Charm++ - Bug #1539
Failure in migration when using RDMA sends
04/28/2017 09:39 AM - Sam White

Status: Merged
Priority: High
Assignee: Vipul Harsh
Category: AMPI
Target version: 6.8.0

Description
Several autobuild targets segfaulted in ampi/Cjacobi3D last night after AMPI RDMA was merged: mpi-linux-x86_64, mpi-linux-x86_64-smp, verbs-linux-x86_64, verbs-linux-x86_64-smp, and mpi-crøyxe all failed in the same place. That's both verbs targets, and most but not all MPI targets.

The test fails consistently in the first RDMA message after the first migration happens on >=3 PEs. It seems to never fail on just 2 PEs, and Isomalloc vs PUP doesn't matter.

Valgrind output on mpi-linux-x86_64 shows that an invalid read of size 8 is always happening after migration when processing an RDMA message.

What's weird is that this looks like it is actually doing the right thing: AMPI is copying memory from the recv buffer allocated in LRTS to the application's buffer (the messages in Cjacobi3D are 80,000 Bytes long).

Related issues:
Related to Charm++ - Feature #1546: RDMA example with migration

05/18/2017
Even if I run with NullLB, these two leaks still show up in Valgrind for a mpi-linux-x86_64 build (I haven't tried valgrind on any other RDMA layer).

The latter leak is fixed if I mark the entry method that receives the CkDataMsg RDMA completion callback with the [nokeep] attribute, but the RDMA documentation and examples do not use [nokeep] and they don't delete the CkDataMsg itself. If you run valgrind on examples/charm++/rdma/ you see these same two leaks.
#2 - 04/28/2017 12:01 PM - Sam White
The second leak above also happen on multicore builds... opened a separate issue for memory leaks in rdma:
https://charm.cs.illinois.edu/redmine/issues/1540

#3 - 04/28/2017 07:29 PM - Sam White
- Priority changed from Normal to High

Still no real diagnosis of the underlying problem here...

```
==7618== 13770 errors in context 4 of 56:
==7618== Invalid read of size 8
==7618==   at 0x4C2F790: memcpy@@GLIBC_2.14 (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==7618==   by 0x677EE4: serializeContig(char*, char*, unsigned long, int) (ddt.C:17)
==7618==   by 0x679844: CkDDT_DataType::serialize(char*, char*, int, int) const (ddt.C:549)
==7618==   by 0x65E4BE: ampi::processRdmaMsg(void*, int, int, void*, int, int) (ampi.C:2855)
==7618==   by 0x631684: IReq::receiveRdma(ampi*, char*, int, int, int, int) (ampi.C:5410)
==7618==   by 0x6287FD: ampi::inorderRdma(char*, int, int, int, int, int) (ampi.C:2599)
==7618==   by 0x6285E7: ampi::genericRdma(char*, int, int, int, int, int) (ampi.C:2553)
==7618==   by 0x64B2FD: CkIndex::call_genericRdma_marshall8(void*, void*) (ampi.def.h:3290)
==7618==   by 0x69D66E: CkDeliverMessageFree (ck.C:593)
==7618==   by 0x6B18BC: CkLocRec::invokeEntry(CkMigratable*, void*, int, bool) (cklocation.C:1969)
==7618==   by 0x6B448D: CkLocMgr::deliverMsg(CkArrayMessage*, CkArrayID, unsigned long, CkArrayIndex const*, CkDeliver_t, int) (cklocation.C:2656)
==7618==   by 0x697849: CkArray::deliver(CkArrayMessage*, CkDeliver_t) (ckarray.h:688)
==7618== Address 0x6d02710 is 105,120 bytes inside an unallocated block of size 839,152 in arena "client"
```

#4 - 05/01/2017 04:52 PM - Sam White
- Subject changed from Failure in AMPI RDMA recv path after migration to Failure in AMPI migration in TCharm::pup when unpacking

This is not related to AMPI's use of RDMA at all, and can be reproduced on commits before the AMPI RDMA patch was merged. I need to do a full git bisect to determine the cause... Autobuild failures first showed up on April 28th, but there were a ton of changes merged on the 27th...

