Datacenter Technologies

Datacenter Technologies Overview

1. Multi-tier architectures for Internet services
2. Cloud computing
3. Cloud and "big data" technologies

Internet Services

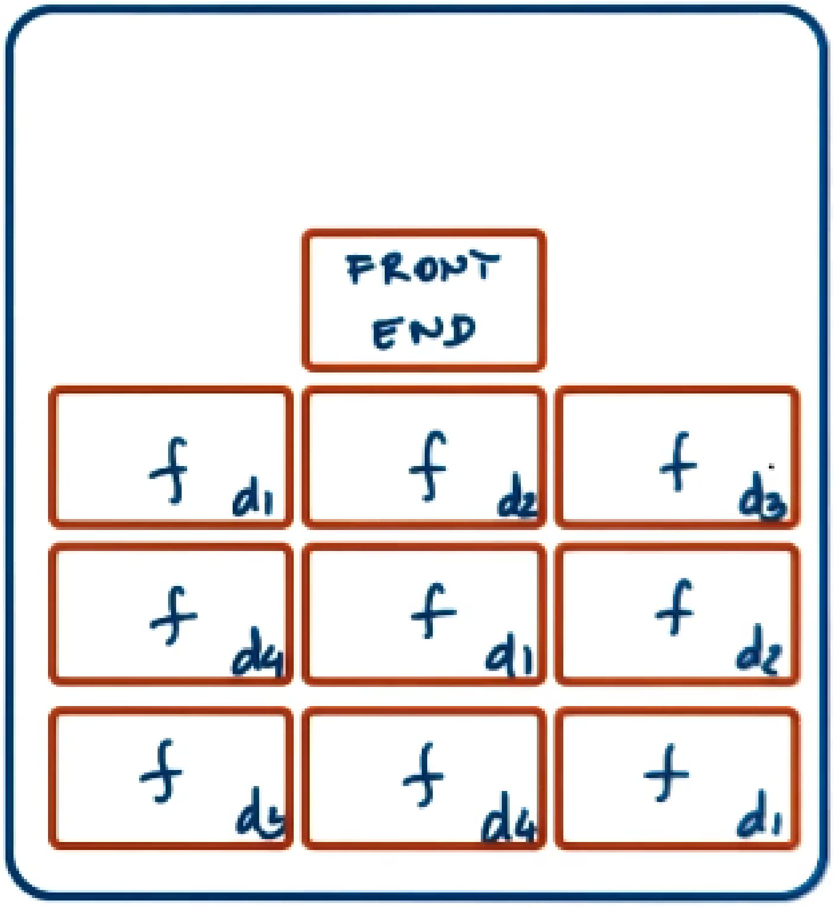
1. Internet service: Any type of service provided via web interface
    - Not necessarily separate processes on separate machines
    - Many available open source and proprietary technologies
    - Middleware: Supporting integrative or value-added software technologies
        - Presentation: Static content
        - Business logic: Dynamic content
        - Database tier: Data store
2. In multiprocess configurations...
    - Some form of IPC used, including RPC/RMI, shared memory, ...

Internet Service Architectures

1. For scale: Multi-process, multi-node
    - "Scale out" architecture
2. "Boss-worker": Front-end distributes requests to nodes
3. "All equal": All nodes execute and possible step in request processing, for any request
4. "Specialized nodes": Nodes execute some specific step(s) in request processing; for some request types
    - Functionally homogeneous
5. Examples: big data analytics, web searches, content sharing, or distributed shared memory (DSM)
    - Functionally heterogeneoous

