Cloud Computing

Next up

- Warehouse scale computers
  - How to build/program data centers
- Google software stack
  - GFS
  - BigTable
  - Sawzall
  - Chubby
  - Map/reduce

What is cloud computing

- Illusion of infinite computing resources available on demand
  - Scale-up for most apps
- Elimination of up-front commitment
  - Small initial investment, scale only as needed
- Pay-per-use on short-term basis
  - transfer purchase risk (will equipment be used?) to cloud provider
- Result: Cost-associativity
  - Use of 1 CPU for 100 hours costs the same as 100 CPUs for one hour

Historical Context

- Utility computing – late 1960's
  - Genesis of Multics at MIT/GE
  - Central pool of mainframes serve dial-up customers
  - Developed as time-sharing bureaus
- Single server, many clients
  - No real connectivity between servers

Death of Utility Computing

- Killer Micros: cheap PCs that afforded more flexibility, more power then central computer
  - text terminals couldn't compete on flexibility, GUI
  - Micros more cost effective EXCEPT for management
    - upgrades, patches, software installs

Rise of Cloud Computing

- Data centers ride commodity-part tide
  - Computing in the large now similar in price to client-side computing
- Internet properties develop technology to efficiently deploy, manage, serve large clusters
  - container-based data centers
  - scalable, replicated, distributed, reliable storage
- Rich programming models allow better client-side UI
  - AJAX, JavaScript, HTML

Enabling Technologies

- Virtualization
  - For platform-as-a-service and remote management
- Web Services
  - XML-RPC, SOA, etc. – APIs to services
- Cluster management experience by large internet companies
  - E.g. Amazon, Google, Microsoft
- Cluster management software
  - Storage: GFS, Dryad

Cloud Computing Models

- First wave: Software-as-a-Service (SaaS)
  - Rather than install software on customer machines, run it in a provider data center
    - E.g.: HotMail, Salesforce.com
  - Web-scale software: serve everybody

Software-as-a-Service

- Think GMail, google docs
- What are the benefits?
  - No end-customer management; provider can do on-line upgrades, provisioning
  - No hardware purchase
  - Vendor sees how customer uses software, can adapt
- What are the downsides?
  - The internet goes down
  - Edge bandwidth can be low
  - Browser lacks polish of real UIs

Platform-as-a-Service

- Second wave (PaaS)
  - Providers supply generic platform for running Software-as-a-Service AND collaborative web software
    - Allow third parties to host their applications on common infrastructure
    - Google AppEngine, Windows Azure (sort of)
  - Still API-rich
    - Provide many services (load balancing, request distribution, scheduling)

Infrastructure-as-a-Service

- Provide virtual machines, networking
  - Augment with higher level services:
    - Storage
    - Monitoring
    - Security
- Customer provides complete software stack
- Prevents lock in
  - Few APIs to write against beyond data management

Coming: Data as a service

- Step 1: storage as a service
  - Amazon Simple Storage Service
  - Windows Sky Drive
  - Cheaper than managing storage locally
- Step 2: data as a service
  - Windows Azure Data Market
    - Rent data from someone else rather than collecting your own
    - E.g make large data sets available for sale
      - Geographical databases
      - Historical data sets (e.g. weather)

What drives Cloud Computing

- Economics:
  - Cost of providing computing services locally
  - Utilization of enterprise infrastructure
  - Provisioning for worst-case

Cost of Provisioning

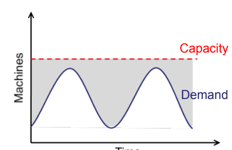
- Within a company:
  - Real estate/power is expensive (often better used for office space)
  - Lacks large economies of scale
  - Management typically 1 person 10-100 machines
- Within a data center:
  - Put where land/power is cheap
  - Buy in bulk (containers, thousands of machines, gigabits of bandwidth)
  - Low management overhead: lots of self-managing systems, 1 person/ 1000-10,000 machines

Elastic Provisioning

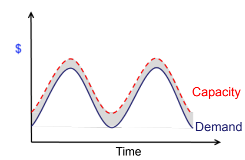
- Companies must provision for worst case
  - Leads to low utilization (1-20%) most of the time
  - Leads to overload some of the time (slashdot effect)
  - Hard to grow rapidly if popularity surges
    - e.g. doubling every day
    - May lose customers if bad service
- Cloud provisioning can start large, multiplex machines over many customers
  - Higher average utilization
  - Higher resource availability to surging sites
  - Cheap to decommission resources as popularity falls

Cloud Economics 101

- Cloud Computing User: Static provisioning for peak - wasteful, but necessary for SLA



"Statically provisioned" data center

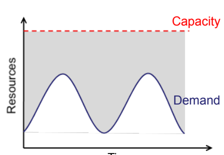


"Virtual" data center in the cloud

Unused resources

Risk of Under Utilization

- Underutilization results if "peak" predictions are too optimistic



Static data center

Unused resources

Elastic Provisioning (2)

- Billing based on usage
  - Shifts risk from customer to provider
  - Separates cost of memory, I/O, disk, CPU and bill appropriately
  - Reduces risk of trying cloud computing
- Shifts capital expenses (buying things) to operational expense (cost of providing service)
  - Small upfront cost good for starting out

When is data center computing cheaper?

