Datenbanksysteme II: Big Data

Ulf Leser

Content of this Lecture

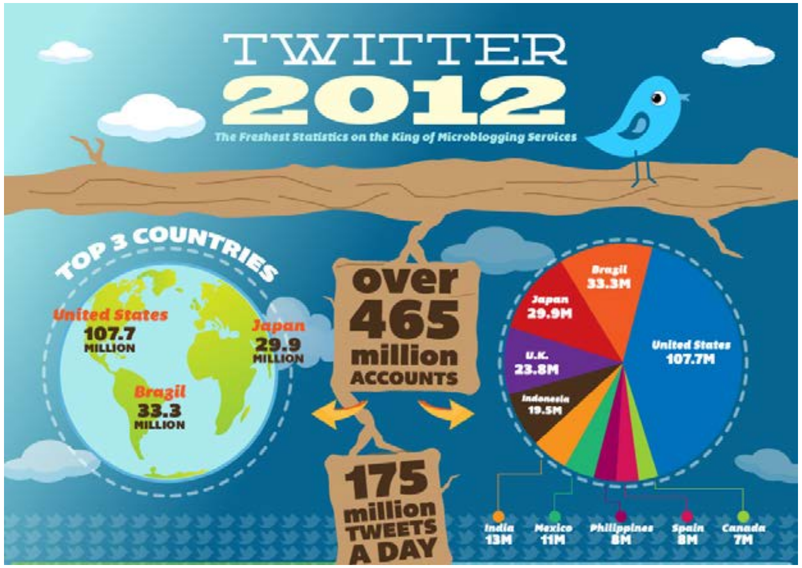
- Big Data Introduction
- Map Reduce Programming Paradigm
- Parallel DBMS or MapReduce?
- Extensions to MR

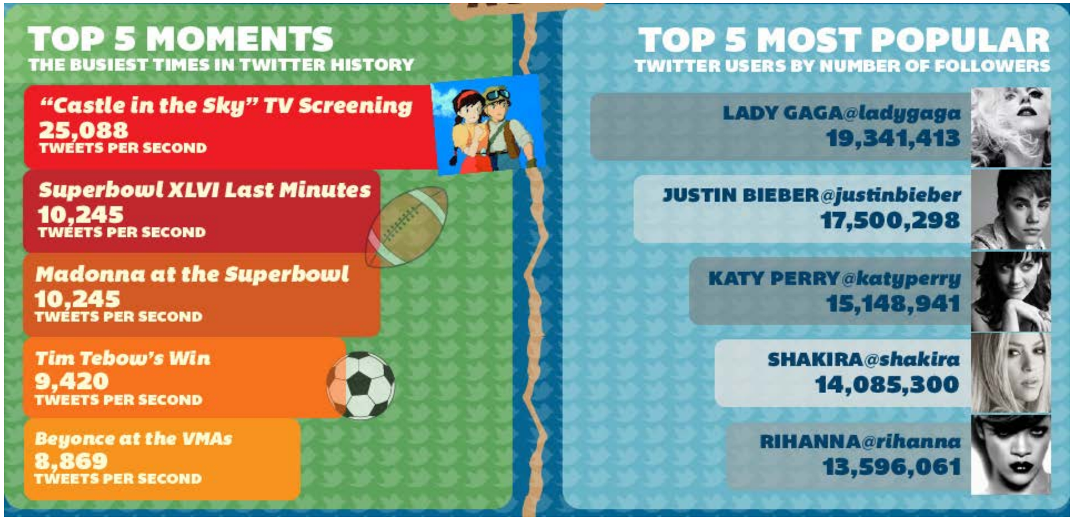
- [based on slides by Astrid Rheinländer, 2012]

What is Big Data?

- A buzz word
- "Collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications" [http://en.wikipedia.org/wiki/Big_data]
  - Terabytes / petabytes
  - Near future: exabytes
- Challenges
  - Storage
  - Analysis
  - Search

Example: Twitter



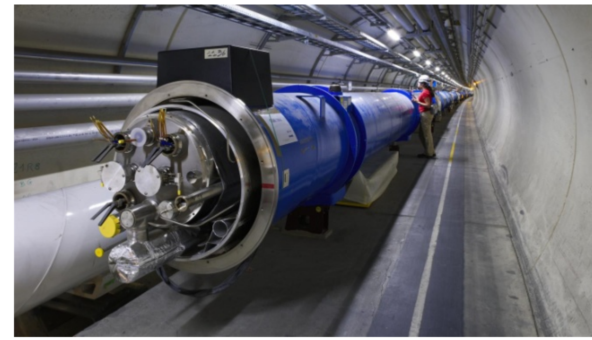


[All images: http://infographiclabs.com/news/twitter-2012/]

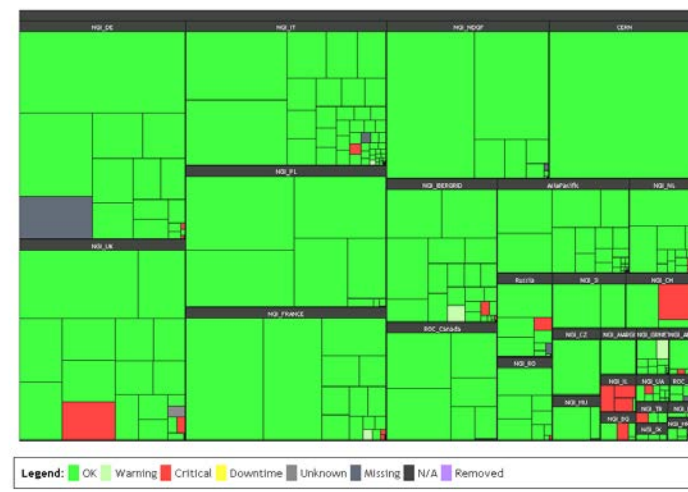
- 2012: 63 Billion tweets, 8.5 TB data
  - 2016: ~200 Billion tweets per year
- Business: Sell access to content
- Analysis: Find emerging trends, sentiment analysis, ...

Example: Cern, LHC

- 2012: LHC experiments generated 22PB of data
  - ~99% are filtered right after creation
- Analysis runs on supercomputer: 1000+ processors
- Data stored & processed in LHC Computing Grid
  - 150 data & compute centers around the world
  - Heterogeneous architecture



[http://lhcathome.web.cern.ch]

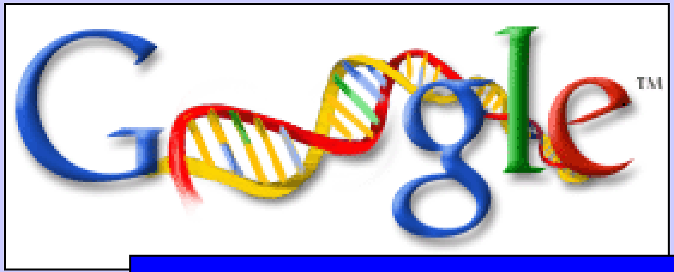


[http://grid-monitoring.cern.ch/myegi/gridmap/]

Will Computers Crash Genomics?

Pennisi, E. (2011). *Will Computers Crash Genomics?* Science, 331(6018), 666–668.

Fast Development



Double

uenced

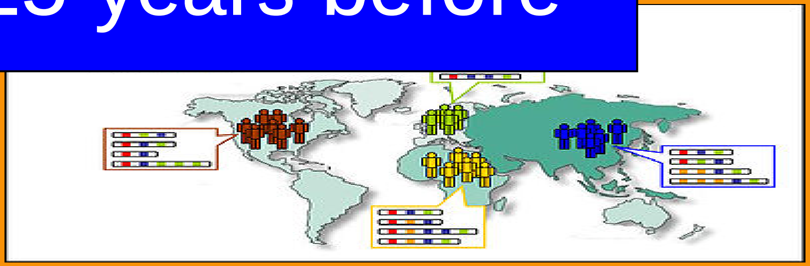
n USD

1000GP releases more data in first 6 months than EMBL collected in the 25 years before

2008

Genome of J. Watson finished

4 Months, 1.5 Million USD



2010

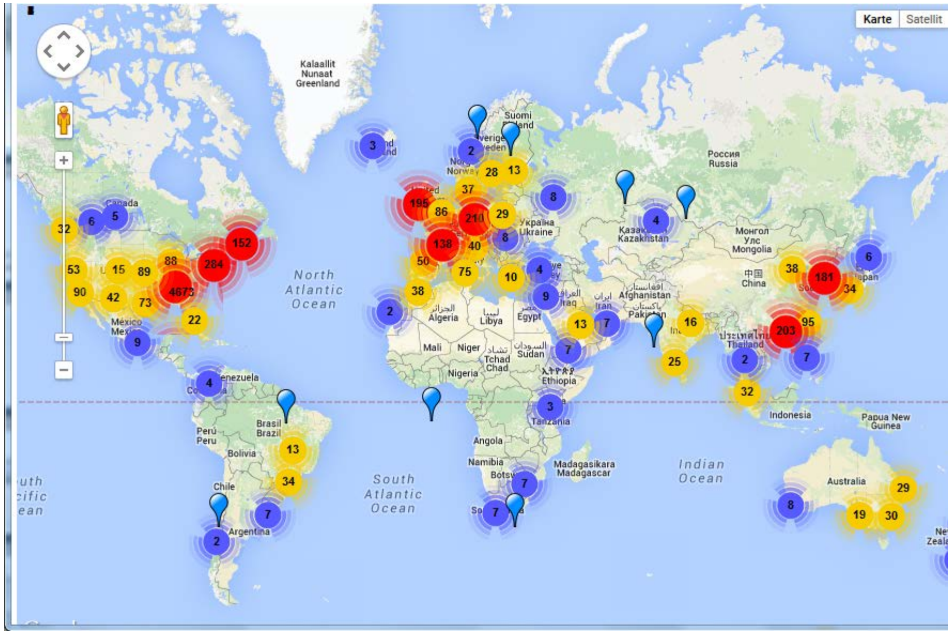
1000 Genomes Project

Next Generation Sequencing

- New generation of sequencers since ~2005
  - Illumina, Solexa, 454, Solid, ...
- Much higher throughput
  - ~15 TB raw data in 3-5 days
  - ~600 GB processed data/week
  - Cost for sequencing a genome down to ~2.000 USD
- 3rd generation sequencers
  - Single molecule sequencing
  - A (human) genome in a day
  - Sequence every human
  - Sequence different cells in every human

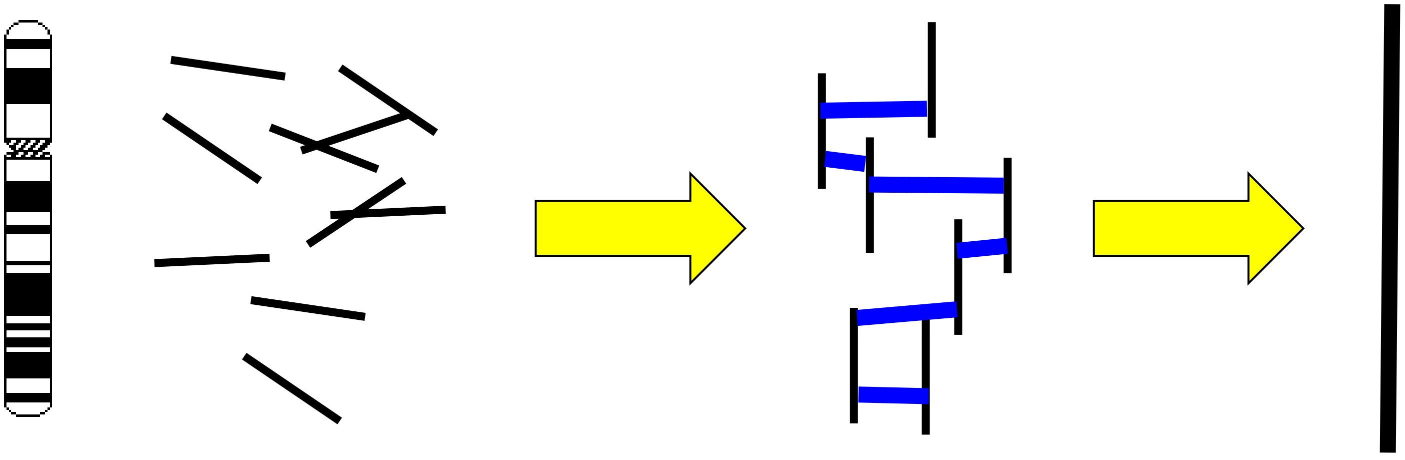
Illumina HiSeq 2000. DNAVision

Sequencing becomes a commodity

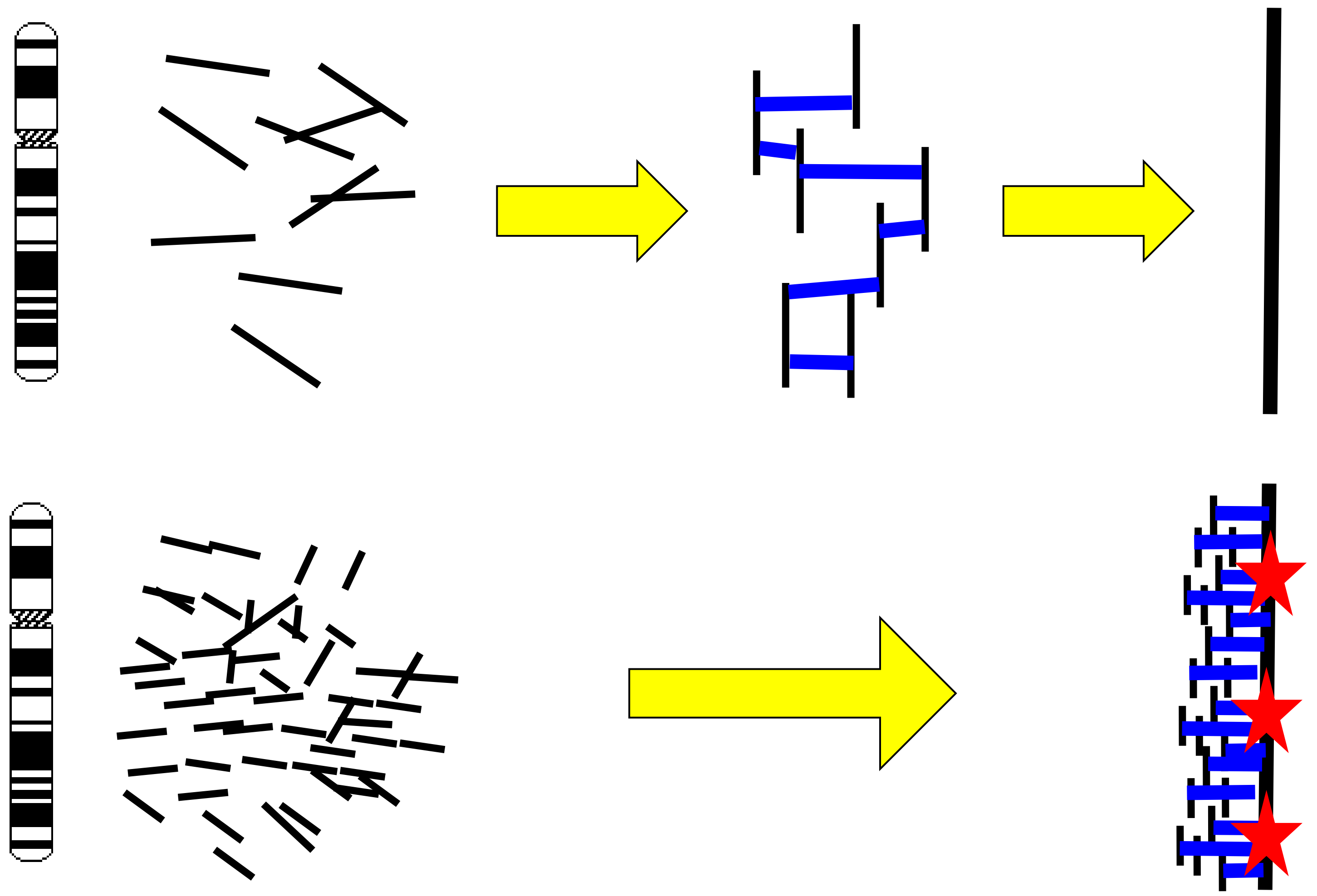


- Sequencing dozens or hundreds of genomes is feasible (now!) for any mid-size research projects

