Cloud Computing & Big Data

PARALLEL & SCALABLE MACHINE LEARNING & DEEP LEARNING

Prof. Dr. – Ing. Morris Riedel
Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

LECTURE 5

@Morris Riedel @MorrisRiedel @MorrisRiedel

Map-Reduce Computing Paradigm

October 8, 2020
Online Lecture

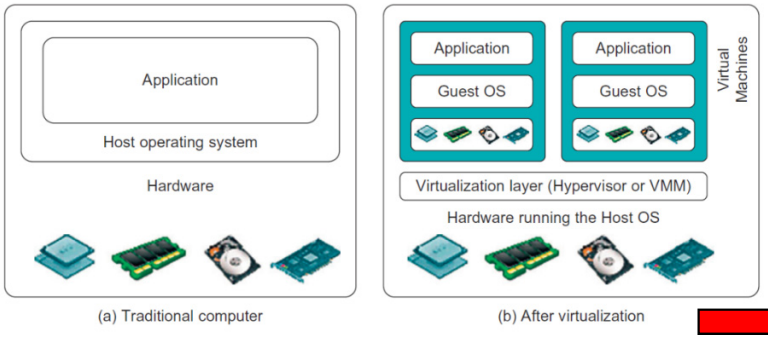
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

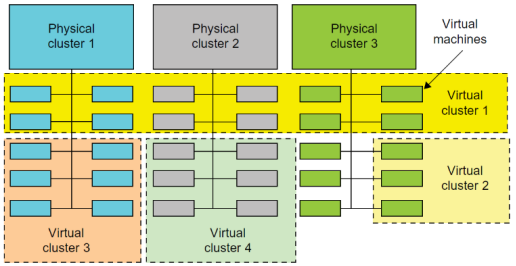
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

DEEP Projects

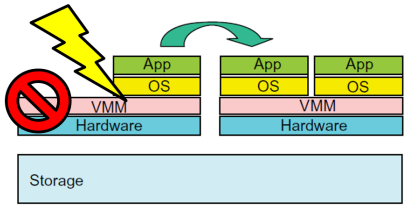
HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 4 – Virtualization & Data Center Design

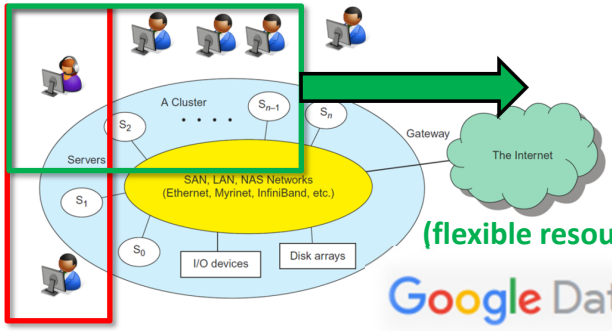




(virtualization drives data center design & enables scalability through 'economics of scale')

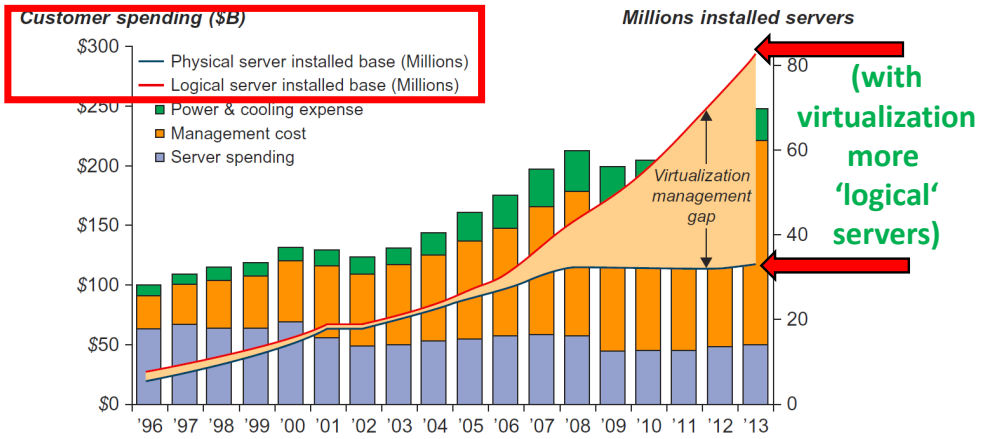


(benefits for data centers, e.g. life migration in case of failures)



(flexible resource sharing)

Google Data Centers



(with virtualization more 'logical' servers)

[2] Apache Spark *[10] Google Data Centers, wired.com* *[3] Distributed & Cloud Computing Book*

Outline of the Course

1. Cloud Computing & Big Data Introduction
2. Machine Learning Models in Clouds
3. Apache Spark for Cloud Applications
4. Virtualization & Data Center Design
5. Map-Reduce Computing Paradigm
6. Deep Learning driven by Big Data
7. Deep Learning Applications in Clouds
8. Infrastructure-As-A-Service (IAAS)
9. Platform-As-A-Service (PAAS)
10. Software-As-A-Service (SAAS)
11. Big Data Analytics & Cloud Data Mining
12. Docker & Container Management
13. OpenStack Cloud Operating System
14. Online Social Networking & Graph Databases
15. Big Data Streaming Tools & Applications
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

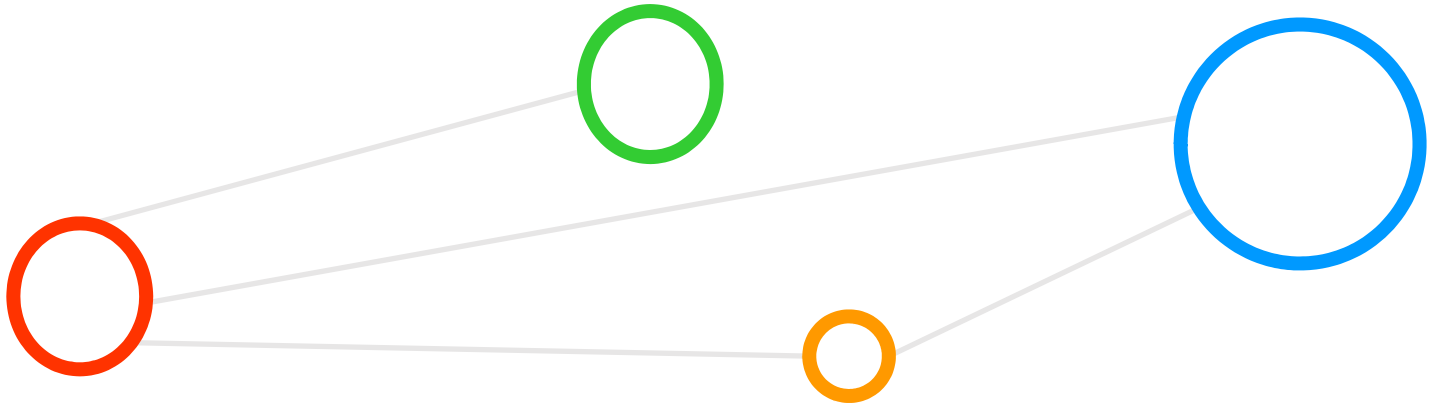
- Practical Topics
- Theoretical / Conceptual Topics

Outline

- Map Reduce Approach
  - Big Data Analytics Requirements & Map-Reduce via Apache Hadoop
  - Motivation for Hiding Computing Complexity & Increase Reliability
  - Divide and Conquer Paradigm & Three ‘phases’ of Map-Reduce
  - Understanding Key-Value Data Structures for Map-Reduce
  - Open Source Apache Hadoop & HBase in Facebook Applications
- Map-Reduce Ecosystem in Clouds & Applications
  - Role of Reliability using Statistics & Hadoop Distributed File System
  - Replication Concept & Benefits for Cloud ‘Big Data’ Applications
  - Amazon Web Services offering Elastic MapReduce & Ecosystem Tools
  - Using Containers with Docker & Cloud Deployment using Kubernetes
  - OpenStack Cloud Operating System & Map-Reduce Limits

- **Promises from previous lecture(s):**
- ***Lecture 1:*** **Lecture 5 provides more details on how the map-reduce paradigm works and how it enables a variety of cloud applications today**
- ***Lecture 1:*** **Lecture 5 provides more details on how the map-reduce paradigm works and how it is implemented in the Apache Hadoop software**
- ***Lecture 3:*** **Lecture 5 provides more details on the Hadoop Distributed File System (HDFS) that is often used in conjunction with Apache Spark**
- ***Lecture 4:*** **Lecture 5 provides insights into Hadoop Distributed File System (HDFS) that assumes failure by default and fits to data center practice**

Map Reduce Approach



Big Data Analytics vs. Data Analysis – Revisited (cf. Lecture 1)

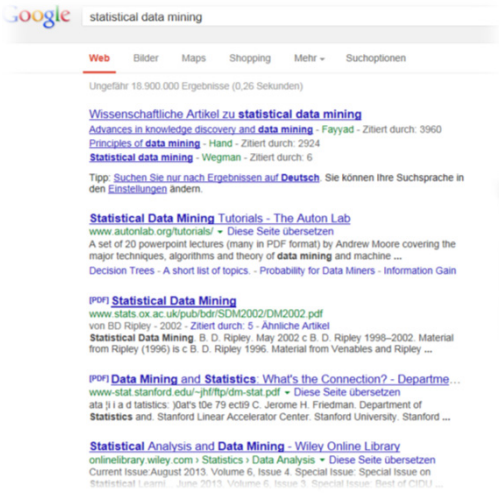
- Data Analysis supports the search for ‘causality’
  - Describing exactly WHY something is happening
  - Understanding causality is hard and time-consuming
  - Searching it often leads us down the wrong paths
  - Focus on the findings in the data and not on enabling infrastructure

- Big Data Analytics focussed on ‘correlation’
  - Not focussed on causality – enough THAT it is happening
  - Discover novel patterns and WHAT is happening more quickly
  - Using correlations for invaluable insights – often data speaks for itself
  - Often includes technologies and enabling infrastructure (e.g., scalability)

- **‘Big Data Analytics‘ are powerful techniques to work on large data including the enabling infrastructure to work with ‘big data‘ using Clouds**
- **Data Analysis is the in-depth interpretation of research data that is often part of a concrete ‘Big Data Analytics‘ approach**

Motivating Example: Machine Learning & Data Mining Applications

- Selected Properties
  - Opportunity to exploit 'nice parallelism'
  - Require handling of immense amounts of 'big data' quickly
  - Provide data that is extremely regular and can be independently processed
- Examples from the Web
  - Ranking of Web pages by importance (includes iterated matrix-vector multiplication)
  - Searches in online social networking sites (includes search in graph with hundreds of nodes/billions of edges)



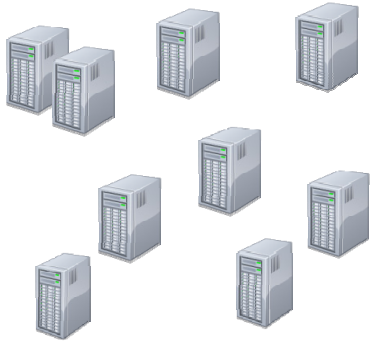
- **Many modern data mining applications require computing on compute nodes (i.e. processors/cores) that operate independently from each other (i.e. HTC)**
- **Independent means there is little or even no communication between tasks**

➢ **Lecture 11 will provide more details on data analytics techniques using parallel computing for data mining applications in Clouds today**

Requirement for a 'Data Processing Machinery' to perform Big Data Analytics

- Specialized Parallel Computers (aka Supercomputers)
  - Interconnent between nodes is expensive (i.e. Infiniband/Myrinet)
  - Interconnect not always needed in data analysis (independence in datasets)
  - Programming is relatively difficult/specific, parallel programming is profession of its own

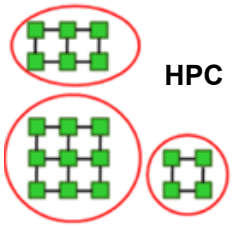
- Large 'compute clusters' dominate data-intensive computing
  - Offer 'cheaper' parallelism (no expensive interconnect & switches between nodes)
  - Compute nodes (i.e. processors/cores) interconnected with usual ethernet cables
  - Provide large collections of commodity hardware (e.g. normal processors/cores)



➢ **Complementary High Performance Computing (HPC) course offers insights to use supercomputers for data mining & machine learning**

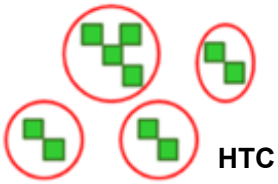
High Performance Computing (HPC) vs. High Throughput Computing (HTC)

- **High Performance Computing (HPC) is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance cpu/core interconnections.**

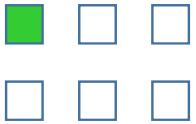


(network connection
very important & costly)

- **High Throughput Computing (HTC) is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of 'farming jobs' without providing a high performance interconnection between the cpu/cores.**



