Charm++ - Bug #2030

_tests/ampi/megampi sometimes fails on mpi-win-x86_64-smp

11/19/2018 04:28 PM - Evan Ramos

<table>
<thead>
<tr>
<th>Status:</th>
<th>In Progress</th>
<th>Start date:</th>
<th>11/19/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>Evan Ramos</td>
<td>% Done:</td>
<td>0%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>6.10.0</td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
</tbody>
</table>

Description

This failure shows up in autobuild every few days.

```
../../../bin/testrun ./pgm +p2 +vp4
```

```
Running on 2 processors: ./pgm +vp4
charmrun> /cygdrive/c/Program Files/Microsoft MPI/Bin/mpiexec -n 2 ./pgm +vp4
```

Charm++> Running on MPI version: 2.0
Charm++> level of thread support used: MPI_THREAD_FUNNELED (desired: MPI_THREAD_FUNNELED)
Charm++> Running in SMP mode: 2 processes, 1 worker threads (PEs) + 1 comm threads per process, 2 PEs total
Charm++> The comm. thread both sends and receives messages
Charm++ warning> fences and atomic operations not available in native assembly
Converse/Charm++ Commit ID: v6.9.0-0-gc3d50ef
Charm++> Disabling isomalloc because mmap() does not work.
CharmLB> Load balancer assumes all CPUs are same.
Charm++> Running on 1 hosts (1 sockets x 4 cores x 2 PUs = 8-way SMP)
Charm++> cpu topology info is gathered in 0.016 seconds.
CharmLB> RandCentLB created.

job aborted:
[ranks] message
[0] terminated

[1] process exited without calling finalize

```---- error analysis ----```

```
job aborted: 
[ranks] message
[0] terminated

[1] process exited without calling finalize
```

```---- error analysis ----```

```
../pgm ended prematurely and may have crashed. exit code 0xc0000005
```

```---- error analysis ----```

```
make[1]: *** [Makefile:37: test-ampi] Error 2
make[3]: Leaving directory '/home/nikhil/autobuild/mpi-win-x86_64-smp/charm/mpi-win-x86_64-smp/tests/ampi/megampi'
```

http://charm.cs.illinois.edu/autobuild/old.2018_11_06__01_01/mpi-win-x86_64-smp.txt
http://charm.cs.illinois.edu/autobuild/old.2018_11_10__01_01/mpi-win-x86_64-smp.txt
http://charm.cs.illinois.edu/autobuild/old.2018_11_14__01_01/mpi-win-x86_64-smp.txt

History

#1 - 11/29/2018 12:57 PM - Eric Bohm
- Assignee set to Evan Ramos
I tried the following change to help diagnose the problem:

```c
index dad98cf50..8f32e8e77 100644
--- a/src/libs/ck-libs/ampi/ampi.C
+++ b/src/libs/ck-libs/ampi/ampi.C
@@ -8990,7 +8990,9 @@ AMPI_API_IMPL(int, MPI_Comm_free, MPI_Comm *comm)
     // set = parent->freeUserKeyvals(*comm, parent->getKeyvals(*comm));
 if (*comm != MPI_COMM_WORLD & & *comm != MPI_COMM_SELF) {
     ampi* ptr = getAmpiInstance(*comm);
+     CmiEnforce(*comm == ptr->getCommStruct().getComm()); // assertion 1
+     ptr->barrier();
+     CmiEnforce(*comm == ptr->getCommStruct().getComm()); // assertion 2
 if (ptr->getRank() == 0) {
     CProxy_CkArray(ptr->ckGetArrayID()).ckDestroy();
 }
```

The odd thing is that assertion 1 succeeds but assertion 2 fails.

ptr->barrier() calls thread->suspend(), which calls CthSuspend(). I suspect the problem is there.

Alternatively, there is the following comment in tcharm_impl.h:
/* SUBTLE: We have to do the get() because "this" may have changed
* during a migration-suspend. If you access *any* members
* from this point onward, you'll cause heap corruption if
* we're resuming from migration! (OSL 2003/9/23) */

I tried changing assertion 2 to CmiEnforce(*comm == getAmpiInstance(*comm)->getCommStruct().getComm()); but it still failed, just with a null pointer deference here:

```
I am more inclined to believe the problem is in CthSuspend(); because this failure does not always occur, and it only occurs on Windows.
```
I tried running megampi on Linux with ThreadSanitizer and the list of data races was substantial. Some of them look like candidates for this issue, including AMPI implementation details relevant to the failure seen on Windows.

