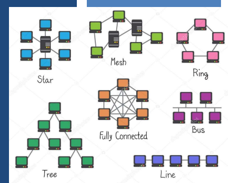
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Distributed Systems: Goals and Types

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



OBJECTIVES

- Course demographics survey – missing surveys
- Feedback from 9/28
- Design goals of distributed systems
  - Resource sharing / availability
  - Distribution transparency
  - Openness
  - Scalability
- Activity: Design goals of distributed systems
- Types of distributed systems
  - HPC, cluster, grid, cloud
  - Distributed information systems
  - Pervasive systems
- Research directions

FEEDBACK – 9/28

- What is the difference between extensibility and scalability?
  - Extensibility – ability for a system implementation to be extended with additional functionality
  - Scalability – ability for a distributed system to scale (up or down) in response to client demand
- What is the loss of availability in a distributed system?
  - Availability refers to “uptime”
  - How many 9s
  - (1 – (down time/ total time)) * 100%
- Transparency: term is confusing
  - Generally means “exposing everything”, obfuscation is better
  - Distribution transparency means the implementation of the distribution cannot be seen

---

FEEDBACK - 2

- What do we mean by replication transparency?
  - Resource are automatically replicated (by the middleware/framework)
  - That fact that there are distributed system has replica nodes is unbeknownst to the users
- How does replication improve system performance?
  - By replicating nodes, system load is “distributed” across replicas
  - Distributed reads – many concurrent users can read
  - Distributed writes – when replicating data, requires synchronization of copies

---

DESIGN GOALS OF DISTRIBUTED SYSTEMS

- Support for sharing resources (accessibility)
- Distribution transparency
- Openness (avoiding vendor lock-in)
- Scalability
- Back to slide 15. . .

OPENNESS

- **Extensible**: easy to reconfigure, add, remove, replace components from different developers
- Example: replace the underlying file system of a distributed system
- To be open, we would like to ***separate policy from mechanism***
- Policy may change
- Mechanism is the technological implementation
- Avoid coupling policy and mechanism
- Enables flexibility
- Similar to separation of concerns, modular/OO design principle

OPENNESS: EXAMPLE

- Separation of policy and mechanism: **web browser caching**
- **Mechanism:** browser provides facility for storing documents
- **Policy:** Users decide which documents, for how long, …
- Goal: Enable users to set policies dynamically
- For example: browser may allow separate component plugin to specify policies
- **Tradeoff**: management complexity vs. policy flexibility
- Static policies are inflexible, but are easy to manage as features are barely revealed.

TYPES OF SCALABILITY

- **Size scalability**: distributed system can grow easily ***without*** impacting performance
  - Supports adding new users, processes, resources
- **Geographical scalability**: users and resources may be dispersed, but communication delays are negligible
- **Administrative scalability:** An administratively scalable system
- Most systems only account for size scalability
- One solution is to operate multiple parallel independent nodes

SIZE SCALABILITY

- Centralized architectures have limitations
- At some point a single central coordinator/arbitrator node can't keep up
  - Centralized server: limited CPU, disk, network capacity
- Scaling requires surmounting bottlenecks

Lloyd W, Pallickara S, David O, Lyon J, Arabi M, Rojas K. Migration of multi-tier applications to infrastructure-as-a-service clouds: An investigation using kernel-based virtual machines. InGrid Computing (GRID), 2011 12th IEEE/ACM International Conference on 2011 Sep 21 (pp. 137-144). IEEE.

GEOGRAPHIC SCALABILITY

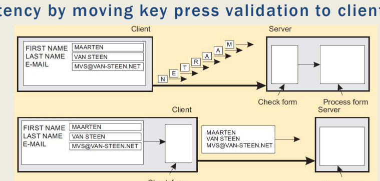
- Nodes dispersed by great distances
  - Communication is slower, less reliable
  - Bandwidth may be constrained
- How do you support synchronous communication?
  - Latencies may be higher
  - Synchronous communication may be too slow and timeout
  - WAN links can be unreliable

ADMINISTRATIVE SCALABILITY

- Conflicting policies regarding usage (payment), management, and security
- How do you manage security for multiple, discrete data centers?
- Grid computing: how can resources be shared across disparate systems at different domains, etc. ?

