Rapid improvements in network and processor performance are revolutionizing high performance computing, transforming clustered commodity workstations into the supercomputing solution of choice. Such a cluster computing system may consist of numbers of workstations and PSs of different hardware and software specifications. These environment is commonly available in industries and academic and can be use to reduce the processing time of many computational science applications.

The main problem with such type of cluster computing environment is the continuous changing in the diversities of performance of individual WSs/PCs which requires an effective task partitioning, scheduling, and load balancing to get better performance.

This dissertation presents empirical studies of task predictions of computing requirement of abrasively parallel application, thread based parallelizing techniques, partitioning, load balancing, and scheduling strategies for heterogonous distributed computing system for image computing applications.

The systems was developed using the Sun Microsystems ONC RPC and XRD libraries, PVM and MPI message passing packages. The heavy weight and light weight processing techniques for controlling the execution on remote machine. The raytracing images were used to evaluate the performance of different task partitioning and load balancing strategies. The following characteristics of the heterogeneous distributed image computing system were observed:

- In this study, we examine the effectiveness of task partitioning, scheduling, and load balancing strategies for image (raytracing) processing application on Heterogeneous Distributed Computing (HDC) system.

- In HDC system, the static and dynamic/Runtime Task Scheduling (RTS) strategies are used to balance the loads among the Workstations (WSs) / Personal Computers (PCs). Since WS/PC have a performance variation characteristic, therefore, the static task distribution is not effective for HDC system. RTS strategy can achieve nearly perfect load balancing, because of the machines’/nodes’ performance variations and non-homogeneous nature of the application (image) is adjusted at runtime.

- The RTS strategy performance depends on the size of the sub-task. If sub-task size is too small then it generates a serious inter-process communication overhead. In other case, if the sub-task size is too large then it may create a load imbalance/ (longer master node waiting time) due to the inappropriate sub-tasks size of the slow performance node.
Two adaptive tasks scheduling strategies are proposed for HDC raytracing system. The adaptive strategies are: i) Adaptive Hybrid Task partitioning, Scheduling (AHTS), and load balancing strategy, ii) Worker Initiated Sub-task size (WIS) strategy.

The adaptive nature of task sizing scheduling strategy is essential to absorb the heterogeneity and performance variation characteristics of the worker.

Performance prediction is important for performance analysis of scalable parallel applications. Although there are several performance tools available that work with MPI programs, but none are capable of accurate performance prediction. We have developed the APAPS (Automated Parallel Application Prediction System) for cluster computing environment. This developed APAPS produces high prediction accuracy in performance predictions for parallel scientific applications.

We studied three threads parallelizing techniques along with remote procedure calls for heterogeneous distributed image computing system. It was observed that the efficient thread management had a good impact on the performance of distributed image computing system. We proposed the threads parallelizing technique III in which the master launched slave threads and then it went to sleep till the whole application was finished and slave threads had the autonomy to report results and take task from the balance automatically. The lightweight thread has now become a common element in current generation languages. We hope that our investigation will be valid for other languages, which support lightweight thread paradigm.