




Inspire…Educate…Transform.

Engineering Big Data

Session 2: Computing @ Scale

Dr. Sreerama KV Murthy
CEO, Quadratyx

July 11, 2015

Wake-Up Quiz

Overview of This Class

- Designing algorithms & evaluating their performance
- Designing Parallel & Distributed algorithms, and evaluating their performance
- Loosely Coupled Architectures
- Hadoop Ecosystem

ALGORITHMS & THEIR PERFORMANCE

Introduction to Algorithm design and analysis

Example: sorting problem.

Input: a sequence of n number $<a_1, a_2, \ldots, a_n>$

Output: a permutation (reordering) $<a_1', a_2', \ldots, a_n'>$

such that $a_1' \leq a_2' \leq \ldots \leq a_n'$.

Different sorting algorithms:

Insertion sort and Mergesort.

Insertion Sort Algorithm

```
INSERTION-SORT(A)
for j = 2 to length[A]
    do key ← A[j]
        //insert A[j] to sorted sequence A[1..j-1]
        i ← j-1
        while i >0 and A[i]>key
            do A[i+1] ← A[i]  //move A[i] one position right
                i ← i-1
        A[i+1] ← key
```

40 15 30 5 25 10 20 35

15 40 30 5 25 10 20 35

15 30 40 5 25 10 20 35

5 15 30 40 25 10 20 35

5 15 25 30 40 10 20 35

5 10 15 25 30 40 20 35

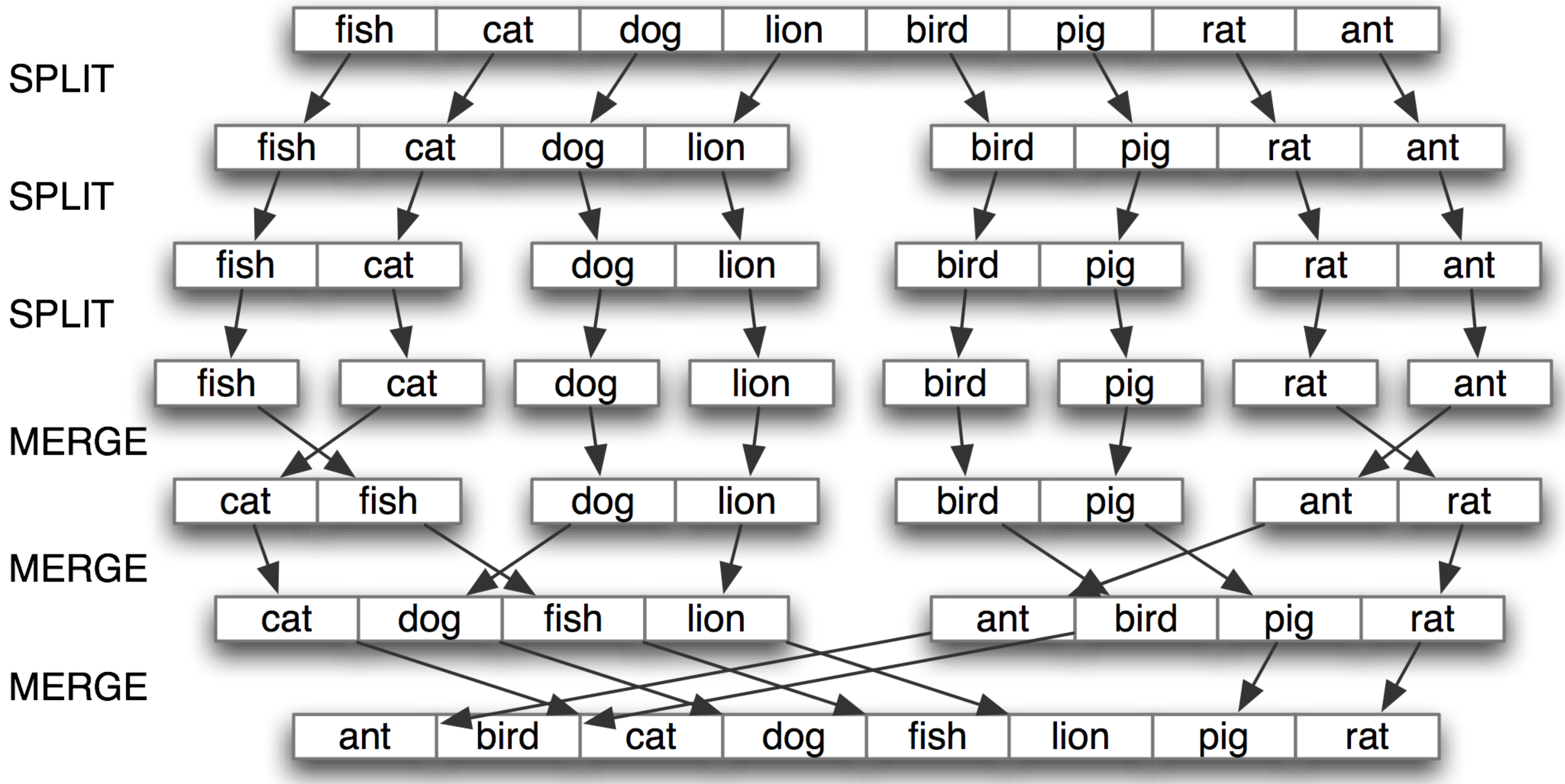
5 10 15 20 25 30 40 35

5 10 15 20 25 30 35 40

MERGE-SORT(A,*p*,*r*)

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor (p+r)/2 \rfloor$
3. MERGE-SORT(A,p,q)
4. MERGE-SORT(A,$q+1$,r)
5. MERGE(A,p,q,r)

Call to MERGE-SORT(A,1,n) (suppose n=length(A))


fish
cat
dog
lion
bird
pig
rat
ant
SPLIT
SPLIT
SPLIT
MERGE
MERGE
MERGE

Time Complexity

- Is the algorithm “fast enough” for my needs?
- How much longer will the algorithm take if I increase the amount of data it must process
- Given a set of algorithms that accomplish the same thing, which is the right one to choose?

Ranking of Algorithmic Behaviors

Function	Common Name
$N!$	factorial
2^N	Exponential
N^d, $d > 3$	Polynomial
N^3	Cubic
N^2	Quadratic
$N\sqrt{N}$	
$N \log N$	
N	Linear
$\sqrt{N}$	Root - n
$\log N$	Logarithmic
1	Constant

slowest

fastest

Efficiency comparison of Insertion Sort & Merge Sort

- Suppose n=10^6 numbers:
  - Insertion sort: $c_1 n^2$
  - Merge sort: $c_2 n$ (lg n)
  - Best programmer (c_1=2), machine language, one billion/second computer A.
  - Bad programmer (c_2=50), high-language, ten million/second computer B.
  - 2 $(10^6)^2$ instructions/10^9 instructions per second = 2000 seconds.
  - 50 (10^6 lg 10^6) instructions/10^7 instructions per second ≈ 100 seconds.
  - Thus, merge sort on B is 20 times faster than insertion sort on A!
  - If sorting ten million numbers, 2.3 days VS. 20 minutes.

