Architecture of Enterprise Applications 24 Cloud Computing

Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
http://reins.se.sjtu.edu.cn/~chenhp
e-mail: chen-hp@sjtu.edu.cn

Contents and Objectives

- Contents
  - Cloud Computing
    - What is cloud computing?
    - Core techniques of cloud computing
    - Obstacles and Opportunities
  - Techniques in Cloud
    - MapReduce
    - Bigtable
    - GFS
    - Percolator
  - Edge Computing

- Objectives
  - 能够根据系统云部署的需求，将本地应用迁移到云中虚拟机集群中进行部署

What is cloud computing?

- There is little consensus on how to define the Cloud
  - A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.
    - In "*Cloud Computing and Grid Computing 360-Degree Compared*"
  - Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), so we use that term. The datacenter hardware and software is what we will call a Cloud.
    - In "*Above the Clouds: A Berkeley View of Cloud Computing*"

What is cloud computing?

- There is little consensus on how to define the Cloud
  - Cloud computing promises to radically change the way computer applications and services are constructed, delivered, and managed. Although the term means different things to different people, and includes a bit of marketing hype and technical redefinition, the potential benefits are clear. Large datacenters permit resource sharing across hosted applications and lead to economies of scale at both the hardware and software level. Software services can obtain seemingly infinite scalability and incremental growth to meet customers' elastic demands. The pay-as-you-go model and rapid provisioning can result in more efficient resource utilization and reduced costs.
    - On "*Introduction to Cloud Computing - ACM Tech pack Committee on CC*"

What is cloud computing?

- Cloud computing is a general term for anything that involves delivering hosted services over the Internet.

- These services are broadly divided into three categories:
  - Infrastructure-as-a-Service (IaaS)
  - Platform-as-a-Service (PaaS)
  - and Software-as-a-Service (SaaS).

- A cloud service has three distinct characteristics that differentiate it from traditional hosting.
  - It is sold on demand, typically by the minute or the hour;
  - it is elastic -- a user can have as much or as little of a service as they want at any given time;
  - and the service is fully managed by the provider.

- Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet and a weak economy, have accelerated interest in cloud computing.

What is cloud computing?

- Cloud in the real world
  - Cloud as reality, as told by industry partners

SaaS	Software-as-a-Service	Google Apps, Microsoft "Software+Services"
PaaS	Platform-as-a-Service	IBM IT Factory, Google AppEngine, Force.com
IaaS	Infrastructure-as-a-Service	Amazon EC2, IBM Blue Cloud, Sun Grid
dSaaS	data-Storage-as-a-Service	Nirvanix SDN, Amazon S3, Cleversafe dsNet

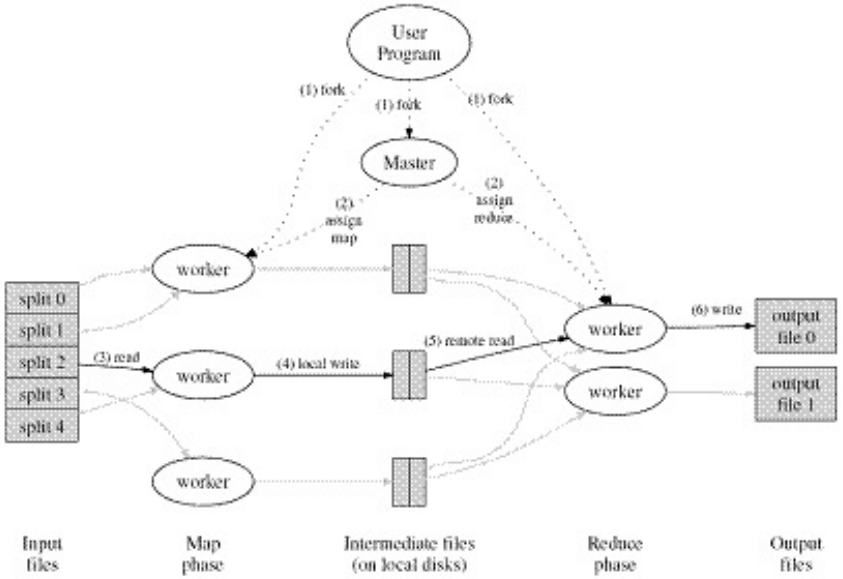
What is cloud computing?

- Cloud in the real world
  - Cloud is different in many dimensions
    - Time to deploy a server
      - Weeks or months -> seconds to minutes
    - Commitment to use service
      - Negotiate & commit year long contract -> select from catalog & Pay as you go
    - Necessary upfront investment
      - $M -> $K
  - Common attributes of clouds
    - Flexible pricing
    - Elastic scaling
    - Rapid provisioning
    - Advanced virtualization

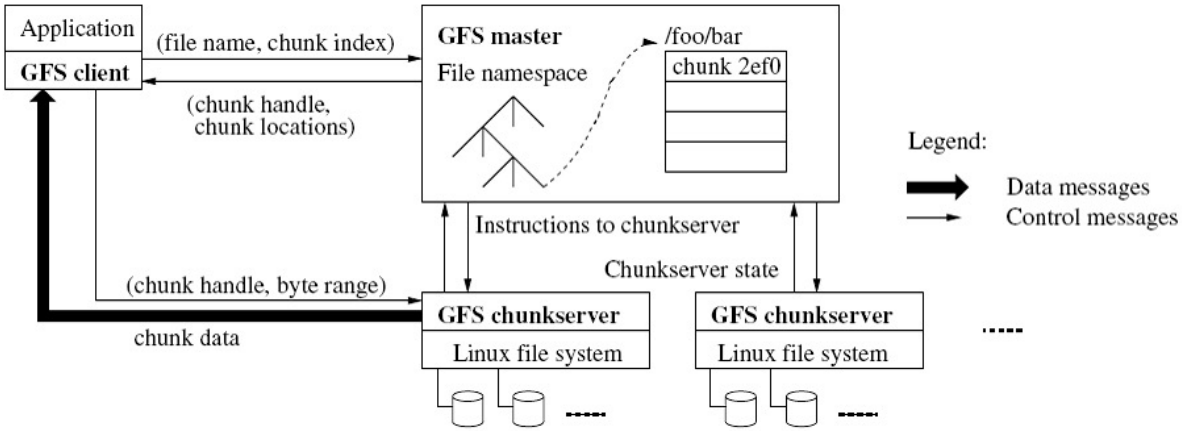
What is cloud computing?

- Cloud in the real world
  - Necessary supprort
    - Workload optimization
    - Integrated service management
      - Service delivery, service request, service monitoring
      - Lowers operational costs, drives efficiency, enhances security
    - Deployment choices
      - Public clouds or private cloud

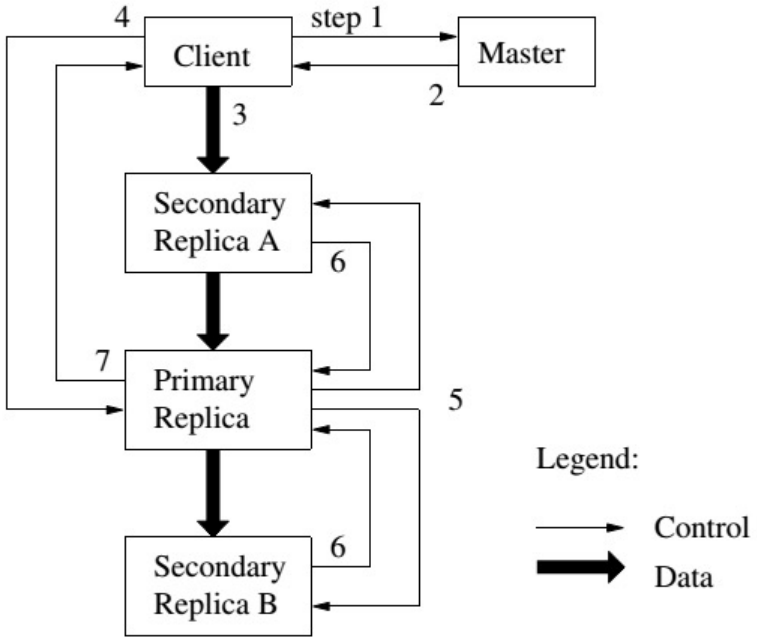
- Google
- MapReduce
  - parallelizes the computation, distributes the data, and handles failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.



