DeltaRho

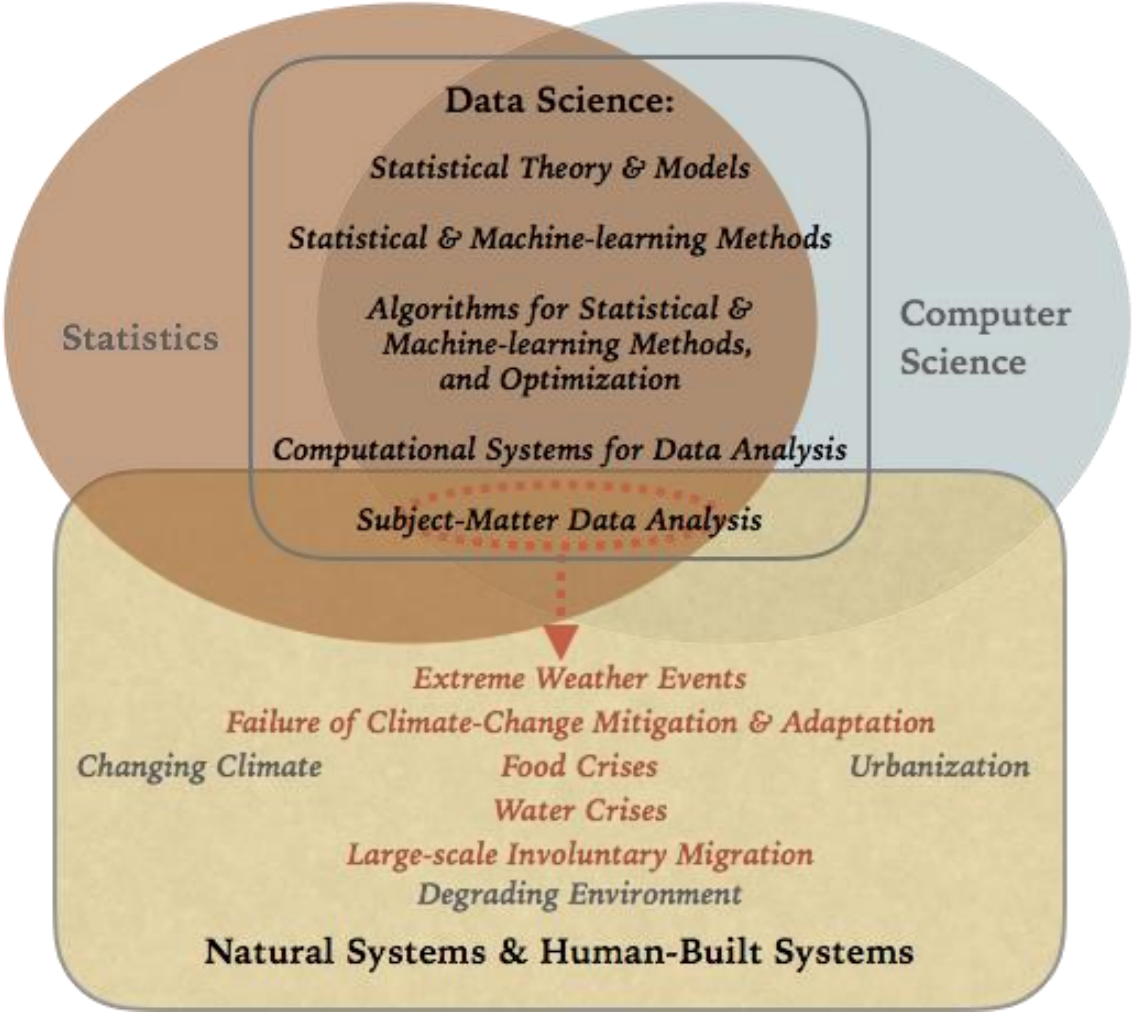
Divide & Recombine for Deep Analysis and Detailed Visualization

