotive

Real-Time Object Detection

Global Manufacturer of Navigation Systems

40時間で5倍

Financial

Valuation of Stock Portfolios using Monte Carlo

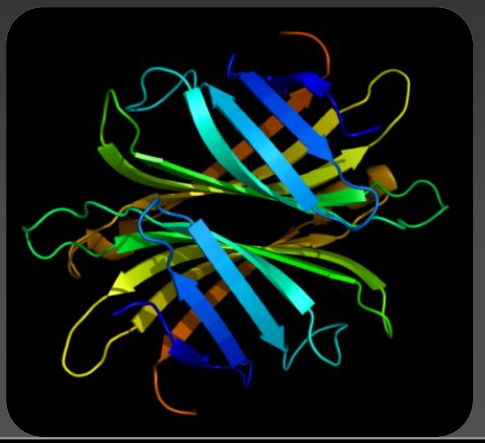
Global Technology Consulting Company

4時間で2倍

Life Science

Interaction of Solvents and Biomolecules

University of Texas at San Antonio



8時間で5倍

コンパイラとツール

2013年10月～

2013年12月～

2014年1月～

2015年(予定)

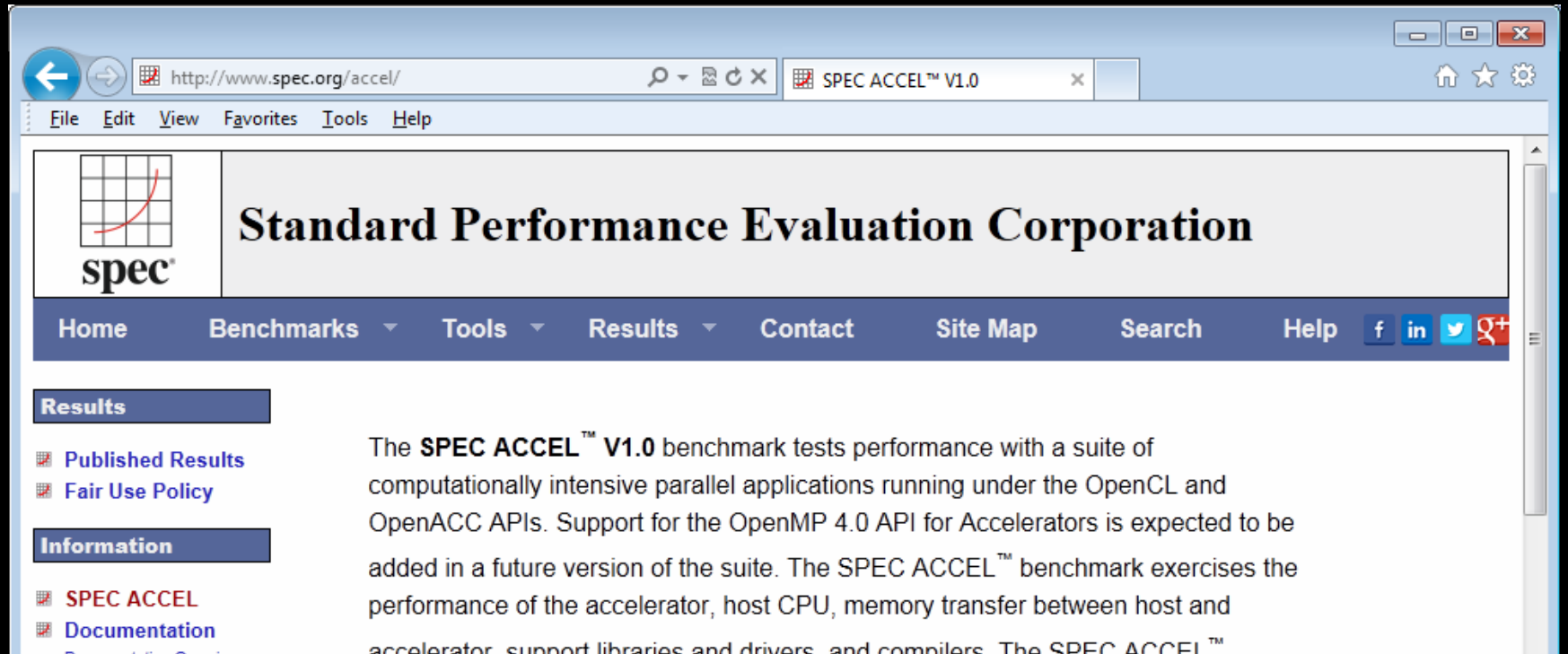
コンパイラ　　OpenACC 2.0対応

デバッグツール

SPEC ACCEL

- 15本のOpenACCベンチマーク

www.spec.org/accel



GPU TECHNOLOGY CONFERENCE

気象・天候・海洋モデル

気候(C)

NCAR-CISL, ORNL / CESM
- CAM-SE (HOMME)
- LANL / POP → MPAS-O

NASA / GEOS-5

NOAA-GFDL / CFSv2
- NOAA-GFDL / MOM6

UKMO / HadGEM3
- UM → GungHo
- NEMO

MPI-M / MPI-ESM
- ECHAM6 → ICON-ATM
- MPIOM → ICON-OCE

RIKEN, UniTokyo / NICAM

IPSL / DYNAMICO

NCAR-M3 / WRF → MPAS-A or NIM

DWD, MCH / COSMO

UniMiami / OLAM

天候(W)

UKMO / UM → GungHo

ECMWF / IFS → PantaRhei

DWD / GME → ICON

NOAA-NCEP / GFS → NIM?

EC, CMC / GEM

USNRL / NAVGEM

NOAA-ESRL / FIM → NIM

→ DWD, MPI-M / ICON

→ NOAA-ESRL / NIM

→ NCAR / MPAS-A

NCAR-M3 / WRF → MPAS-A or NIM

USNRL / COAMPS

DWD, MCH / COSMO

MFR / AROME

MFR, ICHEC / HARMONIE
- HIRLAM ALADIN

JAMSTEC-JMA / ASUCA

CAS-CMA / GRAPES

UniMiami / OLAM

海洋(O)

LANL / POP → MPAS-O

NOAA-GFDL MOM6

CNRS STFC NEMO

USNRL / HYCOM

MIT / MITgcm

→ LANL / MPAS-O

→ MPI-M / ICON-OCE

Rutgers-UCLA / ROMS

UNC-ND / ADCIRC

- GPU Development (8)
  - CAM-SE, GEOS-5, NEMO, WRF, COSMO, NIM, FIM, GRAPES
- GPU Evaluation (15)
  - POP, ICON, NICAM, OLAM, GungHo, PantaRhei, ASUCA, HARMONIE, COAMPS, HYCOM, MITgcm, ROMS, ADCIRC, DYNAMICO, MOM6
- GPU Not Started (7)
  - MPAS-A, MPAS-O, GFS, GEM, NAVGEM, AROME, ICON-OCE

→ Indicates Next-Gen Model

OPENACCへの移行

Model	Focus	GPU Approach	Collaboration
NCAR / WRF	NWP/Climate-R	(1) OpenACC, (2) CUDA	(1) NCAR-MMM, (2) SSEC UW-M
DWD / COSMO	NWP/Climate-R	CUDA+OpenACC	CSCS, MeteoSwiss (MCH)
ORNL / CAM-SE	Climate-G	CUDA-F → OpenACC	ORNL, Cray
NCAR / CAM-SE	Climate- G	CUDA,CUDA-F,OpenACC	NCAR-CISL
NOAA / NIM&FIM	NWP/Climate-G	F2C-ACC,OpenACC	NOAA-ESRL, PGI
NASA / GEOS-5	Climate-G	CUDA-F → OpenACC	NASA, PGI
CNRS / NEMO	Ocean GCM	OpenACC	STFC
UKMO / GungHo	NWP/Climate-G	OpenACC	STFC, UKMO in future?
USNRL / HYCOM	Ocean GCM	OpenACC	US Naval Research Lab
RIKEN / NICAM	Climate-G	OpenACC	RIKEN, UniTokyo
UNC / ADCIRC	Storm Surge	OpenACC (AmgX?)	LSU LONI
NOAA / MOM6	Ocean GCM	OpenACC	NOAA-GFDL
NASA / FV-Core	Atmospheric GCM	OpenACC	NASA, NOAA-GFDL

Other Evaluations: US - COAMPS, MPAS, ROMS, OLAM; Europe - ICON, IFS, HARMONIE; DYNAMICO
Asia-Pacific - ASUCA (JP), GRAPES (CN)

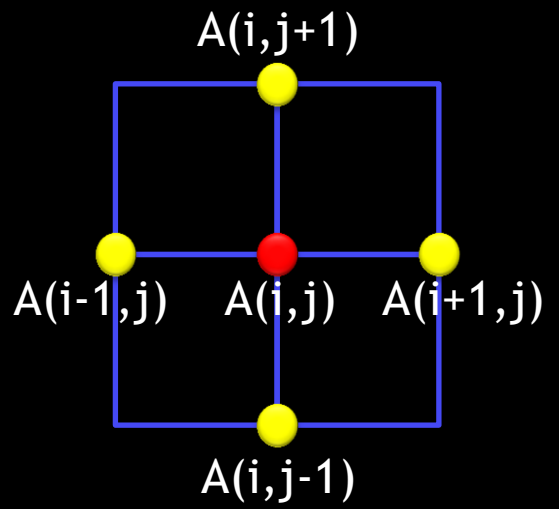
GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

OPENACCでどこまで出来るの？

例: JACOBI ITERATION

```
while ( error > tol ) {
  error = 0.0;

  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]));
    }
  }

  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
}
```



GPU TECHNOLOGY CONFERENCE

並列領域 (KERNELS CONSTRUCT)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  #pragma acc kernels
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
}
```

- Parallels と Kernels
  - 並列領域を指示
- Parallels
  - 並列実行スタート
- Kernels
  - 複数のカーネル

GPU TECHNOLOGY CONFERENCE

並列領域 (KERNELS CONSTRUCT)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }
```

- Parallels と Kernels
  - 並列領域を指示
- Parallels
  - 並列走行の開始
- Kernels

```
$ pgcc -Minfo=acc -acc jacobi.c
jacobi:
     60, Loop carried scalar dependence for 'error' at line 64
         ...
         Accelerator scalar kernel generated
     61, Loop carried scalar dependence for 'error' at line 64
         ...
         Accelerator scalar kernel generated
```

GPU TECHNOLOGY CONFERENCE

リダクション (REDUCTION CLAUSE)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  #pragma acc kernels
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
}
```

- 演算の種類

+	和
*	積
Max	最大
Min	最小
|	ビット和
&	ビット積
^	XOR
||	論理和
&&	論理積

GPU TECHNOLOGY CONFERENCE

リダクション (REDUCTION CLAUSE)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
```

演算の種類	
+	和
*	積
Max	最大

```
$ pgcc -Minfo=acc -acc jacobi.c
jacobi:
     59, Generating present_or_copyout(Anew[1:4094][1:4094])
         Generating present_or_copyin(A[:][:])
         Generating Tesla code
     61, Loop is parallelizable
     63, Loop is parallelizable
         Accelerator kernel generated
         61, #pragma acc loop gang /* blockIdx.y */
         63, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Max reduction generated for error
```

GPU TECHNOLOGY CONFERENCE

データ転送方法 (DATA CLAUSE)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
```

```
$ pgcc -Minfo=acc -acc jacobi.c
jacobi:
     59, Generating present_or_copyout(Anew[1:4094][1:4094])
         Generating present_or_copyin(A[:][:])
         Generating Tesla code
     61, Loop is parallelizable
     63, Loop is parallelizable
         Accelerator kernel generated
         61, #pragma acc loop gang /* blockIdx.y */
         63, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Max reduction generated for error
```

GPU TECHNOLOGY CONFERENCE

データ転送方法 (DATA CLAUSE)

