General Lecture Topic

- 15D Loosely-Coupled Systems
- 15E Cloud Models
- 15H Eventual Consistency
- OS wrap-up

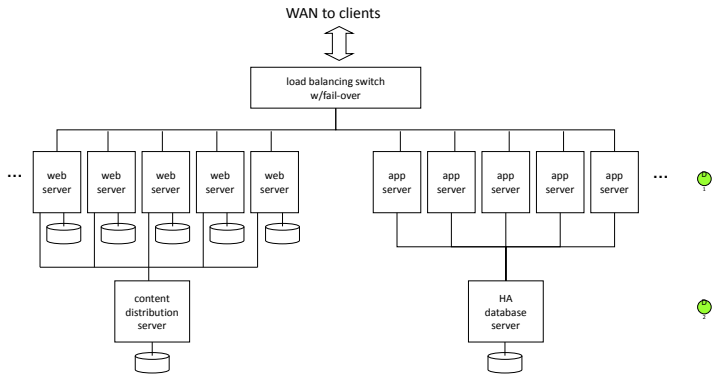
Lessons Learned

- consensus protocols are expensive
  - they converge slowly and scale poorly
- systems have a great many resources
  - resource change notifications are expensive
- location transparency encouraged non-locality
  - remote resource use is much more expensive
- a greatly complicated operating system
  - distributed objects are more complex to manage
  - complex optimizations to reduce added overheads
  - new failure modes, complex recovery procedures

Loosely Coupled Systems

- Characterization:
  - a parallel group of independent computers
  - serving similar but independent requests
  - minimal coordination and cooperation required
- Motivation:
  - scalability and price performance
  - availability – if protocol permits stateless servers
  - ease of management, reconfigurable capacity
- Examples:
  - web servers, Google search farm, Hadoop

Horizontal Scalability w/HA



(elements of architecture)

- farm of independent servers
  - servers run same s/w, serve different requests
  - may share a common back-end database
  - goal: no single points of failure
- front-ending switch
  - single IP address for entire service
  - distributes requests among available servers
  - can do both load balancing and failover
- service protocol
  - stateless servers and idempotent operations
  - global resources sharded across available servers
  - no persistent client-to-server binding

Horizontally scaled performance

- individual servers are very inexpensive
  - blade servers may be only $100-$200 each
- scalability is excellent
  - 100 servers deliver approximately 100x performance
- service availability is excellent
  - front-end automatically bypasses failed servers
  - stateless servers and client retries fail-over easily
- the challenge is managing thousands of servers
  - automated instantiation (don't install, clone!)
  - global configuration services
  - self monitoring, self-healing systems

Limited Transparency Clusters

- Single System Image Clusters had problems
  - all nodes had to agree on state of all objects
  - lots of messages, lots of complexity, poor scalability
- What if they only had to agree on a few objects
  - like cluster membership and global locks
  - fewer objects, fewer operations, much less traffic
  - objects could be designed for distributed use
    - leases, commitment transactions, dynamic server binding
- Simpler, better performance, better scalability
  - combines best features of SSI and Horizontally scaled

Limited LocationTransparency

- what things look the same as local?
  - remote file systems
  - remote terminal sessions, X sessions
  - remote procedure calls
- what things don't look the same as local?
  - primitive synchronization (e.g. mutexes)
  - basic Inter-Process Communication (e.g. signals)
  - process create, destroy, status, authorization
  - Accessing devices (e.g. tape drives)

Goals of Distributed Computing

- better services
  - **scalability**
    - services too big to run on a single computer
    - **grow system capacity to meet growing demand**
  - **improved reliability and availability**
  - improved ease of use, reduced CapEx/OpEx
- new services
  - applications that span multiple system boundaries
  - global resource domains, services (vs. systems)
  - complete location transparency

Clouds: Applied Horizontal Scalability

- Many servers, continuous change
  - dramatic fluctuations in load volume and types
  - continuous node additions for increased load
  - nodes and devices are failing continuously
  - continuous and progressive s/w updates
- Most services delivered via switched HTTP
  - clients/server communication is over WAN links
  - large (whole file) transfers to optimize throughput
  - switches route requests to appropriate servers
  - heavy reliance on edge caching

How to Exploit a Cloud

- Replace physical machines w/virtual machines
  - cloud provides inexpensive elastic resources
  - co-locate the data and computations
- Run massively parallel applications
  - requiring huge numbers of computers
- Massively distributed systems are very difficult
  - to design, build, maintain and manage
  - horizontally scaled services are much easier
- How can we make exploiting parallelism easy?
  - new, tool supported, programming models
  - encapsulate complexity of distributed systems

