NodeGroup Broadcasts creates many copies of the message for point to point sends

Broadcast to a nodegroup results in the message being copied once for each destination node. These are then sent in a loop.

This results in numnodes copies of the message, which has a time, space, and bandwidth cost proportional to the size of the message and the number of hosts.

For large messages (where large varies by machine layer, but 128k should be fine) this should be replaced by an RDMA scheme that arranges for each destination to RDMA get the payload. Thereby reducing the space and time cost. This could be further enriched by a spanning tree approach to reduce the single link bandwidth cost, though at the price of increased latency at the leaves. Probably best to cut over to that scheme when the number of hosts is large, and/or the size of the message is very large (~1G). This should be implementable using our existing zero copy semantics to avoid writing a new implementation at each machine layer. If not, then we should extend our semantics to facilitate this kind of usage.

An even more advanced scheme would use message layer broadcast primitives. However, their applicability and portability has its own research agenda, so a distinct subtask should be created for that by whoever goes that direction.

Related issues:
Related to Charm++ - Feature #1184: SMP-safe CmiReference and CmiFree

History
#1 - 07/14/2017 02:21 PM - Phil Miller
- Related to Feature #1184: SMP-safe CmiReference and CmiFree added

#2 - 07/14/2017 02:26 PM - Phil Miller
This all boils down to src/arch/util/machine-broadcast.c and the setting of CMK_BROADCAST_USE_CMIREFERENCE in src/conv-cor/conv-config.h. It's default set to 0, because CmiReference and CmiFree would have to use atomic increment/decrement operations. We should make a version of those that uses the atomic operations, and performance test them on applications. My expectation is that they shouldn't hurt anything, and make a lot of other things way easier.

The important characteristic to note is that there's nothing in the messages themselves (including any header fields) that should change from one recipient to the next. It looks like the 'destination rank' field does get set in SendSpanningTreeChildren.

There's also the concern about recipient PEs unpacking the message once they have a pointer to it. Depending on whether we want to optimize for latency or memory footprint, there's a tradeoff between making a copy for local delivery while sending the still-packed copy off to other nodes, or waiting for the remote sends to complete and then locally delivering the original.