```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels \
      pcopyout(Anew[1:N-2][1:M-2]) pcopyin(A[0:N][0:M])
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  #pragma acc kernels \
      pcopyout(A[1:N-2][1:M-2]) pcopyin(Anew[1:N-2][1:M-2])
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
```

- copyin (Host→GPU)
- copyout (Host←GPU)
- copy
- create
- present

- pcopyin
- pcopyout
- pcopy
- pcreate

GPU TECHNOLOGY CONFERENCE

データ転送方法 (DATA CLAUSE)

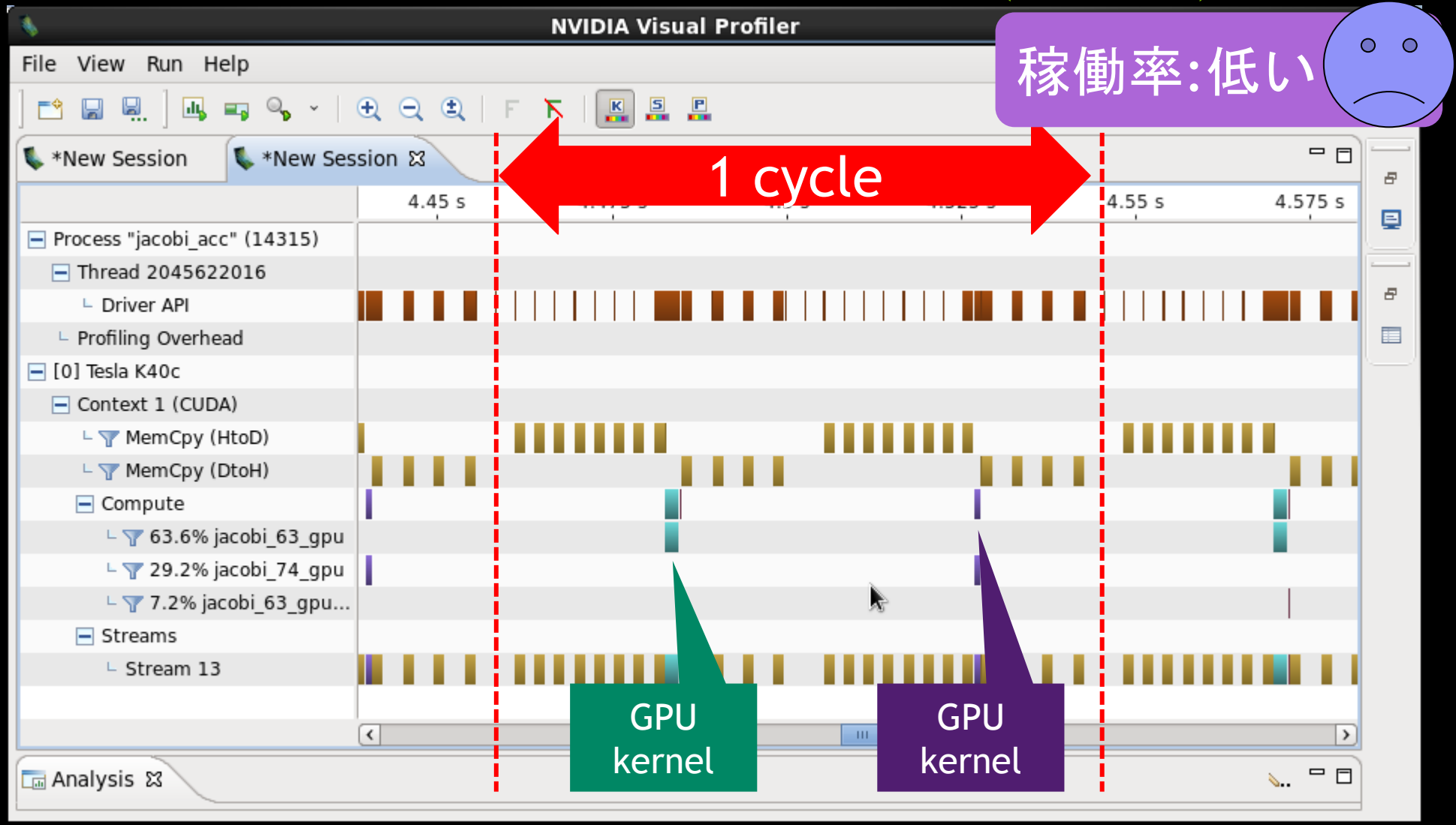
```
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels \
      pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  #pragma acc kernels \
      pcopy(A[:][:]) pcopyin(Anew[:][:])
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
```

- copyin (Host→GPU)
- copyout (Host←GPU)
- copy
- create
- present

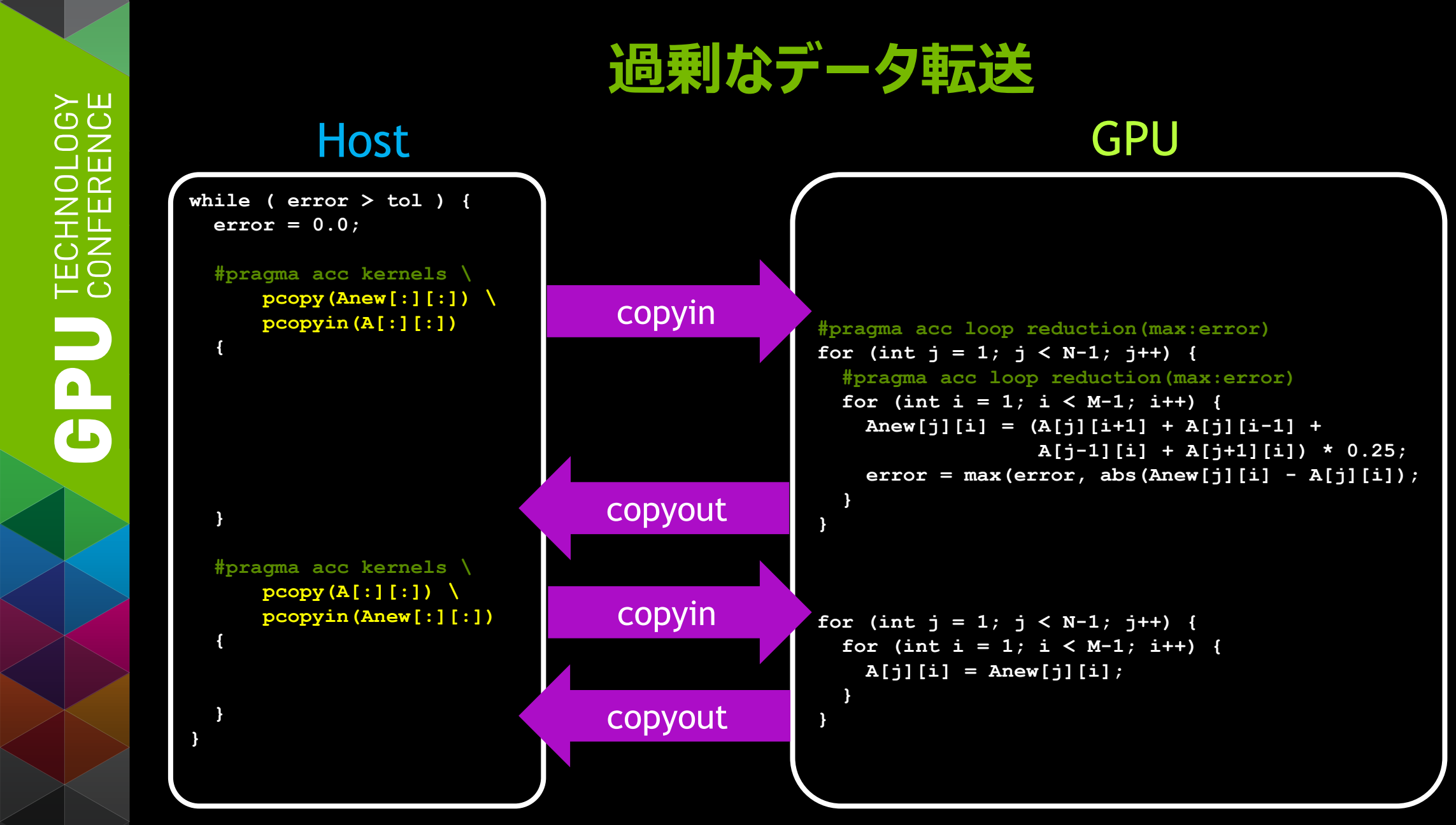
- pcopyin
- pcopyout
- pcopy
- pcreate

GPU TECHNOLOGY CONFERENCE

データ転送がボトルネック (NVVP)



過剰なデータ転送


Host
GPU
while (error > tol) {
error = 0.0;
#pragma acc kernels \
pcopy(Anew[:][:]) \
pcopyin(A[:][:])
{
}
#pragma acc kernels \
pcopy(A[:][:]) \
pcopyin(Anew[:][:])
{
}
}
copyin
copyout
copyin
copyout
#pragma acc loop reduction(max:error)
for (int j = 1; j < N-1; j++) {
#pragma acc loop reduction(max:error)
for (int i = 1; i < M-1; i++) {
Anew[j][i] = (A[j][i+1] + A[j][i-1] +
A[j-1][i] + A[j+1][i]) * 0.25;
error = max(error, abs(Anew[j][i] - A[j][i]);
}
}
for (int j = 1; j < N-1; j++) {
for (int i = 1; i < M-1; i++) {
A[j][i] = Anew[j][i];
}
}

データ領域 (DATA CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  #pragma acc kernels pcopy(A[:][:]) pcopyin(Anew[:][:])
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      A[j][i] = Anew[j][i];
    }
  }
}
```

- copyin (CPU→GPU)
- copyout (CPU←GPU)
- copy
- create
- present

- pcopyin
- pcopyout
- pcopy
- pcreate

GPU TECHNOLOGY CONFERENCE

適正なデータ転送

GPU TECHNOLOGY CONFERENCE

Host

```
#pragma acc data \
    pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels \
      pcopy(Anew[:][:]) \
      pcopyin(A[:][:])
  {









  }


  #pragma acc kernels \
      pcopy(A[:][:]) \
      pcopyin(Anew[:][:])
  {

  }
}
```

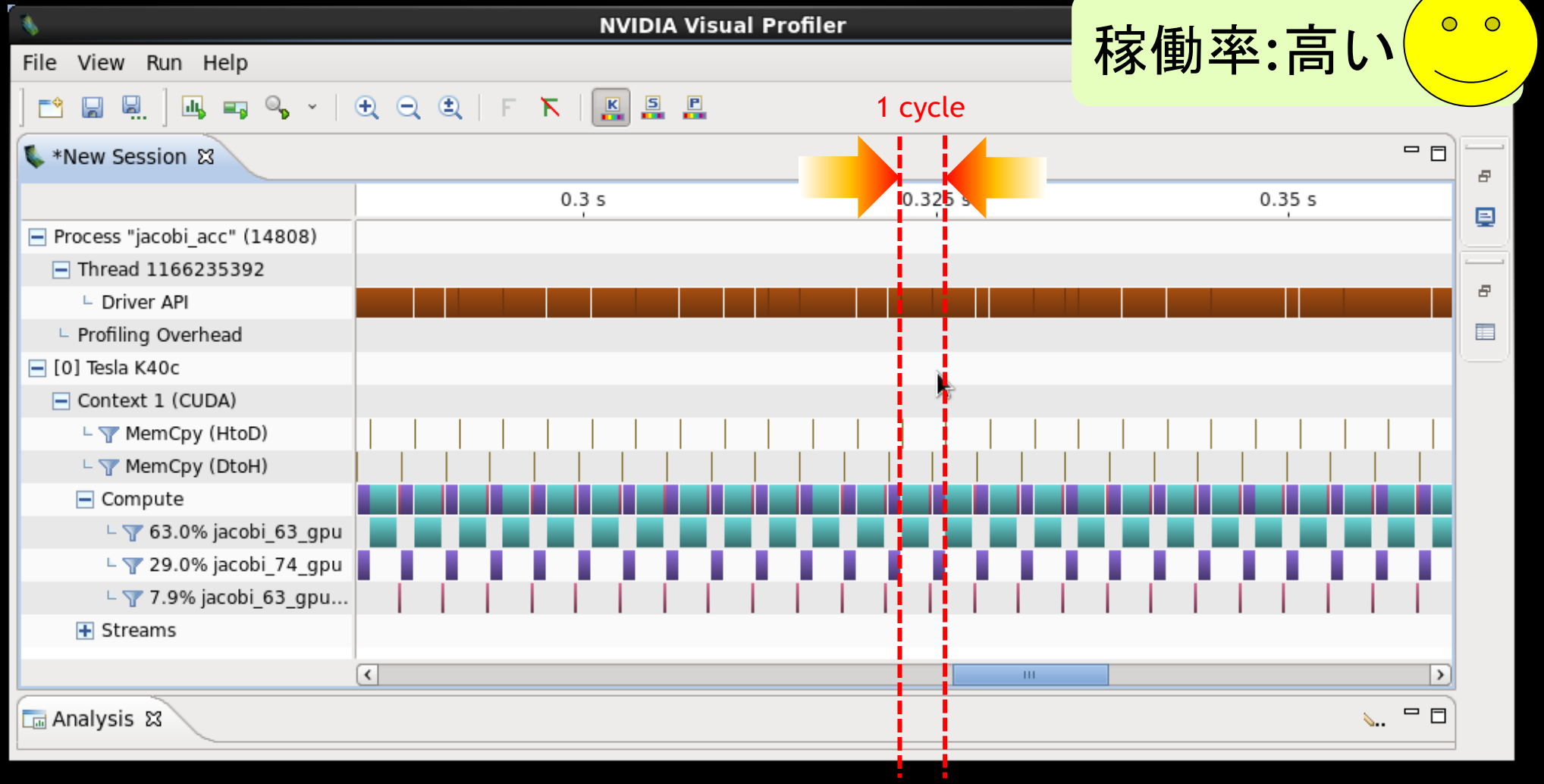
copyin

GPU

```
#pragma acc loop reduction(max:error)
for (int j = 1; j < N-1; j++) {
  #pragma acc loop reduction(max:error)
  for (int i = 1; i < M-1; i++) {
    Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                  A[j-1][i] + A[j+1][i]) * 0.25;
    error = max(error, abs(Anew[j][i] - A[j][i]));
  }
}



for (int j = 1; j < N-1; j++) {
  for (int i = 1; i < M-1; i++) {
    A[j][i] = Anew[j][i];
  }
}
```

copyout

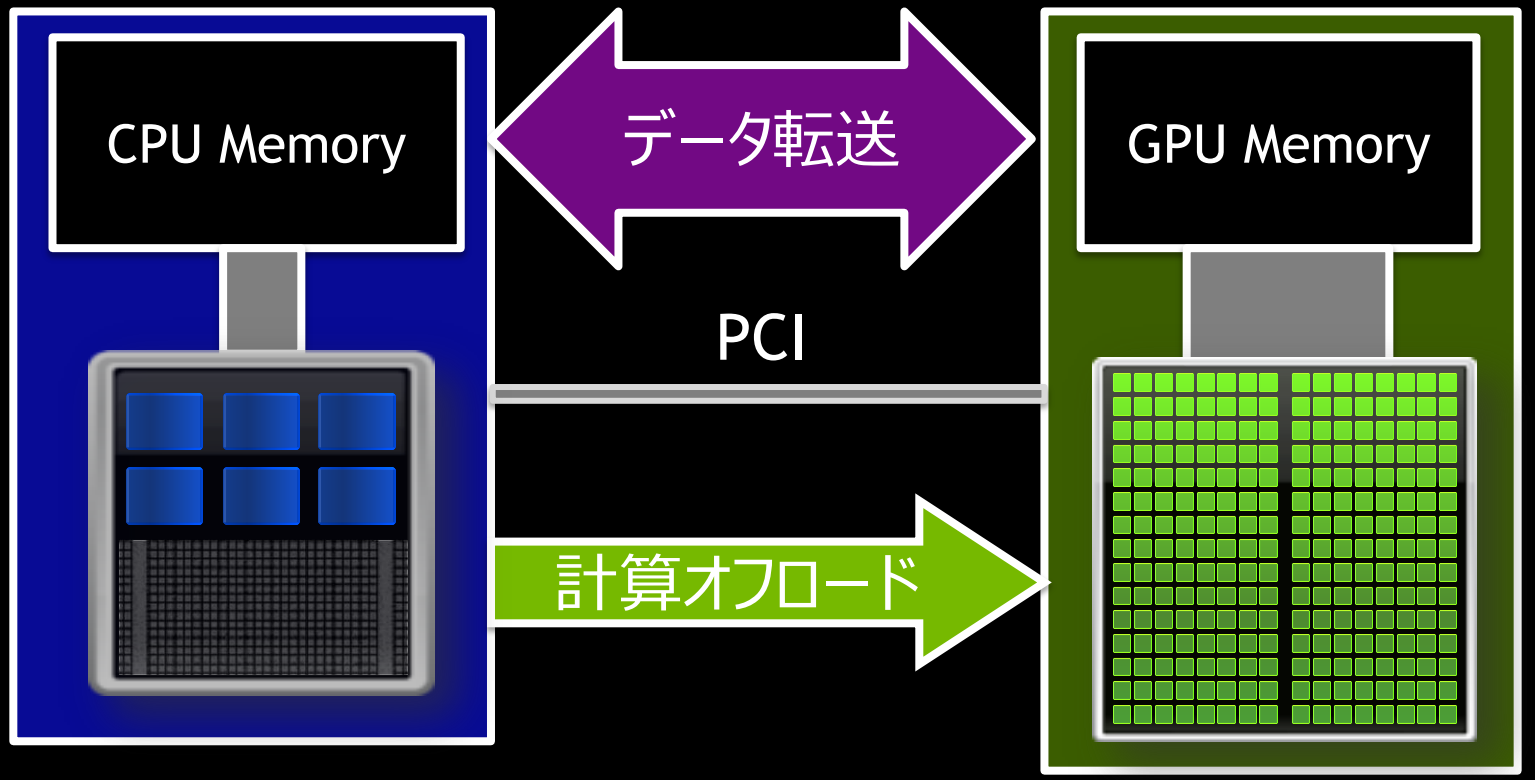
データ転送の削減 (NVVP)



稼働率:高い

GPU TECHNOLOGY CONFERENCE

2つの処理



計算オフロード、データ転送、両方を考慮する必要がある

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

カーネルチューニング

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  ...

}
```

GPU TECHNOLOGY CONFERENCE

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error)
```

- Gang
- Worker
- Vector … SIMD幅

```
$ pgcc -Minfo=acc -acc jacobi.c
jacobi:
     59, Generating present_or_copyout(Anew[1:4094][1:4094])
         Generating present_or_copyin(A[:][:])
         Generating Tesla code
     61, Loop is parallelizable
     63, Loop is parallelizable
         Accelerator kernel generated
         61, #pragma acc loop gang /* blockIdx.y */
         63, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Max reduction generated for error
```

GPU TECHNOLOGY CONFERENCE

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error) gang vector(1)
  for (int j = 1; j < N-1; j++) {
    #pragma acc loop reduction(max:error) gang vector(128)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }

  ...