Old Task: Genome Assembly

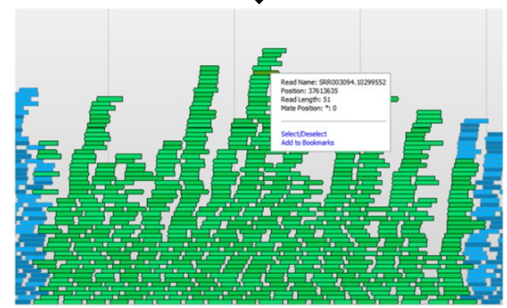


New Task: Read Mapping & SNV Detection

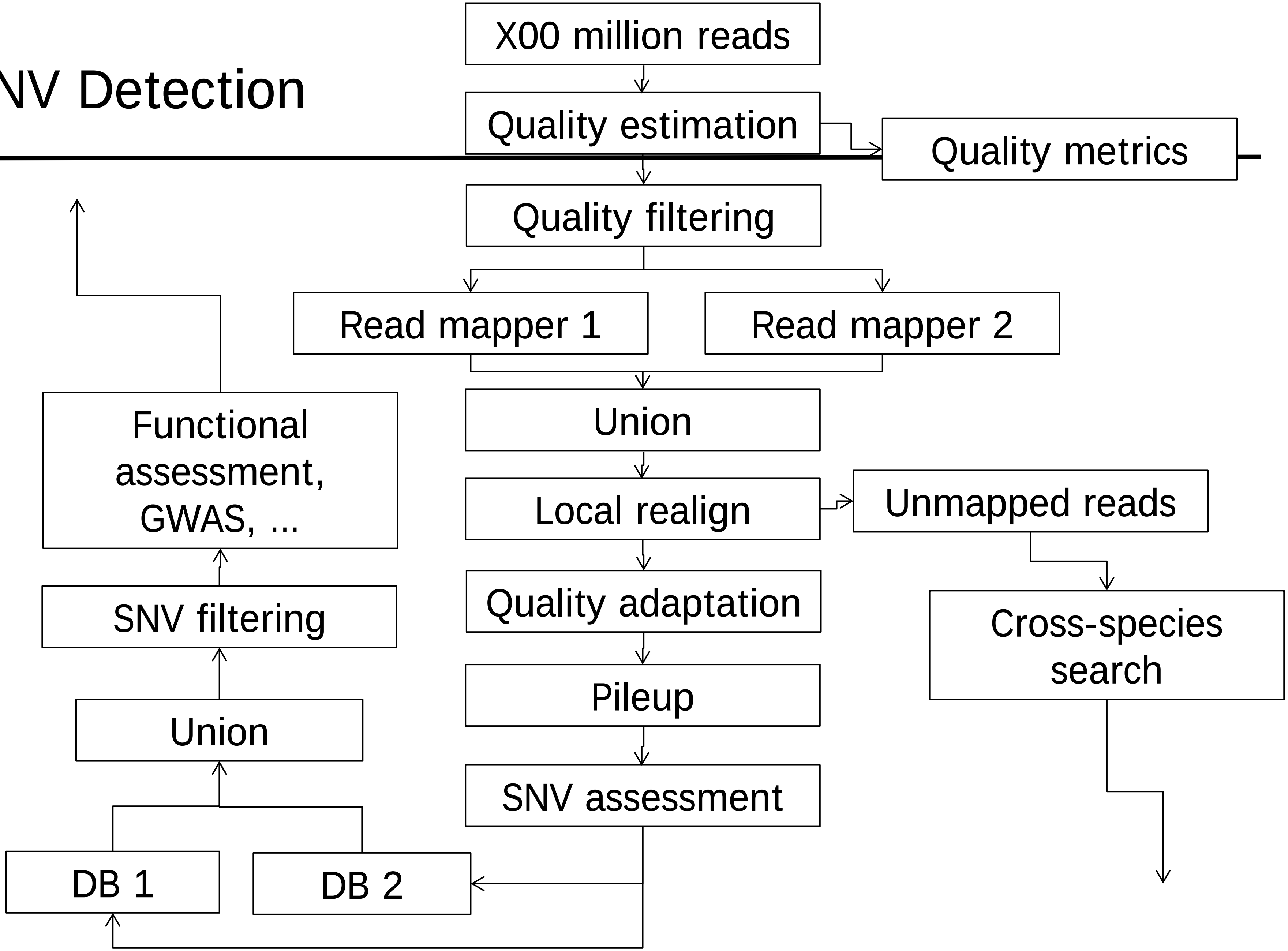


Example: SNV Detection

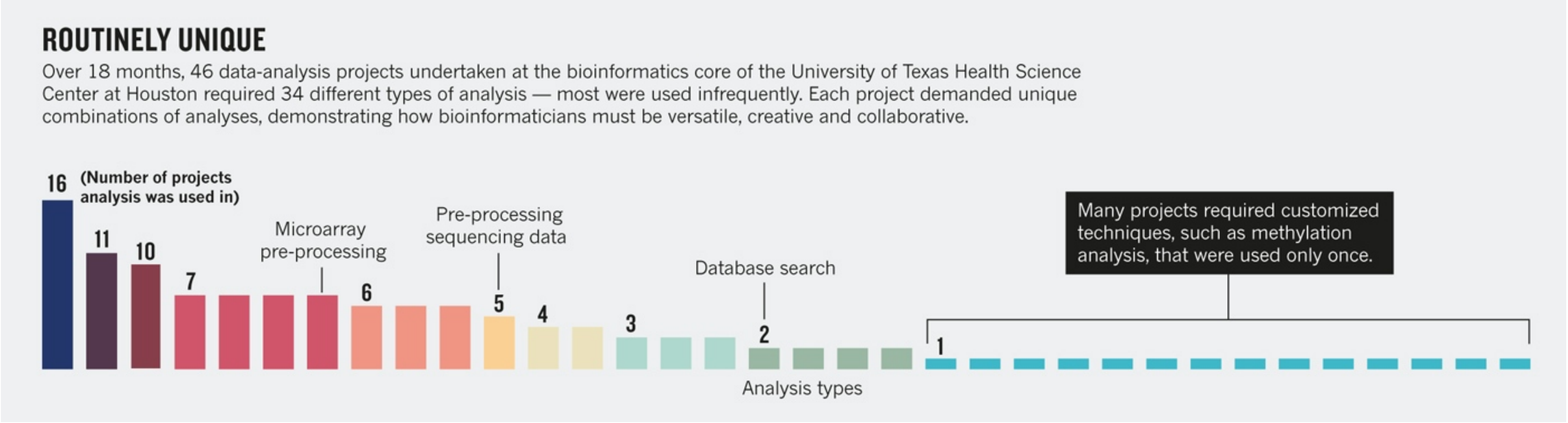
DNA/RNA



SNV Detection



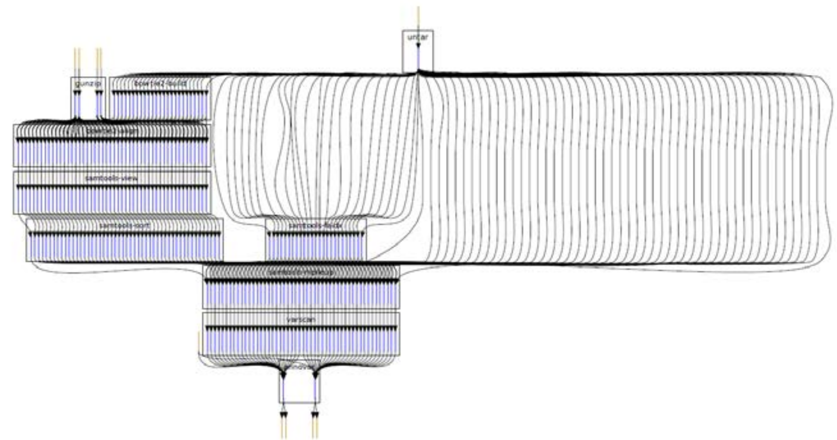
All Analytics is the Same?



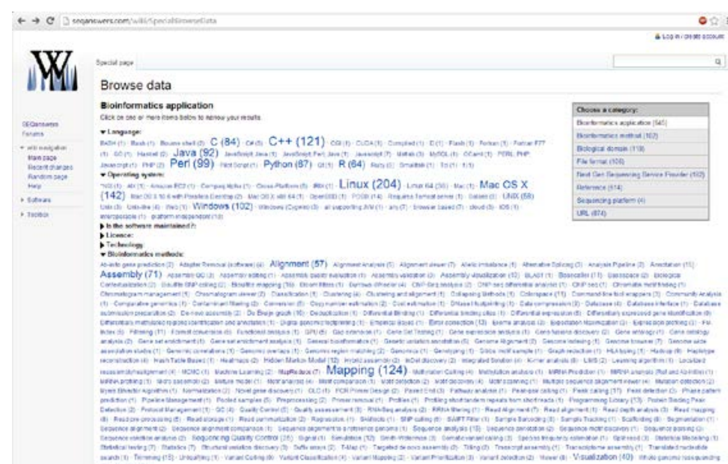
Chang, Core services: Reward bioinformaticians, Nature 2015

Two Main Issues

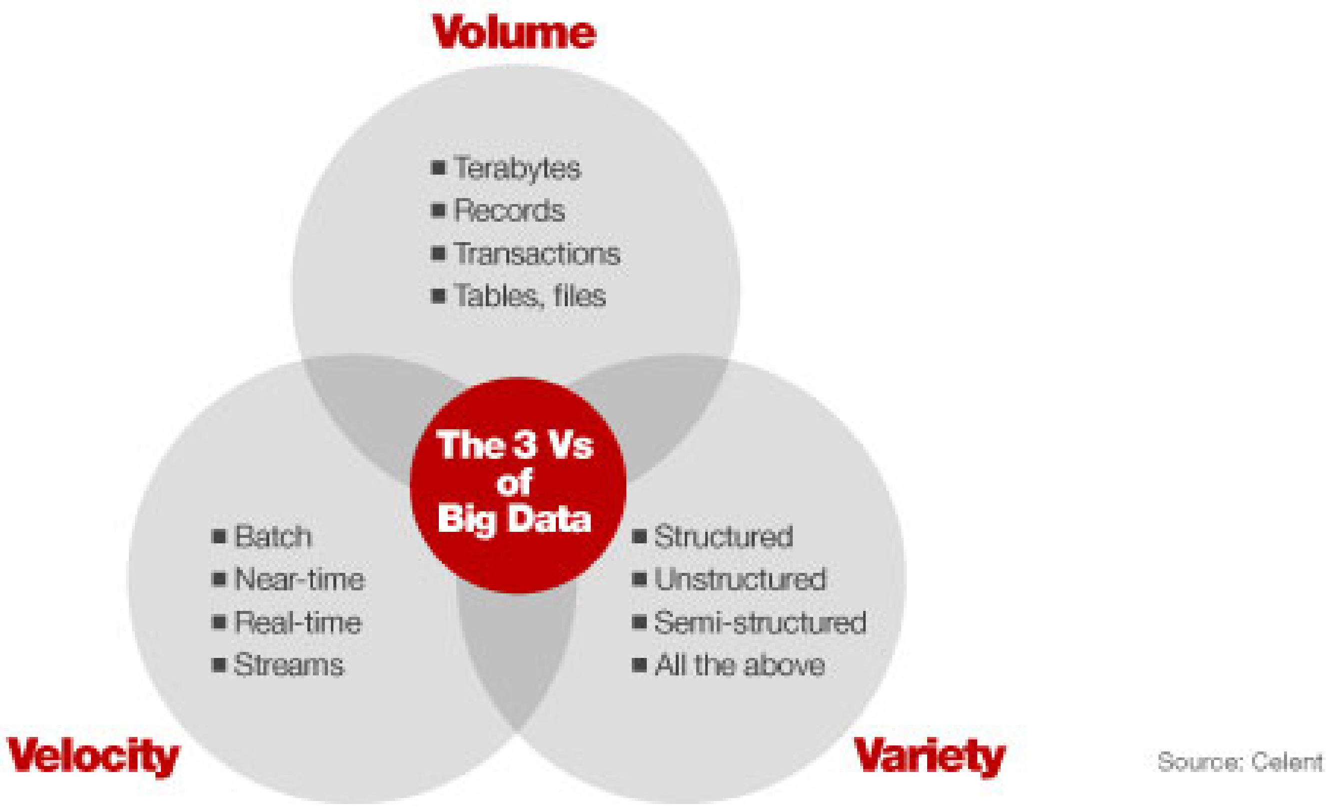
- Runtime
  - Large and growing data volumes
  - HighSeq-X: 18 terabyte a week



- Variety
  - Many different types of pipelines
  - Dozens of tools in every pipeline
  - New tools / pipelines every day
  - No gold standards



Big Data:

