From Small to Big Data Management

Juliana Freire
Computer Science & Engineering
Visualization and Data Analysis (ViDA)
Center for Data Science (CDS)
Center for Urban Science and Progress (CUSP)

VIDA@NYU

Over 30 active members

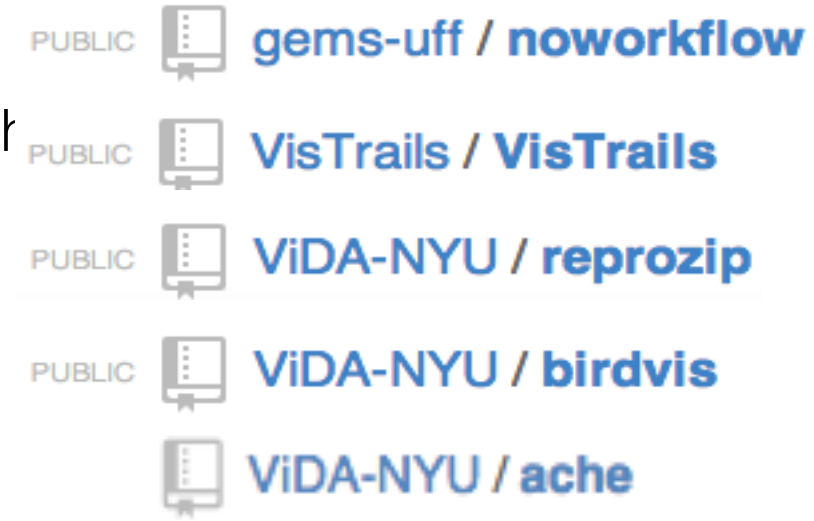
- 5 TT faculty members
- 2 Research faculty
- 6 Postdocs
- 4 Research engineers
- 20+ PhD students
- Many collaborators
- Constant stream of visiting researchers and students

Our Research at the ViDA Center

- Empower a *wide range of users* to *explore* the vast repositories of urban data
  - Data-savvy analysts, domain experts, policy makers and citizens
- Address issues at the different stages of the data lifecycle
- Key ingredients that we work on
  - Finding information on the Web – surface, hidden, and dark
  - Information integration
  - Data analysis
  - Visualization and visual analytics
  - Data and provenance management
- Focus on usability – tools must be powerful and easy to use
- From concepts and algorithms to deployed systems

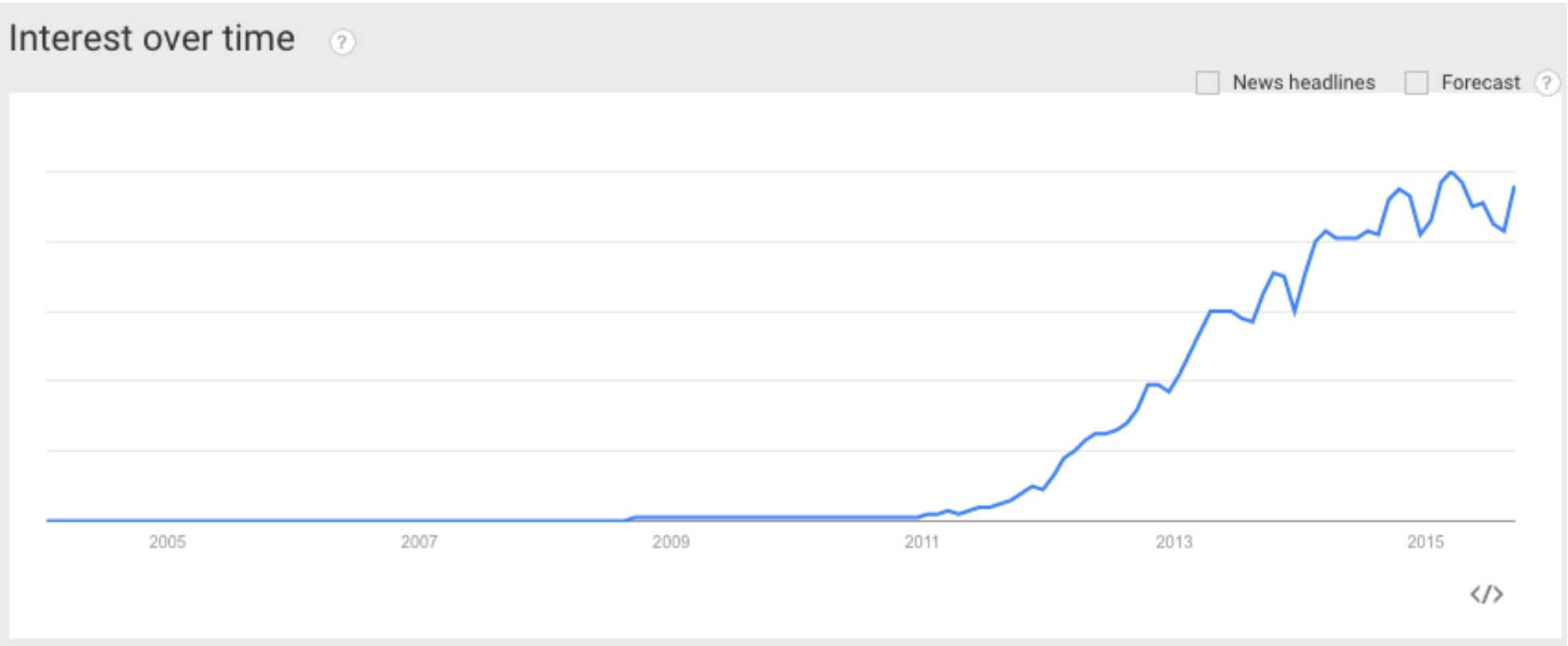
Our Research at the ViDA Center

- Empower a *wide range of users* to *explore* the vast repositories of urban data
  - Data-savvy analysts, domain experts, policy makers and citizens
- Address issues at the different stages of the data lifecycle
- Key ingredients that we work on
  - Finding information on the Web – surface, h
  - Information integration
  - Data analysis
  - Visualization and visual analytics
  - Data and provenance management



- Focus on usability – tools must be powerful and easy to use
- From concepts and algorithms to deployed systems

Big Data: What is the **Big** deal?

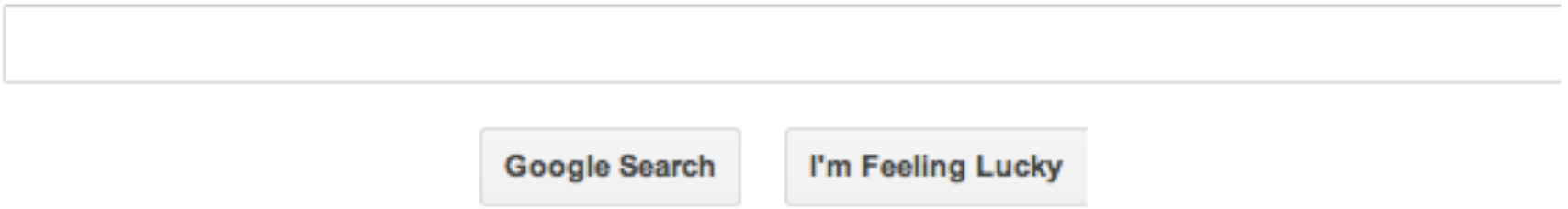


http://www.google.com/trends/explore#q=%22big%20data%22

Big Data: What is the **Big** deal?

Big Data: What is the **Big** deal?

- Many success stories
  - Google: many billions of pages indexed, products, structured data
  - Facebook: 1.1 billion users using the site each month
  - Twitter: 517 million accounts, **250 million tweets/day**
- This is changing society!



Big

- Many suc
  - Google: ma
  - Facebook:
  - Twitter: 51
- This is ch

Google grew from processing 100 TB of data a day with MapReduce in 2004 [45] to processing **20 PB a day** with MapReduce in 2008 [46]. In April 2009, a blog post1 was written about **eBay's two enormous data warehouses: one with 2 petabytes of user data, and the other with 6.5 petabytes of user data spanning 170 trillion records and growing by 150 billion new records per day**. Shortly thereafter, **Facebook** revealed2 similarly impressive numbers, boasting of **2.5 petabytes of user data, growing at about 15 terabytes per day.**

Lin and Dyer, 2010

Big Urban Data: What is the **Big** deal?

- Cities are the loci of economic activity
- 50% of the world population lives in cities --- by 2050 the number will grow to 70%
- Growth leads to problems, e.g., transportation, environment and pollution, housing, infrastructure
- Good news: Lots of data are being collected from traditional and *unsuspecting* sensors
  - Census, crime, emergency visits, taxis, public transportation, real estate, noise, energy, Twitter, …

Opportunity: Make cities more efficient and sustainable, and improve the lives of their residents

Big Urban Data: Success Stories

- Michael Flowers @ NYC

New York City gets 25,000 illegal-conversion complaints a year, but it has only 200 inspectors to handle them.

Flowers' group: (1) integrated information from 19 different agencies that provided indication of issues in buildings

- E.g., Late taxes, foreclosure proceedings, service cuts, ambulance visits, rodent infestation, crime

(2) Compared with 5 years of fire data

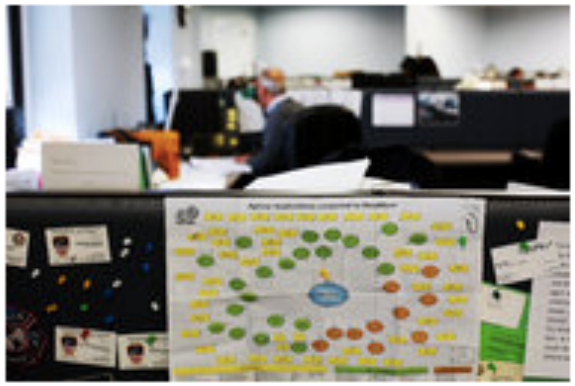
(3) Created a prediction system

Result: hit rate for inspections went from 13% to 70%

Todd Heisler/The New York Times

Michael Flowers, right, oversees a small group of tech-savvy and civic-minded statisticians working across from City Hall.

Enlarge This Image



Todd Heisler/The New York Times

"All we do," Mr. Flowers said, is "process massive amounts of information and use it to do things more effectively."

Big Data: New Opportunities

- *Enable scientific breakthroughs*
- Petabytes of data generated each day, e.g., Australian radio telescopes, Large Hadron Collider, Sloan Sky Survey, genomes, climate data, …
- Social data, e.g., Facebook, Twitter
  - 3,180,000 and 3,410,000 results in Google Scholar!