Running Times

- Assume N = 100,000 and processor speed is 1,000,000 operations per second

Function	Running Time
2^N	over 100 years
N^3	31.7 years
N^2	2.8 hours
$N\sqrt{N}$	31.6 seconds
N log N	1.2 seconds
N	0.1 seconds
$\sqrt{N}$	3.2×10^{-4} seconds
log N	1.2×10^{-5} seconds

Conclusions – Efficiency Comparison

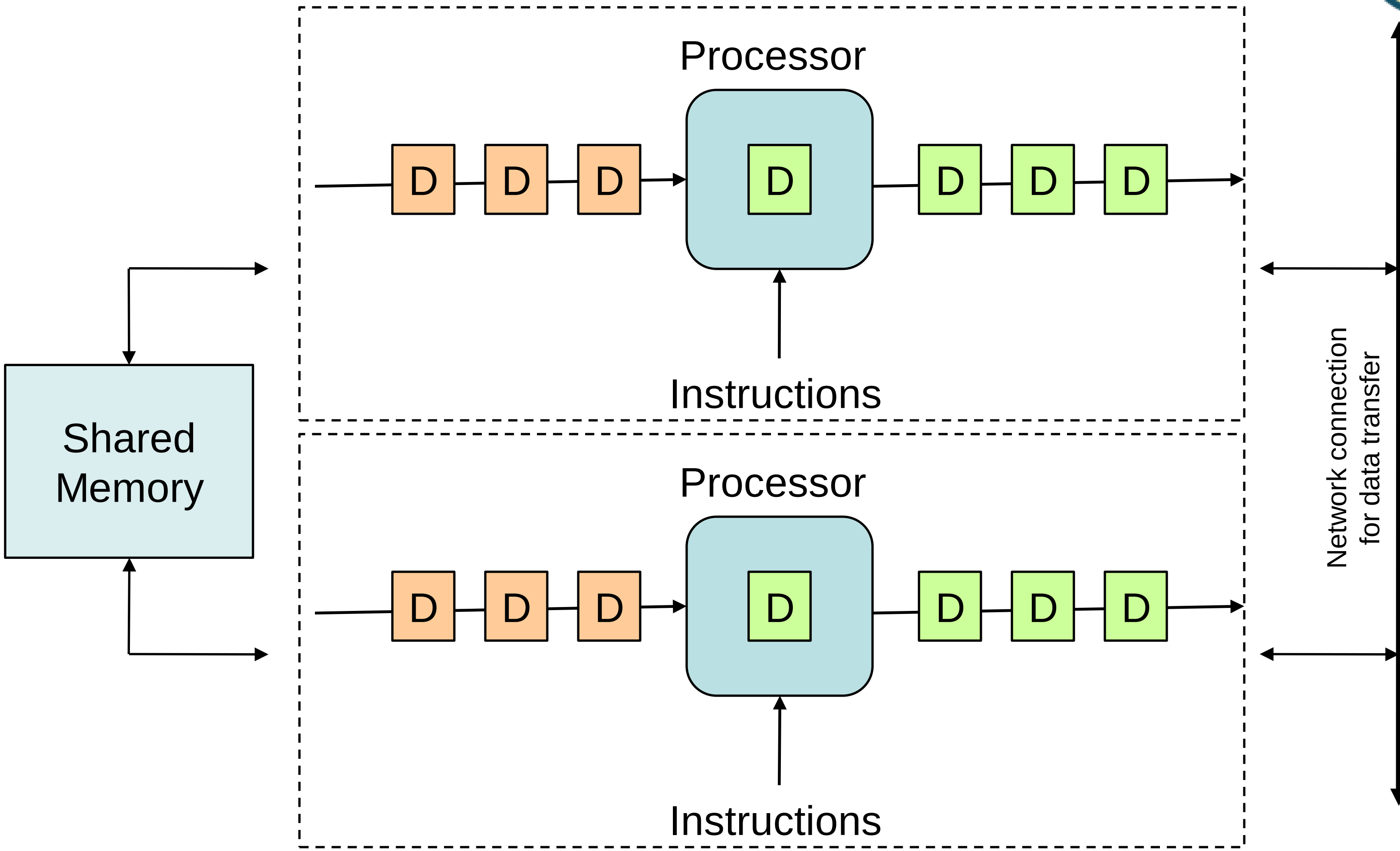
- Algorithms for solving the same problem can differ dramatically in their efficiency.
- These differences are *much more* significant than the differences due to hardware and software.

This is true for concurrent programming also. We must design/choose the right algorithms.

In concurrent programming, we must also look at some more metrics.

