©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 27

Designing a Thread Migration Facility for Linux Operating System

K.KoteswaraRao, GMR Inst.Of Technology, RAJAM-532127,India
Srinivasan Nagaraj, GMR Inst.Of Technology, RAJAM-532127,India
G Appa Rao, GITAM University, Vizag-530041India
G.Anuradha, GMRInst.Tech., Rajam- 532127
Dr GSVP.Raju, Andhra Univ., Vizag-India.

ABSTRACT:

The availability of Low cost and high performance workstations connected by a high-speed network has made distributed computing an attractive and inexpensive mechanism to exploit parallelism at functional level present in the user or application programs. A distributed system can be used effectively by its end users only if its software presents a single system image to the users. Thus all the resources of any node should be easily and transparently accessible from any other. While solutions are available for transfer and sharing of resources such as files and printers, in general on all operating system that support networking technologies, there is an eminent need for operating systems to share the overall computing facilities including the CPUs for better performance and fault tolerance. Sharing of the CPU requires the operating systems on different machines be cooperating for achieving more even load balance. Thus the operating systems together must have a common protocol for process migration. As a further step, it is observed that, in an attempt to exploit any functional level parallelism, a programmer writing user-level application programs would be at ease while using threads rather than using processes. Spreading execution of processes or threads over several processors can exploit parallelism and thus achieve improved performance. As compared to a process, a thread is lighter in terms of overhead associated with creation, context switching, inter process communication and other routine functions. This is because these primitives can be executed within the same address space. So we go for a thread migration rather than a process migration. In this paper, we highlight the Advantages of thread migration for better utilization of computing resources as well as to gain substantial speedups in the execution of parallel and multitasking applications. In particular, we describe design issues for including in the existing Linux kernel a thread migration and thread based scheduling modules, and provide suggestions for an easy implementation of the proposed designs