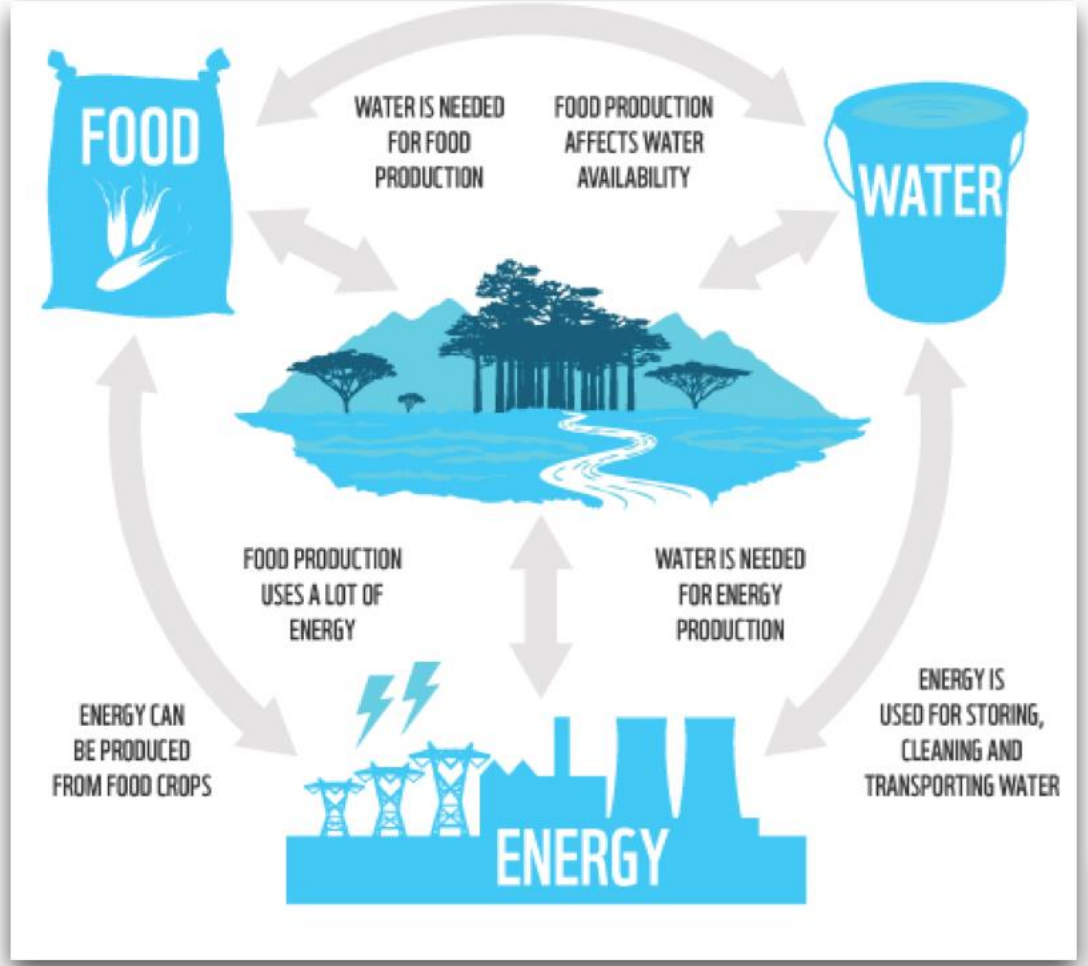
Data Science is multidisciplinary

- Data Science focuses on data analysis

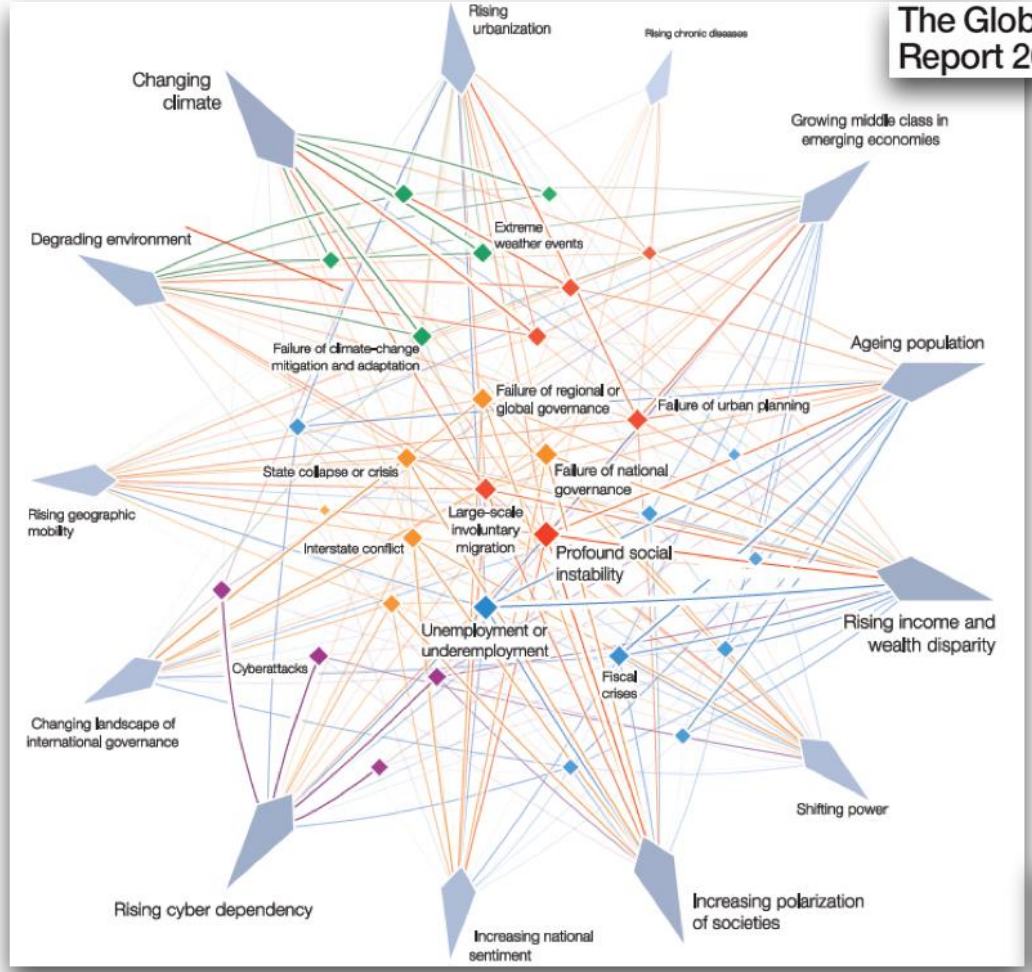
- Technical areas:
  - statistical theory and models,
  - machine learning and statistical methods,
  - data visualization methods,
  - algorithms for these methods,
  - computational environments for data analysis,
  - “Live” analyses judged by the subject matter findings, not the methodology and systems used

(Cleveland 2001, 2005, 2014; Cleveland and Hafen 2014)


Data Science:
Statistical Theory & Models
Statistical & Machine-learning Methods
Algorithms for Statistical & Machine-learning Methods, and Optimization
Computational Systems for Data Analysis
Subject-Matter Data Analysis
Statistics
Computer Science
Extreme Weather Events
Failure of Climate-Change Mitigation & Adaptation
Changing Climate
Food Crises
Urbanization
Water Crises
Large-scale Involuntary Migration
Degrading Environment
Natural Systems & Human-Built Systems

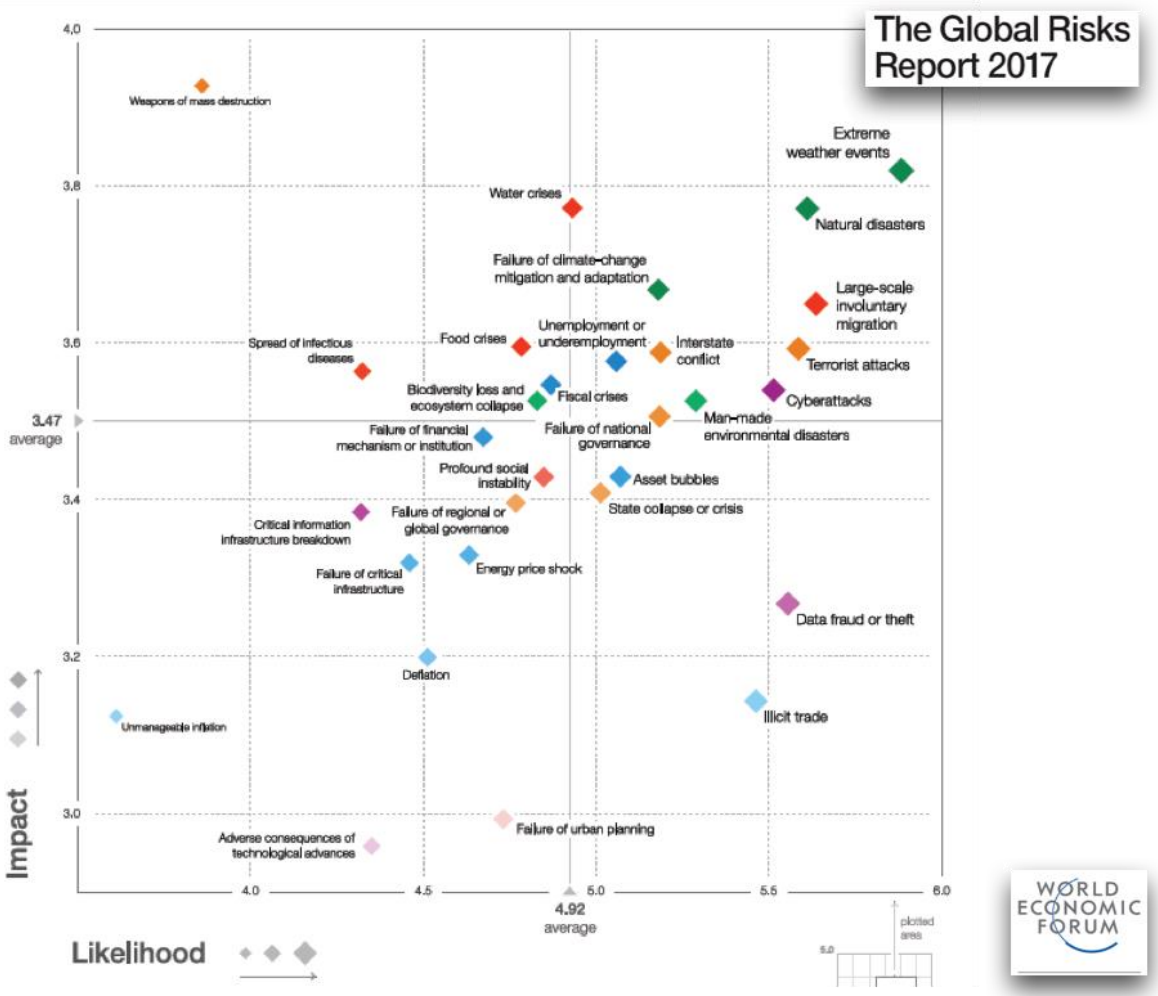

FOOD
WATER IS NEEDED FOR FOOD PRODUCTION
FOOD PRODUCTION AFFECTS WATER AVAILABILITY
WATER
FOOD PRODUCTION USES A LOT OF ENERGY
WATER IS NEEDED FOR ENERGY PRODUCTION
ENERGY CAN BE PRODUCED FROM FOOD CROPS
ENERGY IS USED FOR STORING, CLEANING AND TRANSPORTING WATER
ENERGY

The Global Risks Report 2017



WORLD ECONOMIC FORUM

The Global Risks Report 2017



WORLD ECONOMIC FORUM

Weather and Climate data-science challenges

- Massive model and sensor output
- Multiple spatial and temporal scales of Interests
- High-impact research and operation applications need data analysis
  - Analysis and Modeling of Big Data for Big Cities
  - Forecasting Extreme Event Impacts on Urban Food-Energy-Water Systems
  - Subseasonal to Seasonal (S2S) Forecasts
  - Stochastic Representation of Model Physics
  - Uncertainty quantification in Complex Models