APPROACHES TO SCALING

- **Hide communication latencies**
  - Use asynchronous communication to do other work and hide latency
  - Remote server runs in parallel in the background – client not locked
  - Separate event handler captures return response from server
- Hide latency by moving key press validation to client:



APPROACHES TO SCALING - 2

- Partitioning data and computations across machines
- Just one copy
  - Where is the copy?
- Move computations to the client
  - Thin client → thick client
  - Edge, fog, cloud….
- Decentralized naming services (DNS)
- Decentralized information services (WWW)

APPROACHES TO SCALING - 3

- Replication and caching – make copies of data available at different machines
- Replicated file servers and databases
- Mirrored web sites
- Web caches (in browsers and proxies)
- File caches (at server and client)

PROBLEMS WITH REPLICATION

- Having multiple copies leads to inconsistency (cached or replicated)
- Modifying one copy invalidates all of the others
- Keeping copies consistent requires global synchronization
- Global-synchronization prohibits large-scale up
  - Best to synchronize just a few copies or synchronization latency becomes too long, entire system slows down!
  - ***Consider how synchronization time increases with system size***
- Can inconsistencies be tolerated?
  - Current temperature and wind speed for weather.com
  - Bank account balance – for a read only statement
  - Bank account balance – for a transfer/withdrawal transaction

DESIGN GOALS ACTIVITY

DEVELOPING DISTRIBUTED SYSTEMS

- Developing a distributed system is a formidable task
- Many issues to consider:
- Reliable networks do not exist
- Networked communication is inherently insecure

FALSE ASSUMPTIONS ABOUT DISTRIBUTED SYSTEMS

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

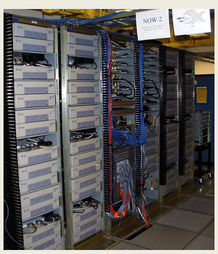
TYPES OF DISTRIBUTED SYSTEMS

TECHNOLOGY INNOVATIONS LEADING TO CLOUD COMPUTING

- Super computers
  - Huge multiprocessor system which shares RAM
  - Technically “not distributed”
  - Hardware all in one location
- High performance distributed computing
  - Cluster computing
  - Grid computing
  - Cloud computing
  - Virtualization
  - Others

EARLY CLUSTER - 1996

- Inktomi search engine on Network of Workstations (NOW) @ UC Berkeley in 1996



CLUSTER COMPUTING

- Cluster computing (clustering)
  - Cluster is a group of independent IT resources interconnected as a single system
  - Off-the-shelf computers connected via a high-speed network
  - Servers configured with homogeneous hardware and software
    - Identical or similar RAM, CPU, HDDs
  - Design emphasizes redundancy as server components are easily interchanged to keep overall system running
    - Example: if a RAID card fails on a key server, the card can be swapped from another redundant server
  - Clusters provide “warm” replication of servers
    - Key servers are duplicated to provide HW failover to ensure high availability (HA)

COMPUTER CLUSTERS

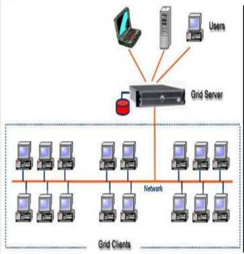
- Clusters: Commodity computers connected by Ethernet switches
  - More scalable than conventional servers
  - Much cheaper than conventional servers
  - Dependability through extensive redundancy
  - Few administrators for 1000s servers
- Careful selection of identical HW/SW
  - Interchangeable components
- Virtual Machine Monitors simplify operation

GRID COMPUTING

- On going research area since early 1990s
- Distributed heterogeneous computing resources organized into logical pools of loosely coupled resources
- For example: heterogeneous servers connected by the internet
- Resources are heterogeneous and geographically dispersed
- Grids use middleware software layer to support workload distribution and coordination functions
- Aspects: load balancing, failover control, autonomic configuration management
- Grids have influenced clouds contributing common features: networked access to machines, resource pooling, scalability, and resiliency

GRID COMPUTING - 2

How Grid computing works ?