}
```

- Gang
- Worker
- Vector … SIMD幅

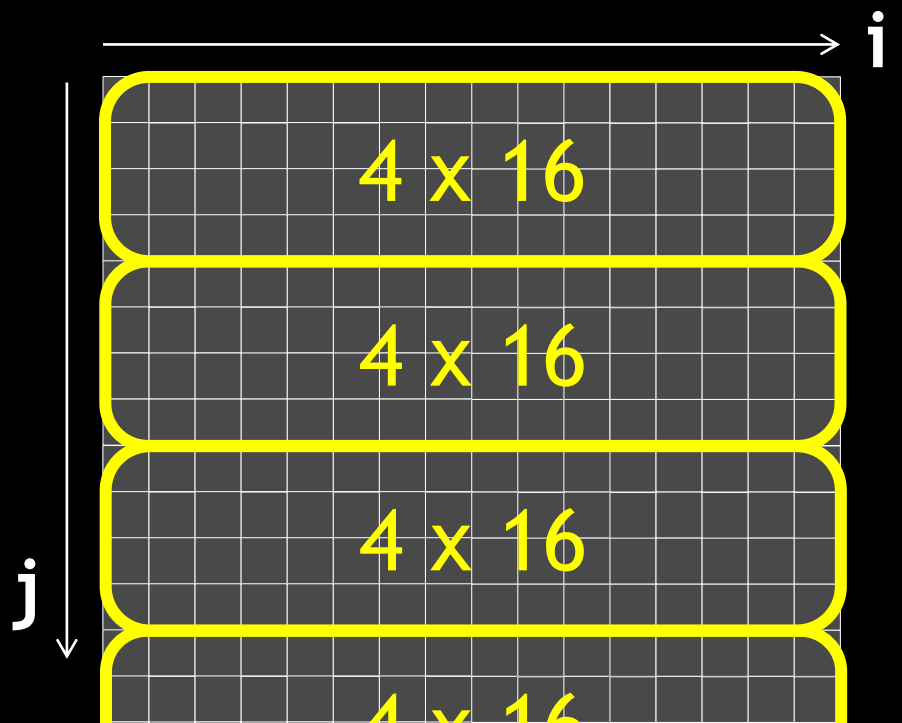
- Collapse
- Independent
- Seq
- Cache
- Tile

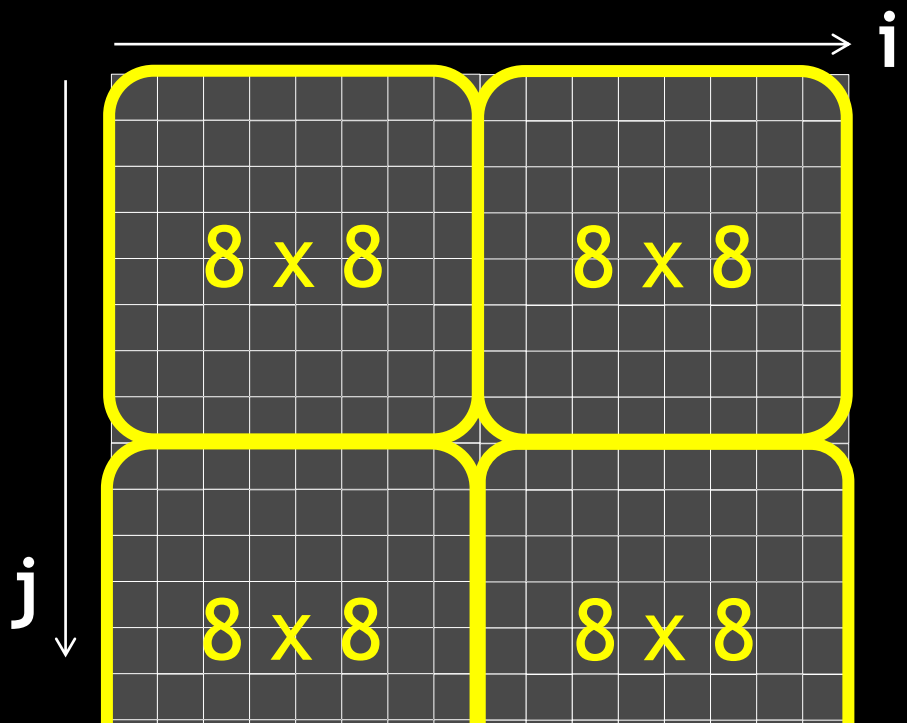
GPU TECHNOLOGY CONFERENCE

実行条件設定 (VECTOR CLAUSE)

```
#pragma acc loop gang vector(4)
for (j = 0; j < 16; j++) {
    #pragma accloop gang vector(16)
    for (i = 0; i < 16; i++) {
        ...
```

```
#pragma acc loop gang vector(8)
for (j = 1; j < 16; j++) {
    #pragma accloop gang vector(8)
    for (i = 0; i < 16; i++) {
        ...
```





GPU TECHNOLOGY CONFERENCE

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error) \
      collapse(2) gang vector(128)
  for (int j = 1; j < N-1; j++) {
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }
  ...
}
```

- Gang
- Worker
- Vector … SIMD幅

- Collapse
- Independent
- Seq
- Cache
- Tile
- …

GPU TECHNOLOGY CONFERENCE

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  error = 0.0;

  #pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
  #pragma acc loop reduction(max:error) independent
  for (int jj = 1; jj < NN-1; jj++) {
    int j = list_j[jj];
    #pragma acc loop reduction(max:error)
    for (int i = 1; i < M-1; i++) {
      Anew[j][i] = (A[j][i+1] + A[j][i-1] +
                    A[j-1][i] + A[j+1][i]) * 0.25;
      error = max(error, abs(Anew[j][i] - A[j][i]);
    }
  }
  ...
}
```

- Gang
- Worker
- Vector … SIMD幅

- Collapse
- Independent
- Seq
- Cache
- Tile
- …

カーネルチューニング (LOOP CONSTRUCT)

```
#pragma acc kernels pcopy(Anew[:][:]) pcopyin(A[:][:])
#pragma acc loop seq
for (int k = 3; k < NK-3; k++) {
  #pragma acc loop
  for (int j = 0; j < NJ; j++) {
    #pragma acc loop
    for (int i = 0; i < NI; i++) {
      Anew[k][j][i] = func(
          A[k-1][j][i], A[k-2][j][i], A[k-3][j][i],
          A[k+1][j][i], A[k+2][j][i], A[k+3][j][i], ...
      );
    }
  }
}
```

- Gang
- Worker
- Vector … SIMD幅

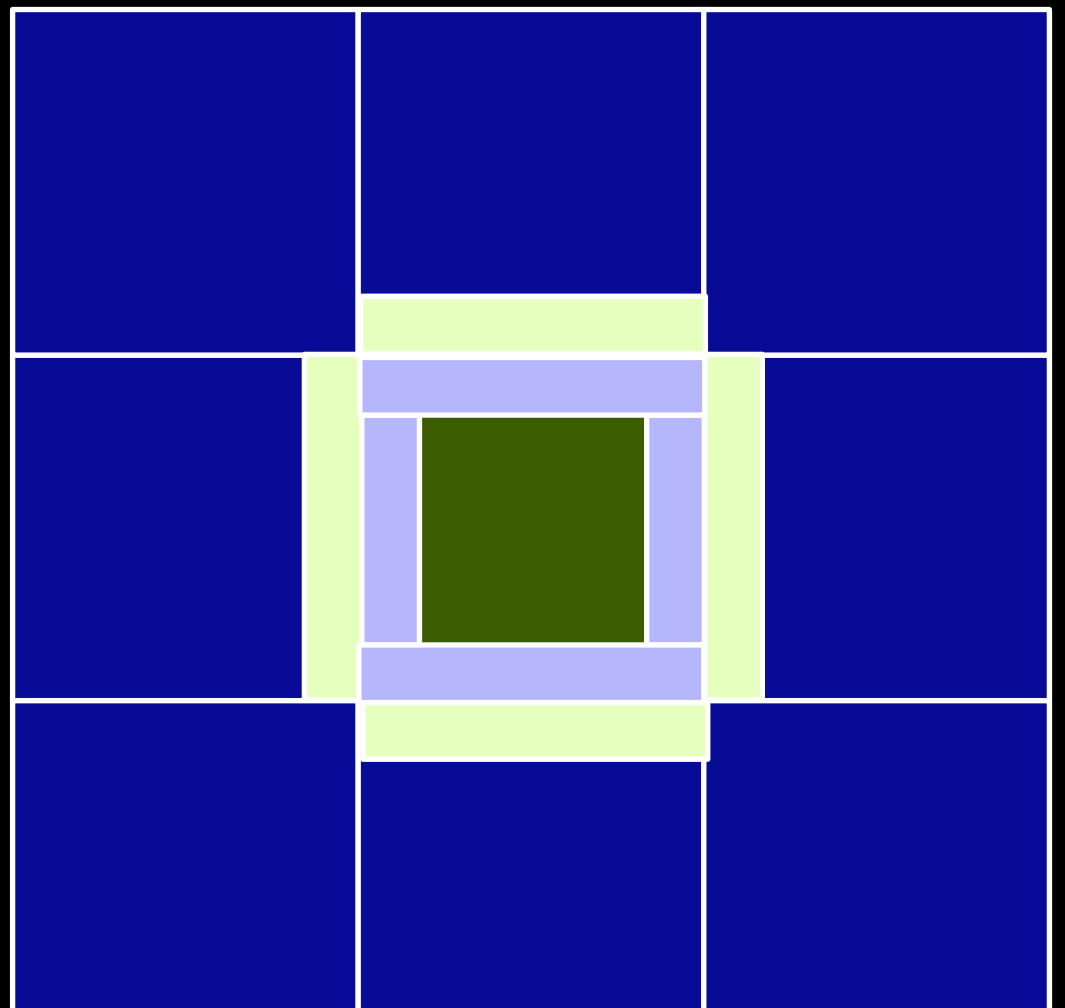
- Collapse
- Independent
- Seq
- Cache
- Tile
- …

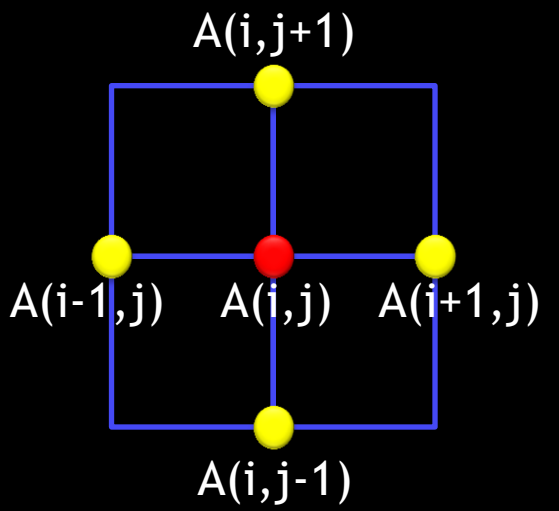
GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

MPIとは簡単に併用できるの?

MPI並列 (HALO EXCHANGE)





- ブロック分割
- 各プロセスは1ブロック担当
- 境界部(halo)のデータ交換

GPU TECHNOLOGY CONFERENCE

MPI JACOBI ITERATION

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew)
  update_A( A, Anew );
}
```

GPU TECHNOLOGY CONFERENCE

MPI JACOBI ITERATION

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {

  pack_data_at_boundary( send_buf, A, ... );

  exchange_data_by_MPI( recv_buf, send_buf, ... );


  unpack_data_to_halo( A, recv_buf, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew)
  update_A( A, Anew );
}
```

GPU
1.送信データの梱包

MPI
2.データの交換

GPU
3.受信データの開梱

GPU TECHNOLOGY CONFERENCE

MPI JACOBI ITERATION

```
#pragma acc data pcopy(A) create(Anew)
while ( error > tol ) {
  #pragma acc kernels pcopyin(A) pcopyout(send_buf)
  pack_data_at_boundary( send_buf, A, ... );

  exchange_data_by_MPI( recv_buf, send_buf, ... );

  #pragma acc kernels pcopy(A) pcopyin(recv_buf)
  unpack_data_to_halo( A, recv_buf, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew)
  update_A( A, Anew );
}
```

GPU

1. GPU上でデータを送信バッファに梱包し、Hostに転送

MPI

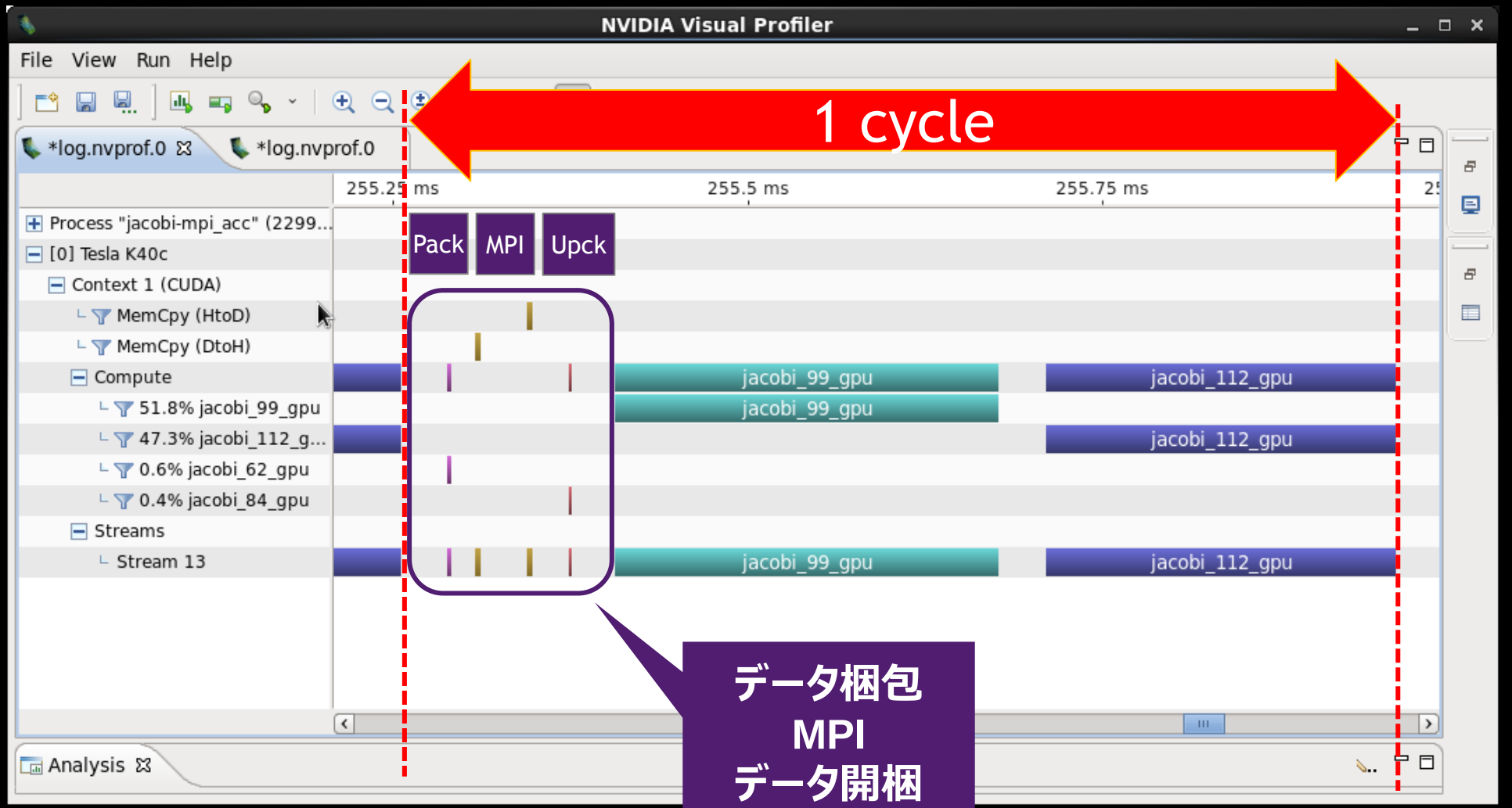
2. 隣接プロセスとデータ交換

GPU

3. GPUに転送、GPU上で受信バッファのデータを開梱

GPU TECHNOLOGY CONFERENCE

MPI JACOBI ITERATION (NVVP)



オーバーラップ (ASYNC/WAIT CLAUSE)

