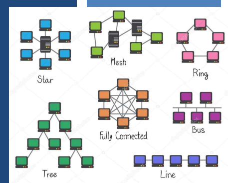
TCSS 558:
APPLIED DISTRIBUTED COMPUTING



Servers, Code Migration

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

OBJECTIVES

- Assignment 1 questions
- Review Quiz 1
- Feedback from 10/26
- Ch. 3 – Processes and threads
  - Servers
  - Code migration
- Practice Quiz for midterm

AWS EDUCATE CREDITS

- Try this website:
- https://www.awseducate.com/Registration?apptype=student&courseview=true
- Register for University of Washington, TCSS 558
- Please report success obtaining credits this way

FEEDBACK FROM 10/26

- .docx version of assignment #1 doesn't work
  - Link fixed – thank you!

FEEDBACK - 2

- What's the difference between cloud systems and distributed systems?
  - GOOD QUESTION
  - Distributed systems are built with multiple computers
  - D/S on LANs: all nodes connect via private subnet (subnet mask 255.255.255.0)
  - D/S on WANs (internet, cloud): nodes spread across multiple subnets, traffic is routed
  - Cloud systems give us ***plenty*** of virtual HW to build any distributed system (and topology) we desire on-the-fly
    - **And then delete it and start over again !**

DYNAMIC TOPOLOGIES CIRCA 1997

FEEDBACK - 3

- What point(s) remain least clear?
  - A few things for implementing threads for to object servers. . .

OBJECT SERVERS

- Host objects and enable remote client access
- Do not provide a specific service
  - Do nothing if there are no objects to host
- Support adding/removing hosted objects
- Provide a home where objects live
- Objects, *themselves*, provide "services"
- Object parts
  - State data
  - Code (methods, etc.)

OBJECT SERVERS - 2

- Consider the implications of object server threading designs:
- **How would these designs impact the implementation of mutual exclusion (*synchronized access to shared memory*)?**
  - Single thread of control for object server
    - Entire server operates as a sequential thread
  - One thread for each object
    - Server has multiple threads, one per object
    - How many clients share each object instance?
    - Objects automatically protected against concurrent access
  - Servers use separate thread for client requests
    - Must implement concurrency
    - Classes should be implemented to be thread-safe

CH. 3.4: SERVERS

WAN REQUEST DISPATCHING

- Goal: minimize network latency using WANs (e.g. Internet)
- Send requests to nearby servers
- Request dispatcher: routes requests to nearby server
- **Example**: Domain Name System
  - Hierarchical decentralized naming system
- Linux: find your DNS servers:

```
# Find you device name of interest
nmcli dev
# Show device configuration
nmcli device show <device name>
```


DNS EXAMPLE

- Ping www.google.com in WA from wireless network:
  - nslookup: 6 alternate addresses returned, choose (74.125.28.147)

Latency to ping VA server in WA: ~64x
Massive slowdown because WA is a wireless network

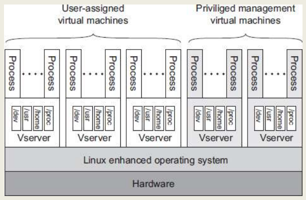
Latency to ping WA server in VA: ~2.8x
Less of a slowdown because VA is a cloud VM

- Local wireless network, ping us-east-1 google (172.217.9.196):
- Ping 74.125.28.147: Average RTT=81.637ms (11 attempts, 15 hops)

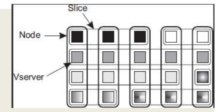
EXAMPLE: PLANETLAB

- Unstructured heterogeneous cluster of servers
- Similar to grid but organized as cluster (no grid middleware)
- Testbed established in 2002 for computer networking and distributed systems research
- Organizations share nodes in the cluster

Leverages Linux Vservers
Early "containers"
similar to Docker



PLANETLAB - 2



- **Slices**: set of Vservers running across PlanetLab
- Acts as a virtual server cluster (similar to Amazon VPC)
- **Node manager**: manages Vservers running on a host
- **Slice creation service (SCS)**: To create virtual server clusters
- Clients must be **slice authorities** to create cluster
- **Rspec**: resource specification
  - Specifies resource requirements for a slice
- **Rcap**: resource capability
  - Specifies resource capabilities of nodes

VSERVERS

- Early container based approach
- Vservers share a single operating system kernel
- Primary task is to support a group of processes
- Provides separation of name spaces
- Linux kernel maps process IDs: host OS → Vservers
- Each Vserver has its own set of libraries and file system
- Similar name separation as the "`chroot`" command
- Additional isolation provided to prevent unauthorized access among Vservers directory trees

VSERVERS - 2

- **Advantages of Vservers (containers) vs. VMs:**
- Simpler resource allocation
- Possible to overbook resources by leveraging dynamic resource allocation - **Example: CPU or RAM** *(assignment 0, config 1)*
- VMs reserve a block of memory
- Containers can oversubscribe memory
  - Memory not formally reserved
  - Linux kernel shares memory among processes
  - Swap filesystem can use disk as extended RAM
- Memory sharing important for PlanetLab
  - Early nodes had limited memory (e.g. 4 GB)
- Vserver hogging most memory reset when out of swap space

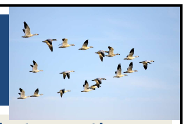
CH. 3.5: CODE MIGRATION

CODE MIGRATION

- Distributed systems can support more than **passing data**
- Some situations call for **passing programs** (e.g. *code*)
- **Live migration** – moving code while it is executing
- **Portability** – transferring code (running or not) across heterogeneous systems:

  Mac OS X → Windows 10 → Linux
- Code migration enables ***flexibility*** of distributed systems
  - Topologies can be dynamically reconfigured on-the-fly

PROCESS MIGRATION

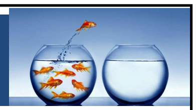


- Move an entire process from one node to another
- Motivation is always to address performance
- Process migration is slow, costly, and intricate
  - Need to pause, save intermediate state, move, resume
  - Consider application ***specific*** vs. ***agnostic*** approaches
- What would be:
  an **application agnostic** approach to migration?
  an **application specific** approach?
- What are advantages and disadvantages of each?