Homogeneous Architectures

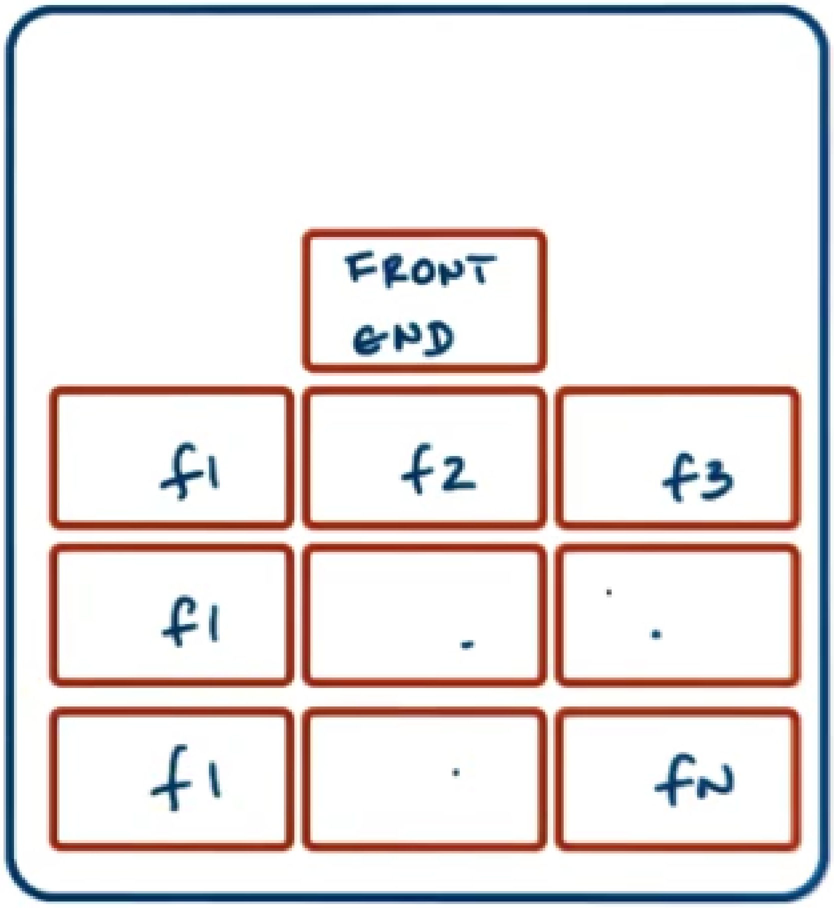
1. Any node can do any processing step
    - Doesn't mean that each node has all data, but each node has access to all data
2. Pros: Keeps front-end simple
3. Cons: How to benefit from caching?
4. To scale, add more processes, servers, storage, ...
    - Management is fairly simple
    - Only works until the resources are so large they can't be managed or reach physical limitations of space
        - Cloud computing addresses these (to some extent)



Homogeneous Architectures

Heterogeneous Architectures

1. Different nodes, different tasks/requests
2. Data doesn't have to be uniformally accessible everywhere
3. Pros: Benefit of locality and caching (each node is specialized for tasks)
4. Cons: Front end is more complex
   - Management is also more complex
5. To scale, understand what is resources are in demand
   - Add more of the appropriate resources/processes



Heterogeneous Architectures

Cloud Computing Poster Child: Animoto

1. Amazon provisioned hardware resources for holiday sale season
    - Resources idle the rest of the year
    - "Opened" access to its resources via web-based APIs
    - Third-party workloads on Amazon hardware for a fee
    - Birth of Amazon Web Services (AWS) and Elastic Compute (EC2)
2. Animoto rented "compute instances" in EC2
    - In April 2008, Animoto became available to Facebook users
        - 750,000 new users in 3 days
        - Mon 50, Tues 400, Wed 500, Fri 3400 (compute instances)
    - Cannot achieve this with traditional in-house machine deployment and provisioning tools

Cloud Computing Requirements

1. Traditional approach:
    - Buy and configure resources
        - Determine capacity based on expected demand (peak)
    - When demand exceeds capacity
        - Dropped requests
        - Lost opportunity
2. Ideal cloud:

- Capacity scales elastically with demand
- Scaling is instantaneous, both up and down
- Cost is proportional to demand, to revenue opportunity
- All of this happens automatically, no need for hacking wizardry
- Can access anytime, anywhere
- Con: Don't "own" resources

3. Requirements:
    - On-demand, elastic resources and services
    - Fine-grained pricing based on usage
    - Professionally managed and hosted
    - API-based access

Cloud Computing Overview

1. Shared resources
    - Infrastructure (physical and virtual) and software/services

2. APIs for access and configuration
    - Web-based, libraries, command line, ...
3. Billing/accounting services
    - Many models: Spot, reservation, entire marketplace
    - Typically discrete quantities: tiny, medium, large, extra-large
4. Managed by cloud provider by some sophisticated software stack
    - OpenStack or VMWare VSphere

Why Does Cloud Computing Work?