Big Data: What is the **Big** deal?

- Big data is not new: financial transactions, call detail records, astronomy, …
- What is new:
- Many more *data enthusiasts*
- More data are widely available, e.g., Web, data.gov, scientific data, social and urban data
- A lot of *variety*
- Computing is cheap and easy to access
  - Server with 64 cores, 512GB RAM ~$11k
  - Cluster with 1000 cores ~$150k
  - Pay as you go: Amazon EC2, Microsoft Azure

Big Data: What is hard?

- Rapidly-evolving technology, many different tools
- There is no one-size-fits-all solution: often, *you have to build your own*

  Need to understand the foundations
- Scalability can be challenging

  Learn techniques from distributed systems, parallel databases – do not reinvent the wheel!
- Leverage new architectures: clouds, GPUs, multiple cores

  Different computation models: need new algorithms
- Lots of diverse and dirty data

  Use and combine different cleaning and integration techniques

Platforms for Large-Scale Data Analysis

- Parallel DBMS technologies
  - Proposed in the late eighties
  - Matured over the last two decades
  - All of the goodies DBMS offer, e.g., relational data model, SQL, indexing, query optimization
  - Multi-billion dollar industry: Proprietary, expensive DBMS Engines intended as Data Warehousing solutions for very large enterprises
  - Run on expensive, reliable servers
- Map Reduce
  - Pioneered by Google, popularized by Yahoo! (open-source Hadoop)
  - Ecosystem of open-source, free tools
  - Run on cheap, commodity clusters – pay as you go
  - Scalability to large data volumes: divide and conquer (=data partitioning)
    - Scan 100 TB on 1 node @ 50 MB/sec = 23 days
    - Scan 100 TB on 1000-node cluster = 33 minutes
- DBMS techniques are being adopted in the Map Reduce ecosystem

Outline

- Introduction to database management systems
  - Why study databases?
  - Why use databases?
  - Relational Databases: Basic Concepts
  - The Relational Model and SQL
- Map-Reduce vs. Databases
- Large-scale data analysis on Map-Reduce/Hadoop

Why study databases?

- Databases used to be *specialized applications,* now they are a *central component* in computing environments
- Knowledge of database concepts is *essential* for computer scientists and for anyone who needs to *manipulate* data
- Databases are everywhere, even when you don't see them
  - Bank and credit card transactions, airline reservations, phone calls, Web sites, sales, NYU Albert, …
- Data needs to be managed!
- Data is valuable and needs to be protected
  - From failures and from people
- Data is often structured

Why study databases?

- Databases used to be *specialized applications,* now they are a *central component* in computing environments

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address
1	26361767	09/24/2013 02:37:36 AM	09/24/2013 03:09:54 AM	NYPD	New York City Police Depa	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	10304	645 BAY STREET
2	26361929	09/24/2013 02:34:20 AM	09/24/2013 03:17:14 AM	NYPD	New York City Police Depa	Blocked Driveway	No Access	Street/Sidewalk	11205	31 ELLIOTT PLACE
3	26366604	09/24/2013 02:20:22 AM	09/24/2013 02:32:46 AM	NYPD	New York City Police Depa	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	10012	620 BROADWAY
4	26362779	09/24/2013 02:14:43 AM	09/24/2013 03:15:10 AM	NYPD	New York City Police Depa	Blocked Driveway	Partial Access	Street/Sidewalk	11218	562 EAST 9 STREET
5	26364390	09/24/2013 02:10:33 AM		NYPD	New York City Police Depa	Blocked Driveway	Partial Access	Street/Sidewalk	11379	58-15 86 STREET
6	26366039	09/24/2013 02:04:47 AM		DOHMH	Department of Health and	Indoor Air Quality	Chemical Vapors/Gases/Odors	3+ Family Apartment Building	10031	541 WEST 142 STREET
7	26365752	09/24/2013 02:04:40 AM	09/24/2013 02:28:36 AM	NYPD	New York City Police Depa	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	10458	2357 CROTONA AVENUE
8	26364370	09/24/2013 02:03:19 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	Commercial Building	10460	605 MORRIS PARK AVENUE
9	26361893	09/24/2013 01:56:32 AM	09/24/2013 03:04:45 AM	NYPD	New York City Police Depa	Blocked Driveway	No Access	Street/Sidewalk	11370	20-15 74 STREET
10	26364368	09/24/2013 01:55:45 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	1-2 Family Dwelling	10460	584 MORRIS PARK AVENUE
11	26364997	09/24/2013 01:54:56 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	1-2 Family Dwelling	10460	582 MORRIS PARK AVENUE
12	26363537	09/24/2013 01:46:30 AM		NYPD	New York City Police Depa	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11421	87-06 77TH STREET
13	26367406	09/24/2013 01:42:14 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	3+ Family Mixed Use Building	10460	534 MORRIS PARK AVENUE
14	26361809	09/24/2013 01:40:34 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	3+ Family Mixed Use Building	10460	1750 VAN BUREN STREET
15	26362539	09/24/2013 01:38:57 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	Government Building	10460	461 MORRIS PARK AVENUE
16	26364157	09/24/2013 01:36:14 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	1-2 Family Dwelling	10460	518 MORRIS PARK AVENUE

- sites, sales, NYU Albert .
- Data ne naged!
- Data is v needs t
  - From fail ople
- Data is d ed

Steven Spielberg

From Wikipedia, the free encyclopedia

Steven Allan Spielberg (born December 18, 1946)[1] is an American film director, screenwriter, and film producer. In a career spanning six decades, Spielberg's films have taken up many themes and genres. Spielberg's early science-fiction and adventure films were seen as an archetype of modern Hollywood blockbuster filmmaking. In later years, his films began addressing such issues as the Holocaust, slavery, war and terrorism.

Spielberg won the Academy Award for Best Director for *Schindler's List* (1993) and *Saving Private Ryan* (1998). Three of Spielberg's films - *Jaws* (1975), *E.T. the Extra-Terrestrial* (1982), and *Jurassic Park* (1993) - achieved box office records, each becoming the highest-grossing film made at the

Steven Spielberg

Spielberg at the Pentagon (August 11, 1999).

Born	Steven Allan Spielberg December 18, 1946 (age 63) Cincinnati, Ohio, U.S.
Occupation	Film director, producer, screenwriter
Years active	1964–present
Spouse(s)	Amy Irving (m. 1985–1989) Kate Capshaw (m. 1991–present)

See larger photo

Hitachi Deskstar 7K500 - hard drive - 500 GB - SATA-300

$53 and up (6 stores) cashback · 2%

★★★★☆ user reviews (1)

The Hitachi Deskstar 7K500 hard disk drive extends the company's long-standing tradition of performance and reliability leadership. Hitachi's standardized features in desktop solutions enable fast transfer rates, low power utilization and quiet acoustics.... more...

Share Facebook Twitter Email

See also: **Product Summary** · Where to Buy · User Reviews · Expert Reviews · Specifications

WHERE TO BUY »

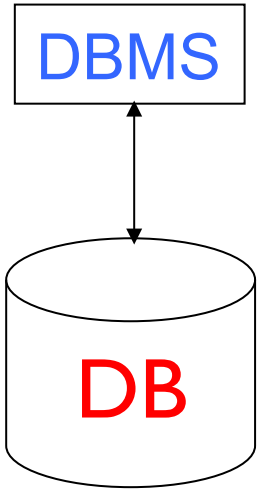
PRODUCT	SELLER	PRICE	
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	ServerSupply.com	**$53.00**	Go to store
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	ALLHDD.COM	**$64.00**	Go to store
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	Assembly Alliance Electronics	**$69.69**	Go to store

Why study databases?

- Databases used to be *specialized applications,* now they are a *central component* in computing environments
- Knowledge of database concepts is *essential* for computer scientists and for anyone who needs to *manipulate* data
- Databases are everywhere, even when you don't see them
  - Bank and credit card transactions, airline reservations, phone calls, Web sites, sales, NYU Albert, …
- Data needs to be managed!
- Data is valuable and needs to be protected
  - From failures and from people
- Data is often structured
  - We can exploit this regular structure to retrieve data in useful ways (that is, we can use a *query* language), and to store data efficiently

Database Management Systems

- Database (DB) is an integrated collection of data
  - Models real-world objects
    - Entities (e.g., people, residence, Christmas cards)
    - Relationships (e.g., John Doe lives on 123 Elm St)
  - Captures *structure* – allows data to be queried
- A Database Management System (DBMS) is a software suite designed to store and manage databases
  - Provides environment that is both *convenient* and *efficient* to use
  - Address many of the *complications* involved in managing data



Storing Data: Database vs. File System

- Once upon a time database applications were built on top of file systems, but this has many drawbacks:
  - Data redundancy, inconsistency and isolation: Multiple file formats, duplication of information in different files
  - Difficulty in accessing data: Need to write a new program to carry out each new task, e.g., search people by zip code or last name; update telephone number
  - Integrity problems: constraints (e.g., num_residence = 1) become part of program code -- hard to add new constraints or change existing ones
- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out, e.g., John and Mary get married, add new residence, update John's entry, and database crashes while Mary's entry is being updated…
- Concurrent access by multiple users
  - Needed for performance, but must be controlled, ow we can end up with inconsistencies

Database systems offer solutions to all these above problems

Why use Database Systems?

- Data independence and efficient access
  - Easy + efficient access through declarative query languages and optimization
- Data integrity and security
  - Safeguarding data from failures and malicious access
- Concurrent access
- Reduced application development time
- Uniform data administration
- *When should you not use a database?*

What's in a DBMS?

