Automatic Resource-Centric Process Migration for MPI

Amnon Barak, Alexander Margolin, and Amnon Shiloh

The Hebrew University of Jerusalem, Department of Computer Science, 91904, Israel
{amnon,alexam02,amnons}@cs.huji.ac.il

Abstract. Process migration refers to the ability to move a running process from one node and make it continue on another. The MPI standard prescribes support for process migration, but so far it was implemented mostly via checkpoint-restart. This paper presents an automatic and transparent process migration framework that can be used for MPI processes. This framework is advantageous when migration of individual processes for purposes such as load-balancing is more adequate than checkpointing the whole job. The paper describes this framework for process migration in clusters and multi-clusters, how it was tuned for Open MPI and the performance of migrated MPI processes.

Keywords: Cluster, MPI, process migration, load-balancing, checkpoint.

1 Introduction

The term "process migration" refers to the ability to stop a running process on one node and make it continue from the same point on another. The main advantage of process migration is run-time flexibility. This includes redistribution of processes for improved performance and resource utilization, e.g., for load-balancing; and flexible cluster configuration, including orderly shutdown, addition of nodes and inclusion in a multi-cluster. In spite of the advantages, the main drawbacks of process migration are the complexity of maintaining migrated processes seamlessly and the need for an adequate policy to decide when, where and which process(es) to migrate.

The MPI standard [1] prescribes support for process migration, but so far direct support was implemented mostly via Checkpoint-Restart (CR) of a whole job, using a package such as the Berkeley Lab BLCR [2]. In the CR approach, all the processes of a job are stopped and their images are saved to persistent storage. The CR approach is reasonable when the demand for resources is relatively stable, but inadequate when resources and/or demand for resources change frequently, such as when running processes with uneven loads; when processes change their demand for resources (cores, GPUs, memory); when temporarily oversubscribing; when nodes are reclaimed by users with higher priority; and when CPUs slow down due to overheating. In such cases, direct migration of individual processes is lighter and more adequate because it does not stop the whole job. This is the main contribution of the current paper.

J.L. Träff, S. Benkner, and J. Dongarra (Eds.): EuroMPI 2012, LNCS 7490, pp. 163–172, 2012.

As an example where process migration can benefit queued MPI jobs, consider a situation where a job is scheduled to start on a given time, but few of its designated nodes are not available at that time, either because they are down or because they are used by overdue jobs. When process migration is available and memory is sufficient, the processes of the scheduled job can be temporarily assigned to those nodes that are already available, despite oversubscribing, and later be migrated to additional nodes as they become available. This avoids situations where jobs are queued while nodes are available but remain idle.

This paper presents a transparent (proactive) process migration framework for MPI based on features of MOSIX [3], a management system targeted for HPC on Linux clusters and multi-clusters. Its relevant features include a decentralized gossip algorithm that provides each node with information about cluster-wide resources [4]; a set of online algorithms that use this information to assign and actively reassign (migrate) processes to nodes, to optimize the performance [5]; and the actual process migration software.

The paper is organized as follows: Sec. 2 describes our process migration framework, including the run-time environment and the algorithms for initiating and managing process migrations in clusters and multi-clusters. Sec. 3 describes how our framework runs migratable Open MPI jobs, including allocation of resources and direct communication between migrated MPI processes. Sec. 4 presents various performance aspects of our process migration using standard benchmarks. Related works are described in Sec. 5 and our conclusion in Sec. 6.

2 A Framework for Process Migration

Process migration refers to the ability to stop a running process on one node, preserve all its important elements and then make it continue from the same point on another node. The elements to be preserved depend on the interface between the process and its immediate run-time environment. In a previous project, we developed a process migration framework for general Linux processes [3]. Since MPI processes run on Linux, we used that framework for migration of MPI processes.

This section presents relevant features of our process migration framework, including the run-time environment, the algorithms for initiating and managing migrations and support of multi-clusters.

2.1 Our Run-Time Environment

In order to support generic Linux processes, it is important that a process sees the same environment, including files, sockets, process IDs, etc., regardless where it runs or is migrated to. To achieve that, we developed a virtual environment (sandbox) in which each migrated process seem to run as if it is still in its original "home-node", where it was created. This is accomplished by intercepting all the system-calls of the process, then forwarding most of them to the home-node, performing them there and returning the results to the migrated process.