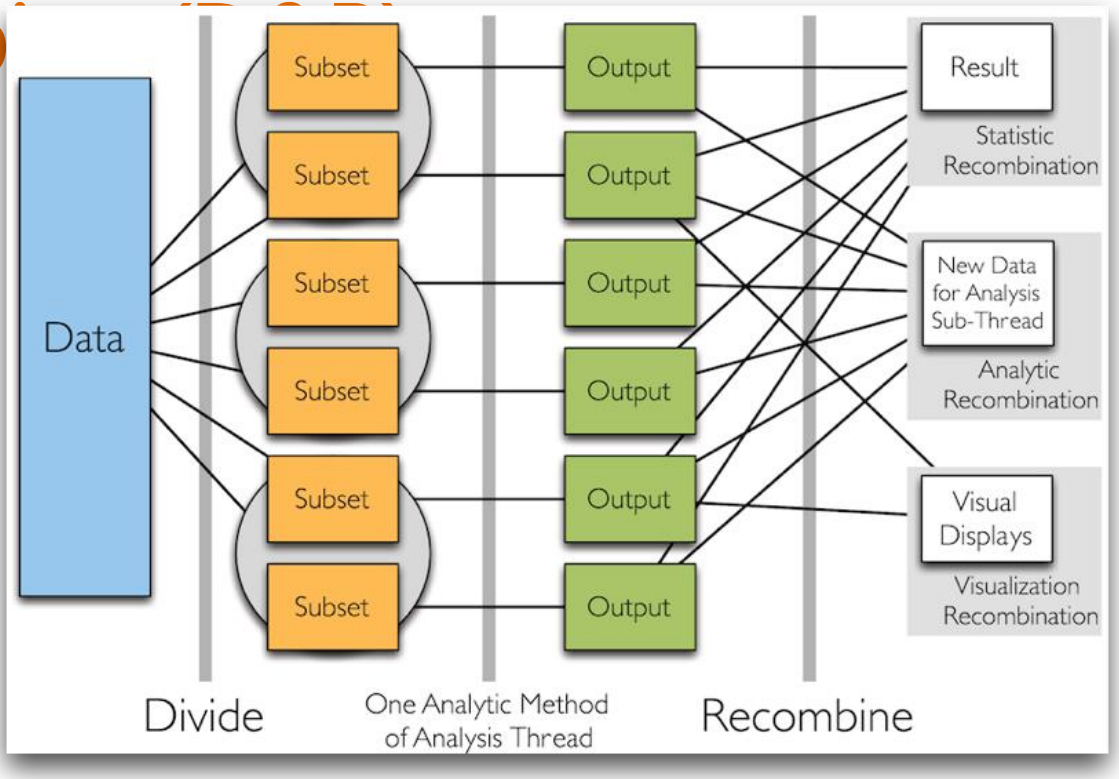
Notable Public Data Initiatives in Geosciences

- NASA Earthdata https://earthdata.nasa.gov/
- AWS https://aws.amazon.com/opendata/public-datasets/
  - Registry: https://registry.opendata.aws/
- Google Public Data https://www.google.com/publicdata/directory
- US government open data https://www.data.gov/
- Chicago City data https://data.cityofchicago.org/
- NCAR Research Data Archive https://rda.ucar.edu/
- Incorporated Research Institutions for Seismology (IRIS) https://www.iris.edu/hq/
- OneGeology http://www.onegeology.org/, part of ongoing Deep-time Digital Earth (DDE)

Analyze and Visualize Large Complex Data in R

http://deltarho.org

DeltaRho is based on Divide and Recomb



Hadoop provides a scalable back end to power the divide and recombine approach

Hadoop distributed file system (HDFS)

Parallel compute engine (Map/Reduce) and more

http://hadoop.apache.org

Current Capacity of DeltaRho at Purdue

- Access to analysis methods—thousands of methods of machine learning, statistics, and data visualization through R
- Easy programming of analyses—R-Hadoop Integrated Programming Environment protects analysts from direct programming in Hadoop
- Bridge to other Apache software ecosystem for data analysis backended by Hadoop: Dask (Python) and Storm (realtime streaming)
- Frontend interface: Jupyter Notebook (internal current) and Purdue’ s HubZero (external future, for example http://mygeohub.org)
- High computational performance—increase dramatically with data size and analytic computational complexity.
- Deep analysis of the data—analysis of the detailed data at their finest granularity

Introduction to Hadoop

White, Tom, 2015, Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale. 4th Ed., O’ Reilly Media, 756pp.

What is Hadoop?

- The Apache Hadoop project (https://hadoop.apache.org/) develops open-source software for reliable, scalable, distributed computing.
- Apache Hadoop software library is a framework for storing data and running applications on clusters of commodity hardware
  - Rather than rely on hardware to deliver high-availability, the library is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.
  - Massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Modules of Hadoop

- **Hadoop Common:** The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN:** A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce:** A YARN-based system for parallel processing of large data sets.
- **Hadoop Ozone:** An object store for Hadoop.
- **Hadoop Submarine:** A machine learning engine for Hadoop.

Challenges of Big Data Storage and Analysis

Challenge: Although the storage capacities of hard drives (HD) have increased massively over the years, access speeds -- the rate at which data can be read from drives -- have not kept up.

- 1990 HD: 1370 MB storage, 4.4 MB/s transfer speed, 5 mins to read full disk
- 2010 HD: 1TB storage, 100 MB/s transfer speed, 2.5 hr to read full disk
  - Writing is even slower

Obvious Solution: Read from multiple disks at once

More issues

There is more to being able to read and write data in parallel or to form multiple disks:

Problem 1: Hardware failure

Solution:

- Common solution: replication: redundant copies of data
- RAID (Redundant Array of Independent Disks), Hadoop Distributed File System (HDFS)

More issues

Problem 2: Analysis tasks need to be able to combine the data in some way; data read from one disk may need to be combined with data from any of the other 99 disks

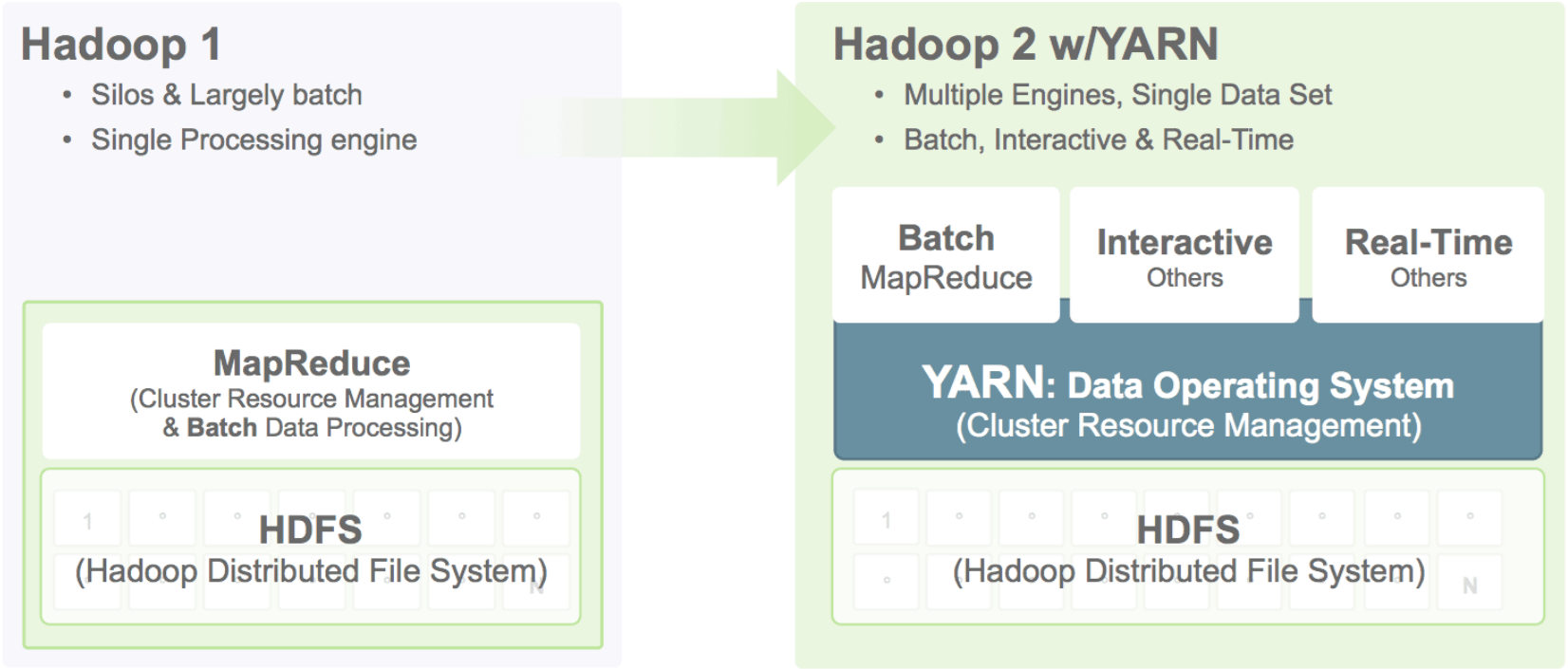
Solution:

- Hadoop’ s MapReduce provides a programming model that abstracts the problem from disk reads and writes, transforming it into a computation over sets of keys and values

Hadoop’s solutions

- Hadoop provides a reliable, scalable platform for storage and analysis.
- It runs on commodity hardware and is open source, so Hadoop is affordable.