CONCURRENT ALGORITHM DESIGN & PERFORMANCE

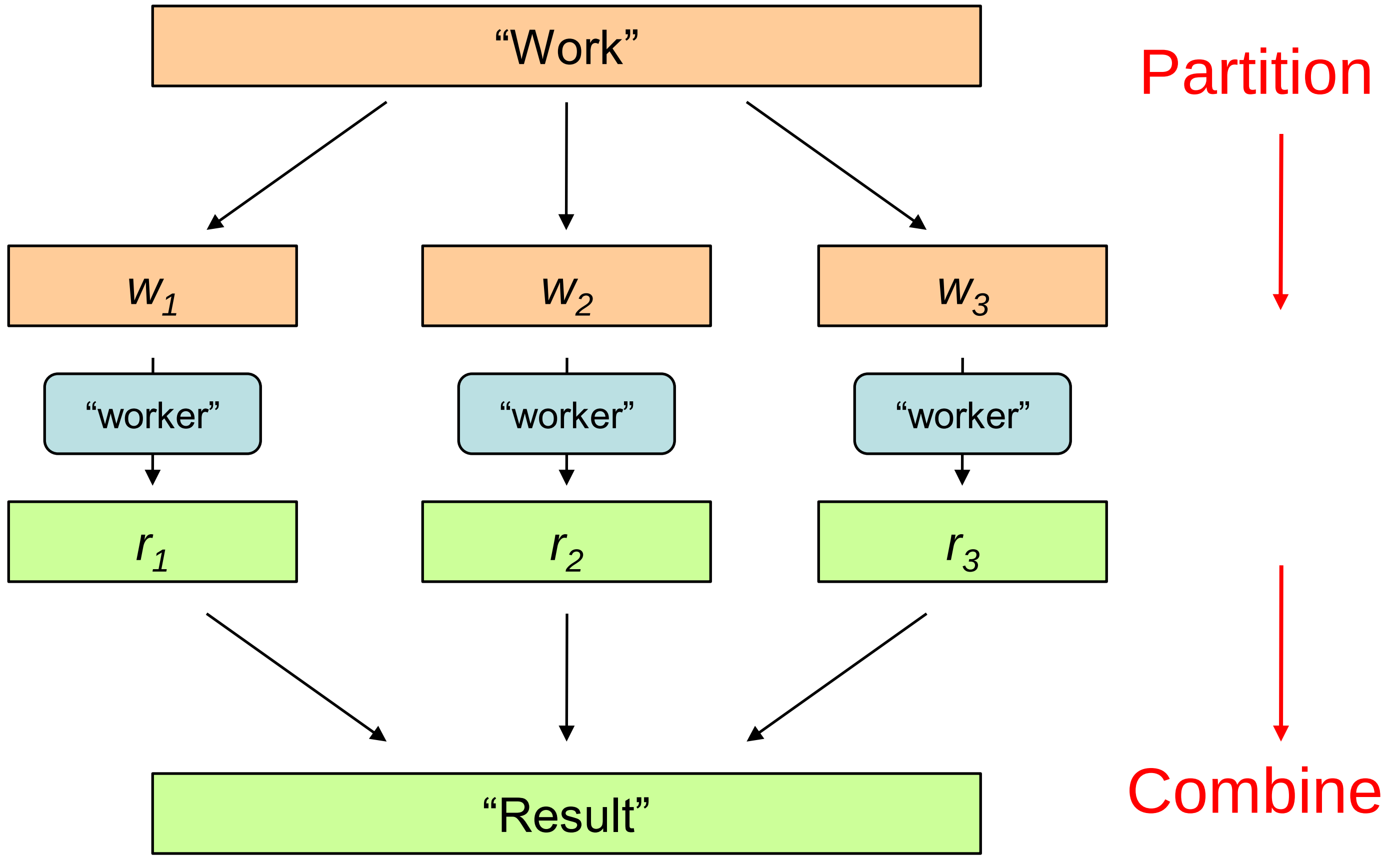
Parallel vs. Distributed



Parallel: Multiple CPUs within a shared memory machine

Distributed: Multiple machines with own memory connected over a network

Divide and Conquer



Different Workers

- Different threads in the same core
- Different cores in the same CPU
- Different CPUs in a multi-processor system
- Different machines in a distributed system

Parallelization Problems

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What is the common theme of all of these problems?

Failure: The Defining Characteristic

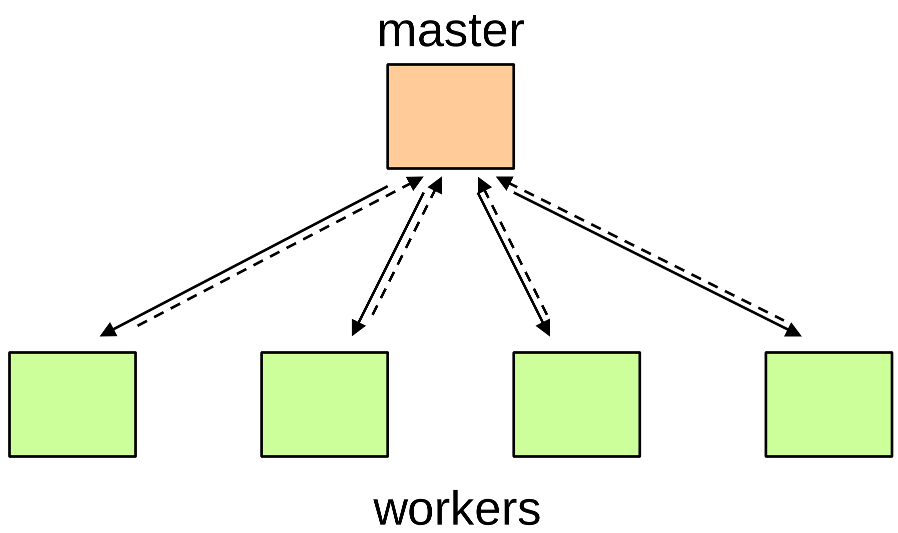
- Defining characteristic of distributed computing is the ability to deal with failure
  - Performance should degrade gracefully with partial failure of the system
  - Failure should not result in loss of any data (Data recovery)
  - Recovered components should be able to rejoin the system without needing a bootup (Component Recovery)
  - Partial failures should not affect outcome (consistency)

Patterns for Parallelism

- Several programming methodologies exist to build parallelism into programs
- Here are some…

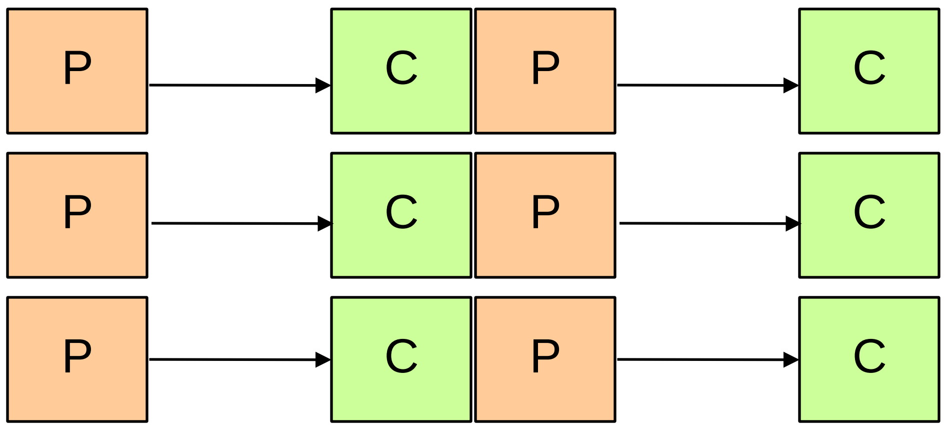
Master/Workers

- The master initially owns all data
- The master creates workers and assigns tasks
- The master waits for workers to report back



