Derecho 09/17

Sagar Jha

September 23, 2018
Overview

1. Source code structure
2. Derecho group
3. Compilation
4. Logistics
Directory structure

/  
  
  derecho .............. Main group functions, RPC/P2P layer
  experiments .. Derecho test files (unit tests, performance tests, other experiments)

  sst .......... Shared state table for control plane operations
  experiments .............. basic SST experiments/tests

  rdmc....... RDMA Multicast (data plane) - for sending large messages, to many receivers

  tcp ....................... Code for forming TCP connections

  conf ....... Derecho configuration (provider, domain, ports)

  persistent ....................... Code for persisting data

  time .......................... Simple timing functions

  third_party....... Third-party libraries, as git submodules
Important files

```
/
  derecho
    group.h, group_impl.h.... Defines Group class and its functions
    view_manager.h, view_manager.cpp... Manages views (memberships) and view changes
    derecho_sst.h, derecho_sst.cpp Defines the columns for the SST table
    multicast_group.h, multicast_group.cpp. Manages RDMC and SST multicast groups for the subgroups. Implements logic for sending, receiving and delivering messages
    view.h, view.cpp ........ View information, top-level membership information, member IP addresses, subviews for every subgroup
```

Sagar Jha  Derecho 09/17
Important files

/  
  derecho
     p2p_connections.h  p2p_connections.cpp  Implements separate RDMA connections between every pair of nodes
     replicated.h.........  Implements ReplicatedObject - encapsulating user-defined subgroup classes, providing functions send_object (over the network), ordered_send_or_query, p2p_send_or_query
     rpc_manager.h, rpc_manager.cpp .... RPC functions: Serialization, de-serialization of subgroup functions, handling function invocations and collecting replies
     subgroup_info.h, subgroup_functions.h, subgroup_functions.cpp..... Defines SubgroupInfo for specifying subgroup structure
Third-party libraries

```
/   
  third_party
     libfabric ......... Common interface to RDMA fabric technologies, by OpenFabrics Alliance
     mutils, mutils-containers, mutils-serialization. Serialization, De-serialization of C++ objects, by Matthew
     spdlog ................. Asynchronous logging library
```
Input (for each node):

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>ID, IP, leader IP</td>
</tr>
<tr>
<td>Type 2</td>
<td>ID, num_nodes, IP for all nodes</td>
</tr>
</tbody>
</table>

Table 1: Type 1 is recommended.

Group constructor parameters:

<table>
<thead>
<tr>
<th>Starting leader</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting leader</td>
<td>my_id, my_ip, callbacks, subgroup_info, derecho_params, ...</td>
</tr>
<tr>
<td>Starting non-leader</td>
<td>my_id, my_ip, leader_ip, callbacks, subgroup_info, ...</td>
</tr>
</tbody>
</table>

Table 2: derecho_params is transferred by the leader to other nodes. subgroup_info must be the same on all the nodes! (it cannot be serialized, so it is not sent by the leader)
### Constructor parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_id</td>
<td>unique ID of the node</td>
</tr>
<tr>
<td>my_ip</td>
<td>IP address of the node</td>
</tr>
<tr>
<td>leader_ip</td>
<td>Leader node’s IP address</td>
</tr>
<tr>
<td>callbacks</td>
<td>Set of two callbacks, first being a function called at message delivery</td>
</tr>
<tr>
<td>subgroup_info</td>
<td>Specifies subgroup structures, contains a map from each subgroup type to its membership function</td>
</tr>
<tr>
<td>derecho_params</td>
<td>Specifies max message size, block size, ...</td>
</tr>
</tbody>
</table>
RawObject: Special subgroup type

- Has no associated state
- Can only send and receive messages
- Associated subgroups are called Raw Subgroups
Compilation

- Out of source compilation (for example in /Release or /Debug)
- Different modes of compilation: Release for production code, Debug for debugging
- To compile a new test file
  /derecho/experiments/new_test.cpp, add the following lines to /derecho/experiments/CMakeLists.txt:
  add_executable(new_test new_test.cpp)
  target_link_libraries(new_test derecho)

Add all the .cpp files for your test (ideally, only one) in the add_executable command. target_link_libraries command links your executable to the derecho library.
Next meeting

- Sending and receiving in raw subgroups
- Typed subgroups
- More examples of defining SubgroupInfo
Find this slide and other material on https://www.cs.cornell.edu/~sagarjha/derecho-cs4999

Find the sample application code developed in the meetings in /derecho/experiments/sample_application.cpp in the derecho-cs4999 branch (https://github.com/Derecho-Project/derecho-unified/blob/derecho-cs4999/derecho/experiments/sample_application.cpp)