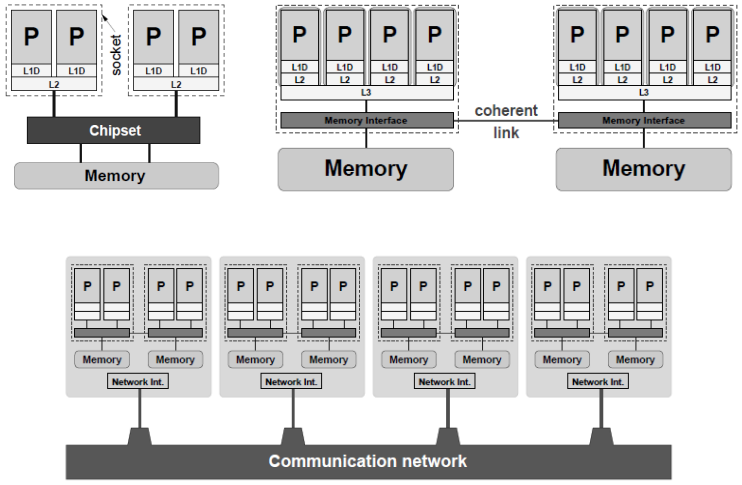
(network connection
less important)

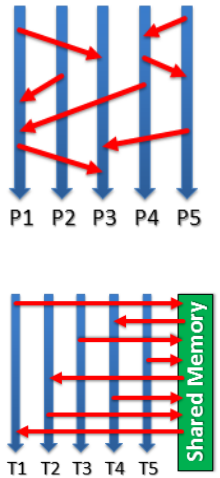


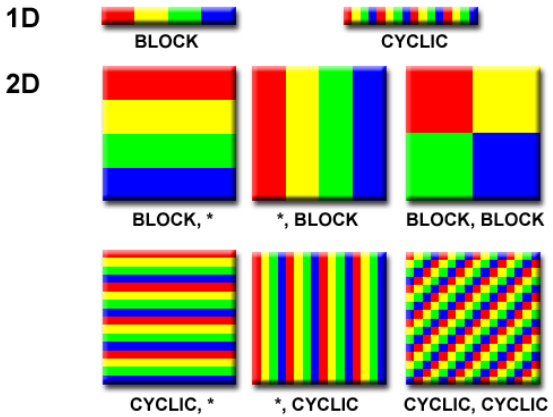
(primarily distributed
processing)

Map-Reduce Motivation: Avoid Increased HPC Complexity & Keep it Simple

- Complexities
  - Concurrency & Computation
  - Interconnects
  - Parallel Programming
  - Domain Decomposition
  - Quick Technology Changes

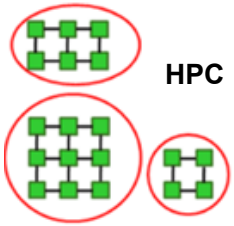






[5] Introduction to High Performance Computing for Scientists and Engineers

[6] Parallel Computing Tutorial

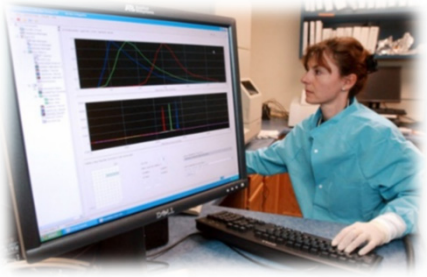


- **Different HPC Programming elements (barriers, mutexes, shared-/distributed memory, MPI, etc.)**
- **Task distribution issues (scheduling, synchronization, inter-process-communication, etc.)**
- **Complex heterogenous architectures (UMA, NUMA, hybrid, various network topologies, etc.)**
- **Data/Functional parallelism approaches (SPMD, MPMD, domain decomposition, ghosts/halo, etc.)**

➢ **Complementary High Performance Computing (HPC) course offers parallel programming with UMA, NUMA, SPMD, MPMD, etc.**

Key Approach: Hiding the Complexity of Underlying Computing Technologies

- Many users of parallel data processing machines are not technical savvy users and don't need to know system details
  - Scientific domain-scientists (e.g. biology) need to focus on their science & data
  - Data Scientists from statistics/machine-learning need to focus on their algorithms & data
- Non-technical users raise the following requirements for a 'data processing machinery':
  - KISS: Keep it simple & stupid ('but not too simple')



[7] Science Progress

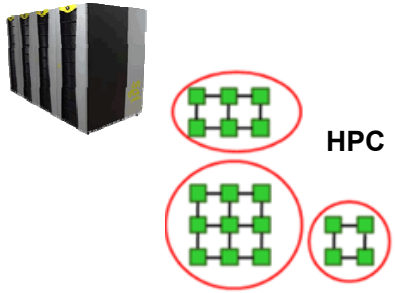
(scientific & engineering community end-users are extremely frustrated when they are exposed to technical errors & have to do tasks again)

- **The 'data processing machinery' needs to be easy to program**
- **The machinery must hide the complexities of computing (e.g. different networks, various operating systems, etc.)**
- **It needs to take care of the complexities of parallelization (e.g. scheduling, task distribution, synchronization, etc.)**

Map-Reduce Motivation: Need for Reliability & Handle Failures during Operation

- Reasons for failures (cf. Lecture 4)
  - Loss of a single node within a rack (e.g. hard disk crash, memory errors)
  - Loss of an entire rack (e.g. network issues)
  - Operating software ‘errors/bugs’
- Consequences of failures
  - Long-running compute tasks (e.g. hours/days) need to be restarted
  - Already written data may become inconsistent and needs to be removed
  - Access to unique datasets maybe hindered or even unavailable in one particular filesystem
- Example: Juelich Supercomputing Centre
  - Failures happen: using the machine needs to go on!

[10] Google Data Centers, wired.com



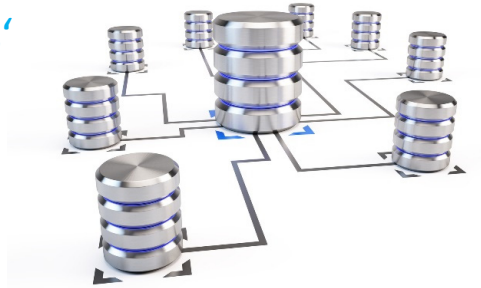
[8] Supercomputer JUWELS Cluster Module Configuration

- 2271 standard compute nodes - Dual Intel Xeon Platinum 8168
  - 2x 24 cores, 2.7 GHz
  - 12x 8 GB, 2666 MHz
  - EDR-Infiniband (Connect-X4)
  - Intel Hyperthreading Technology (Simultaneous Multithreading)
  - diskless
- 240 large memory compute nodes - Dual Intel Xeon Platinum 8168
  - 2x 24 cores, 2.7 GHz
  - 12x 16 GB, 2666 MHz
  - EDR-Infiniband (Connect-X4)
  - Intel Hyperthreading Technology (Simultaneous Multithreading)
  - diskless
- 56 accelerated compute nodes - Dual Intel Xeon Gold 6148
  - 2x 20 cores, 2.4 GHz
  - 12x 16 GB, 2666MHz
  - Dual EDR-Infiniband (Connect-X4)
  - Intel Hyperthreading Technology (Simultaneous Multithreading)
  - diskless
  - 4x Nvidia V100 GPU

Motivation: Role of Scalability & Wide-Area Networks – Revisited (cf. Lecture 1)

- High-bandwidth networking
  - Increases the capability of building massively distributed systems
  - Enables elastic computing and scalability beyond one server or one data center
  - Interconnected cloud data centers & servers raise demands for 'perfect networks'
- 'Data locality' as major Cloud & 'Big Data Analytics' movement
  - Requirements for scalable programming models and tools
  - CPU speed has surpassed IO capabilities of existing cloud resources
  - Data-intensive clouds with advanced analytics and analysis capabilities
  - Considering moving compute task to data vs. moving data to compute
  - E.g., map-reduce paradigm had a major impact in cloud adoptions

(does it make sense to transfer TBs/PBs again and again for applications?)



Google

[4] MapReduce: Simplified Dataset on Large Clusters, 2004

[3] Distributed & Cloud Computing Book

- **We observe tremendous price/performance ratio of commodity hardware that is driven by the desktop, notebook, and tablet computing markets today**
- **Price/performance ratio has also driven the adoption and use of commodity hardware in large-scale and distributed computing including high-bandwidth network increases**

Big Data Analytics Frameworks – Simple, Reliable & Hiding Complexity

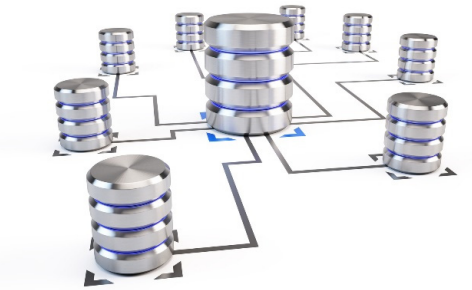
- Distributed Processing
  - ‘Map-reduce via files‘: Tackle large problems with many small tasks
  - Advantage of ‘data replication‘ via specialized distributed file system
  - E.g. Apache Hadoop

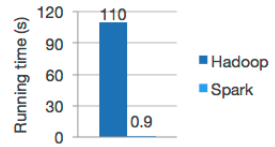
[1] Apache Hadoop

- In-Memory Processing
  - Perform many operations fast via ‘in-memory‘
  - Enable tasks such as ‘map-reduce in-memory‘
  - Needs hardware systems that offer large memory
  - E.g. Apache Spark, Apache Flink

[2] Apache Spark

(does it make sense to transfer TBs/PBs again and again for applications?)



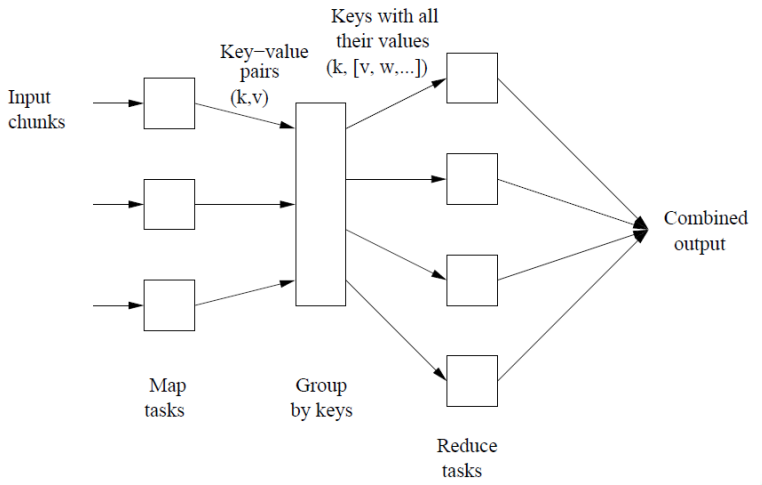


Logistic regression in Hadoop and Spark

- **Big Data analytics frameworks shift the approach from ‘bring data to compute resources‘ into ‘bring compute tasks close to data‘ including infrastructure technologies (e.g., Apache Hadoop and/or Apache Spark)**

Origins of the Map-Reduce Programming Model

- Origin: Invented via the proprietary Google technology by Google technologists
  - Drivers: Applications and 'data mining approaches' around the Web
  - Foundations go back to functional programming (e.g. LISP)
- Large 'open source community'
  - Apache Hadoop, versions 1/2/3
  - Open Source Implementation of the 'map-reduce' programming model
  - Based on Java programming language, e.g. map & reduce tasks objects
  - Offers a scheduler for distributed computing
  - Broadly used – also by commercial vendors within added-value software
  - Foundation for many higher-level algorithms, frameworks, and approaches



Google

[4] MapReduce: Simplified Dataset on Large Clusters, 2004

hadoop

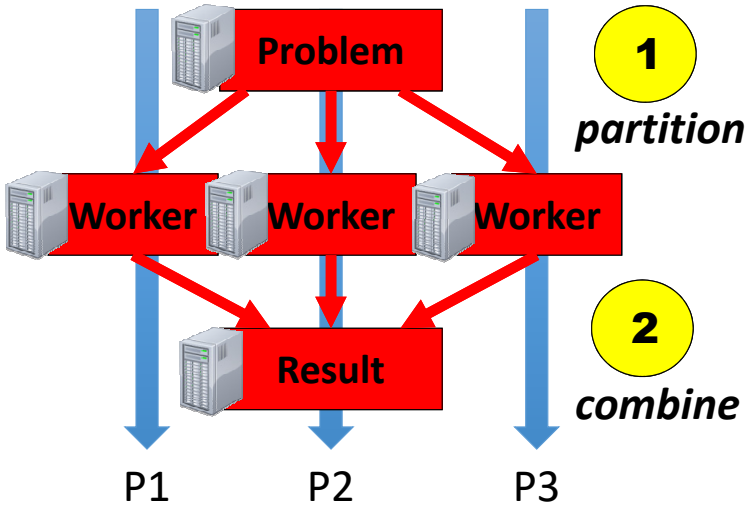
[1] Apache Hadoop

➢ **Practical Lecture 5.1 will offer some concrete examples of map-reduce applications in modern Cloud computing environments today**

Map-Reduce Approch derived from Divide & Conquer Computing

- Map-Reduce Computing Paradigm
  - Break 'big tasks' in many sub-tasks and aggregate/combine results
  - E.g. word counts in large text document collections, movie ratings, etc. to enable recommendations

Divide & Conquer

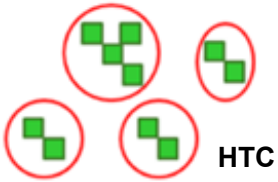


(e.g. all ratings of movies)