1. Law of Large Numbers
    - Per customer there is large variation in resource needs
    - Average across many customers is roughly constant
2. Economies of scale
    - Unit cost of providing resources or service drops at "bulk"
    - Amortize cost of hardware resource over all instances

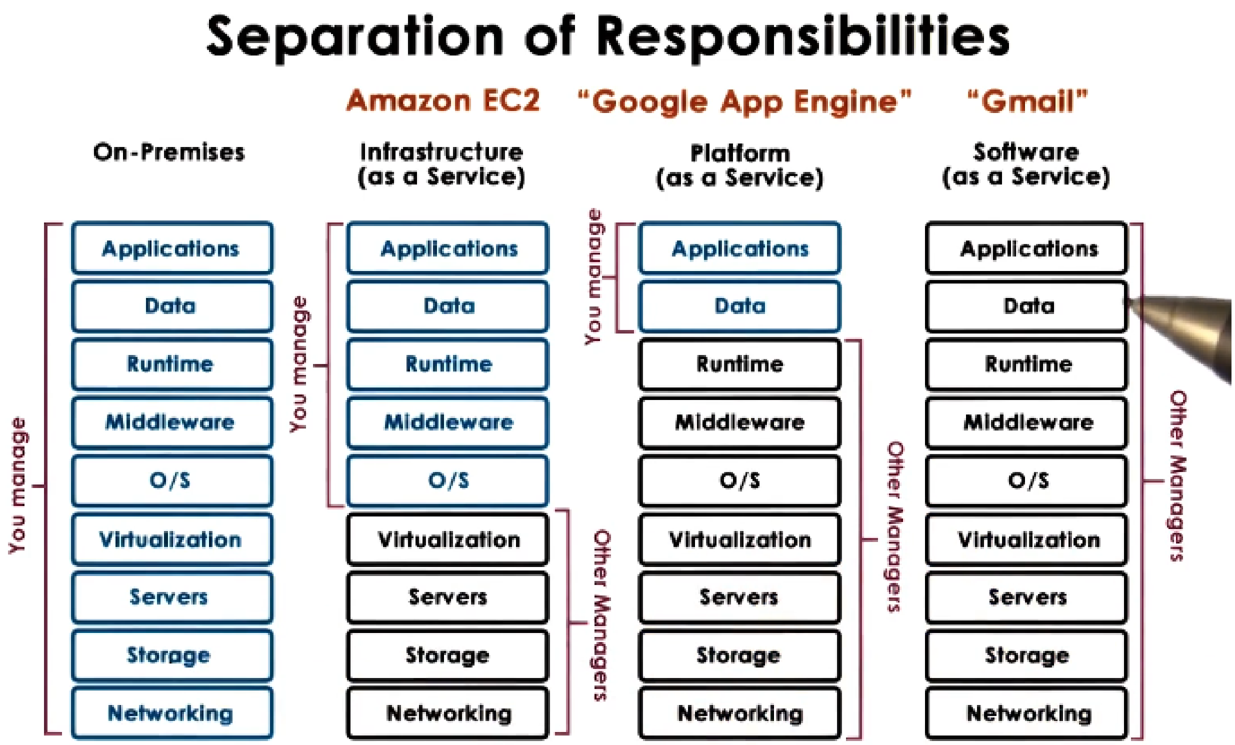
Cloud Computing Vision

1. "If computers of the kind I have advocated become the computers of the future, then computing may some day be organized as a public utility, just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."
    - John McCarthy, MIT Centennial, 1961
2. Computing == Fungible utility
3. Limitations: API lock-in, hardware dependence, latency, privacy, security
4. "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., network, servers, ...) that can rapidly be provisioned and released with minimal management effort or service provider interactions."
    - National Institute of Standards and Technology - October 25, 2011