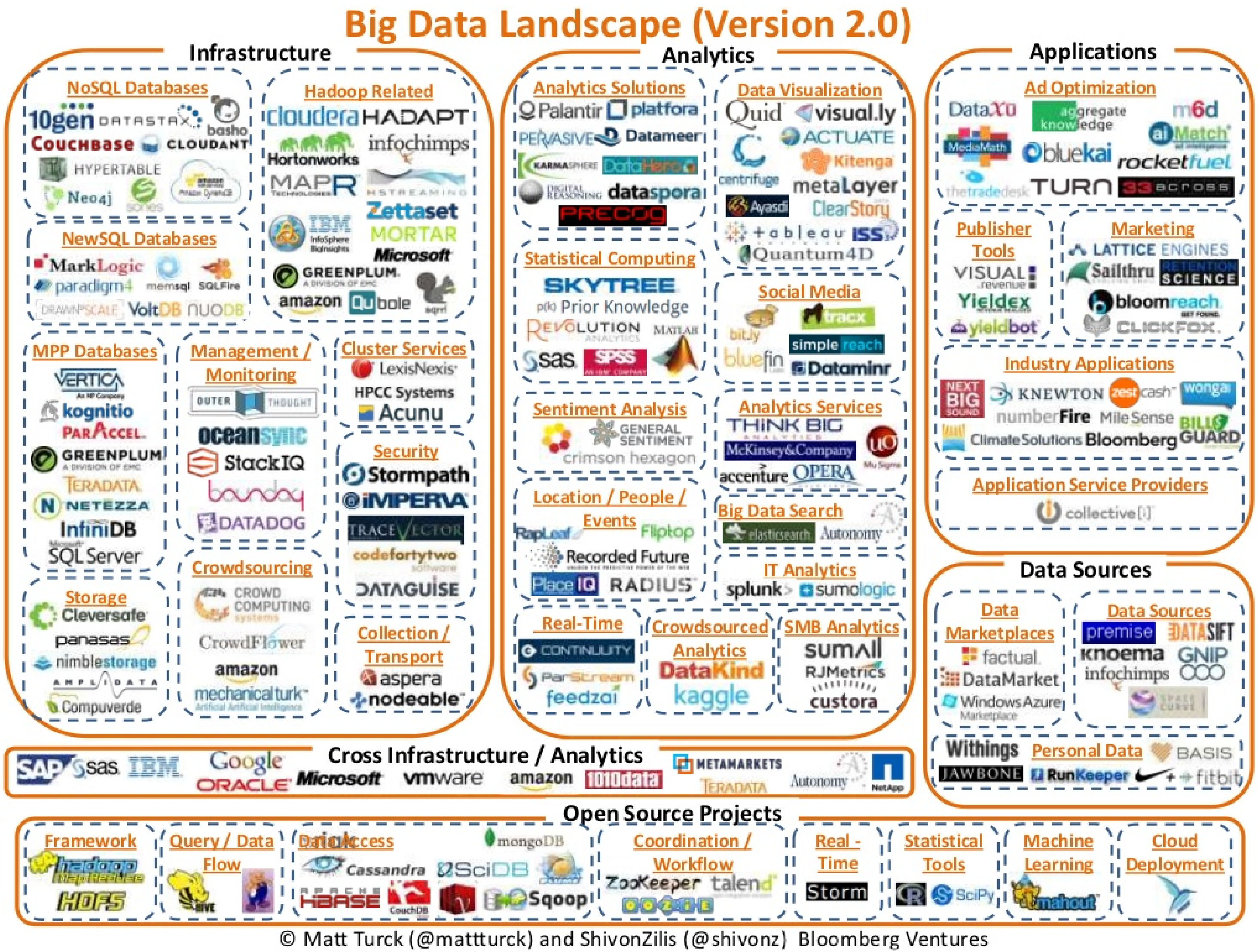


- Fourth V: Veracity
  - Big data is often very noisy (unfiltered)

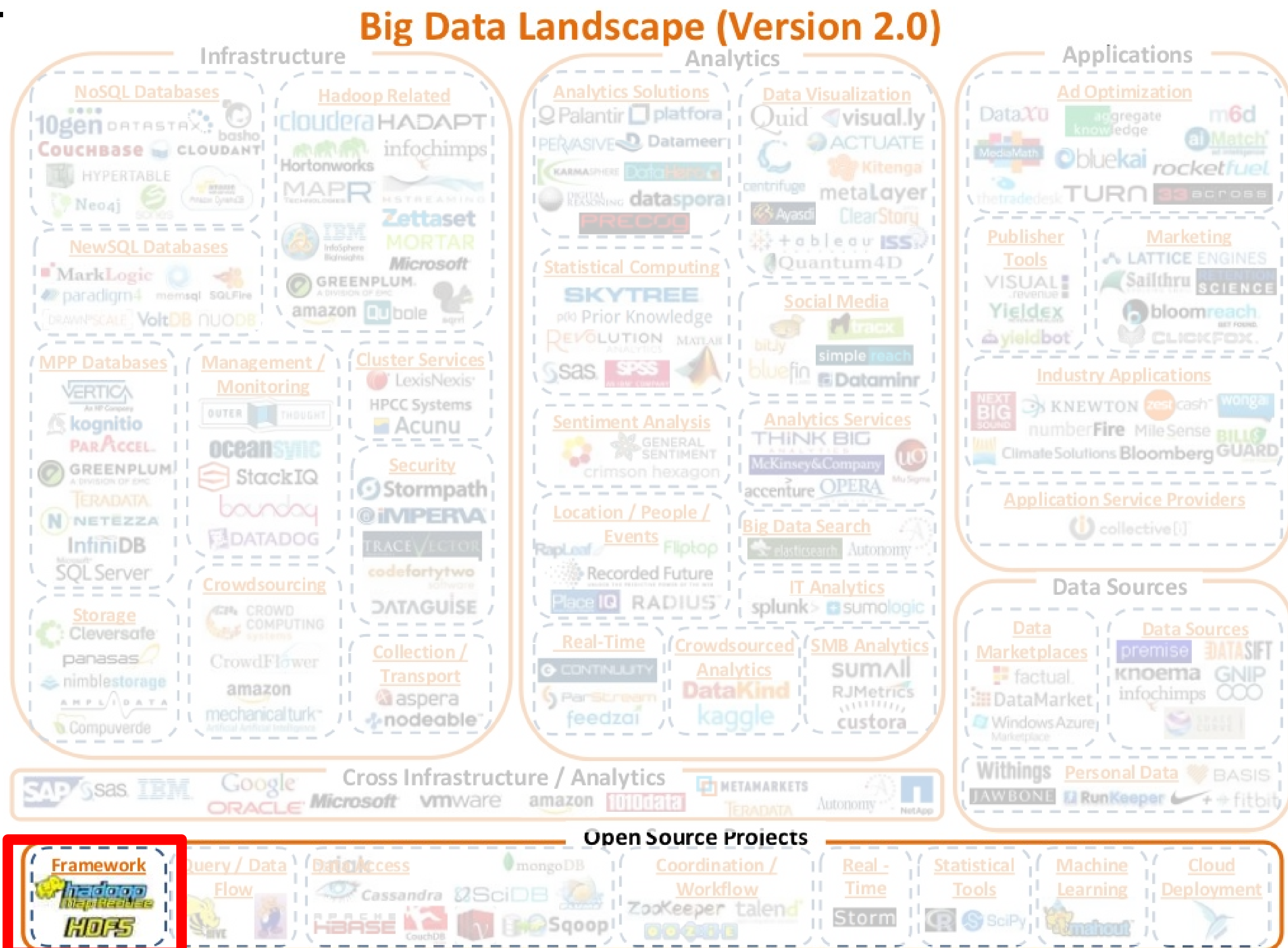
Is Big Data a Database Topic?

- We need faster algorithms: Machine learning, algorithms
- We need more scalable systems: Distributed systems
- We need more CPUs: High-Performance Computing
- We need professional services: Companies
- We need: Databases
  - Declarativity: Specialized, simple, powerful languages
  - Optimization: Compilation in efficient, parallel execution plans
  - Comprehensive data management: Robustness, data models, index structures, access control, ...

Big Data Landscape



Big Data Landscape



© Matt Turck (@mattturck) and ShivonZilis (@shivonz) Bloomberg Ventures

Content of this Lecture

- Big Data Introduction
- Map Reduce
  - Programming model
  - Framework
  - Distributed File System – HDFS
  - Error handling
- Parallel DBMS or MapReduce?
- Extensions to MR

Underlying Idea

- Observations
  - Supercomputers are expensive
  - Large commodity-clusters are error-prone
  - No simple approach to express Big Data problems and let them robustly run on a commodity cluster
  - Most parallel analysis is embarrassingly parallel
  - Functional programming has some good ideas
    - Side-effect free functions
    - 2nd order functions
- Result: MapReduce
  - Programming model: The idea
  - Software stack: A framework
  - GDFS: Google distributed file system

Some History

- Proposed by Dean & Ghemawat (Google) in 2004
  - Dean, J. and Ghemawat, S. (2004). "MapReduce: Simplified Data Processing on Large Clusters ". 6th Symposium on Operating System Design and Implementation, San Francisco, USA
  - One of the most-cited recent papers in computer science
- Yahoo adopts idea and puts code open source in 2008
  - Hadoop: Java framework for Map Reduce programming
  - HDFS: Hadoop Distributed File Systems
- Hadoop2 / Yarn in 2013
  - More flexible programming model
  - Resource management and scheduler
- Numerous companies, mostly for DWH'ish scenarios

Embarrassingly Parallel

- Data parallel problems: Partition the data, do partition-wise computation, then summarize results
  - Sum, min, max, mean, ... of a set of values
  - Find all stars in this set of satellite sky images
  - Find all company names in this set of web pages
  - Compute this join (partitioned hash join)
- Not emb. parallel: Everything influences everything (a bit)
  - Median of a set of values
  - Graphs: Traversal, routing, shortest paths, ...
    - But: Graph cluster coefficient
  - Simulations: Weather, molecular dynamics
  - Iterative problems: Clustering, PageRank, heuristic optimization
  - Solving equations, linear optimization

Parallel Execution of SQL Queries

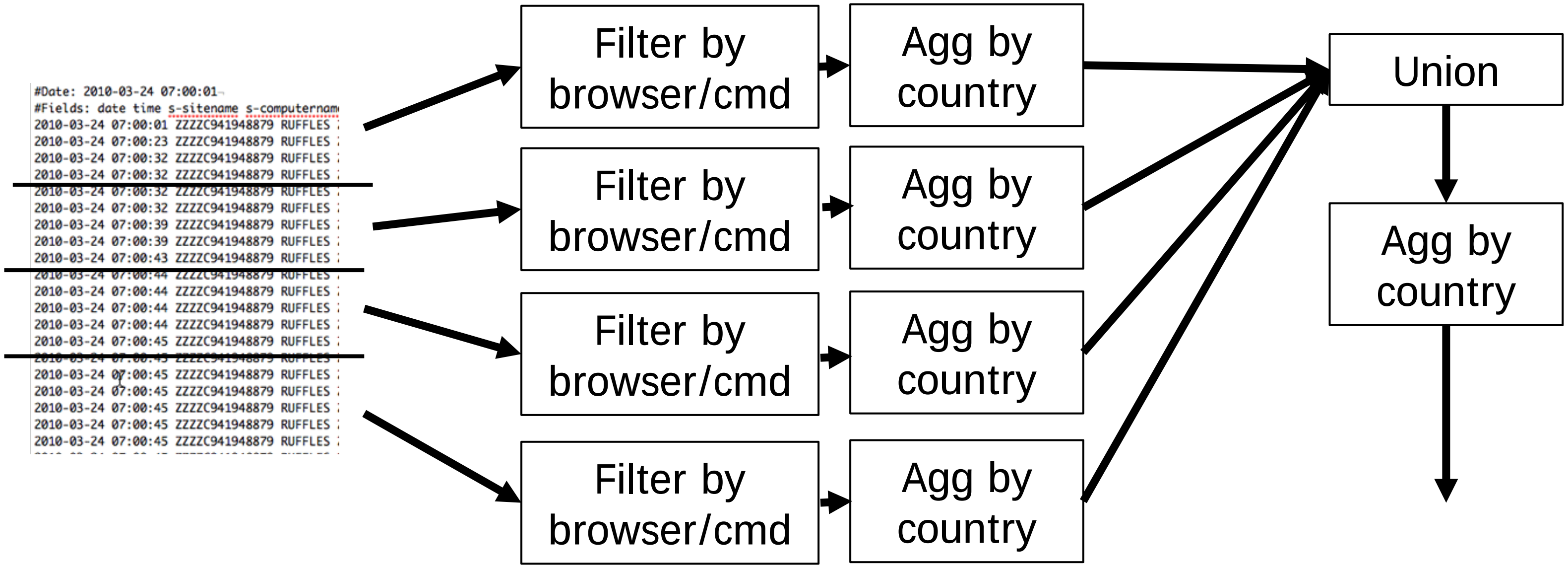
- Forget joins, include grouping + aggregations
  - MR scenarios: Web log file analysis, search engines
  - No joins, just one input set (often in many files)
  - Many aggregations and groupings (by country, by day, ...)

```
SELECT lookupCountry(parse_IP(W.IP, 1)), sum(*)
FROM   weblog W
WHERE  W.cmd="GET" & W.browser="FireFox"
GROUP BY parse_IP(W.IP, 1);
```

```
#Date: 2010-03-24 07:00:01
#Fields: date time s-sitename s-computername s-ip cs-method cs-uri-stem cs-uri-query s-port cs
2010-03-24 07:00:01 ZZZZC941948879 RUFFLES 222.222.222.222 GET / - 80 - 220.181.7.113 HTTP/1.
2010-03-24 07:00:23 ZZZZC941948879 RUFFLES 222.222.222.222 GET /2009/12/im_not_mean_im_just_ar
2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-blank.gif - 80 - 217.
2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /grep-options.gif - 80 - 217.2
2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-cat.gif - 80 - 217.2
2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-pwd-cd.gif - 80 - 21
2010-03-24 07:00:39 ZZZZC941948879 RUFFLES 222.222.222.222 GET /robots.txt - 80 - 95.55.207.9
2010-03-24 07:00:39 ZZZZC941948879 RUFFLES 222.222.222.222 GET /rss-short.xml - 80 - 173.45.2
2010-03-24 07:00:43 ZZZZC941948879 RUFFLES 222.222.222.222 GET /2009/08/22-things-you-dont-kn
2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /screen.css - 80 - 98.88.35.13
2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/rss-header-red.gif - 80 -
2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/logo.jpg - 80 - 98.88.35.
2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/input-emailsend.jpg - 80 -
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /images/cm-ebook-banner.gif -
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/bg.jpg - 80 - 98.88.35.13
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/bg-top.jpg - 80 - 98.88.3
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /21things/checkout-login.gif -
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/topnav-contact.jpg - 80 -
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /21things/portent-email-sub.gi
2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /rss-header.jpg - 80 - 98.88.3
```

Parallel Plan

- Partition log-file in X sets (X: Number of machines/cores?)
- Perform parallel, partition-wise filtering, grouping, and aggregation
- Aggregate partition-wise results (can also be done in parallel)



Why not a Parallel Database System?