- **The Divide & Conquer Computing paradigm consists of: (1) Partition the whole problem space and (2) Combine the partly solutions of each partition to a whole solution of the problem**
- **The Divide & Conquer approach is especially useful for nicely parallel applications that require independent computing tasks**

(e.g. work to compute similiarity of users & analyze ratings)

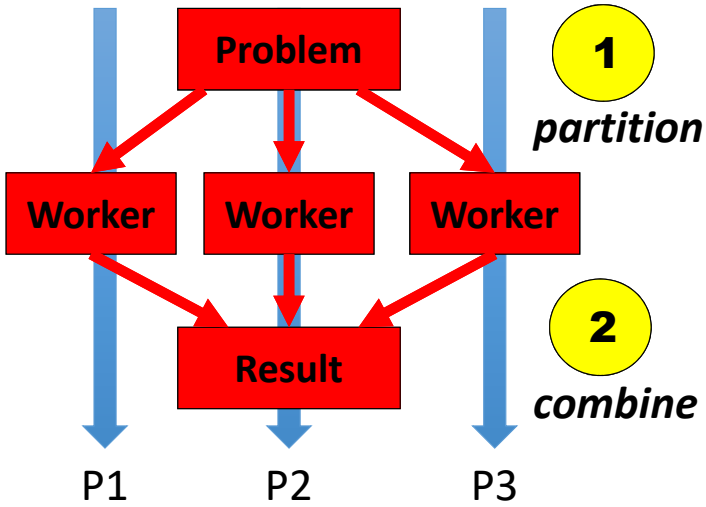
(e.g. combine recommendations for a user)



Map-Reduce Computing Paradigm & Open Source Implementation

- Idea not completely new
  - Derived divide & conquer strategy
  - Needs some smart addition (e.g., grouping intermediate results)

Divide & Conquer



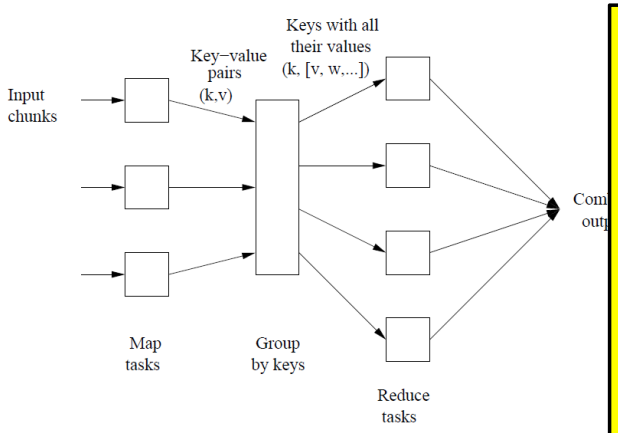
- Map-Reduce Paradigm
  - Follows divide & conquer approach
  - Injects a key element between this process: short/shuffle/group
  - Open Source Implementation: Apache Hadoop

Google

[4] MapReduce: Simplified Dataset on Large Clusters, 2004

hadoop

[1] Apache Hadoop



- **Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models**
- **Apache Hadoop is an open source implementation of the map-reduce computing paradigm**
- **Apache Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage**

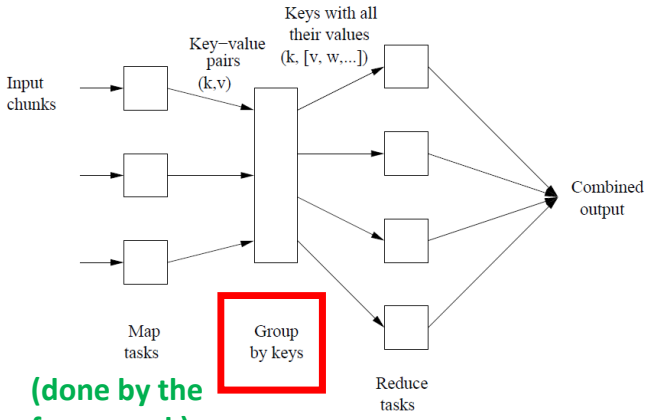
➢ **Practical Lecture 5.1 will offer some concrete examples of map-reduce applications in modern Cloud computing environments today**

Map-Reduce Computing Paradigm – The Programming Model

- Enables many 'common calculations easily' on large-scale data
  - Efficiently performed on computing clusters (but security critics exists)
- Offers reliable system
  - System is tolerant of hardware failures in computation
- Simple Programming Model
  - Users need to provide two functions Map & Reduce with key/value pairs
  - Tunings are possible with many configurations & combine operation
- Key to the understanding: The Map-Reduce Run-Time
  - Three phases – not just 'map-reduce'

[4] MapReduce: Simplified Dataset on Large Clusters, 2004

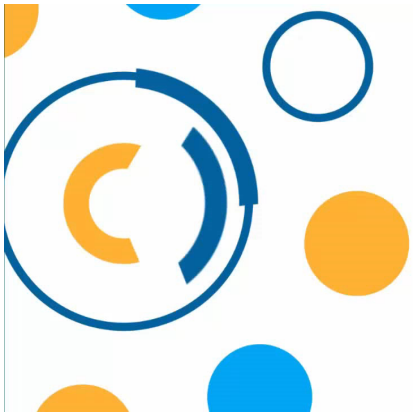
Modified from [9] Mining of Massive Datasets



- **The Map-Reduce Computing paradigm offers the following key functionalities:**
- **(1) Takes care or the partitioning of input data and the communication**
- **(2) Manages parallel execution and performs sort/shuffle/grouping**
- **(3) Coordinates/schedules all tasks that either run Map and Reduce tasks**
- **(4) Handles faults/errors in execution and even re-submit tasks if necessary**

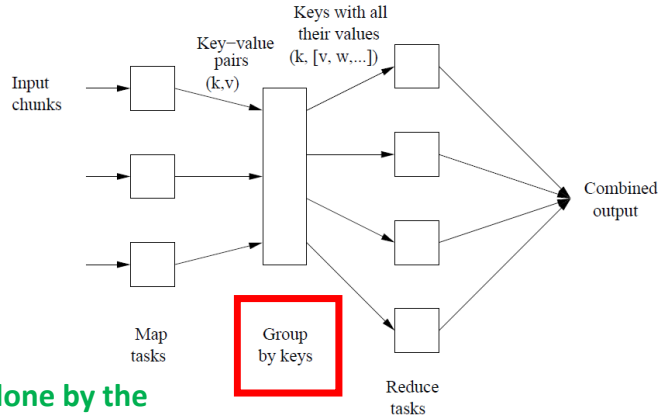
Understanding Map-[Sort/Shuffle/Group]-Reduce & Key-Value Data Structures

- Programming Model
  - Known as 'two phase approach', but actually 'three phases'
  - Two key functions to program by user: map and reduce
  - Third phase 'sort/shuffle' works with keys and sorts/groups them
  - Input keys and values **(k1,v1)** are drawn from a different domain than the output keys and values **(k2,v2)**
  - Intermediate keys and values **(k2,v2)** are from the same domain as the output keys and values
  - Definition of 'generic' keys is dependent on application problem



- **map (k1,v1) → list(k2,v2)**
- **reduce (k2,list(v2)) → list(v2)**

[14] YouTube video, Map-Reduce explained



(done by the framework)

Modified from [9] Mining of Massive Datasets

➢ **Practical Lecture 5.1 will offer some concrete examples of map-reduce applications that leverage the group/sort/shuffle techniques**