Hadoop 1 vs. Hadoop 2



https://hortonworks.com/blog/office-hours-qa-on-yarn-in-hadoop-2/

Hadoop's MapReduce enables query big data

- The premise of MapReduce is that the entire dataset or at least a good portion of it can be processed for each query.
- MapReduce is a batch query processor, and the ability to run an ad hoc query against your whole dataset and get the results in a reasonable time is transformative.
- A batch processing system is not suitable for interactive analysis - you can't run a query and get results back in a few seconds or less.
  - MapReduce is best for offline use

Beyond MapReduce

- Hadoop 2 provides different processing patterns with YARN (Yet Another Resource Negotiator)
- YARN is a cluster resource management system, which allows any distributed program (not just MapReduce) to run on data in a Hadoop cluster: Interactive, Iterative, Streaming,...

Comparison with SQL Database Management Systems

SQL: Relational Database, with ACID properties

- **Atomicity** requires a transaction to execute completely or not at all.
- **Consistency** requires that when a transaction has been committed, the data must conform to the database schema.
- **Isolation** requires that concurrent transactions execute separately from each other.
- **Durability** requires the ability to recover from an unexpected system failure or power outage to the last known state.

https://aws.amazon.com/sql/

Hadoop uses key-value databases

NO-SQL: Flexibility, Scalability, High-performance, High-functional

- **Flexibility:** NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes NoSQL databases ideal for semi-structured and unstructured data.
- **Scalability:** NoSQL databases are generally designed to scale out by using distributed clusters of hardware instead of scaling up by adding expensive and robust servers. Some cloud providers handle these operations behind-the-scenes as a fully managed service.
- **High-performance:** NoSQL database are optimized for specific data models (such as document, key-value, and graph) and access patterns that enable higher performance than trying to accomplish similar functionality with relational databases.
- **Highly functional:** NoSQL databases provide highly functional APIs and data types that are purposely built for each of their respective data models.

https://aws.amazon.com/nosql/

Comparison between Hadoop with Grid Computing

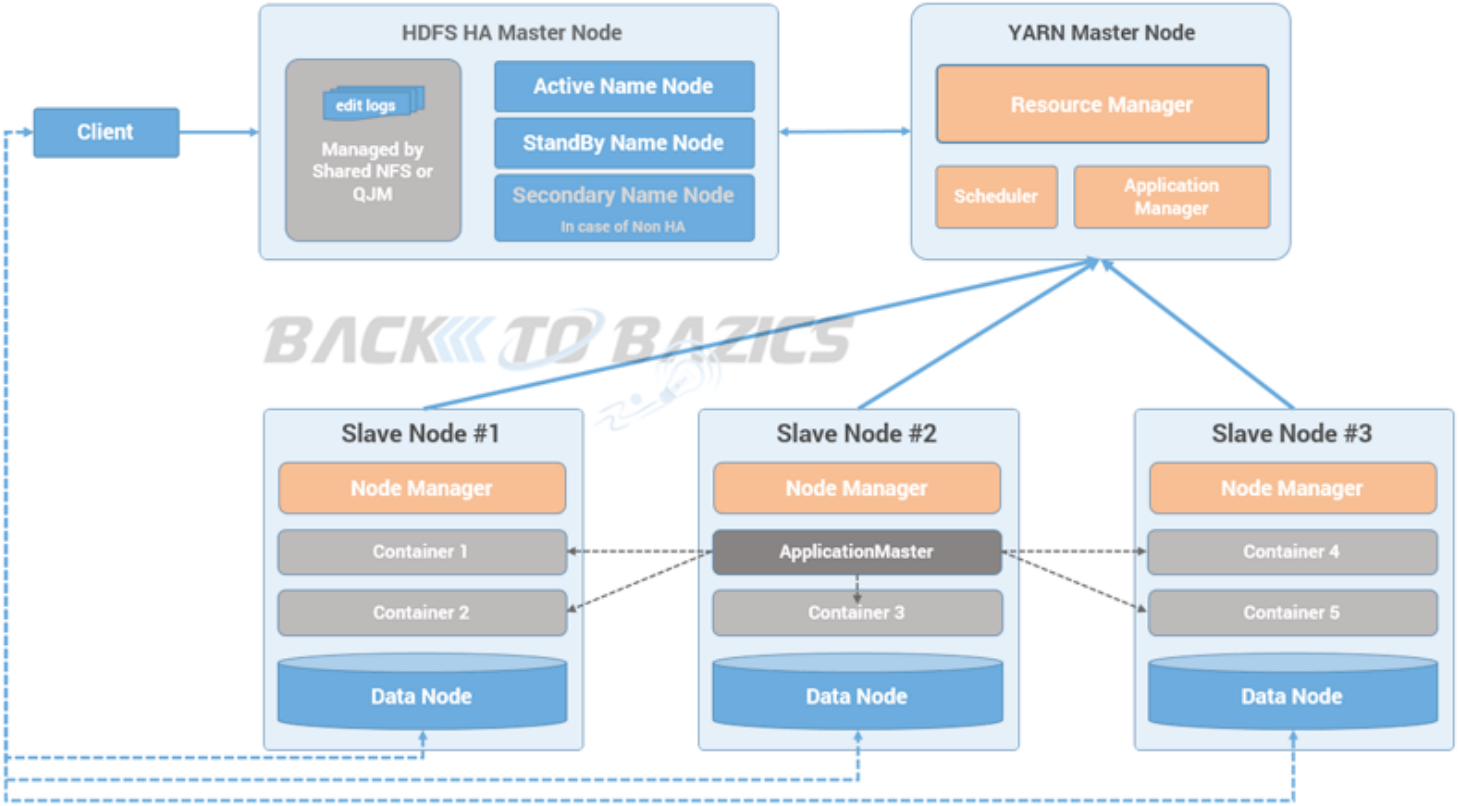
Traditional High Performance Computing and Grid Computing

- MPI: Message Passing Interface
- Distribute the work across a cluster of machines, which access a shared filesystem, hosted by a storage area network (SAN)
- Works well for predominantly compute-intensive jobs, problem with nodes needing to access large data volumes (100s of GB, for example); the network bandwidth is the bottleneck and compute nodes become idle
- MPI gives great control to programmers, good or bad

Hadoop

- Co-locate the data with the compute nodes, so data access is fast because it is local
  - Data locality, the heart of data processing in Hadoop
- Hadoop conserves network bandwidth by explicitly modeling network topology
- This arrangement does not preclude high-CPU analysis in Hadoop
- Hadoop programmers think in terms of data model (such as key-value pairs for MapReduce), while data flow remains implicit
- Hadoop’ s “shared-nothing” architecture allows handling of partial failure

Hadoop 2 Architecture



Shared-Nothing architectures

- In distributed systems, this is an architecture where each node is completely independent of other nodes in the system.
- There are no shared resources that can become bottlenecks.

The lack of shared resources refers to

- Lack of physical resources such as memory, disks, and CPUs
  - Instead of using centralized storage, Hadoop’ s processing framework uses the distributed HDFS storage.
- Lack of shared data
  - each node is processing a distinct subset of the data and there’ s no need to manage access to shared data.

Shared-Nothing architectures are very scalable

- Because there are no shared resources, addition of nodes adds resources to the system and does not introduce further contention.

These architectures are also fault-tolerant:

- Each node is independent, so there are no single points of failure, and the system can quickly recover from a failure of an individual node.