```
./build AMPI multicore-linux-x86_64 tsan -j8 -g3 -fsanitize=thread & cd multicore-linux-x86_64-tsan/tests/ampli/megampi/ & make -j8 OPTS="-g3 -fsanitize=thread" & TSAN_OPTIONS='log_path=tsan.log' ./pgm +p4 +vp2 +tcharm _nomig +noisomalloc
```

#8 - 03/15/2019 03:00 PM - Sam White
Can you post the tsan output here?

#9 - 03/15/2019 03:30 PM - Evan Ramos
- File tsan.log.3262 added

#10 - 03/15/2019 05:58 PM - Evan Ramos
I ran megampi on Windows with a Microsoft tool called Application Verifier and it pointed out these two additional problems but I'm not sure either can be blamed for this issue.

1. "Invalid TLS index used for current stack trace."

```
<avrf:message>Invalid TLS index used for current stack trace.</avrf:message>
<avrf:parameter1>ffffffff - Invalid TLS index.</avrf:parameter1>
<avrf:parameter2>abba - Expected lower part of the index.</avrf:parameter2>
<avrf:parameter3>0 - Not used.</avrf:parameter3>
<avrf:parameter4>0 - Not used.</avrf:parameter4>
<avrf:stackTrace>
  <avrf:trace>vfbasics!+7ffe620caef9 ( @ 0)</avrf:trace>
  <avrf:trace>vfbasics!+7ffe620cb12f ( @ 0)</avrf:trace>
  <avrf:trace>pgm!CmiGetState+10 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\machine-smp.c @ 115)</avrf:trace>
  <avrf:trace>pgm!CmiMyPe+9 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\machine-common-core.c @ 399)</avrf:trace>
  <avrf:trace>pgm!CmiAddCLA+18 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\convcore.c @ 325)</avrf:trace>
  <avrf:trace>pgm!CmiGetArgFlagDesc+29 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\convcore.c @ 579)</avrf:trace>
  <avrf:trace>pgm!CmiGetArgFlag+24 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\convcore.c @ 589)</avrf:trace>
  <avrf:trace>pgm!ConverseInit+2e (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\machine-common-core.c @ 1197)</avrf:trace>
  <avrf:trace>pgm!charm_main+41 (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\init.c @ 1713)</avrf:trace>
  <avrf:trace>pgm!main+1b (c:\msys64\home\evan\charm\multicore-win-x86_64\tmp\main.c @ 6)</avrf:trace>
  <avrf:trace>pgm!invoke_main+34 (d:\agent\__work\1\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 79)</avrf:trace>
  <avrf:trace>pgm!__scrt_common_main_seh+12e (d:\agent\__work\1\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 288)</avrf:trace>
  <avrf:trace>pgm!__scrt_common_main+e (d:\agent\__work\1\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 331)</avrf:trace>
  <avrf:trace>pgm!mainCRTStartup+9 (d:\agent\__work\1\s\src\vctools\crt\vcstartup\src\startup\exe_main.cpp @ 17)</avrf:trace>
  <avrf:trace>KERNEL32!BaseThreadInitThunk+14 ( @ 0)</avrf:trace>
  <avrf:trace>ntdll!RtlUserThreadStart+21 ( @ 0)</avrf:trace>
</avrf:stackTrace>
```

CmiState CmiGetState(void) {
}
2. "NULL handle passed as parameter. A valid handle must be used."

```
void CmiArgInit(char **argv) {
    int i;
    CmiLock(_smp_mutex); // _smp_mutex is null here
```
These global variables in TCharm and AMPI are potential candidates for causing this issue due to data races:

```c
static mpi_comm_worlds mpi_worlds;
int _mpi_nworlds;

static CProxy_amiWorlds ampiWorldsGroup;
CtvExtern(TCharm *,_curTCharm);
```

Files

<table>
<thead>
<tr>
<th>Files</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsan.log.3262</td>
<td>402 KB</td>
<td>03/15/2019</td>
<td>Evan Ramos</td>
</tr>
</tbody>
</table>