Key-Value Data Structure – Simple 'Wordcount' Application Example

```
// counting words example

map(String key, String value):
  // key: document name
  // value: document contents
for each word w in value:
EmitIntermediate(w, "1");

// the framework performs sort/shuffle
// with the specified keys

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

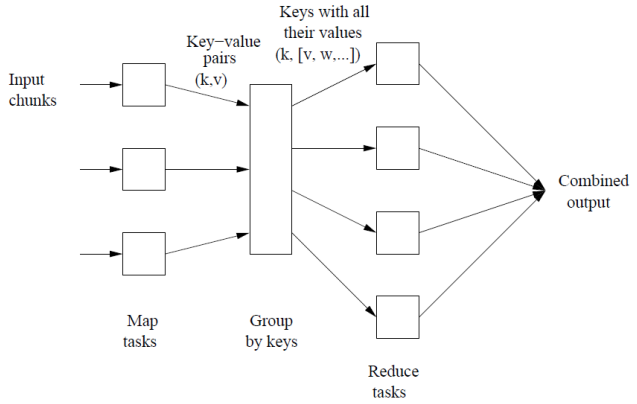
- Goal: Counting the number of each word appearing in a document (or text-stream more general)
- Wordcount in documents or Twitter textstreams are excellent examples for independent 'nicely parallel'

- Key-Value pairs are implemented as Strings in this text-processing example for each function and as 'Iterator' over a list

- Map (docname, doctext) →list (wordkey, 1), …

- Reduce (wordkey, list (1, …)) →list (numbercounted)

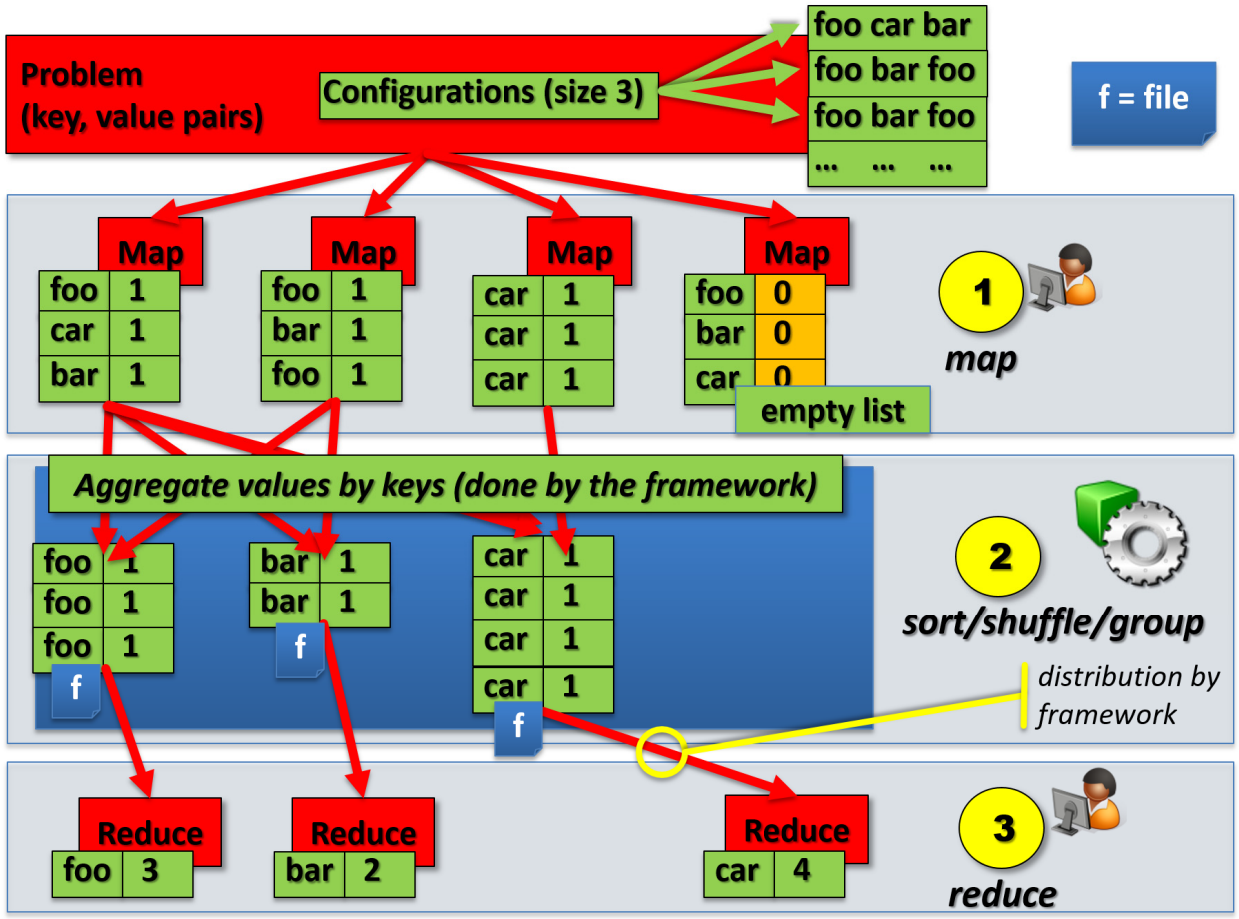
- map (k1,v1) → list(k2,v2)
- reduce (k2,list(v2)) → list(v2)

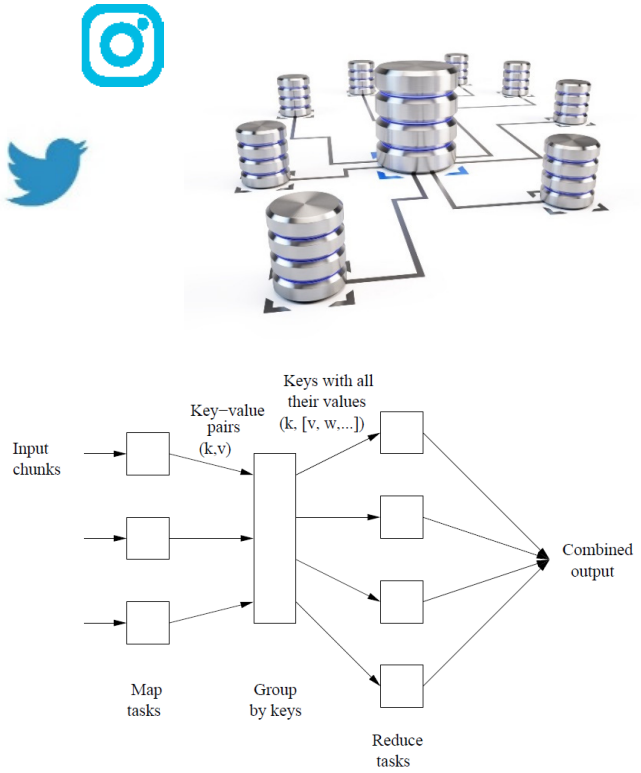


[4] MapReduce: Simplified Dataset on Large Clusters, 2004

➢ **Practical Lecture 5.1 will offer some concrete examples of map-reduce applications that leverage the group/sort/shuffle techniques**

Map-Reduce Practical Application Example: Text(stream) Processing

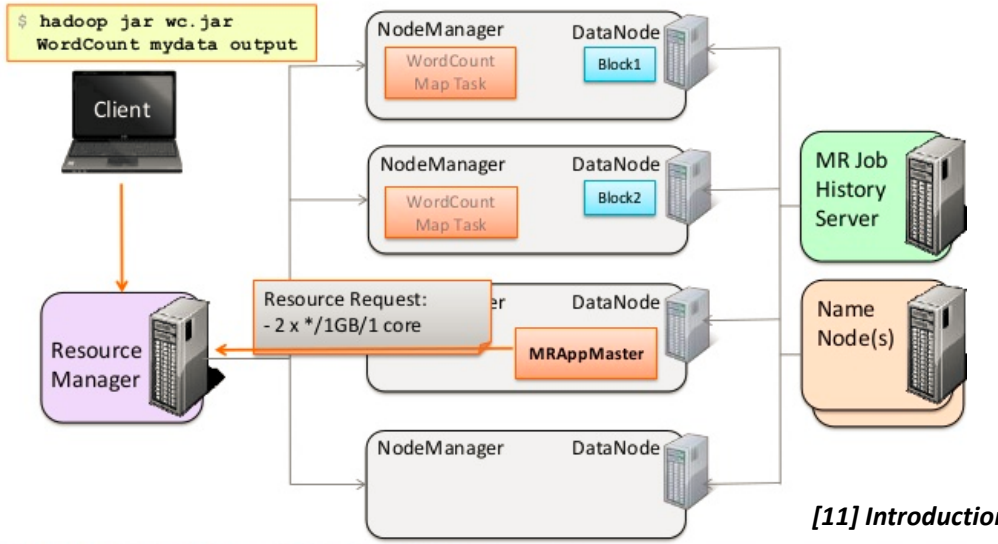




[4] MapReduce: Simplified Dataset on Large Clusters, 2004

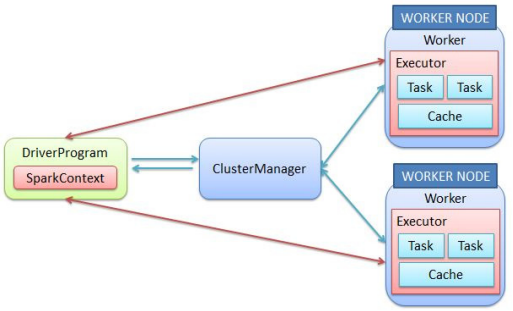
Hadoop Resource Manager YARN & Wordcount Example – Revisited

- Apache Hadoop offers the Yet Another Resource Negotiator (YARN) scheduler for map-reduce jobs
- Idea of Yarn is to split up the functionalities of resource management & job scheduling/monitoring
- The ResourceManager is a scheduler that controls resources among all applications in the system
- The NodeManager is the per-machine system that is reponsible for containers, monitoring of their resource uage (cpu, memory, disk, network) and reporting to the ResourceManager



[11] Introduction to Yarn

[1] Apache Hadoop Web page



[12] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

➢ **Practical Lecture 5.1 will offer some concrete examples of map-reduce applications in modern Cloud computing environments today**

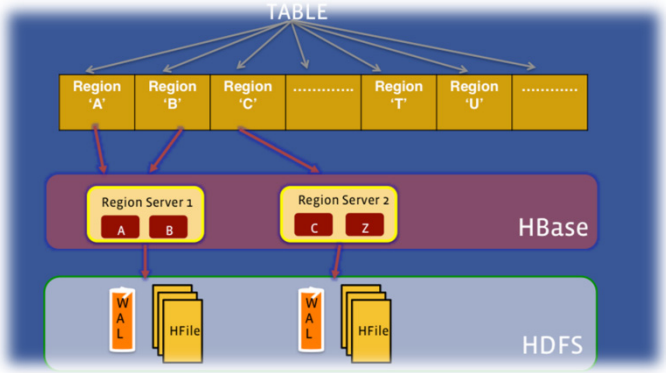
Online Social Network (OSN) Architecture – Facebook Example Map-Reduce

- HBase for Messaging & Indexing
  - Hbase is known as the 'NoSQL key-value store' for Apache Hadoop
  - Scales to Facebook 600 million monthly visitors
  - Used for messages (integrate SMS, chat, email & Facebook Messages all into one inbox)
  - Used for search functionality (posts indexing)
- 'Big Data' Challenges addressed
  - One billion new posts added per day
  - Search index contains more than one trillion total posts (i.e. ~ 700 TB)
  - MySQL DB data → harvested data in an HBase cluster → execute Hadoop map-reduce jobs to build index in parallel

[16] Facebook Code, HBase at Facebook

[15] Apache HBase

(Facebook data physically shared in regions)
(Writes → in-memory write-ahead log WAL →HFile)

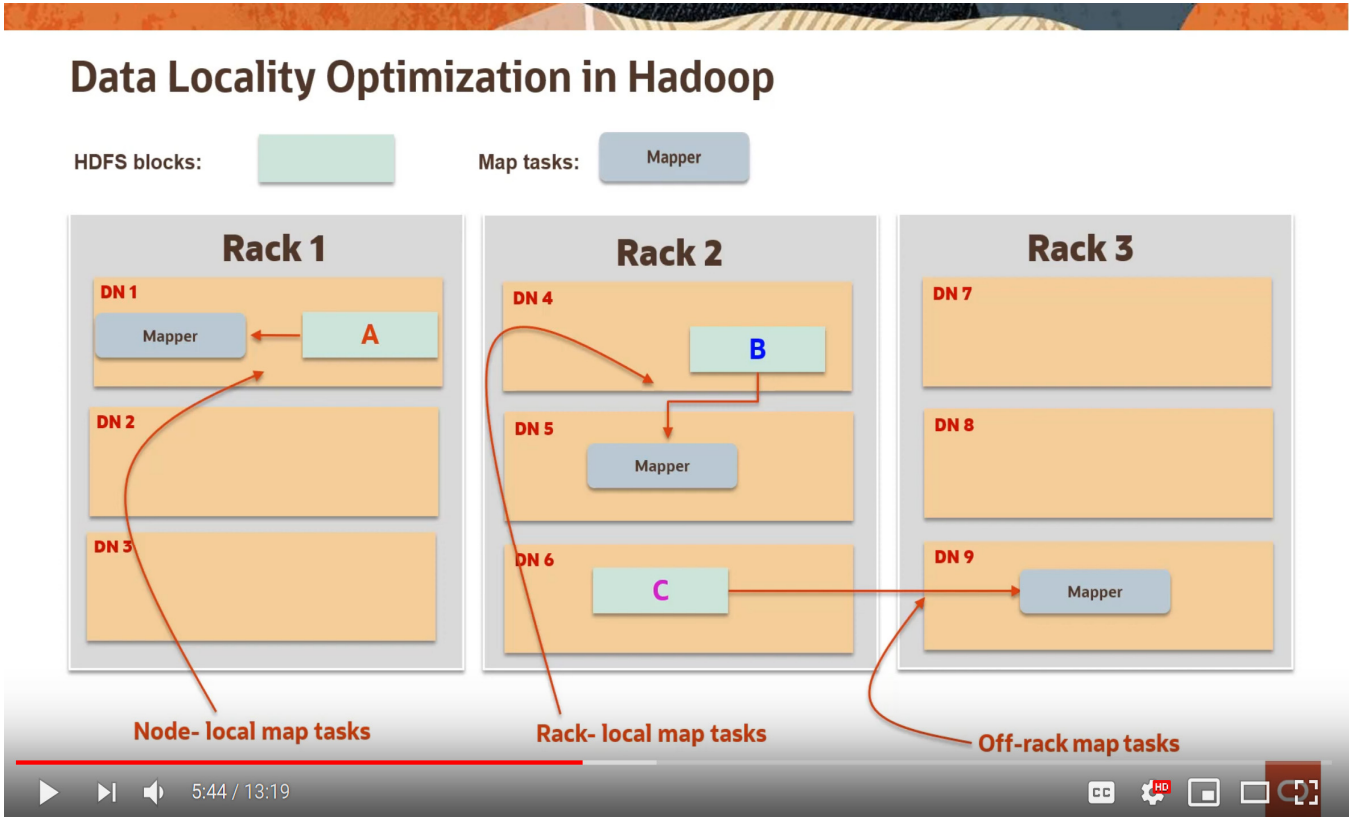


[17] Facebook Engineering, Building Post Search

- **Facebook is using the NoSQL key-value store Apache Hbase & Apache Hadoop map-reduce jobs to build the post index in parallel with ~700 TB of data**

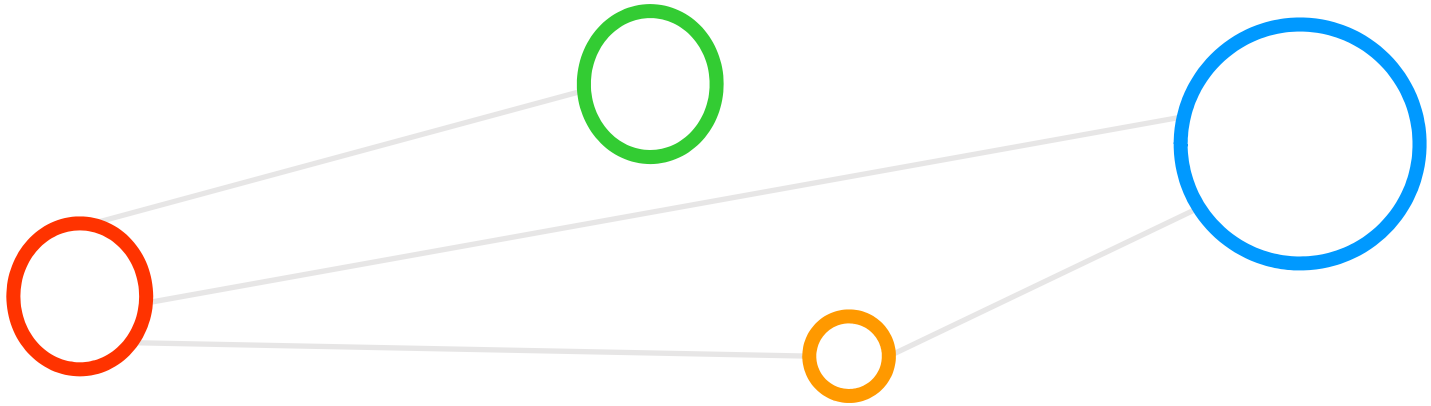
➢ **Lecture 11 will explain the concept of NoSQL key-value stores in the context of Online Social Networking (OSN) and differences to SQL**

[Video] Map-Reduce Summary with Examples



[13] YouTube video, Introduction to Map-Reduce Framework

Map-Reduce Ecosystem in Clouds & Applications



Networking & Big Data Impacts on Cloud Computing – Revisited (cf. Lecture 3)

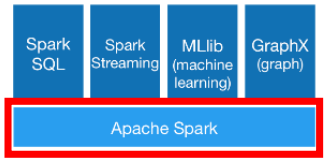
- Requirements for scalable programming models and tools
  - CPU speed has surpassed IO capabilities of existing cloud resources
  - Data-intensive clouds with advanced analytics and analysis capabilities
  - Considering moving compute task to data vs. moving data to compute
  - E.g., MS Azure HDInsight is only one concrete example of many cloud solutions

Microsoft Azure

HDInsight

[2] Microsoft Azure HDInsight Service

  - E.g., clouds often use open source implementations like Apache Spark & Apache Hadoop & related technologies



[4] Apache Spark

- **Apache Spark is a fast & general engine for large-scale data processing optimized for memory usage**
- **Open source & compatible with cloud data storage systems like Amazon Web Services (AWS) S3**
- **Apache Spark is often used as cloud service offerings in commercial clouds like MS Azure, AWS, or Google Cloud**

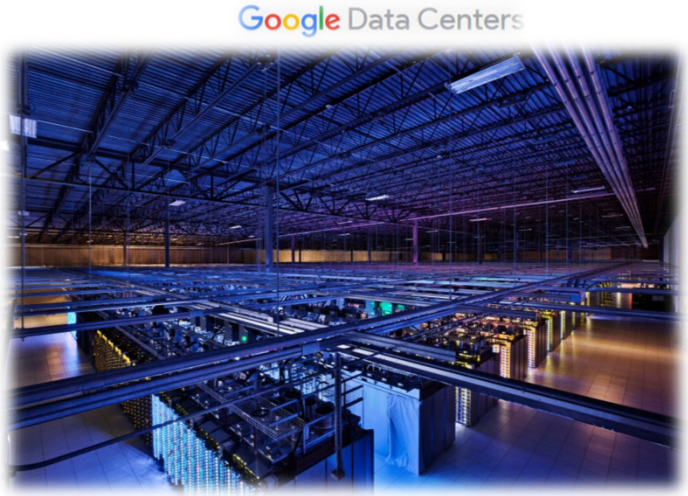
- Requirements for Reliable Filesystems
  - Traditional parallel filesystems need to prove their 'big data' feasibility
  - Emerging new forms of filesystems that assume hardware error constantly
  - E.g. Hadoop distributed file system (HDFS) as a specialized filesystem that assumes errors as default

[3] Apache Hadoop

Assume Failure in Data Center Design is Realistic – Revisited (cf. Lecture 4)

- Example: Google Data Center, Council Bluffs, Iowa, USA
  - Scale of thousands of servers means always concurrent failure
  - Hardware failure or software failure (1 percent of nodes is common)
  - CPU failure, disk I/O failure
  - Network failure, switch error