Hadoop2: Computation

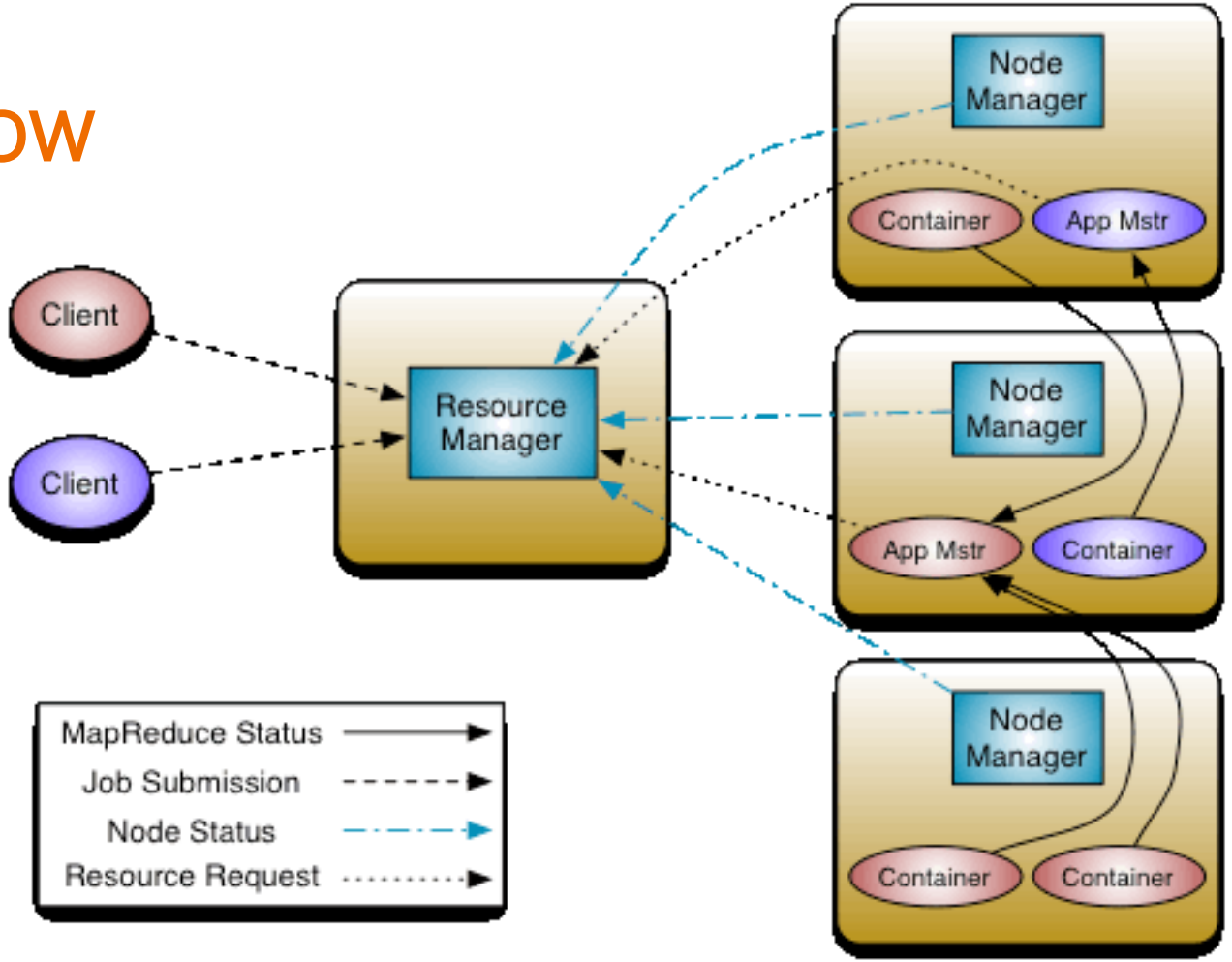
Resource Manager(RM) node:

- Jobs are submitted to Resource Manager (asks for job ID, checks the output path ...)
- Applications Manager(AsM) - manages running jobs in the cluster
- Scheduler - manages and enforces the resource scheduling policy in the cluster

NodeManager(NM) node:

- ApplicationMaster(AM) - A per-job master that manages the application’ s life cycle jobs on the cluster
- Container, it is an Unix process which is assigned with specific amount of core and memory

Workflow


Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

Workflow: Procedure of submitting a job

1. Client ⟷ Applications Manager in Resource Manager
2. Resource Manager(RM) ⟷ Node Manager: finds an available container for running the ApplicationMaster
3. ApplicationMaster ⟷ ResourceManager: ask for containers for all map and reduce tasks.
4. ApplicationMaster ⟷ NodeManager: starts the containers and run JVMs

Simulating a MapReduce job

What is MapReduce?

- It is a computation engine in Hadoop ecosystem
- A programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster
- MapReduce programs are embarrassingly parallel, putting very large-scale data analysis into the hands of anyone with enough machines at their disposal
- Hadoop can run MapReduce programs written in various languages: Java, Ruby, Python, R,...

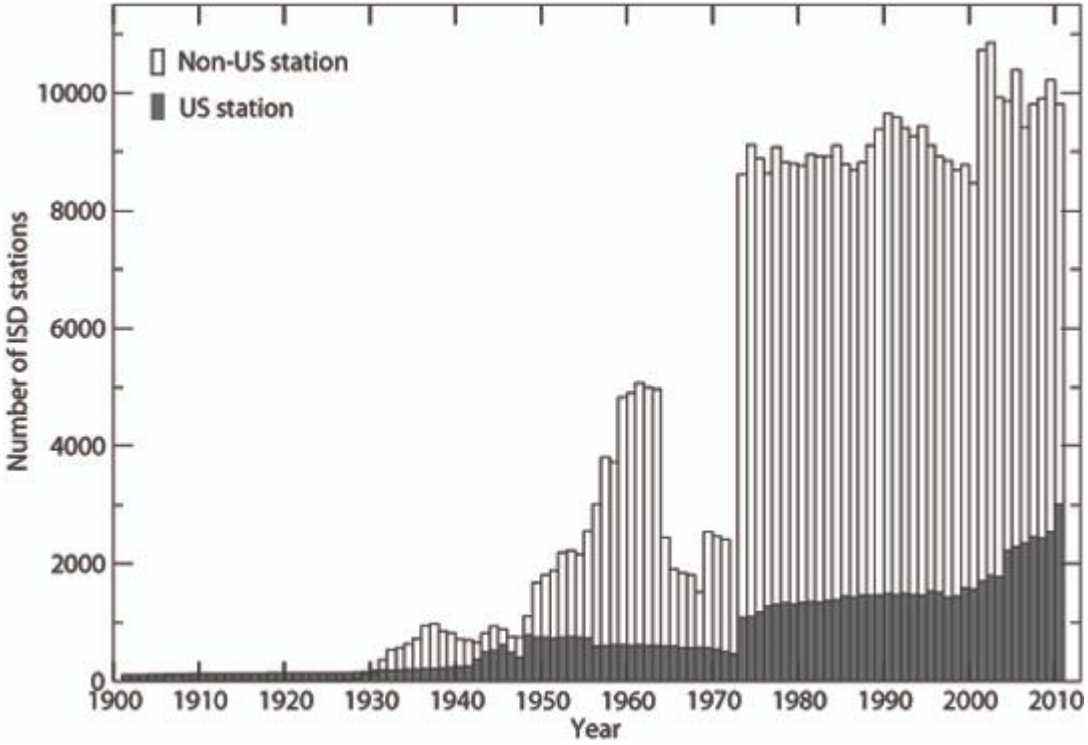
Example: Weather Data

- National Climate Data Center Integrated Surface Data
  https://www1.ncdc.noaa.gov/pub/data/noaa
  - Hourly Surface Station Data Worldwide from 1901 to 2019 (Feb)
    https://www1.ncdc.noaa.gov/pub/data/noaa/readme.txt
  - Station Inventory https://www1.ncdc.noaa.gov/pub/data/noaa/isd-inventory.txt