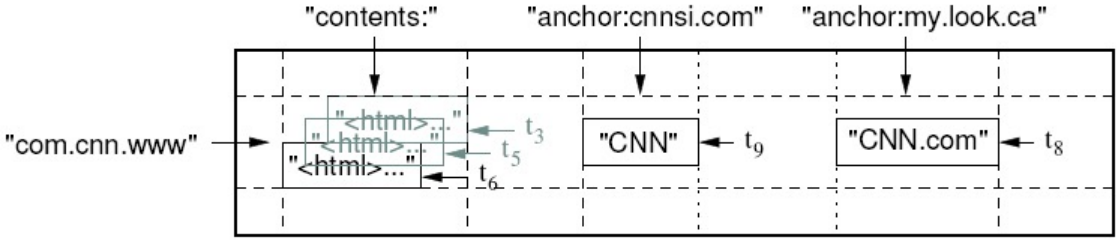
- Google
- Distributed Google File System
  - Google File System(GFS) to meet the rapidly growing demands of Google's data processing needs.



- Write Control and Data Flow

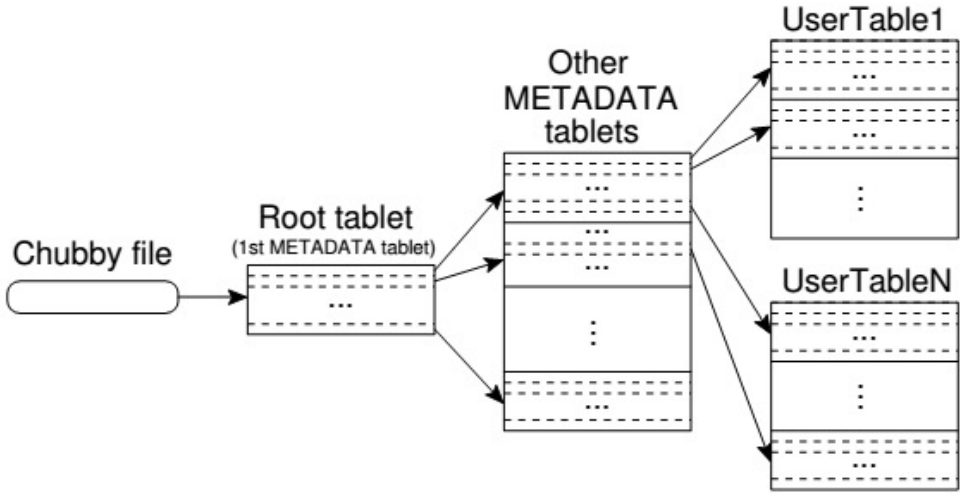


- Google
- Bigtable
  - Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers.

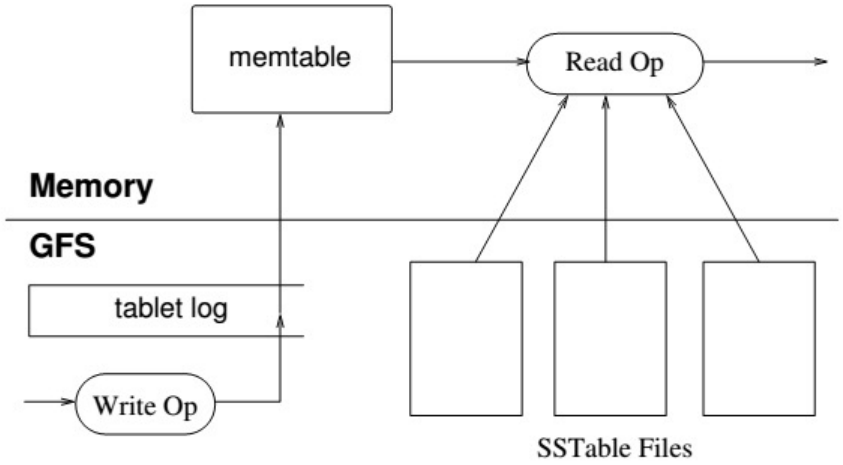


Bigtable

- A Bigtable is a sparse, distributed, persistent multidimensional sorted map.
  - The map is indexed by a row key , column key , and a timestamp; each value in the map is an uninterpreted array of bytes.



- The persistent state of a tablet is stored in GFS.
  - Updates are committed to a commit log that stores redo records.
  - Of these updates, the recently committed ones are stored in memory in a sorted buffer called a memtable; the older updates are stored in a sequence of SSTables.

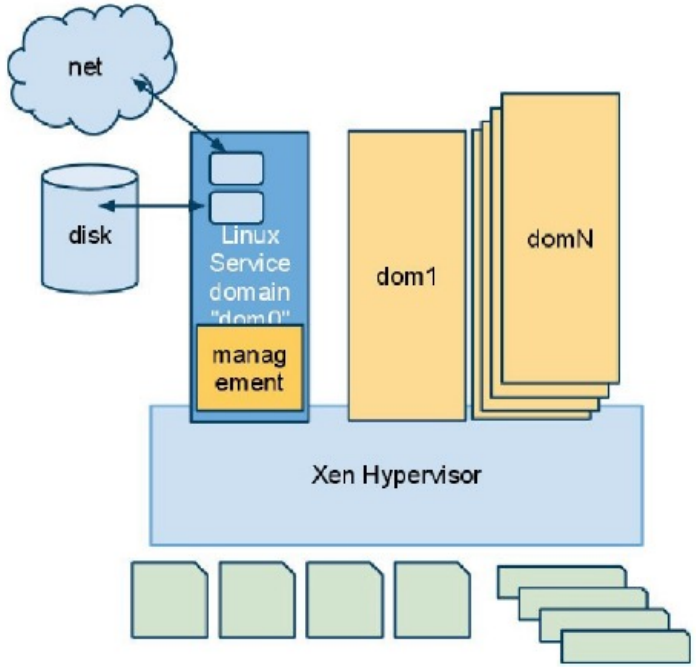


- 

- The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. Hadoop includes these subprojects:
  - **Hadoop Common**: The common utilities that support the other Hadoop subprojects.
  - **HDFS**: A distributed file system that provides high throughput access to application data.
  - **MapReduce**: A software framework for distributed processing of large data sets on compute clusters.

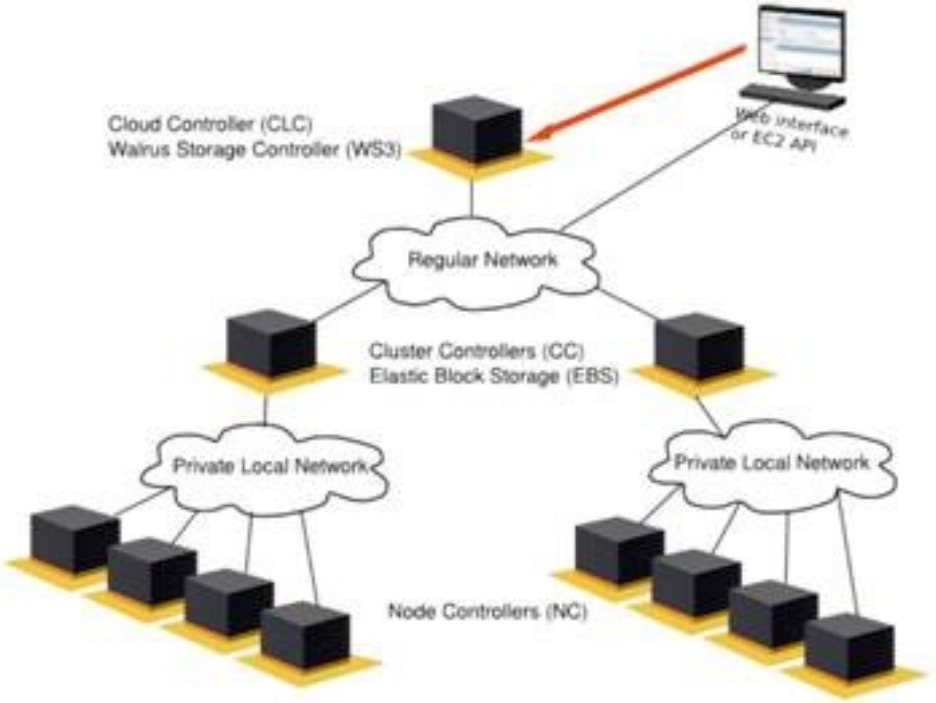
- IBM, Amazon, Yahoo
  - Base stone

- Xen
- Xen Hypervisor
  - Server Virtualization with the Xen Hypervisor
    - Enterprises looking to increase server utilization, consolidate server farms, reduce complexity, and decrease total cost of ownership are embracing server virtualization.
    - The Xen® hypervisor is the fastest and most secure infrastructure virtualization solution available today, supporting a wide range of guest operating systems including Windows®, Linux®, Solaris®, and various versions of the BSD operating systems.



