TRILINOS TOOLS: IDENTIFYING HOST-TO-DEVICE TRANSFERS AND STATE OF THE CODE

Christopher Siefert & Jonathan Hu
TUG 2023, November 1, 2023

Special Thanks: Tpetra Team and Christian Trott
MOTIVATION

• With code as complicated as Trilinos (let alone an app) it is hard to tell:
  • Where host-to-device (H2D) or device-to-host (D2H) data traffic happens.
  • Where Kokkos::fence() calls happen.
  • What impact this has on performance.

• For codes using Teuchos::StackedTimer we have tools to answer these questions!

• This talk has two parts:
  • What the tools are and how to use them (Chris).
  • What the state-of-the-code is for portions of Trilinos and select proxy apps (Jonathan).

• WARNING: These tools will add fences which will distort your timings.
LIFE WITHOUT TEUCHOS::STACKEDTIMER 😞

- What if your app doesn’t use Teuchos::StackedTimer?
  - If you use Teuchos::Timer, then adding StackedTimer is very easy (see MueLu Driver for example).

- If you don’t use Teuchos::Timer, we’ve added some bells & whistles to kokkos-tools/profiling/space-time-stack.

- This will now print the memory space names for all calls to Kokkos::deep_copy, so you can identify which calls generated H2D/D2H traffic:

```plaintext
| -> 1.44e-02 sec 8.2% 28.9% 0.0% 71.1% 1.32e+03 | Galeri: Laplace 2D FillComplete [region] |
| | -> 1.37e-03 sec 0.8% 100.0% 0.0% ------- 1 | sortAndMergeIndicesAndValues [reduce] |
| | | -> 9.04e-04 sec 0.5% 100.0% 0.0% ------- 1 | "Tpetra::CrsGraph::lclInd_mirror"="Tpetra::CrsGraph::lclInd" [copy] |
| | | | -> 5.23e-04 sec 0.3% 100.0% 0.0% ------- 1 | "Tpetra::CrsMatrix::values"="Tpetra::CrsMatrix::values_mirror" [copy] |
| | | | | -> 5.23e-04 sec 0.3% 100.0% 0.0% ------- 1 | "Tpetra::CrsGraph::gblInd="Tpetra::CrsGraph::gblInd_mirror" [copy] |
| | | | -> 3.15e-04 sec 0.2% 100.0% 0.0% ------- 1 | "Tpetra::CrsGraph::lclInd"="Tpetra::CrsGraph::lclInd_mirror" [copy] |
```
TRILINOS TOOLING (DIAGNOSTIC)

• ENV variables to add Teuchos::Timer objects to various Kokkos things

1. TPETRA_USE_TEUCHOS_TIMERS – Adds Teuchos::Timers to all Tpetra::ProfilingRegions.

2. TPETRA_TIME_KOKKOS_FENCE – Adds Teuchos::Timers to all Kokkos::fence() calls.

3. TPETRA_TIME_KOKKOS_DEEP_COPY – Adds Teuchos::Timers to all Kokkos::deep_copy() calls. This includes space names.
   • TPETRA_TIME_KOKKOS_DEEP_COPY_VERBOSE1 – Adds src/dest View names.
   • TPETRA_TIME_KOKKOS_DEEP_COPY_VERBOSE2 – Adds vector sizes (can generate lots of output in parallel since sizes are different between ranks)

4. TPETRA_TIME_KOKKOS_FUNCTIONS – Adds Teuchos::Timers to Kokkos::parallel_* calls.

• #2 and #3 can also be enabled through explicit function calls.
EXAMPLE: TPETRA::IMPORT CONSTRUCTOR

• We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.

• Output without any ENV variables set:

```
Driver: 3.32417 [1]
| Global: 3.32411 - 99.9983% [1]
| | createImport: 0.000472806 - 0.0142235% [1]
| | Remainder: 3.32364 - 99.9858%
| Remainder: 5.5183e-05 - 0.00166006%
[Passed] (3.32 sec)
```

What we named the Timer surrounding the test
EXAMPLE: TPETRA::IMPORT CONSTRUCTOR

- We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.
  - TPETRA_USE_TEUCHOS_TIMERS

```
Driver: 1.60914 [1]
| Global: 1.60908 - 99.9965% [1]
| | createImport: 0.000482423 - 0.0299813% [1]
| | | Tpetra::Import::init: 0.000376352 - 78.0129% [1]
| | | | Tpetra::Import::setupSamePermuteRemote: 0.000317982 - 84.4906% [1]
| | | | | Remainder: 5.837e-05 - 15.5094%
| | | | | | Remainder: 0.000106071 - 21.9871%
| | | | | | | Remainder: 1.6086 - 99.97%
| | | | | | | | Remainder: 5.6199e-05 - 0.00349249%
```

Profiling Regions
**EXAMPLE: TPETRA::IMPORT CONSTRUCTOR**

- We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.
  - TPETRA_USE_TEUCHOS_TIMERS
  - TPETRA_TIME_KOKKOS_DEEP_COPY

<table>
<thead>
<tr>
<th>Driver: 3.34351 [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global: 3.34346 - 99.9985% [1]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
| | | | | | | | | | Remaining: 4.8542e-05 - 0.00145183%

Deep Copies
EXAMPLE: TPETRA::IMPORT CONSTRUCTOR

- We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.
  
