Solaris-2 Operating Systems

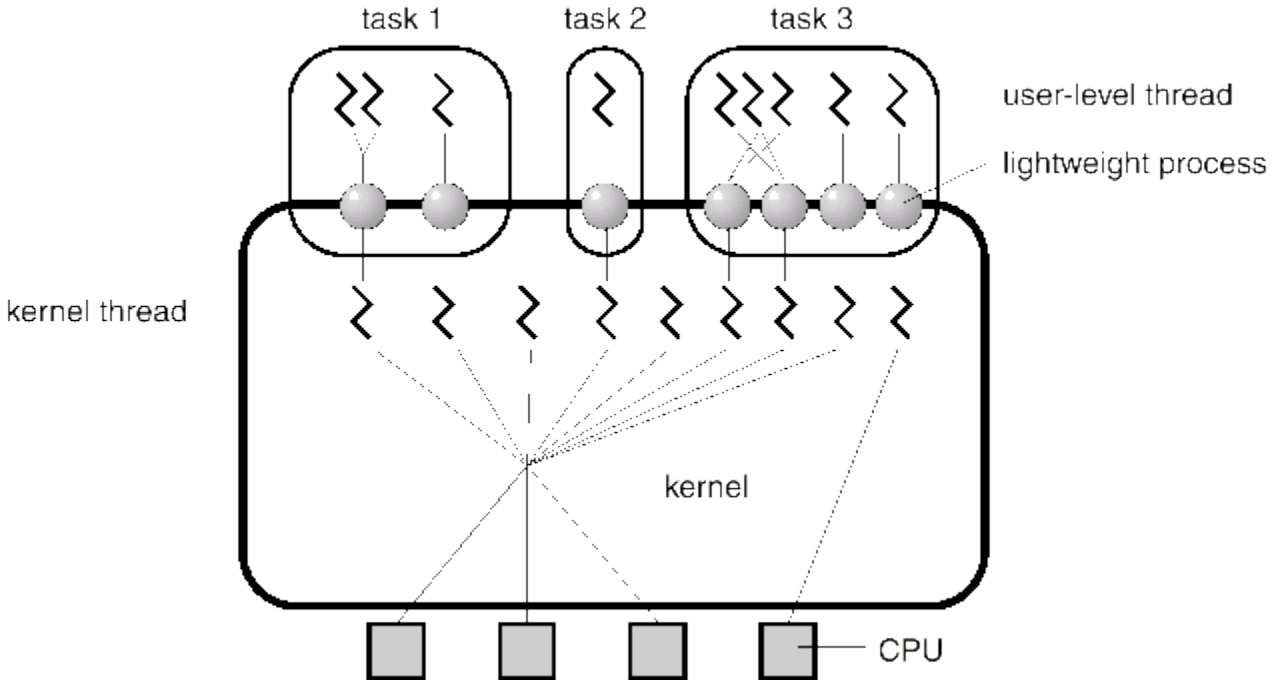
- Introduction
- At user-level
- At Intermediate-level
- At kernel-level

Introduction

The solaris-2 Operating Systems supports:

- threads at the user-level.
- threads at the kernel-level.
- symmetric multiprocessing and
- real-time scheduling.

The entire thread system in Solaris is depicted in following figure.



At user-level

- The user-level threads are supported by a library for the creation and scheduling and kernel knows nothing of these threads.
- These user-level threads are supported by lightweight processes (LWPs). Each LWP is connected to exactly one kernel-level thread is independent of the kernel.
- Many user-level threads may perform one task. These threads may be scheduled and switched among LWPs without intervention of the kernel.
- User-level threads are extremely efficient because no context switch is needs to block one thread another to start running.

Resource needs of User-level Threads

- A user-thread needs a stack and program counter. Absolutely no kernel resource are required.
- Since the kernel is not involved in scheduling these user-level threads, switching among user-level threads are fast and efficient.

At Intermediate-level

The lightweight processes (LWPs) are located between the user-level threads and kernel-level threads. These LWPs serve as a "Virtual CPUs" where user-threads can run. Each task contains at least one LWp.

The user-level threads are multiplexed on the LWPs of the process.

Resource needs of LWP

An LWP contains a process control block (PCB) with register data, accounting information and memory information. Therefore, switching between LWPs requires quite a bit of work and LWPs are relatively slow as compared to user-level threads.

At kernel-level

The standard kernel-level threads execute all operations within the kernel. There is a kernel-level thread for each LWP and there are some threads that run only on the

kernels behalf and have associated LWP. For example, a thread to service disk requests. By request, a kernel-level thread can be pinned to a processor (CPU). See the rightmost thread in figure. The kernel-level threads are scheduled by the kernel's scheduler and user-level threads blocks.

SEE the diagram in NOTES

In modern solaris-2 a task no longer must block just because a kernel-level threads blocks, the processor (CPU) is free to run another thread.

Resource needs of Kernel-level Thread

A kernel thread has only small data structure and stack. Switching between kernel threads does not require changing memory access information and therefore, kernel-level threads are relating fast and efficient.

Source:

http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/solaris.htm