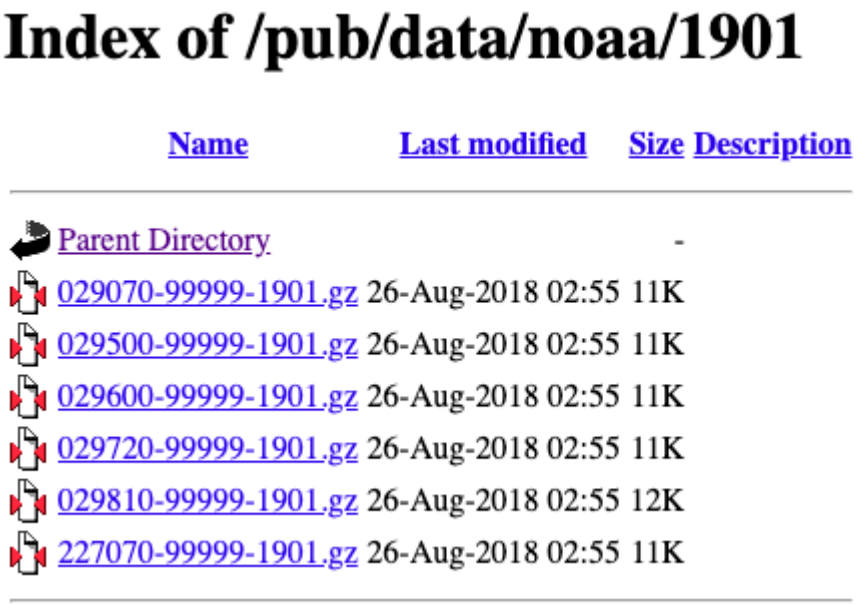
*** FEDERAL CLIMATE COMPLEX INTEGRATED SURFACE DATA INVENTORY ***

THIS INVENTORY SHOWS THE NUMBER OF WEATHER OBSERVATIONS BY STATION-YEAR-MONTH FOR BEGINNING OF RECORD THROUGH FEBRUARY 2019. THE DATABASE CONTINUES TO BE UPDATED AND ENHANCED, AND THIS INVENTORY WILL BE UPDATED ON A REGULAR BASIS.

USAF	WBAN	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
007018	99999	2011	0	0	2104	2797	2543	2614	382	0	0	0	0	0
007018	99999	2013	0	0	0	0	0	0	710	0	0	0	0	0
007026	99999	2012	0	0	0	0	0	0	367	0	0	0	0	7
007026	99999	2014	0	0	0	0	0	0	180	0	4	0	552	0
007026	99999	2016	0	0	0	0	0	794	0	0	0	0	0	0


Non-US station
US station
Number of ISD stations
0
2000
4000
6000
8000
10000
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
Year

Data files: https://www1.ncdc.noaa.gov/pub/data/noaa/1901/



Index of /pub/data/noaa/1901

Name	Last modified	Size	Description
Parent Directory		-	
029070-99999-1901.gz	26-Aug-2018 02:55	11K	
029500-99999-1901.gz	26-Aug-2018 02:55	11K	
029600-99999-1901.gz	26-Aug-2018 02:55	11K	
029720-99999-1901.gz	26-Aug-2018 02:55	11K	
029810-99999-1901.gz	26-Aug-2018 02:55	12K	
227070-99999-1901.gz	26-Aug-2018 02:55	11K	

Organized by weather station and year.
The whole dataset is a large number of relatively small files, but can be pre-processed into small number of relatively large files.

Surface hourly abbreviated format

https://www1.ncdc.noaa.gov/pub/data/noaa/ish-abbreviated.txt

06/11/2012

SURFACE HOURLY ABBREVIATED FORMAT

ONE HEADER RECORD FOLLOWED BY DATA RECORDS:

COLUMN	DATA DESCRIPTION
01-06	USAF = AIR FORCE CATALOG STATION NUMBER
08-12	WBAN = NCDC WBAN NUMBER
14-25	YR--MODAHRMN = YEAR-MONTH-DAY-HOUR-MINUTE IN GREENWICH MEAN TIME (GMT)
27-29	DIR = WIND DIRECTION IN COMPASS DEGREES, 990 = VARIABLE, REPORTED AS '***' WHEN AIR IS CALM (SPD WILL THEN BE 000)
31-37	SPD & GUS = WIND SPEED & GUST IN MILES PER HOUR
39-41	CLG = CLOUD CEILING--LOWEST OPAQUE LAYER WITH 5/8 OR GREATER COVERAGE, IN HUNDREDS OF FEET, 722 = UNLIMITED
43-45	SKC = SKY COVER -- CLR-CLEAR, SCT-SCATTERED-1/8 TO 4/8, BKN-BROKEN-5/8 TO 7/8, OVC-OVERCAST, OBS-OBSCURED, POB-PARTIAL OBSCURATION
47-47	L = LOW CLOUD TYPE, SEE BELOW
49-49	M = MIDDLE CLOUD TYPE, SEE BELOW
51-51	H = HIGH CLOUD TYPE, SEE BELOW
53-56	VSB = VISIBILITY IN STATUTE MILES TO NEAREST TENTH NOTE: FOR SOME STATIONS, VISIBILITY IS REPORTED ONLY UP TO A

Find the maximum temperature

```
0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
```

“Simulating” a MapReduce job

- MapReduce works by breaking the processing into two phases: the map phase and the reduce phase.
- Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.
- The programmer also specifies two functions: the map function and the reduce function.

3-step workflow

- Map: A map function is applied to each input key-value pair, which does some user-defined processing and emits new key-value pairs to intermediate storage to be processed by the reduce.
- Shuffle/Sort: The map output values are collected for each unique map output key and passed to a reduce function.
- Reduce: A reduce function is applied in parallel to all values corresponding to each unique map output key and emits output key-value pairs.

Map or Reduce?

- The input to the map phase is the raw weather data.
  - each line in the dataset is a text value.
  - These lines are presented to the map function as the key-value pairs
  - The key is the offset of the beginning of the line from the beginning of the file, but as we have no need for this, we can ignore it.
  - The map function is also a good place to drop bad records

```
(0,
0067011990999991950051507004...9999999N9+00001+99999999999...)
(106,
0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-
00111+99999999999...)
(318,
0043012650999991949032412004...0500001N9+01111+99999999999...)
```

Map or Reduce?

This function extracts the year and the air temperature (indicated in bold text previously), and emits them as its output (the temperature values have been interpreted as integers).

Note these new key-value pairs.

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
```

Map or Reduce?

This process sorts and groups the key-value pairs by key. So, each year appears with a list of all its air temperature readings.

```
(1949, [111, 78])
(1950, [0, 22, -11])
```

Map or Reduce?

This function iterates through the list and pick up the maximum reading:

