SYNCHRONIZING THE REPLAY ENGINE WITH THE RECORD ENGINE FOR X86 ARCHITECTURE
An Interim Report (11/7/2006)

SOLVED ISSUES

- Old replay is in much better shape than the new replay. Much of the functionality is not implemented in new replay like basic parsing for MPI_Send, MPI_Recv etc. This means that we can only look at using old replay with the old record and then finding the minor modifications that may be there in the new record to make the corresponding changes in old replay to get it in sync with new record.
- It was found that the old record had the functionality of relative offsets already built in. But the old replay was not handling it correctly. Some minor changes have been made in the replay engine to this effect and we have found that small programs involving MPI calls like MPI_Send, MPI_Recv, MPI_Barrier, MPI_Reduce are working with our modifications.
- We were able to successfully compile old record and old replay. We needed minor modifications in the file mpi_wrappers.c. This fix was already done in the new record and it has been merged. Some minor changes with regards to changing the environment variables in the Makefiles were also done.

CONTRIBUTIONS BY MEMBERS

APOORVA KULKARNI - was instrumental in compiling both the record and replay engines and doing the major modifications needed in the Makefile, also performed testing with the modified replay engine and he is maintaining our web page.

VIVEK THAKKAR - tested and debugged the old replay engine and made some useful modifications in files prsd_utils.c and request_handler.c to get the framework in running state, also has made this report.

Valuable suggestions have been made by both the members in helping each other out with their respective tasks.

OPEN ISSUES

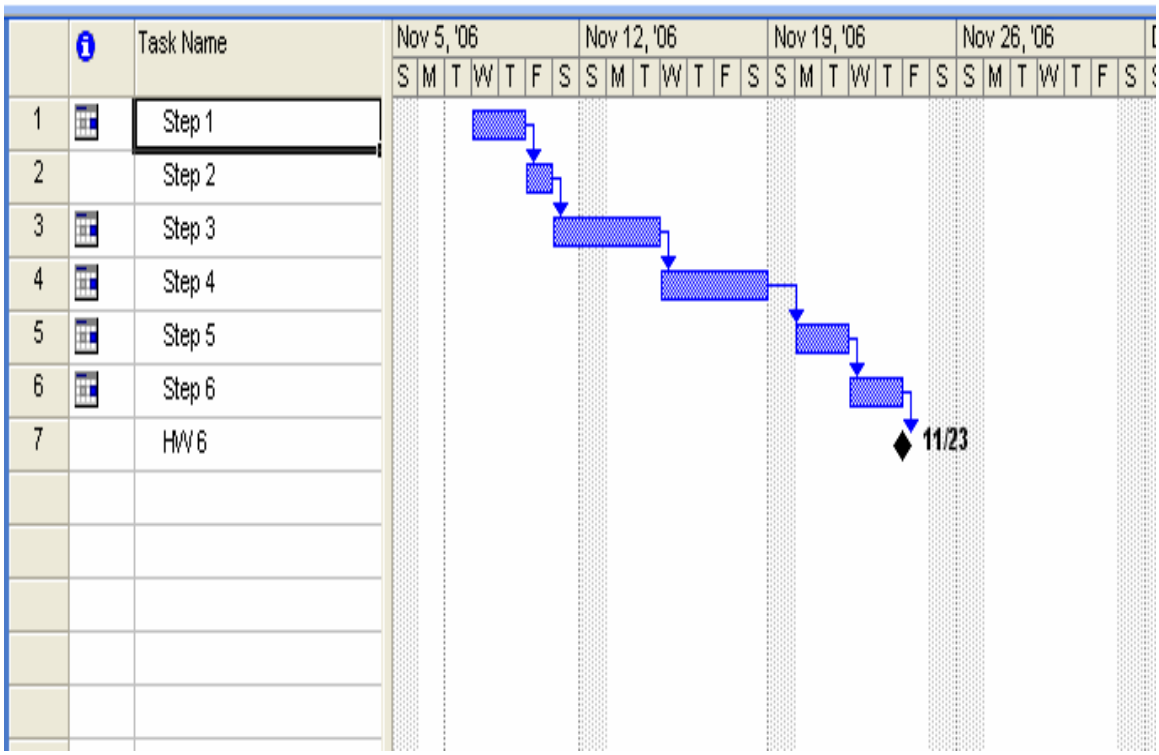
1. Asynchronous calls are not working with old replay. Currently it looks that the record and replay engine are not in sync here as the replay engine is not able to retrieve the request handle that it has stored in global static array earlier. This is probably because there is difference in the perception of record and replay with regards to handling asynchronous calls like MPI_Isend and MPI_Irecv.
2. On the similar lines, support for the other remaining MPI calls will need to be incorporated.
3. Planned testing with NAS and ASCI benchmarks will have to be done to finally ensure the sync of old record and old replay.

4. We have found out that the old record has some functionality like handling relative offsets. Therefore, we expect the differences between old record and old replay to be very minimal. Nonetheless, the planned task of identifying the changes needs to be done quickly. Some of this task will be inherently accomplished in steps 1 and 2.
5. Implementing the corresponding changes in the old replay would be the next task and it should not be a very big effort.
6. Finally, testing the resultant code with small and large benchmarks (and verifying with mpiP) shall need to be done.

TASK DISTRIBUTION AND TIMELINES FOR THE ABOVE TASKS

APOORVA KULKARNI – major responsibility in handling tasks 3, 4 and 6
VIVEK THAKKAR- major responsibility in handling tasks 1, 2 and 5.

The planning chart can be shown as below:



LINK TO OUR WEBPAGE
http://www4.ncsu.edu/~askulkar/mpitrace.html

REFERENCES

[1] Class Slides and Paper by M. Noeth, F. Mueller, M. Schulz, B. de Supinski, Scalable Compression and Replay of Communication Traces in Massively Parallel Envrionments, submitted to IPDPS 2007
[2] Dynamic Software Testing of MPI Applications with **Umpire**
www.llnl.gov/CASC/people/vetter/pubs/sc00-umpire-vetter.pdf
[3] ASCI purple benchmark
http://www.llnl.gov/asci/purple/benchmarks/limited/code_list.html
[4] NAS parallel benchmark
http://www.nas.nasa.gov/Resources/Software/npb.html