Variable MAX_TASK_SET_SIZE Proposal

December 7, 1998
Akihiro Shimura, CANON INC.

This document proposes a new command that makes MAX_TASK_SET_SIZE parameter variable (dynamically alterable) during logged-in period (between login and logout), and enables efficient target's resource usage.

1. Background and Problems

At last meeting, there was a definition of a "queue" as "an ordered set of ORB's that does not block with respect to other queues", and there was a definition of a "connection" as "a queue or two queues that affords access to a service". Also, there was a discussion that "Number of connection and number of queue are not related to MAX_TASK_SET_SIZE".

It will be possible to have many queues independently of the MAX_TASK_SET_SIZE as long as the total number of active ORB's over all queues does not exceed the value.

But the number of the queues is actually restricted (or upper bounded) by MAX_TASK_SET_SIZE because at least one task space for each queue should be preserved by the initiator to satisfy that "a queue cannot block other queues".

For example, assuming MAX_TASK_SET_SIZE equal five, and there exists six active queues, and five queues from six have pending ORB's (five ORB's), then the initiator cannot append a new ORB for sixth queue because it makes number of ORB's in the task list exceeded MAX_TASK_SET_SIZE. In this case, sixth queue is blocked by other queues.

So, during the logged-in period (between login and logout), it is always required the number of queues to be smaller than or equal to MAX_TASK_SET_SIZE to satisfy the queue definition. Without the knowledge of maximum possible number of queues prior to the connections, initiator will not be able to flexibly utilize task spaces specified by

the MAX_TASK_SET_SIZE, and may use only one task space per queue to preserve task spaces for future occurrence of connections (queue allocations). Otherwise, new connection (queue) may be blocked while connection itself may be established because target does not know if there are spaces for new queues or not.

Current profile defines MAX_TASK_SET_SIZE as read only parameter, and MAX_TASK_SET_SIZE is fixed value during a logged-in period.

In the current profile, MAX_TASK_SET_SIZE is determined by the target when logged-in (i.e., very initial stage). The target needs to preserve resources for MAX_TASK_SET_SIZE at initial stage. If target assumes maximum of 256 queues per login (MAX_TASK_SET_SIZE>=256) and assumes maximum of 16 initiator's login, required resources would be minimally 256 x 16 x 32 (assuming the size of queue entry = size of ORB) = 128KB.

Even when only one queue is activated by each six-teen initiators, 128KB of memory should be preserved for this purpose while actual required amount is 1 x 16 x 32=0.5KB.

The target may assume some smaller number of queues, say 10, and may want to offer larger MAX_TASK_SET_SIZE than assumed number of queues, say 15.

Because the initiator doesn't know the assumption made by the target, initiator may try to use these 15 spaces with 15 queues those may or may not be used in the future. At the 11th queue allocation, initiator will be denied to connect and then 5 spaces remain unused.

In summary, there are three problem here.

a) Without the knowledge of maximum possible number of queues prior to the connections, initiator will not be able to flexibly utilize task spaces specified by the MAX_TASK_SET_SIZE.

   This may be concluded that initiator uses only one task space for each queue and pipelining never happens.

b) Target needs to preserve many resources even if they are not actually used.

c) Without the knowledge of maximum number of queues target assumes, the

initiator may behave as not expected by the target and the target may waste resources.

Making initiator aware of the target assumption on number of queue via parameter may solve problem a) and c), but will not solve problem b).

2. Variable MAX_TASK_SET_SIZE Proposal

To resolve these problems, I would like to propose a new command that makes MAX_TASK_SET_SIZE parameter variable (dynamically alterable) during logged-in period (between login and logout).

The new command (to queue #0) requests target to increase/decrease MAX_TASK_SET_SIZE parameter at any time during logged-in period.

By this command, initiator will be able to manage task spaces in the target independently of connect/disconnect.

If the initiator finds that more task spaces are necessary for a new connection, initiator may ask target to increase MAX_TASK_SET_SIZE prior to connect. Target may or may not accept this command. If this command fails, the initiator may not issue new connect command.

Also, initiator may ask target to decrease MAX_TASK_SET_SIZE when it is no longer necessary.

For example, assuming current MAX_TASK_SET_SIZE as seven, and there are already six pending ORB's in the task list, initiator will be able to append "connect" command (7th command) to the task list.

Even after the "connect" command to queue #0 finished, initiator will not be able to append a command to new allocated queue because initiator needs to preserve one space for queue #0. In this case, by using this new command, initiator can increase MAX_TASK_SET_SIZE prior to connect. After increasing MAX_TASK_SET_SIZE, say 9 by this command, initiator can issue "connect" command to queue #0 and then can issue a command to new queue(s).

By introducing this command, initiator will be able to utilize task spaces specified by the MAX_TASK_SET_SIZE with currently known number of queues and will not

need to worry about future occurrence of connections.

Also, target does not need to allocate possible maximum spaces at initial stage for future connections, and can allocate spaces when asked by the initiator on demand.

Because initiator manages the MAX_TASK_SET_SIZE based on the initiator's necessity, initiator will be able to efficiently utilize target resources with known number of queues, and target does not need to worry about possible maximum number of queues for future connections at initial stage (at login time).

3. Command Example

The new command will look like following;

```
Update_Parameter (ParamID, new MAX_TASK_SET_SIZE value)
```

ParamID field will be the ID for MAX_TASK_SET_SIZE parameter.

The response for this command will be "success" or "fail".