Charm++ - Bug #1462
Programs hang at startup with CUDA build
03/07/2017 05:20 PM - Jaemin Choi

<table>
<thead>
<tr>
<th>Status:</th>
<th>In Progress</th>
<th>Start date:</th>
<th>03/07/2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>High</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>Jaemin Choi</td>
<td>% Done:</td>
<td>0%</td>
</tr>
<tr>
<td>Category:</td>
<td>GPU Support</td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
</tbody>
</table>

Description
When using the CUDA build of Charm++, example programs located both under examples/charm++/cuda hang at startup.

With charmrun:

- Hello 1Ddrray (examples/charm++/hello/1darray)

```
[jchoi157@nano6 1darray]$ ./charmrun +p4 ++local ./hello
Charmrun> scalable start enabled.
Charmrun> error attaching to node '127.0.0.1':
Timeout waiting for node-program to connect
```

- CUDA Hello (examples/charm++/cuda/hello)

```
[jchoi157@nano6 hello]$ ./charmrun +p4 ++local ./hello 5
Charmrun> scalable start enabled.
Charmrun> error attaching to node '127.0.0.1':
Timeout waiting for node-program to connect
```

Without charmrun:

- OverlapTestGPUManager (examples/charm++/cuda/overlapTestGPUManager)

```
[jchoi157@nano6 overlapTestGPUManager]$ ./overlapTest
Charm++: standalone mode (not using charmrun)
Charm++> Running in non-SMP mode: numPes 1
Converse/Charm++ Commit ID: v6.7.0-682-gc59183a
Warning> Randomization of stack pointer is turned on in kernel, thread migration may not work!
Run 'echo 0 > /proc/sys/kernel/randomize_va_space' as root to disable it, or try run with '+isomaloc_sync'.
Charm++> scheduler running in netpoll mode.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 unique compute nodes (24-way SMP).
Charm++> cpu topology info is gathered in 0.002 seconds.
MATMUL KERNEL
MATMUL KERNEL
MATMUL KERNEL
MATMUL KERNEL
```

Some programs seem to work without charmrun though. (Hello 1darray, for example)

History
Problem seems to be in initHybridAPI() in ck-core/init.C, because programs run fine if I comment this out along with exitHybridAPI() in conv-core/convcore.c.

The culprit is cudaStreamCreate() in GPUManager::initHybridAPIHelper(). No idea why this is the cause yet, because only rank 0 calls this function so there should be no contention or anything.

This was due to the GPU being heavily used by other processes. nvidia-smi -q shows the current usage. When tested on nano7 where the 4 GPUs were all free instead of nano6 where the usage was more than 60%, the example programs (both CUDA and non-CUDA) ran fine.

There is no way cudaStreamCreate should hang due to GPU load so this would be good to track down. All I can think is that when you launch 4 processes with ++local they are all running cudaStreamCreate and other calls on the same device almost simultaneously. Is there code that spreads the four processes across the GPUs? You really don't want to have multiple processes accessing the same GPU unless you are using the CUDA multi-process server.

Exactly same code (examples/charm++/hello/1darray) with 4 PEs crashes on nano6 but runs fine on nano7. GPUManager does spread the four processes across GPUs, but even if I manually set it to use only the first GPU on nano7 it runs without a problem. It seems like only two processes are able to create streams and it crashes as soon as the third process tries to create them. (Only two CUDA contexts allowed per device? But the successful run on nano7 suggests otherwise)

To Jim: we surely do not want to have multiple processes that access the same GPU because they will be all serialized without MPS, but shouldn't it still be able to run like it does on nano7?

The error from the run on nano6 is as follows:

```
[jchoi157@nano6 1darray]$ ./charmrun +p4 ++local ./hello
Charmrun> scalable start enabled.
Charmrun> started all node programs in 0.031 seconds.
Charm++> Running in non-SMP mode: numPes 4
Converse/Charm++ Commit ID: v6.7.0-682-g421c4d7
Warning> Randomization of stack pointer is turned on in kernel, thread migration may not work! Run 'echo 0 > /proc/sys/kernel/randomize_va_space' as root to disable it, or try run with '+isomalloc_sync'.
Charm++> scheduler running in netpoll mode.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 unique compute nodes (24-way SMP).
hello! my rank is 0
calling initHybridAPI
Charm++> cpu topology info is gathered in 0.007 seconds.
hello! my rank is 0
calling initHybridAPI
hello! my rank is 0
calling initHybridAPI
hello! my rank is 0
calling initHybridAPI
initHybridAPI done!!!
```
Fatal CUDA Error out of memory at cuda-hybrid-api.cu:660.
Return value 2 from 'cudaStreamCreate(&(CsvAccess(gpuManager).streams[i]))'.
Processor 0 Exiting:
Called CmiAbort
Reason: Exiting!
[0] Stack Traceback:
[0:0] CmiAbortHelper+0xb3 [0x5db5f7]
[0:1] CmiAbort+0x2d [0x5db632]
[0:2] cudaErrorDie+0x5f [0x60db88]
[0:3] initHybridAPI+0x52 [0x60e64f]
[0:4] __210_initCharmiPPc+0x678 [0x506abd]
[0:5] [0x5db3e5]
[0:6] ConverseInit+0x324 [0x5db303]
[0:7] main+0x3f [0x5047fc]
[0:8] __libc_start_main+0xf5 [0x7fb82df1fb35]
[0:9] [0x500299]
Fatal error on PE 0> Exiting!

You noticed this line, right?

You can't oversubscribe and swap memory on the GPU, so it's not load but memory usage that is causing the failure on the busy node. I assume there are some unchecked device memory allocations that are failing prior to this.
How much memory is the API allocating for buffers and how is it calculated?

There is a mempool initialization step that does a bunch of cudaMallocHost calls (which I believe has nothing to do with GPU memory), but even if I disable this the problem still persists.
There are no other device memory allocations done by the Charm++ runtime.
It does seem to be related to memory usage, but streams should not take too much memory and it is strange that stream creation fails even with the GPU memory usage at around 40%.