TORSTEN HOEFLER

Progress in automatic GPU compilation and why you want to run MPI on your GPU

with Tobias Grosser and Tobias Gysi @ SPCL
presented at CCDSC, Lyon, France, 2016
#pragma ivdep
!$ACC DATA &
!$ACC PRESENT(density1,energy1) &
!$ACC PRESENT(vol_flux_x,vol_flux_y,volume,mass_flux_x,mass_flux_y,vertexdx,vertexdy) &
!$ACC PRESENT(pre_vol,post_vol,ener_flux)

!$ACC KERNELS

IF(dir.EQ.g_xdir) THEN

IF(sweep_number.EQ.1) THEN

!$ACC LOOP INDEPENDENT
DO k=y_min-2,y_max+2
!$ACC LOOP INDEPENDENT
DO j=x_min-2,x_max+2
pre_vol(j,k)=volume(j,k)+(vol_flux_x(j+1,k)-vol_flux_x(j,k)+vol_flux_y(j,k+1)-vol_flux_y(j,k))
post_vol(j,k)=pre_vol(j,k)-(vol_flux_x(j+1,k)-vol_flux_x(j,k))
ENDDO
ENDDO
ELSE
!$ACC LOOP INDEPENDENT
DO k=y_min-2,y_max+2
!$ACC LOOP INDEPENDENT
DO j=x_min-2,x_max+2
pre_vol(j,k)=volume(j,k)+vol_flux_x(j+1,k)-vol_flux_x(j,k)
post_vol(j,k)=volume(j,k)
ENDDO
ENDDO
ENDIF
<table>
<thead>
<tr>
<th>ISO 9126 maintainability</th>
<th>source code properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>volume</td>
</tr>
<tr>
<td>analysability</td>
<td>x</td>
</tr>
<tr>
<td>changeability</td>
<td></td>
</tr>
<tr>
<td>stability</td>
<td></td>
</tr>
<tr>
<td>testability</td>
<td></td>
</tr>
</tbody>
</table>
!$ACC DATA &
!$ACC COPY(chunk%tiles(1)%field%density0) &
!$ACC COPY(chunk%tiles(1)%field%density1) &
!$ACC COPY(chunk%tiles(1)%field%energy0) &
!$ACC COPY(chunk%tiles(1)%field%energy1) &
!$ACC COPY(chunk%tiles(1)%field%pressure) &
!$ACC COPY(chunk%tiles(1)%field%soundspeed) &
!$ACC COPY(chunk%tiles(1)%field%viscosity) &
!$ACC COPY(chunk%tiles(1)%field%volume) &
!$ACC COPY(chunk%tiles(1)%field%work_array1) &
!$ACC COPY(chunk%tiles(1)%field%work_array2) &
!$ACC COPY(chunk%tiles(1)%field%work_array3) &
!$ACC COPY(chunk%tiles(1)%field%work_array4) &
!$ACC COPY(chunk%tiles(1)%field%work_array5) &
!$ACC COPY(chunk%tiles(1)%field%work_array6) &
!$ACC COPY(chunk%tiles(1)%field%work_array7) &
!$ACC COPY(chunk%tiles(1)%field%cellx) &
!$ACC COPY(chunk%tiles(1)%field%celly) &
!$ACC COPY(chunk%tiles(1)%field%celldx) &
!$ACC COPY(chunk%tiles(1)%field%celldy) &
!$ACC COPY(chunk%tiles(1)%field%vertexx) &
!$ACC COPY(chunk%tiles(1)%field%vertexdx) &
!$ACC COPY(chunk%tiles(1)%field%vertexy) &
!$ACC COPY(chunk%tiles(1)%field%vertexdy) &
!$ACC COPY(chunk%tiles(1)%field%xarea) &
!$ACC COPY(chunk%tiles(1)%field%yarea) &
!$ACC COPY(chunk%left_snd_buffer) &
!$ACC COPY(chunk%left_rcv_buffer) &
!$ACC COPY(chunk%right_snd_buffer) &
!$ACC COPY(chunk%right_rcv_buffer) &
!$ACC COPY(chunk%bottom_snd_buffer) &
!$ACC COPY(chunk%bottom_rcv_buffer) &
!$ACC COPY(chunk%top_snd_buffer) &
!$ACC COPY(chunk%top_rcv_buffer)