Data model	*Query language*	*Transactions and crash recovery*
Logical DB design Relational Model XML data model	SQL, QBE, views XPath, XQuery	Transactions
Map data to files Clustering Indexes	Query optimization Query evaluation	Locking Concurrency control Recovery Logs

"above the water"

"below the water"

Designing a database: The Conceptual Model

- What are the *entities* and *relationships* among these entities in the application?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hold?
- Different applications have different needs, and different perspectives – even to model the *same* object

billing department: patient(id, name, insurance, address)
visit(patientId, procedure, date, charge)

inpatient: patient(id,name,age,address)

alergies(id,alergies)

prescription(patientId,date,medicine)

Designing a database: The Conceptual Model

- What are the *entities* and *relationships* among these entities in the application?
- What information about these entities and relationships should we store in the database?
- Wha[illegible] hat hold[illegible]
- Diffe[illegible] rent perspectives – even to model the *same* object

Requires a good understanding of the *semantics* of the application

billing department: patient(id, name, insurance, address)
visit(patientId, procedure, date, charge)

inpatient: patient(id,name,age,address)
alergies(id,alergies)
prescription(patientId,date,medicine)

Database Design: Desiderata

- Find a "good" collection of relation schemas
- Design Goals:
  - Avoid redundant data and inconsistencies
  - Ensure that relationships among attributes are represented – you can get answers for your questions!
  - Ensure constraints are properly modeled: updates check for violation of database integrity constraints

Query Languages

- *Query languages:* Allow *manipulation* and *retrieval* of data from a database
- Queries are posed wrt data model
  - Operations over objects defined in data model
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for optimization
- Query Languages != programming languages
  - QLs support easy, efficient access to large data sets
  - QLs not expected to be "Turing complete"
  - QLs not intended to be used for complex calculations

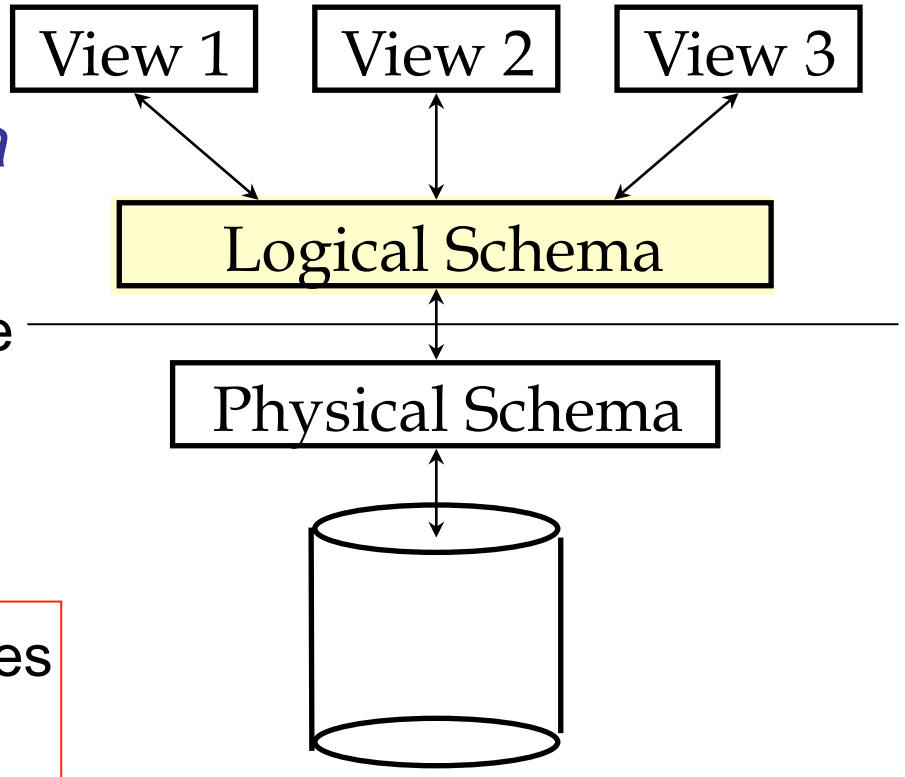
Query Languages

- *Query languages:* Allow *manipulation* and *retrieval* of data from a database
- Queries are posed wrt data model
  - Operations over objects defined in data model
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for optimization
- Query Languages != programming lar
  - QLs support easy, efficient access to large
  - QLs not expected to be "Turing complete"
  - QLs not intended to be used for complex calculations

a language that can compute anything that can be computed

Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*
  - Views describe how users see the data
  - Logical schema defines logical structure
  - Physical schema describes the files and indexes used



Key to good performance

Example: University Database

- Physical schema:
  - Students stored in id order
  - Index on last name
- Logical schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence

- Applications insulated from how data is structured and stored
- *Logical data independence*: Protection from changes in *logical* structure of data
  - Changes in the logical schema do not affect users as long as their views are still available
- *Physical data independence:* Protection from changes in *physical* structure of data
  - Changes in the physical layout of the data or in the indexes used do not affect the logical relations

One of the most important benefits of using a DBMS!

Storage and Indexing

- The *DB administrator* designs the physical structures
- Nowadays, database systems can do (some of) this automatically: autoadmin, index advisors
- File structures: sequential, hashing, clustering, single or multiple disks, etc.
- Indexes:
  - Select attributes to index
  - Select the type of index
- Storage manager is a module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

Query Optimization and Evaluation

- DBMS must provide efficient access to data
- Declarative queries are translated into imperative query plans
  - Declarative queries → logical data model
  - Imperative plans → physical structure
- Relational optimizers aim to find the best imperative plans (i.e., shortest execution time)
  - In practice they avoid the worst plans…

Database Model

- Provides the means for
  - *specifying particular data structures*
  - *constraining the data sets* associated with these structures, and
  - *manipulating* the data
- Data *definition* language (DDL): define structures and constraints
- Data *manipulation* language (DML): specify manipulations/ operations over the data
- There are different models, e.g., Relational (sets), XML (trees), object oriented, …

Relational Model

- A relation is a set of tuples/records

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

- Queries consist of operations that manipulate sets
- Queries are written wrt the relation, but the relation can be stored in many different ways: Physical independence!

Example: Relation Schema

Constraints → *balance>=0*

Number is a key

The schema for the relation

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Account(Number, Owner, Balance, Type)

The schema sets the structure of the relation---it is the *definition* of the relation.

(Note: the schema specifies more information than what is shown, e.g., keys, constraints…)

Relational Database

- A database consists of multiple relations
- Information is broken up---each relation store one part of the information
  E.g.: *account* : information about accounts
  deposit: information about deposits into accounts
  check : information about checks
- What would happen if we stored all information in a single table?
  - repetition of information (e.g., two customers own an account, two deposits on the same account)
  - the need for null values (e.g., represent a customer without an account)

To avoid these problems we *normalize* databases

Relational Database Example

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00

Check	Account	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Relational Database Example (cont.)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checki
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	saving
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/2/00	10,000.00

Check	Account	Check-number	Date	Amount
	101	924	10/23/98	125.00
	101	925	10/24/99	23.98

Each Relation has a key…. where the values must be unique.

Querying with SQL

- SQL is a high-level language
  - Say “what to do” rather than “how to do it”
  - Avoid a lot of data-manipulation details needed in procedural languages like C++, Python, Java
- Database management system figures out “best” way to execute query
  - Called “query optimization”

SQL in Action: Find tuples that satisfy a condition

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/1/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/1/2000 10:05:00
4	2	100	$40.00	11/1/2000 10:07:00
5	2	100	$200.00	11/8/2000 14:14:00

Attributes of the resulting relation

Relation to which the query refers

```
SELECT  *
FROM    ATMWithdrawal
WHERE   Amount < 50;
```

Condition that must be satisfied

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/1/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/1/2000 10:05:00
4	2	100	$40.00	11/1/2000 10:07:00
5	2	100	$200.00	11/8/2000 14:14:00

```
SELECT  *
FROM    ATMWithdrawal
WHERE  Amount < 50;
```

Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/1/2000 9:45:00
3	2	101	$40.00	11/1/2000 10:05:00
4	2	100	$40.00	11/1/2000 10:07:00

SQL in Action: Eliminating Attributes

TransactionID		CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/1/2000	9:45:00
3	2	101	$40.00	11/1/2000	10:05:00
4	2	100	$40.00	11/1/2000	10:07:00

```
SELECT AcctNo, Amount
FROM   ATMWithdrawal
WHERE  Amount < 50;
```

Consider the attributes listed in the SELECT clause.

Throw away attributes that are not listed.

Thus the final query answer is:

Final Query Answer table

AcctNo	Amount
102	$25.00
101	$40.00
100	$40.00

SQL in Action: Combining Tables

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/2/00	10,000.00

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.Account = A.Number and A.Balance > 1000;
```

SQL in Action: Combining Tables

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105			

We must check every combination of one row from Account with one row from Deposit!

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/2/00	10,000.00

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.Account = A.Number and A.Balance > 1000;
```

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/2/00	10,000.00

WHERE D.Account = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	Account	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/2/00	10,000.00

Comments on Queries

Because the answer to a relational query is always a **table**

.....

we can use the answer from one query as input to another query.

This means that we can create arbitrarily complex queries!

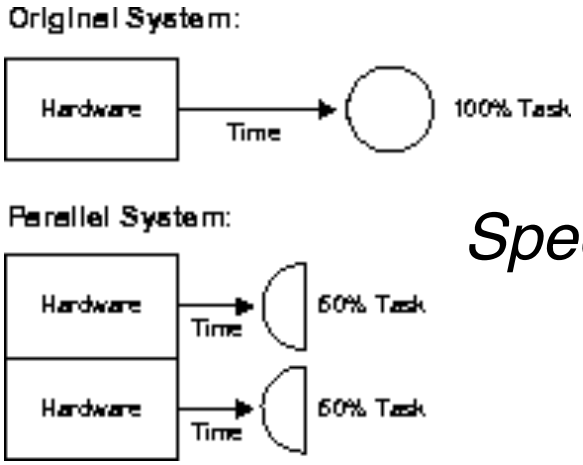
We say that relational query languages are **closed** when they have this property.

What happens when we have too much data?