Geographic Disaster Recovery

- Cloud reliability/availability are key
  - one data center serves many (10^3-10^7) clients
- Local redundancy can only provide 4-5 nines
  - fires, power and communications disruptions
  - regional scale (e.g. flood, earthquake) disasters
- Data Centers in distant Availability Zones
  - may be running active/active or active/stand-by
  - key data is replicated to multiple data centers
  - traffic can be redirected if a primary site fails

WAN-Scale Replication

- WAN-scale mirroring is slow and expensive
  - much slower than local RAID or network mirroring
- Synchronous Mirroring
  - each write must be ACKed by remote servers
- Asynchronous Mirroring
  - write locally, queue for remote replication
- Mirrored Snapshots
  - writes are local, snapshots are mirrored
- Fundamental tradeoff: durability vs. latency

WAN-Scale Consistency

- CAP theorem - it is not possible to assure:
  - Consistency (all readers see the same result)
  - Availability (bounded response time)
  - Partition Tolerance (survive node failures)
- ACID databases sacrifice partition tolerance
- BASE semantics make a different trade-off
  - Basic Availability (most services most of the time)
  - Soft state (there is no global consistent state)
  - Eventual consistency (changes propagate, slowly)

Dealing with Eventual Consistency

- distributed system has no single, global state
  - state updates are not globally serialized events
  - different nodes may have different opinions
- expose the inconsistencies to the applications
  - ask cloud, receive multiple/inconsistent results
  - let each application reconcile the inconsistencies
    - merge, most recent, highest precedence, ask for help
- BASE semantics are neither simple nor pretty
  - they embrace parallelism and independence
  - they reflect the complexity of distributed systems

Distributed Computing Reformation

- systems must be more loosely coupled
  - tight coupling is complex, slow, and error-prone
  - move towards coordinated independent systems
- move away from old single system APIs
  - local objects and services don't generalize
  - services are obtained through messages (or RPCs)
  - in-memory objects, local calls are a special case
- embrace the brave new (distributed) world
  - topology and partnerships are ever-changing
  - failure-aware services (commits, leases, rebinds)
  - accept (deal with) distributed (BASE) semantics

Changing Architectural Paradigms

- a "System" is a collection of services
  - interacting via stable and standardized protocols
  - implemented by app software deployed on nodes
- Operating Systems
  - manage the hardware on which the apps run
  - implement the services/ABIs the apps need
- The operating system is a platform
  - upon which higher level software can be built
  - goodness is measured by how well it does that job

What Operating Systems Do

- Originally (and at the start of this course)
  - abstract heterogeneous hardware into useful services
  - manage system resources for user-mode processes
  - ensure resource integrity and trusted resource sharing
  - provide a powerful platform for developers
- None of this has changed, but …
  - notion of a self-contained system becoming obsolete
  - hardware and OS heterogeneity is a given
  - most important interfaces are higher level protocols
- Operating Systems continue to evolve as
  - new applications demand new services
  - new hardware must be integrated and exploited

What Operating Systems Do

Could you and I with Him conspire,
to grasp this sorry scheme of things entire,
Would we not shatter it to bits – and then
re-mould it nearer to the heart's desire?

Omar Khayyam
The Rubaiyat, XCIX

Was uns nicht umbringt macht uns nur stärker!

(That which does not kill us
only makes us stronger!)

Friedrich Wilhelm Nietzsche

Man and Superman

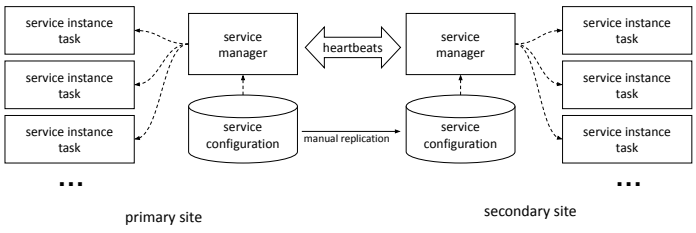
Closed Book Final Exam

- Exam Times
  - Mon 12/11 14:00-17:00
- Part I … covering lectures 16-28
  - ten questions, similar to mid-term
  - basic understanding of concepts and principles
- Part II … covering the entire course
  - six questions (choose any three to answer)
  - analyze/approach practical problems

Supplementary Slides

Loosely Coupled Availability

(Kimberlite HA Linux platforms)



There is global resource view or synchronization of resource state. The systems are completely Independently of one-another, but agree on the division of tasks, and are prepared to take over the partner's work load if he fails.