Core techniques of cloud computing

- Eucalyptus Systems
- **Ubuntu Enterprise Cloud** brings Amazon EC2-like infrastructure capabilities inside the firewall. The Ubuntu Enterprise Cloud is powered by Eucalyptus, an open source implementation for the emerging standard of EC2.

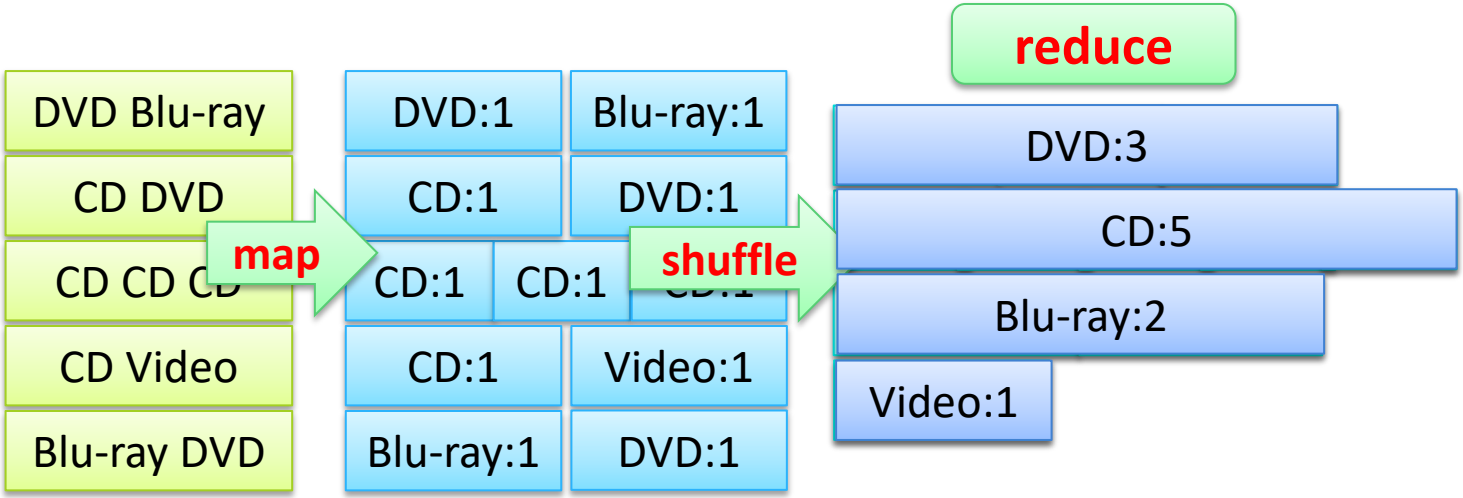


- Map,
  - written by the user, takes an input pair and produces a set of intermediate key/value pairs.
  - The MapReduce library groups together all intermediate values associated with the same intermediate key *I* and passes them to the Reduce function.

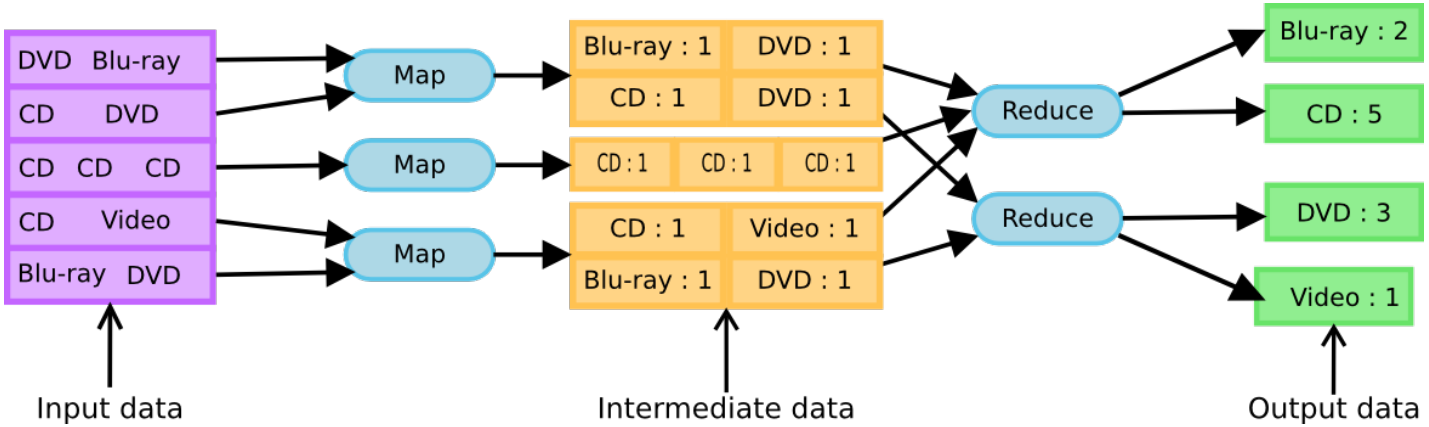
- Reduce,
  - also written by the user, accepts an intermediate key *I* and a set of values for that key .
  - It merges together these values to form a possibly smaller set of values.
  - Typically just zero or one output value is produced per Reduce in vocation.
  - The intermediate values are supplied to the user' s reduce function via an iterator . This allows us to handle lists of values that are too large to fit in memory.