PROCESS MIGRATION - 2

- Move processes:
  from heavily loaded → lightly loaded nodes
- When do we consider a node as heavily loaded?
  - Load average
  - CPU utilization
  - CPU queue length
- Which process(es) should be moved?
  - Must consider ***resource requirements*** for the task
- Where should process(es) be moved to?

MOTIVATIONS FOR MIGRATION



- Can migrate **processes** or entire **virtual machines**
- **Goals:**
  - Off-loading machines: reduce load on oversubscribed servers
  - Loading machine: ensure machine has enough work to do
  - Minimize total hosts/servers in use to save energy/cost
- **VM migration:**
- Migrate complete VMs with apps to lightly loaded hosts
- Generally, VM migration is easier than process migration
- **Is VM migration application specific or agnostic?**

LOAD DISTRIBUTION ALGORITHMS

- Make decisions concerning allocation and redistribution of tasks across machines
- Provide resource management for compute intensive systems
- Often CPU centric
  - Algorithms should also account for other resources
  - Network capacity may be larger bottleneck that CPU capacity

WHEN TO MIGRATE?

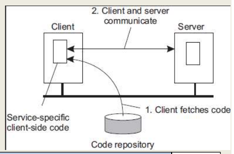
- Decisions to migrate code often based on qualitative reasoning or adhoc decisions vs. formal mathematical models
  - Difficult to formalize solutions due to heterogeneous composition and state of systems and networks
- **Is it better to migrate code or data?**
- **What factors should be considered?**
  - Size of code
  - Size of data
  - Available network transfer speed
  - Cost of data transfer
  - Processing power of nodes
  - Cost of processing
  - Are there security requirements for the data?

APPROACHES TO CODE MIGRATION

- Traditional clients
  - Client interacts with server using specific protocol
  - Tight coupling of client->server limits system flexibility
  - Difficult to change protocol when there are ***many*** clients
- Dynamic web clients
  - Web browser downloads client code immediately before use
  - New versions can readily be distributed

DYNAMIC WEB CLIENTS

- Advantages
  - Client code loaded in as necessary
  - Discarded when no longer needed
  - Can easily change the client/server protocol
- Disadvantages
  - Security: we have to trust the code
  - Downloading client requires network bandwidth & time



CODE MIGRATION

- Sender-initiated: (upload the code)… e.g. Github
- Receiver-initiated: (download the code)… e.g. web broswer
- **Remote cloning**
  - Produce a copy of the process on another machine while parent runs

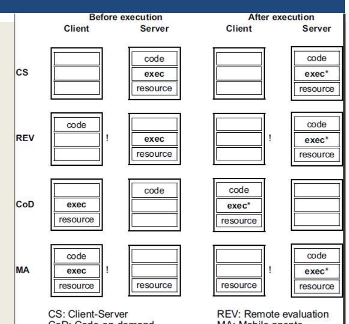
CODE MIGRATION - 2

- What is migrated?
  - ***Code*** segment
  - ***Resource*** segment (device info)
  - ***Execution*** segment (process info: data, statem stack, PC)
- **Weak mobility**
  - Only ***code*** segment, no state
  - Code always restarts
- **Strong mobility**
  - ***Code*** + ***execution*** segment
  - Process stopped, state saved, moved, resumed
  - Represents true ***process migration***

CODE MOBILITY TYPES

- CS: Client-Server
- REV: Remote Evaluation
- CoD: Code-on-demand
- MA: Mobile agents
- Where does state get modified?
- State is stored in **exec**

* *shows what is modified*



MIGRATION OF HETEROGENEOUS SYSTEMS

- Assumption: code will always work at new node
- Invalid if node architecture is different (*heterogeneous*)
- What approaches are available to migrate code across heterogeneous systems?
- Intermediate code
  - 1970s Pascal: generate machine-independent intermediate code
  - Programs could then run anywhere
  - **Today:** web languages: Javascript, Java
- VM Migration

VIRTUAL MACHINE MIGRATION

- Four approaches:

1. **PRECOPY**: Push all memory pages to new machine *(slow)*, resend modified pages later, transfer control
2. **STOP-AND-COPY**: Stop the VM, migrate memory pages, start new VM
3. **ON DEMAND**: Start new VM, copy memory as needed
4. **HYBRID**: PRECOPY followed by brief STOP-AND-COPY

- **What are some advantages and disadvantages of 1-4?**

1. **PRECOPY**: Push all memory pages to new machine *(slow)*, resend modified pages later, transfer control
2. **STOP-AND-COPY**: Stop the VM, migrate memory pages, start new VM
3. **ON DEMAND**: Start new VM, copy memory pages as needed
4. **HYBRID**: PRECOPY and followed by brief STOP-AND-COPY

- **What are some advantages and disadvantages of 1-4?**
  - 1/3: no loss of service
  - 4: fast transfer, minimal loss of service
  - 2: fastest data transfer
  - 3: new VM immediately available

  - 1: must track modified pages during full page copy
  - 2: longest downtime - unacceptable for live services
  - 3: prolonged, slow, migration
  - 3: original VM must stay online for quite a while
  - 1/3: network load while original VM still in service

QUESTIONS

EXTRA SLIDES