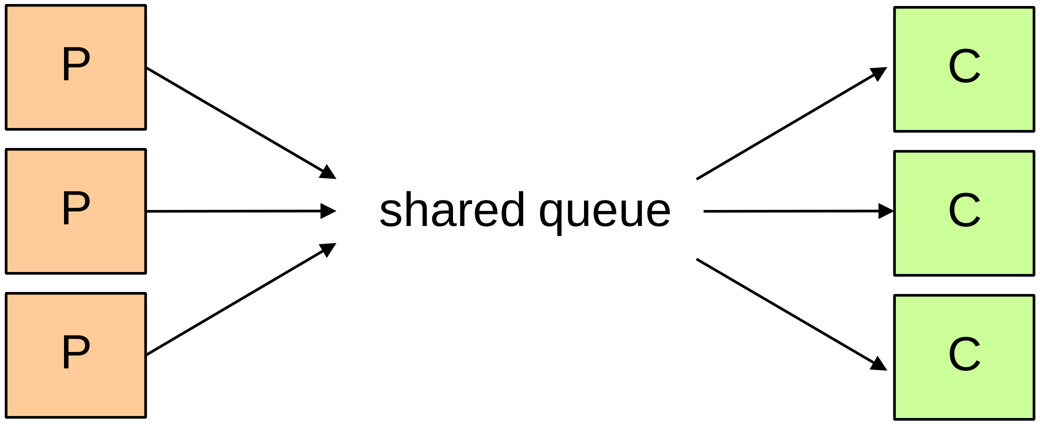
Producer/Consumer Flow

- Producers create work items
- Consumers process them
- Can be daisy-chained



Work Queues

- All available consumers should be available to process data from any producer
- Work queues divorce 1:1 relationship from producers to consumers



And …

- The above solutions represent general patterns
- In reality:
  - Lots of one-off solutions, custom code
  - Burden on the programmer to manage everything
- Can we push the complexity onto the system?
  - MapReduce (later)

Concurrent Algorithms: Additional Metrics

Decomposition, Tasks, Task Dependency Graphs

- The first step in developing a parallel algorithm is to decompose the problem into tasks that can be executed concurrently

- A given problem may be decomposed into tasks in many different ways. Tasks may be of same, different, or even interminate sizes.

- ***Task dependency graph:*** A directed graph with nodes corresponding to tasks and edges indicating that the result of one task is required for processing the next

Example: Database Query Processing

Consider the execution of the query:

MODEL = ``CIVIC" AND YEAR = 2001 AND (COLOR = ``GREEN" OR COLOR = ``WHITE)

on the following database:

ID#	Model	Year	Color	Dealer	Price
4523	Civic	2002	Blue	MN	$18,000
3476	Corolla	1999	White	IL	$15,000
7623	Camry	2001	Green	NY	$21,000
9834	Prius	2001	Green	CA	$18,000
6734	Civic	2001	White	OR	$17,000
5342	Altima	2001	Green	FL	$19,000
3845	Maxima	2001	Blue	NY	$22,000
8354	Accord	2000	Green	VT	$18,000
4395	Civic	2001	Red	CA	$17,000
7352	Civic	2002	Red	WA	$18,000

Example: Database Query Processing

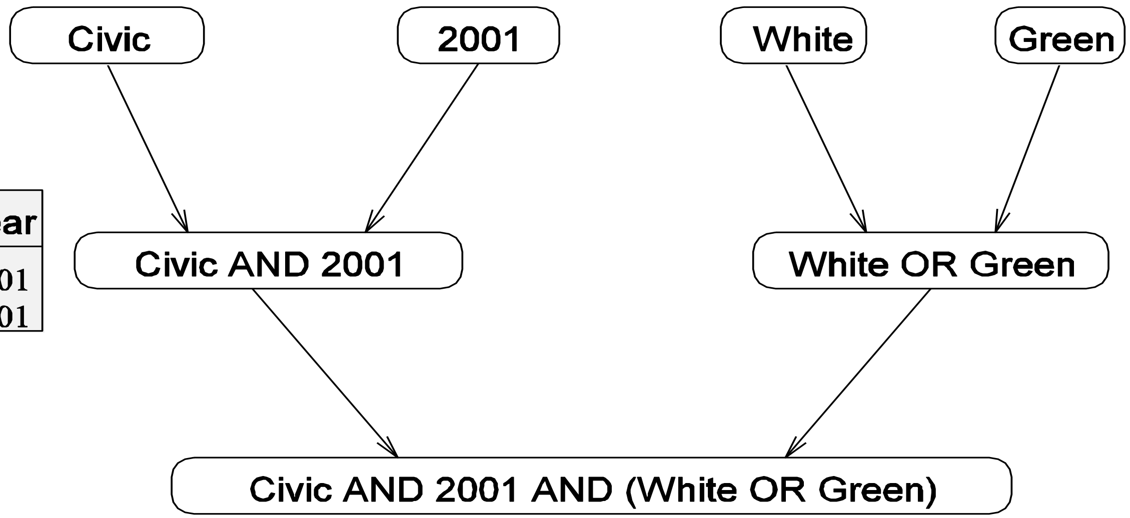
Execution of the query can be divided into subtasks in various ways.

ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2001
6734	2001
5342	2001
3845	2001
4395	2001

