Process Migration in Heterogeneous Systems

Nirali Thakkar\textsuperscript{1} Anand Pandya\textsuperscript{2}
\textsuperscript{1,2}Assistant Professor
\textsuperscript{1}A. D. Patel Institute of Technology, New V. V. Nagar, Gujarat
\textsuperscript{2}Madhuben & Bhanubhai Institute of technology, New V. V. Nagar, Gujarat, India

Abstract—Process migration is very important aspect of distributed operating system. It is the act of transferring a process between two machines. This aspect of process migration deals with the decisions of transferring the state of process from one computer to another i.e. distributing the processes. The process migration mechanism must package the entire state of the process executing in the originating machine so that the destination machine may continue its execution. The process should not normally be concerned by any changes in the environment, other than in obtaining better performance. This paper will present the mechanism of Process Migration in heterogeneous distributed system.

I. INTRODUCTION

Process migration is the relocation of a process from its current location to another location [1]. Process migration refers to the act of disembodiment an active process on the machine that is currently running, and transferring its state to another machine, where it will resume execution from the point at which it was suspended. The important features one hopes to achieve while implementing a process migration facility would be automatic allocation and scheduling, with fairness, transparency, automatic eviction and overall performance improvement by load sharing. A process may be migrated either before it starts execution on its source node or during the course of its execution. The former is known as pre-emptive process migration and the latter are known as non-pre-emptive process migration. Pre-emptive process migration is more expensive than non-pre-emptive process migration since the process environment must also accompany the process to its new node for already executing process.

Process migration involves the major three steps. First, select process that should be migrated. Second, Select destination node to which the selected process should be migrated. Third, actual transfer of the selected process to destination node is started [1].

II. FEATURES OF A MIGRATION MECHANISM

A good process migration mechanism must possess following features:

1 Transparency-Migration should not affect the behaviour of either the process or its peers, in that its execution environment must appear the same. To the rest of the world the process should appear as if it is never left its home machine [2].

2 No Residual dependency- No residual dependency should be left on previous node. A migrated process should not depend on its previous node once it has started executing on its new node. [2]

3 Efficiency is another important issue in implementing the migration of process. There are three main source of inefficiency involved with process migration.

4 Migration of a process should cause minimal interference to the progress of the process involved and to the system as a whole. One method to achieve this is by minimizing the freezing time of the process being migrated. Freezing time is defined as the time period for which the execution of process is stopped for transferring to the destination node.

5 The results obtained by the execution of a migrated process should be exactly the same as obtained by its execution on the home machine.

6 Migration of a process to a remote machine should not compromise the security of the process.

7 The execution environment and access restrictions of the migrated process should be exactly the same as that on the home machine. [2]

8 The checkpoints and migration time should be kept low so that the overhead on the process’s execution is minimal [2].

III. PROCESS MIGRATION IN HETEROGENEOUS SYSTEM

When a process is migrated in a homogeneous environment, the interpretation of the data is consistent on both the source and destination nodes. Therefore, the question of data translation is does not arise. When a process is migrated in heterogeneous environment, all the concerned data must be translated from source CPU format to the destination CPU format before it can be executed on the destination node.

![Fig. 1: Four types of heterogeneous system without having external data representation](image-url)
representations and that of the other two processors. In general, a heterogeneous system having n CPU types must have the $n(n-1)$ pieces of translation software. The example for four processors types are shown in figure 1. The arrow represents the translation software. This technique is not efficient, as adding a new CPU type becomes a more difficult task over a time. In this method the software complexity is very high. If system has three types of processor, it requires total six translation software. The alternative solution is external data representation. It is a standard representation for transportation of data. Each processor has to convert its data to standard form. A heterogeneous system having n CPU types must have the $2^n$ pieces of translation software for this case. This bounds the complexity of the translation mechanism.

![Diagram of heterogeneous system](image)

**Fig. 2** Four types of heterogeneous system with having external data representation [7]

An example of three processors is shown in figure 2. The process of converting from a particular machine representation to external data representation (EDR) format is called serializing, and the reverse process is called deserializing. The handling of floating point needs special precaution. The floating point number representation consists of the exponent, mantissa and sign part.

