Chained GPC

Regular GPC executes a function on the remote side and returns an acknowledgment and data upon completion. Chained GPC is defined as follows:

Function interface and parameters

Send Request to Execute a chained GPC

- func - handle to the function executed at each process in the chain
- callba- handle to the callback to be executed when
- hdr - header data - used to pack extra args for callback (local buffer)
- hlen - size of header data < ARMCI_GPC_HLEN
- data - bulk data passed to callback (local buffer)
- dlen - length of bulk data
- rhdr - ptr to reply header (return args from callback)
- rhlen - length of buffer to store reply header < ARMCI_GPC_HLEN
- rdata - ptr to where reply data from callback should be stored (local buf)
- rdlen - size of the buffer to store reply data
- idlen - number of ID's
- idslst- list of id's in the chained GPC
- nbh - nonblocking handle which also acts as a context for each individual GPC
- Tree - the id of tree function used (default is 0=>binary, 1=>binomial, n=> user defined)

```
int ARMCI_Gpc_chained_exec(int func, int callback, void  *hdr, int hlen, void *data,
        int dlen, void *rhdr, int rhlen, void *rdata, int rdlen,
        int idlen, int *idlst, gpc_hdl_t* nbh, int TREE)
```

Description

ID : a set of id's { n_0....n_k }
rood_id : initiator of GPC
$parent\_id_i$: the parent of an ID i.
$CHILD\_ID_i$: a subset of the set ID representing set of ID's that are children to id i.

A chained GPC's propagation works in the following steps

1. root_id executes func and subsequently initiates a chained_gpc $\forall\, n_c$ s.t. $n_c \in CHILD\_ID_{root\_id}$
2. For each id $n_i \in$ ID
    1. Receive GPC from parent_id
    2. Executes func: func(func,callback,hdr,hlen,data,dlen,rhdr,rhlen,rdata,rdlen,idlen,idlst,Tree);
    3. initiate a chained_gpc $\forall\, n_c$ s.t. $n_c \in CHILD\_ID_i$

The action for any response to a chained_gpc with handle nbh

1. execute callback function:
   callback(from, hdr, hlen, data, dlen, rhdr, rhlen, rdata, rdlen, idlen, idlist, nbh);
2. look in nbh if responses from all children received, if yes:
   1. If no data expected as a part of response (i.e. All of rhlen, rhdr, rdlen, rdata are NULL) merely send a GPC completed response.
   2. if data expected as a part of response, respond with my rhdr,rhlen,rdata,rdlen to parent along with GPC_COMPLETED message.

Internal implementation details for a chained GPC are as follows:

1. each chained GPC can have a function and a callback.
2. Any incoming message for an id i goes through a *setup* process which includes receiving and queuing the message.
3. This *setup* process is not interruptible.
4. func and callback are interruptible but only by a message arrival and only the *setup* process for the incoming message is allowed as an interrupt handler

Alternative approach

An alternative method would be to call follow current GPC semantics and not have a separate callback. The function func will be called with the last parameter GPC_INIT to accomplish what the 'func' described above is doing and with GPC_WAIT to accomplish what the callback is doing.

Exceptions

1. Each id can have only one parent
2. Any failure at any stage in execution of GPC results in an ID sending a failure message to the parent and ignoring all subsequent responses with the same gph.
3. Current failure model only propagates success/failure and only up the tree (to the parent)