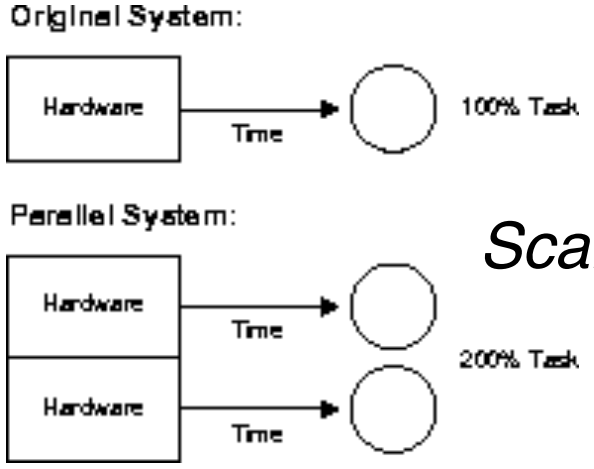
Parallel DBMSs

- Aim to improve performance by executing operations in parallel
- Benefit: easy and cheap to scale
  - Add more nodes, get higher speed
  - Reduce the time to run queries
  - Higher transaction rates
  - Ability to process more data
- Challenge: minimize overheads and efficiently deal with contention

docs.oracle.com



Speedup

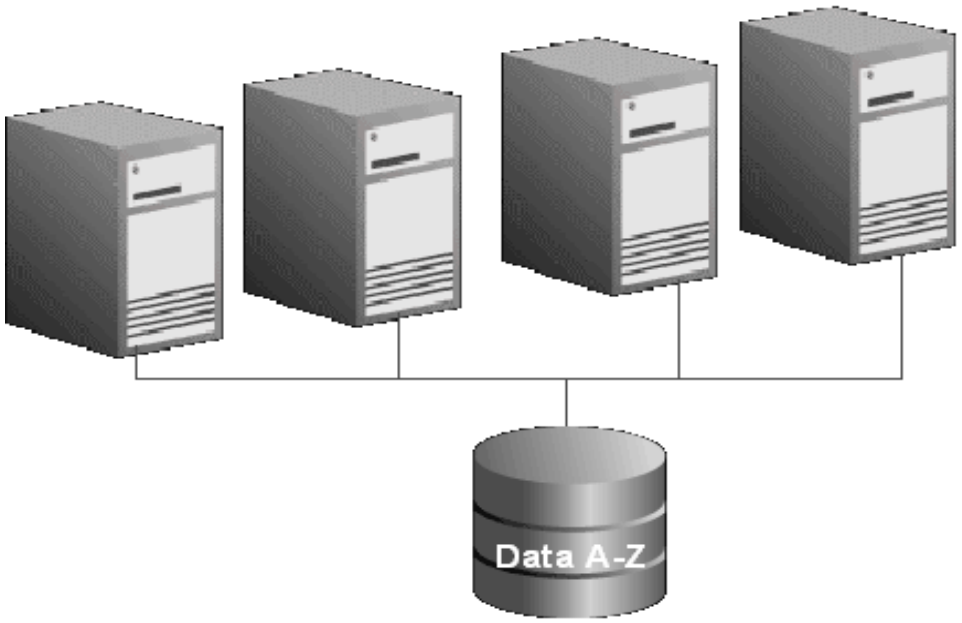


Scaleup

Different Architectures

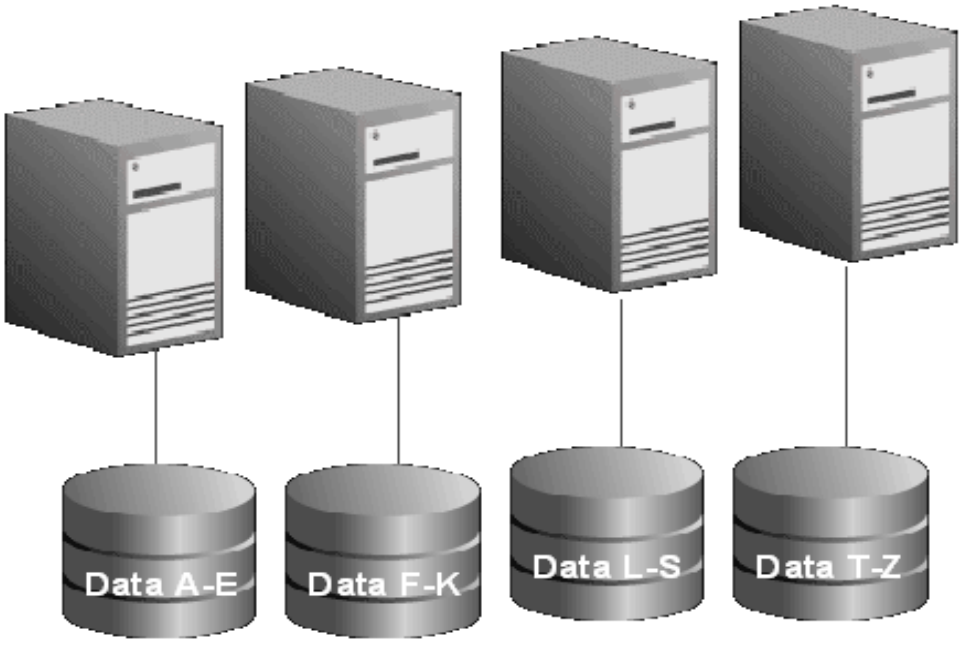
Shared Everything

No Data partitioning Required

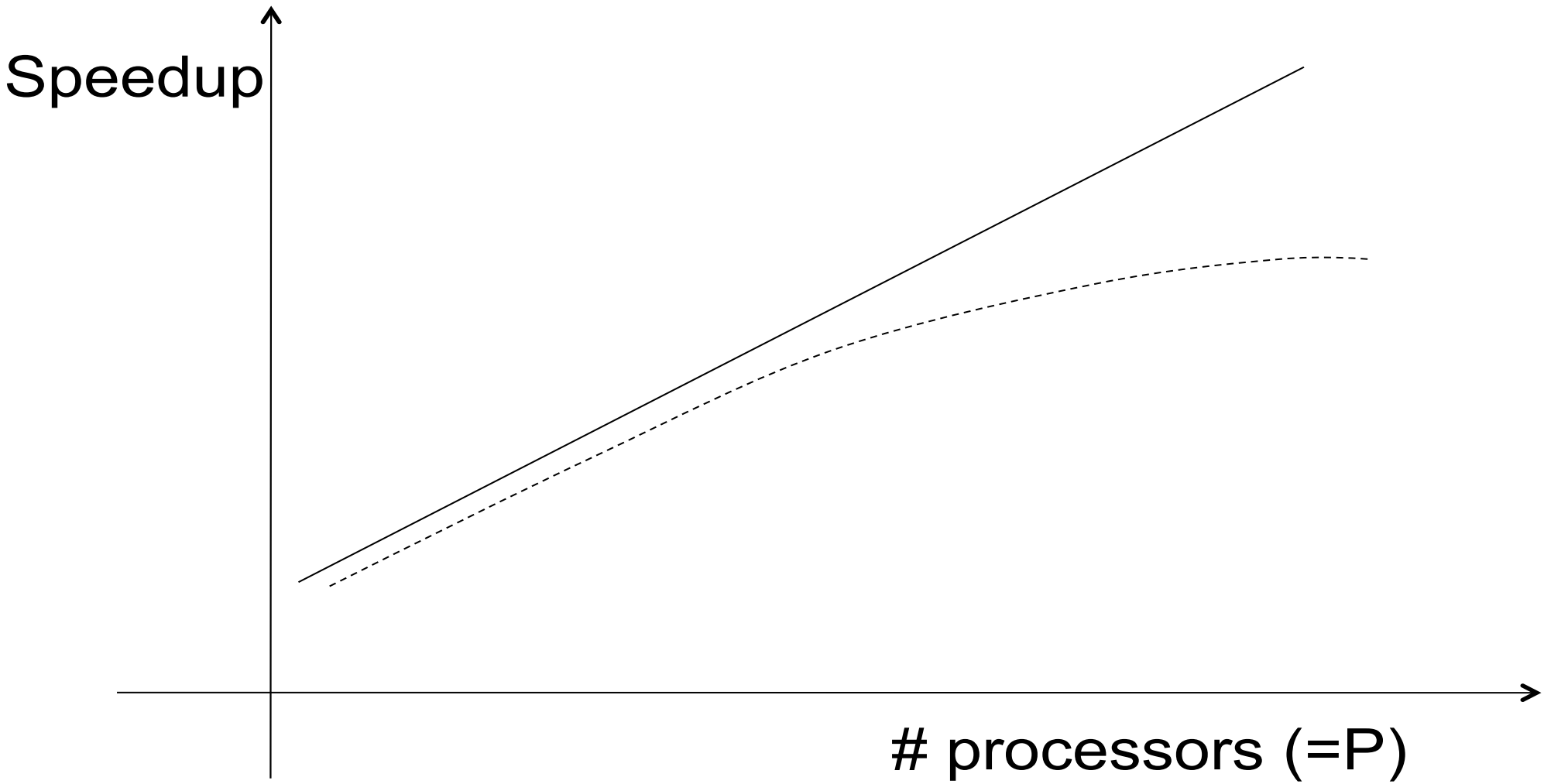


Shared Nothing

Static Data partitioning is a prerequisite



Linear vs. Non-Linear Speedup



Achieving Linear Speedup: Challenges

- Start-up cost: starting an operation on many processors
- Contention for resources between processors
- Data transfer
- Slowest processor becomes the bottleneck
- Data skew → load balancing
- Shared-nothing:
  - Most scalable architecture
  - Minimizes resource sharing across processors
  - Can use commodity hardware
  - Hard to program and manage

Any feeling of déjà vu?

What to Parallelize?

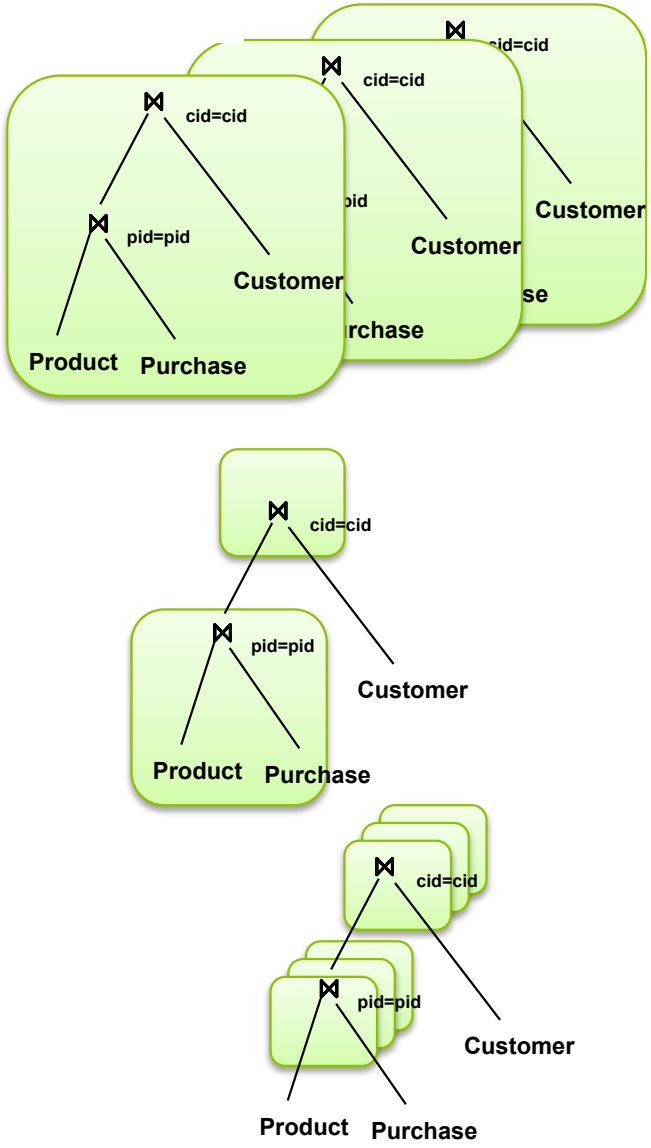
- Inter-query parallelism
  – Each query runs on one processor