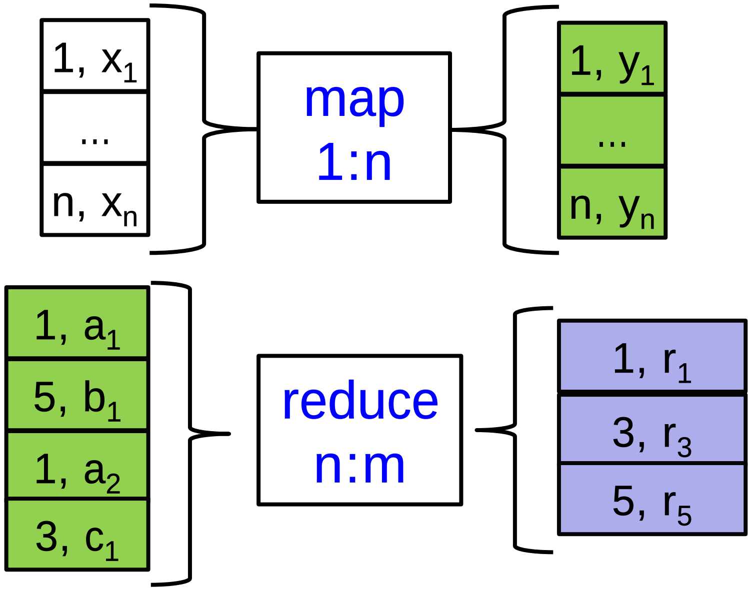
- Parallel RDBMS do not scale to large clusters
  - Synchronization: Locks and aborts lead to contention
  - Scalability: Stragglers, data transport and access, central control
  - Fault-tolerance: Redundancy, partial restarts, central control
  - CAP Theorem: Could cannot have all of these three properties
    - Gilbert & Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT 2002
- There is no open source parallel DBMS (at that time)
  - Today: Many NoSQL Systems
  - Simple data model (documents, key-value) and/or limited synchronization (eventually consistent)
- Commercial RDBMS extremely expensive

Content of this Lecture

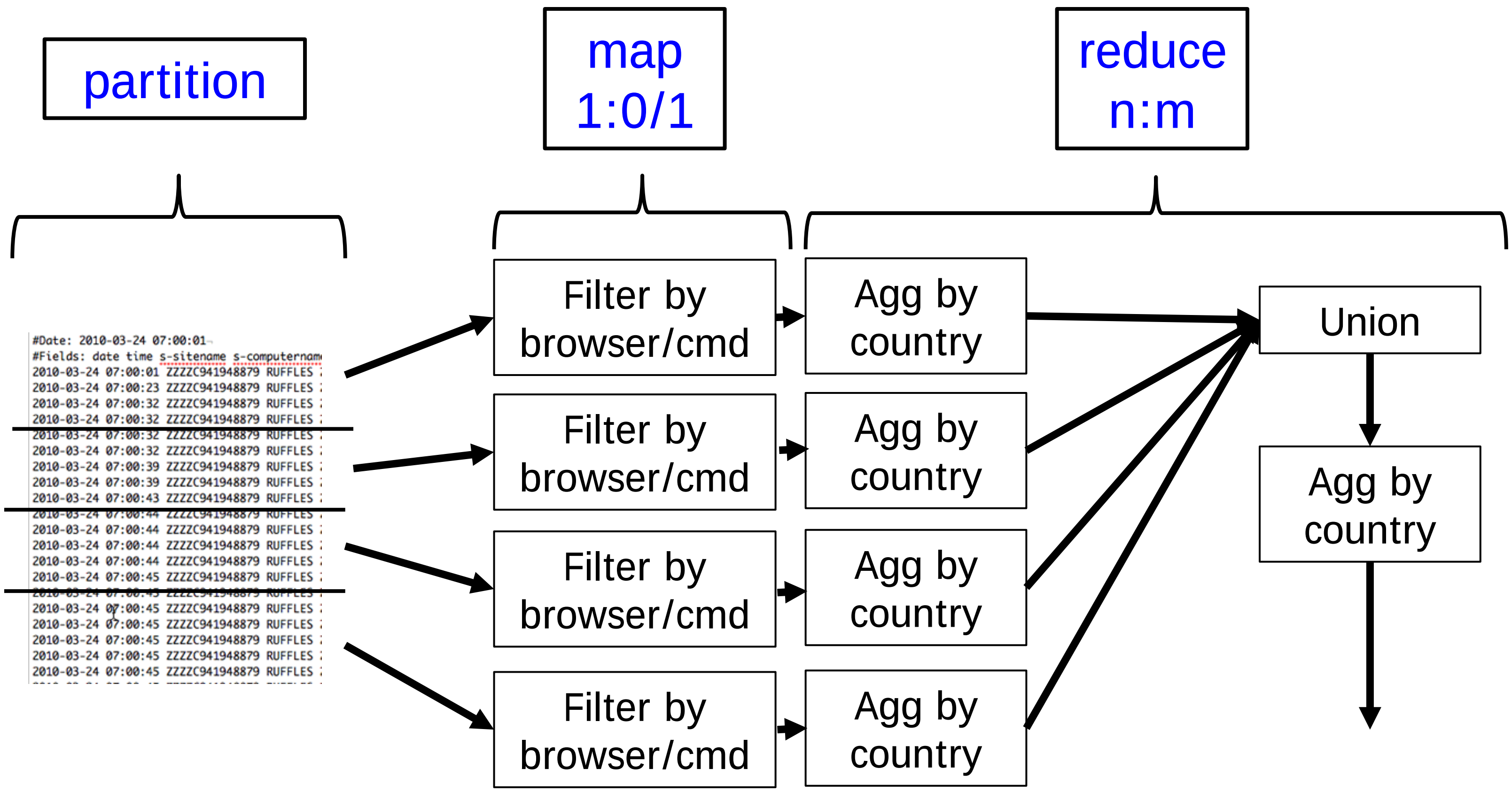
- Big Data Introduction
- Map Reduce
  - Programming model
  - Framework
  - Distributed File System – HDFS
  - Error handling
- Parallel DBMS or MapReduce?
- Extensions to MR

Map Reduce Programming Model

- Operates on sets of key-value pairs
- Inspired by FP 2nd order functions Map and Reduce
  - Map: Apply user function f on a key-value pair
  - Reduce: Group the following set on key and apply user function f on values of n all pairs with equal key



Map Reduce and SQL



Famous None-SQL Example: Word Count

- Given a set of documents, count the frequency of all unique words
  - Important for building a search engine index
- Things we need to do
  - Break documents into their words
  - Group set of all words on word
  - Compute per-word counts

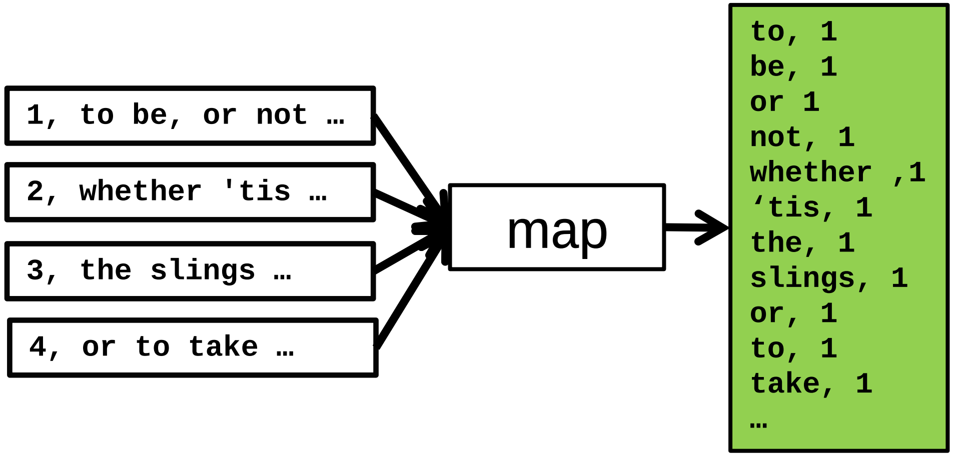
```
1, to be, or not to be, that is the question:
2, whether 'tis nobler in the mind to suffer
3, the slings and arrows of outrageous fortune,
4, or to take arms against a sea of troubles

…
```

⇒

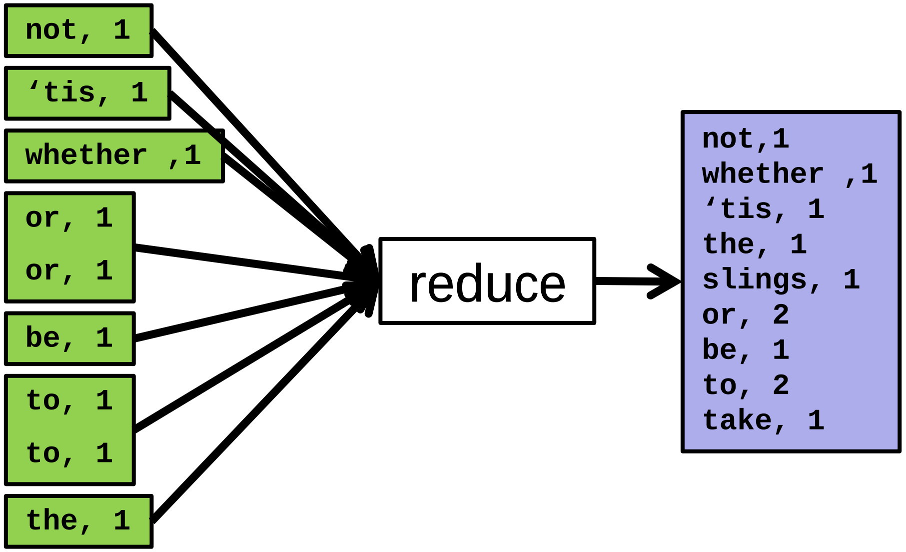
```
to: 4
be: 2
or: 2
…
mind: 1
sea: 1
…
```

Word Count in MapReduce

- Step 1: List of all words
  - Map on set of documents
  - Foreach doc, tokenize and output one key-value pair per token



- Step 2: Group by word and count
  - Sort key-value pairs by key
  - Compute sum of values per key



Content of this Lecture

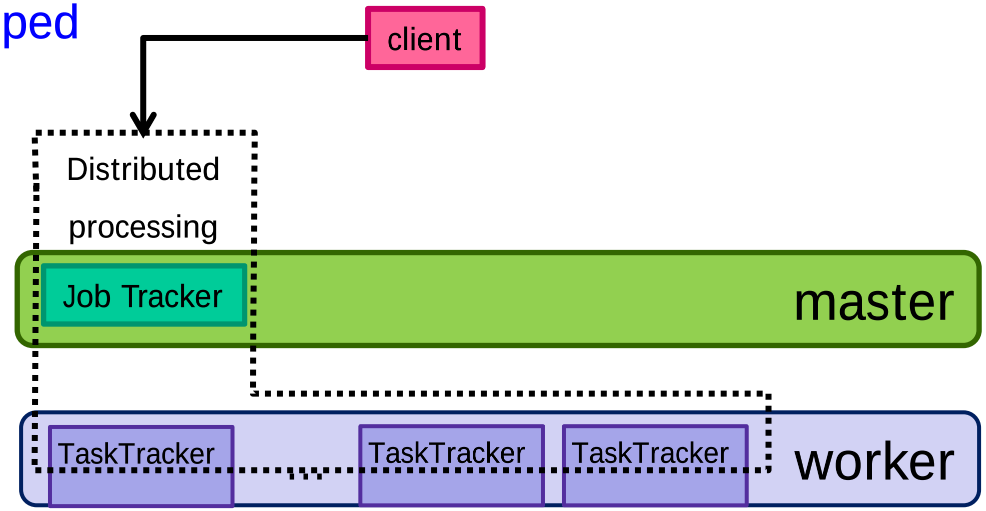
- Big Data Introduction
- Map Reduce
  - Programming model
  - Framework
  - Distributed File System – HDFS
  - Error handling
- Parallel DBMS or MapReduce?
- Extensions to MR

Operational View

- User provides
  - Map function (Java class with certain interface)
  - Reduce function (Java class with certain interface)
  - Option: Partitioner (Default: Partition by lines)
    - Default not applicable to binary files
  - System configuration: Machines, IP, access
- System provides
  - Java framework where code is integrated
  - Jobs are automatically created and distributed in cluster
  - Jobs that fail get restarted
  - Sort/combine functions for reduce
  - Distributed file system with in-build redundancy
  - Simple scheduling (greedy) and locality (unclear)

