Process Migration - I

Models and Approaches

1

Outline

- Introduction to Process Migration
- Models and Possibilities
- Load Balancing using Process Migration
- Migrating Resource Segment
- Challenges for creating Global Reference

20 June 2024

2

Outline

- Introduction to Process Migration
- Models and Possibilities
- Load Balancing using Process Migration
- Migrating Resource Segment
- Challenges for creating Global Reference

3

Definition

- Process migration is the mechanism to migrate code, data and state of execution from one node to another in a distributed systems.
- Moving a running process to a different machine is a costly and intricate task, and there has to be good reason(s) for doing so.

4

Motivations

- Load balancing between nodes
- Minimize communication
- Due to the heterogeneity in the system, performance improvement through process migration is often based on qualitative reasoning
  - Migration of process to data location
  - Migration of data to process location

5

Fugetta's Framework

- Code Segment:
  - contains the set of instructions that builds the program being executed.
- Resource segment:
  - stores external resources or references to those like files, devices, data, etc.
- Execution segment:
  - stores current execution state of a process.

6

Outline

- Introduction to Process Migration
- Models and Possibilities
- Load Balancing using Process Migration
- Migrating Resource Segment
- Challenges for creating Global Reference

7

Migration Models

- Weak Mobility / Non pre-emptive:
  - In this model, it is possible to transfer only the code segment, along with perhaps some initialization data.
  - Migrated process restarts from its initial state.
  - The benefit of this approach is its simplicity.
  - Weak mobility only requires that the target machine executes the code - this essentially boils down to making the code portable.
  - *e.g., Java applets*

8

Migration Models

- Strong Mobility / Pre-emptive:
  - Here, the execution segment can be transferred along with the code segment.
  - The characteristic feature of strong mobility is that a running process can be stopped, subsequently moved to another machine, and then resume execution where it left off.
  - More powerful than weak mobility and harder to implement.

20 June 2024

9

Migration Models

- Sender Initiated Migration:
  - Sender-initiated migration is initiated at the machine where the code currently resides or is being executed.
    - sending a search query across the system to find a compute server
    - uploading programs to the compute server

20 June 2024

10

Migration Models

- Receiver Initiated Migration:
  - The initiative for code migration is taken by the target machine.
  - Java applets are an example of this approach.
- Receiver-initiated migration is often simpler to implement than sender-initiated migration.

Uploading versus Downloading

- Securely uploading code to a server often initiated by sender requires that the client is previously registered and authenticated for the server.
- The client is expected to access the server's resources such as its disk, file system, etc.
- Protecting such resources is essential.

Uploading versus Downloading

- In contrast, downloading code at receiver initiation can be anonymous.
- The server is generally not interested in the client's resources.
- Code migration to the client is done for improving client-side performance.
- Thus only a few of the resources like memory are to be protected.

20 June 2024

13

Execution in Target Process

- In case of weak mobility, two things may happen:
  - the migrated code is executed in the target process
  - a separate process is created.
- e.g., Java applets are simply downloaded by a Web browser and are executed in the browser's address space.

20 June 2024

14

Execution in Target Process

- In this approach, there is no need to start a separate process – lower overhead for communication at the target machine.
- The main drawback is that the target process needs to be protected against malicious or inadvertent code executions.
- A simple solution is to let the operating system take care of that by creating a separate process to execute the migrated code.

20 June 2024

15

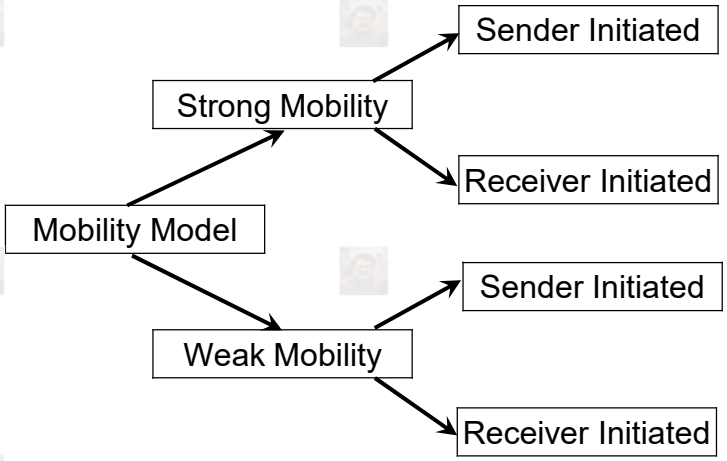
Outline

- Introduction to Process Migration
- Models and Possibilities
- Load Balancing using Process Migration
- Migrating Resource Segment
- Challenges for creating Global Reference

20 June 2024

16

Process Migration Options



20 June 2024

17

Sender Initiated Migration

- A node having load higher than a pre-set upper threshold is referred as a Sender node
- Possible metrics could be, e.g.,
  - Number of jobs in ready queue
  - Throughput
  - Number of jobs in running, or blocked state
- Similarly, a node with load less than a pre-set lower threshold is called a Receiver node

20 June 2024

18

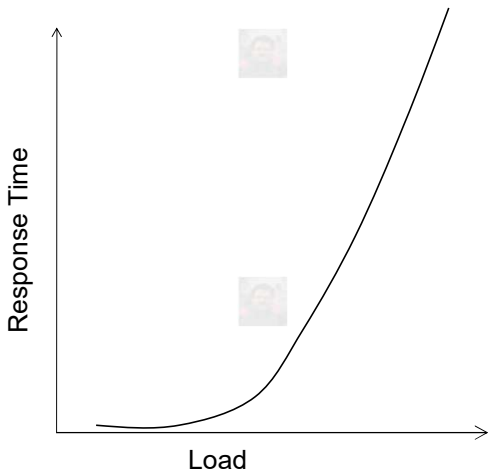
Sender Initiated Migration

- There should be an interval between the two thresholds, so that by releasing or accepting one, or two jobs a Sender does not become Receiver and vice versa
- A Sender node poles an arbitrary node and waits for response
- When the overall load of the system is low, this response time will be low as well
- Response time keeps on increasing with heavier loads; Sender poles another node