```
while ( error > tol ) {
  #pragma acc kernels pcopyin(A) pcopyout(send_buf)
  pack_data_at_boundary( send_buf, A, ... );

  exchange_data_by_MPI( recv_buf, send_buf, ... );

  #pragma acc kernels pcopy(A) pcopyin(recv_buf)
  unpack_data_to_halo( A, recv_buf, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew)
  update_A( A, Anew );
}
```

GPU TECHNOLOGY CONFERENCE

オーバーラップ (ASYNC/WAIT CLAUSE)

```
while ( error > tol ) {
  #pragma acc kernels pcopyin(A) pcopyout(send_buf)
  pack_data_at_boundary( send_buf, A, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A_inside( Anew, A, ... );

  exchange_data_by_MPI( recv_buf, send_buf, ... );

  #pragma acc kernels pcopy(A) pcopyin(recv_buf)
  unpack_data_to_halo( A, recv_buf, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A)
  calc_new_A_at_boundary( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew)
  update_A( A, Anew );
}
```

内部

境界部

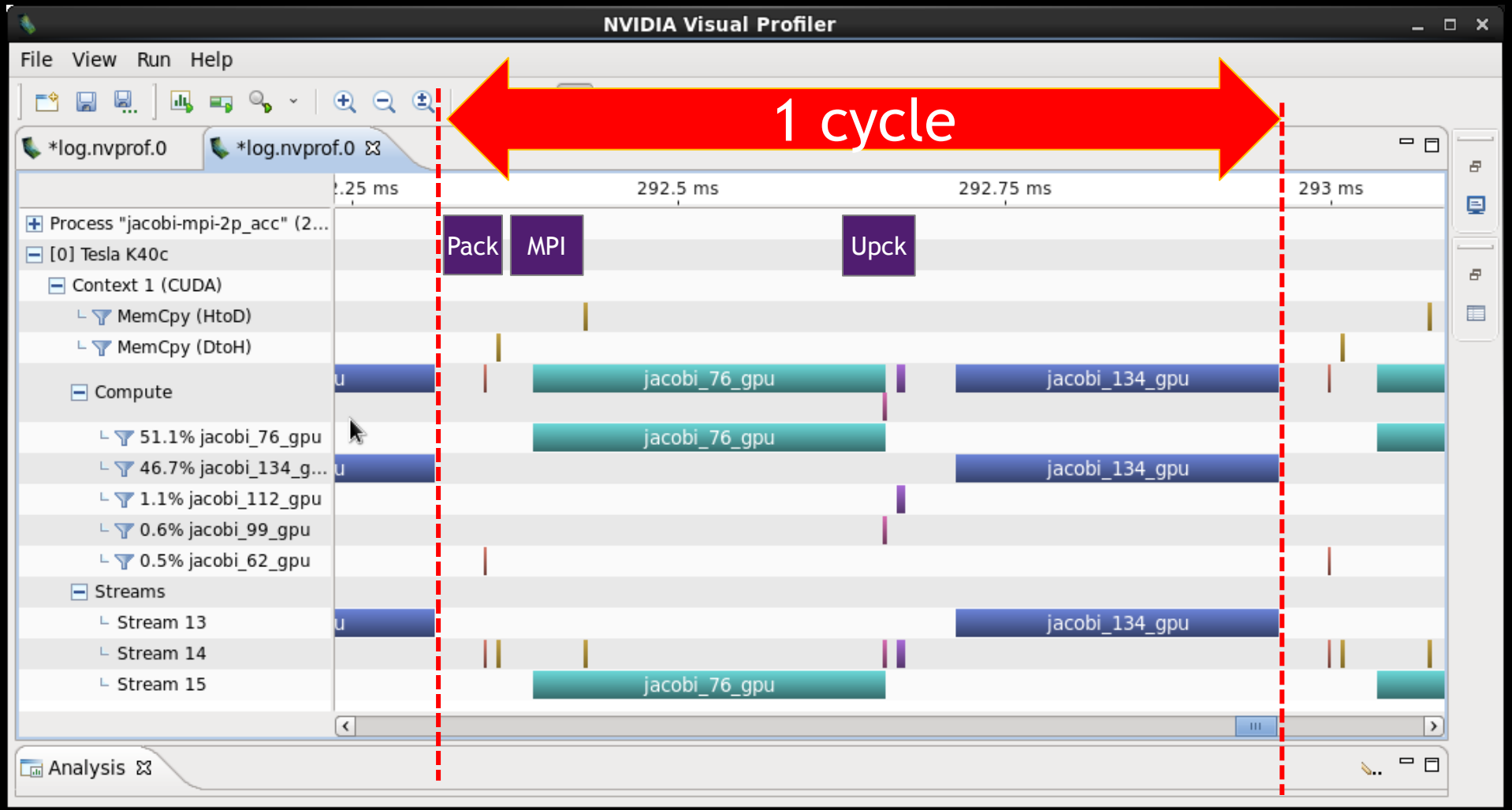
GPU TECHNOLOGY CONFERENCE

オーバーラップ (ASYNC/WAIT CLAUSE)

```
while ( error > tol ) {
  #pragma acc kernels pcopyin(A) pcopyout(send_buf) async(2)
  pack_data_at_boundary( send_buf, A, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A) async(1)
  calc_new_A_inside( Anew, A, ... );

  #pragma acc wait(2)
  exchange_data_by_MPI( recv_buf, send_buf, ... );

  #pragma acc kernels pcopy(A) pcopyin(recv_buf) async(2)
  unpack_data_to_halo( A, recv_buf, ... );

  #pragma acc kernels pcopy(Anew) pcopyin(A) async(2)
  calc_new_A_at_boundary( Anew, A, ... );