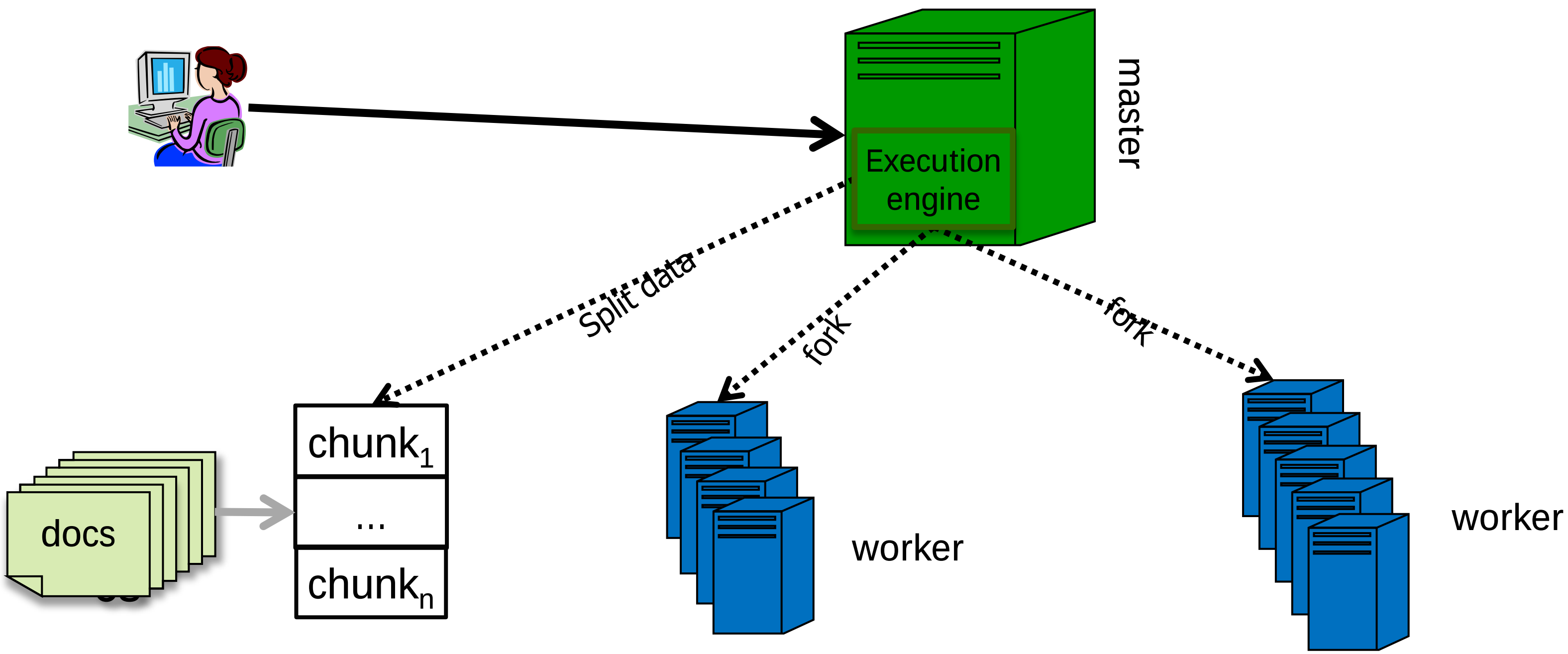
Hadoop Architecture

- Master/Worker architecture
  - Workers are commodity hardware
  - Masters are usually well equipped
  - Shared nothing
  - File exchange by HDFS



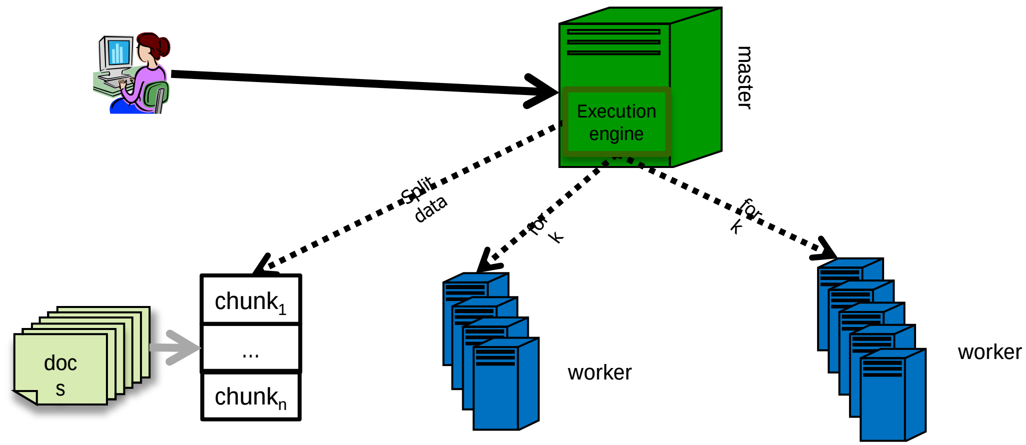
- Reports on installations with several thousand machines

Execution



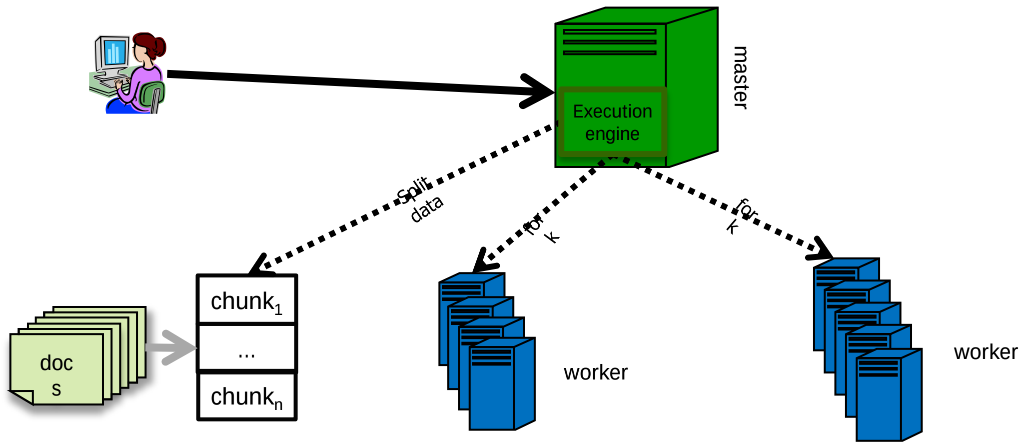
- Master: Manages entire execution
  - Assigns tasks and input splits to idle workers
  - Tracks status of current jobs (waiting, running, finished)
  - Tracks status of worker nodes

Execution



- Map-Worker
  - Reads assigned splits
  - Parses key-value pairs
  - Executes map function for each pair
  - Buffered intermediate data are periodically written to local disks
  - Notify master about locations of result when finished
  - Master pushes data incrementally to reduce-workers

Execution



- Reduce-Worker
  - Read assigned splits from map workers (through HDFS)
  - Usually processes more than one key group
  - Performs intermediate sort of MAP-results
  - Executes reduce() function once for each key - aggregation
  - Result is written to HDFS
- Master notifies host program, data accessible in HDFS

Content of this Lecture

- Big Data Introduction
- Map Reduce
  - Programming model
  - Framework
  - Distributed File System – HDFS
  - Error handling
- Parallel DBMS or MapReduce?
- Extensions to MR

HDFS architecture

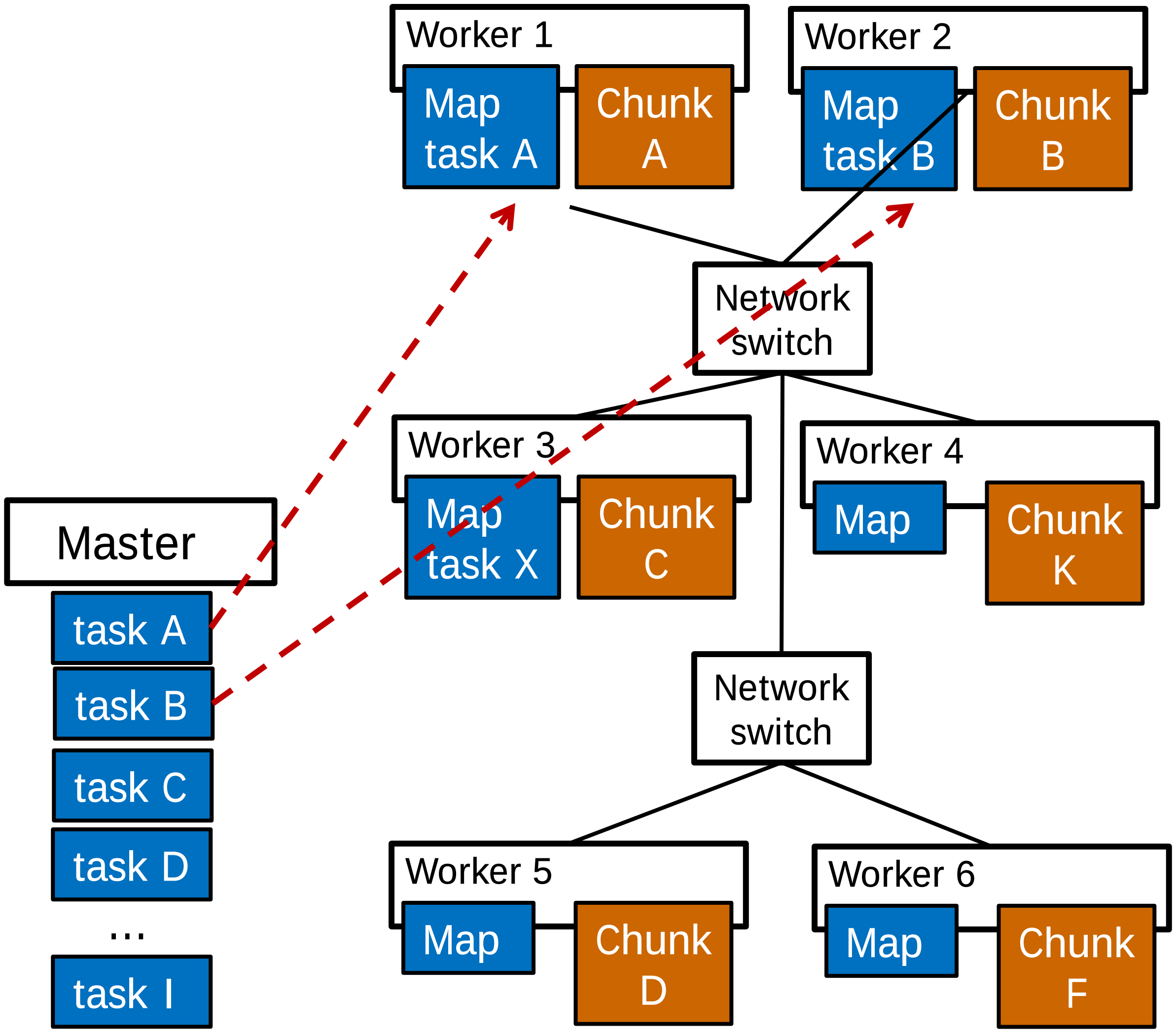
- HDFS: Hadoop distributed file system
- Why a special file system?
  - Goal: store very large data redundantly on commodity hardware
  - Network latency should not hinder computations too much
  - Mostly read/append file operations, few rewrites
  - Relatively small number of large files
  - Manage much larger data compared to ‚standard‘ fs
- Data distributed over commodity hardware
  - Nodes fails all the time
  - Data must be kept redundant and distributed
  - Single point of access

Some Details

- Master/Worker architecture
  - Master: Access to DFS, manages replication and metadata
  - Workers: Local storage in cluster, I/O and maintenance
- Client talks to
  - Master: To get file handles
  - Workers: To (directly) read/write data
- Replication
  - Files chopped in chunks (64MB), chunks are replicated (3 times)
  - Advantage: Fixed file size, easier calculations, files larger than disk
- Interference with local FS
  - HDFS workers have a reserved space in “usual” FS
  - Access only through HDFS, no posix ☹

Locality: Co-locate Data and Computation

- Goals
  - Increase speed of access
  - Avoid network congestion
- What is local?
  - Physically on same machine
  - Close in the network
  - Within the same rack



Error Handling

- Large commodity-hardware clusters:
  Errors (node failure) are the norm, not the exception
- Hadoop
  - Master nodes fail: System gone
    - Use robust machines, failover, RAID, ...
  - Worker nodes fail
    - Tasks crash: Identify and restart
    - Node crashed: Identify (heart-beat) and mark as “dead”
    - Data is replicated: Identify gone chunks and replicate to new nodes
  - Stragglers: Tasks taking much longer than others
    - Create many more (small) tasks than machines
    - When few tasks left, start them multiple times and use first finished
      - Often system hangs, network bottleneck, ...

Summary

- Focus on cheap hardware yet large systems
- Made by distributed systems people: No real data model, ad-hoc analysis, no indexes, no query languages, ...
- Made for programmers: Java (not SQL)
- Made for non-relational workloads: Files (not records)
- Does not solve many (exotic) DS problems, but focuses on average workloads
  - E.g., we didn't mention time synchronization
- Many shortcomings: Scheduling, data placements, strict MapReduce framework, joins, ...
- Extremely influential, many subsequent developments

Content of this Lecture

- Big Data Introduction
- Map Reduce
  - Programming model
  - Framework
  - Distributed File System – HDFS
  - Error handling
- Parallel DBMS or MapReduce?
- Extensions to MR

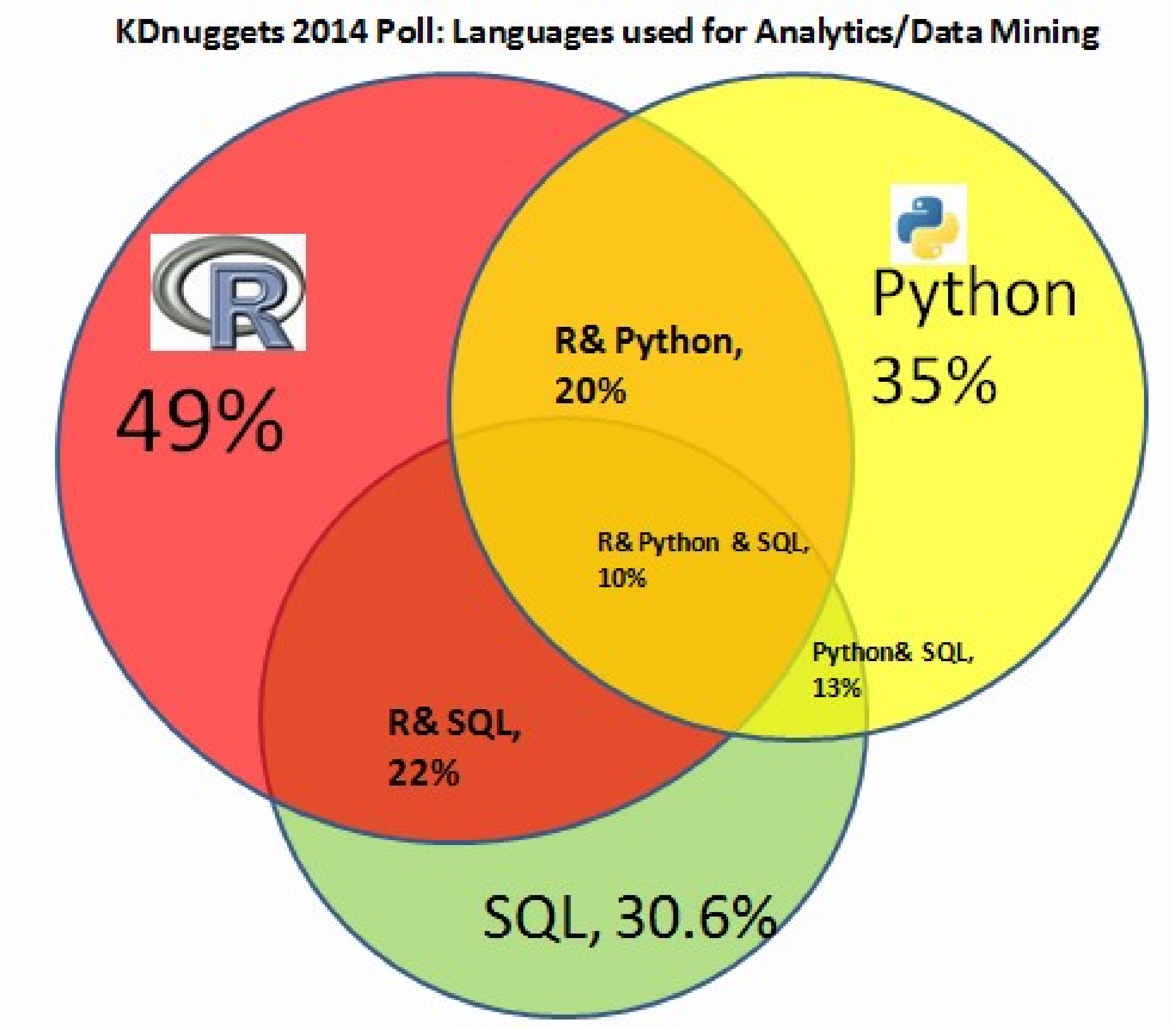
"Major Step Backward"

- Many database people criticizing MapReduce
  - Pavlo, Paulson, Rasin, Abadi, DeWitt, Madden, Stonebraker. A comparison of approaches to large-scale data analysis. SIGMOD 2009
  - Nothing really new
    - Functional programming, distributed systems, Teradata
  - Functionality can be reached by UDFs in parallel DBMS
  - Parallel DBMS provide good scalability (and much more)
  - Interface too low-level and not declarative
  - Disk-based batched data exchange instead of pipelining/streaming
  - Lack of schemata obstructs performance optimizations
  - No indexing for recurring analyses
  - No statistics for cost-based optimizations
  - Frequent re-parsing of ASCII data
  - ...