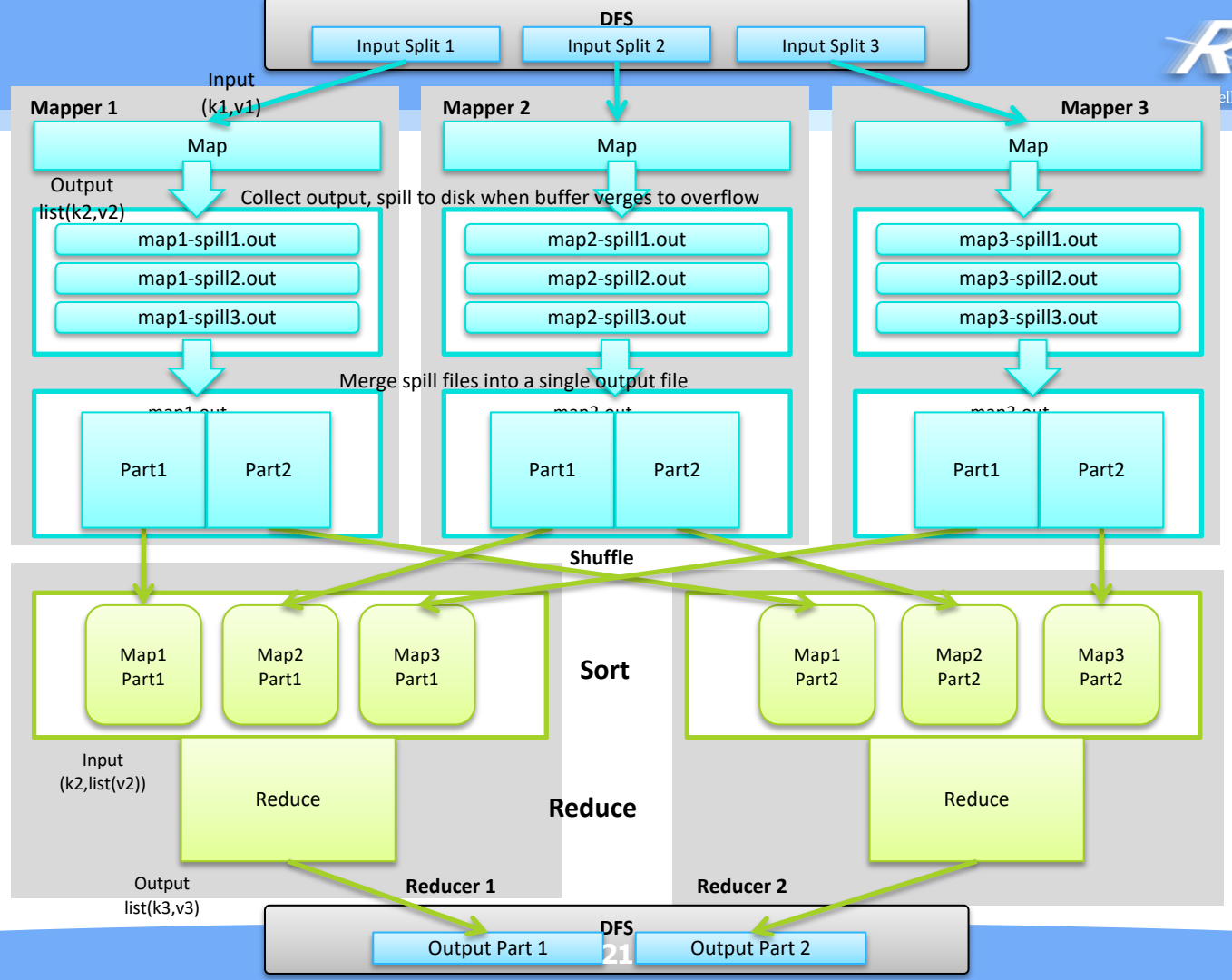
MapReduce Basics

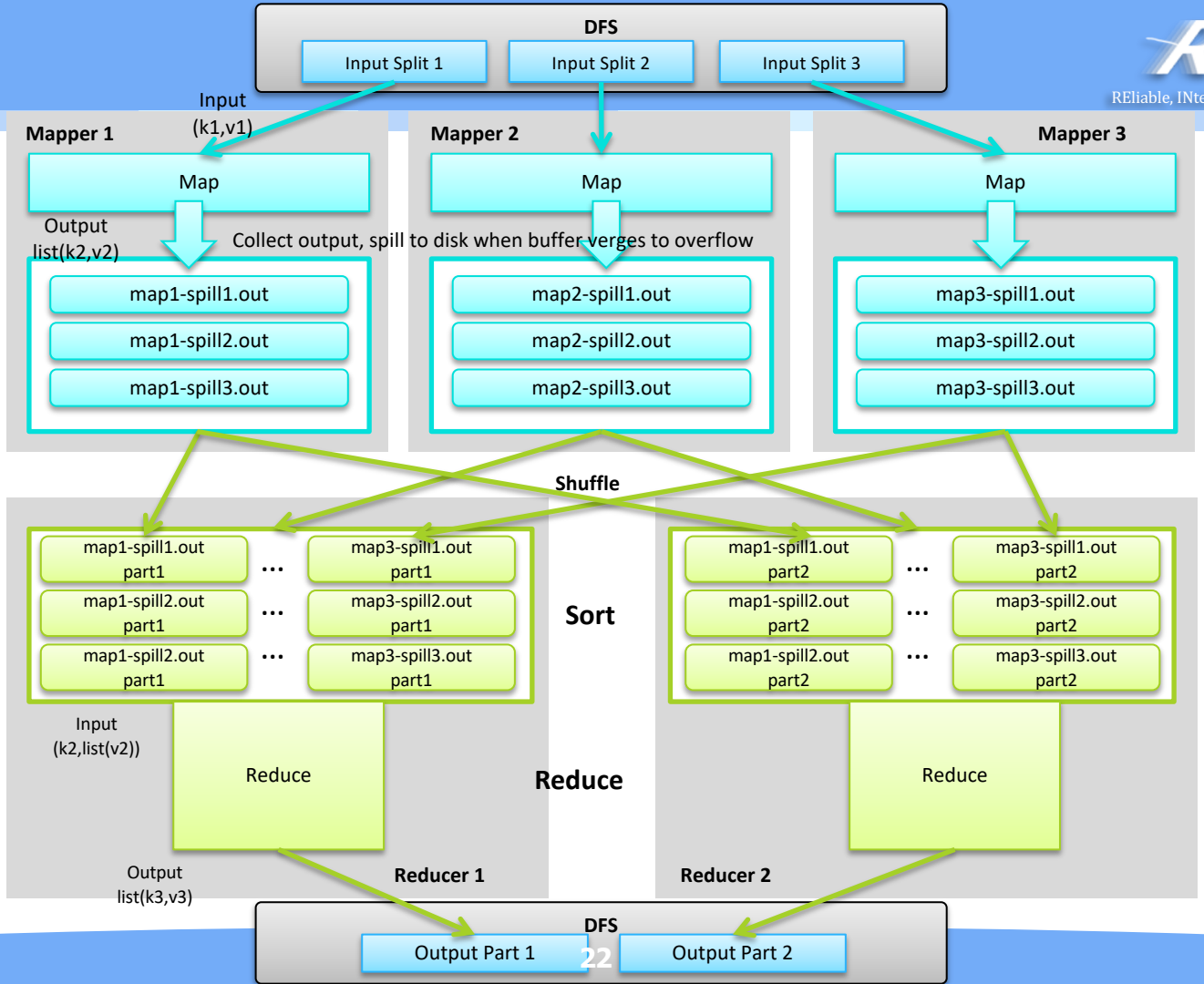
- Paradigm
  - map $(k1, v1)$ $\rightarrow list(k2, v2)$
  - $reduce$ $(k2, list(v2))$ $\rightarrow list(k3, v3)$
- Word Count



- Execution View




DFS
Input Split 1
Input Split 2
Input Split 3
Input
(k1,v1)
Mapper 1
Map
Output
list(k2,v2)
Collect output, spill to disk when buffer verges to overflow
map1-spill1.out
map1-spill2.out
map1-spill3.out
Merge spill files into a single output file
Part1
Part2
Mapper 2
Map
map2-spill1.out
map2-spill2.out
map2-spill3.out
Part1
Part2
Mapper 3
Map
map3-spill1.out
map3-spill2.out
map3-spill3.out
Part1
Part2
Shuffle
Map1
Part1
Map2
Part1
Map3
Part1
Sort
Map1
Part2
Map2
Part2
Map3
Part2
Input
(k2,list(v2))
Reduce
Reduce
Reduce
Output
list(k3,v3)
Reducer 1
Reducer 2
DFS
Output Part 1
Output Part 2


DFS
Input Split 1
Input Split 2
Input Split 3
Input
(k1,v1)
Mapper 1
Map
Output
list(k2,v2)
Collect output, spill to disk when buffer verges to overflow
map1-spill1.out
map1-spill2.out
map1-spill3.out
Mapper 2
Map
map2-spill1.out
map2-spill2.out
map2-spill3.out
Mapper 3
Map
map3-spill1.out
map3-spill2.out
map3-spill3.out
Shuffle
map1-spill1.out part1
map3-spill1.out part1
map1-spill2.out part1
map3-spill2.out part1
map1-spill2.out part1
map3-spill3.out part1
Sort
map1-spill1.out part2
map3-spill1.out part2
map1-spill2.out part2
map3-spill2.out part2
map1-spill2.out part2
map3-spill3.out part2
Input
(k2,list(v2))
Reduce
Reduce
Reduce
Output
list(k3,v3)
Reducer 1
Reducer 2
DFS
Output Part 1
Output Part 2

Google Percolator

- Daniel Peng and Frank Dabek
  - OSDI 2010

Google

- Percolator is a system for incrementally processing updates to a large data set, and deployed it to create the Google web search index.

- Problem
  - Existing DBMSs can't handle the sheer volume of data: Google's indexing system stores tens of petabytes across thousands of machines
  - Given that the system will be processing many small updates concurrently, and ideal system would also provide mechanisms for maintaining invariants despite concurrent updates and for keeping track of which updates have been processed.

Google Percolator

- Percolator
  - Provides the user with random access to a multi-PB repository.
  - Random access allows us to process documents individually, avoiding the global scans of the repository that MapReduce requires.
  - To achieve high throughput, many threads on many machines need to transform the repository concurrently, so Percolator provides ACID-compliant transactions to make it easier for programmers to reason about the state of the repository.
  - Percolator provides observers: pieces of code that are invoked by the system whenever a user-specified column changes.