- Inter-operator parallelism
  – A query runs on multiple processors

an operator runs on one processor

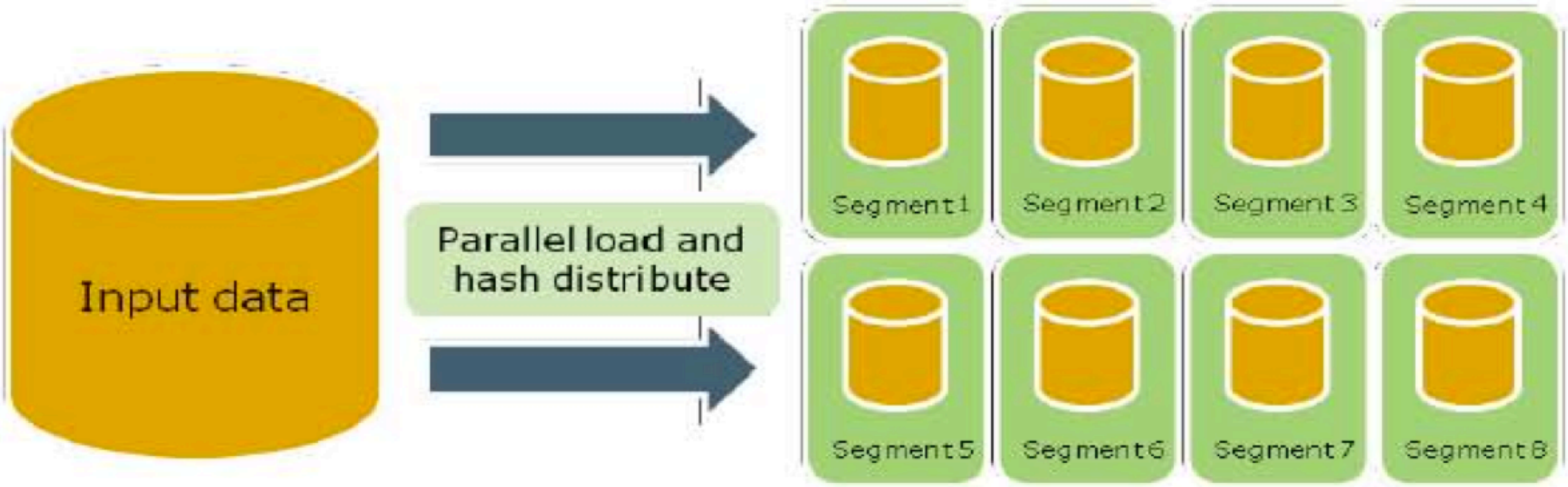
- Intra-operator parallelism
  – An operator runs on multiple

processors



Partitioning Data

Figure 3 - Automatic hash-based data distribution



From: Greenplum Database Whitepaper

Partitioning a Relation across Disks

- If a relation contains only a few tuples which will fit into a single disk block, then assign the relation to a single disk
- Large relations are preferably partitioned across all the available disks
- The distribution of tuples to disks may be **skewed** — that is, some disks have many tuples, while others may have fewer tuples
- Ideal: partitioning should be balanced

Parallel DBMS technologies

- Popularly used for more than two decades
  - Research Projects: Gamma, Grace, …
  - Commercial: Multi-billion dollar industry *but access to only a privileged few*
- Relational Data Model
- Indexing
- Familiar SQL interface
- Advanced query optimization
- *Well understood and studied*
- *Very reliable*
- *Database system takes care of everything!*

Fast forward to 2010…

Motivation: Indexing the Web

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web
- ~1,000 hard drives to store the web
- Takes even more to **do** something useful with the data!
- **A standard architecture for such problems has emerged**
  - Cluster of commodity Linux nodes
  - Commodity network (ethernet) to connect them
- **MapReduce is born!**

MapReduce

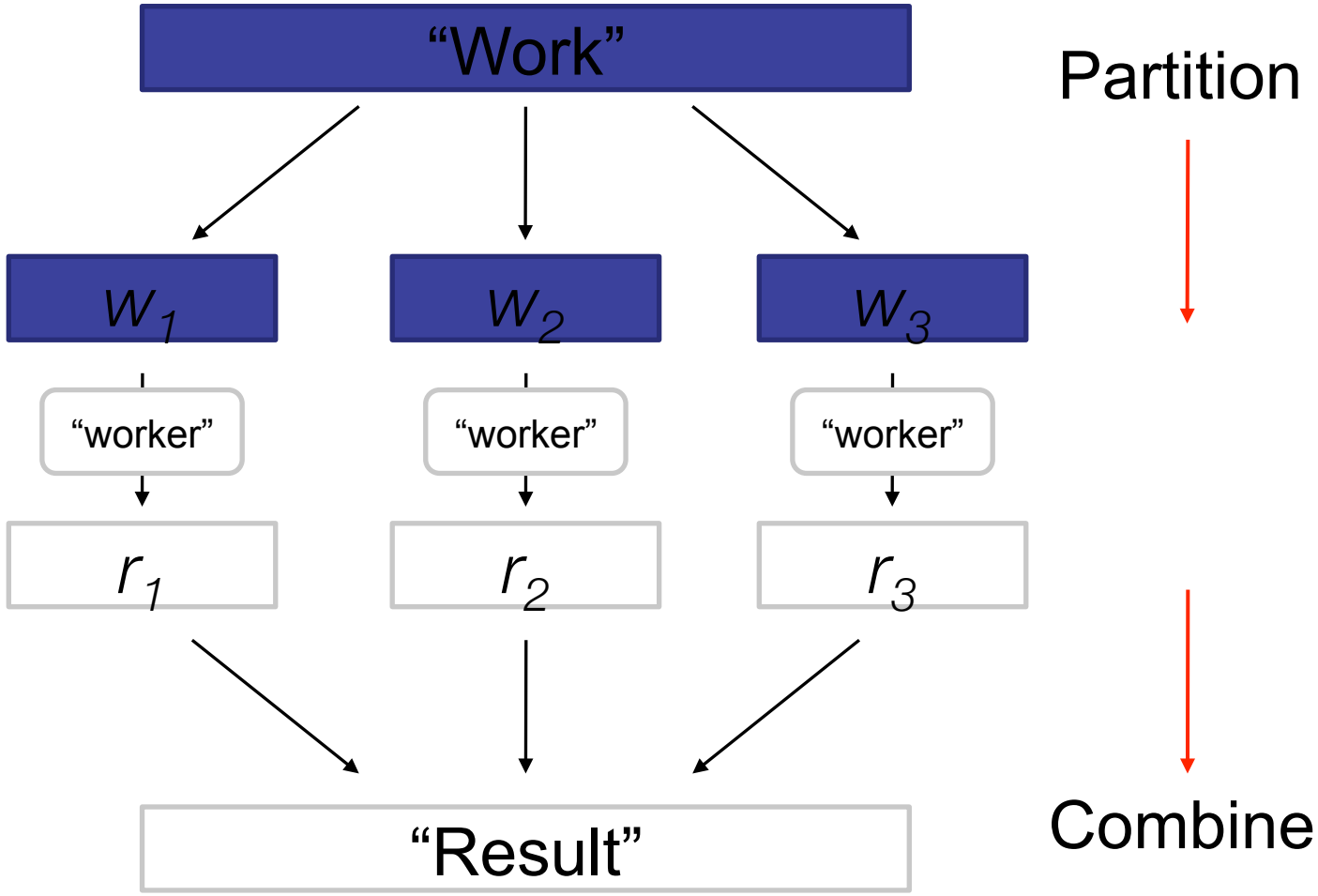
- Overview:
  - Data-parallel *programming model*
  - An associated parallel and distributed implementation for commodity clusters
- Pioneered by Google
  - Processing 20 PB of data per day (circa 2008)
- Popularized by open-source Hadoop project
  - Used by Yahoo!, Facebook, Amazon, and the list is growing …

[Dean et al., OSDI 2004, CACM Jan 2008, CACM Jan 2010]

Hadoop

- Open source of MapReduce framework of Apache Project
- Key components
  - MapReduce - distributes applications
  - Hadoop Distributed File System (HDFS) - distributes data
- Hadoop Distributed File System (HDFS)
  - Store big files across machines
  - Store each file as a sequence of blocks
  - Blocks of a file are replicated for fault tolerance
- Distribute processing of large data across thousands of commodity machines
- You have to *program* your data processing and analysis

Divide and Conquer



Big Ideas: Failures are Common

- Suppose a cluster is built using machines with a *mean-time between failures (MTBF) of 1000 days*
- For a 10,000 server cluster, there are on average 10 failures per day!
- MapReduce implementation cope with failures
  - Automatic task restarts
  - Store files multiple times for reliability

Big Ideas: Move Processing to Data

- Supercomputers often have processing nodes and storage nodes
  - Computationally expensive tasks
  - High-capacity interconnect to move data around
- Many data-intensive applications are not very processor-demanding
  - Data movement leads to a bottleneck in the network!
  - New idea: move processing to where the data reside
- In MapReduce, processors and storage are co-located
  - Leverage locality

Big Ideas: Avoid Random Access

Example:

- 1 TB database containing 10^{10} 100 byte records
- Random access: each update takes ~30ms (seek, read, write)
  - Updating 1% of the records takes ~35 days
- Sequential access: 100MB/s throughput
  - Reading the whole database and rewriting all the records, takes 5.6 hours

- MapReduce was designed for batch processing --- organize computations into long streaming operations

Big Ideas: Abstract System-Level Details

- Developing distributed software is very challenging
  - Manage multiple threads, processes, machines
  - Concurrency control and synchronization
  - Race conditions, deadlocks – a nightmare to debug!
- MapReduce isolates developers from these details
  - Programmer defines *what* computations are to be performed
  - MapReduce execution framework takes care of *how* the computations are carried out

MapReduce

- Programmers specify two functions:

  map $(k_1, v_1) \rightarrow [(k_2, v_2)]$

  reduce $(k_2, [v_2]) \rightarrow [(k_3, v_3)]$

  - All values with the same key are sent to the same reducer
- The execution framework handles everything else…