ID#	Color
3476	White
6734	White

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green



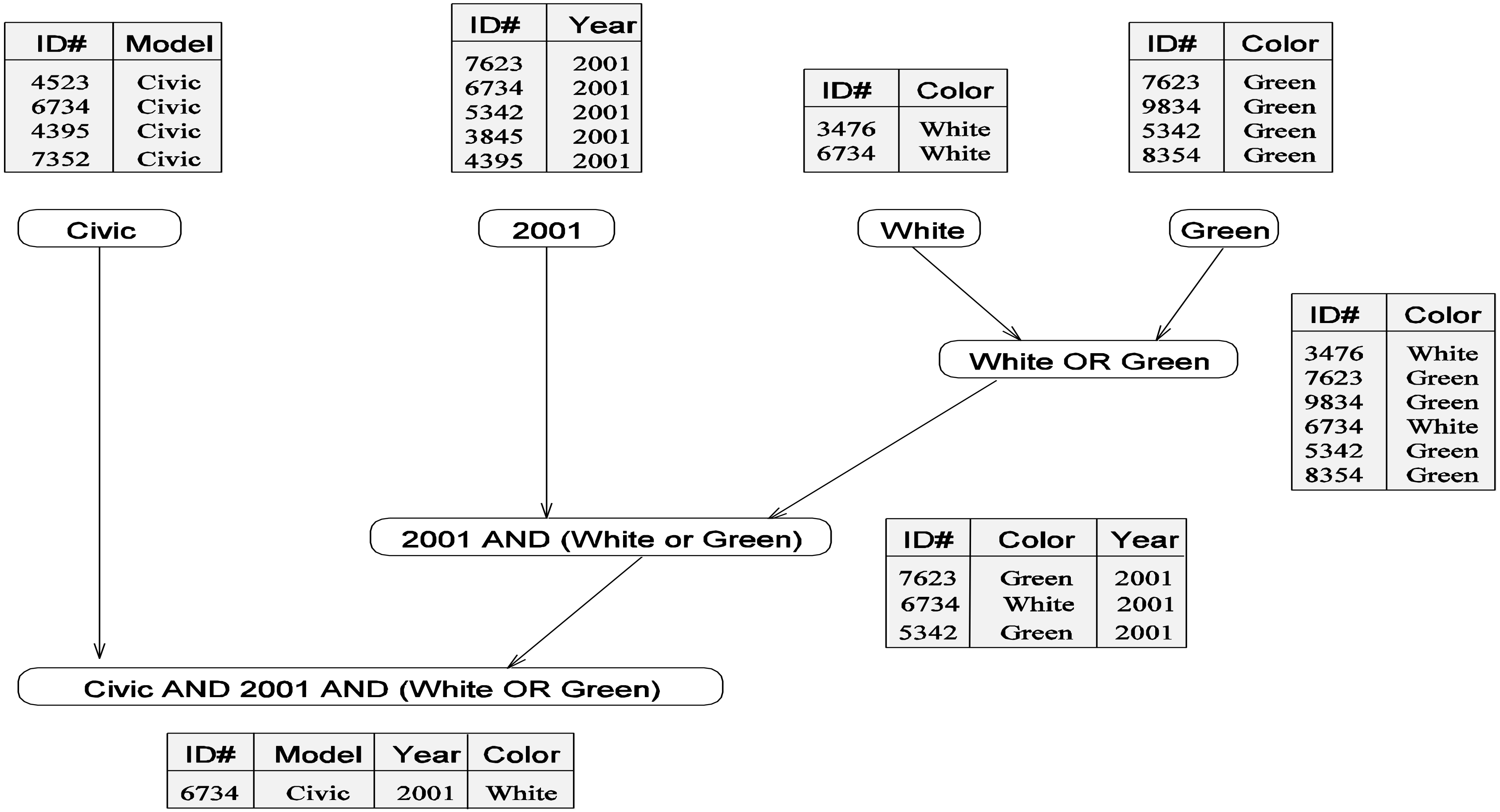
ID#	Model	Year
6734	Civic	2001
4395	Civic	2001

ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Model	Year	Color
6734	Civic	2001	White

Example: Database Query Processing

Another Way:



ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2001
6734	2001
5342	2001
3845	2001
4395	2001

ID#	Color
3476	White
6734	White

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green

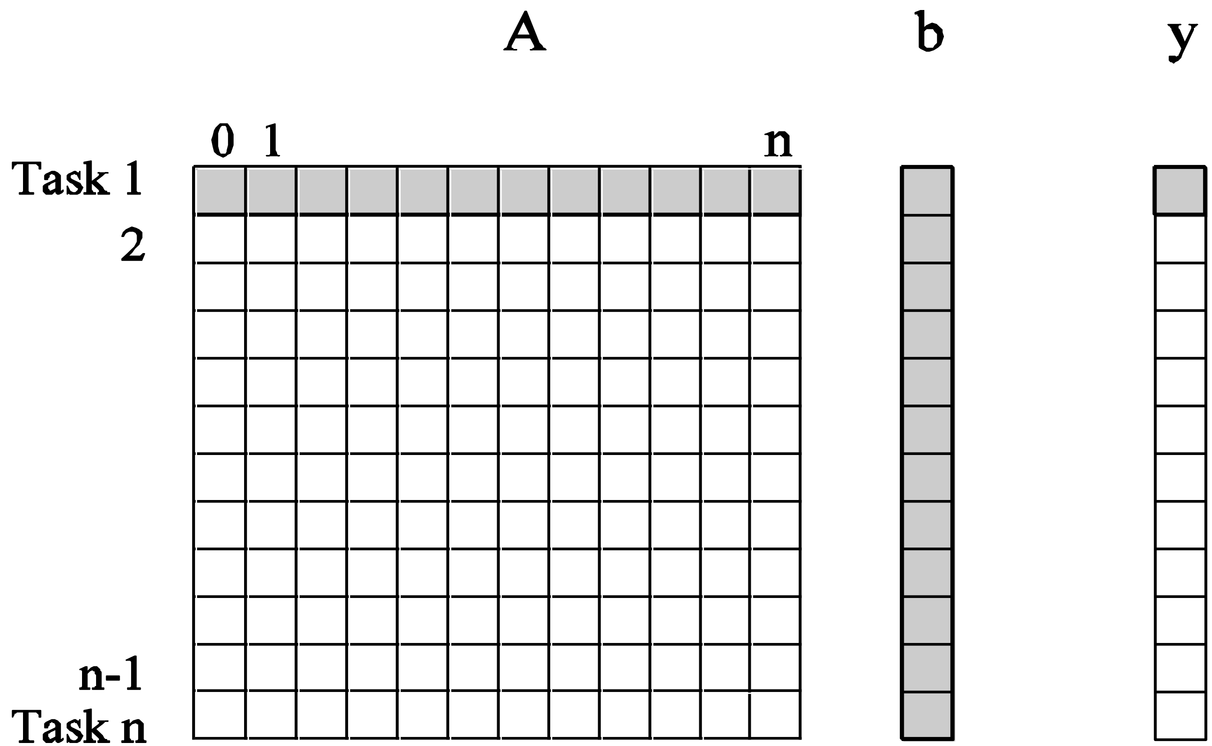
ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Color	Year
7623	Green	2001
6734	White	2001
5342	Green	2001

ID#	Model	Year	Color
6734	Civic	2001	White

Different task decompositions may have significantly different parallel performance.