  - Sofware failure, scheduler error
  - E.g., error in execution as part of as Apache Spark cluster job means being (partly) resubmitted for execution in an automatic way (cf. Lecture 3)
- Whole data center does not work
  - E.g. in the case of a power crash?!
  - E.g. vendor of software has serious bug
  - E.g. central aspects of filesystems are down

[2] Apache Spark



[10] Google Data Centers, wired.com

- **Cloud data centres need to operate with failures: Computing service and user data should not be lost in a failure situation and reliability can be achieved by redundant hardware in centres**
- **Software keeps multiple copies of data in different locations and keeps data accesible during errors**

Relevant Statistics & Formalize 'good data center' – Faults, Errors, and Failures?

- 'Module reliability'
  - Measure of continuous service accomplishment
  - Quantified in terms of 'mean time to failure (MTTF)'
  - Reciprocal: 'failures in time (FIT)'
  - Service interruption quantified as 'mean time to repair (MTTR)'
  - 'Mean time between failures (MTBF) = MTTF + MTTR'
- 'Module availability'
  - Measure of service accomplishment with respect to the alternation between accomplishment & interruption
  - The formula is relevant for non-redundant systems with repair:

- **Basic assumption: Priority for storage is to remember information**
- **Dependability: Quality of delivered service such that reliance can justifiably placed on this service**
- **'System failure' occurs when actual behavior deviates from specified behavior**
- **System failure occurs because of an 'error'**
- **The cause of an error is a 'fault'**
- **A fault creates a 'latent error', which becomes effective when it is activated**
- **Time between error and failure is the 'error latency'**
- **Need to Quantifying Dependability of Cloud data centers: Reliability & Availability**

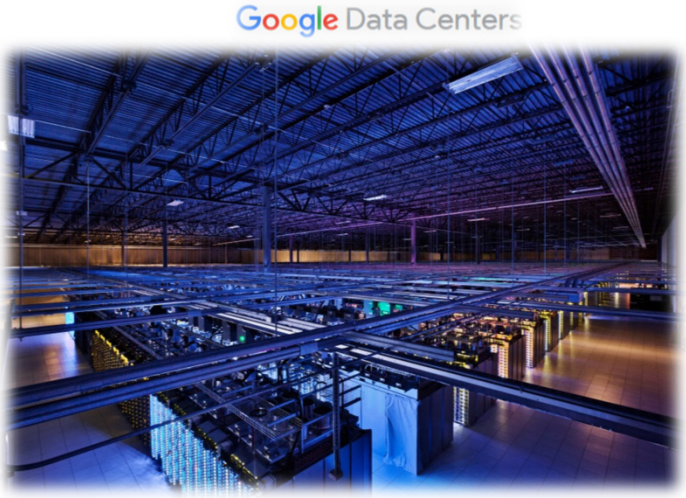
$$\textbf{Module Availability} = \frac{MTTF}{MTTF + MTTR}$$

[18] Laprie et al., 1986

Faults by Cause as experienced by Large Cloud Data Centers

- Hardware faults
  - Devices that fail (e.g. due to an alpha particle hitting a memory cell)
- Design faults
  - Faults in software (usually) and hardware design (occasionally)
- Operation faults
  - Mistakes by operations and maintenance personnel
- Environmental faults
  - Fire, flood, earthquake, power failure, and sabotage
- One approach to be prepared for Faults: RAID Levels
  - Another approach: 'specialized filesystems'

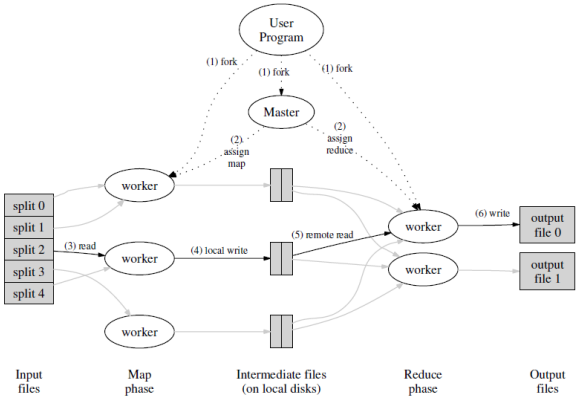
- **Rredundant array of independent disks (RAID) is a system to prepare data centers for faults**
- **Primary cause of failures in large big data systems today is faults by human operators**
- **The Map-Reduce ecosystem approach is to rely on a specialized filesystem**



[10] Google Data Centers, wired.com

Hadoop Distributed File System (HDFS) as ‘Specialized Filesystem’

- A ‘specialized filesystem’ designed for reliability and scalability
  - Designed to run on ‘comodity hardware’
  - Many similarities with existing ‘distributed file systems’
  - Takes advantage of ‘file and data replication concept’
- Differences to traditional distributed file systems are significant
  - Designed to be ‘highly fault-tolerant’ → improves dependability!
  - Enables use with applications that have ‘extremely large data sets’
- Provides ‘high throughput access’ to application data
  - HDFS relaxes a few POSIX requirements
  - Enables ‘streaming access’ to file system data
- Origin
  - HDFS was originally built as infrastructure for the ‘Apache Nutch web search engine project’



[4] MapReduce: Simplified Dataset on Large Clusters, 2004

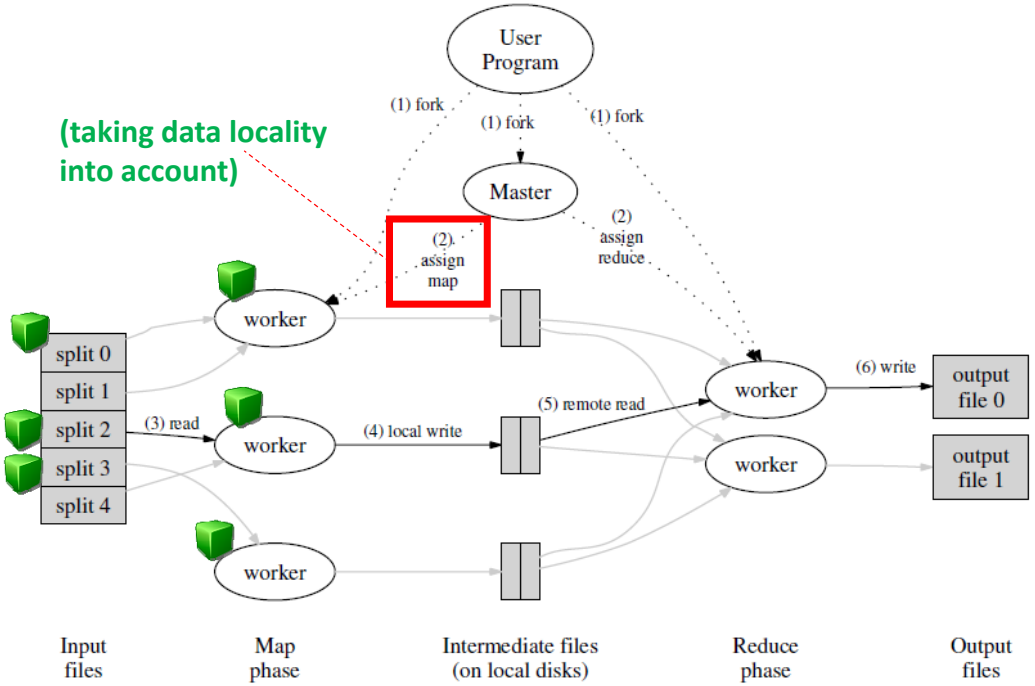
[3] Apache Hadoop

[19] The Hadoop Distributed File System

‘Data Locality’ Scheduling & Communication in Map-Reduce Algorithms

- Challenges in Communication & Scheduling of ‘Tasks’
  - Greatest cost is in the communication overhead of a map-reduce algorithm vs. use ‘data locality’
  - Algorithms trade communication cost against degree of parallelism and use principle of ‘data locality’

(use less network)

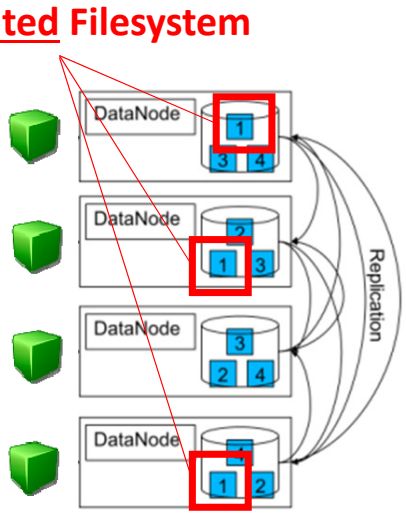


- **The ‘Data locality means that network bandwidth is conserved by taking advantage of the approach that the input data (managed by DFS) is stored on (or very near, e.g. same network switch) the local disks of the machines that make up the computing clusters**
- **Relying on ‘data locality’ drives the HDFS design further towards replication strategies**

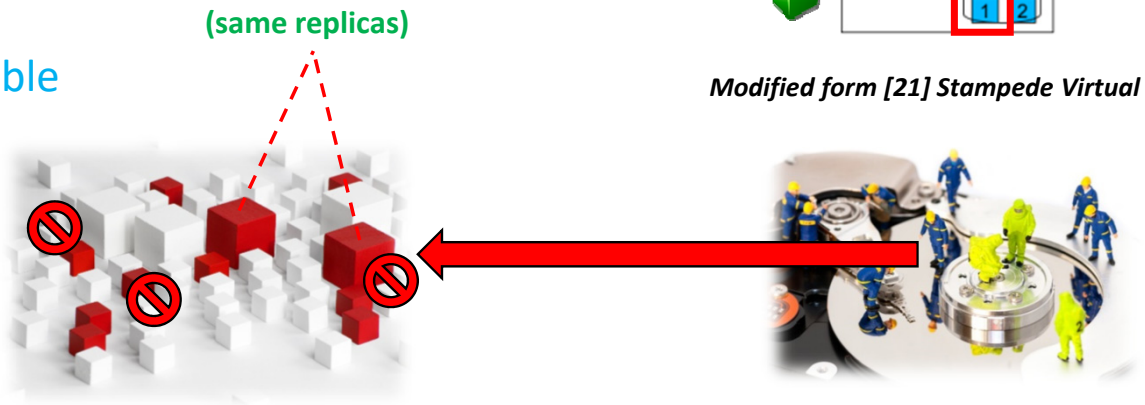
[4] MapReduce: Simplified Dataset on Large Clusters, 2004

Replication Concept Design of HDFS

- Ideal: Replicas reside on 'failure-independent machines'
  - Availability of one replica should not depend on availability of others
  - Requires ability to place replica on particular machine
  - 'Failure-independent machines' hard to find, but in 'system design' easier
- Replication should be hidden from users
  - But replicas must be distinguishable at lower level
  - Different DataNode's are not visible to end-users
- Replication control at higher level
  - Degree of replication must be adjustable (e.g. Hadoop configuration files)



Modified form [21] Stampede Virtual Workshop

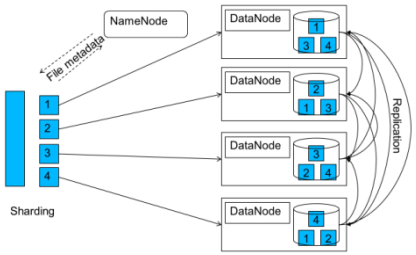