Points of Views

	Parallel RDBMS	MapReduce-Style Processing
Data sets	Large, structured, discrete attributes, normalized, correct	Unstructured, binary, redundant, noisy
Analysis	Relational queries	Everything, domain-specific
Data / queries	Same data, many different queries	Same data, few queries
Pre-Analysis	Pays off • Ingestion: Loading into DB • Parsing, statistics, indexing	Does not pay-off • Ad hoc analysis • Data too large for pre-analysis
Units of work	Transactions	Read-only
Updates	Multi-user and synchronization	Read-only
Reliability	High reliability	Pragmatism instead of perfect reliability
Availability	Industry-proven system	Many open source systems
Cost	Very expensive	Much cheaper

What Most People Like



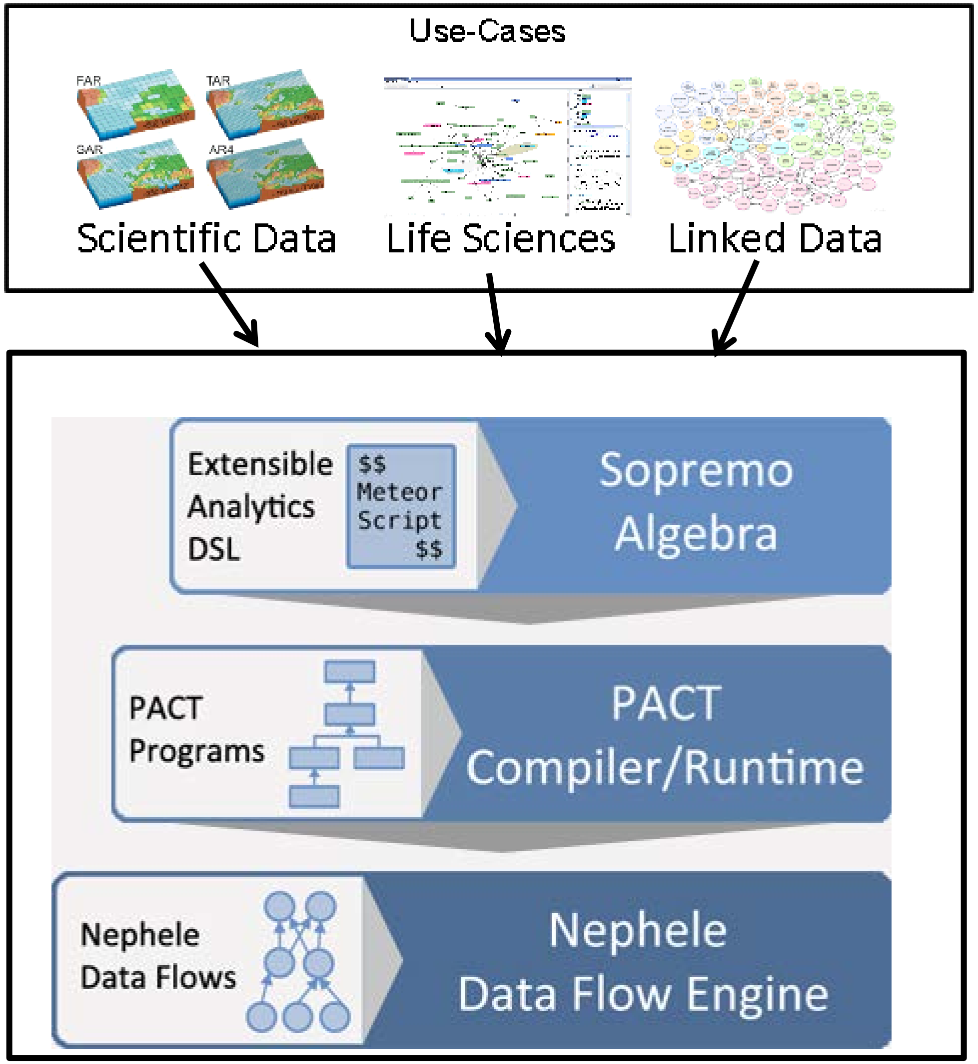
Content of this Lecture

- Big Data Introduction
- Map Reduce
- Parallel DBMS or MapReduce?
- Extensions to MR
  - Stratosphere
  - Scientific Workflow Systems und SaasFee

Programming Model has Many Limitations

- Always scan all input data
- No support for schemata
- Rigid schema: map & reduce
- No joins, no heterogeneous inputs
- Low-level, imperative access
- Only Java
- No iterative workflows (recursion etc.)
- Naïve scheduler
- Too much IO, too much network traffic
- ...
- [Google abandoned MR in ~2012]

Stratosphere



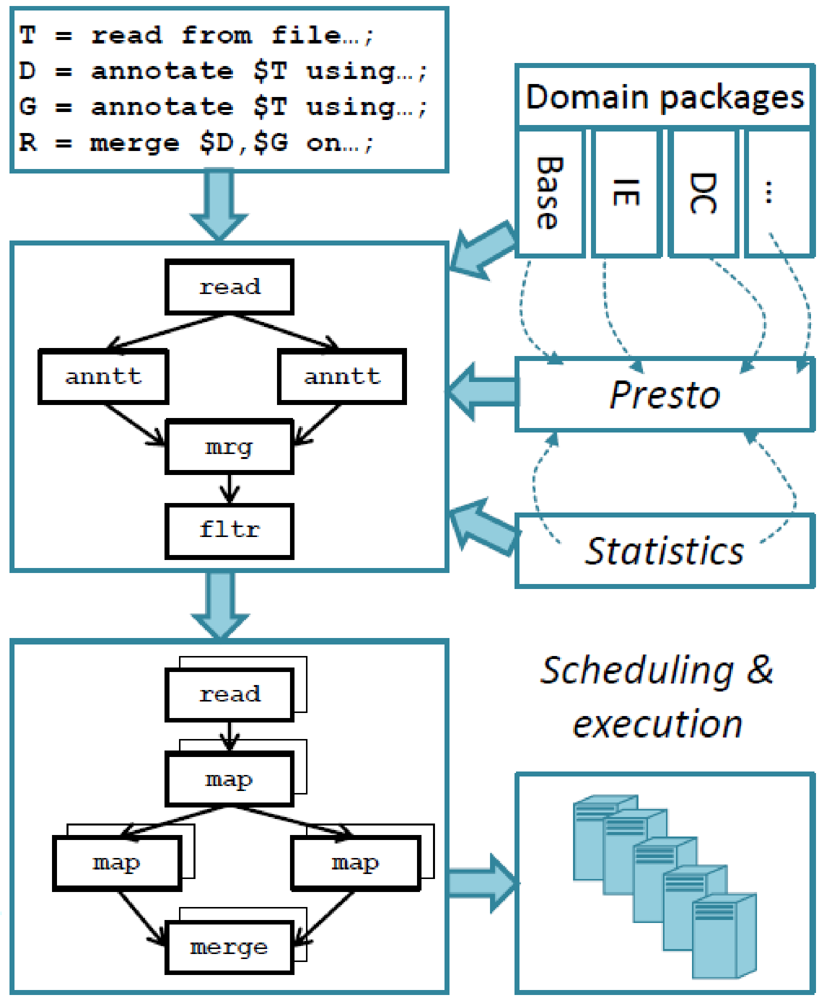
- Research project
  - TU, HU, and HPI
- “Web-scale” distributed data analytics system
- Database-inspired
  - Semi-structured data model
  - Analytics as queries
  - Declarative languages
  - Optimization
  - Extensible by domain-specific operations
- More flexible program-ming model

StratoSphere
Above the Clouds

Declarative language with domain-specific predicates

Logical optimizer to generate optimal execution plans

Cloud-enabled processing engine for fully parallel execution



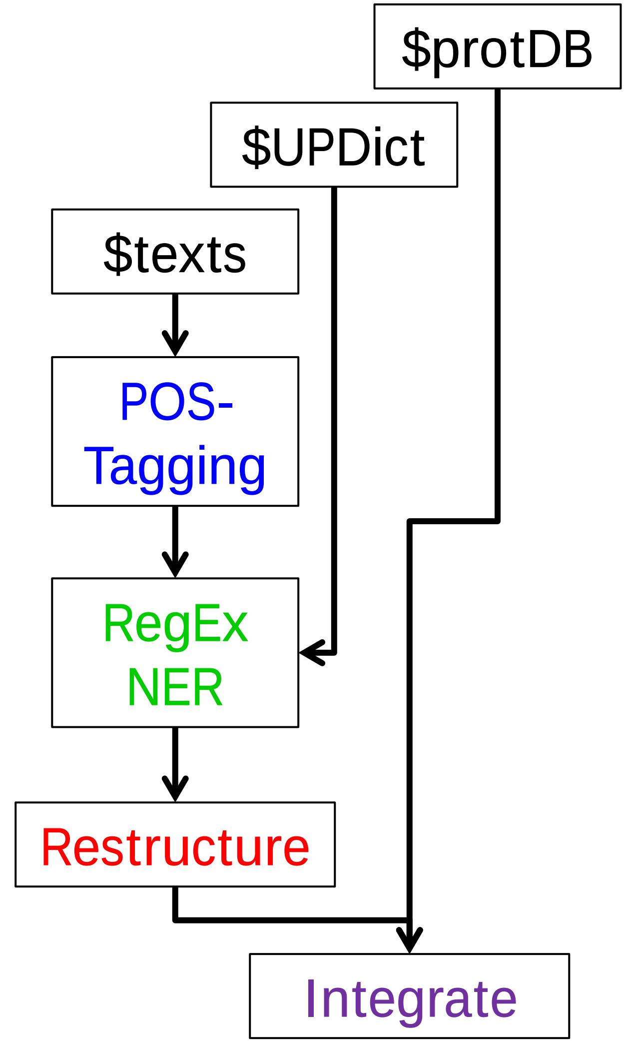
METEOR: A Dataflow Specification Language

```
$texts = read from ‚articles.json‘;
$recent = filter $text in $texts
            where $text.year >= 2000;
write $recent to ‚recentTexts.json‘;
```

- JAQL style
- Semi-structured data model
- Designed for extensibility
- UDFs in domain-specific packages
- Root package = relational operators
- Uniform optimization framework across all packages

More Complex Example

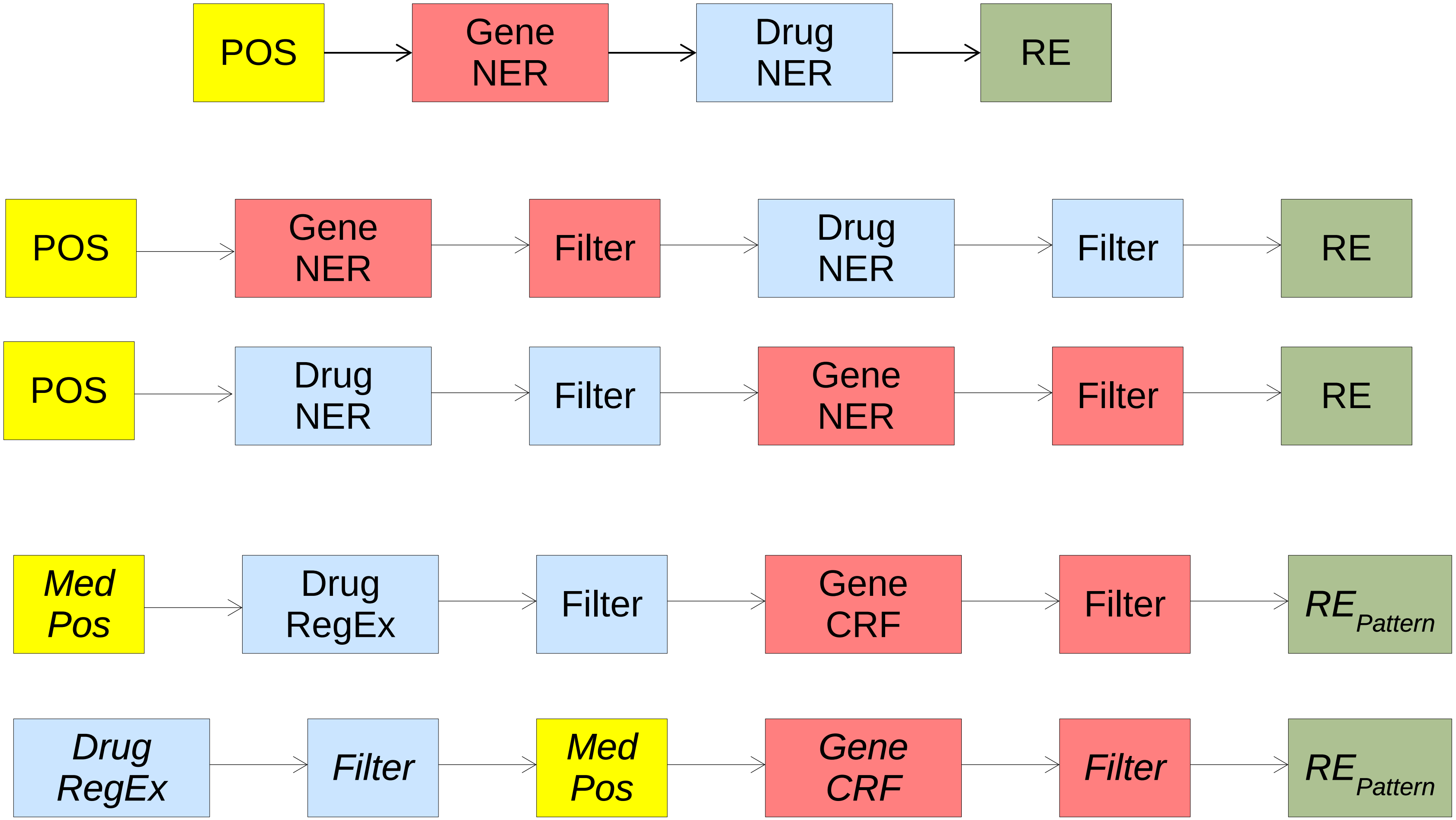
```
using ie;

$texts = read from ‚pubmed.json‘;
$protDB = read from ‚uniprotKB.json‘;

$texts= annotate sentences in $texts
  using medPost;
$texts = annotate entities in $texts
  using type.protein and regex ‚UPDict';

$pText = pivot $texts around
  $ent = $text.annotations[*].entity
  into { protein:$ent, art:$arts};

$refs = join $p in $pText,$prot in $protDB
  where $p.protein.id=$prot.id
  into { protein:$prot, reference:$p.art};

write $refs to ‚uniprotWithRefs.json‘;
```