  #pragma acc kernels pcopy(A) pcopyin(Anew) wait(1,2)
  update_A( A, Anew );
}
```

GPU TECHNOLOGY CONFERENCE

オーバーラップ(NVVP)

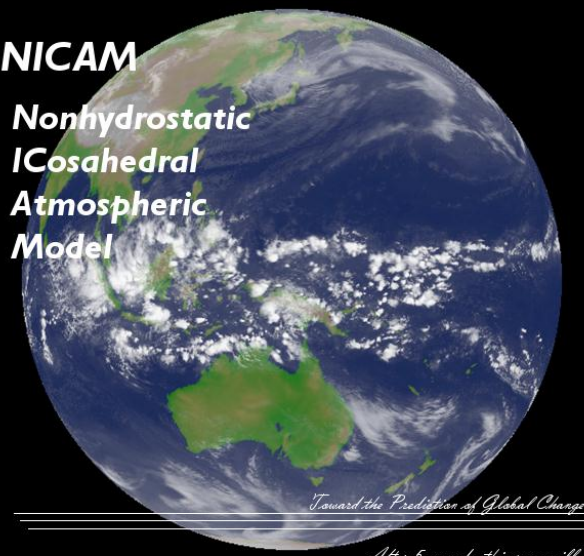


GPU TECHNOLOGY CONFERENCE

OPENACCって、
実際に使われているの?

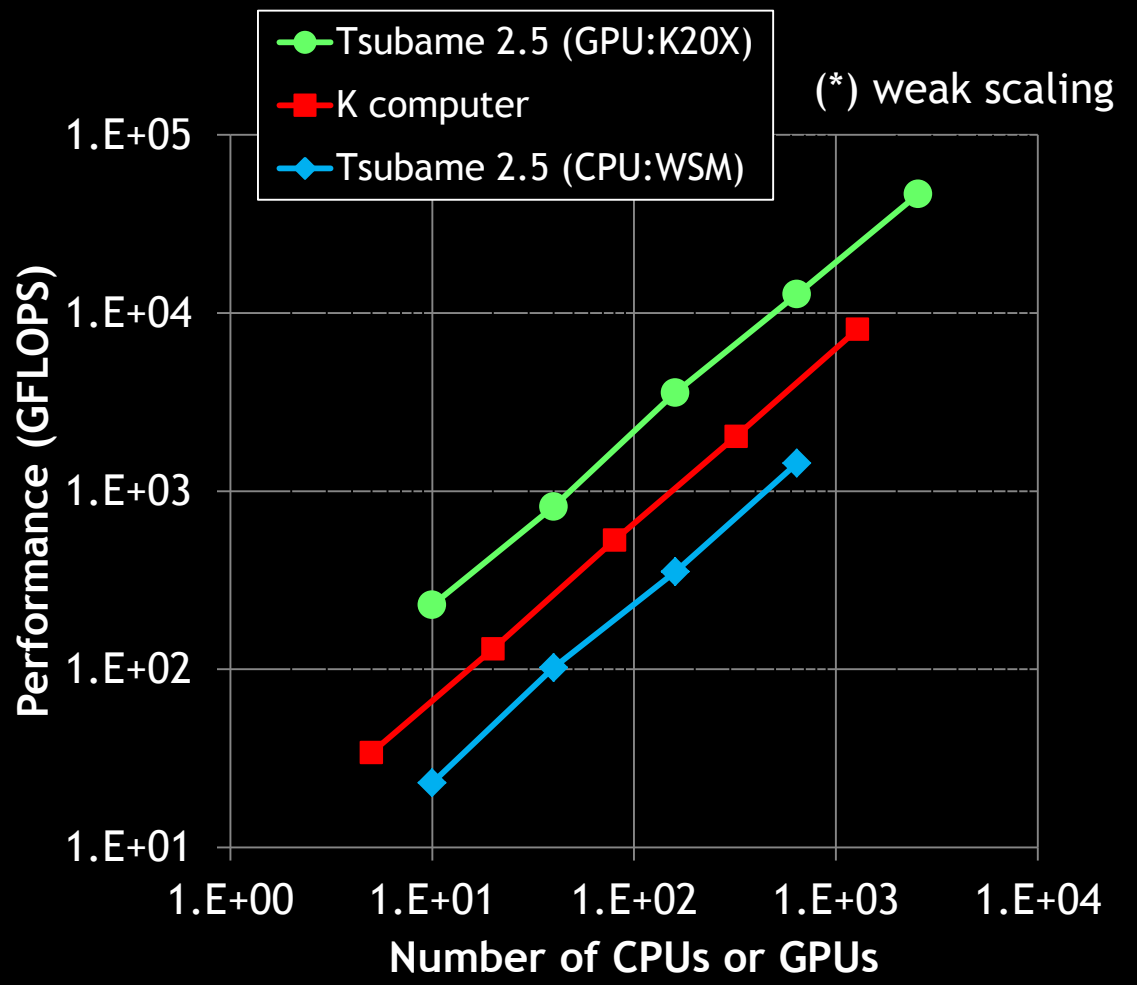
NICAM

- 気象・気候モデル by 理研AICS/東大
  - 膨大なコード (数十万行)
  - ホットスポットがない (~~パレートの法則~~)
- 特性の異なる2種類の処理
  - 力学系 … メモリバンド幅ネック
  - 物理系 … 演算ネック



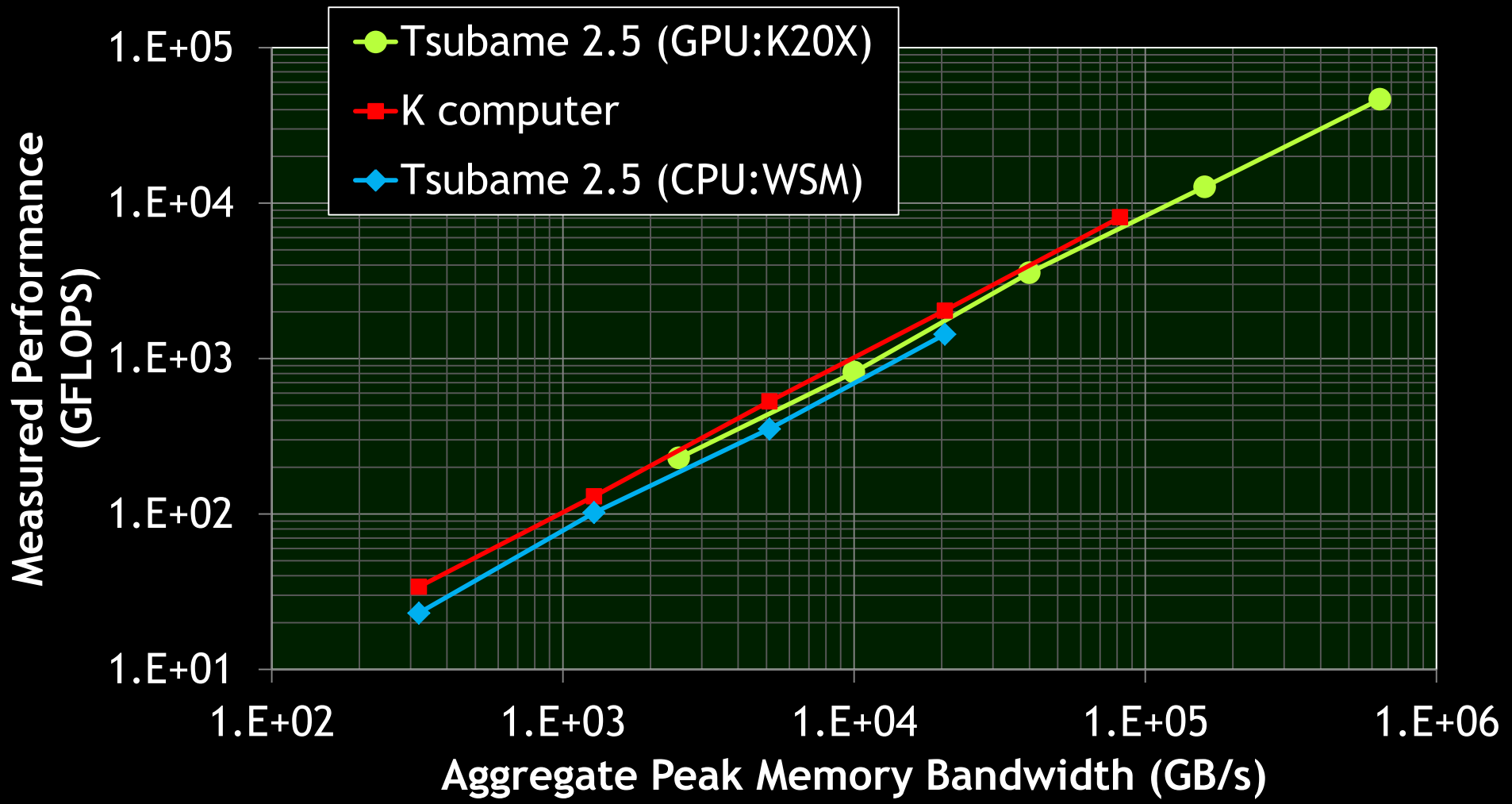
NICAM: 力学系(NICAM-DC)

- OpenACCによるGPU化
  - 主要サブルーチンは、全てGPU上で動作(50以上)
  - MPI対応済み
  - 2週間
- 良好なスケーラビリティ
  - Tsubame 2.5, 最大2560 GPUs
  - Scaling factor: 0.8



GPU TECHNOLOGY CONFERENCE

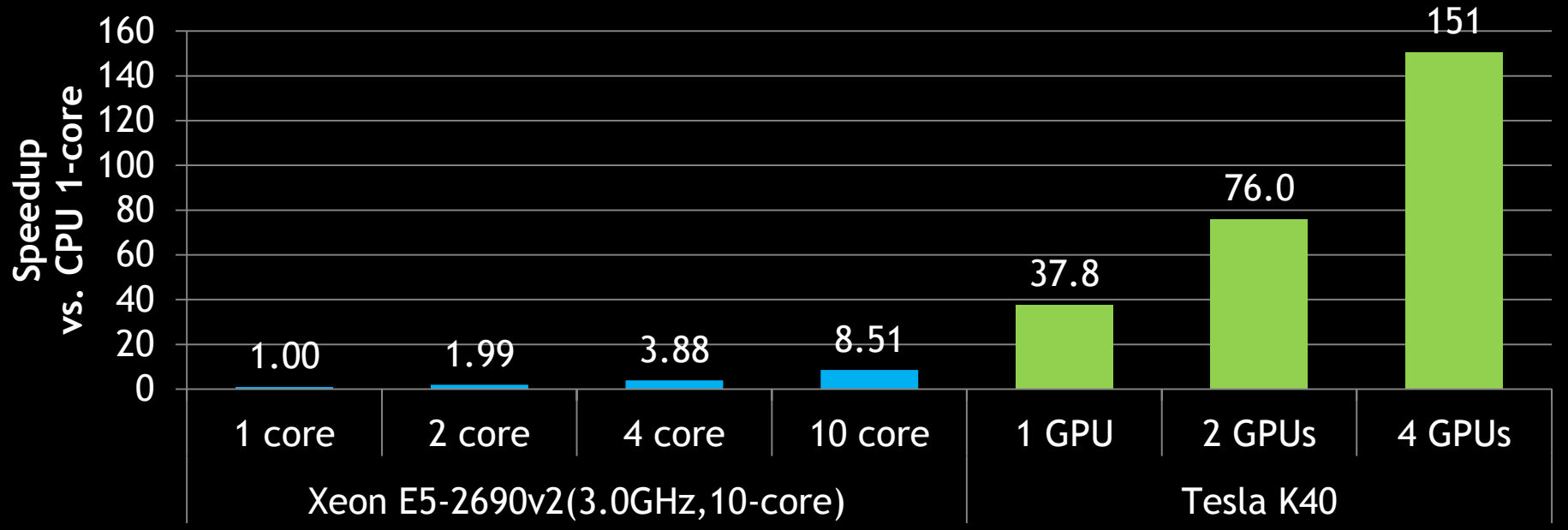
NICAM: 力学系(NICAM-DC)



GPU TECHNOLOGY CONFERENCE

NICAM: 物理系(SCALE-LES)

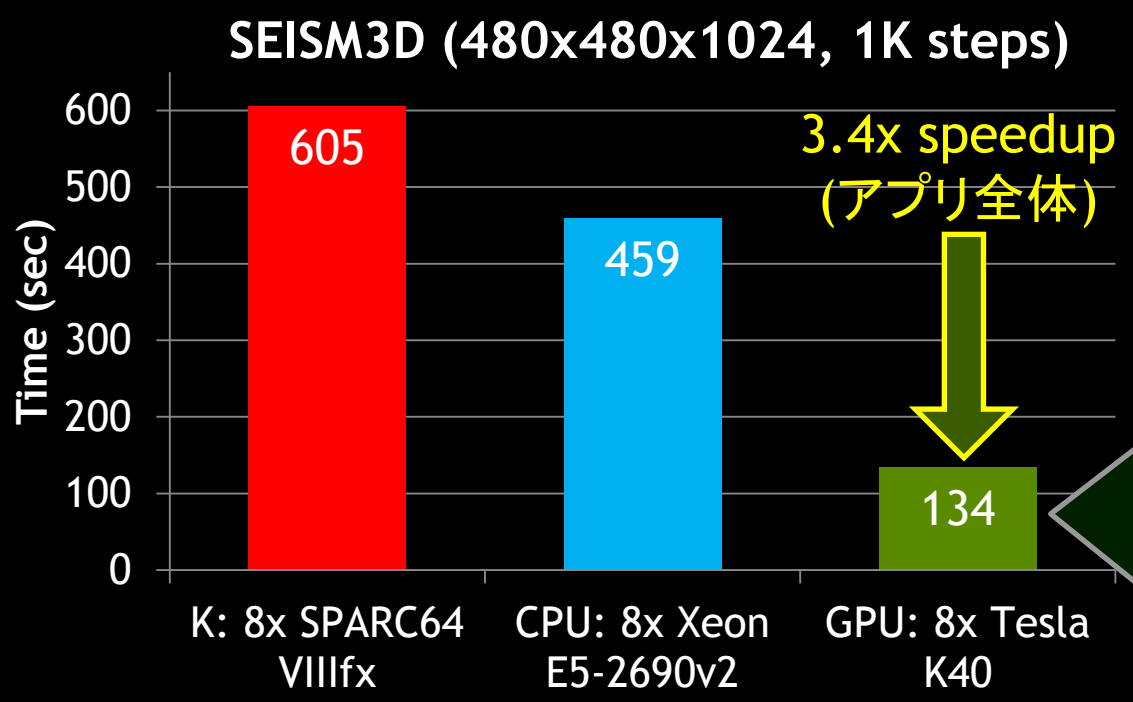
- Atmospheric radiation transfer
  - 物理系の中で、最も重い計算
  - OpenACCによるGPU対応、完了

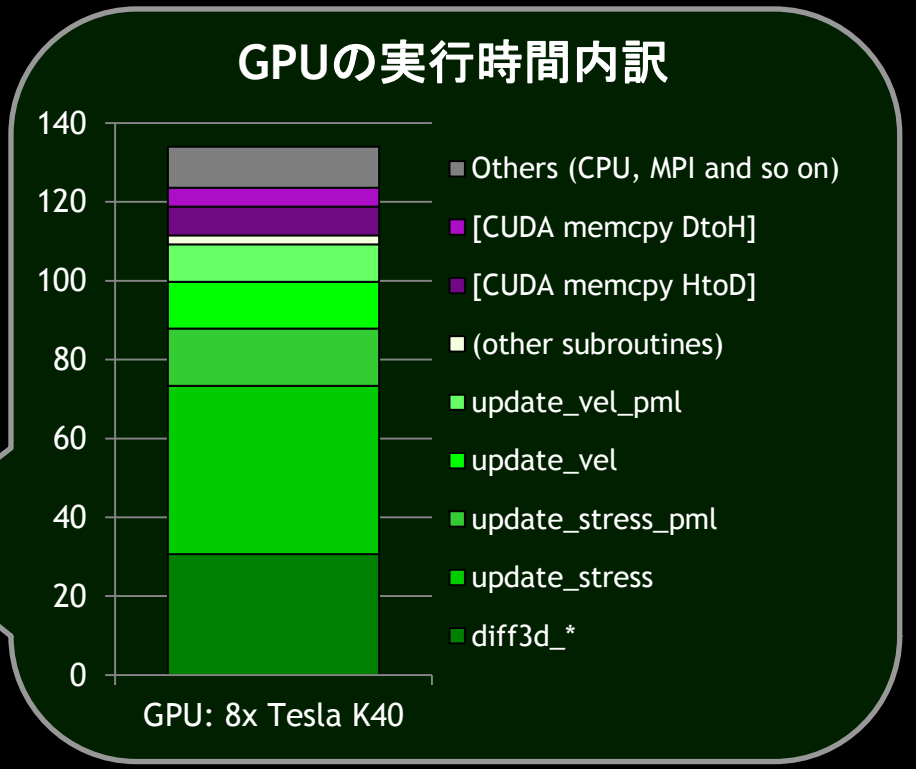


(*) PCIデータ転送時間込み, グリッドサイズ:1256x32x32

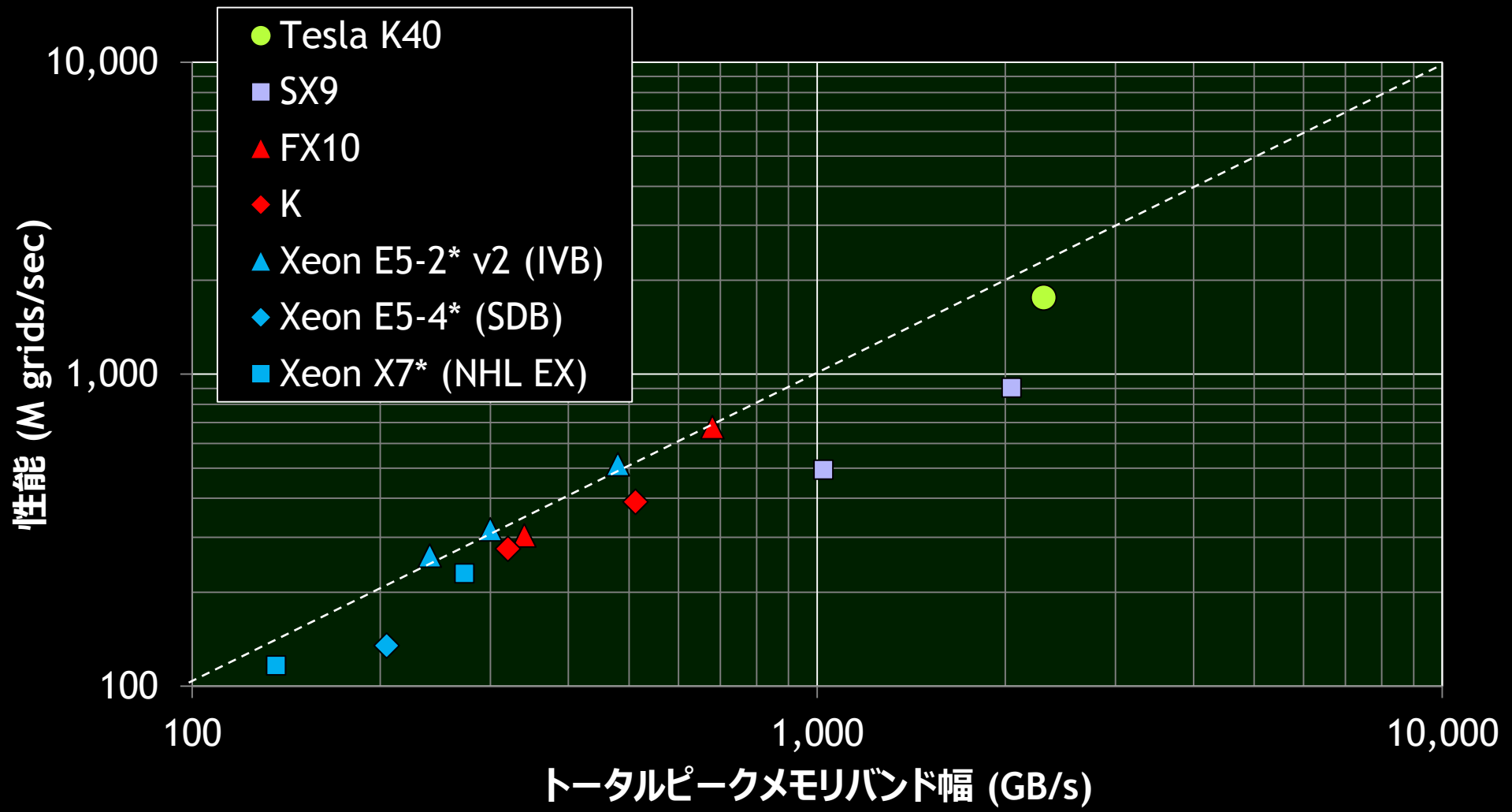
SEISM3D

- 地震シミュレーション　by 東大地震研(古村教授)
- 主要サブルーチンのGPU対応が完了
  - メモリバンド幅ネック、3次元モデル(2次元分割)、隣接プロセス間通信





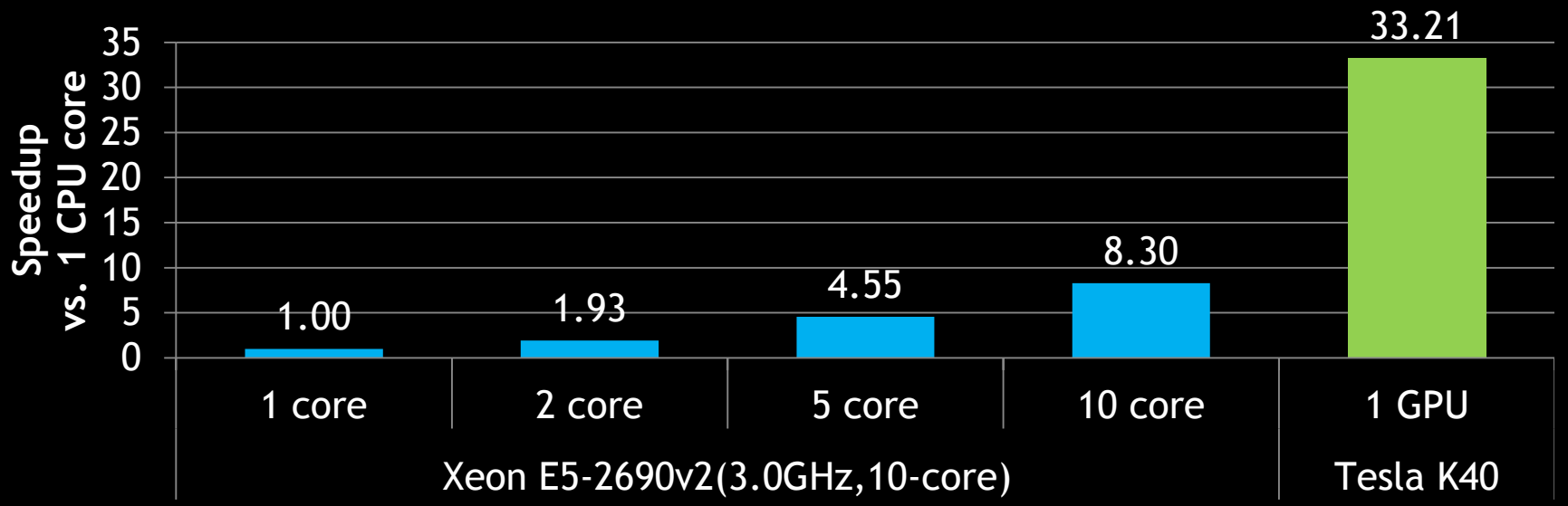
SEISM3D



GPU TECHNOLOGY CONFERENCE

FFR/BCM

- 次世代CFD　by 理研AICS/北大(坪倉准教授)
- MUSCL_bench:
  - MUSCLスキームに基づくFlux計算 (とても複雑な計算)
  - CFD計算の主要部分 (60-70%)
  - OpenACCによるGPU対応、完了



(*) PCIデータ転送時間込み、サイズ:80x32x32x32

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

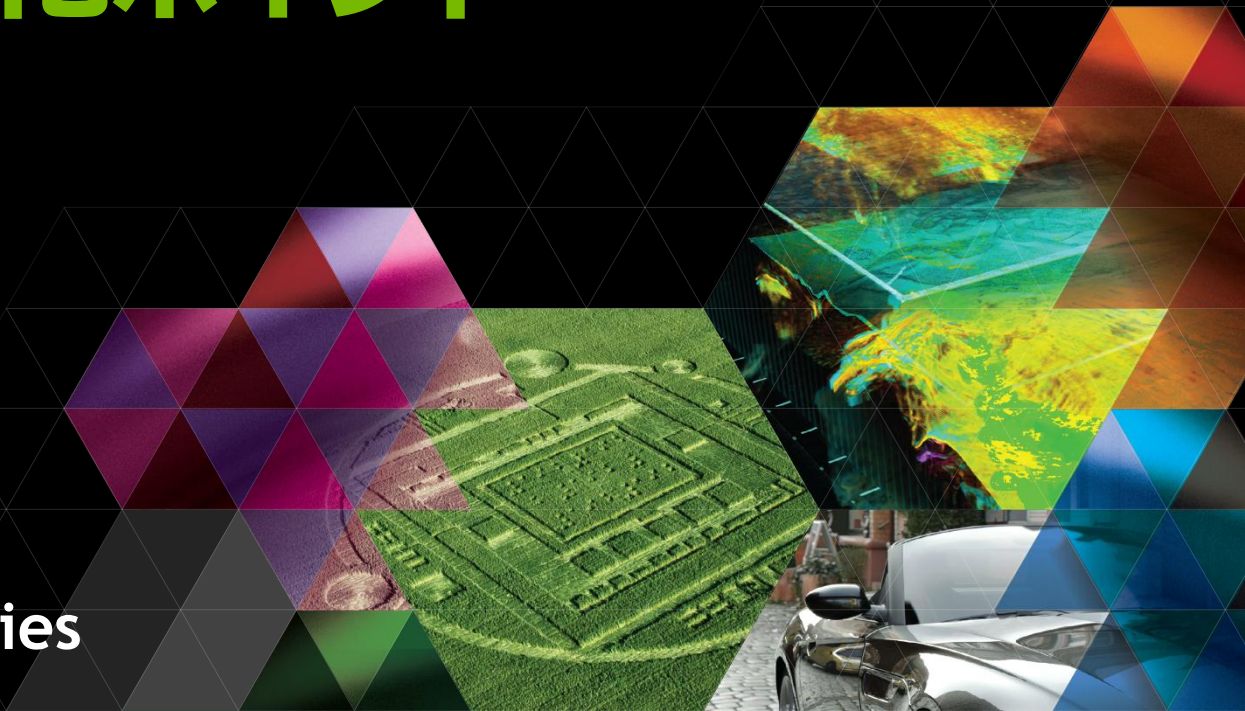
まとめ

- OpenACCの現状を紹介
- 簡単: 既存コードへのディレクティブ追加
- 強力: 少ない労力でGPU利用可能
- オープン: 採用事例の増加

GPU TECHNOLOGY CONFERENCE

NVIDIA

CUDA 6の強化ポイント



Akira Naruse
NVIDAI Developer Technologies

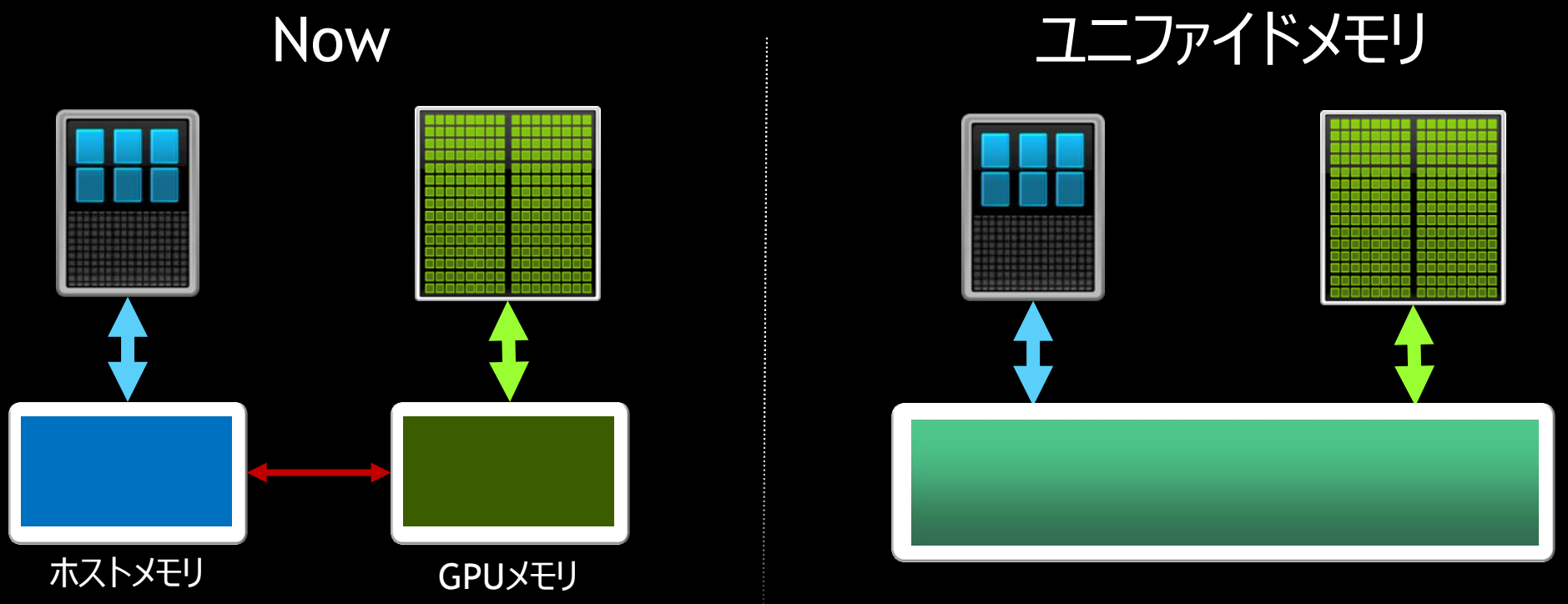
GPU TECHNOLOGY CONFERENCE

CUDA 6

ユニファイド・メモリ
XTライブラリ
ドロップイン・ライブラリ
GPUDirect RDMA
開発ツール

ユニファイドメモリ

開発者から見えるメモリモデル



GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

煩雑なメモリマネジメント

CPUコード