Word Count in Python

```
def word_count_dict(filename):
  """Returns a word/count dict for this filename."""
  # Utility used by count() and Topcount().
  word_count = {} # Map each word to its count
  input_file = open(filename, 'r')
  for line in input_file:
    words = line.split()
    for word in words:
      word = word.lower()
      # Special case if we're seeing this word for the first time.
      if not word in word_count:
        word_count[word] = 1
      else:
        word_count[word] = word_count[word] + 1
  input_file.close() # Not strictly required, but good form.
  return word_count
```

Word Count in MapReduce

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);
```

k_1 v_1 k_2 v_2

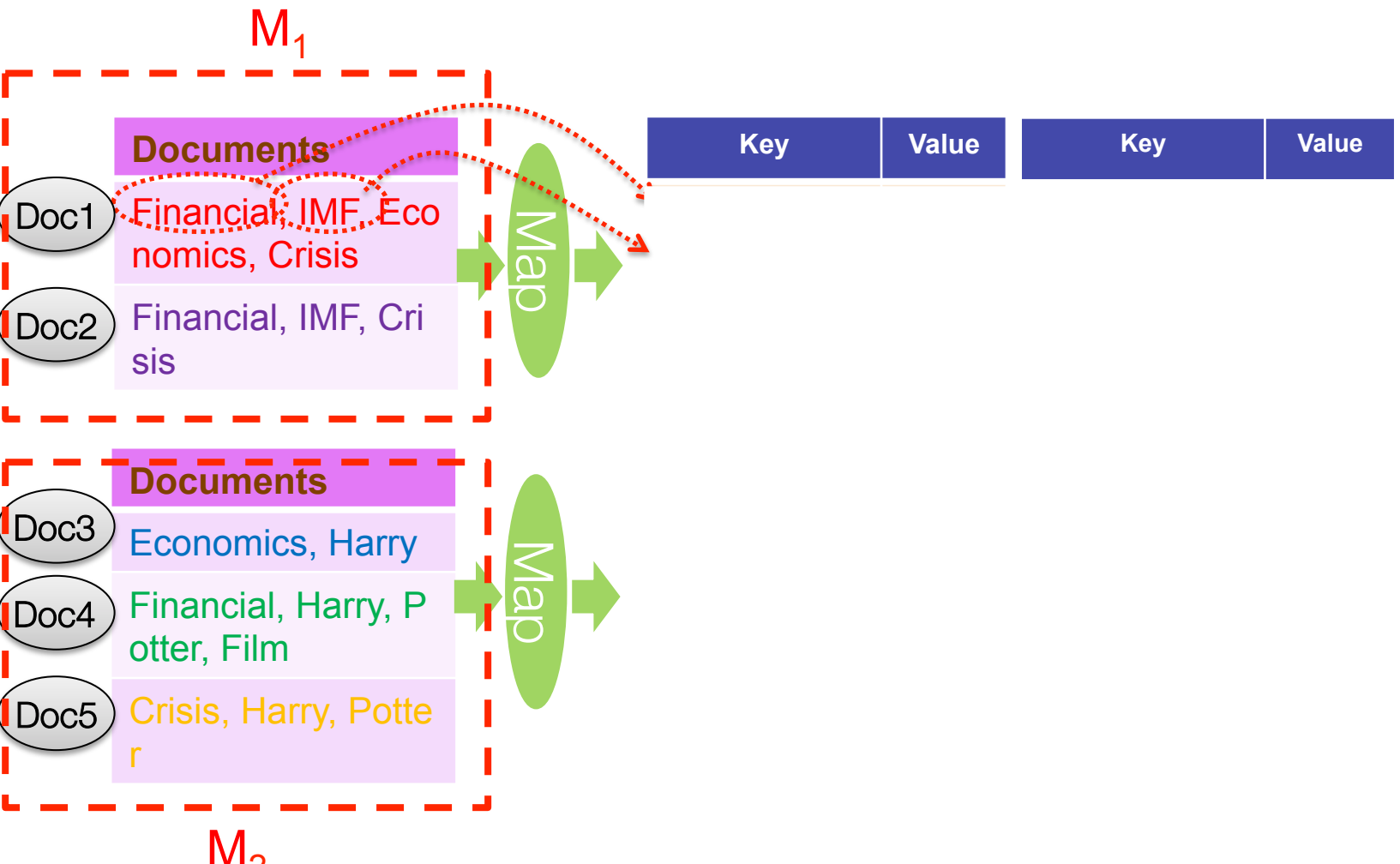
map (docid, text) → [(word, 1)]