  - TPETRA_USE_TEUCHOS_TIMERS
  - TPETRA_TIME_KOKKOS_DEEP_COPY_VERBOSE1

Driver: 3.2199 [1]
| Global: 3.21985 - 99.9983% [1]
| | createImport: 0.000559054 - 0.0173628% [1]
| | | Tpetra::Import::init: 0.000465691 - 83.2998% [1]
| | | | Tpetra::Import::setupSamePermuteRemote: 0.000431026 - 92.5562% [1]
| | | | | Kokkos::deep_copy [Cuda=>Host] {lgMap=>lgMap_mirror}: 3.6773e-05 ...
| | | | | Kokkos::deep_copy_small [Cuda=>Host] {lgMap=>lgMap_mirror}: 1.0237e-05 ...
| | | | | Remainder: 0.000384016 - 89.0935%
| | | | | | Remainder: 3.4665e-05 - 7.44378%
| | | | | | Remainder: 9.3363e-05 - 16.7002%
| | | | | | Remainder: 5.3361e-05 - 0.00165723%
| | | | | | Remainder: 3.21929 - 99.9826%
| | | | | | | Passed] (3.22 sec)
EXAMPLE: TPETRA::IMPORT CONSTRUCTOR

- We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.
  - TPETRA_USE_TEUCHOS_TIMERS
  - TPETRA_TIME_KOKKOS_DEEP_COPY_VERBOSE2

Driver: 1.88643 [1]
|   Global: 1.88638 - 99.9973% [1]
|   | createImport: 0.000558055 - 0.0295833% [1]
|   | | Tpetra::Import::init: 0.000466191 - 83.5385% [1]
|   | | | | Tpetra::Import::setupSamePermuteRemote: 0.000428582 - 91.9327% [1]
|   | | | | | Kokkos::deep_copy [Cuda=>Host] {lgMap=>lgMap_mirror, 80}:...
|   | | | | | Kokkos::deep_copy_small [Cuda=>Host] {lgMap=>lgMap_mirror, 40}:...
|   | | | | | | Remainder: 0.000383877 - 89.5691%
|   | | | | | | Remainder: 3.7609e-05 - 8.06729%
|   | | | | | | Remainder: 9.1864e-05 - 16.4615%
|   | | | | | | Remainder: 1.88582 - 99.9704%
|   | | | | | | Remainder: 5.0911e-05 - 0.0026988%
[Passed] (1.89 sec)
EXAMPLE: TPETRA::IMPORT CONSTRUCTOR

- We added Tpetra::StackedTimer to a test which builds a Tpetra::Import object.
  
  - TPETRA_USE_TEUCHOS_TIMERS
  - TPETRA_TIME_KOKKOS_DEEP_COPY_VERBOSE2
  - TPETRA_TIME_KOKKOS_FENCE

Driver: 3.33936 [1]
| Global: 3.3393 - 99.9984% [1]
| | createImport: 0.000616963 - 0.0184758% [1]
| | | Tpetra::Import::init: 0.000510063 - 82.6732% [1]
| | | | Tpetra::Import::setupSamePermuteRemote: 0.000473831 - 92.8966% [1]
| | | | | Kokkos::fence SharedAllocationRecord<Kokkos::CudaSpace, void>::SharedAllocationRecord(): fence after copying header from HostSpace (Cuda Instance 1): 1.9424e-05 ...
| | | | | | Kokkos::fence Kokkos::Impl::ViewValue Functor: View init/destroy fence (Cuda Instance 1): ...
| | | | | | Kokkos::fence HostSpace fence (Serial Instance 1): 1.009e-06 - 0.212945% [2]
| | | | | | Kokkos::fence Kokkos::Impl::ViewValue Functor: View init/destroy fence (Serial Instance 1): ...
| | | | | | Kokkos::deep_copy [Cuda=>Host]: 2.9377e-05 - 6.19989% [1]
| | | | | | Kokkos::fence Kokkos::Cuda::fence(): Unnamed Instance Fence (Cuda Instance 1): ...
| | | | | | Kokkos::deep_copy_small [Cuda=>Host]: 1.0142e-05 - 2.14043% [1]
| | | | | | Remainder: 0.000400651 - 84.5557%
| | | | | | Remainder: 3.6232e-05 - 7.10344%
| | | | | | Remainder: 0.0001069 - 17.3268%
| | | | | | Remainder: 3.33868 - 99.9815%
| | | | | | Remainder: 5.4954e-05 - 0.00164565%
| | | | | | [Passed] (3.34 sec)

Execution space fences. Device fences would say “All Instances”
In addition to diagnostic tools, Tpetra also provides count-based tools for use in testing.

Verify: “This code should only have X Kokkos::deep_copy() calls between Spaces”

We can do this through Tpetra’s DeepCopyCounter and FenceCounter.

These require code modification since they’re designed to test specific code fragments.
EXAMPLE: DEEPCOPY COUNTER

using Tpetra::Details;
DeepCopyCounter::start();
Kokkos::deep_copy(view1, view2);
size_t ct_same = DeepCopyCounter::get_count_same_space();
size_t ct_diff = DeepCopyCounter::get_count_different_space();

• The counter separately tallies copies *within the same* space vs. copies *between* spaces.

• To verify there are no H2D/D2H copies, you want `get_count_different_space();`

• You can also `stop()` the counter and `reset()` it to zero.
EXAMPLE: FENCE COUNTER

using Tpetra::Details;
FenceCounter::start();
Kokkos::fence();
size_t ct_inst = FenceCounter::get_count_instance(“Cuda”);
size_t ct_gbl = FenceCounter::get_count_global(“Cuda”);

• Instance and Global fences are tallied separately.
  • Think cudaMemcpyAsync vs. cudaMemcpy.

• Fences are tracked on a space by space basis and you have to ask for the one you want. execution_space().name() will give you the string you want.

• You can also stop() the counter and reset() it to zero.