```
void sortfile(FILE *fp, int N) {
  char *data;

  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

GPUコード

```
void sortfile(FILE *fp, int N) {
  char *data
  char *d_data;
  data = (char *)malloc(N);
  cudaMalloc(&d_data, N);
  fread(data, 1, N, fp);
  cudaMemcpy(d_data, data, N, ..);
  qsort<<<...>>>(d_data,N,1,compare);
  cudaDeviceSynchronize();
  cudaMemcpy(data, d_data, N, ..);
  use_data(data);
  cudaFree(d_data);
  free(data);
}
```

メモリマネジメントを簡素化

CPUコード

```
void sortfile(FILE *fp, int N) {
  char *data;

  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

ユニファイドメモリ(CUDA6)

```
void sortfile(FILE *fp, int N) {
  char *data

  cudaMallocManaged(&d_data, N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(d_data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  cudaFree(data);
}
```

GPU TECHNOLOGY CONFERENCE

メモリマネジメントの統合(将来)

CPUコード

```
void sortfile(FILE *fp, int N) {
  char *data;

  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

将来?

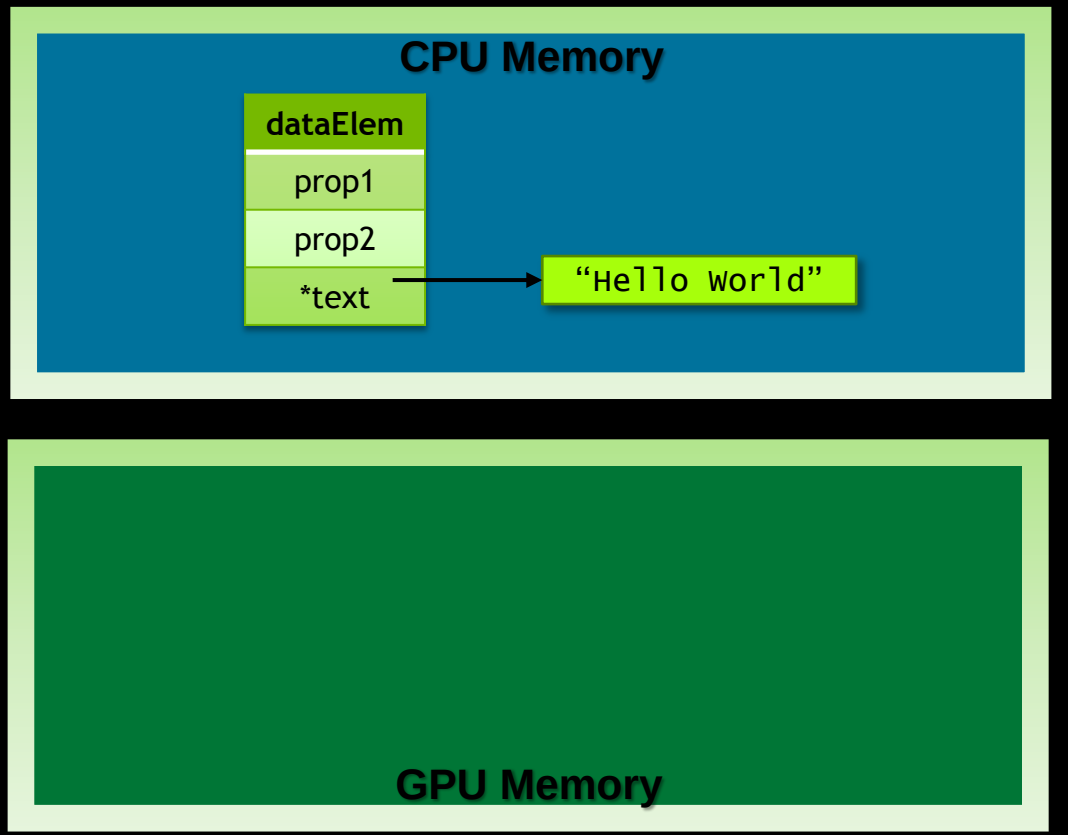
```
void sortfile(FILE *fp, int N) {
  char *data

  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(d_data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  free(data);
}
```

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

DEEP COPY

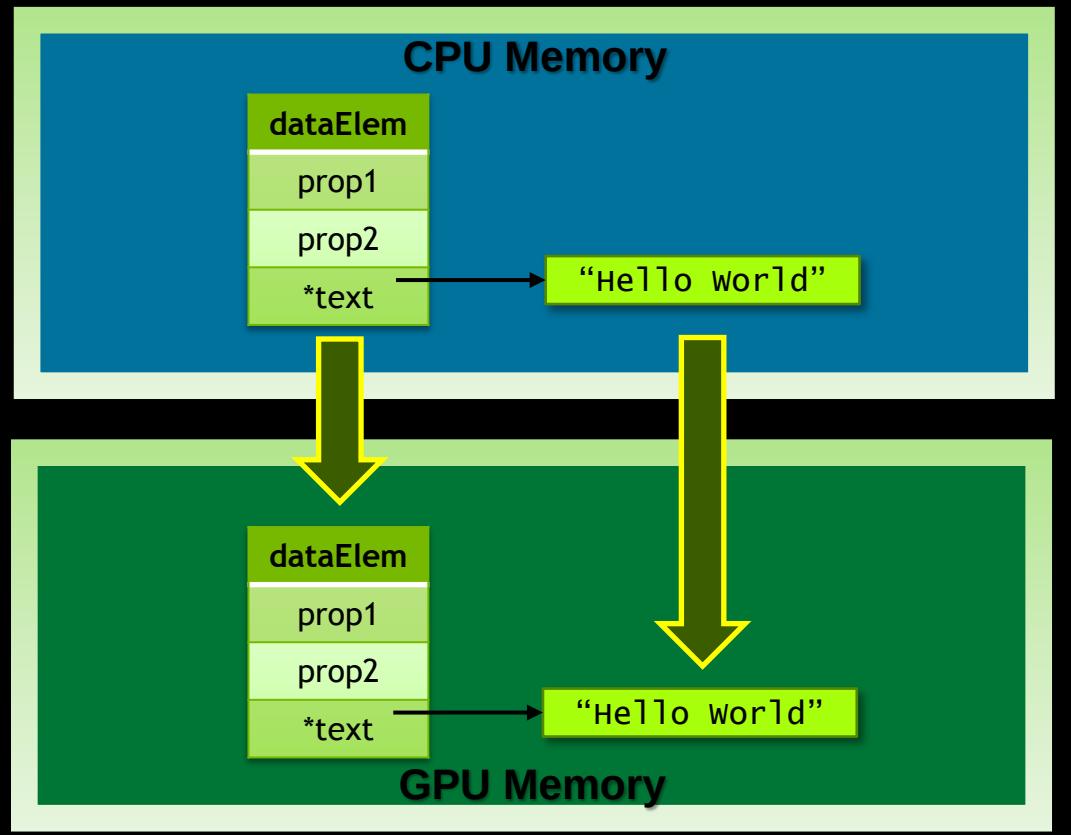
```
struct dataElem {
    int prop1;
    int prop2;
    char *text;
};
```



DEEP COPY

```
struct dataElem {
    int prop1;
    int prop2;
    char *text;
};
```