#5 - 05/02/2017 11:11 AM - Sam White
Nevermind, the issue seems to be different on mpi-darwin-x86_64, and not being able to use ++debug on MPI builds hurts. It is related to AMPI RDMA on linux.

#6 - 05/02/2017 11:13 AM - Phil Miller
The range of commits that appear in the time window in question and look like they could be the culprit is as follows:

```
e276540 Bug #1530: move isomalloc_sync warning to memory-isomalloc.
9ddf086 AMPI: nicer abort message.
27e9038 AMPI #1163: use disp_unit from window creation
#465ccce charmxi #1527: Add dummy static_cast to impl_e_opts to avoid compiler warning
```

05/18/2017
76aa31 AMPI: fix accumulate routine's ptr arithmetic and eliminate unused arguments
05e5a4b AMPI: support derived datatypes in Alltoall and nbor coll routines
#5d303c charmxi #1526: Remove unused parameter created in migration constructor
#fda165f AMPI: fix Sendrecv_replace and assume Sendrecv buffers are not the same
#38deb5f Bug #1528 charmxi: Message type with no variable arrays can't be declared with { }
#e351ad Bug #1529: Easy 'nolb' build-time option to elide LB support
d12a6a7 Bug #1493: Fix the original patch by moving CkLocRec deletion
a0d9ed5 AMPI: fix memory leak in tuple redn handling for gathers and non-commutative redns
9cb75c AMPI: relax ordering of sends/recvs in Alltoall(v,w) and nbor coll routines
9349919 AMPI #1111: avoid sender-side copy for large contiguous messages using Charm++'s zero copy RDMA send API
#ebb101f Bug #1487 Enable cuda compilation without GPU_MEMPOOL being defined
#79c979 Bug #1488 Restrict maximum initial pinned memory to 1 GiB
#e90d142 Bug #1506: Fix: examples/hello/4darray wasn't using messages for multicast
f4d1581 Bug #1410: fix memory leak in Tuple reducers consisting of set(concat/custom reducers
$ git log --oneline f4d158^..e27654 | wc -l
 20

I've #ed out the commits that seem like they can't be related, leaving just 12 suspects. That should be about 4 builds to bisect

---

#7 - 05/02/2017 11:17 AM - Sam White
The culprit is the AMPI RDMA patch (below) on mpi-linux-x86_64:

9349919 AMPI #1111: avoid sender-side copy for large contiguous messages using Charm++'s zero copy RDMA send API

---

#8 - 05/02/2017 04:02 PM - Sam White
- Subject changed from Failure in AMPI migration in TCharm::pup when unpacking to Failure in AMPI migration when using RDMA sends

---

#9 - 05/04/2017 09:24 AM - Sam White
I think I've found the problem, though I have no idea why it's happening: RDMA messages are not being forwarded to objects after they have migrated.

Basically I've added a print statement at the very top of ampi::genericRdma (the rdma entry method where this is failing), and after migration the buffer passed into the method is bad and the segfault happens in the print after migration.

Here's the output from ckarray.C, cklocation.C, and ampi.C for examples/ampi/Cjacobi3D/ for 8 VPs on 3 PEs with RotateLB on an mpi-linux-x86_64 build. The failure happens on VP 6 after it migrates from PE 2 to PE 0:

... LocMgr on 2: Load balancer wants to migrate 6 to 0 LocMgr on 2: In CkMigratable::pup 6 LocMgr on 2: Removing barrier for element 6
LocMgr on 2: Destroying record for element 6
LocMgr on 2: Unregistering element 6 from load balancer
LocMgr on 2: In CkMigratable::~CkMigratable

[1] AMPI vp 2 send: tag=1, src=2, comm=9000000 (to 6)  <--- RDMA message send
LocMgr on 1: send 6

LocMgr on 0: Adding new record for element 6
LocMgr on 0: Registering element 6 with load balancer
LocMgr on 0: Element 6 migrated in
LocMgr on 0: Constructing element 6 of array
LocMgr on 0: Invoking entry 178 on element 6
LocMgr on 0: Constructing element 6 of array
LocMgr on 0: Invoking entry 140 on element 6
LocMgr on 0: Constructing element 6 of array
LocMgr on 0: Invoking entry 300 on element 6
ArrayBOC on 0: ArrayElement::pup()
LocMgr on 0: In CkMigratable::pup
LocMgr on 0: Registering barrier client for 6
LocMgr on 0: Registering barrier client for 6
LocMgr on 0: Registering barrier client for 6