Google Percolator

- Design
  - ACID transaction over a random-access repository
  - Observers, a way to organize an incremental computation

- A Percolator system consists of three binaries that run on every machine in the cluster:
  - A Percolator worker
  - A Bigtable tablet server
  - and a GFS chunk server
  - All observers are linked into the Percolator worker

- The system also depends on two small services:
  - the timestamp oracle
  - and the lightweight lock service.

- Two phases of commit

Column	*Use*
c:lock	An uncommitted transaction is writing this cell; contains the location of primary lock
c:write	Committed data present; stores the Bigtable timestamp of the data
c:data	Stores the data itself
c:notify	Hint: observers may need to run
c:ack_O	Observer "O" has run ; stores start timestamp of successful last run

Google Percolator

- Two phases of commit

key	*bal:data*	*bal:lock*	*bal:write*
Bob	*6:* *5:* $10	*6:* *5:*	*6:* data @ 5 *5:*
Joe	*6:* *5:* $2	*6:* *5:*	*6:* data @ 5 *5:*

1. Initial state: Joe's account contains $2 dollars, Bob's $10.

Google Percolator

- Two phases of commit

Bob	*7:* **$3** *6:* *5:* $10	*7:* **I am primary** *6:* *5:*	*7:* *6:* data @ 5 *5:*
Joe	*6:* *5:* $2	*6:* *5:*	*6:* data @ 5 *5:*

2. The transfer transaction begins by locking Bob's account balance by writing the lock column. This lock is the primary for the transaction. The transaction also writes data at its start timestamp, 7.

Google Percolator

- Two phases of commit

Bob	*7:* $3 *6:* *5:* $10	*7:* I am primary *6:* *5:*	*7:* *6:* data @ 5 *5:*
Joe	***7:* $9** *6:* *5:* $2	***7:* primary @ Bob.bal** *6:* *5:*	*7:* *6:* data @ 5 *5:*

3. The transaction now locks Joe's account and writes Joe's new balance (again, at the start timestamp). The lock is a secondary for the transaction and contains a reference to the primary lock (stored in row "Bob," column "bal"); in case this lock is stranded due to a crash, a transaction that wishes to clean up the lock needs the location of the primary to synchronize the cleanup.

Google Percolator

- Two phases of commit

Bob	*8:* *7:* $3 *6:* *5:* $10	*8:* ***7:*** *6:* *5:*	*8:* **data @ 7** *7:* *6:* data @ 5 *5:*
Joe	*7:* $9 *6:* *5:* $2	*7:* primary @ Bob.bal *6:* *5:*	*7:* *6:*data @ 5 *5:*

4. The transaction has now reached the commit point: it erases the primary lock and replaces it with a write record at a new timestamp (called the commit timestamp): 8. The write record contains a pointer to the timestamp where the data is stored. Future readers of the column "bal" in row "Bob" will now see the value $3.

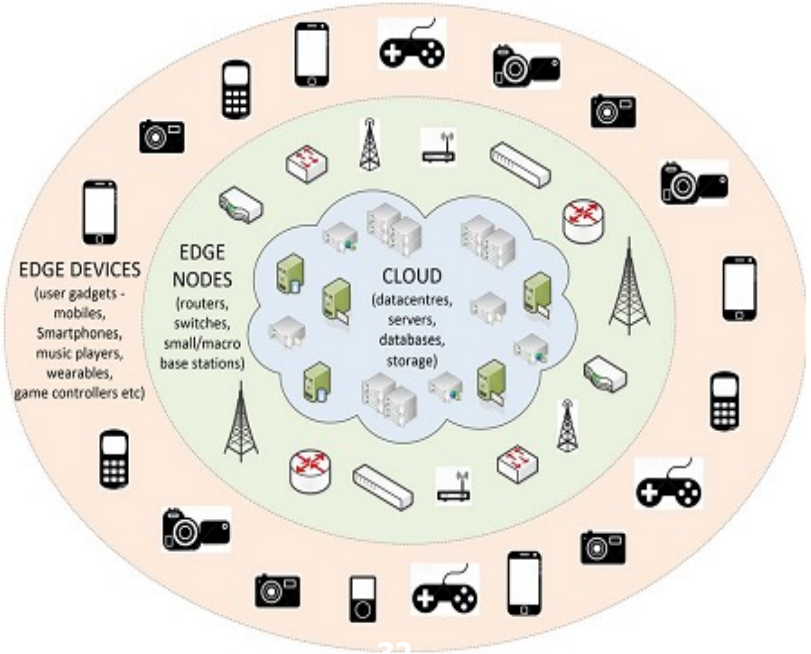
- Two phases of commit

Bob	*8:* *7:* $3 *6:* *5:* $10	*8:* *7:* *6:* *5:*	*8:* data @ 7 *7:* *6:* data @ 5 *5:*
Joe	*8:* *7:* $9 *6:* *5:*$2	*8:* ***7:*** *6:* *5:*	***8:* data @ 7** *7:* *6:* data @ 5 *5:*

5. The transaction completes by adding write records and deleting locks at the secondary cells. In this case, there is only one secondary: Joe.

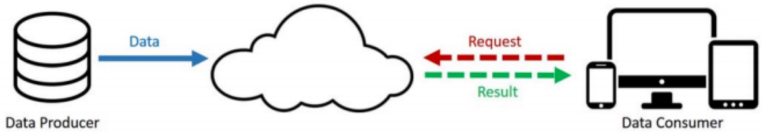
What is Edge Computing ?

- **Edge computing**
  - is a method of optimizing cloud computing systems by performing data processing at the edge of the network, near the source of the data.

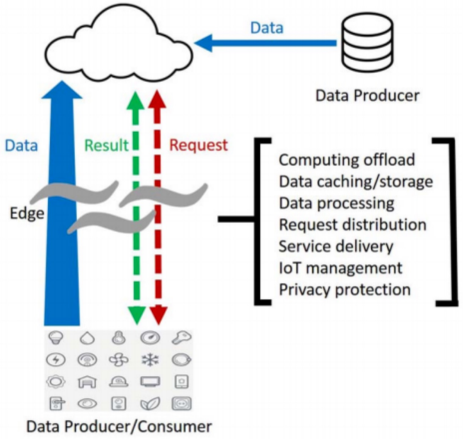


What is Edge Computing ?

- **Cloud computing**

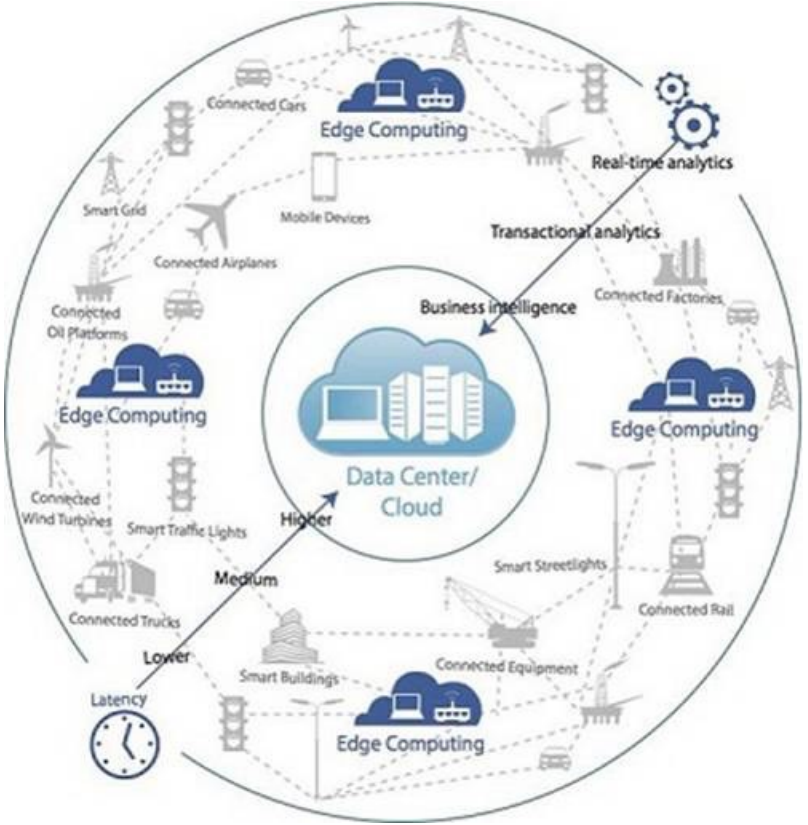


- **Edge computing**



What is Edge Computing ?

