International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.05,Issue.15
June-2016,
Pages:2954-2957

www.ijsetr.com

A Survey on Different Implementation Categories of Process Migration and Different Process Migration Algorithms

REVATHI. S[1], SANDHYA. R[2], SUDHA. V[3]

[1]Assistant Professor, Dept of CSE, Dr.TTIT, KGF, India.
[2]Assistant Professor, Dept of CSE, Dr.TTIT, KGF, India.
[3]Assistant Professor, Dept of CSE, Dr.TTIT, KGF, India.

Abstract: The most interesting research and the challenging technique is the development of load balancing strategy especially in the distributed operating system and in distributed systems. The load balancing can be done in many ways one of the most popular variant is the process migration. The load in distributed operating system and in distributed system can be balanced by process migration technique. This survey paper helps in understanding the most important implementations categories of process migration mechanisms. Also, describes the different process migration algorithms.

Keywords: Migration, Microkernels.

I. INTRODUCTION

Process migration[1] is an abstraction of moving a process from one node to another node in the network of nodes. It is a very powerful and useful mechanism for balancing the load on distributed system. There are two types of process migration. (1) Preemptive process migration, which suspends the running process in the sender machine, transfer the process state to the destination machine, where it executes further is called preemptive migration. (2) Non-Preemptive process migration, it involves transferring of process that have not begun its execution. The migration can be done in both homogeneous and heterogeneous environment.

II. DIFFERENT IMPLEMENTATION CATEGORIES OF PROCESS MIGRATION

- Unix- like system early work.
- Unix- like system supporting transparent migration
- System with message passing interface
- Microkernel
- User- space migration
- Application- specific migration

A. Unix-Like System Early Work

Provides operating system level migration, as it is optimized for underlying hardware architecture. It is hardware dependent, migration complexity is low due to lack of infrastructure and transparency of migration is limited. Eg: XOS, Worm, Demos/MP and Butler.

XOS [2]	XOS is a Xerox operating system, to facilitate migration it represents the process and its state. The process work object (PWO) encapsulates process state, stack pointers and registers. The process migration is achieved by the movement of PWO objects between XOS nodes.
Worm[3]	Worm is self-replicating computer program residing on one or more machines. Migration is achieved by the movement of worm from one machine to another by occupying needed resources and replicating itself. Worm is aware of network topology.
Butler[4]	It supports both remote execution and process migration. Process migration is only through remote invocation. Migration occurs when the guest process appears at the machine where the resource exceeds, leads to the transfer of complete state of guest process to new node, it also provides security, protection and autonomy.
Demos/MP[5]	Process migration is fully transparent. A process can be moved during its execution and continue its execution on another processor with the support of message passing, location-independent communication. Kernel participate in process migration therefore it is hardware dependent

B. Unix-Like System Supporting Transparent Migration

Requires major changes to the underlying kernel, migration complexity is high as it is operating system dependent, migration is fully transparent. There are approaches for addressing distribution and migration. (1) Distribution at lower level of a system. Eg: Mosix/Spirit. (2) Distribution at higher level. Eg: Locus.

Locus[6]	Supports process migration and initial placement. Migration is fully transparent and it requires kernel modifications
Mosix[7]	Distributed operating system supports process migration on top of a single system and in a network of workstations environment. Process migration is fully transparent. It supports automatic load balancing between Mosix nodes by process migration, also supports dynamic load balancing and dynamic process migration.Eg. Eager(dirty) and COR migration
Spirit[8]	Spirit is a network operating provides transparent process migration both to users and to application. It supports migration in the form of remote invocation in which entire executing process is migrated. Eg. Flushing

C. Systems with Message-Passing Interfaces

The process migration for messages passing OS can be easily designed and implemented. It is operating system dependent, migration complexity is low as message can be easily redirected between the sender. Message passing is based on interposing forwarding and encapsulating state. Eg: Charlotte, Accent &kernel.

Charlotte[9]	Charlotte is a message passing OS designed for the multicomputer. Migration completely relies on the underlying OS 7 IT communication mechanism. Transparency is obtained after modification to the communication network. Provides fault resilience migration, where no residual dependency on the source machine.
Accent[10]	Accent is a distributed OS, it migrates scheme was the first. Scheme to use the COR (copy on ref) technique. The program would not access all of it address space, the destination node creates a link to the source node to copy its address space based on references.
V kernel[11]	Based on microkernel OS. Benefited from a communication protocol that dynamically rebind to alternate destination host during its implementation. V Kernel relies on multicast to find the destination process location. Instead of maintaining the end-point details. It introduced precopy migration technique, where the transfer of address space &executing the process on the source machine is done in parallel resulting in copy of dirty pages.

