Virtual Machines and Processes

Peter J. Denning

5/20/19

In the early 1960s, a debate raged among operating systems designers about how to characterize the activities of different users on a time-sharing system. The term "process" was used to distinguish the evolving sequence of actions of an executing program from its static statements. A program was a set of instructions to control a machine; a process was the dynamic behavior of a program executing. Each time a user issued a command to the operating system, the operating system would initiate a process to execute the code of the command.

With this idea in mind, it became clear how to implement processes and the mechanisms for scheduling the CPU from one to the next so that all could appear to be progressing in parallel. Around 1965, Edsger Dijkstra had the insight that an operating system should be designed of as a "society of harmoniously cooperating sequential processes." He used the term "harmonious" to recognize that interactions are required for proper coordination of action and interactions must be designed properly to avoid unwanted behaviors such as races or deadlocks. By 1970, the society-of-processes principle was widely accepted by designers of operating systems.

For a while, there was some confusion over terminology. Some designers said that a *process* is "a program in execution on a virtual machine"; a virtual machine was a simulation of the CPU, memory, and input-output of the real machine. Others used the term *thread* to mean "the sequence of instructions executed by a CPU in a program". Eventually it was decided that a thread is a component of a process, and that a process could house multiple threads. Today the term process is synonymous with virtual machine and thread with the execution sequences within a process.

The virtual machines level of the OS kernel is charged with the responsibility of creating and deleting virtual machines and connecting them together as needed for for proper synchronization. The thread component of a virtual machine is realized by the concurrency level of the OS; that level creates and deletes threads, schedules them for time-slices on a CPU, and synchronizes them with semaphores. Virtual machines are higher up in the OS levels because they also contain an address space component in which the threads execute, and an input-output component that connects otherwise independent processes together. We situated the virtual machine manager level of the OS above information objects because all the levels below a virtual machine provide components of a virtual machine.

The virtual machine is a useful way to isolate processes so that by default they cannot interfere with one another or see each other's memory. The input-output

capabilities of virtual machines allow processes to exchange data over controlled channels without directly accessing their address spaces.

The term virtual machine has acquired several meanings in computer science. One is the simulation of one machine on another; this allows one operating system to simulate another and it supports portability platforms such as Java Virtual Machine.. A second is the creation of replicas of the CPU, which are identical to the real CPU but with smaller main memory. A third is any abstract machine; each level of the OS is an abstract machine that manages all objects of a given kind. A fourth is a template for a standard form of implementing a process; this meaning is used in Unix and is the basis of our treatment here.