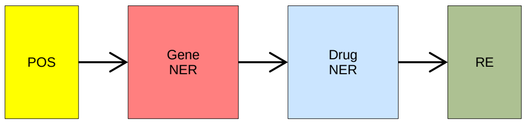
Logical Operators

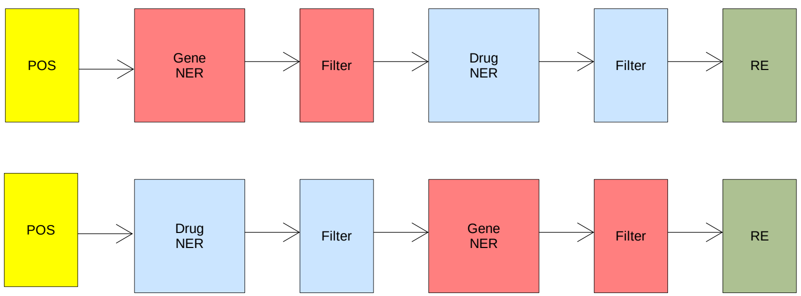
Relational	Information Integration
All: Filter, project, join, intersect, union, ...	Data scrubbing
Information Extraction	Duplicate detection
Sentence splitting	Entity mapping
Tokenization	Record linkage
N-Gram extraction	
Part-of-speech tagging	**Web Extraction**
Dependency parsing	HTML-scrubbing
Entity annotation	Metadata extraction
Relationship extraction	Boilerplate detection
...	...

Optimization: Plan Reordering

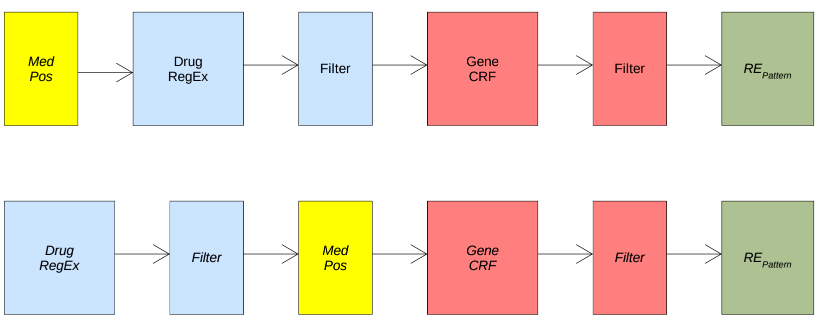


Optimization: Plan Reordering



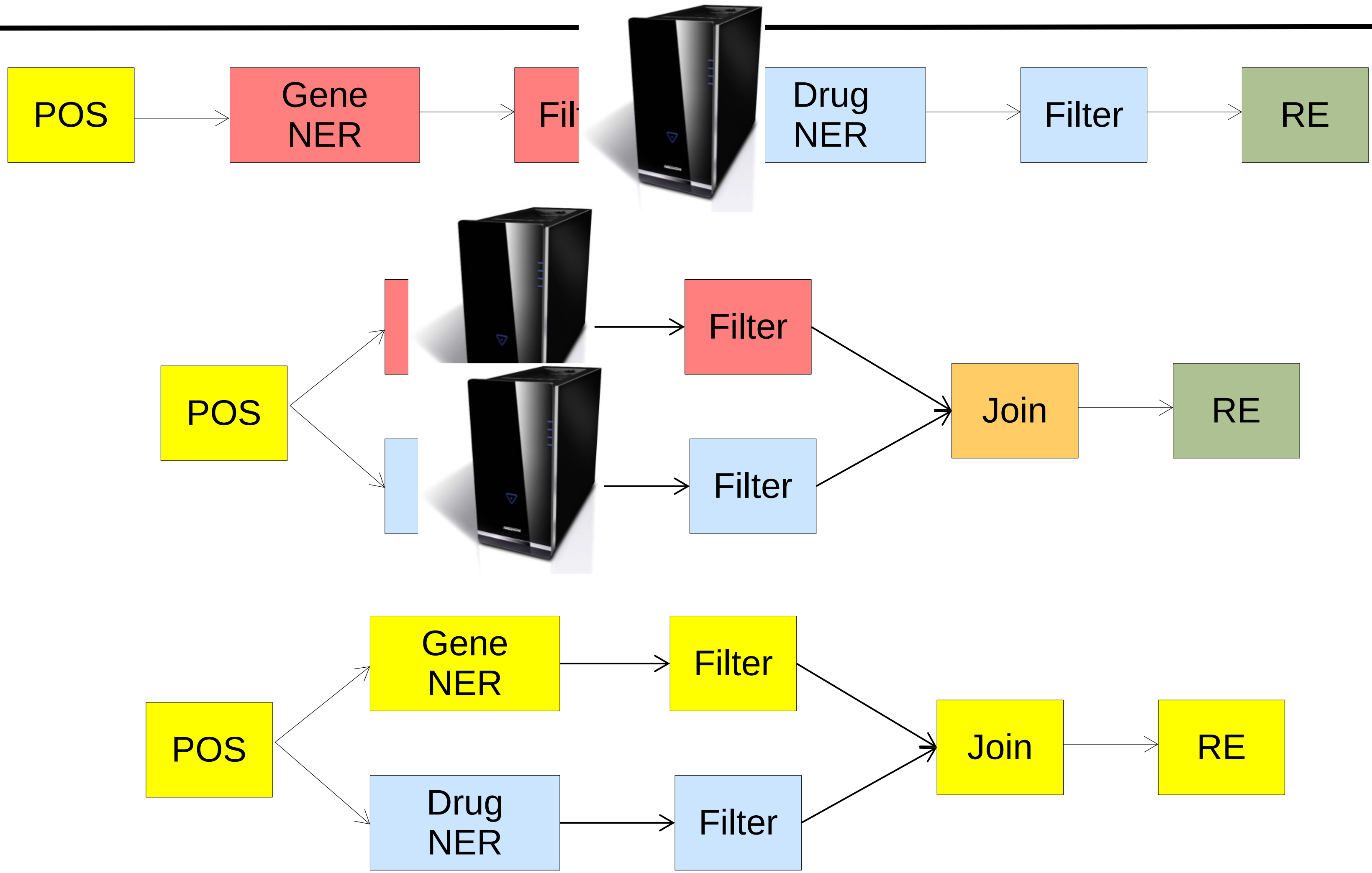


- Data-specific preconditions allow early filtering
- Cost estimates allow operator reordering



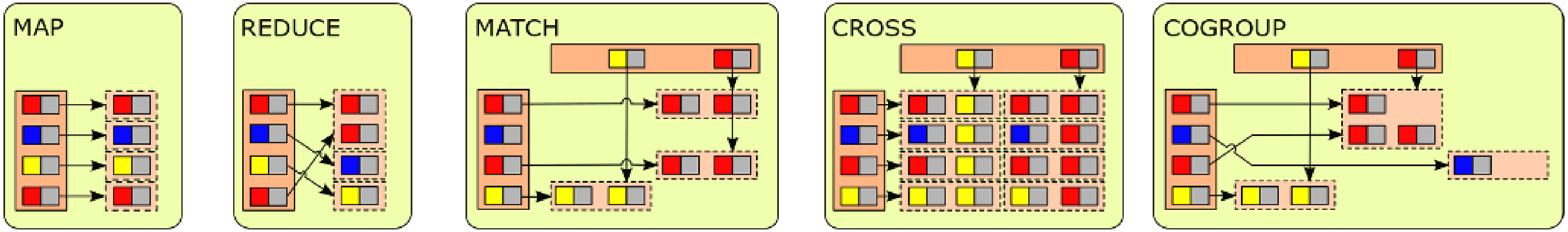
- Instantiation: Logical to physical operators
- Dependency resolution allows operator reordering

Parallelization



Stratosphere programming model

- PArallelization ConTracts (PACTs)
  → Generalization of MapReduce



Many Other Variations / Improvements

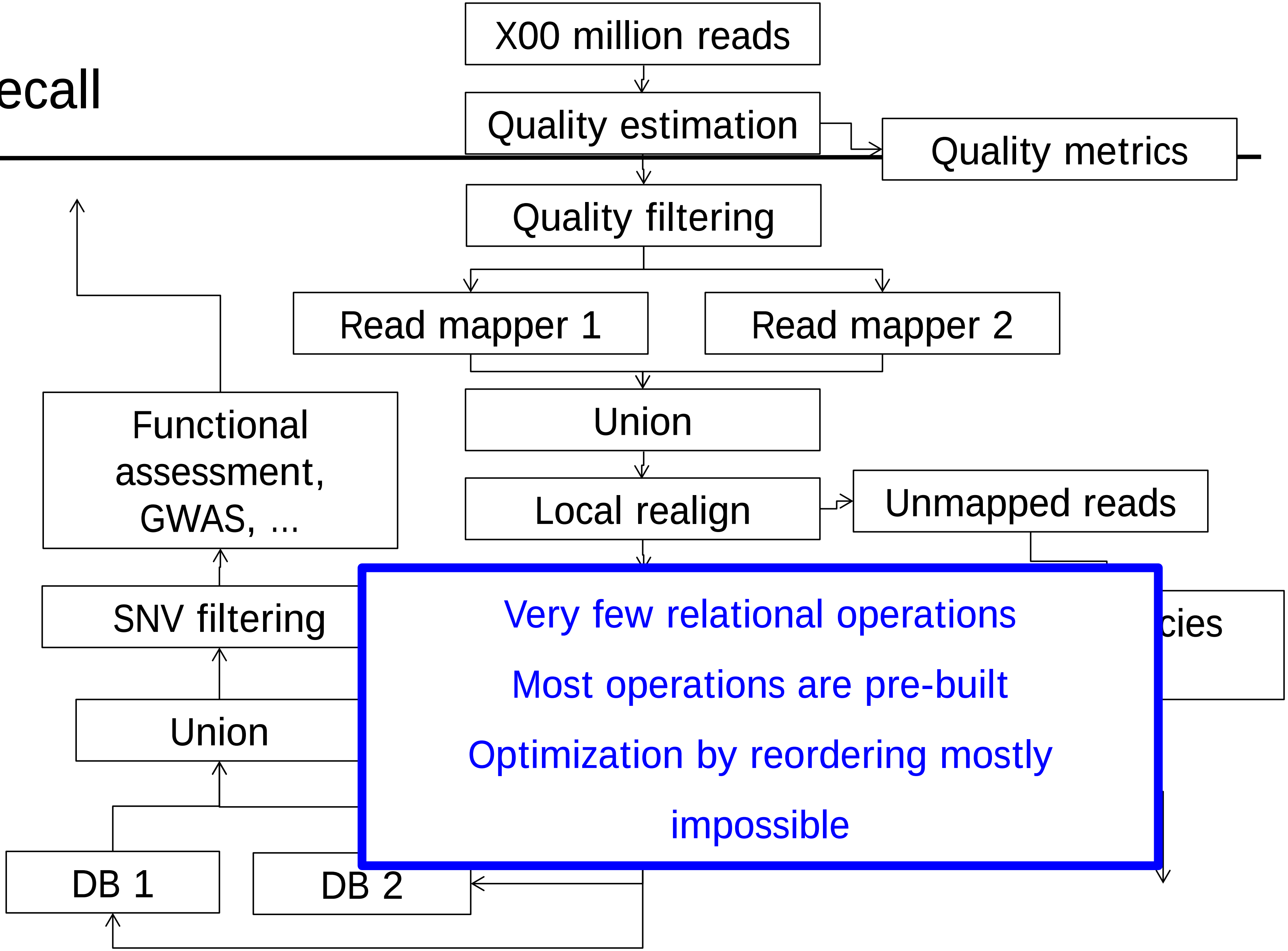
- **Hbase:** ist eine skalierbare, einfache Datenbank zur Verwaltung sehr großer Datenmengen innerhalb eines Hadoop-Clusters....
- **Hive:** erweitert Hadoop um Data-Warehouse-Funktionalitäten, namentlich die Anfragesprache *HiveQL* und Indizes. ... Seit Hive 2.0 wird Hybrid Procedural SQL On Hadoop (HPL/SQL) unterstützt
  - Im Sommer 2008 stellte Facebook, der ursprüngliche Entwickler von Hive, das Projekt der Open-Source-Gemeinde zur Verfügung.[14] Die von Facebook verwendete Hadoop-Datenbank gehört mit etwas mehr als 100 Petabyte (Stand: August 2012) zu den größten der Welt.[15] Dieses Warehouse wuchs bis 2014 auf 300 PB an[16].
- **Pig:** kann Hadoop MapReduce-Programme in der High-Level-Sprache Pig Latin erstellen. Einfach, erweiterbar, optimiert
- **Spark:** ist eine in-memory Batch Processing Engine, welche vornehmlich für Machine-Learning-Anwendungen entwickelt wurde
- **Flink:** ist wie Spark eine in-memory Batch Processing Engine und bietet grundsätzlich ähnliche Funktionen, wobei der Fokus stärker auf Machine Learning und Complex Event Processing liegt. Sie basiert auf dem europäischen Forschungsprojekt Stratosphere.
- ...