Cloud Deployment Models

1. Public: Third-party customers/tenants
2. Private: Leverage technology internally
3. Hybrid (public + private): Failover, dealing with spikes, testing
4. Community: Used by certain types of user (public cloud)

Cloud Service Models



Cloud Service Models

Requirements for the Cloud

1. "Fungible" resources - Easily repurposed to support different customers
2. Elastic, dynamic resource allocation methods
3. Scale: Management at scale, scalable resource allocations
4. Dealing with failures
5. Multi-tenancy: Performance and isolation
6. Security (isolation of state being accessed)

Cloud Enabling Technologies

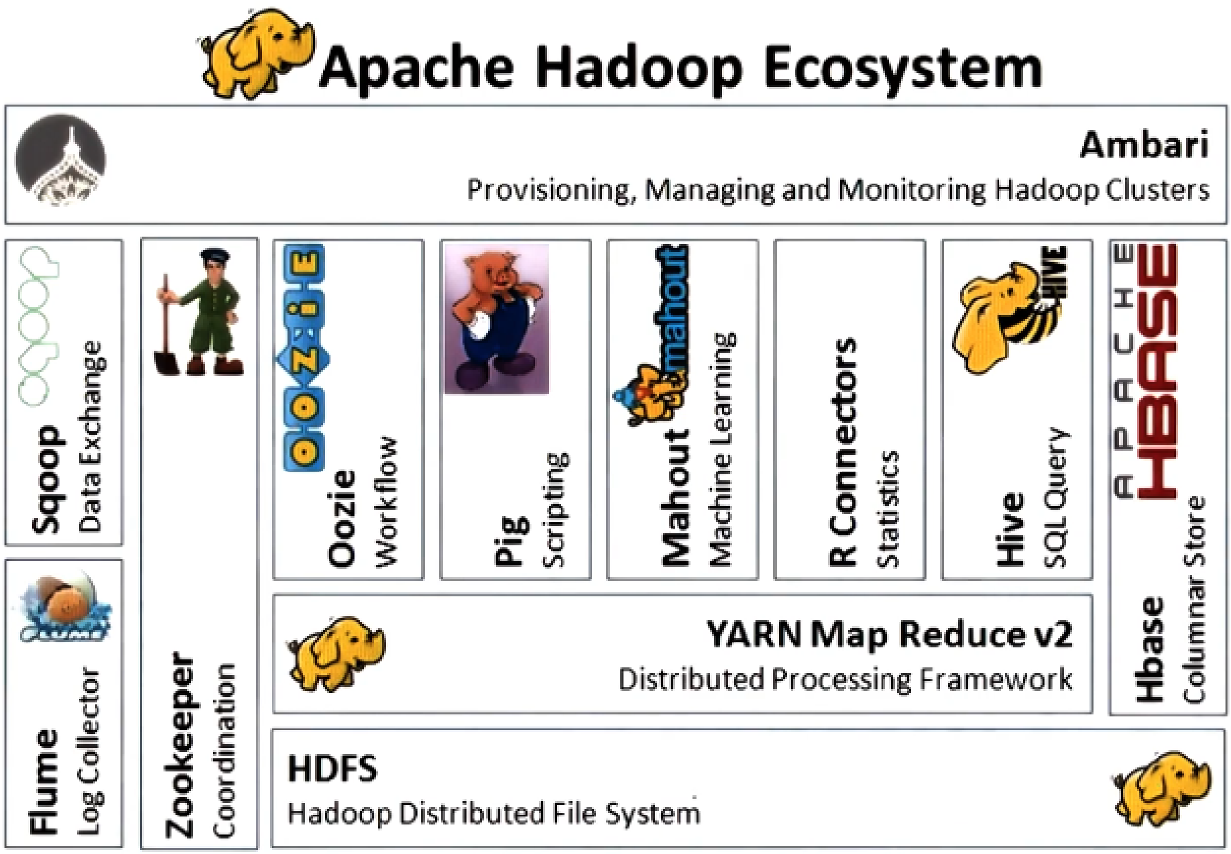
1. Virtualization
2. Resource provisioning (scheduling - Mesos, Yarn)
3. Big data processing (Hadoop, MapReduce, Spark, ...)
    - Storage
        - Distributed FS ("append only")
        - NoSQL, distributed in-memory caches
4. Software-defined networking, storage, datacenters, ...
5. Monitoring: Real time log processing (Flume, CloudWatch, Log Insight)