- **File replication is a useful redundancy for improving availability and performance**

HDFS Architecture

- HDFS cluster consists of a single 'NameNode' (i.e., Master/Head)
  - Master server that manages the file system namespace
  - Regulates access to files by clients
- HDFS cluster consists of N 'DataNode(s)'
  - Typically one per node in the cluster
  - Manage storage attached to the nodes that they run on
- HDFS exposes a 'file system namespace'
  - User data can be stored in files and directories ('as usual')
  - Not directly accessible outside
- File Handling
  - Files enormously in size (GBs, TBs, etc.), better 'accumulated over time'
  - Files are rarely updated (optimized rather for read/append – not re-write)
  - Bring computation to the files, rather the files to the computing resource ('data locality principle')

Master / Head **Workers**



- **HDFS enables 'parallel reading & processing' of the data files and their replicas if needed**
- **Files are divided into blocks – also named as 'chunks' (default 64MB), relatively 'large blocks' for a filesystem**
- **Blocks are replicated at different compute nodes (default three times, configurable)**
- **Blocks holding copy of one dataset are distributed across different racks**

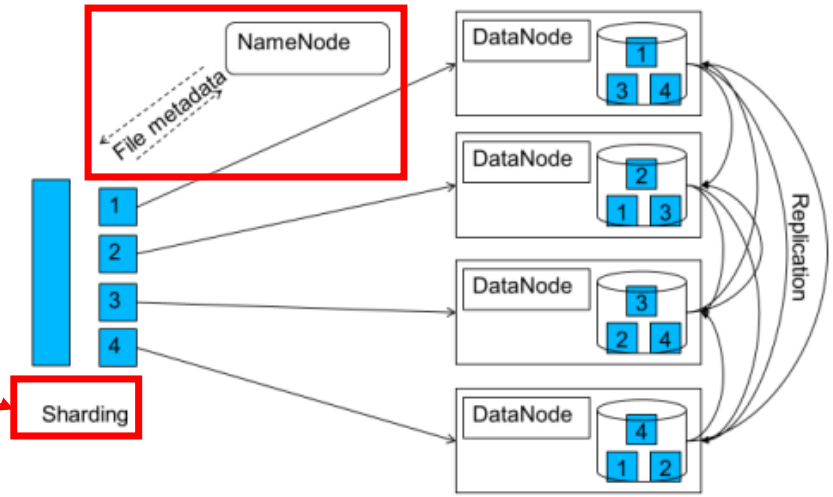
Modified form [21] Stampede Virtual Workshop

Specific Master & Head Node as NameNode

- The master / head as name node is replicated
  - A directory for the file system as a whole knows where to find the copies
  - All participants using the HDFS know where the directory copies are
  - Determines the mapping of ‘pure data blocks’ to DataNodes (i.e., also callled file metadata, but term bigger)
  - Executes file system namespace operations
  - E.g. opening, closing, and renaming files and directories
- DataNode
  - Serving ‘read and write requests’ from HDFS filesystem clients
  - Performs block creation, deletion, and replication (upon instruction from the NameNode)

- **The Master/ Head as Name node keeps metadata (e.g. node knows about the blocks of files in HDFS)**

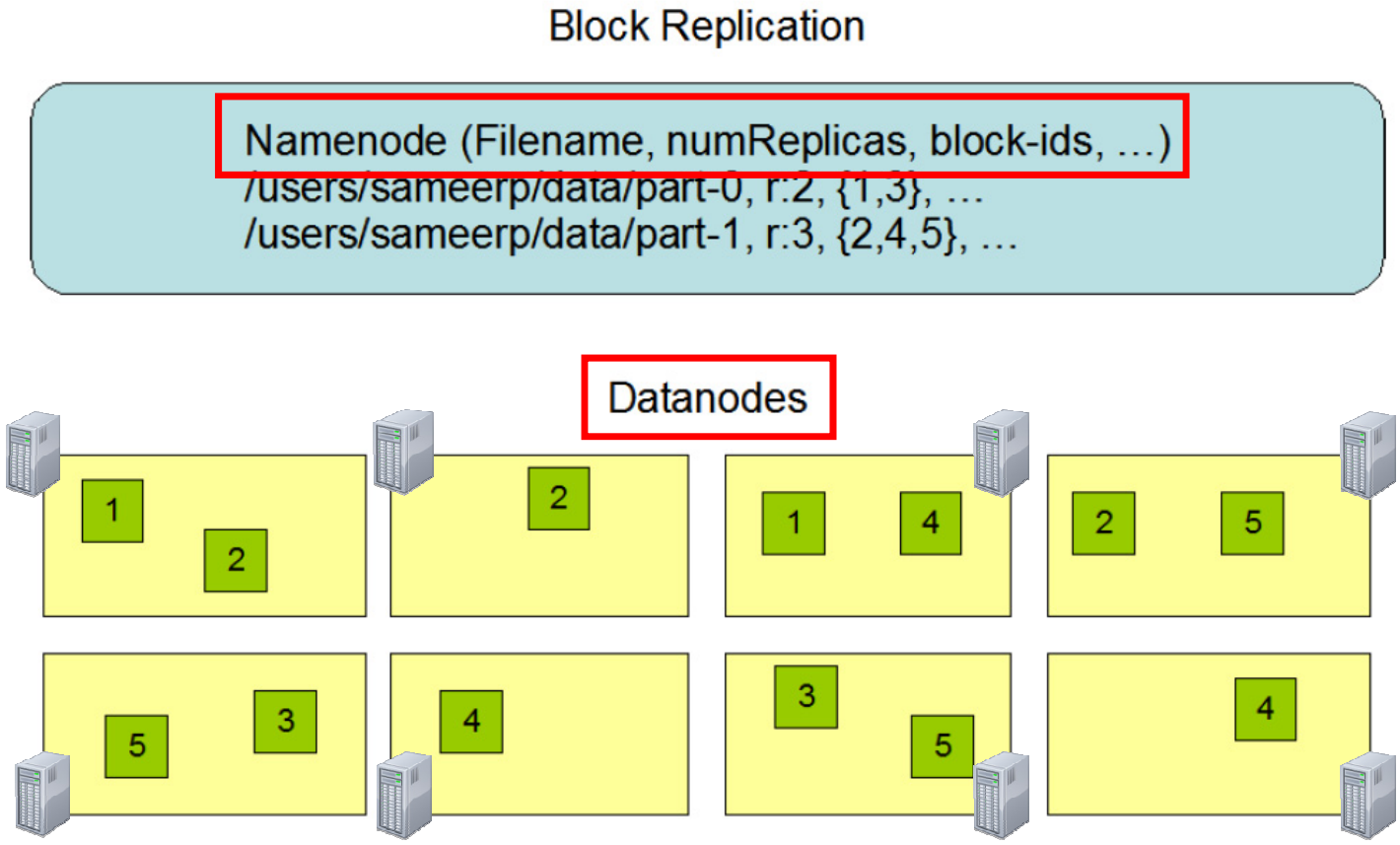
e.g. horizontal partitioning



Modified form [21] Stampede Virtual Workshop

- **Some cloud installations today have two head nodes with name nodes in order to prevent a single point of failure in the overall larger system design (e.g., 2 head nodes, 30 worker nodes)**

Summary with HDFS File/Block(s) Distribution Example



[19] The Hadoop Distributed File System

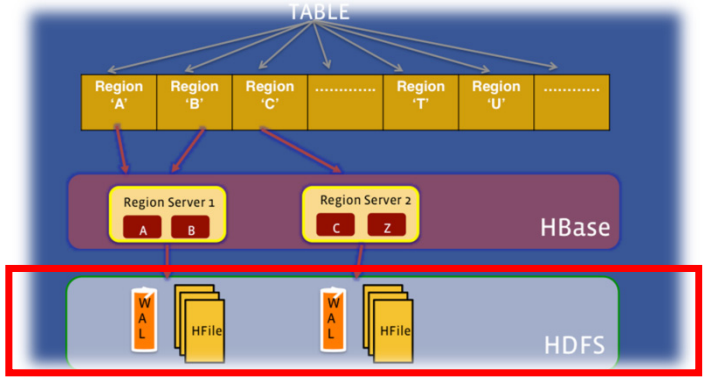
Online Social Network (OSN) Architecture – Facebook Example using HDFS

- HBase for Messaging & Indexing
  - Hbase is known as the 'NoSQL key-value store' for Apache Hadoop
  - Scales to Facebook 600 million monthly visitors
  - Used for messages (integrate SMS, chat, email & Facebook Messages all into one inbox)
  - Used for search functionality (posts indexing)
- 'Big Data' Challenges addressed
  - One billion new posts added per day
  - Search index contains more than one trillion total posts (i.e. ~ 700 TB)
  - MySQL DB data → harvested data in an HBase cluster → execute Hadoop map-reduce jobs to build index in parallel

[16] Facebook Code, HBase at Facebook

[15] Apache HBase

(Facebook data physically shared in regions)
(Writes → in-memory write-ahead log WAL →HFile)



[17] Facebook Engineering, Building Post Search

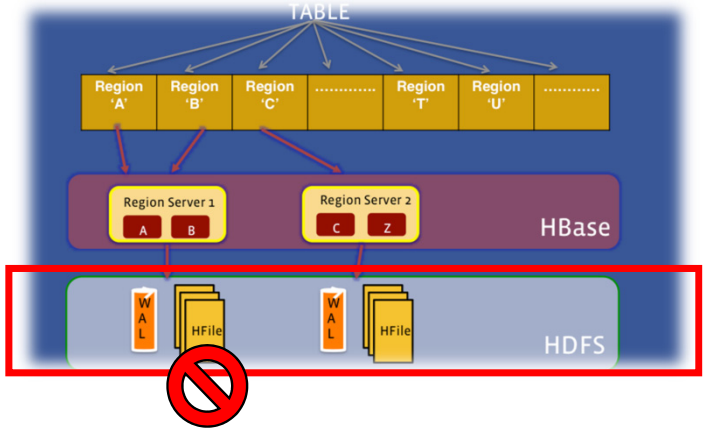
- **Facebook is using the NoSQL key-value store Apache Hbase & Apache Hadoop map-reduce jobs to build the post index in parallel with ~700 TB of data**

➢ **Lecture 11 will explain the concept of NoSQL key-value stores in the context of Online Social Networking (OSN) and differences to SQL**

Benefits of File Replication

- Tradeoff between ‘update consistency’ and performance
  - HDFS is completely optimized towards aggregating data/time, poor updates!
  - Good performance when ‘data read occurs more often than data writes’
- Fault Tolerance
  - HDFS provides high ‘reliability and dependability’ via data replication
- Easy Management
  - HDFS is scalable towards data, but also towards ‘operating the system’
  - No immediate action required if disk fails (→ can be different in RAIDs)
  - No immediate repair action required if node fails (→ other replicas exist)

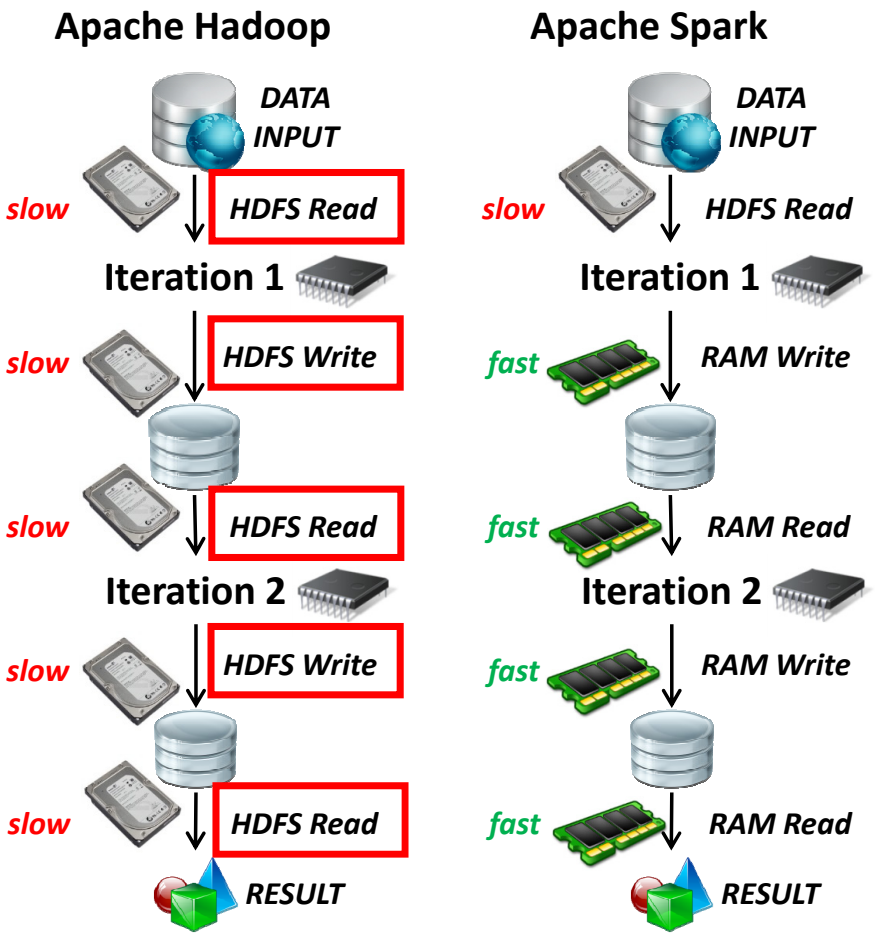
[3] Apache Hadoop



modified from [17] Facebook Engineering, Building Post Search

- **The benefits of file (and data) replication is performance, fault tolerance, and (easy) management**
- **Easy management means that no immediate action is required to keep the system operating**
- **Repairs of large-scale systems are typically done periodically for a whole collection of failures**

Apache Spark vs. Apache Hadoop – Revisited (cf. Lecture 3)

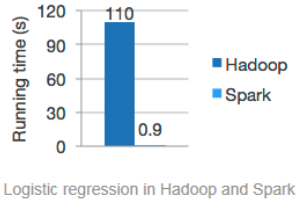


- Apache Spark Results
  - Up to 100% faster then traditional Hadoop (for some applications)
  - Requires 'big memory' systems to perform well

- **One key difference between Apache Spark vs. Apache Hadoop is that slow HDFS reads and writes are exchanged with fast RAM reads and writes in several iterations that might be part of a larger workflow**
- **A workflow is a sequence of processing tasks on data that leads to results via 'iterations'**

[20] big-data.tips, Apache Spark vs. Hadoop