```
Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
        Emit(term, value);
```

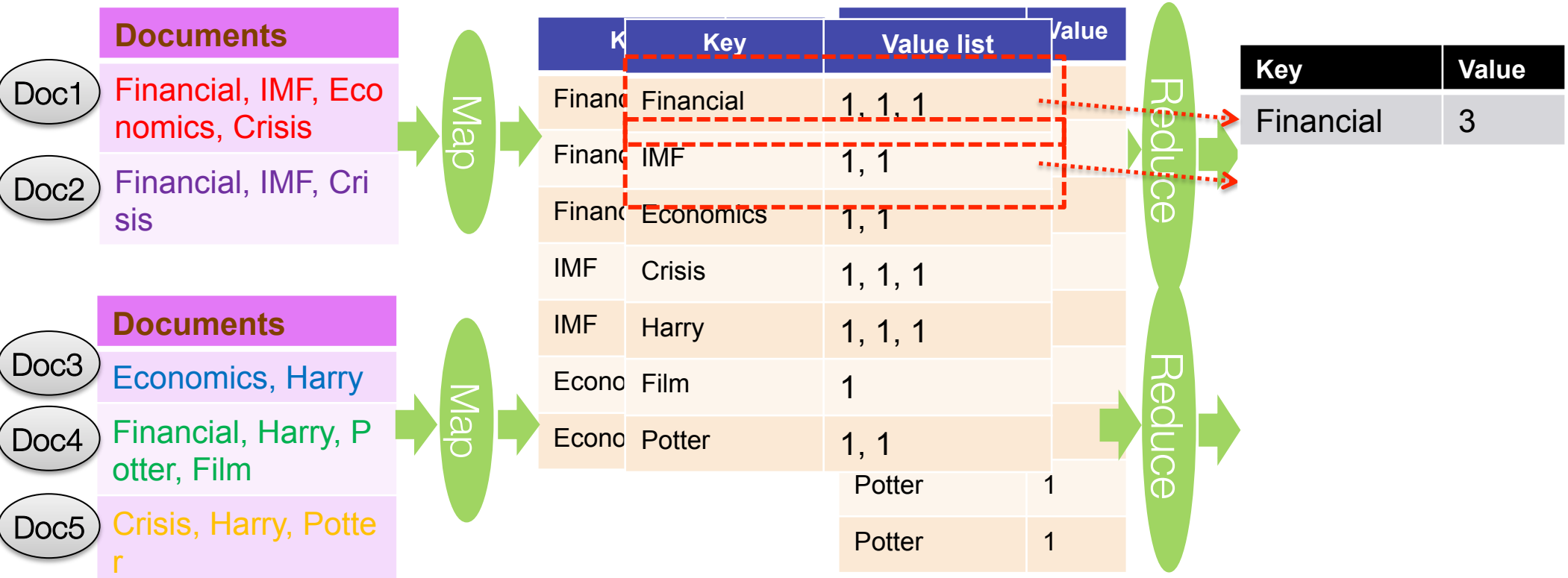
k_2 $[v_2]$ k_3 v_3

reduce (word, [1, …, 1]) → [(word, count)]

Word Count with MapReduce



Word Count with MapReduce



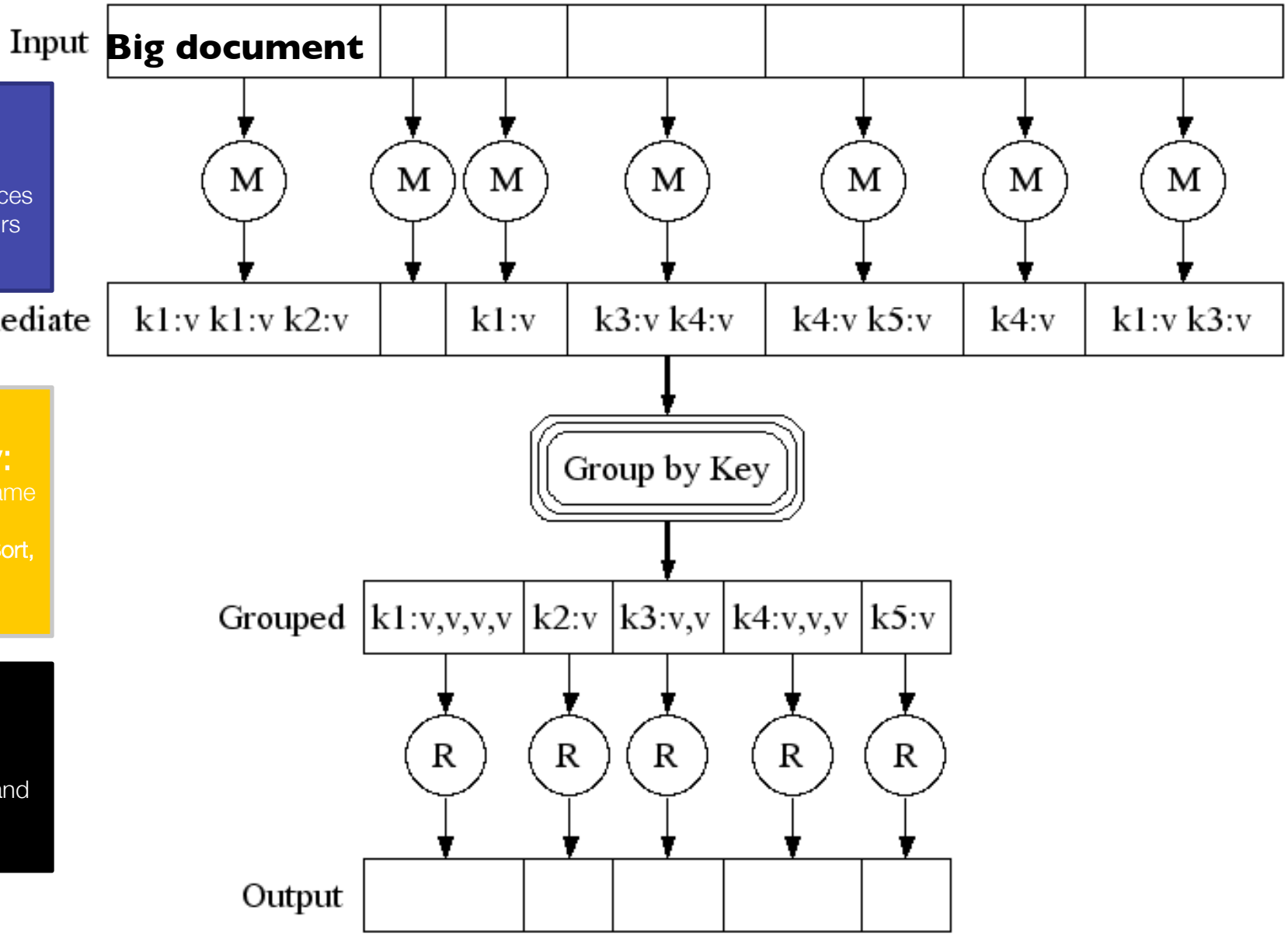
Before reduce functions are called,
for each distinct key, the list of its values is generated

Map-Reduce: A Diagram

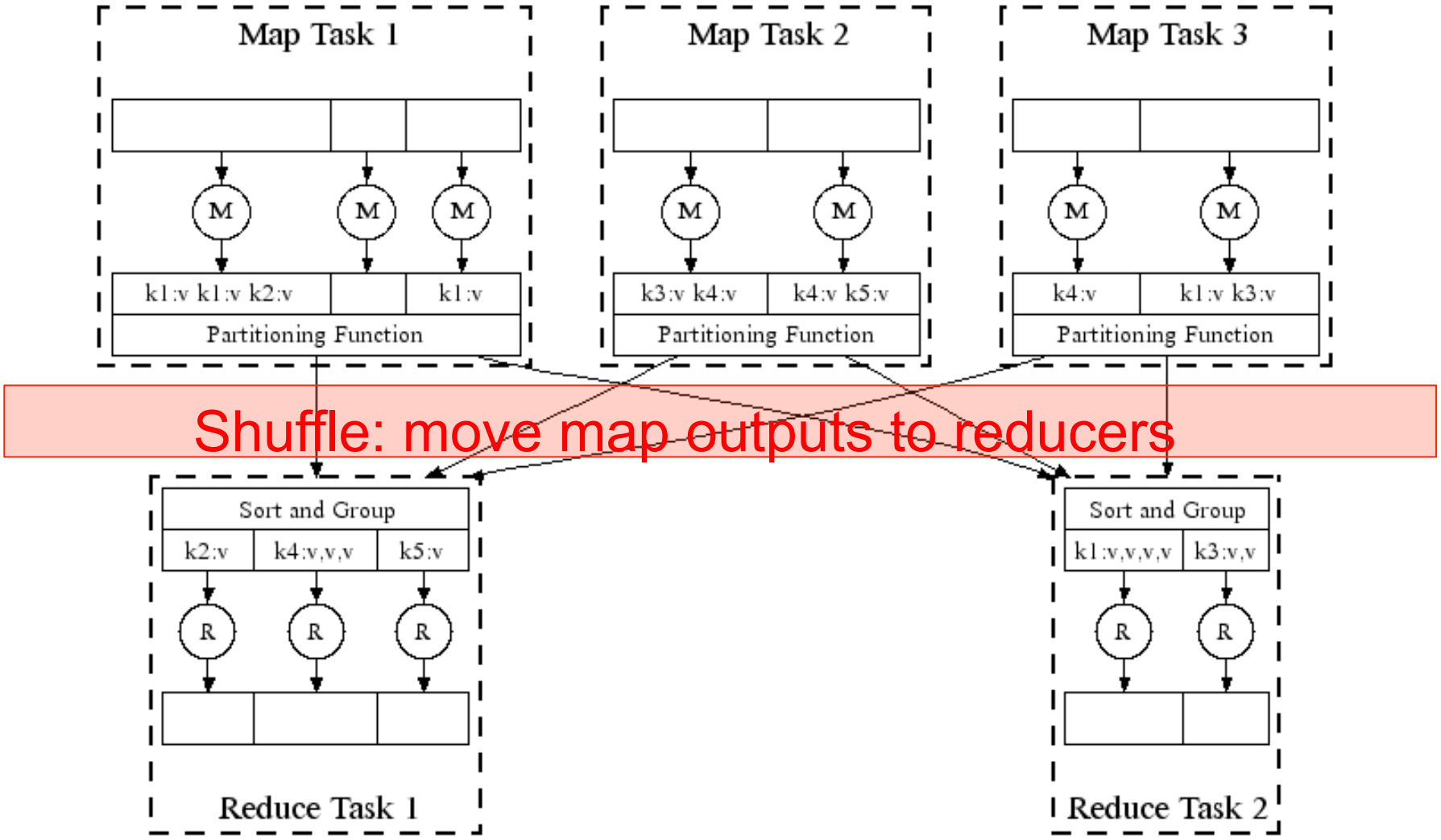
MAP:
Read input and produces a set of key-value pairs

Group by key:
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)

Reduce:
Collect all values belonging to the key and output



Map-Reduce: In Parallel



All phases are distributed with many tasks doing the work

MapReduce “Runtime”

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles “data distribution”
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a *distributed filesystem*

MapReduce: Data Flow

- *Input and final output* are stored on the *distributed file system* (DFS)
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data

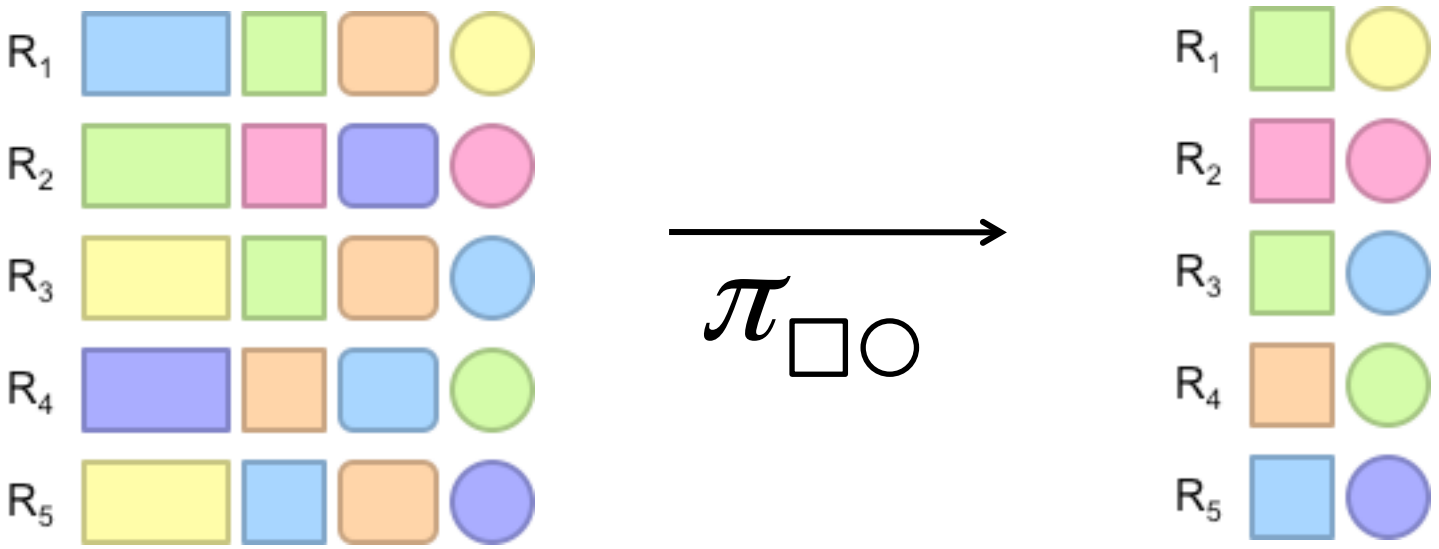
    You can specify a directory where your input files reside using *MultipleInputs.addInputPath*
- Typical usage pattern:
  - Huge files (100s of GB to TB)
  - Data rarely updated in place – mostly reads and appends
- *Intermediate results* are stored on *local FS* of Map and Reduce workers
- Output is often input to another MapReduce task

Designing Algorithms for MapReduce

- Need to adapt to a restricted model of computation
- Goals
  - Scalability: adding machines will make the algorithm run faster
  - Efficiency: resources will not be wasted
- The translation some algorithms into MapReduce isn't always obvious
- But there are useful *design patterns* that can help
  - See Data-Intensive Text Processing with MapReduce, by Lin and Dyer http://lintool.github.io/MapReduceAlgorithms/

Map Reduce Algorithms for Relational Data

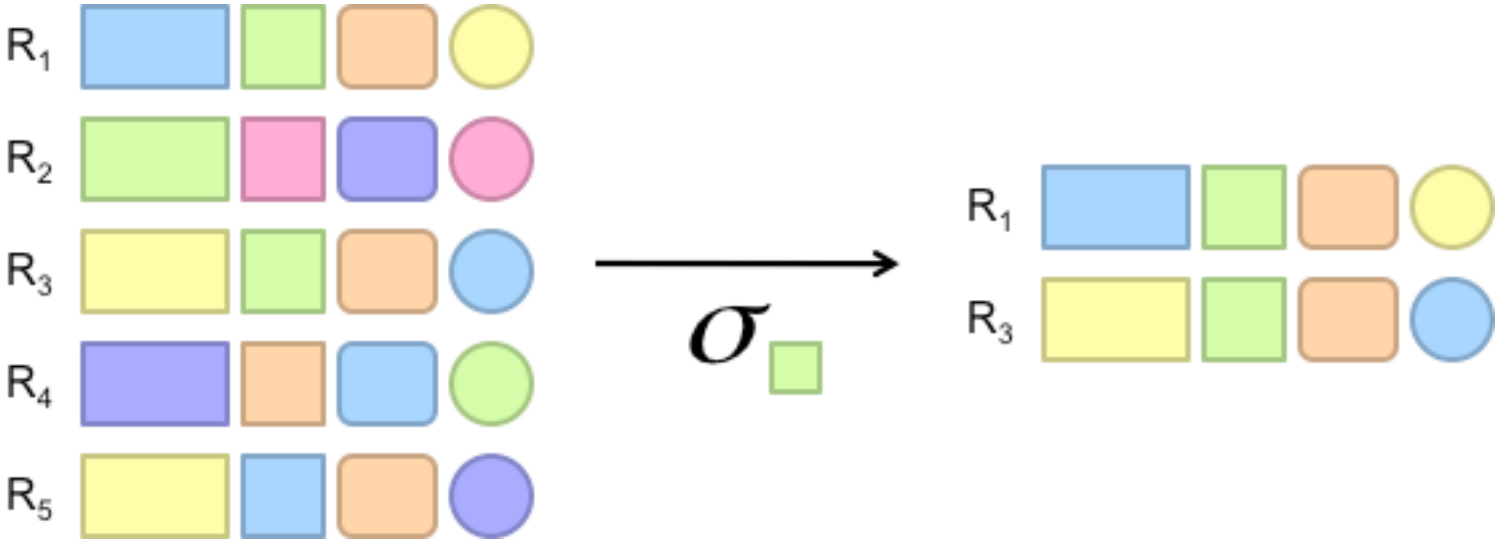
Projection



Projection in MapReduce

- Easy
  - Map over tuples, emit new tuples with appropriate attributes
  - No reducers, unless for regrouping or re-sorting tuples
  - Alternative: do projection in reducer, after some other processing
- Limited by HDFS streaming speeds
  - Speed of encoding/decoding tuples becomes important
  - Semi-structured (XML) data? No problem!

Selection



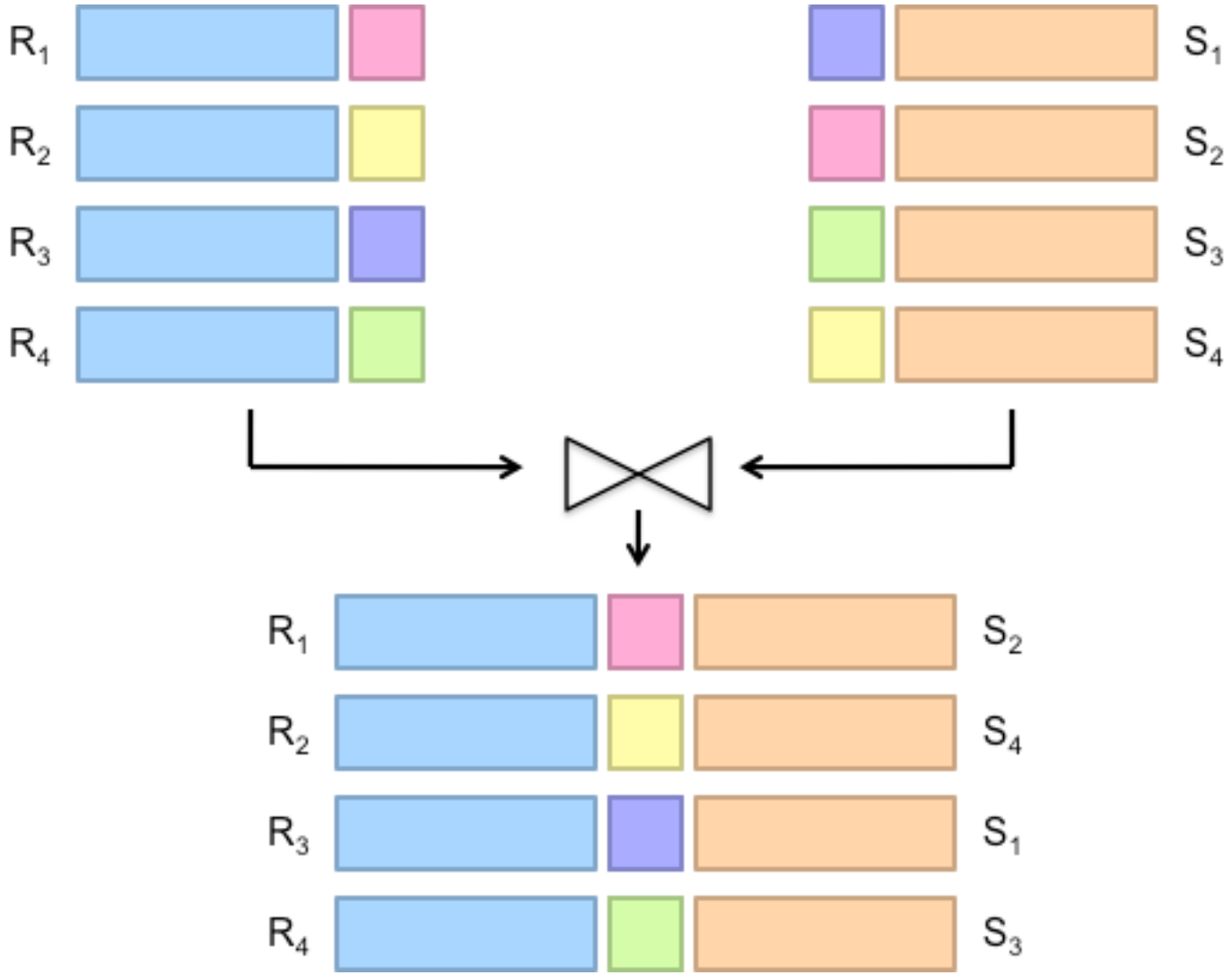
Selection in MapReduce

- Easy
  - Map over tuples, emit new tuples with appropriate attributes
  - No reducers, unless for regrouping or re-sorting tuples
  - Alternative: do projection in reducer, after some other processing
- Limited by HDFS streaming speeds
  - Speed of encoding/decoding tuples becomes important
  - Semi-structured (XML) data? No problem!

Aggregation in MapReduce

- Example – Log analysis: What is the average time spent per URL?
- In SQL:
  - SELECT url, AVG(time) FROM visits GROUP BY url
- In MapReduce:
  - Map over tuples, emit time, keyed by url
  - Framework *automatically* groups values by keys
  - Compute average in reducer

Relational Joins

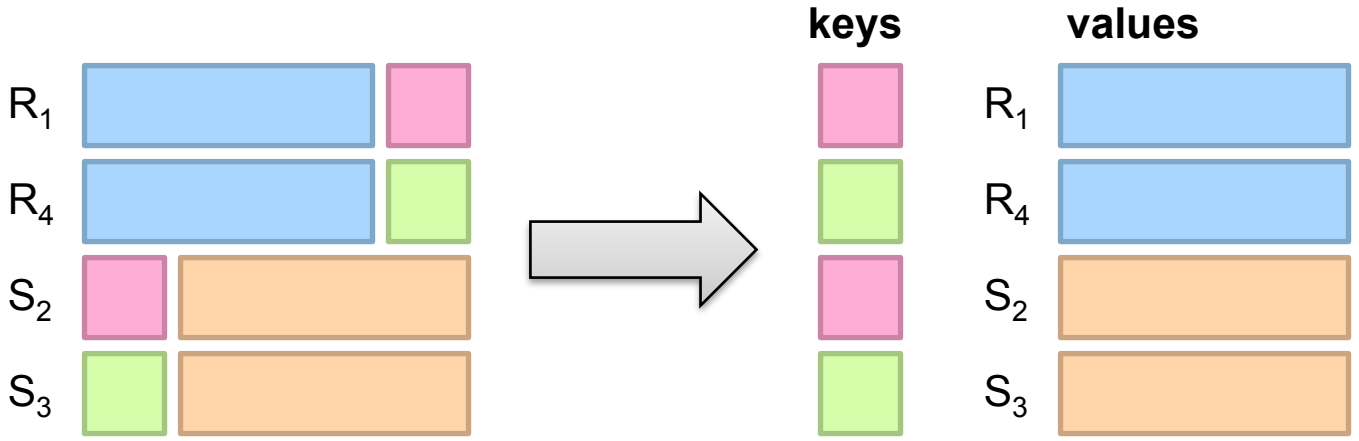


Reduce-Side Join

- Basic idea: *group by join key*
  - Map over both sets of tuples
  - Emit tuple as value with **join key as the intermediate key**
  - Execution framework brings together tuples sharing the same key
  - Perform actual join in reducer