The Cloud as a Big Data Engine

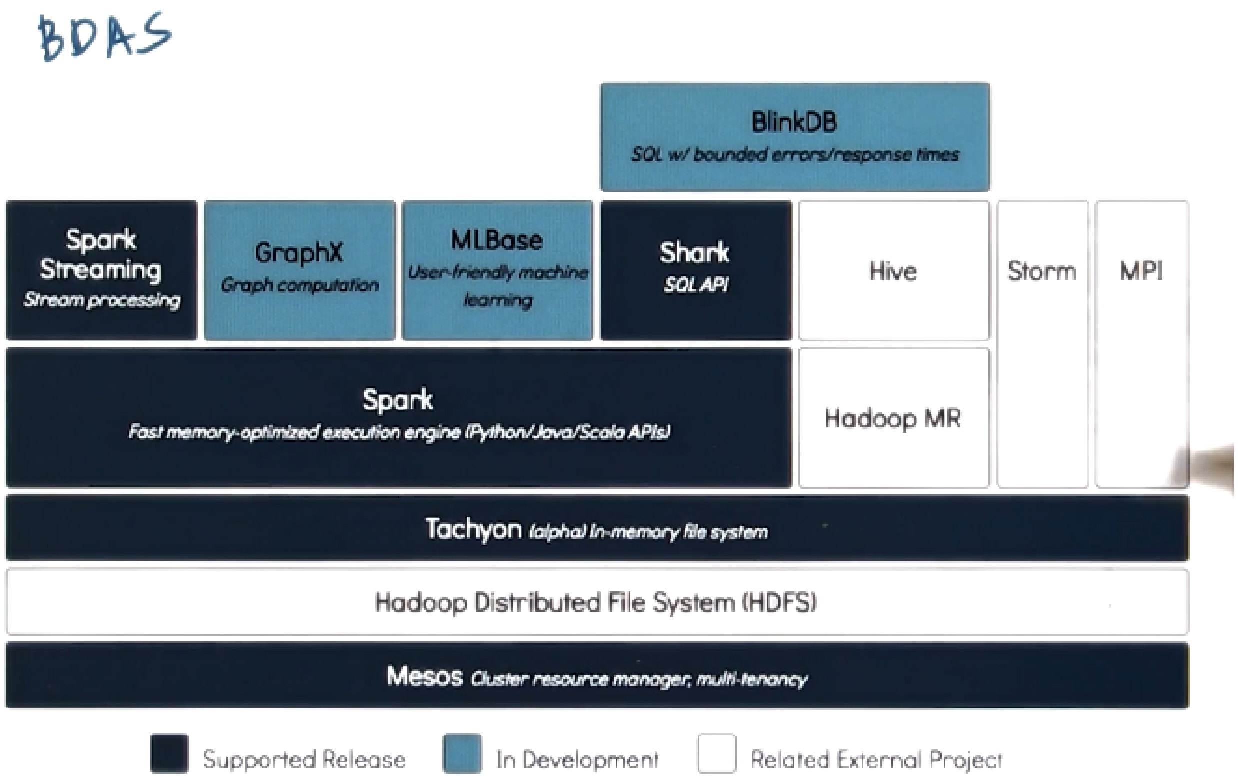
1. Data storage layer
2. Data processing layer
3. Caching layer

4. Language front-ends (querying)
5. Analytics libraries (ML)
6. Continuously streaming data

Example Big Data Stacks



Apache Hadoop Ecosystem



Berkeley Data Analytics Stack Ecosystem