- Edge Computing
  - This reduces the communications bandwidth needed between sensors and the central data center by performing analytics and knowledge generation at or near the source of the data.
  - This approach requires leveraging resources that may not be continuously connected to a network such as laptops, smartphones, tablets and sensors



Application Scenario

- Cloud offloading
  - In the cloud computing paradigm, most of the computations happen in the cloud, which means data and requests are processed in the centralized cloud.
  - In the edge computing paradigm, not only data but also operations applied on the data should be cached at the edge.
  - Challenges:
    - The data at the edge node should be synchronized with the cloud
    - Another issue involves the collaboration of multiple edges when a user moves from one edge node to another

Application Scenario

- Video Analytics
  - Cloud computing is no longer suitable for applications that requires video analytics due to the long data transmission latency and privacy concerns.
  - With the edge computing paradigm, the request can be generated from the cloud and pushed to all the things in a target area. Each thing, for example, a smart phone, can perform the request and search its local camera data and only report the result back to the cloud.
  - In this paradigm, it is possible to leverage the data and computing power on every thing and get the result much faster compared with solitary cloud computing.

Application Scenario

- Smart Home
  - Smart home would report an impressive amount of data and for the consideration of data transportation pressure and privacy protection, this data should be mostly consumed in the home.
  - This feature makes the cloud computing paradigm unsuitable for a smart home.
  - Edge computing is considered perfect for building a smart home:
    - with an edge gateway running a specialized edge operating system (edgeOS) in the home,
    - the things can be connected and managed easily in the home,
    - the data can be processed locally to release the burdens for Internet bandwidth,
    - and the service can also be deployed on the edgeOS for better management and delivery.

Cloud-terminal fusion computing

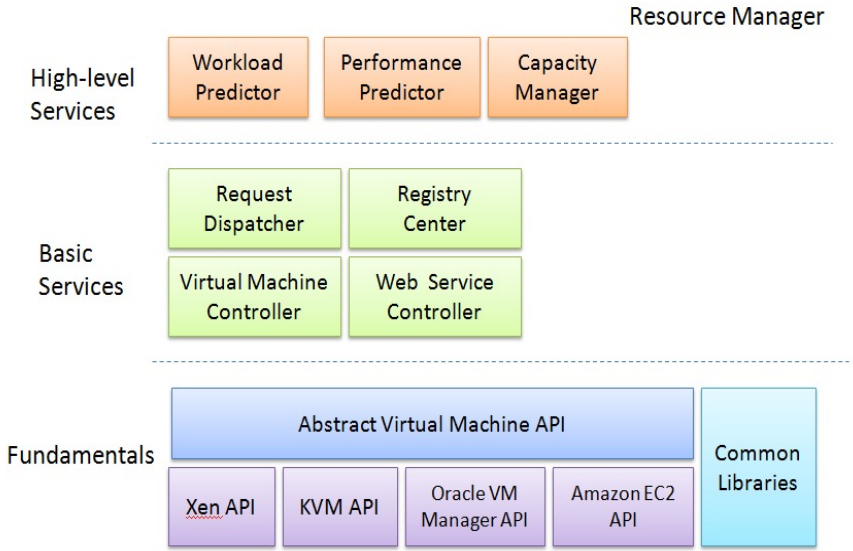
- Most of terminals have powerful computing abilities
  - Such as mobile phone, car computer, game console and so on.
  - All of these terminals connect with data centers which provide data computing, storage and complex task processing service.

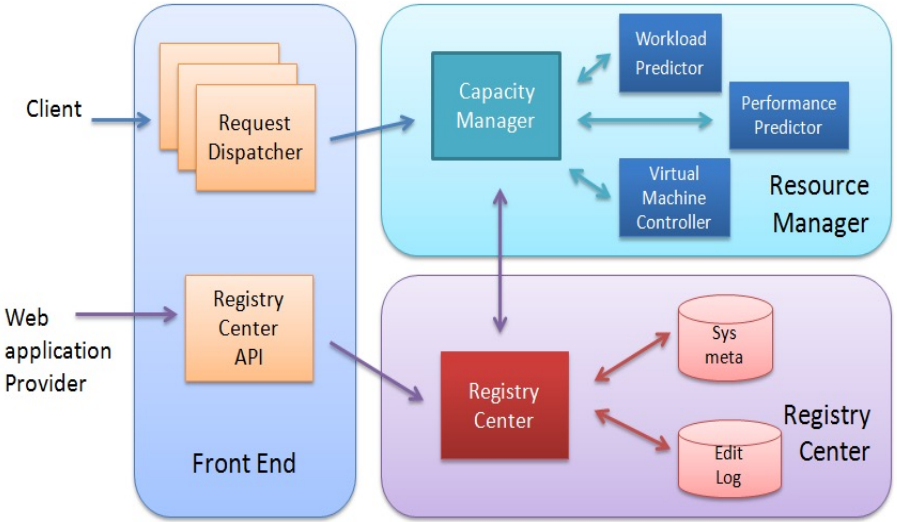
- However, the size of data center grows bigger following the increment of terminals.
  - How to reduce the computing pressure of data center and introduce the computing ability of terminal into computing processing is a difficult question.
  - Cloud-Terminal Fusion is a trend of future computing architecture which can solve the question mentioned before. In this architecture, one computing task can be handled by terminal and data center together and the total processing time will be reduced.

Cloud Computing

REliable, INtelligent & Scalable Systems

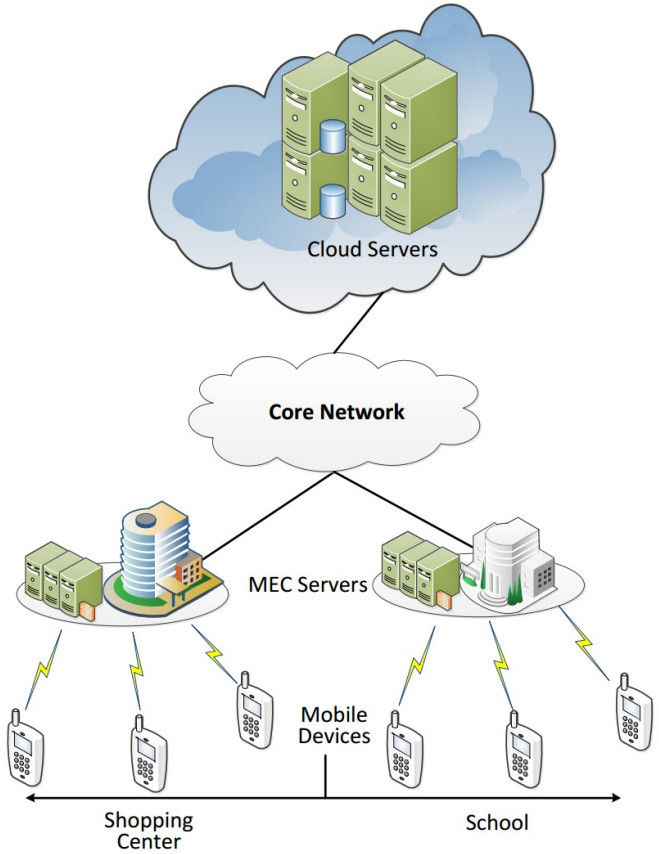
- Computation in cloud
  - Resource Management
  - Service Management
  - Workload Optimization





Edge Computing

- The Rise of Cloud-Terminal Fusion Computing
  - Cloud Servers
  - Mobile Edge Computing Servers
  - Mobile Devices (Edge Devices)



Challenges

- Programmability
  - In the edge computing, computation is offloaded from the cloud, and the edge nodes are most likely heterogeneous platforms.
- Naming
  - In edge computing, one important assumption is that the number of things is tremendously large.
- Data Abstraction
  - In edge computing, data abstraction becomes more challenging. With IoT, there would be a huge number of data generators in the network.

Challenges

- Service Management
  - There are four fundamental features should be supported to guarantee a reliable system, including differentiation, extensibility, isolation, and reliability
- Privacy and Security
  - First is the awareness of privacy and security to the community.
  - Second is the ownership of the data collected from things at edge.
  - Third is the missing of efficient tools to protect data privacy and security at the edge of the network.

