Chapter 2
Related Work

A large percentage of today's HPC efficiency and resources are wasted due to various failures. These failures significantly affect the capacity of HPC clusters. These may occur due to either HPC system failures or the application level failures. The logs generated by the clusters are abundant in number and mining these logs to detect the error is very cumbersome.

Existing research work investigates system diagnostic logs and primarily focuses on detecting the faults generated with HPC system as a frame of reference. A large number of mining tools have been developed to study and analyze the logs. Martino et al. [6] studied the impact of system failures on the Blue Waters. Their study emphasized the effect of these failures on the available node hours. To study and analyze these failures, they have developed *LogDiver* which is a tool to automate the system log data processing. *LogDiver* handles large amount of textual data extracted from the system and application level logs and decodes the specific types of events and exit codes. They have shown that the failing applications contribute to 9% of the total node hours affecting the system resources. Their study also shows that probability of application failures due to HPC system failures is just 0.162. The detection of errors and failures at the application level is therefore inevitable and testing the faults with application as a frame of reference is equally important.

The current ongoing research primarily focuses on providing fault tolerance strategies with an objective to minimize the fault and failure effects on supercomputer resources. These studies show how to combat the effect of system failures and predict them to minimize the errors and faults. Chuah et al. [9] studied the root causes for the failures using system logs of the Ranger supercomputer at the Texas Advanced Computing Center (TACC). *FDig*, a diagnostic tool had been developed to extract the log entries as structured message templates to analyze the faults. *FDig* detects the frequency of the specific errors by extracting the error template messages from the stored system logs and supports system administrators in the fault diagnostic processes.

Gainaru et al. [10] extensively exploited the concepts of data mining techniques to determine system errors from the logs generated by Blue Gene/L machine. According to Papers [6], [8], [9], [11], console logs are a primary source of information about the condition of a cluster or HPC system. These error logs help system administrators to analyze the causes of system failures.

A study by Oliner et al. [12] on the system logs from five supercomputers, namely, Blue Gene/L, Red Storm, Thunderbird, Spirit and Liberty which analyzed failure alerts and actual failure shows that a large number of alerts are generated due to hardware issues, but most of the system failures are due the software issues. A possible explanation for system failure may also be due to software upgrades. The applications running on these resources might not be compatible with the software upgrades and hence leading to their failure. Our thesis is also capable to detect the application failure due to HPC system upgrades.

All of the novel approaches discussed earlier in this section describes the strategies to anatomize the system logs and error log files to detect the system errors. These techniques make use of past logs to predict the plausible causes of failure. There is a window of opportunity for further improvement in the efficiency of supercomputer resources if we can minimize the errors during application job submission and faults occurring at the application level. The large sector of computing resources is used by MD simulations. These applications are responsible for complex simulation and analysis of various molecular structures. There is always a major cost associated if these MD applications fail. The scientific simulation workflows such as Amber, CoCo, etc. are the major tools in the field of MD domain. These workflows serve as input to various tools such as Radical EnsembleMD [24] and RepEx [25]. With the advancement in these workflows, it is highly possible to have software bugs. In order to minimize the effect of application failures due to source code bugs, improper environment loading or transferring incorrect input files, we have developed a testing tool, *Simple Application Testing Lite (SATLite)* to detect the application deployment level and run-time errors. As discussed earlier, SATLite also uses log mining approach to detect faults and exceptions. As discussed in Paper [6], failed applications contribute majorly towards the wastage of supercomputer resources, SATLite aims at minimizing these failures before distributing it to the end user, and thus saving large number of node hours.

The developers of scientific workflows can directly check for errors while their software is still under development, before releasing it to their user base. The dependency of these scientific simulation packages on different operating systems and HPC clusters remains undetected unless a full compilation is made, and errors with"make clean" [13] can build successfully on a developer's machine, but can fail on user machines. Betz et al. [13] have discussed the effect of application bugs with respect to Amber development in depth. Since, major research in the field of HPC fault-tolerance system uses logs to predict or combat the failures at the supercomputer end, we make use of these logs with an aim to detect the application failures both due to supercomputer issues as well as providing unit testing results for Radical tools.

pyPcazip [29] is a Python based analysis tool for MD simulation data similar to Radical EnsembleMD and RepEx. pyPcazip has a testing suite that tests all the APIs with different inputs on different HPC clusters. Similar test framework has been implemented for Mist [30], a C++ library for the MD simulations. The test framework developed for Mist generates test report for each testing cycle providing developers a detailed analysis of all the software bugs and failures. Radical Pilot [26] serves as the basic layer for EnsembleMD toolkit and Repex. Radical Pilot has a fully developed functional test [28] framework to test all its components and it is able to capture most of the software bugs. We have designed the testing framework for EnsembleMD toolkit and RepEx on the similar tracks separating each iteration of test in different report folders based on date and time. Also, different parts of the API tests can be invoked independently. The study of different test frameworks and their implementation provided a background for developing a testing framework for Radical EnsembleMD and Radical Pilot. Using the available research, this thesis accommodates log mining based fault detection approach from MD application's perspective and provides unit testing results of Radical EnsembleMD and RepEx to aid Radical developers with a motivation to optimize the use of HPC resources.