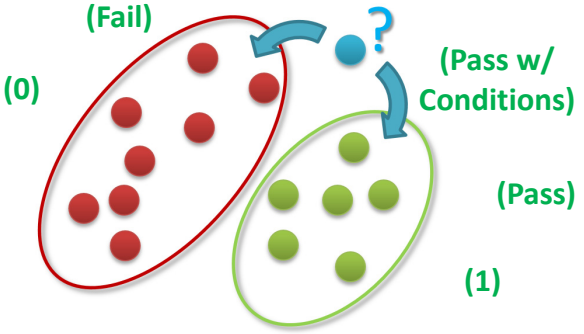
(e.g., application of food inspection analysis, cf. Lecture 2)



Logistic regression in Hadoop and Spark

Microsoft Azure

HDInsight



Understand AWS Cloud Service Portfolio – Analytics Services

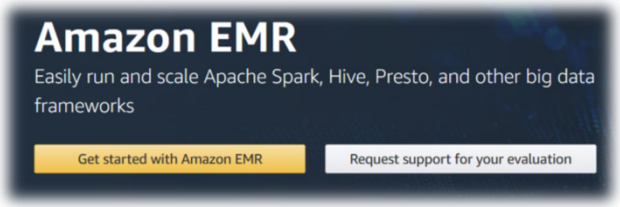
- Multiple analytics products
  - Extracting insights and actionable information from data requires technologies like analytics & machine learning
- Analytics Services
  - Amazon Athena: Serverless Query Service
  - Amazon ElasticMapReduce (EMR): Hadoop
  - Amazon ElasticSearch Service: Elasticsearch on AWS
  - Amazon Kinesis: Streaming Data
  - Amazon QuickSight: Business Analytics
  - Amazon Redshift: Data Warehouse

- **AWS offers a wide variety of Big Data Analytics Cloud services such as Amazon Athena (interactive analytics), Amazon EMR (big data processing with Apache Hadoop & Spark), Amazon Redshift (data warehousing), Amazon Kinesis (real-time analytics), Amazon Elasticsearch Service (operational analytics), and Amazon Quicksight (dashboards and visualizations)**

[22] AWS Web page

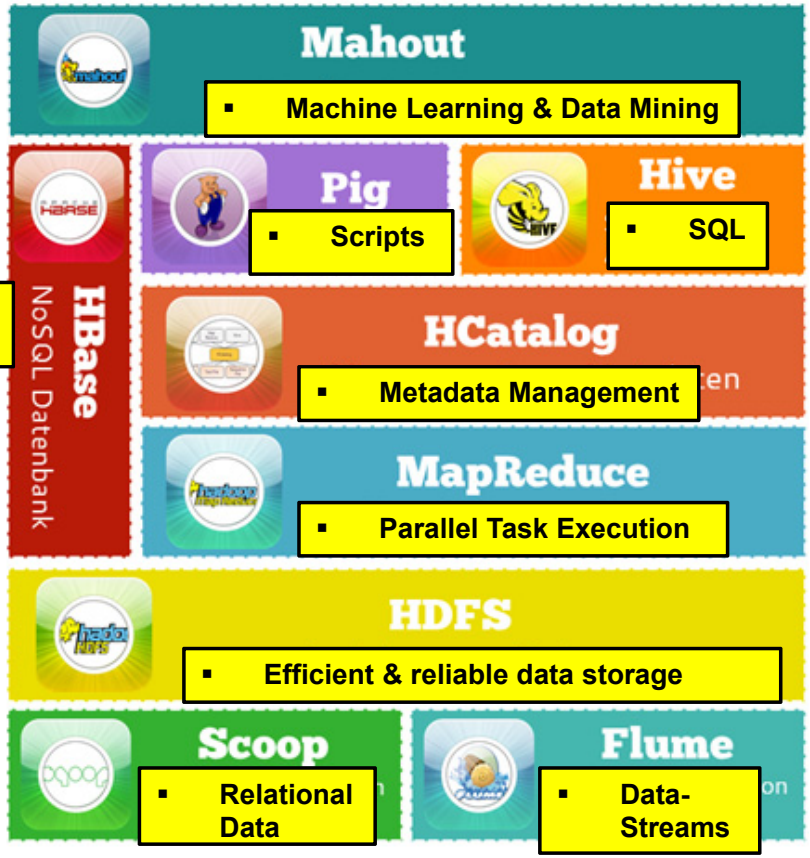
AWS Analytics services

Category	Use cases	AWS service
Analytics	Interactive analytics	Amazon Athena
	Big data processing	Amazon EMR
	Data warehousing	Amazon Redshift
	Real-time analytics	Amazon Kinesis
	Operational analytics	Amazon Elasticsearch Service
	Dashboards and visualizations	Amazon Quicksight



➢ **Practical Lecture 5.1 will explore a concrete Cloud application example of using map-reduce via Apache Hadoop & its ecosystem tools**

Larger Hadoop Ecosystem with Big Data Analytics ‘Tools‘

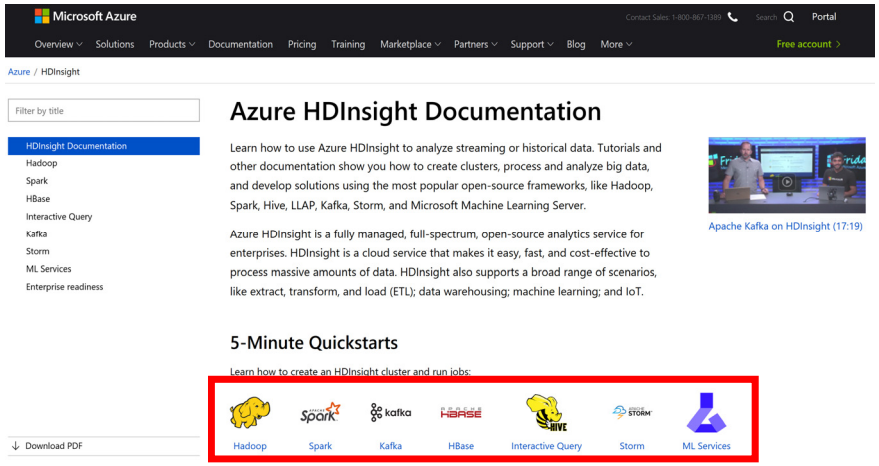


[29] Google Dataproc service

[25] Microsoft Azure HDInsight Service

Modified from [24] Map-Reduce

[23] Azure HDInsight Web page



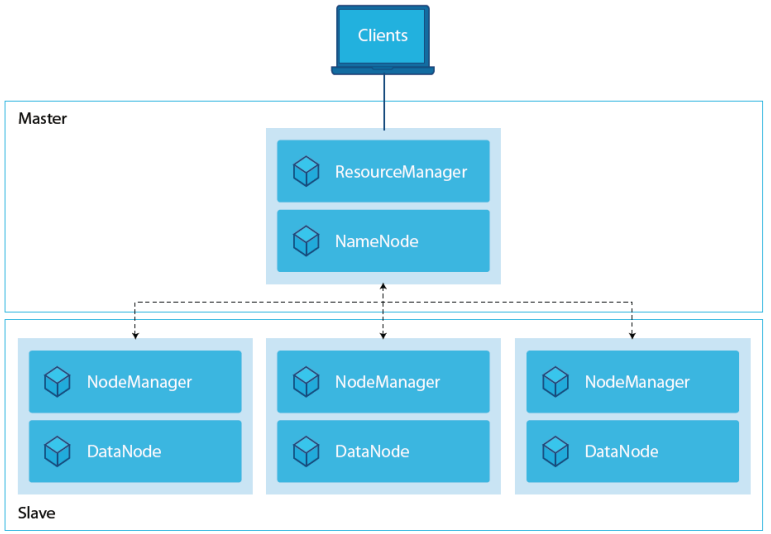
(known open source analytics frameworks)

➢ **Practical Lecture 5.1 will explore a concrete Cloud application example of using map-reduce via Apache Hadoop & its ecosystem tools**

Big Data Analytics with Apache Hadoop on OpenStack & Kafka

- Resource Manager (uses Nova)
  - Scheduler Yarn to allocate resources to various applications on the cluster
- NameNode (uses Nova & Cinder)
  - Metadata about the data blocks are stored in the NameNode
  - Provides lookup functionality and tracking for all data or files in the Apache Hadoop cluster
- NodeManager (uses Nova)
  - Takes instructions from YARN and responsible to execute and monitor applications
- Datanode (uses Nova & Cinder)
  - Store and process the data

[28] OpenStack Web page



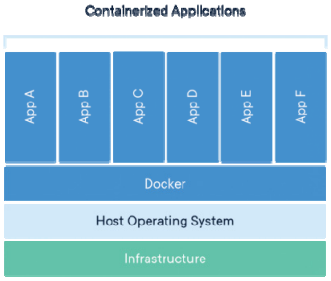
[27] OpenStack Paper 'Big Data'

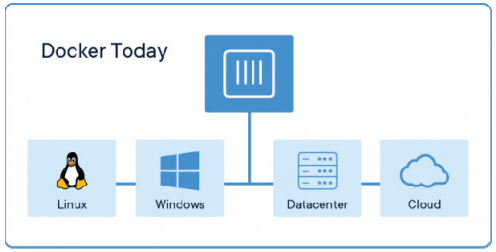
- **One particular challenge when working with Apache Hadoop & the large ecosystem with many tools in a scalable fashion is the enormous configuration that needs to be (semi-) automated**

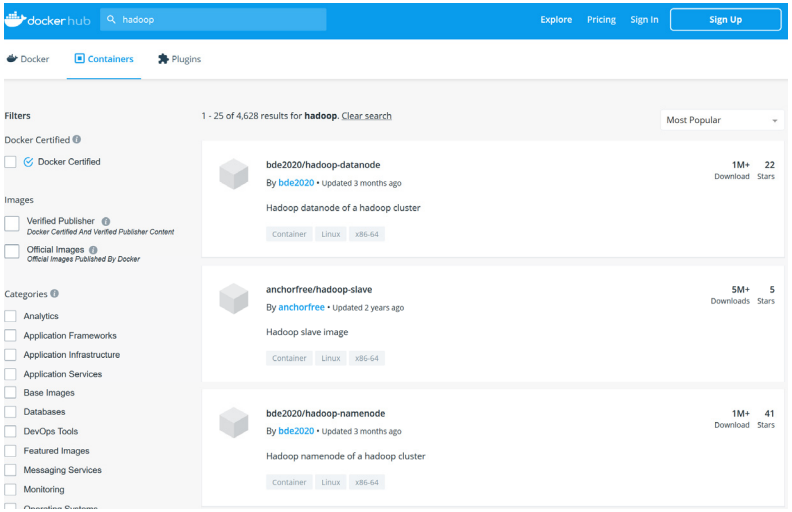
➢ **Lecture 13 will provide more details on using the OpenStack Cloud Operating System for creating Cloud Environments for Data Analytics**

Semi-Automatic Configuration & Deployment Tools

- Docker
  - Open Source Container technology
  - Docker Hub: community image repository with free applications, e.g. including Hadoop



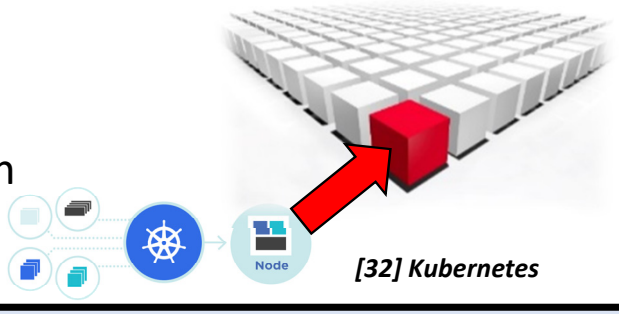




[30] Docker Hub Web page

[31] Docker Web page

- Kubernetes **(originally designed by Google)**
  - Open Source Container-Orchestration system for automating computer application deployment, scaling, and management



[32] Kubernetes

[29] Google Dataproc service

➢ **Lecture 12 will provide more details on container technologies like Docker and related container management solutions like Kubernetes**

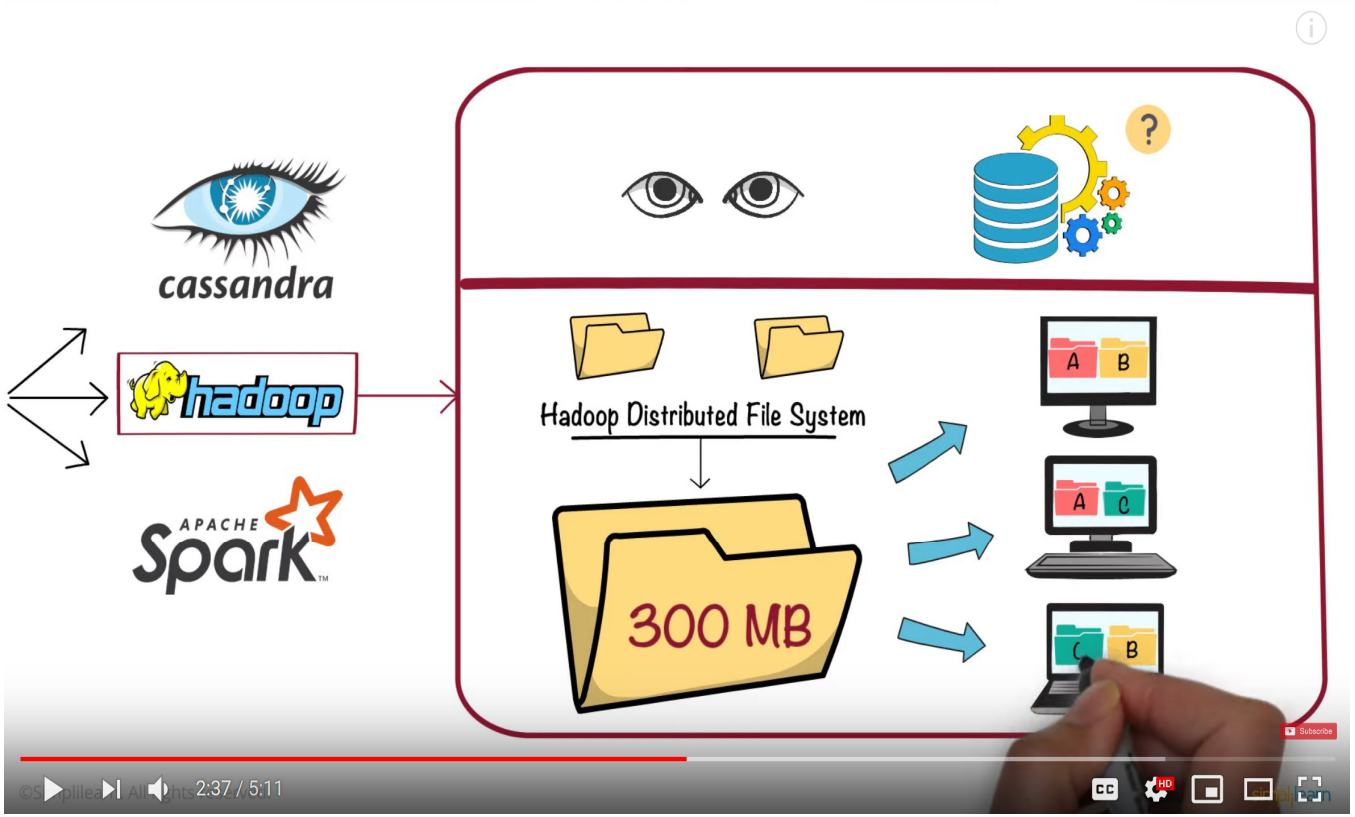
Limits: Not every problem can be solved with Map-Reduce

- Example: Amazon Online retail sales
  - Requires a lot of updates on their product databases on each user action (a problem for the underlying file system optimization)
  - Processes that involve 'little calculation' but still change the database
  - Employs thousands of computing nodes and offers them ('Amazon Cloud')
  - (maybe they using map-reduce for certain analytics: buying patterns)
- Example: Selected machine learning methods
  - Some learning algorithms require rather 'iterative map-reduce'
  - The traditional 'map-reduce → done' approach needs to be extended

- **Map-Reduce is not a solution to every parallelizable problem**
- **Only specific algorithms benefit from the map-reduce approach**
- **No communication-intensive parallel tasks (e.g. PDE solver) such as done in parallel computing with the Message Passing Interface (MPI)**
- **Applications that require often updates of existing datasets (writes)**
- **Implementations often have severe security limitations (distributed)**

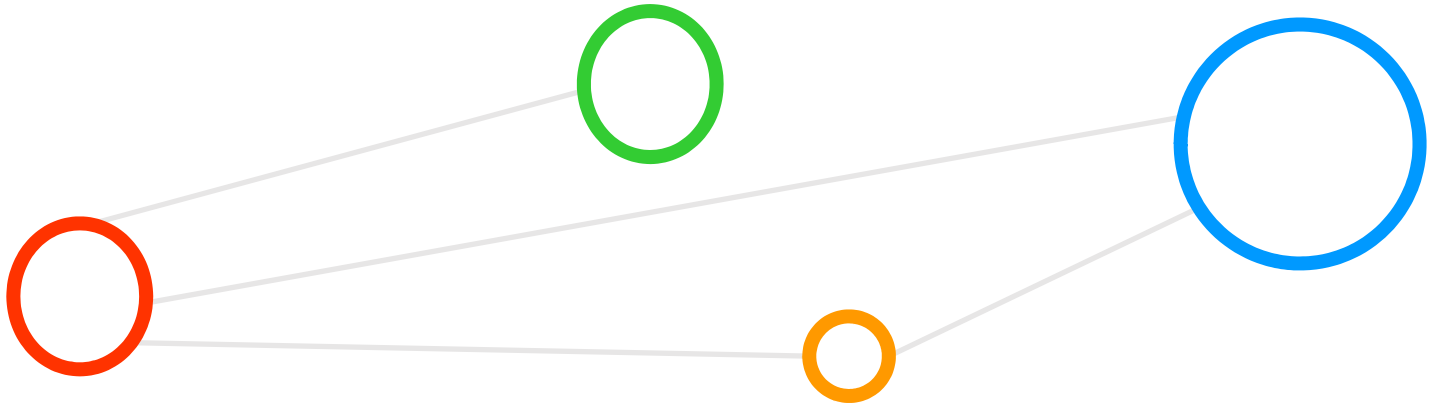
[9] Mining of Massive Datasets

[Video] Map-Reduce Ecosystem & Relationships to Big Data with Applications



[26] YouTube video, Big Data in 5 Minutes

Lecture Bibliography



Lecture Bibliography (1)

- [1] Apache Hadoop Web page, Online: http://hadoop.apache.org/
- [2] Apache Spark Web page, Online: http://spark.apache.org/
