RDMA zero copy send implementation for multicore builds

Currently, the RDMA send API uses the "generic" implementation in multicore builds. The generic implementation does the extra copy and sends the message like a normal parameter marshalled entry method invocation. This is a waste because in multicore we are guaranteed that the recv'er is in the same process as the sender, so we can do the direct memcpy from source buffer to receiver message that we already do on PAMI, GNI, Verbs, and MPI layers when the sender and receiver are co-located in the same logical node.

History

#1 - 04/05/2017 03:03 PM - Sam White
- Parent task set to #1459

#2 - 04/05/2017 03:29 PM - Phil Miller
Isn't this already covered by the generic SMP support? And even if it isn't, any implementation work here would just change which thread is allocating and performing the copy, not how much data needs to be allocated or copied. I.e. Either the sender sizes and packs stuff into a marshalled message, and then passes a pointer to that message to the recipient, or the sender sizes and passes a metadata message to the recipient, which then allocates and copies the data into a message.

Am I missing something?

#3 - 04/05/2017 05:02 PM - Sam White
I assume what you mean by "generic SMP support" is the direct memcpy of the source buffer to a message on the recv'er, but that is only done when CMK_ONESIDED_IMPL is defined, which we don't currently do on netlrlts (which multicore uses). There are no exceptions to that for multicore currently. But your point about the only difference being who does the copy is true, I didn't think about that. Should we mark this 'rejected'?

#4 - 04/05/2017 05:08 PM - Phil Miller
It suggests that the fix for this issue is very easy - just add

```c
#if CMK_MULTICORE
#define CMK_ONESIDED_IMPL 1
#endif
```

At some appropriate spot, with a comment describing our discussion here.

#5 - 04/05/2017 05:09 PM - Phil Miller
Though it may make sense to separate the operability of the SMP implementation from the network-layer implementation, since this applies...
within-process even on layers like GNI where we don't have them working & integrated yet.

#6 - 04/05/2017 05:20 PM - Sam White

Just defining CMK_ONESIDED_Impl to CMK_MULTICORE is not enough. When you link something that uses RDMA it fails with this:

```bash
../../../../bin/charm --language charm++ -o pgm pingpong.o
../../../../bin/libck.a(ckrdma.o): In function `CkUpdateRdmaPtrs(envelope*, int, char*, char*)':
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:223: undefined reference to `CmiSetRdmaRecvInfo'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:227: undefined reference to `CmiGetRdmaGenRecvInfoSize'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:233: undefined reference to `CmiSetRdmaRecvOpInfo'
../../../../bin/libck.a(ckrdma.o): In function `CkRdmaCreateMetadataMsg(envelope*, int)':
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:47: undefined reference to `CmiGetRdmaInfoSize'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:62: undefined reference to `CmiSetRdmaInfo'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:63: undefined reference to `CmiGetRdmaGenInfoSize'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:73: undefined reference to `CmiSetRdmaOpInfo'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:76: undefined reference to `CmiSetRdmaAck'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:87: undefined reference to `CmiGetRdmaOpInfoSize'
../../../../bin/libck.a(ckrdma.o): In function `CkRdmaIssueRgets(envelope*)':
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:179: undefined reference to `CmiGetRdmaInfoSize'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:191: undefined reference to `CmiGetRdmaRecvInfoSize'
../../../../home/swhite/charm/multicore-linux64/tmp/ckrdma.C:204: undefined reference to `CmiIssueRget'
```

But we can provide no-op implementations of those, at least until netlrts has actual rdma support...

#7 - 04/05/2017 06:04 PM - Sam White

- Tags changed from #rdma, #machine-layers, #lrt to #rdma, #machine-layers, #lrt, #smp
- Assignee changed from Nitin Bhat to Sam White
- Parent task deleted (#1459)

Implemented multicore RDMA support in netlrts here: https://charm.cs.illinois.edu/gerrit/#/c/2358/

#8 - 04/11/2017 03:55 PM - Sam White

- Status changed from New to Implemented