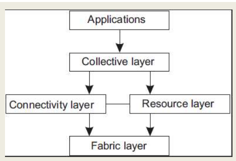
In general, a grid computing system requires:

- **At least one computer, usually a server, which handles all the administrative duties for the System**
- **A network of computers running special grid computing network software.**
- **A collection of computer software called middleware**

GRID COMPUTING - 3

- Grids are built by federating compute resources together from many organizations
- Virtual organization
  - Users from different organizations participate together in a virtual organization
  - Jobs belonging to a virtual organization can harness resources owned by the virtual organization
- Grids bring together heterogeneous hardware owned by many organizations

GRID COMPUTING LAYERS

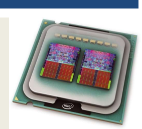


- Application layer
  - Applications operating within a virtual organization sharing grid resources
- Middleware layers
  - Collective layer
    - Provides access to multiple resources
    - Services for discovery, allocation, scheduling, data replication, etc.
  - Connectivity layer
    - Communication protocols to support transactions across grid
    - Data transfer, access to resources, security (authentication) protocols
  - Resource layer
    - Manages access to a single resource via fabric layer
    - Configuration of a specific resource
    - Security (access control)
- Fabric layer

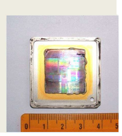
CLOUD COMPUTING NIST GENERAL DEFINITION

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and reused with minimal management effort or service provider interaction"…

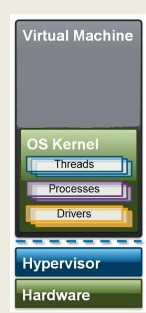
MICROPROCESSORS ADVANCEMENTS



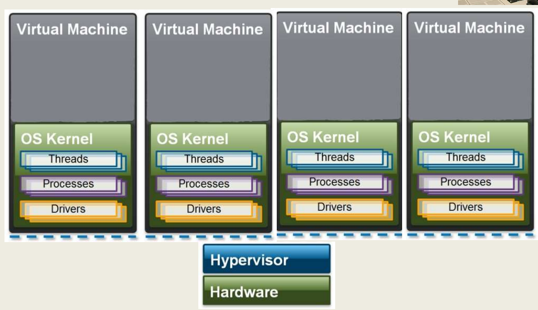
- Smaller die sizes (microns)
  - Lower voltages
  - Improved heat dissipation
  - Energy conservation
  - More transistors, but with similar clock rates
- Leads to multicore CPUs
  - Means to harness new transistor density
  - Improve overall computational throughput
- How do we utilize many-core processors?