What's the TDG? Multiplying a Dense Matrix with a Vector



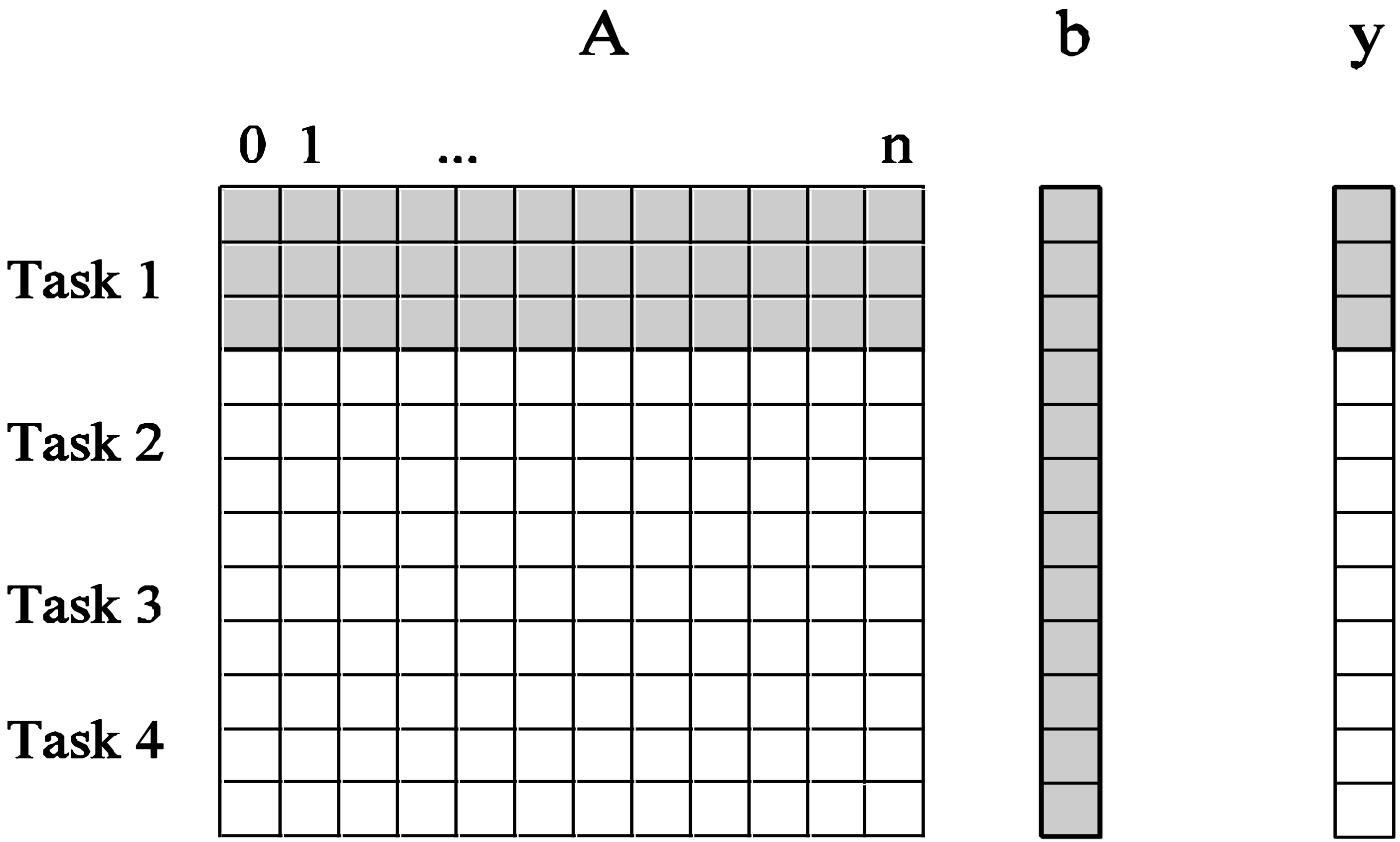
Computation of each element of ***y*** is independent of other elements.
A dense matrix-vector product can be decomposed into ***n*** tasks.

Tasks share data (namely, the vector ***b***), but do not have any control dependencies . No task needs to wait for the (partial) completion of any other.

Is this the maximum number of tasks we could decompose this problem into?

Granularity of Task Decompositions

- Number of tasks into which a problem is decomposed
- Large number of tasks: ***fine-grained decomposition***;
- Small number of tasks: ***coarse grained decomposition***



A coarse grained counterpart to the dense matrix-vector product example.

Degree of Concurrency

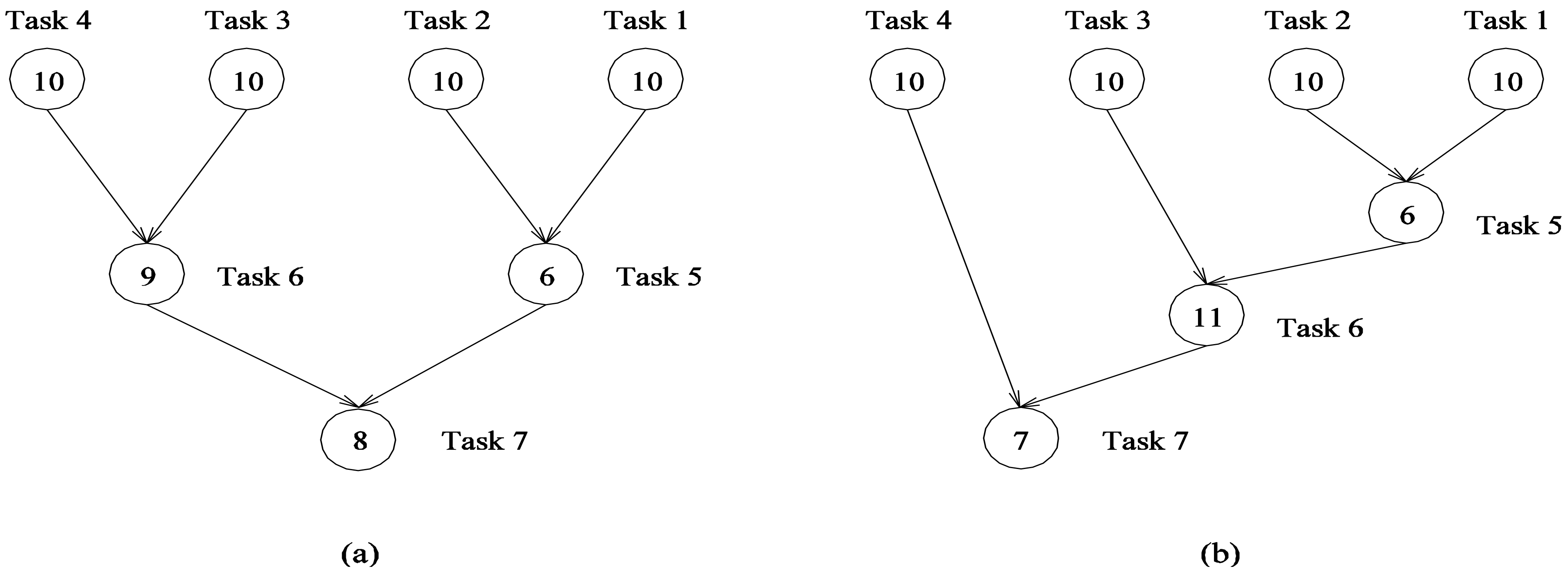
- The number of tasks that can be executed in parallel is the ***degree of concurrency*** of a decomposition.
- Number of tasks that can be executed in parallel may change over program execution
  - ***Maximum*** *degree of concurrency*
  - ***Average*** *degree of concurrency*
- The degree of concurrency increases as the decomposition becomes finer in granularity and vice versa.

