Background

- Fault tolerance techniques enable a system to continue operation in the event of a failure or a fault.
- Checkpointing consists of storing the snapshot of the current application state and using it for restarting the application in the event of a failure.
- Rollback Recovery: the process state is reloaded to memory from the last checkpoint and the execution resumes.
- Reliability Micro Kernel (RMK), a loadable kernel module for providing application-specific reliability mechanisms, is used to deploy the application checkpointing.

Research Plan

- Learning the Kernel Module Programming and writing small test modules to get familiar with it.
- Understanding the Reliability Micro Kernel framework (RMK) and the code implementing Checkpointing in the framework.
- Leveraging the RMK interface and developing the Checkpointing kernel module.

Goals

- Developing a loadable kernel module for LINUX systems:
  - Checkpoints the application periodically.
  - Recovers the application on a failure from the last saved checkpoint.
- Demonstrate the Kernel Module on some real world applications.

Fundamental Questions/Challenges

- What is the essential memory and processor state needed to recover the application from failures?
- How to detect an application process failure in the kernel mode?
- How to make the checkpoint to be transparent to the application?

Research Results

- Successfully implemented the Checkpointing Module in LINUX.
- Demonstrated Kernel Module with the real world application, GCC compiler, by:
  - Running the compilation process in the background.
  - Issuing a kill signal to terminate the process.
  - Initiating the recovery.
- Demonstrated the recovery of the application with the same Process ID.

Related Work/Interaction with Other Projects

- Reliability Micro Kernel: An OS level framework for providing Application-aware reliability.
- M-Checkpointing: Transparent Local and In-place checkpointing for propagation-limited errors.