Source: Wikipedia

Content of this Lecture

- Big Data Introduction
- Map Reduce
- Parallel DBMS or MapReduce?
- Extensions to MR
  - Stratosphere
  - Scientific Workflow Systems und SaasFee

Recall



Very few relational operations

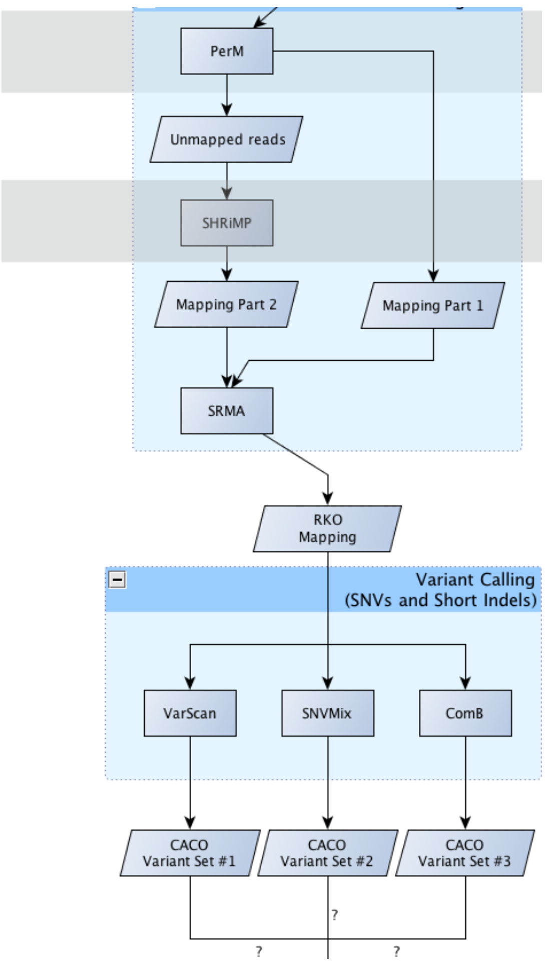
Most operations are pre-built

Optimization by reordering mostly impossible

Other Approaches to Big Data Analytics: Scientific Workflows

Links

- Connect input and output ports
- Implemented as files, memory, network
- Determine data dependencies / orders of execution



Tasks

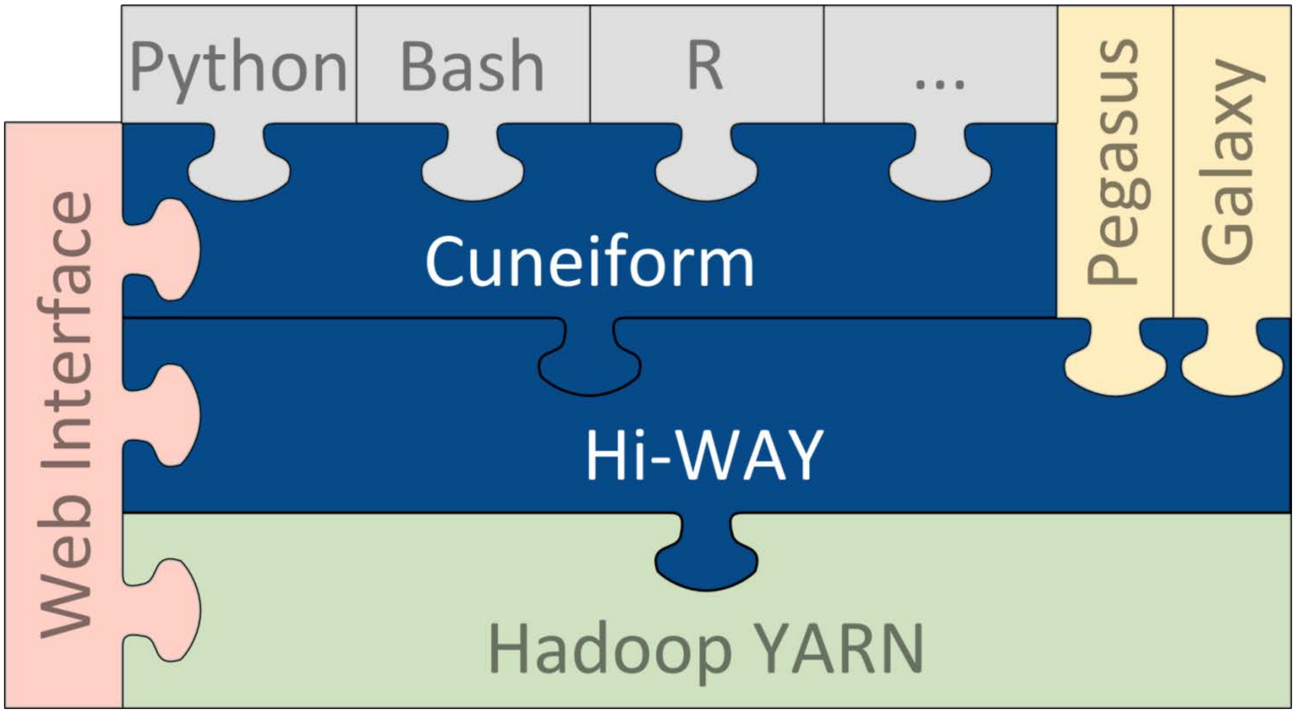
- Input and output ports
- Executables or web services
- Typically black box implementation

Features

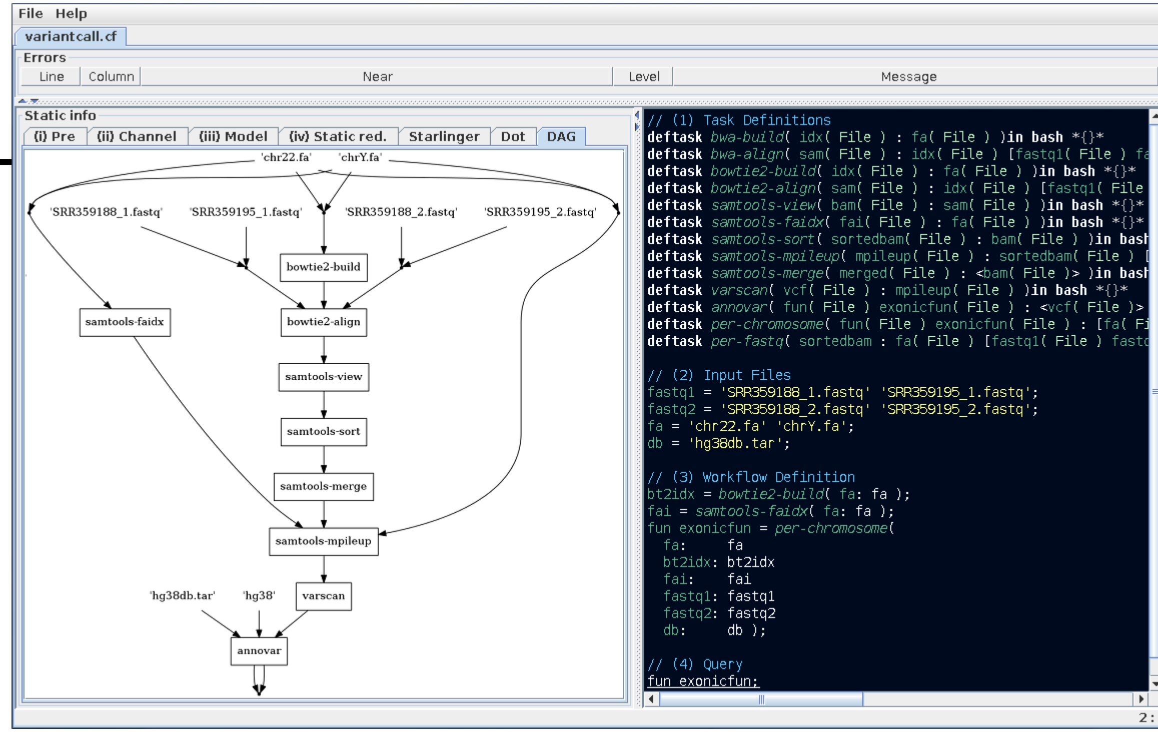
- Controlled assembly of black box scripts
- Execution monitoring and failure recovery
- Parallelization and scheduling
- Workflows: Understandable & sharable
- Limited expressivity: Easier to read and develop
- Often with graphical user interfaces for wf composition
  - SWMF for end users / for developers
- Reproducibility

- Requires Scientific Workflow Management System

SaasFee Software Stack



www.saasfee.io (video tutorials available)



- Light-weight statically typed functional dataflow language
- Compiles into dynamic pipelines of black-box tools
- Make foreign code integration as easy as possible
- Allow complex, iterative workflows

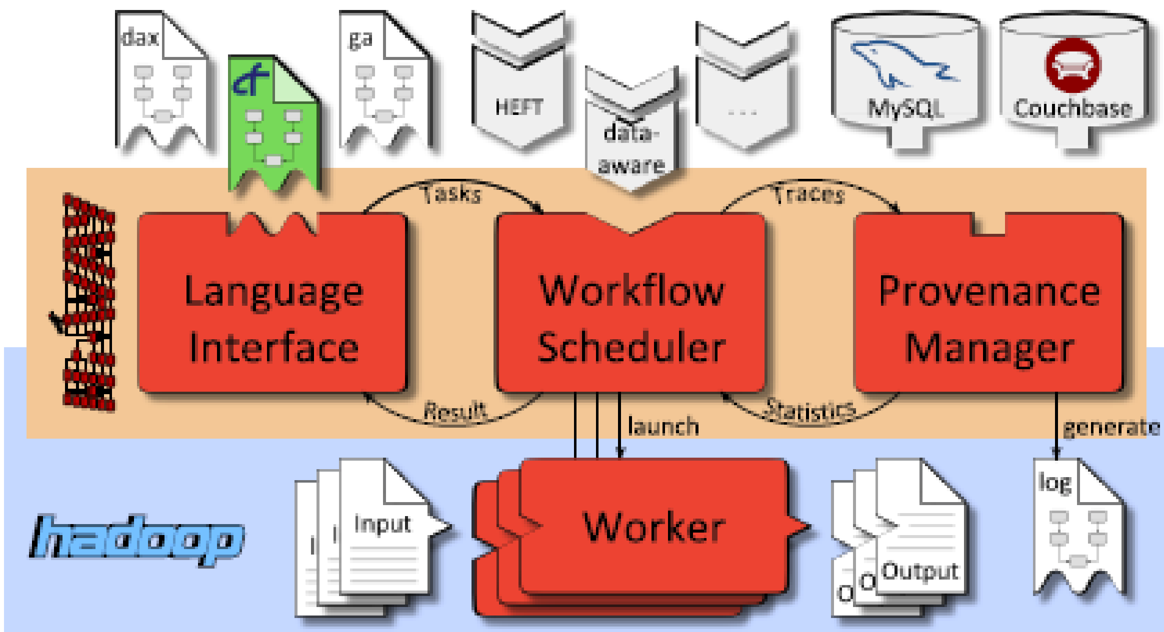
Foreign Code Interface

- Directly integrates BASH, LISP, R, MatLab, Python ...
- No wrapping, no data (un)marshalling, no API
- Communication via variables or files
- Mixing of several languages
- Snippets are shipped and executed by Hi-Way

```
deftask greet( out : person )in bash *{
    out="Hello $person"
}*
deftask greet( out : person )in r *{
    out = paste( "Hello", person )
}*
```

Hi-Way

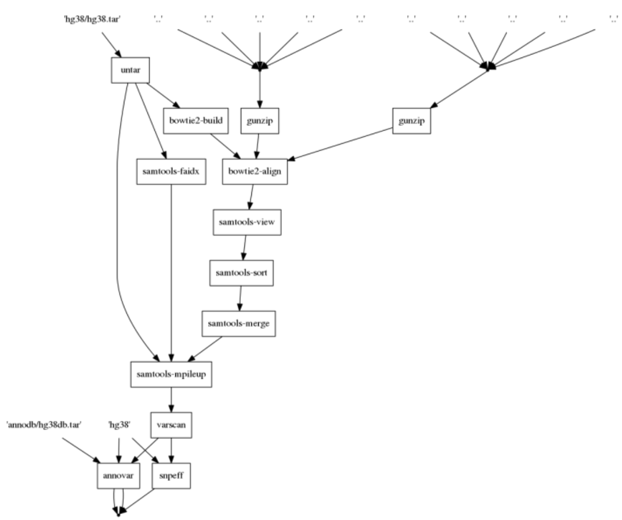
- Hi-Way Workflow Application Master for YARN
- Executes workflows on Hadoop YARN
  - Scalability, maintenance, fault tolerance, ...



Achieving Parallelism

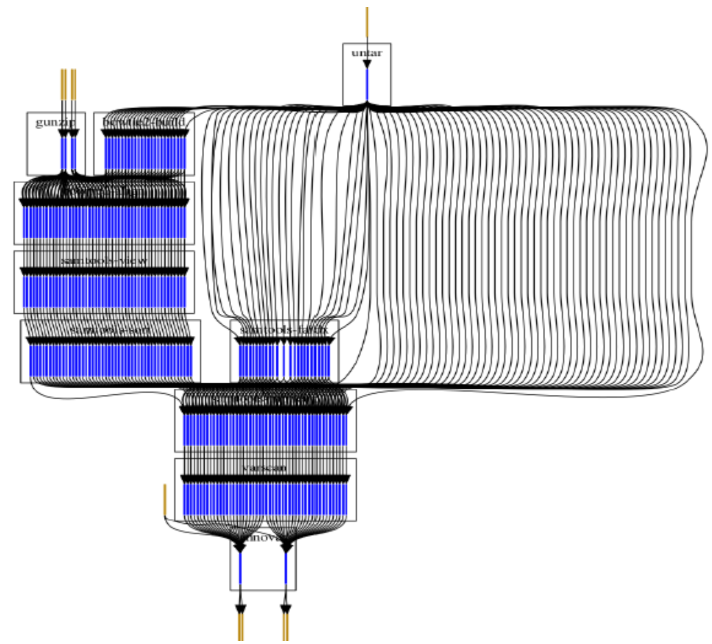
Task Parallelism

- Data dependencies



Data Parallelism

- Custom partitioning
  - Default for record-oriented files
-
-



Scalability

- Cluster with >500 cores
- Variant calling, unsplit reference, split data
- Almost perfect scalability