D. Microkernels

Builds process migration facilities mechanism on top of the operating system as a separate module. Process migration is implemented easily as it supports message passing communication and transparencies. Eg: RHODOS, Arcade, Chorus, Amoeba, Birlix and Mach.

Mach[12]	Support task & process migration, it require minimal changes to the microkernel. Provides transparent task migration at user-lever with new interfaces provided. Process migration relies on task massage mechanism. The process migration techniques like eagles copy, flushing, COR, precopy supported in this implementation.
RHODOS[13]	The migration mechanism similar to that in sprite with some modification to the RHODOS kernel with the support of IPC and process migration managers, the RHODOS operating system obtains the process migration.
ARCADE[14]	This is the microkernel based implementation when the groups of tasks are migration. It includes framework to support task grouping necessary for migration. Also uses group management software to ensure the grouping of tasks.
CHORUS[1]	The migration mechanism is similar to task migration on top of mach, however migration is at process level. It does not support port migration.
AMOEBA[15]	Fully transparent, which relies on the location independence of the FLIP protocol ,memory transfer for process migration is achieved by physical coping
BIRLIX[16]	Supports object migration. The meta object encapsulates the data needed for migration and information collection.

E. User-Space Migration

It is less transparency, does not require any modification to the kernel. Migration complexity is low as it uses system calls for communication. It is application dependent. Eg: Condor, Alonso & Kyrimis, Mandelberg & Sunderam, MPVM and Load Sharing Facility.

Condor[1]	Support user space cheek pointing &process migration in locally distributed systems. Check pointing is meant for long running process &expensive for short processes. Here migration is that it generates a core fix of a process, combine with executable &transfer to the receiver to system calls.
Alonso and kyrimis[1]	Requires minor modification to the Unix kernel in order to support process migration at user-space. The implementation is limited to the process that is not location or process dependent and the process that do not communicate.
Mandelbergsunderam[1]	Important present a process migration scheme for Unix that performs input/output on non –NFS files. A new terminal interface is needed to detach a process from it terminal and to monitor the request for input/output on the process migration port.
MPVM[1]	Migratory Parallel Virtual Machine, supports the process migration among the homogeneous machine, it provide transparent process migration at user-level.
Load sharing facility[1]	Supports process scheduling, transparent remote execution on top of kernel without any modification. Supports user-level process migration and also support check pointing (initial load placement for load balancing)

F. Application Specific Migration

Here, migration is implemented as a part of an application. Independent of operating system, where modification is at application level. Requires more application knowledge, minimal transparency migration complexity is lowest (application migration). Migration has to be adjusted for each new application. Implementation is simplified &optimized. Eg: Freedman, Skordos and Bharat &Cardelli.

Freedman[1]	Process migration achieved by establishing the cooperation between the migration process &the migration module.
Skordos[1]	This involves the migration among the work stations. The migration is initiated when the workstation become overloaded .The process is restarted upon migration after synchronization with processes participating in the migration.
Bharat &Cardelli[1]	This type of migration suitable for mobile application, where the user travels from one environment to another. Describes migratory application, user INTERFACE &the application text.