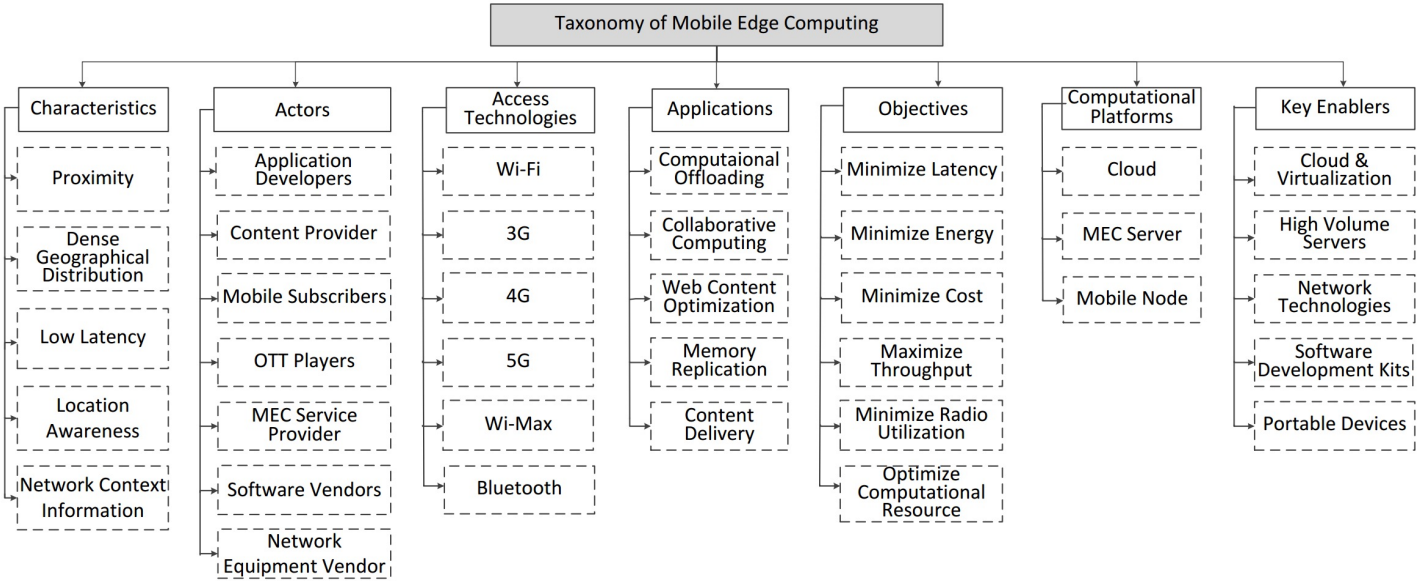
Optimization Metrics

- Latency
  - Latency is one of the most important metrics to evaluate the performance, especially in interaction applications/services.
  - To reduce the latency, the workload should better be finished in the nearest layer which has enough computation capability to the things at the edge of the network.
- Bandwidth
  - High bandwidth can reduce transmission time, especially for large data.
  - we need to evaluate if a high bandwidth connection is needed and which speed is suitable for an edge.
  - Besides, to correctly determine the workload allocation in each layer, we need to consider the computation capability and bandwidth usage information in layers to avoid competition and delay.

Optimization Metrics

- Energy
  - Battery is the most precious resource for things at the edge of the network.
  - The key is the tradeoff between the computation energy consumption and transmission energy consumption.
- Cost
  - From the service providers' perspective, edge computing provides them less latency and energy consumption, potential increased throughput and improved user experience.
  - As a result, they can earn more money for handling the same unit of workload.

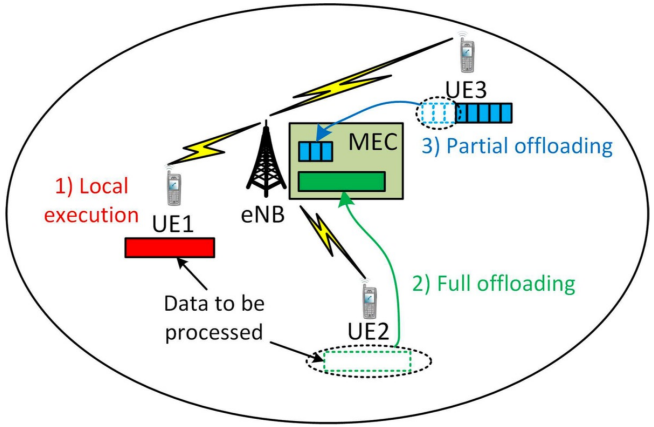
Taxonomy of Mobile Edge Computing



Computation Offloading

- From the user perspective,
  - a critical use case regarding the MEC is a computation offloading as this can save energy and/or speed up the process of computation.

- Local Execution
- Full offloading
- Partial offloading

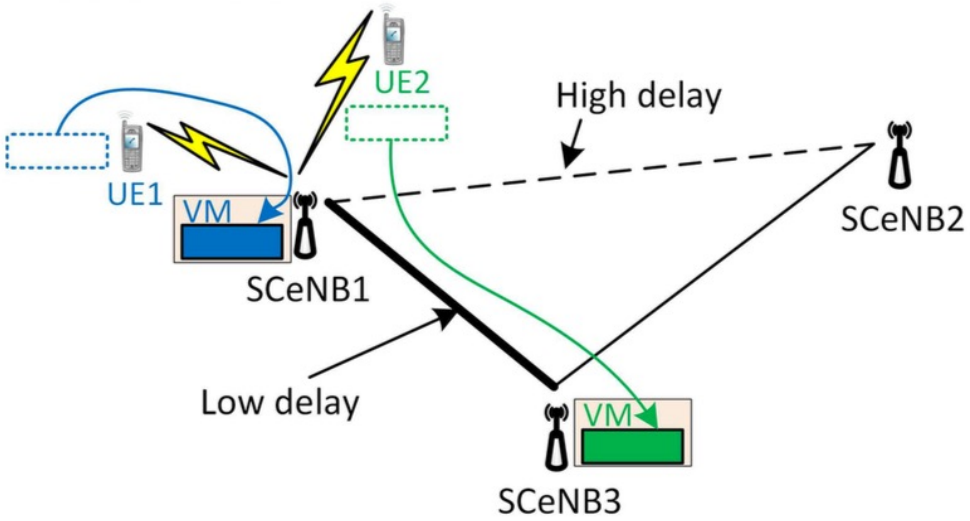


Computation Offloading

- Full offloading vs. Partial offloading
  - Minimization of execution delay
  - Minimization of energy consumption while satisfying execution delay constraint
  - Trade-off between energy consumption and execution delay