コピーが
2回必要

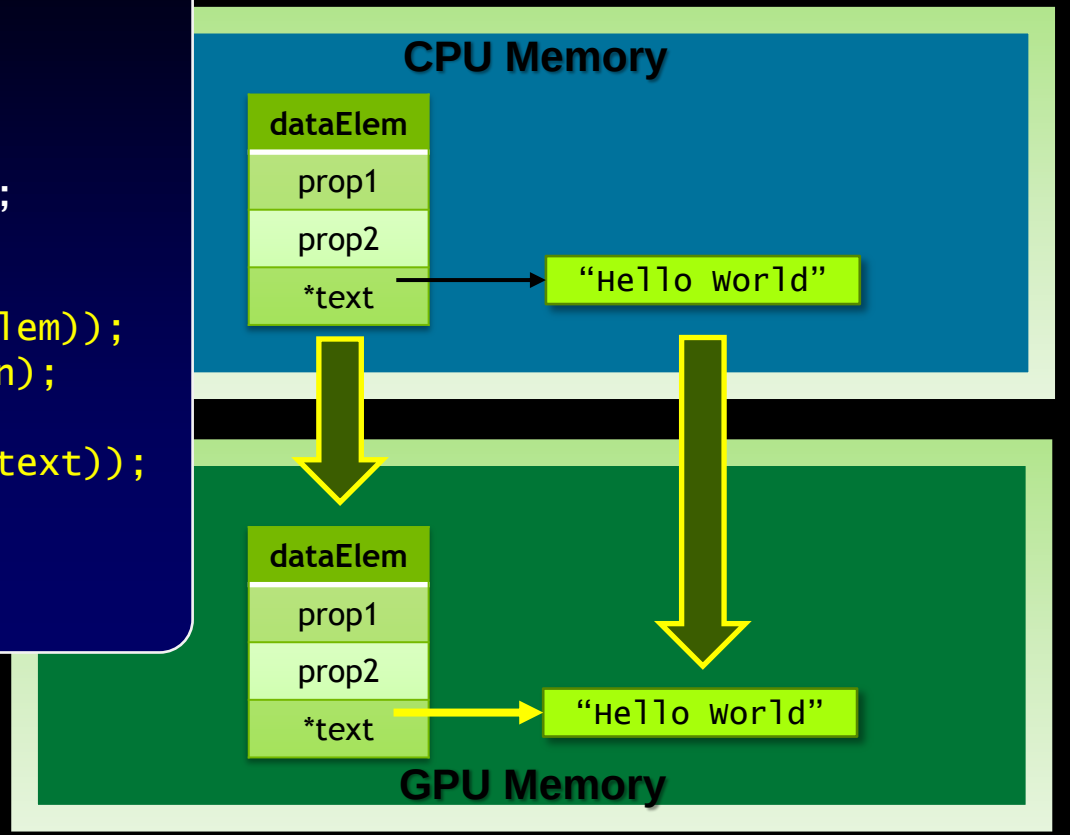


GPU TECHNOLOGY CONFERENCE

DEEP COPY

```
void launch(dataElem *elem) {
  dataElem *g_elem;
  char *g_text;

  int textlen = strlen(elem->text);

  cudaMalloc(&g_elem, sizeof(dataElem));
  cudaMalloc(&g_text, textlen);

  cudaMemcpy(g_elem, elem, sizeof(dataElem));
  cudaMemcpy(g_text, elem->text, textlen);
  cudaMemcpy(&(g_elem->text), &g_text,
                              sizeof(g_text));

  kernel<<< ... >>>(g_elem);
}
```

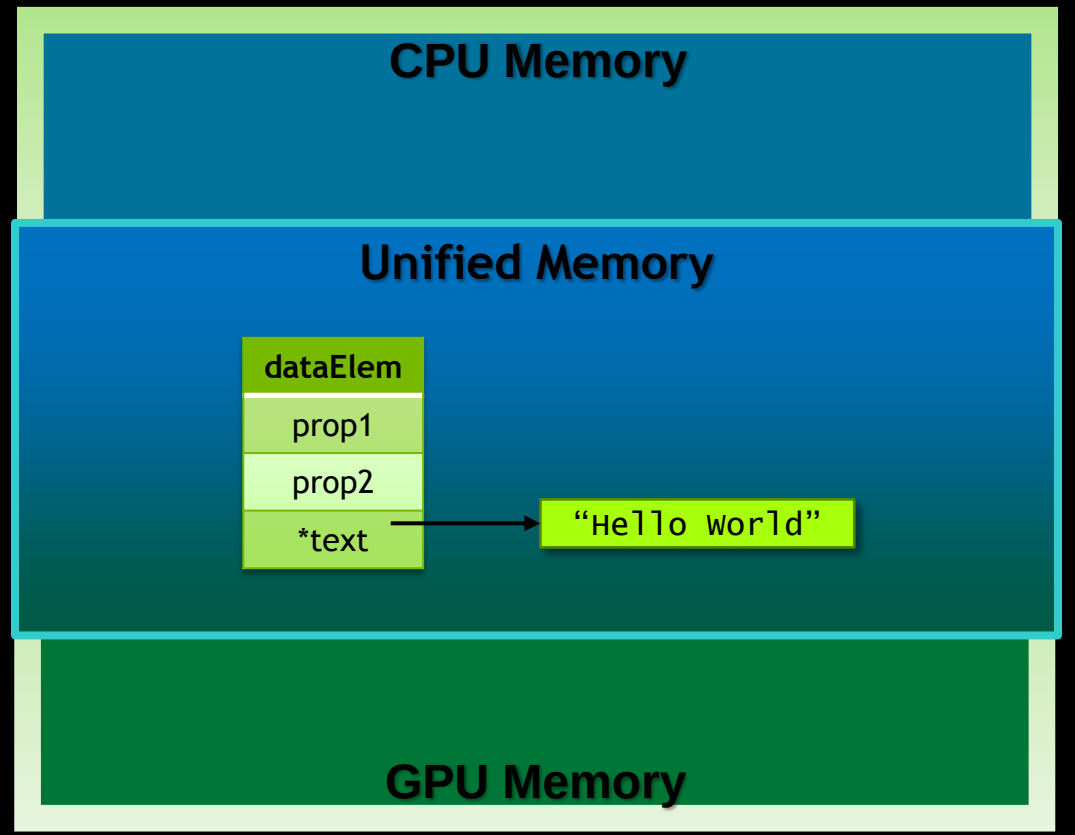
実際は
3回必要



GPU TECHNOLOGY CONFERENCE

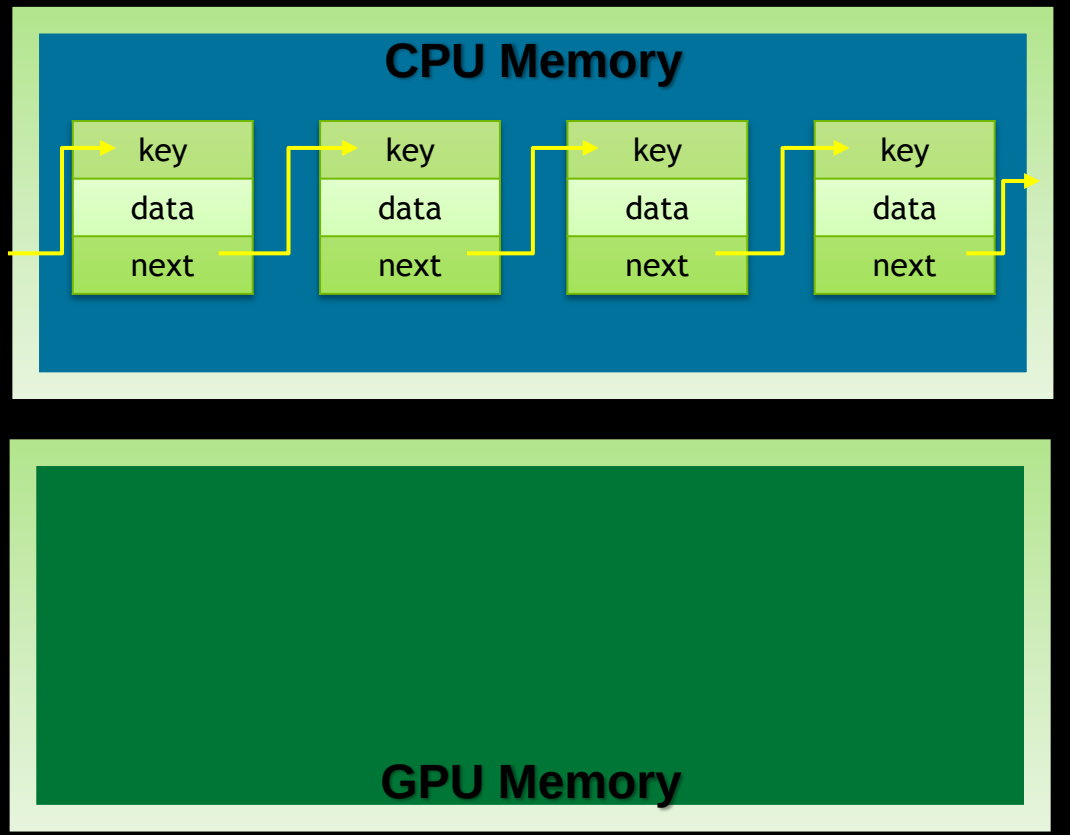
DEEP COPY (ユニファイドメモリ)

```
void launch(dataElem *elem) {
    kernel<<< ... >>>(elem);
}
```



GPU TECHNOLOGY CONFERENCE

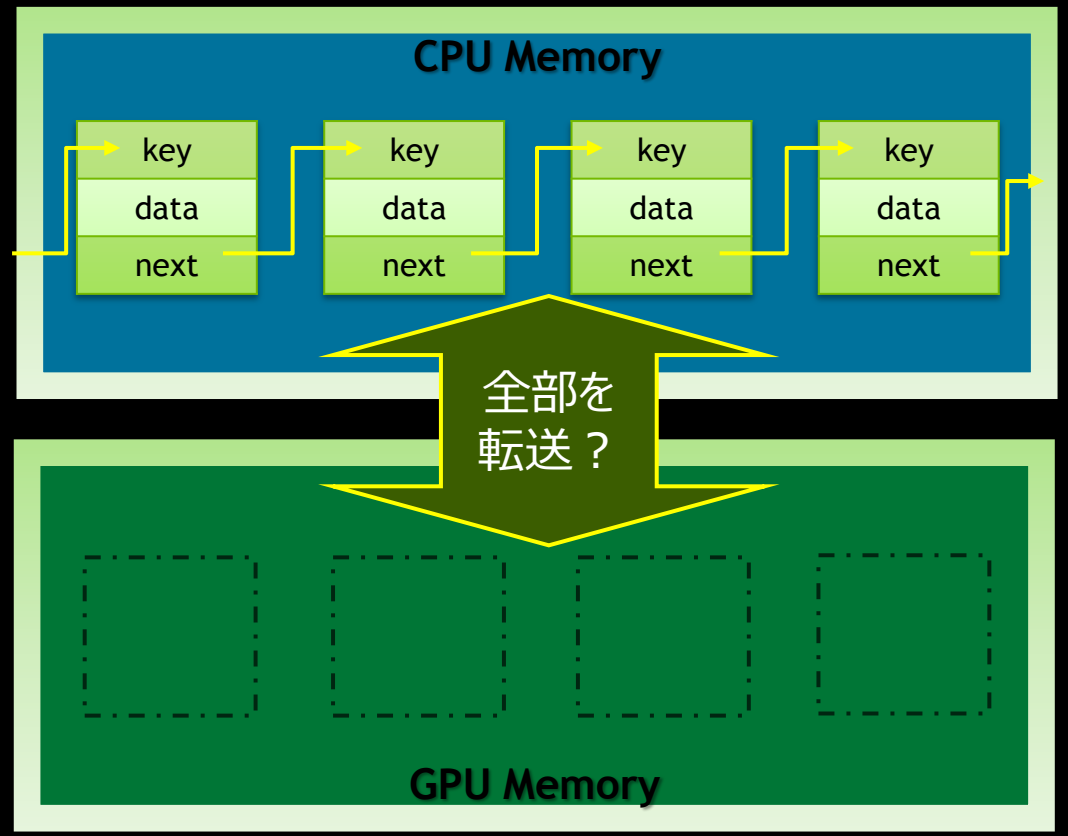
連結リスト



GPU TECHNOLOGY CONFERENCE

連結リスト

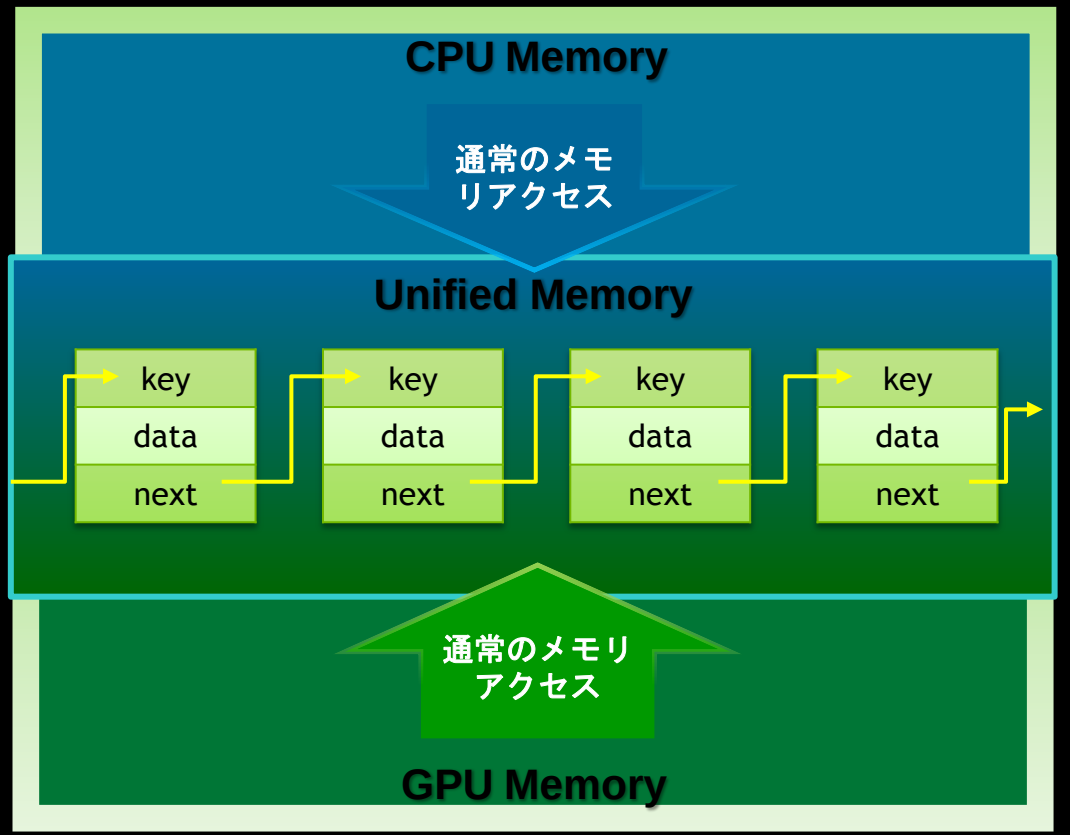
- 毎回、全部転送
  - PCIのバンド幅ネック
- 最初は全部転送、以降は更新箇所だけ転送
  - とても複雑な処理
- CPUメモリにデータを配置、GPUはPCI経由のアクセス
  - PCI経由、遅い



GPU TECHNOLOGY CONFERENCE

連結リスト (ユニファイドメモリ)

- CPUからもGPUからもリスト操作が可能
  - 挿入、削除
- リスト更新後に、CPUメモリとGPUメモリ間の明示的な同期は不要
- CPUとGPUから同時アクセスはNG、排他制御必要



ロードマップ

CUDA 6: 簡単に利用

Next: 最適化

Pascal

単一のポインタ

Memcpy記述不要

ホスト側プログラムとデータ構造を共有

プリフェッチ

データ移動ヒント

OSサポートの追加

システムアロケータの統合

スタックメモリの統合

メモリコヒーレンシをHWでアクセラレート

GPU TECHNOLOGY CONFERENCE

XTライブラリ

GPU TECHNOLOGY CONFERENCE

XTライブラリ

- cuBLAS-XT and cuFFT-XT

- 明示的なデータ転送の指示は不要
  - 必要なGPUメモリはライブラリが確保
- マルチGPUに自動対応
  - マルチGPU向けのコード記述は不要
- GPUメモリ容量を超えるサイズに対応 (out-of-core)
  - カーネル実行とデータ転送をオーバーラップ(BLAS level 3)