- Unpredictable/fluxuating workloads
  - Expensive to provision your own
  - Good for:
    - startups who don't know their popularity
    - big companies with predictable peak loads (e.g. christmas, olympics)
- Not super-CPU intensive
  - Raw CPU, memory, disk is cheaper locally (not scalable, not replicated)
  - Extra charge for renting relatively small (2.5 x for CPU, 20-50% for disk)
  - QUESTION: Why disk cheaper? it must be replicated ….

Cloud Economics

- Some costs cheaper in cloud
  - Fixed costs of buildings, buy machines, power
    - Cheaper when bought in buld
  - Variable costs cheaper
    - Bandwidth much cheaper in bulk (e.g. 10x)
    - System management much cheaper (e.g. 10-100x)
      - Massive redundancy
      - Massive homogeneity

Utility Computing Arrives

- Amazon Elastic Compute Cloud (EC2)
- "Compute unit" rental: $0.10-0.80/hr.
  - 1 CU ≈ 1.0-1.2 GHz 2007 AMD Opteron/Xeon core

"Instances"	Platform	Cores	Memory	Disk
Small - $0.10 / hr	32-bit	1	1.7 GB	160 GB
Large - $0.40 / hr	64-bit	4	7.5 GB	850 GB – 2 spindles
XLarge - $0.80 / hr	64-bit	8	15.0 GB	1690 GB – 3 spindles

- Billing rounded to nearest hour; pay-as-you-go storage also available
- A new paradigm (!) for deploying services?

Energy & Cloud Computing?

- Cloud Computing saves Energy?
  - Don't buy machines for local use that are often idle
  - More efficient cooling
    - Newest data centers are air cooled only using outside air
- Better to ship bits as photons over fiber vs. ship electrons over transmission lines to spin disks, power processors locally
  - Clouds use nearby (hydroelectric) power
  - Leverage economies of scale of cooling, power distribution

What apps can move to the cloud?

- Compute over same data multiple times
  - Amortize cost of uploading data
- Perfectly scalable parallel apps (batch)
  - Can compute N times faster on N more machines
  - E.g. NY times, WA post scanning documents
- CPU/data intensive apps for mobile devices
  - provide heavy lifting, data persistence
- Apps with widely variable resource demands
  - Animoto rendering service

Differences to App Writers

- State no longer lives in a file system
  - Storage systems for shared data (e.g. S3, databases) instead
- Applications scale both ways
  - Up for bigger loads
  - Down for multi-tenancy
    - Idle cycles aren't free

What apps cannot move to the cloud?

- Video games?
  - Heavy client-side CPU component
  - But:
    - Compute graphics in cloud and stream to client
    - OnLive.com
- Banking
  - Need better security
- Offline apps

Cloud vs. Grid

- Grid: more about batch scheduling parallel jobs
  - Each machine generally runs only jobs for one customer
  - Jobs typically use multiple machines
  - Jobs are not interactive
  - Big data set, large data objects
  - Scientific computing
- Cloud
  - Multi-tenancy (multiple customers on a machine)
  - Resource-based billing
  - Public (often)
  - Fine-grained storage

Cloud Computing Platforms

- Before Amazon:
  - Rackspace, Sun Grid Compute Facility
  - Real machines, no virtualization
  - QUESTION: why a problem?
    - Cannot scale down
    - Must fully utilize machine (no sharing)
- Amazon EC2:
  - Virtual machine + set of images
  - Services: block storage, database, content distribution
- Google AppEngine
  - Python/Java environment for web apps
  - Google handles scalability with BigTable, load balancing, scaling, authentication

More platforms

- Windows Azure
  - .net execution environment, like a JVM
  - Storage in a file system, database
  - Less than a whole machine, no choice of OS, but can run almost-arbitrary applications
- Who wins?
  - AppEngine very limited in what it can do, but does more for you.
  - EC2 most flexible – run any code – seems to have most marketshare now
    - For apps with simple scalability, services are enough

Platform Issues

- Lock-in
  - If you write for Windows Azure, you will always run on Windows Azure (and pay rent)
  - Same for AppEngine

Challenges

- Getting data into clouds
  - Bandwidth at endpoints much lower than within cloud.
  - Loading large data sets can be slow
  - Can actually ship disks now…
- Efficiency
  - How can you use idle CPU cycles of interactive web sites?
    - They demand low latency, cannot have more than 40% utilization or so unless they fall over under load
    - Idle periods are often short (10ms-1sec), to short for condor-style scheduling

Cloud Computing Issues

- Infrastructure
  - How do you build a data center?
- Programming model
  - How do you write an app for the cloud?
  - Map/reduce
  - Azure, AppEngine
- Reliability/scalability
  - How do you write apps to have these?
- Storage
  - What is the right storage abstraction?
    - Files
    - Databases
    - Tables
    - key-value?
- Security
  - How do you provide the security of locally-controlled, off-the-internet nodes?
  - How do you store the data you trust
  - How do you trust the CPU to compute properly
  - How do you trust the service to maintain your privacy
    - Would Barnes & Noble host on Amazon?
- Efficiency
  - How do you make data centers efficient (power and computing)?
  - They still may have low utilization

Cloud Programming models

- Data-driven programming
  - Examples:
    - Map-reduce
    - Dryad/Linq
    - Pig
  - Goal: how express computation over distributed data, flow of data through system
- Data-drive web sites
  - Google App Engine
- Interactive web sites
  - Example:
    - Google search infrastructure
  - Challenge:
    - Low latency response from many servers