20 June 2024

19

Sender Initiated Migration



20 June 2024

20

Sender Initiated Migration

- Excessive poling in a heavily loaded system adds to congestion and could worsen the performance of the system as a whole
- This is a type of thrashing
- In order to prevent this, a Sender node is allowed to pole a MAX number of times before it must wait for D interval to allow change of load in the system

20 June 2024

21

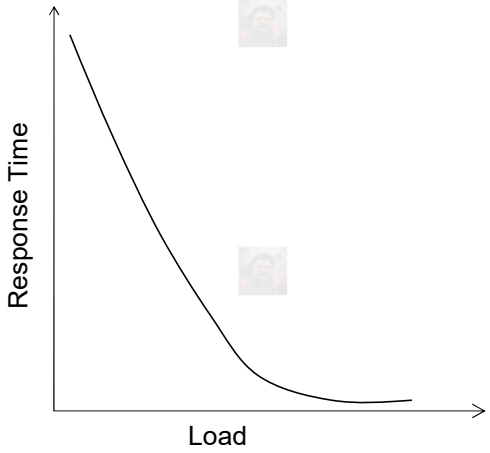
Receiver Initiated Migration

- A Receiver node poles an arbitrary node requesting job and waits for response
- When the overall load of the system is low, the response time will be high
- Response time keeps on decreasing for increase in system load
- Here too, excessive poling adds to congestion; but this happens when the system is having low average load

20 June 2024

22

Receiver Initiated Migration



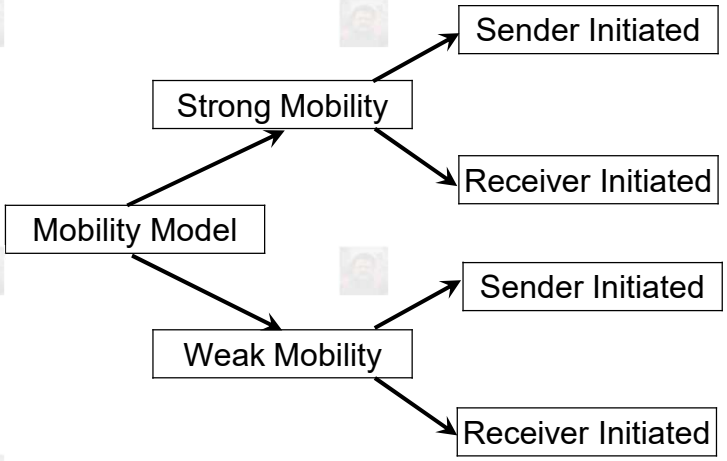
23

Receiver Initiated Migration

- In order to prevent polling, a Receiver node is allowed to pole a MAX number of times in a single go
- Thus receiver-initiated migration ensures low response time in highly loaded system and appears to be a better option
- Is it necessarily so?

24

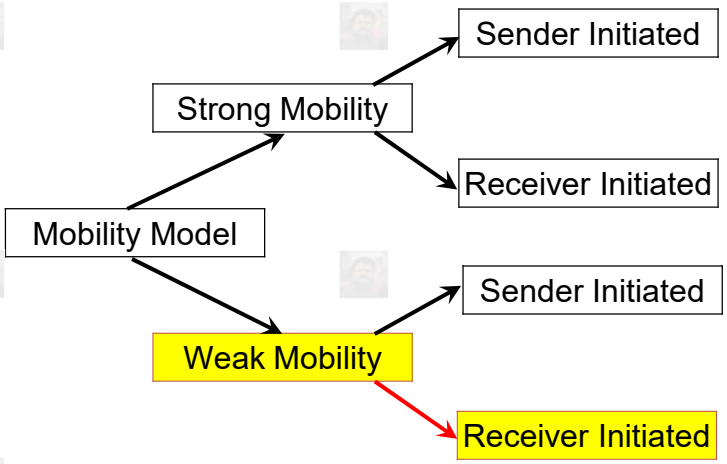
Process Migration Options



20 June 2024

25

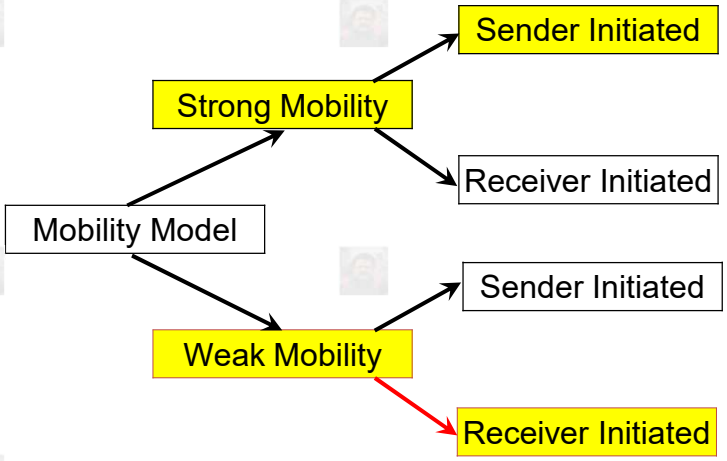
Process Migration Options



20 June 2024

26

Process Migration Options



20 June 2024

27

Thanks for your kind attention

Questions??

28