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

CUBLAS

行列積コード
cuBLAS

```
cublasHandle_t handle;
cublasCreate(&handle);
cudaMalloc(&d_A, ..);
cudaMalloc(&d_B, ..);
cudaMalloc(&d_C, ..);
cudaSetMatrix(.., d_A, .., A, ..);
cudaSetMatrix(.., d_B, .., B, ..);
cublasDgemm(handle, .., d_A, ..,
d_B, .., d_C, ..);
cudaGetMatrix(.., d_C, .., C, ..);
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
cublasDestroy(handle);
```

CUBLAS → CUBLAS-XT

行列積コード

cuBLAS

```
cublasHandle_t handle;
cublasCreate(&handle);
cudaMalloc(&d_A, ..);
cudaMalloc(&d_B, ..);
cudaMalloc(&d_C, ..);
cudaSetMatrix(.., d_A, .., A, ..);
cudaSetMatrix(.., d_B, .., B, ..);
cublasDgemm(handle, .., d_A, ..,
d_B, .., d_C, ..);
cudaGetMatrix(.., d_C, .., C, ..);
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
cublasDestroy(handle);
```

cuBLAS-XT

```
cublasXtHandle_t handle;
cublasXtCreate(&handle);

cublasXtDgemm(handle, .., A, .., B,
.., C, ..);

cublasXtDestroy(handle);
```

GPU TECHNOLOGY CONFERENCE

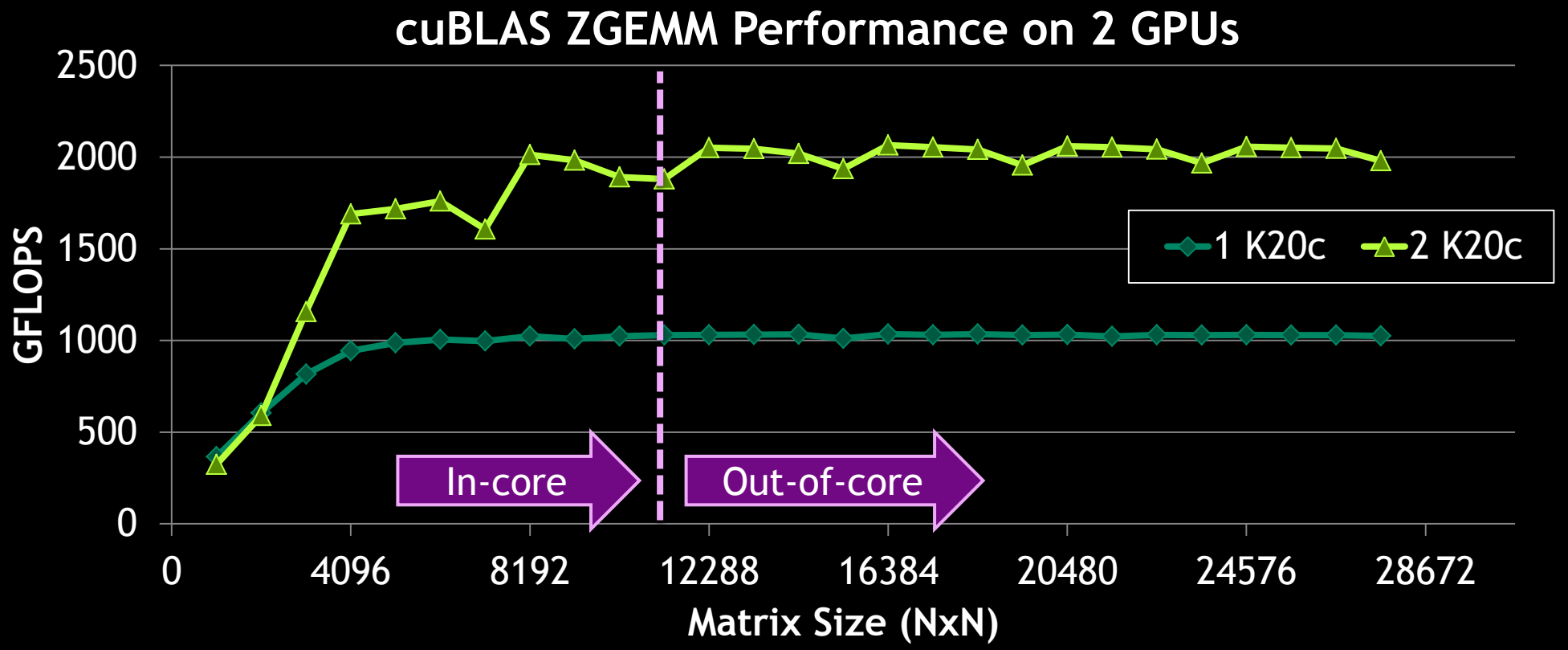
CUBLAS-XT API

- 使用GPU
  - cublasXtDeviceSelect() GPU数、使用GPU IDs
- ブロッキングサイズ
  - cublasXtSetBlockDim() ブロッキングサイズの設定
  - cublasXtGetBloskDim() (現設定の取得)
- CPU・GPUハイブリッド実行
  - cublasXtSetCpuRoutine() CPU版BLASの設定
  - cublasXtSetCpuRatio() CPU比率の設定
- Pinnedメモリ
  - cublasXtSetPinningMemMode() Pinnedメモリの設定
  - cublasXtGetPinningMemMode() (現設定の取得)

GPU TECHNOLOGY CONFERENCE

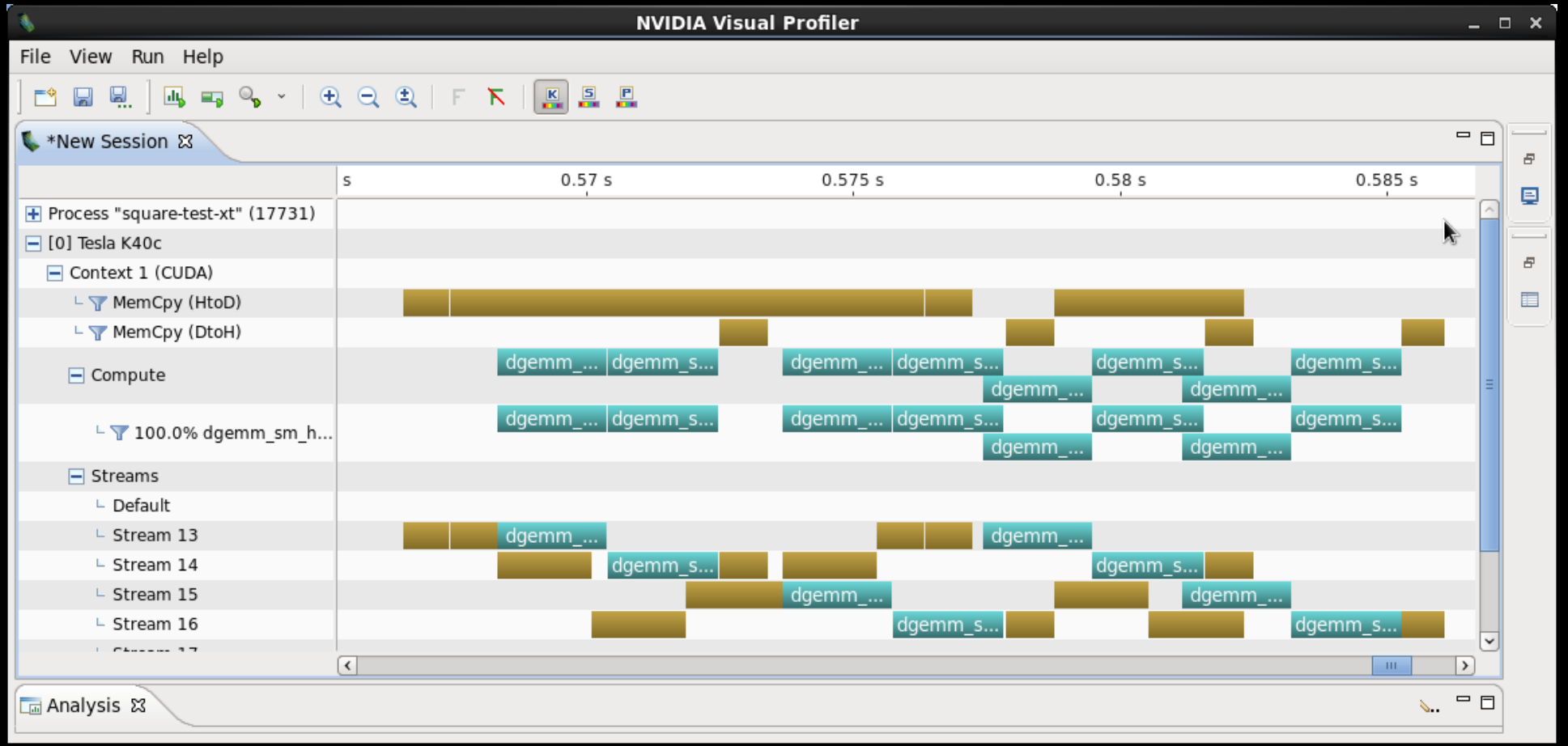
CUBLAS-XT

- 全てのBLAS level 3 ルーチンをサポート
- 行列サイズがGPUメモリ容量超でもOK (out-of-core)



GPU TECHNOLOGY CONFERENCE

CUBLAS-XT (NVVP)



ドロップイン・ライブラリ

GPU TECHNOLOGY CONFERENCE

ドロップイン・ライブラリ

- 標準ライブラリAPIでのGPU利用を可能に
- NVBLAS
  - BLAS level 3関数呼び出しを、自動的にcuBLASに置き換え
  - cuBLAS利用のためのソース変更は不要

CPU

```
dgemm(.., A, .., B, .., C, ..);
```

- 使い方
  - NVBLASを入れて再コンパイル
  - Linuxは、LD_PRELOAD設定で使用可能 (最コンパイル不要)

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

ドロップイン・ライブラリ

- 標準ライブラリAPIでのGPU利用を可能に
- NVBLAS
  - BLAS level 3関数呼び出しを、自動的にcuBLASに置き換え
  - cuBLAS利用のためのソース変更は不要

CPU

```
dgemm(.., A, .., B, .., C, ..);
```

NVBLAS

```
dgemm(.., A, .., B, .., C, ..);
```

- 使い方
  - NVBLASを入れて再コンパイル
  - Linuxは、LD_PRELOAD設定で使用可能 (最コンパイル不要)

NVBLAS (LINUX)

設定ファイル (nvblas.conf)

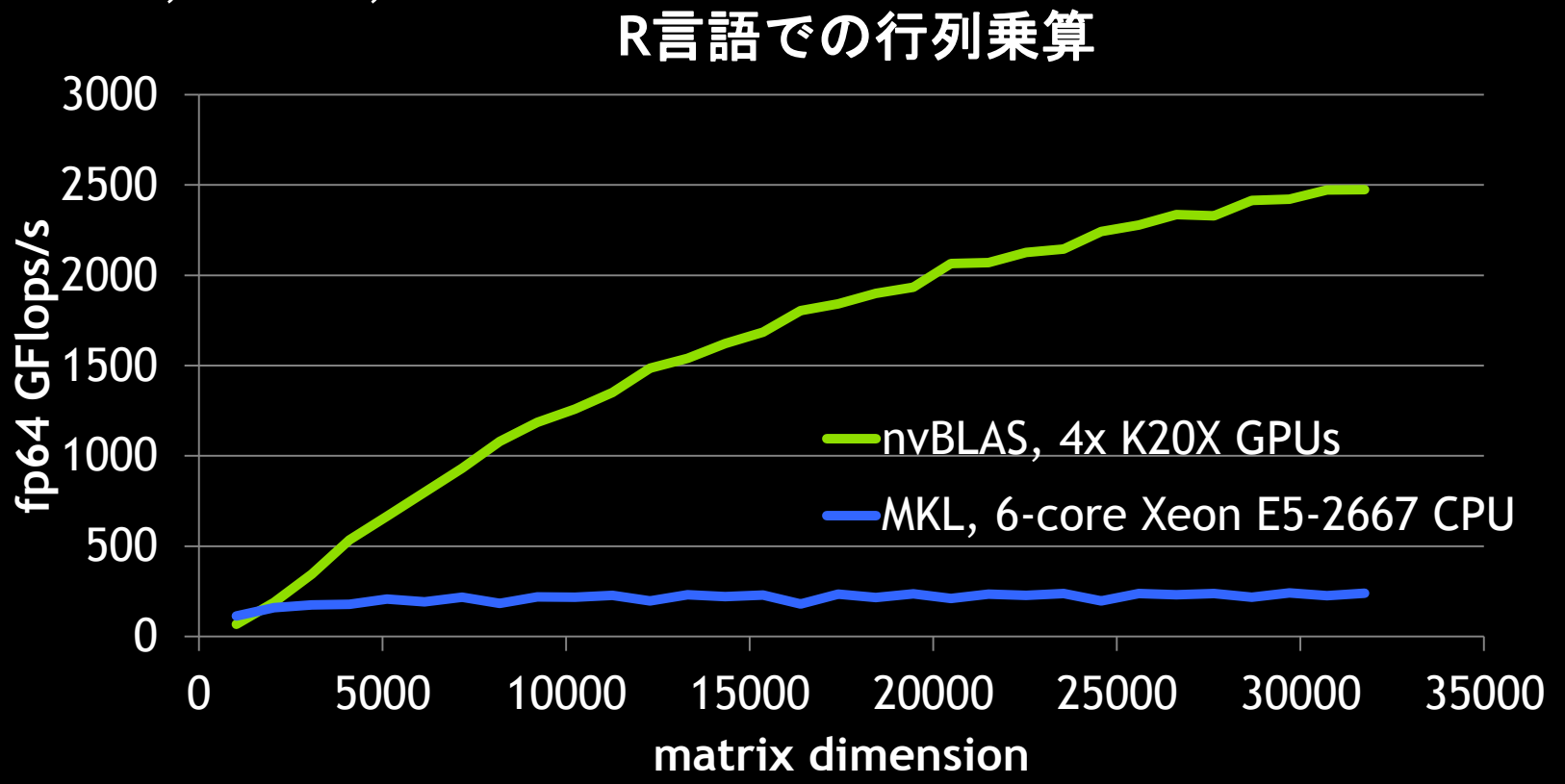
```
NVBLAS_LOGFILE          nvblas.log
NVBLAS_CPU_BLAS_LIB     libmkl_intel_lp64.so \
                        libmkl_core.so \
                        libmkl_intel_thread.so
NVBLAS_GPU_LIST         0  # ALL, ALL0
NVBLAS_TILE_DIM         2048
NVBLAS_AUTOPIN_MEM_ENABLED
```

```
$ LD_PRELOAD=/usr/local/cuda-6.0/lib64/libnvblas.so ./a.out
```

GPU TECHNOLOGY CONFERENCE

NVBLAS

- BLAS level 3使用のアプリに適用可能
  - Octave, Scilab, など.



GPU TECHNOLOGY CONFERENCE

NVBLASデモ

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

CUDA 6

ユニファイド・メモリ
XTライブラリ
ドロップイン・ライブラリ
GPUDirect RDMA
開発ツール

CUDA 6

並列コンピューティング
を簡単に

CUDA Registered Developer Program

developer.nvidia.com/cuda-toolkit

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

CUDA 6.5 RC

- 64-bit ARMマシン
- Microsoft Visual Studio 2013 (VC12)
- cuFFT callbacks
- cuSPARSE (BSR格納形式)
- CUDA占有率計算API
- CUDA FORTRAN デバッグ機能
- アプリケーションリプレイモード (Visual Profile and nvprof)
- Nvprune ユーティリティ (objectサイズ削減)