CSCI 5105

Instructor: Abhishek Chandra

Today

- Distributed Computing
- Distributed Scheduling
  - Parallel Processing
  - Clusters

Distributed Computing

- Processes executing on distributed computing resources
  - Typically group of processes executing together to accomplish common task
- Main issues:
  - How to manage distributed computing resources efficiently (E.g.: high utilization, throughput, etc.)?
  - How to coordinate related processes, part of a single application?

Distributed Computing Environments

- Multiprocessor systems:
  - Tightly-coupled, may have shared memory
  - Small SMPs vs. large MPPs (Parallel computing)
- Cluster computing:
  - Independent machines connected over LAN

Distributed Scheduling

- Question: Given a set of nodes and a set of processes, where to execute these processes?
- Depends on computing platform
  - MPPs
  - Clusters

5

Parallel Computing

- Massively Parallel Processors (MPPs)
  - 100s-1000s of CPUs connected via fast interconnect bus
  - May have NUMA architecture, distributed memory
  - E.g.: supercomputers
- Parallel jobs:
  - Consist of large number of processes
  - Processes typically work in sync, may communicate
  - Processes submitted in batches, scheduled on the system

6

Space Sharing

- Used for groups of processes
  - Need to run together
- Partition CPUs into sets
  - Each set runs one process group
  - Non-preemptible until all processes finish
- Job scheduling:
  - FIFO: Jobs run in arrival order
  - Priority: High-priority jobs assigned CPUs first
- What about utilization?

7

Backfilling

- Allow low priority jobs to get ahead if:
  - They can fit
  - Do not impact the start time of high priority jobs

8

Time Sharing

- Allow each CPU to run multiple processes
- Use local scheduler on each CPU
  - E.g.: Round-robin
- Benefits?

9

Gang Scheduling

- Extends time-sharing
- Problem: Processes communicating heavily
- Goal: Avoid delay in communication, synchronization
- Ensure processes from same group (gang) are scheduled together
- Need synchronization of scheduling quanta
- Co-scheduling: Variant of gang scheduling that allows part of the gang to execute in parallel

10

Cluster Computing

- Multi-computer system:
  - Machines connected over a LAN
  - Could be homogeneous or heterogeneous
- Issues:
  - No shared memory
  - Processes cannot move easily
  - Each node has a local scheduler
- Question: How to allocate processes across nodes?
  - Case 1: Processes originate on machines
  - Case 2: Jobs submitted by users

11

Distributed Processor Allocation

- Assume homogeneous cluster
- Processes arrive continuously
  - Started on one of the machines
- Approach 1: Keep each process on original machine, use local scheduler
  - Problem?
- Load imbalance: Probability that at least one processor is idle while a job is waiting
  - How does this relate to avg. load in the system?

12

Load Balancing

- Allocate/move processes to balance load across machines
- Design issues:
  - Centralized vs. distributed?
  - How to measure load?
  - Pre-emptive vs. non-preemptive?
  - Who initiates the allocation/migration?

13

Centralized Load Balancing

- Central monitor:
  - Measure loads
  - Assigns processes to nodes
- Advantages?
- Disadvantages?

14

Distributed Load Balancing

- Each node makes its own decisions
- Transfer policy: When to transfer a process?
  - E.g.: Threshold-based
- Selection policy: Which process to transfer?
  - New, low migration cost, long execution time
- Location policy: Where to send the process?
  - Random, nearest, least-loaded
- Information policy: Who sends the information, to whom, and when?
  - On-demand, periodic, state-change-driven

15

Initiating Process Migration

- Sender-initiated: Overloaded node initiates process migration
- Receiver-initiated: Underloaded node looks for work
- Hybrid: Each node can act as both sender and receiver
  - Maintain two thresholds (high and low load)

16

Resource Management

- Assume possibly heterogeneous cluster
- Jobs submitted by users
  - Could consist of multiple processes
- Assign processes to nodes
- Goals:
  - Meet job requirements
  - Maximize resource utilization

17

Matchmaking

- Resource providers, resource consumers
  - Providers specify their resource constraints. E.g.: OS, CPU speed, memory, etc.
  - Consumers specify their resource requirements. E.g.: need Windows XP, 1GHz CPU, 512M RAM, etc.
- Specifications provided in a specification language (e.g.: XML)
- Matchmaker matches requirements to constraints
- E.g.: Condor

18

Bidding-based Process Allocation

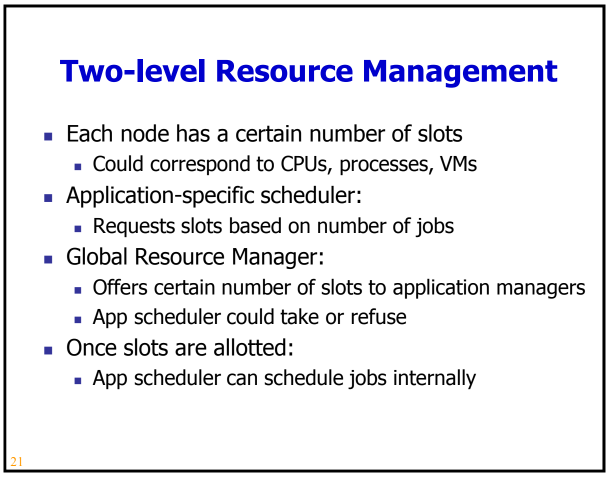
- Economic model
  - Nodes advertise resources with prices
  - Processes bid on resources
  - Highest bidders win resources
- Questions:
  - Who organizes bids and advertisements?
  - How to determine prices?
- E.g.: Amazon Spot Instances

19

Multi-Framework Resource Management

- Two-level resource management
- Global resource manager: Provides coarse-grained resource allocation
- Application-specific resource schedulers: Perform application-specific scheduling
- E.g.: Mesos, YARN, Awan

20



Two-level Resource Management

- Each node has a certain number of slots
  - Could correspond to CPUs, processes, VMs
- Application-specific scheduler:
  - Requests slots based on number of jobs
- Global Resource Manager:
  - Offers certain number of slots to application managers
  - App scheduler could take or refuse
- Once slots are allotted:
  - App scheduler can schedule jobs internally

21