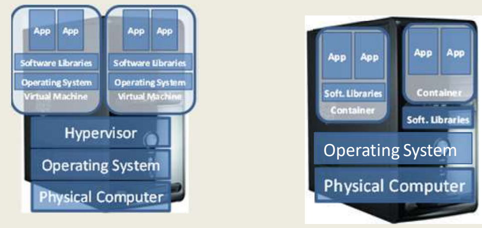
VIRTUALIZATION



VIRTUALIZATION



CONTAINERIZATION

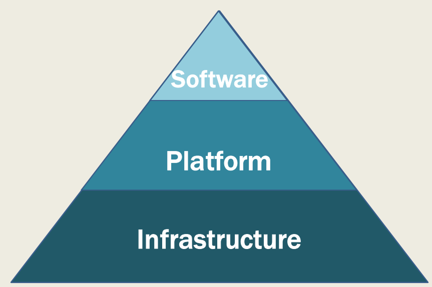


Virtualization **Containerization**

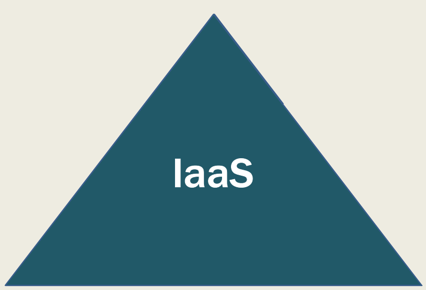
HOW WAREHOUSE SCALE COMPUTING BECAME THE CLOUD

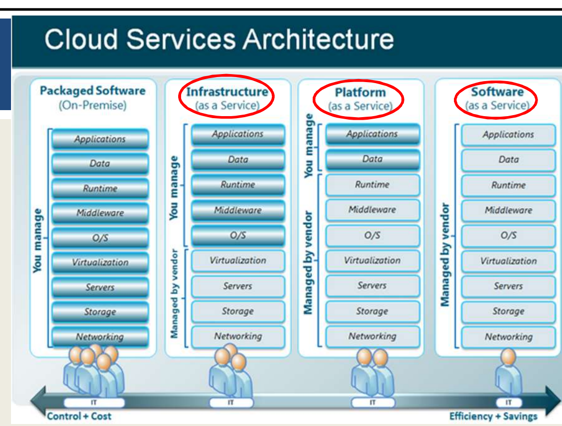
- Clusters grew from 1,000 servers to 100,000+ based on customer demand for SaaS apps
- Economies of scale pushed down costs by 3X to 8X
  - Purchase, house, operate 100K vs. 1K computers
  - Traditional datacenters utilization is ~ 10% - 20%
- Earn $ offering pay-as-you-go computing at prices lower than customer's costs;
  - Scalable → as many computers as customer needs

CLOUD COMPUTING STACK



CLOUD COMPUTING STACK





PUBLIC CLOUD EXAMPLE: NETFLIX

- Amazon Elastic Compute Cloud (EC2)
  - Continuously run 20,000 to 90,000 VM instances
  - Across 3 regions
  - Host 100s of microservices
  - Process over 100,000 requests/second
  - Host over 1 billion hours of monthly content

NETFLIX

PUBLIC CLOUD COMPUTING

- Offers computing, storage, communication at ¢ per hour
- No premium to scale:

```
        1000 computers   @    1 hour
   =      1 computer     @ 1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Leading examples:
  Amazon Web Services, Google App Engine, Microsoft Azure
- Amazon runs its own e-commerce on AWS!
- Billing models are becoming increasingly granular
  - By the minute, second, tenth of a second
  - Obfuscated pricing-Lambda $0.0000002 per request
    $0.000000208 to rent 128MB / 100-ms

October 3, 2017 | TCSS 558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma | L2.38

PUBLIC CLOUD COMPUTING

- Offers computing, storage, communication at ¢ per hour
- No pre[mium to scale:]
- Illusio[n of infinite scalability to cloud user]
- As ma[ny computers as you can afford]
- Leadin[g examples:]
  Amazo[n Web Services, Google App Engine, Microsoft Az]ure
- Amaz[on runs its own e-commerce on AWS!]
- Billing [models are becoming increasingly granular]
  - By the minute, second, tenth of a second
  - Obfuscated pricing-Lambda $0.0000002 per request
    $0.000000208 to rent 128MB / 100-ms

m4.large example
2 vCPU cores, 8 GB RAM, Intel Xeon E5-2666 v3
10¢ an hour
24hrs/day
30 day/month → $72.00/month
on-demand EC2 instance

AWS Lambda? **$346.51**

October 3, 2017 | TCSS 558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma | L2.39

QUESTIONS

October 3, 2017 | TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma | L2.40

EXTRA SLIDES

41