Critical Path Length

- A directed path in the task dependency graph represents a sequence of tasks that must be processed one after the other.
- The longest such path determines the shortest time in which the program can be executed in parallel.
- The length of the longest path in a task dependency graph is called the ***critical path length***.

Critical Path Length: An Example

Task dependency graphs of the two database query decompositions:



- What is the shortest parallel execution time for each decomposition?
- How many processors are needed in each case to achieve this minimum parallel execution time?
- What is the maximum degree of concurrency?

Decomposition Techniques

- How does one decompose a task into various subtasks?

- There is no single recipe that works for all problems. Set of commonly used techniques:
  - recursive decomposition
  - data decomposition
  - exploratory decomposition
  - speculative decomposition

There are also hybrid decompositions.

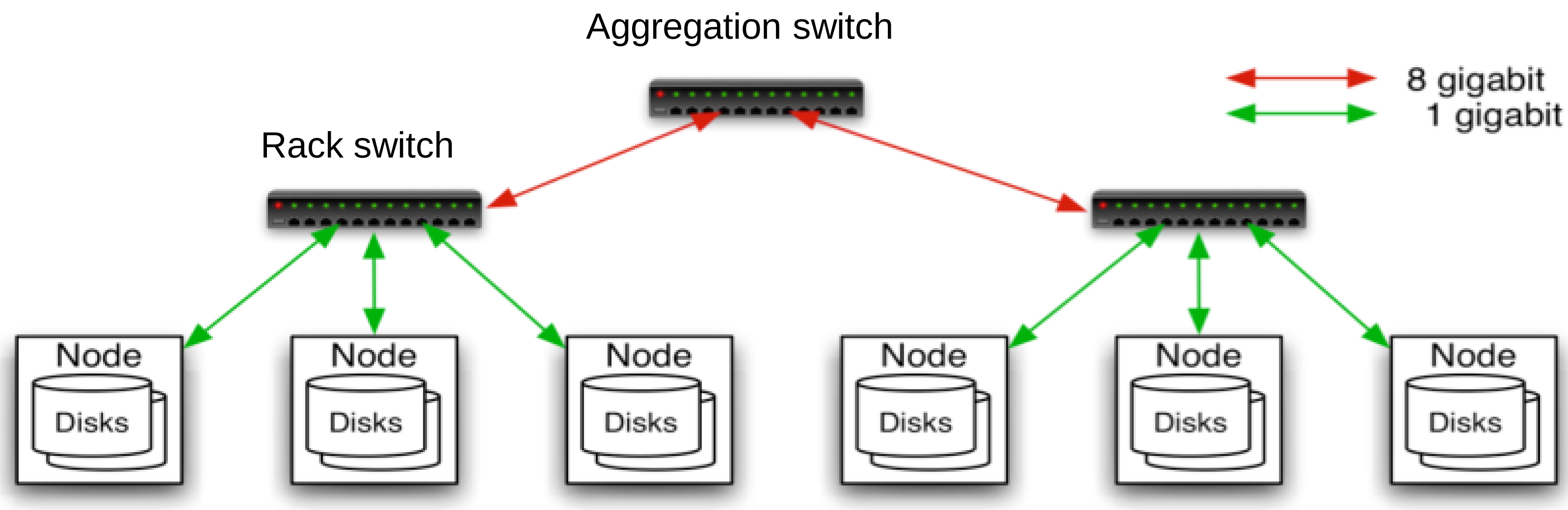
Additional References

(Not NECESSARY. Only for optional, extra reading)

- http://en.wikipedia.org/wiki/Sorting_algorithm
- http://en.wikipedia.org/wiki/Big_O_notation
- http://www.amazon.com/Art-Computer-Programming-Sorting-Searching/dp/0201896850
- http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/026203384/
- http://people.cs.aau.dk/~adavid/teaching/MVP-08/03-Task%20decomposition%20and%20mapping.pdf

THE HADOOP ECOSYSTEM

Commodity Hardware



- Typically in 2 level architecture
  - Nodes are commodity PCs
  - 30-40 nodes/rack
  - Uplink from rack is 3-4 gigabit
  - Rack-internal is 1 gigabit

Hadoop - Why ?

- Need to process huge datasets on large clusters of computers
- Very expensive to build reliability into each application
- Nodes fail every day
  - Failure is expected, rather than exceptional
  - The number of nodes in a cluster is not constant
- Need a common infrastructure
  - Efficient, reliable, easy to use
  - Open Source, Apache Licence

Who is using Hadoop?

SNo.	Company	Nodes
1	Yahoo!	42,000
2	LinkedIn	4100
3	Facebook	1400
4	NetSeer	1050
5	Quantcast	750
6	EBay	532
7	CRS4	400
8	Powerset / Microsoft	400
9	Adknowledge	200
10	Neptune	200
11	AOL	150
12	Inmobi	150
13	FOX Audience Network	140
14	Specific Media	138
15	Search Wikia	125
16	eCircle	120
17	Spotify	120
18	The Lydia News Analysis Project	120
19	A9.com	100
20	ARA.COM.TR	100
21	Cornell	100
22	Last.fm	100

