CS418 — Operating Systems

Lecture 16

Process Migration and Distributed Global States

Textbook: Operating Systems
by William Stallings

1. Process Migration

- Motivation
  - 1. Load sharing — processes can be moved to lightly loaded systems to improve overall performance.
  - 2. Communication performance — processes interacting intensively can be moved to the same node to reduce communication cost.
  - 3. Availability — long-running process may need to move to survive in advance of scheduled downtime.
  - 4. Utilizing special capabilities — a process can migrate to a node with special software/hardware.

2. Questions on Process Migration

- **Question 1:** Who initiates the migration?

  - 1. If the goal is load balancing, then the node which monitors the system load will do that.
  - 2. If the goal is to reach particular resources, then a process can migrate itself.

- **Question 2:** What is migrated?
  The movement of PCB's is easy, but not on the process address space and any open files assigned to the process. Assuming paging is used,...
  - 1. **Eager (all)**—transfer the entire address space.
  - 2. **Precopy**—process continues to run at the source node while the address space is copied to the target. Pages modified during the precopy operation have to be re-copied.
  - 3. **Eager (dirty)**—transfer only those pages of the address space that are in main memory and have been modified. Additional pages will be transferred on demand.
  - 4. **Copy-on-reference**—variation of eager (dirty).
  - 5. **Flushing**— pages of the process are flushed to disk, pages are then accessed as needed from disk instead from main memory.

- **Question 3:** What about those outstanding messages and signals?
  Let's consider a distributed UNIX OS -IBM's AIX system

  – 1. When process P decides to migrate, it selects a machine S and sends a remote tasking message carrying a part of the process image and open file information.

  – 2. At S a kernel forks a child which is given this information.

  – 3. This new process at S pulls over everything (data, environment, arguments, stack) necessary to finish it. (If a page is on demand at the source, it should also be copied to S.)

  – 4. On completion of the migration, P is signaled and consequently it will destroy itself.

3. Negotiation of Migration

- Under some situation, even if a process can migrate to another system its response time might be seriously affected. So it makes sense for some system to participate in the decision.

  We will focus on the system called Charlotte, the Starter facility there handles migration policy (**when to migrate which process to what destination**).

4. Distributed Global States

- In distributed systems, each nodes only knows the local state and there might not be a globally synchronized clock.

  Why this is a problem?

- **Channel:** a unidirectional path by which messages are transferred.
- **State:** sequence of messages that are sent and received along channels of that process.
- **Snapshot:** a snapshot records the state of a process.
- **Global state:** the combined state of all processes.
- **Distributed snapshot:** a collection of snapshots, one for each process

The true global state is hard to determine because of the time lapse associated with message transfer. We can do our best by collecting snapshots from all processes.

An **inconsistent global state** arises when A sends a message to B and B records the receipt of the message but A hasn't recorded that the message has been sent.

A **consistent global state** is that, for all A and B, when A sends a message to B and B records the receipt of the message then A must have recorded that the message has been sent.

5. Distributed Snapshot Algorithm

- **Marker**: a special control message.

- Assumption: messages are delivered in order and no message is lost.

- **Algorithm:**
  Initializer: record its own state and send a marker on all outgoing channels.
  Other process p's (upon the first receipt of the marker):
  - **1.** p records its own state S_p.
  - **2.** p records the state of the incoming channel from q to p as empty.
  - **3.** p propagates the marker to all of its neighbors along all outgoing channels.

  All process p's (after recording its own state, when p receives another marker from r):
  - **1.** p records the state of the channel from r to p as the sequence of messages p has received from r from the time p records S_p to the time p received the marker from r.

  This solution was proposed by Chandy and Lamport (1985).

6. Properties of the Distributed Snapshot Algorithm

- Any process may start the algorithm by sending out a marker. (Several nodes can independently start to record the state.)
- The algorithm always terminates if every message (marker) is delivered in finite time.
- The algorithm is distributed: every node only handles local information.
- After the algorithm terminates, the initiating process can poll all processes to acquire the global state.
- The algorithm does not interfer with any other distributed algorithm (e.g., using some of the messages for different purpose).