```
(1949, 111)
(1950, 22)
```

MapReduce workflow example:

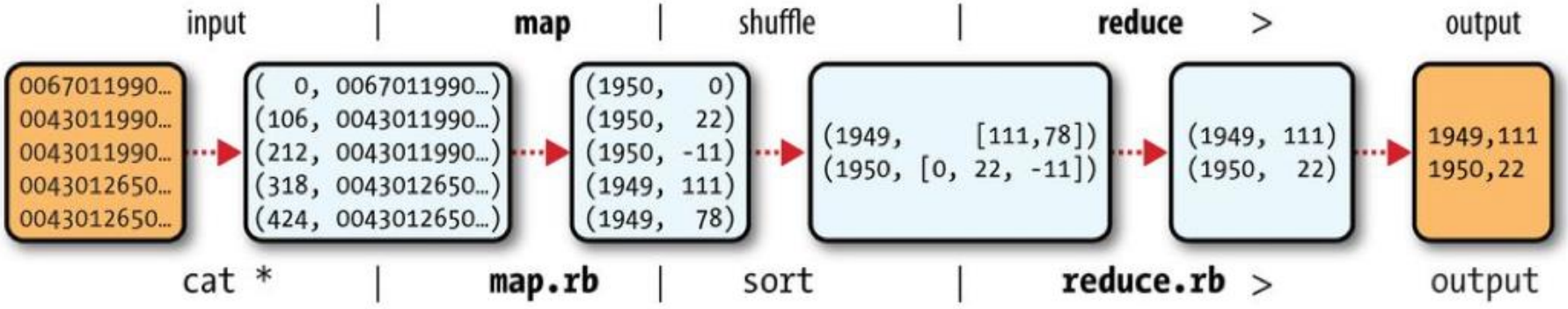


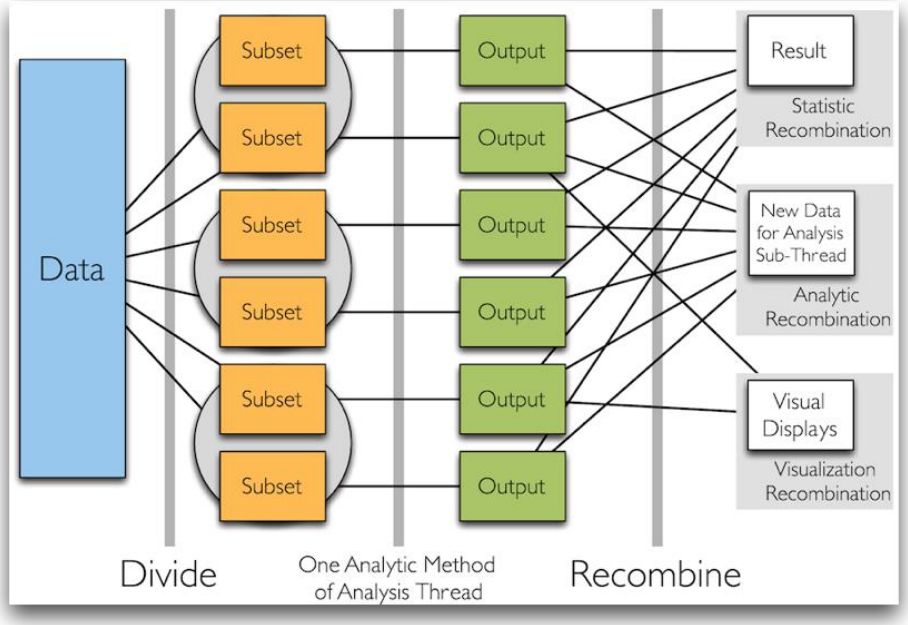
Figure 2-1. MapReduce logical data flow

MapReduce and D&R

D&R for Large Complex Data

Divide and Recombine approach

- Divide the data into subsets
- Apply statistical methods to each subset independently
- Recombine the results in a statistically valid way



RHIPE: R and Hadoop Integrated Programming Environment

Backgroud

Hadoop is written in Java

R is used by most statisticians, with a vast number of analysis packages

What is RHIPE?

- RHIPE is the R and Hadoop Integrated Programming Environment
- Manages communication between the R user and Hadoop to carry out D&R data analysis
- Allows an analyst to run Hadoop MapReduce jobs wholly from within R.
- First developed by Saptarshi Guha as part of his PhD thesis in the Purdue Statistics Department

RHIPE Workflow

- User writes R code for D&R comupations
- R commands are passed to RHIPE R commands that communicate with Hadoop
- R objects are distributed by Hadoop across the nodes of the cluster
- Hadoop implements computations
- Outputs from D&R computations are written to HDFS as R objects
- Documentations:

  http://deltarho.org/docs-RHIPE/index.html
  http://deltarho.org/docs-RHIPE/functionref.html

Programming MapReduce

Structure of a MapReduce Program

1. Map Expression: Mapper, a map function

   Expression which will be evaluated for each map task

1. Reduce Expression: Reducer, a reduce function

   Expression which will be evaluated for each reduce task

1. Execution function: some code to run a MapReduce job

   Specifies all parameters needed for a MapReduce job, and triggers the MapReduce job in Rhipe if programmed with Rhipe

A map or a reduce function (a mapper or a reducer) contains following four formal type parameters specifying

1. Input keys
2. Input values
3. Output keys
4. Output values