In mpi machine.c on 2, handling a recv'ed RDMA msg 0x2a06860  <--- recv'ed RDMA address on PE 2

2 updateLocation on 1
LocMgr on 1: Sending update back to 2 for element 1

LocMgr on 0: DummyResumeFromSync called
LocMgr on 0: dummyAtSync called
2 updateLocation on 1
CharmLB> RotateLB: PE [0] step 0 finished at 174.775194 duration 0.059976 s

LocMgr on 0: Sending update back to 1 for element 6
LocMgr on 0: Invoking entry 149 on element 6
[0] in ampi::genericRdma on index 6, size=80000, seq=2, srcRank=2, tag=0, comm=9000000, ssendReq=0, buf 0x2a06900  --- recv'ed RDMA address on PE 0

------------- Processor 0 Exiting: Caught Signal -------------
Reason: Segmentation fault

[0] Stack Traceback:
[0:0]  [0x7a555f]
[0:1] +0x10330  [0x7ffff6e11330]
[0:2] _ZN4ampi11genericRdmaEPciiiiii+0x8c  [0x625d1e]
[0:3] _ZN12CkIndex_ampi27_call_genericRdma_marshall8EPvS0_+0x158  [0x6482ba]
[0:4] CkDeliverMessageFree+0x53  [0x68cf03]
[0:5] _ZN8CkLocRec11invokeEntryEP12CkMigratablePvib+0x220  [0x6add30]
[0:6] _ZN7CkArray7deliverEP14CkArrayMessage11CkDeliver_t+0x68  [0x6939de]
[0:7] _ZN7CkArray7deliverEP14CkArrayMessage11CkDeliver_t+0x68  [0x6939de]
[0:8]  [0x68f04e]
[0:9] _Z15_processHandlerPvP11CkCoreState+0x222  [0x68f272]
[0:10] OmCHandleMessage+0x54  [0x7a787b]
[0:11] CsSchedu[...]
[0:12] CsSchedu[...]
[0:13]  [0x7a723ffca]
[0:14] ConverseInit+0x3ac  [0x7a3ee8]
[0:15] main+0x44  [0x67a469]
[0:16] _libc_start_main+0xf5  [0x7ffffff6a5af45]
1 updateLocation on 0

Note the two addresses differ by 160 Bytes, which I've verified is the size of the message envelope. So it looks like the message is not being forwarded for some reason.

The failure doesn't show up when using multicore or MPI-SMP (with 1 process) builds, even though the AMPI code path is exactly the same regardless. This makes sense since the RDMA message pointer would still be valid within a process.

I can't see how this could be an AMPI issue, since the buffer is bad before it is ever passed into AMPI, though I can't rule out that AMPI is doing something weird in migration that then trips this up. But until we have any proof that RDMA + migration works, I think we should assume it is not working. I opened issue #1546 to get an example RDMA + migration program.

05/18/2017
To reproduce this, do `./build AMPI mpi-linux-x86_64 -g -O0` then `make test` in examples/ampi/Cjacobi3D/.

Basically the problem arises when a chare does an rdma send to another chare, and then the receiver chare migrates between the time when the rget finishes and the receiver actually has the entry method invoked on it. If that happens, then the message needs to be forwarded to the receiver's new PE, which happens but for some reason the rdma parameter's ptr still points to the address at which it recv'ed the rget, not to a new address.

I looked at the code and it never actually changes the pointers inside the rdma wrappers in the message, hence the rdma pointers are invalid in another PE. I am trying to think of a fix.

If that's all, I think the message just needs to have pack called in it before it gets forwarded, and unpack after it arrives.

Patch: [https://charm.cs.illinois.edu/gerrit/#/c/2520/](https://charm.cs.illinois.edu/gerrit/#/c/2520/)

Assignee changed from Sam White to Vipul Harsh

Status changed from Implemented to Merged