CSS499 Undergraduate Research Spring Quarterly Report
An Enhancement of AgentTeamwork's Job Resumption/Migration Components

Fumitaka Kawasaki
June 8, 2007

My assignment of the research project is divided into six phases, and three of which have been completed in the spring quarter. Each of those phases is described in this report.

Phase 1 – GridTcp Re-Design
The major goal of this phase is to redesign GridTcp for the purpose of memory saving and thread-incurred overheads mitigation. To realize the goal, I implemented barrier synchronization in the GridTcp module. The barrier synchronization ensures all user processes to synchronize their executions when they call takeSnapshot() function; therefore, we can exclude rollback messages from snapshot because rollback messages before synchronization are no more needed. However, there are some differences between my implementation and original re-design requirement. They are summarized below.

	Original re-design requirement	My implementation
Save rollback messages in memory	No	Yes- however, they can be flushed at barrier synchronization.
Snapshot contains rollback messages	No	No
Handle connection error.	All user processes must restart their execution from the previous barrier synchronization when connection error.	Only the broken process has to restart its execution from the previous barrier synchronization when connection
Maintain snapshot at the local/temp directory	Yes	No

As the table indicates, my implementation uses more memory than the original re-design requirement because it saves rollback messages in memory. However, the memory less likely grow larger because it can be flushed when every barrier synchronization.

On the other hand, my implementation has two advantages in performance. First, it does not require all user processes to restart their execution from the previous barrier synchronization when connection error. Second, it does not need to maintain snapshot at the local/temp directory.

Modifications of GridTcp

1. Made backupQue transient so that we exclude it from snapshot.
2. Implement barrier synchronization
   - After sending commitment messages to all partner connections, the user process waits until receiving all corresponding commitments from partners.
   - To synchronize each node with commitment, a function id is added to the commitment message. The function id is saved in the communication node

when it received the commitment message, and it is compared with the id of the function that issues the takeSnapshot().

3. Modified rollback process
    - Clear the backuQue when commitment message is received. (Previously we only cleared messages whose sequence number is less than confirmSeq that is included in received commitment message.)
    - The backupQue is cleared at barrier synchronization, and it is not saved in snapshot. This means the backupQue is used only when rollback process; therefore, existsInBackup( ) (which is called from send(), sendCommit(), and sendEof()) was eliminated.
    - Rollback all messages in the backupQue. (Previously we need a sequence number that is a parameter of rollback() function. When rolling back messages, only messages whose sequence number is greater than the sequence number is sent.)
    - Added commitment message to the backupQue to make sure commitment is sent when rolling back.

4. Modified commitment message handling
    - Added error check after sending commitment message.

Phase 2 – Sentinel Agent Modification

There are three work items of modification:

1. IP Caching: IP cashing was already implemented in the current sentinel code.
2. Direct snapshot transfer by the wrapper: eliminate the thread for calling sendSnapshot() function so that a user program wrapper directly calls the function.
3. Faster TCP error correction: eliminate the thread for TCP error correction. Modified sentinel agent to handle TCP error correction directly when "restart_userprogram" is received from a resumed process.

Phase 3 – Job Resumption Test and Debugging

1. Intra-cluster job resumption test
    - Number of nodes: 4, number of bookkeepers: 1, number of extra nodes: 2, test applications: MasterSlaveAteam.java, HeartBeat.java, Bcast.java, and AllReduce.java – confirmed that crashed sentinels (up to two sentinels) can be resumed.
    - CommanderAgent argument:

      java -Xmx512M UWAgent.UWInject -p 54321 localhost AgentTeamwork.Agents.CommanderAgent -m 4 -j jars/Agents.jar,benchmark/Benchmark.jar -u AgentTeamwork/Agents/ -up benchmark/ S_medusa_mnode0_mnode1_mnode2_mnode3 B_mnode6 E_mnode4_mnode5 U_$1_10000_20000_3 C_Timer AP_60002

2. Inter-cluster job resumption test
    - Number of nodes: 4, number of gateways: 2, number of bookkeepers: 2, test applications: MasterSlaveAteam.java – confirmed that crashed sentinels (up to two sentinels) can be resumed except gateways.
    - CommanderAgent argument:

      java -Xmx512M UWAgent.UWInject -p 54321 localhost AgentTeamwork.Agents.CommanderAgent -m 4 -j jars/Agents.jar,benchmark/Benchmark.jar -u AgentTeamwork/Agents/ -up benchmark/ S_perseus CL_mnode3_mnode3_mnode0_mnode1 CL_priam_priam_uw1-320-00_uw1-320-01 ECL_mnode3_mnode3_mnode20_mnode21 ECL_priam_priam_uw1-320-20_uw1-320-21 ECL_mnode22_mnode22_mnode23_mnode24 E_mnode10 B_phoebe_uw1-320-31 U_$1_10000_20000_3 C_Timer AP_60004

3. Current issues
    - When the second sentinel node is killed at the same execution cycle of the first killed sentinel node, the resumption of the second sentinel will fail. The reason of this problem is that the second resumed sentinel node cannot get the first resumed node name. To fix this problem, we need to main the IP-table in the commander agent.
    - When a gateway node is killed, the resumption of the gateway and its cluster nodes will fail.

The following list is the brief description of modifications/bug fixes about resumption process.

Changed the start index of wrapperArgs from 1 to 2 because port number was added to wrapperArgs. (resume() in SentinelAgent.java)

Inherit previous Ateam instance when resumption. (initUser() in UserProgWrapper.java)

Added resumption check at MPJ.Init() to skip GridConnection initialization when restart.

Modified GridErrorhandler so that it can handle multiple connection errors simultaneously.

Set ‘-‘ to wrapperArgs when gateName() returns null. (updateAgentLocation() in AgentUtil.java)

Delete space from gateway name. (run() in UWPlace.java)

Fixed calculation of bookkeeper id from rank. (mapSentinelRanktoBookkeeperId() in AgentUtil.java)

Handle commitment message in gateway. (GridGatewayThread.java)

Fixed calculation of rank from id. (calculateSentinelRankFromId())

Ignore second search snapshot until restartUserProg() is called.

Fixed calculation of child sentinel id. (resume() in SentinelAgent.java())

Reset ping timer when resumption. (Make lastPingTime transient in AgentUtil.java)

Continue to send restart user program messages after error. (broadcastRestartUserProg())