In the reducer, the input keys and values match the output keys and values of the mapper

A MapReduce Job

- A unit of work that the client wants to be performed
- It is composed of

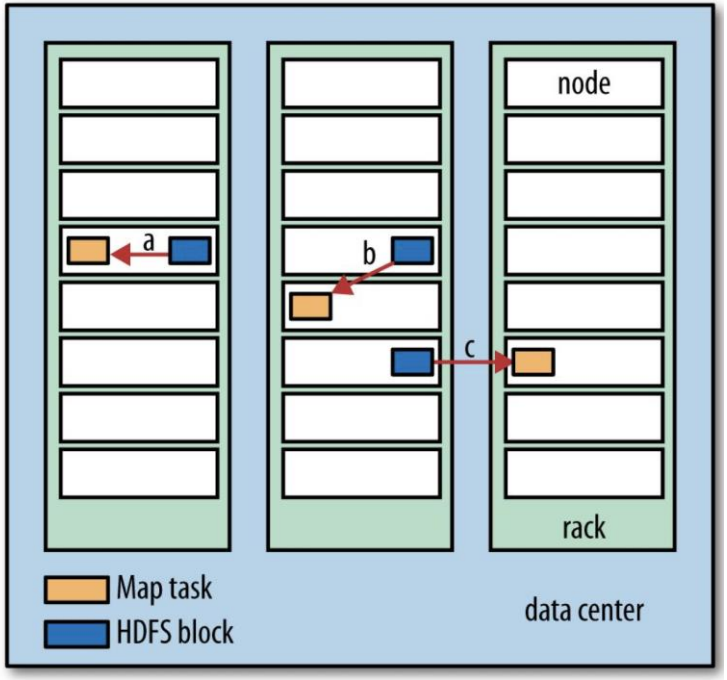
1. Input Data: a single file, a directory (all files in the directory), or a file pattern
2. MapReduce Program
3. Configuration Information

Data Flow of a MapReduce job

- Hadoop runs the MapReduce job by dividing it into tasks: map tasks and reduce tasks
- The tasks are scheduled using YARN and run on nodes in the cluster. If a task fails, it will be automatically rescheduled to run on a different node. -
- Hadoop divides the input to a MapReduce job into fixed-size pieces called input splits, or just splits
- Hadoop creates one map task for each split, which runs the user-defined map function for each record in the split.
    - Map tasks write their output to the local disk, not to HDFS, because it’ s considered intermediate/transient

Hadoop does its best to run the map task on a node where the input data resides in HDFS, because it doesn't use valuable cluster bandwidth (data locality optimization).

Data-local (a), rack-local (b), and off-rack (c) map tasks



Question: what's the optimal split size?

- Reduce tasks don’ t have the advantage of data locality
  - the input to a single reduce task is normally the output from all mappers.
  - Therefore, the sorted map outputs have to be transferred across the network to the node where the reduce task is running, where they are merged and then passed to the user-defined reduce function.
  - The output of the reduce is normally stored in HDFS for reliability.

.

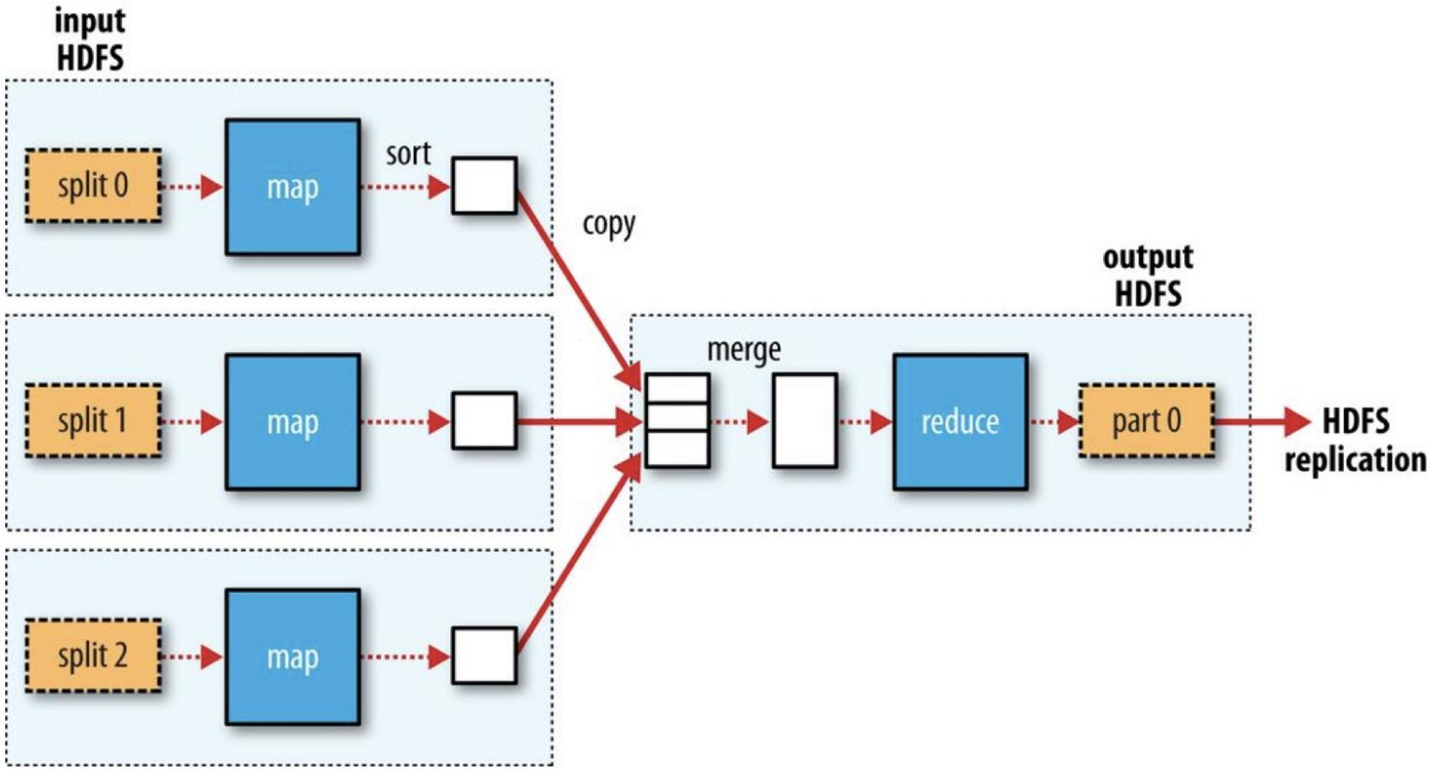


Figure 2-3. MapReduce data flow with a single reduce task

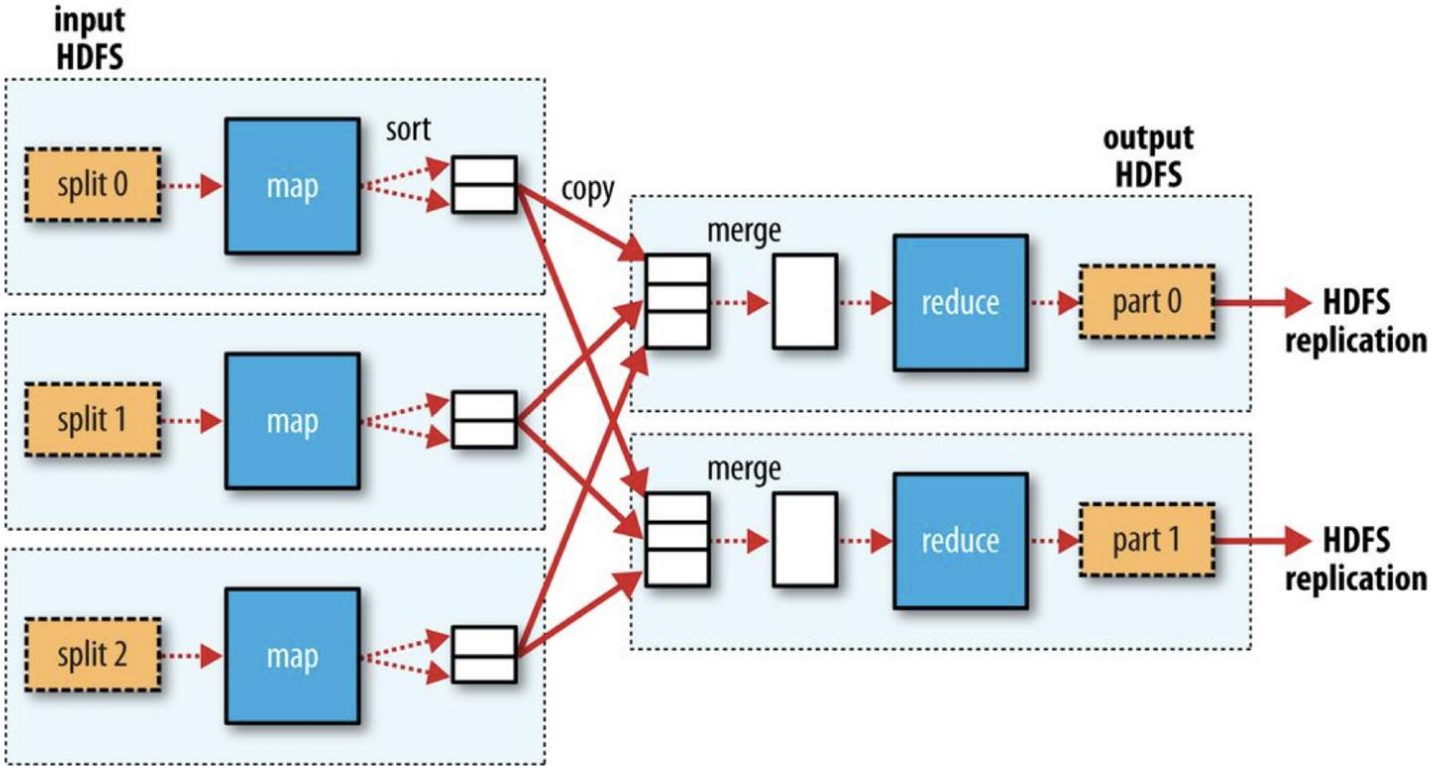


Figure 2-4. MapReduce data flow with multiple reduce tasks

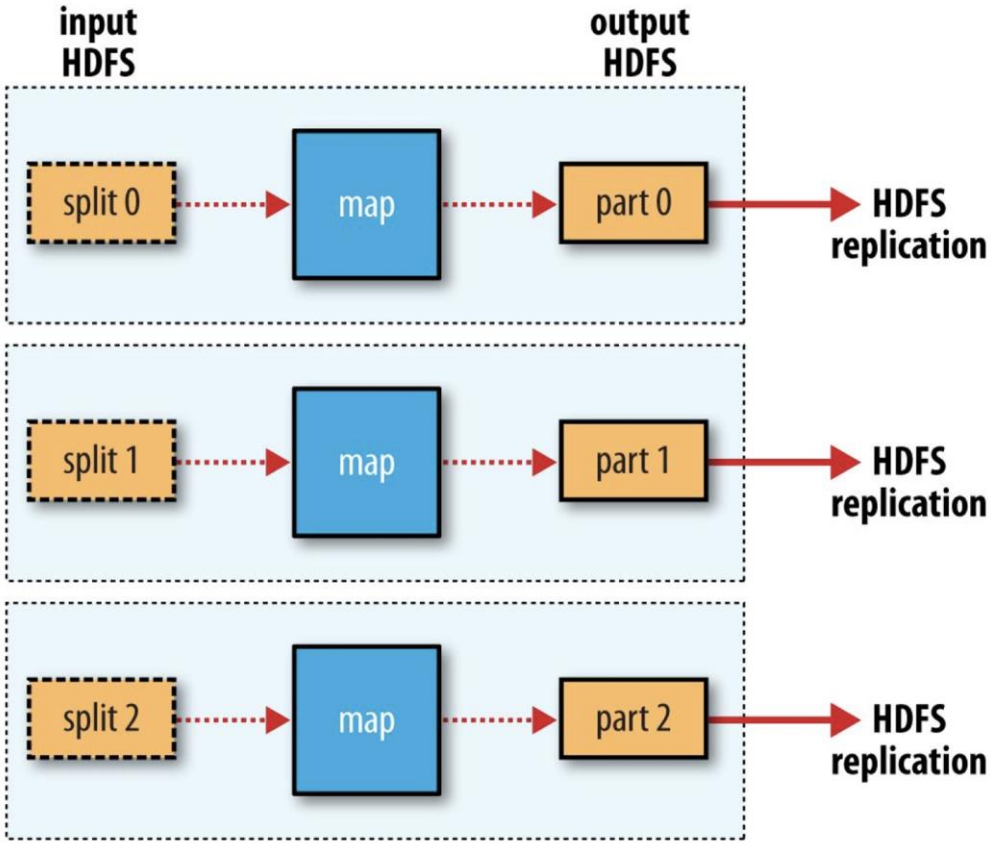


Figure 2-5. MapReduce data flow with no reduce tasks