- [3] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book, Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [4] J. Dean, S. Ghemawat, 'MapReduce: Simplified Data Processing on Large Clusters', OSDI'04: Sixth Symposium on Operating System Design and Implementation, December, 2004
- [5] Introduction to High Performance Computing for Scientists and Engineers, Georg Hager & Gerhard Wellein, Chapman & Hall/CRC Computational Science, ISBN 143981192X
- [6] Introduction to Parallel Computing Tutorial, Online: https://computing.llnl.gov/tutorials/parallel_comp/
- [7] Science Progress, Online: http://scienceprogress.org/2009/11/dna-confidential/
- [8] Supercomputer JUWELS Cluster Module – Configuration, Online: https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUWELS/Configuration/Configuration_node.html
- [9] Mining of Massive Datasets, Online: http://infolab.stanford.edu/~ullman/mmds/book.pdf
- [10] 'Google Throws Open Doors to Its Top-Secret Data Center', wired.com, Online: https://www.wired.com/2012/10/ff-inside-google-data-center/all/
- [11] SlideShare, ,'Introduction to Yarn and MapReduce', Online: https://www.slideshare.net/cloudera/introduction-to-yarn-and-mapreduce-2

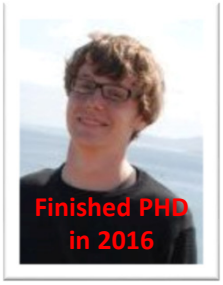
Lecture Bibliography (2)

- [12] Understanding Parallelization of Machine Learning Algorithms in Apache Spark, Online: https://www.slideshare.net/databricks/understanding-parallelization-of-machine-learning-algorithms-in-apache-spark/11
- [13] YouTube, Introduction to Map-Reduce Framework, Online: https://www.youtube.com/watch?v=UzR7XB74tuk
- [14] YouTube, Tutorial: MapReduce explained, Online: https://www.youtube.com/watch?v=lgWy7BwIKKQ
- [15] Apache Hbase Web Page, Online: https://hbase.apache.org/
- [16] Facebook Code, ‚HydraBase – The evolution of HBase@Facebook‘, Online: https://code.facebook.com/posts/321111638043166/hydrabase-the-evolution-of-hbase-facebook/
- [17] Facebook Engineering, ‘Under the Hood: Building posts search‘, Online: https://www.facebook.com/notes/facebook-engineering/under-the-hood-building-posts-search/10151755593228920/
- [18] Jean-claude Laprie, ‘Dependable computing: From concepts to design diversity, Proceedings of the IEEE, 1986, pages 629-638
- [19] Dhruba Borthakur, ‘The Hadoop Distributed File System: Architecture and Design’, Online: http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf
- [20] Big Data Tips, ‘Apache Spark vs Hadoop‘, Online: http://www.big-data.tips/apache-spark-vs-hadoop
- [21] Stampede Virtual Workshop, Online: http://www.cac.cornell.edu/Ranger/MapReduce/dfs.aspx
- [22] Amazon Web Services Web Page, Online: https://aws.amazon.com

Lecture Bibliography (3)

- [23] Microsoft Azure HDInsight Cluster Web page, Online: https://azure.microsoft.com/en-us/services/hdinsight/
- [24] Einführung in Hadoop (German language), Online: http://blog.codecentric.de/2013/08/einfuhrung-in-hadoop-die-wichtigsten-komponenten-von-hadoop-teil-3-von-5/
- [25] Microsoft Azure HDInsight Service, Online: https://azure.microsoft.com/en-us/services/hdinsight/
- [26] YouTube, Big Data in 5 Minutes, Online: https://www.youtube.com/watch?v=bAyrObl7TYE
- [27] OpenStack Paper, 'OpenStack Workload Reference Architecture: Big Data', Online: https://www.openstack.org/assets/software/mitaka/OpenStack-WorkloadRefBigData-v4.pdf
- [28] OpenStack Cloud Operating System, Online: https://www.openstack.org/
- [29] Google DataProc Service, Online: https://cloud.google.com/dataproc/
- [30] Docker Hub, Online: https://hub.docker.com/
- [31] Docker Web page – What is a Container, Online: https://www.docker.com/resources/what-container
- [32] Kubernetes, Online: https://kubernetes.io/

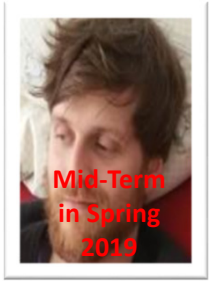
Acknowledgements – High Productivity Data Processing Research Group



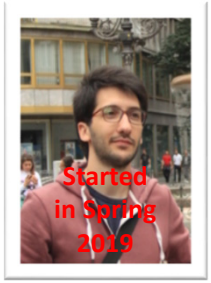
PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon



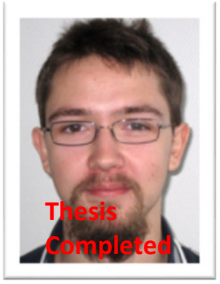
PhD Student
E. Erlingsson



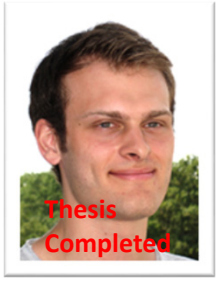
PhD Student
S. Bakarat

PhD Student
R. Sedona

Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)

MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)


Data
Research
Science
Big
science
analysis
processing
simulations
methods
big
technologies
TB
European
computing
computational
Scientific
Supercomputing
Computational
datasets
research
Centre
climate
Computing
nodes
centers
storage
ENES
EUDAT
Juelich
HPC
access
Climate
modelling
structures
System
databases
DLCL
DLCLs
performance
images
often
management
data-intensive
Service
distributed
large-scale
community
PB
E.g
e.g
services
measurement
within
fusion
new
funding
concepts
Policy-based
forms
cross-disciplinary
resources
year
like
Cross-Disciplinary
EU
Test
High
device
Analysis
Computer
expertise
disciplines
results
Approaches
construction
Fusion
Enable
Modelling
different
Environment
Key
scientific
important
files
DKRZ
Use
etc
basis
Infrastructure
physical
IPCC
Common
Methods
Using
Work
increasing
via
Project
MapReduce
create
GB
formats
brain
model
international
SMAQ
engineering
References
codes
JSC
use
acquisition
stored
Resources
analyze
Large
Summary
Level
levels
external
Matters
Concrete
system
Landscape
compute
cores
hundreds
Nutshell
Services
network
AR5
Hardware
Technologies
Earth
directory
Structure
General
current
project
Understanding
Germany
Provide
NASA
Energy
Simulation
systems
day
Processing
Health