**A. HANDLING THE EXPONENT**

The number of bits used for the exponent of a floating-point number varies from processor to processor. For example, processor “A” uses 8 bits and “B” uses 16 bits, and the external data representation designed by the users of processors architecture “A” provides 12 bits (an extra 4 bits for safety). Also assume that all three representations use the same mantissa. In this case, a process can be migrated from processor “A” to “B” without any problem in representing its floating point numbers because the two step translation process of the exponent involves the conversion of 8 bits of data to 12 bits and then 12 bits to 16 bits, having penalty of room for the converted data in both the steps.

A process that has some floating point data whose exponent requires more than 12-bits cannot be represented in the external data presentation. The problem here is with the design of the external data representation, which will not even allow data transfer between two processors, both of which use 16 bits for the exponent, because the external data representation have 12 bits for this purpose. This problem can be eliminated by guaranteeing that external data representation have at least as many bits in the exponent as the longest exponent of any processor in the distributed system.

**B. HANDLING THE MANTISSA**

The first problem in handling mantissa is the same as that of handling the exponent. Assume that exponent field is same in all the processors, and for the mantissa presentation, processor a uses 32 bits and B uses 64 bits, the external data representation uses 48 bits. In this case also the migration of a process from A to B will have no problem, but the migration of a process from B to A will result in the computation being carried out in “half-precision”. This may not be acceptable when accuracy is important. To overcome this problem, the external must have sufficient precision to handle largest mantissa, and the direction of migration should be restricted only to the nodes having mantissa at least as large as source node.

The second problem in handling the mantissa is the loss of precision due to multiple migrations between a set of processors. This is concern only in mantissa part because loss of one or more bits of the exponent is catastrophic, while loss of bits in mantissa only degrades the precision computation. It may appear that the loss in precision due to multiple migrations may be cumulative, and thus the series of migration may totally invalidate the computation. If the external data representation is properly designed to be adequate enough to represent the longest mantissa of any processor of the system, the resulting precision will never be worse than performing the calculation on the processor that has least precision among all the processor of the system.

**C. HANDLING SIGNED-INFINITY AND SIGNED-ZERO REPRESENTATIONS**

Signed-infinity is a value supported by some architectures that indicates that the generated result is too large (overflow) or too small (underflow) to store. Other architecture may use the sign bit of a value that would otherwise be zero, thus giving rise to a signed zero. The problem is that these representations may not be supported on all systems. The external data representation must take care of these values so that a given processor can either take advantage of this extra information or simply discard it.

**IV. ADVANTAGES OF PROCESS MIGRATION**

Process migration facility may be implemented in a distributed system to provide following advantages:

1. Reducing the average response time of process: Process migration is used to reduce the average response time of the processes of a heavily loaded node by migrating and processing some of its processes on a node that is idle.
2. Speeding up individual jobs: Task of the one job is migrated to the different node to execute both the job concurrently.
3. Gaining higher throughput: In a system that
does not support the process migration, it is very likely that CPUs of all the nodes are not fully utilized. But in a system with process migration facility, the capability of the CPUs of all the nodes can be better utilized by using a load-balancing policy.

4 Utilizing resources efficiently: Depending upon the nature of a process, it can be migrated to the most suitable node to utilize the system resources in most efficient manner.

5 Improving system reliability: One way to achieve system reliability is to simply migration of a critical process to a node whose reliability is higher than other nodes in the system. Another method is to migrate a copy of a critical process to some another node and execute both the process concurrently.

V. CONCLUSION

Process migration deals with the transparent relocation of a process from one node to another node in distributed system. A process may be relocated before it starts executing or during the course of its execution. Process migration in homogeneous system becomes more complex than in homogeneous system due to the need for data translation from source node format to destination node data format. The external data representation mechanism helps in reducing the software complexity of the translation process.

REFERENCES