Sloccount *f90: 6,440

!$ACC: 833 (13%)
\[ \text{do } i = 0, N \]
\[ \text{do } j = 0, i \]
\[ y(i,j) = \frac{y(i,j) + y(i,j+1)}{2} \]
Some results: Polybench 3.2

Xeon E5-2690 (10 cores, 0.5Tflop) vs. Titan Black Kepler GPU (2.9k cores, 1.7Tflop)

arithmean: ~30x
geomean: ~6x

T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS’16
Compiles all of SPEC CPU 2006 – Example: LBM

essentially my 4-core x86 laptop with the (free) GPU that’s in there

Runtime (m:s)

Mobile

Workstation

Xeon E5-2690 (10 cores, 0.5Tflop) vs. Titan Black Kepler GPU (2.9k cores, 1.7Tflop)

~20%

~4x

T. Grosser, TH: Polly-ACC: Transparent compilation to heterogeneous hardware, ACM ICS’16
GPU latency hiding vs. MPI

CUDA
• over-subscribe hardware
• use spare parallel slack for latency hiding

MPI
• host controlled
• full device synchronization

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Hardware latency hiding at the cluster level?

dCUDA (distributed CUDA)
- unified programming model for GPU clusters
- avoid unnecessary device synchronization to enable system wide latency hiding

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
dCUDA: MPI-3 RMA extensions

\[
\text{for (int } i = 0; i < \text{steps}; ++i) { }
\text{for (int } idx = \text{from}; idx < \text{to}; idx += jstride) }
\text{out[idx] = -4.0 * in[idx] +}
\text{in[idx + 1] + in[idx - 1] +}
\text{in[idx + jstride] + in[idx - jstride]; }
\]

\text{if (lsend) }
\text{dcuda_put_notify(ctx, wout, rank - 1,}
\text{len + jstride, jstride, &out[jstride], tag); }
\text{if (rsend) }
\text{dcuda_put_notify(ctx, wout, rank + 1,}
\text{0, jstride, &out[len], tag); }
\text{dcuda_wait_notifications(ctx, wout,}
\text{DCUDA_ANY_SOURCE, tag, lsend + rsend); }
\text{swap(in, out); swap(win, wout); }
\]

- iterative stencil kernel
- thread specific idx
- map ranks to blocks
- device-side put/get operations
- notifications for synchronization
- shared and distributed memory

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Hardware supported communication overlap

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
The dCUDA runtime system

- Event handler
- Block manager
- Logging queue
- Command queue
- Ack queue
- Notification queue
- Device library
- Context
- MPI

Host-side

Device-side

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
(Very) simple stencil benchmark

- Benchmarked on 8 Haswell nodes with 1x Tesla K80 per node

![Graph showing execution time vs. # of copy iterations per exchange]

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Real stencil (COSMO weather/climate code)

- Benchmarked on 8 Haswell nodes with 1x Tesla K80 per node
Particle simulation code (Barnes Hut)

- Benchmarked on 8 Haswell nodes with 1x Tesla K80 per node

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Sparse matrix-vector multiplication

- Benchmarked on 8 Haswell nodes with 1x Tesla K80 per node

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
for (int i = 0; i < steps; ++i) {
    for (int idx = from; idx < to; idx += jstride)
        out[idx] = -4.0 * in[idx] + in[idx + 1] + in[idx - 1] +
                    in[idx + jstride] + in[idx - jstride];
    if (lsend)
        dcuda_put_notify(ctx, wout, rank - 1, len + jstride, jstride, &out[jstride], tag);
    if (rsend)
        dcuda_put_notify(ctx, wout, rank + 1, 0, jstride, &out[len], tag);
    dcuda_wait_notifications(ctx, wout, DCUDA_ANY_SOURCE, tag, lsend + rsend);
    swap(in, out);
    swap(win, wout);
}
LLVM Nightly Test Suite

No Heuristics

Heuristics
Cactus ADM (SPEC 2006)

Performance [iterations/second]

Mobile

- Polly ACC cached
- Polly ACC
- icc parallel
- icc
- LLVM

Report output interval

Workstation

- Polly ACC cached
- Polly ACC
- icc parallel
- icc
- LLVM

Report output interval

Performance [iterations/second]
Evading various “ends” – the hardware view

Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond