Internet service

- Service: a set of actions that can be used to accomplish a goal
- Service Providers, Service consumers agree upon or have an understanding of the goal
- Internet Service: service over the net
  - Ebay service: Buy and sell goods
  - On-line Banking service: offers banking related services
  - Amazon: sells goods
  - E-loan: sells mortgages
  - Google: search web pages

 definitions

- Architecture of a typical internet service
  - General architecture is
    - 1(LB) + 3 (web, app, db) + 1(storage) tiers
  - Can be just 3 tiers (LB + AppServer + storage)
  - Hasn’t changed much since late 1990s
- Main components of the architecture
  - Users, network, Redirector/Load balancer, webs server, app server, db, storage
Users

- User population growing globally
  - eBay, 69 M users, 1 B hits a day
  - Yahoo, 400 M users
  - Hotmail, 150 M users
  - Google, 0.2 B queries per day
  - Amazon, 50 M users

Load balancers (LB)

- Requests from users classified and directed/routed
- Differentiation based on source, destination addresses
  - User address, ISP etc Level-3 redirector or rotating DNS

Hardware load balancers

- Use IP rewriting or MAC layer forwarding
- Servers on a simple LAN connected to a load balancer
  - (S,D) → (S,S1) on return (S1,S) → (D,S)
- Use an algorithm to decide which packets go to which server
- Too many servers, then use bridge
- Know which Server is on which LAN segment

Load balancing using Packet filters

- Each server has a packet filter
- Load balancer broadcasts all packets
- Each server decides what packet to absorb
- Common algorithm /hash function needed to decide packet filters
- Servers have Virtual IP address (the same for all machines on the cluster) and a dedicated IP address for each machine for non-cluster related traffic
- E.g., Windows Load balancing system -WLBS
Load balancing using IP tunnels

- All machines have an extra IP — Virtual IP address
- The tunnel is established between two machines (can be geographically separated)
- Reply goes directly to client

Encapsulate/decapsulate

Using direct routing (mac rewriting)
**Direct routing**
- LB and servers must be on the same network
- No tunnel overhead

**L4, L7 load balancers**
- Level 4, understands TCP connections
  - Differentiation based on port numbers
  - Web requested routed differently than SMTP request
- Level 7, understands application level information
  - Differentiation based on application-level info
    - Can parse URLs
    - Look at parameter of GET

**Load balancers**
- DNS rotation
  - Static, if redirected node has failed
  - Takes a long time to switch to another node
- Level 3 switch
  - Fast, can detect failed nodes
  - Bottleneck link into the switch, single point of failure
  - Difficult to balance load for global users
- Level 4, 7 switch
  - Need to parse port numbers, URLs at line speed
  - Good TCP-level load partitioning
  - Bottleneck link into the switch, single point of failure
  - Can use from-end applications to do smarter balancing

**Servers**
- Web servers
  - Process HTML requests, render HTML content, process user input
- App servers
  - Process Business logic, business rules, security, authentication
  - Prepare data for web servers
- DB servers
  - Execute DBMS queries, retrieve user+app data
- Storage servers
  - SAN, NAS, combo, Disk farms
  - Maintain persistent data pertaining to service

*Lessons from Giant-Scale Services – Internet Computing – July, ’01
- Eric Brewer*
Background: WWW & Internet Services

- Tremendous growth in network size since early 1990s
  - WWW, search
  - Commercialization
  - Cheap PCs

- Services' popularity
  - Integral part; e.g., Yahoo, Ebay, Google Gmail, Amazon, Buy.com, weather.com, etc.
  - Growing client base: Google handles ~300 million requests/day

Scale

<table>
<thead>
<tr>
<th>Service</th>
<th>Nodes</th>
<th>Queries</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOL web cache</td>
<td>&gt;200</td>
<td>2.7B/day</td>
<td>4-CPU DEC 4010s</td>
</tr>
<tr>
<td>Internet search engine</td>
<td>800</td>
<td>&gt;60M/day</td>
<td>2-CPU Sun Workstations</td>
</tr>
<tr>
<td>Gnutella</td>
<td>&gt;600</td>
<td>&gt;250M/day</td>
<td>PC Based</td>
</tr>
<tr>
<td>Anonymous web-based e-mail</td>
<td>&gt;200</td>
<td>250M/day</td>
<td>FreeBSD PCs</td>
</tr>
</tbody>
</table>

Table 1: Example Clusters for Giant-Scale Services

Ebay Stats from 2003

- 69 Million Registered Users
- 18 Million Items across 28,000 categories
- 2002: Gross Sales = $14.87 Billion
- Global community
- 1 Billion hits per day
- Over 1200 URLs

A Billion Hits a Day – JavaOne ’03
- Deepak Alur, Rajmohan Krishnamurthy and Arnold Goldberg

Hotmail: 170M users [NY Times]
Google: 1B hits/day > 100,000 nodes,
  - Google FS serves a PByte
Ebay: > 1
Scale

Ebay traffic

<table>
<thead>
<tr>
<th>Year</th>
<th>Outbound Traffic (at peak)</th>
<th>Dynamic Page Views (per day)</th>
<th>Searches (per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>410 Mbps</td>
<td>300M</td>
<td>1M</td>
</tr>
<tr>
<td>2002</td>
<td>&gt; 2 Gbps</td>
<td>&gt; 1B</td>
<td>50M</td>
</tr>
<tr>
<td>2005*</td>
<td>&gt; 5 Gbps</td>
<td>&gt; 1B</td>
<td>150M</td>
</tr>
</tbody>
</table>

A Billion Hits a Day – JavaOne ’03
Deepak Alur, Rajmohan Krishnamurthy and Arnold Goldberg
EBay booms as member numbers grow – ENN – Apr, 04
- Ciaran Buckley

Maintenance

- Gmail maintenance cost ->$2/GB [NY Times]

Ebay maintenance
- More than 15 major features go live-to-site every quarter
- Nearly 30K lines of code change per week
- 4 times/day, 5 days/week HTML rolls to the site
- Internationalized releases phased in 3 weeks
- Instantaneous feedback from millions of users

Availability

Service availability is in the range of 2 to 3 nines

Availability metrics

- Uptime (MTBF-MTTR)/MTBF
- Either design with high MTBF or live with low MTBF but provided negligible MTTR
- Basis of PC clusters, server farms
- Improve MTTF and apply best effort to MTBF
### Availability metrics
- **yield** = queries completed/queries offered
  - Not all seconds have equal value
  - Differentiate between off-peak and peak
  - Maintenance, back up during off-peak
- **Harvest** = data available/complete data
  - Reflect all of available data
  - Some portion of the database not available
  - Faults at db can impact harvest
- Replication (yield) vs partition (harvest)

### Data vs capacity
- **DQ** = data per query x queries /second
- Increase Q by replicating servers
  - On a fault, capacity degrades (reduce Q) but D same
- Keep Q constant by partitioning
  - On a fault, harvest declines (reduce D)

### Availability
- **Availability** is the percentage of time when system is operational.
  - \( A = \frac{MTBF}{MTBF+MTTR} \)
  - Availability is typically specified in nines notation.
  - For example 3-nines availability corresponds to 99.9% availability.
  - A 5-nines availability corresponds to 99.999% availability.
- **Availability Downtime**
  - 90% (1-nine) 36.5 days/year
  - 99% (2-nines) 3.65 days/year
  - 99.9% (3-nines) 8.76 hours/year
  - 99.99% (4-nines) 52 minutes/year
  - 99.999% (5-nines) 5 minutes/year
  - 99.9999% (6-nines) 31 seconds/year

### Web services requirements
- **Scalability**: capability to meet demand
  - New processing
  - New storage
  - New networking
  - All additions transparent to end-user
- **Availability**: Provide continuous service
  - Online change of all components (hardware and software)
  - Multiple service sites
  - Multiple network providers
- **Manageability**: tools for automated management (reduce human RTT)
  - Manage the system
  - Change the application several times per year.
  - Add new services several times per year.
Service growth

- Collection of apps, h/w, data, storage ➔ farm
  - Farm grows in two ways
    - Cloning (RACS)
      - Resources are replicated
    - Partitioning (RAPS)
      - Additional resources are added, but load is divided

Cloning (Reliable array of cloned services)

- Shared-nothing RACS
  - Each clone has identical storage
  - Highly available
  - Writes on all copies
  - Good for read-mostly
- Shared-disk a.k.a cluster
  - Clones share common disk pool
  - Storage should be fault-tolerant
  - Update traffic can become bottleneck
  - Use SAN

RACS

- RACS (Reliable Array of Cloned Services)
  - A collection of clones for a particular service
  - Shared-nothing RACS
    - Each clone duplicates the storage locally
    - Updates should be applied to all clone’s storage
  - Shared-disk RACS (cluster)
    - All the clones share a common storage manager
    - Storage server should be fault-tolerant
    - Subtle algorithms need to manage updates (cache invalidation, lock managers, etc.)

Clones and RACS

- Can be used for read-mostly applications with low consistency requirements.
  - I.e., Web servers, file servers, security servers...
- The requirements of cloned services:
  - Automatic replication of software and data to new clones
  - Automatic request routing to load balance the work
  - Route around failures
  - Recognize repaired and new nodes
**Partitioning**

- **PACK**: Add resources but partition data

Data automatically partitioned by application middleware
Failures affect result in partial service failure
Even with Fault tolerant storage, H/W, S/W affects availability

**PACKS**

- Shared disk packs
  - Two or more nodes per pack
    - Can tolerate S/W, H/W faults
  - Packs have access to all storage
    - One Pack can takeover when primary fails
  - Data still partitioned

- Shared-nothing
  - Packs serve different partitions
    - Active-active
      - All nodes are active
      - Failure within a pack, another node takes over
    - Active-passive
      - Secondary-nodes in hot standby mode
      - Takes over only on failure

**RAPS**

- **RAPS** (Reliable Array of Partitioned Services)
  - nodes that support a packed-partitioned service
  - shared-nothing RAPS, shared-disk RAPS
- Update-intensive and large database applications are better served by routing requests to servers dedicated to serving a partition of the data (RAPS).
Case Study

**MSNBC (1999)**
- 42 FE, 2 BE, 5 other boxes,
- DNS/windows Load balancer system

**Hotmail (1999)**
- 2198 FE, 58 BE, 54 other boxes,
- Load director (NAT)

Challenges to Building Scalable Services

A Survey of Microsoft’s Internet Services
Version 1.0
September 24, 1999

Steven Levi and Galen H

Taxonomy for Scaleability

- **Farms of servers:**
  - **Clones:** identical
    - Scaleability + availability
  - **Partitions:**
    - Scaleability
  - **Packs**
    - Partition availability via fail-over

**GeoPlex**
  - for disaster tolerance
  - georedundancy

Summary

- **Scalability technique**
  - *Replicate* a service at many nodes
- **Simplest form of replication**
  - Duplicate both programs and data: **RACS**
- **Data partitioning technique for database/ update-intensive services**
  - Data partitioned: **RAPS**
  - **Packs** make partitions highly available
  - Middleware required for management of failovers
- **Against disaster**
  - The entire farm is replicated to form a **geoplex**
The Google Cluster Architecture

Barroso, Dean and Holzle, IEEE Micro 2003

GOOGLE ARCHITECTURE OVERVIEW

- Reliability
  - Software is the king
  - H/W is commodity, can be swapped
  - so that commodity PCs can be used
    - to build a cluster at a low price

- Design
  - for best aggregate throughput
  - rather than peak server response time

- Building a reliable computing infrastructure
  - from clusters of unreliable commodity PCs
    - RAID for disks; PC clusters is to server farms

SERVING A GOOGLE QUERY

- When user enters a query
  - e.g. [www.google.com/search?q=ieee+society](http://www.google.com/search?q=ieee+society)
  - User browser
    - Domain Name System (DNS) lookup
    - to map to a particular IP address

- Multiple Google clusters distributed worldwide
  - each cluster with a few thousand machines
  - to handle query traffic

Global Google Name Servers

- Over 30 Google clusters around the world.
- DNS based & geo location driven load-balancing:
  - Domain Name: GOOGLE.COM
    - 4 name servers
  - GOOGLE.COM NAME SERVERS
    - Name Server IP Location
      - ns1.google.com 216.239.32.10 Mountain View, CA, US
      - ns2.google.com 216.239.34.10 Mountain View, CA, US
      - ns3.google.com 216.239.36.10 Mountain View, CA, US
      - ns4.google.com 216.239.38.10 Mountain View, CA, US
**CLUSTER ARCHITECTURES**

- Cluster
  - collection of independent computers using switched network to provide a common service
- Main frames, multiprocessor servers offer services at a higher price point.
- Adv of clusters: Scale, cost, repair, failover
- Most web portals (amazon, ebay, etc) use clusters

**DESIGN PRINCIPLES OF GOOGLE CLUSTERS**

- Software level reliability
  - No fault-tolerant hardware features; e.g.
  - redundant power supplies
  - A redundant array of inexpensive disks (RAID)
  - instead tolerate failures in software
- Use replication
  - for better request throughput and availability
- Price/performance beats peak performance
  - CPUs giving the best performance per unit price
  - Not the CPUs with best absolute performance
- Using commodity PCs
  - reduces the cost of computation

**Google rack**

- Google’s racks
  - consist of up to 80 x86-based servers
  - Server components similar to mid-range desktop PC
    - except for larger disk drives
- Ranging
  - from single processor 533-MHz Intel-Celeron based servers
  - to dual 1.4-GHz Intel Pentium III servers
- Servers on each rack interconnected via 100 Mbps Ethernet
- All racks interconnected via a gigabit switch

**THE POWER PROBLEM**

- A mid-range server with dual 1.4-GHz Pentium III processors
  - 90 W of DC power
  - 55 W for the two CPUs
  - 10 W for a disk drive
  - 25 W for DRAM and motherboard
- Typical efficiency of an ATX power supply -> 75%
  - means 120 W of AC power per server
  - roughly 10 (120*80) kW per rack
THE POWER PROBLEM (Cont.)

- A rack
  - 25 ft² of space
  - Corresponding power density: 400 W/ft²
  - With higher-end processors: 700 W/ft²
- Typical power density for commercial data centers: between 70 and 150 W/ft²
  - Much lower than that required for PC clusters
  - Special cooling or additional space required
    - to decrease power density to a tolerable level