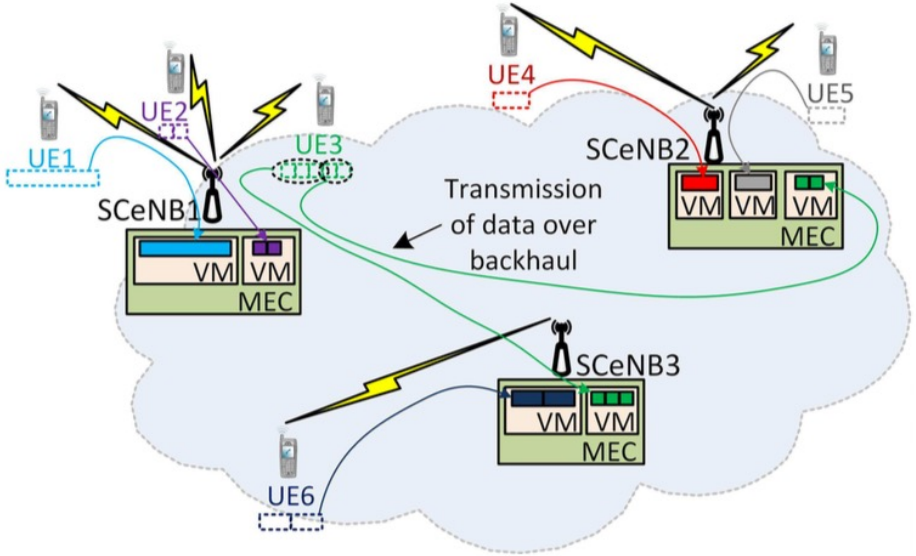
Allocation of Computing Resource

- Allocation of computation resources at a single node



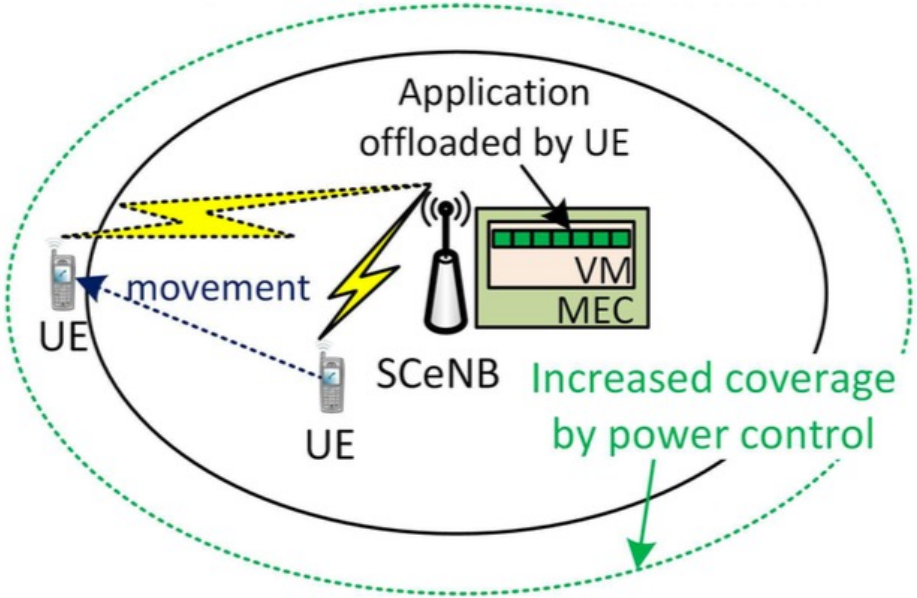
Allocation of Computing Resource

- Allocation of computation resources at multiple nodes (federated clouds)



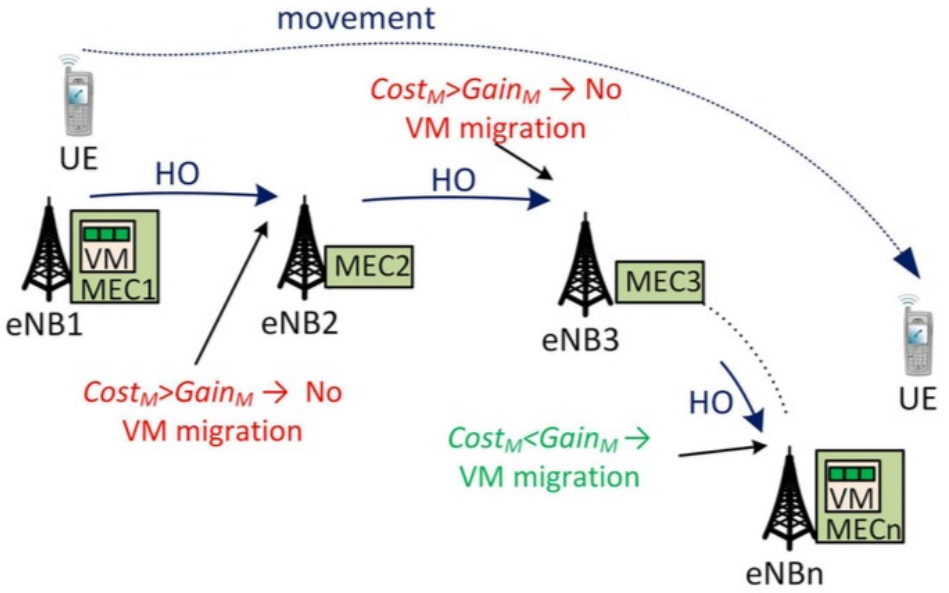
Mobility Management

- Power control

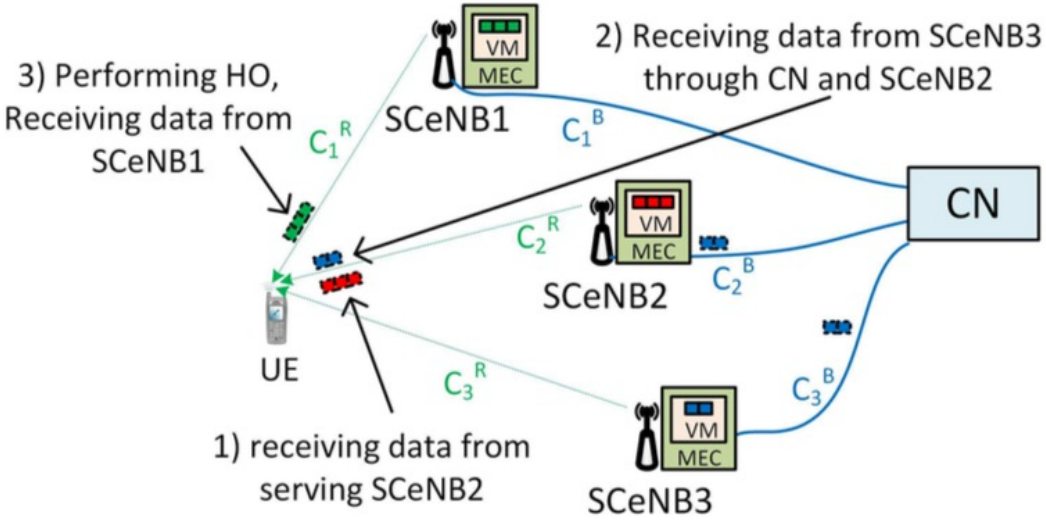


Mobility Management

- VM migration



- Path selection and/or VM migration



References

- Cloud Computing and Grid Computing 360-Degree Compared
  - Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu
  - IEEE Grid Computing Environment 2008
- Above the Clouds: A Berkeley View of Cloud Computing
  - Michael Armbrust, Armando Fox, et.al,
  - http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf, 2009
- Introduction to Cloud Computing
  - Doug Terry: Chairman, ACM Tech Pack Committee on Cloud Computing
  - http://techpack.acm.org/cloud/
- Cloud Computing
  - http://searchcloudcomputing.techtarget.com/definition/cloud-computing

References

- MapReduce: Simplied Data Processing on Large Clusters
  - Jeffrey Dean and Sanjay Ghemawat
  - OSDI 2004
- The Google File System
  - Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
  - SOSP 2003
- Bigtable: A Distributed Storage System for Structured Data
  - Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
  - OSDI 2006
- Apache Hadoop
  - http://hadoop.apache.org/#What+Is+Hadoop%3F
- Xen
  - http://www.xen.org/products/xenhyp.html
- Ubuntu Enterprise Cloud
  - http://www.ubuntu.org.cn/products/whatisubuntu/serveredition/cloud/uec/

References

- Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li and Lanyu Xu, “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal, June 2016.
- Pavel Mach, Zdenek Becvar , “Mobile Edge Computing: A Survey on Architecture and Computation Offloading”, IEEE Communications Surveys & Tutorials, Auguest 2017.
- M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, “The Case for VM-Based Cloudlets in Mobile Computing,” in IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009.
- S. Cao, X. Tao, Y. Hou and Q. Cui, “An energy-optimal offloading algorithm of mobile computing based on HetNets,” 2015 International Conference on Connected Vehicles and Expo (ICCVE), Shenzhen, 2015, pp. 254-258.
- Maofei Deng, Hui Tian and Bo Fan, “Fine-granularity based application offloading policy in cloud-enhanced small cell networks,” 2016 IEEE International Conference on Communications Workshops (ICC), Kuala Lumpur, 2016, pp. 638-643.
- S.Ou, K.Yang, J.Zhang,“An effective offloading middleware for pervasive services on mobile devices,”Pervasive Mobile Comput, vol.3, no.4, pp.362-385, 2007.
- M. Kamoun, W. Labidi, and M. Sarkiss, “Joint resource allocation and offloading strategies in cloud enabled cellular networks”, IEEE International Conference on Communications (ICC), 5529-34, 2015.
- Kwon YongChul, Balazinska Magdalena, Howe Bill, Rolia Jerry, “Mitigating skew in MapReduce applications”, Proceedings of the VLDB Endowment, Auguest 2012.

Thank You!