III. THE DIFFERENT PROCESS MIGRATION ALGORITHMS

S.no	Strategy	Approach	Advantages	Disadvantage
1	Eager copy(all) [1]	Transfers all information to process. On request to migrate, the process on source node is suspended. On completion of full transfer process is resumed at the destination node	1. simple 2. Most commonly used 3. straightforward approach	1. The delay length between suspension and resumption is dependent on the size of the process.
2	Eager copy (dirty) [1]	Transfers only modified(dirty)pages and unmodified pages are transferred upon request	1.Reduces the initial cost even for the large address apace. 2.The cost of eager (dirty) is lower than Total-Copy	1.This algorithm can be used only if the system has remote paging facilities.
3	Lazy copy[1]	The algorithm transfers only the minimum necessary information that are needed for the process toresume its execution on the destination machine.	1.Transfer of minimal information leads to reduction in migration delay 2. overall reduction in network traffic	1. delay between suspension and resumption 2. cost of process creation on the destination host 3. results in residual dependencies on the source node thus cannot improve the system reliability
4	Pre copy[17-18]	Transfer of address space is done in parallel with continued execution of process in the source machine and assuming that the migration is accepted at the destination machine	1. Reduces the migration delay caused in total copy algorithm.	1. Need to transfer some information twice 2. Final delay may be quite long for the large section of address space
5	Flushing[1]	Involves a third party entity(Network file server) Eg. Back storage in Sprite. Depends upon the operating system implementation of virtual memory. It uses standard page fault handling mechanism	1.Found to be effective strategy for migration under sprite 2.Page fault can be efficiently resolved using the file server 3. Reduces time lag between suspension & resumption 4. Residual dependencies are avoided	1. Implementation cost is more
6	Post copy[18]	Combination of Pre and Lazy copy. Similar to Pre copy but it avoids residual dependencies by moving the process address as it is executing on the destination node	1. Receiving good attention Eg: Open Mosix	1. Implementation is difficult
7	Total copy[17-18]	It is the basic process migration algorithm. The mechanism of this algorithm is to suspend the process, migrates all state information, and then resumes the process	1. It eliminates residual dependencies. 2. Its conceptual simplicity allows straightforward implementation.	1. leads to a lengthy delay in the process execution time
8	Copy on reference[1]	It is a network version of demand paging. Pages are transferred only upon reference	1.Has lowest initial cost	1.Increases run-time costs 2, Requires substantial changes to the underlying operating system& to the paging support
9	Demand page [17]	It uses lazy copying or COR for similar to demand page. Similar to total copy algorithm except that no virtual memory page are transferred at migration time	1. No wasted transfer of unused pages 2. Shorter freeze & process migration latency time.	1.Residual dependency exit 2. The algorithm is not fault tolerant
10	Freeze Free[17]	Intended for a distributed system. Using message communication & supported by distributed file server. All host should be homogeneous & run on the same operating system.	1. Migration time is reduced	1. Implementation cost is more

IV. REFERENCES

[1] Dejan S. Milojicic, Fred Doudlis, Yves Paindaveine, Richard Wheeler and Songnian Zhou, " Process Migration" . ACM Computing Surveys, September 2000.

[2] Miller, B. &Presotto, D. XOS: an Operating System for the XTREE Architecture. Operating System Review, 2,15,21-32. 1981.

[3] Shoch, J. &Hupp, J. "The Worm Programs, Early Experience with Distributed Computing, Communication of the ACM 25, 3, 172-180, 1982.

[4] Dannenberg, R.B. , "Resource Sharing in a Network of Personal Computers, Ph.D. Thesis, Technical Report CMU-CS-82-152, Carnegie Mellon University. 1982.

[5] Miller, B. ,Presotto, D., & Powell, M. " Demos/MP: The Development of a Distributed operating System. Software-Practice and Experience 17,4,277-290. 1987.

[6] Walker B.J and Mathews, R.M., " Process Migration in AIX's Transparent Computing Facility(TCF). IEEE Technical Committee on Operating System Newsletter, 3,1 (1) 5-7, 1989.

[7] Barak, A, Shiloh, A & Wheeler, R. " Flood Prevention in theMosix Load-Balancing Scheme. IEEE Technical Committeeon Operating System Newsletter,3,1,24-27., 1989.

[8] Douglis, F. &Ousterhout, J. "Transparent Process Migration: Design Alternatives and the Sprite Implementation". Software –Practice and Experience 21, 8, 757-785, 1991.

[9] Artsy, Y. &Finkel. R, "Designing a Process Migration Facility: The Charlotte Experience". IEEE Computer, 47-56, 1989.

[10] Rashid, R, " From RIG to Accent to Mach: The Evolution of a Network operating System. Proceedings of the ACM/IEEE Computer Society Fall Joint Computer Conference, 1128-1137, 1986.

[11] Cheriton. D, " The V Distributed System. Communications of the ACM 31, 3, 314-333, 1988.

[12] Milojicic. D., Zint, W. DangelA& Giese P. "Task Migration on the top of the Mach Microkernel. Proceedings of the third USENIX Mach Symposium, 273-290.fc, 1993.

[13] ZHU, W. "The Development of an Environment to Study Load Balancing Algorithm, Process Migration and Load data collection. Ph.D. Thesis, Technical Report, University of New South Wales. 1992.

[14] Tracey, K.M " Processor Sharing for Cooperative Multi-task Applications. Ph.D. Thsis, Technical Report, Department of Electrical Engineering, Notre Dame, Indiana, 1991

[15] Steketee, C, Zhu, W and Moseley, P., "Implemetation of Process Migration in Amoeba. Proceeding of the 14th International Conference on Distributed Computer Systems, 1994.

[16] Lux, W, "Adaptable Object Migration: Concept and Implementation". 1995.

[17] Ramon Lawrence, "A survey of process migration mechanism". May 29 1998

[18] NOACK, M. "Comparative evaluation of process migration algorithms". Master's thesis, Dresden University of Technology - Operating Systems Group, 2003.