  - Similar to a “sort-merge join” in database terminology

Reduce-Side Join: 1-to-1

Map



Reduce

There are many different join algorithms for Map Reduce!

MapReduce vs. Databases

- On MapReduce, you have to program everything!
  - A “major step backwards” http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html

Architectural Elements: ParDB vs. MR

- Schema support:
  - Relational paradigm: rigid structure of rows and columns
  - Flexible structure, but need to write parsers and challenging to share results
- Indexing
  - B-trees to speed up access
  - No built-in indexes --- programmers must code indexes
- Programming model
  - Declarative, high-level language
  - Imperative, write programs
- Data distribution
  - Use knowledge of data distribution to automatically optimize queries
  - Programmer must optimize the access

MapReduce vs. Parallel Databases

- MapReduce provides storage independence and fine-grained fault tolerance
- Supports complex transformations
- Production environments use a plethora of storage systems: files, RDBMS, Bigtable, column stores
- MapReduce can be extended to support different storage backends --- it can be used to combine data from different sources
- Parallel databases require all data to be loaded
  - Would you use a ParDB to load Web pages retrieved by a crawler and build an inverted index?
- Techniques used by DBMSs can also be applied to MapReduce, e.g., indices

[Dean and Ghemawat, CACM 2010]

MapReduce vs. Parallel Databases

- MapReduce was designed for *complex* tasks that manipulate diverse data:
  - Extract links from Web pages and aggregating them by target document
  - Generate inverted index files to support efficient search queries
  - Process all road segments in the world and rendering map images
- These data do not fit well in the relational paradigm
  - Remember: SQL is not Turing-complete!

[Dean and Ghemawat, CACM 2010]

MapReduce and Databases: Towards Convergence

- DB features are being added to the MapReduce environment
  - Pig (https://pig.apache.org): high-level language for data analysis that runs on top of Hadoop
  - HadoopDB: hybrid of DBMS and MapReduce technologies that targets analytical workloads
  - Hive (https://hive.apache.org): data warehouse for querying and managing large data in distributed storage
  - ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud
  - …
- And alternative systems have been proposed
  - Spark and Spark SQL: support for streaming and data analysis

[Dean and Ghemawat, CACM 2010]

Conclusions

- Analyzing/processing large volumes of data is hard
- There are many tools/techniques – it is challenging to select the right combination for a given problem
- And sometimes you have to implement your own…
- Knowledge of database concepts is essential!
- Visualization is a powerful tool for data exploration, in particular for large data
  - Pictures help us think – substitute perception for cognition

Merci

Obrigada

धन्यवाद

Thank you

Tack

Danke

Ευχαριστω

NYU POLYTECHNIC SCHOOL OF ENGINEERING

CUSP CENTER FOR URBAN SCIENCE+PROGRESS

Some Links

- GPU-based spatio-temporal index integrated with MongoDB: https://github.com/harishd10/mongodb
- VisTrails -- a workflow-based data exploration system that supports reproducibility: http://www.vistrails.org
- Other tools developed by the NYU VIDA team: https://github.com/ViDA-NYU
- rcloud: https://github.com/att/rcloud