Source: Apache Hadoop Wiki

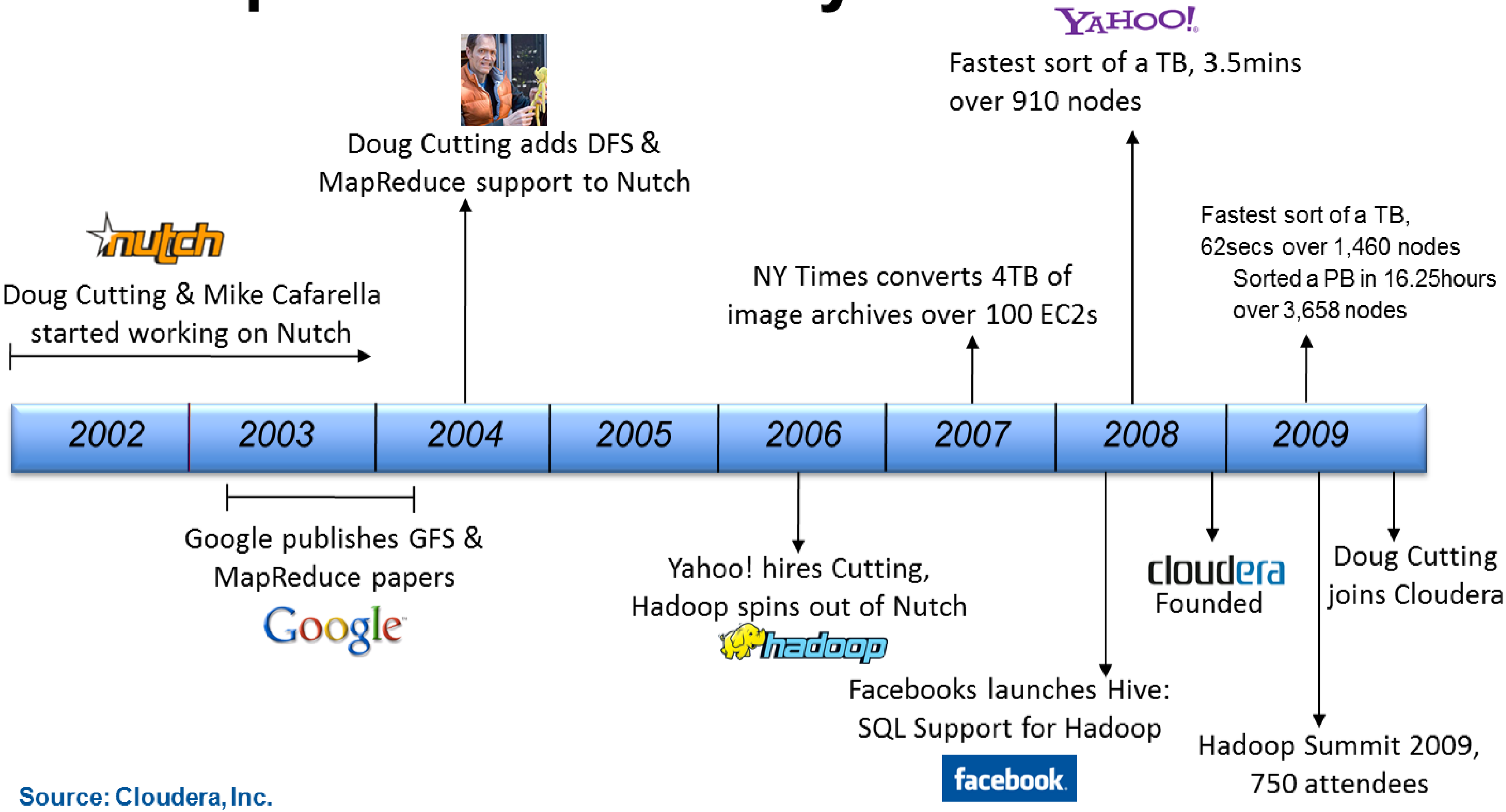
"Powered By Hadoop" page last

updated December 20, 2012

Notably missing: Google, Walmart, NSA, …

http://www.hadoopwizard.com/which-big-data-company-has-the-worlds-biggest-hadoop-cluster/

Hadoop Creation History



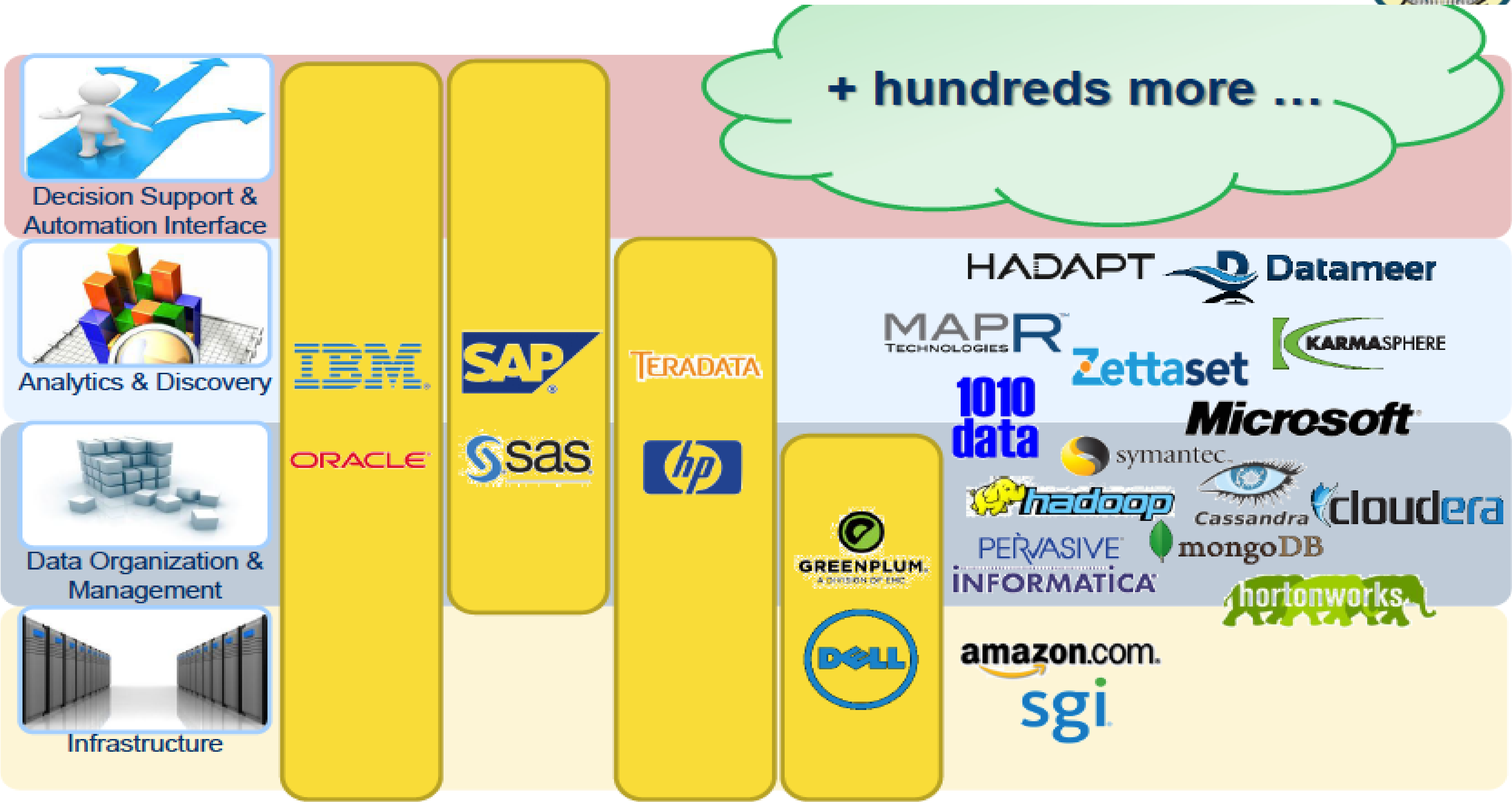
Source: Cloudera, Inc.

Is Big Data the same as Hadoop?

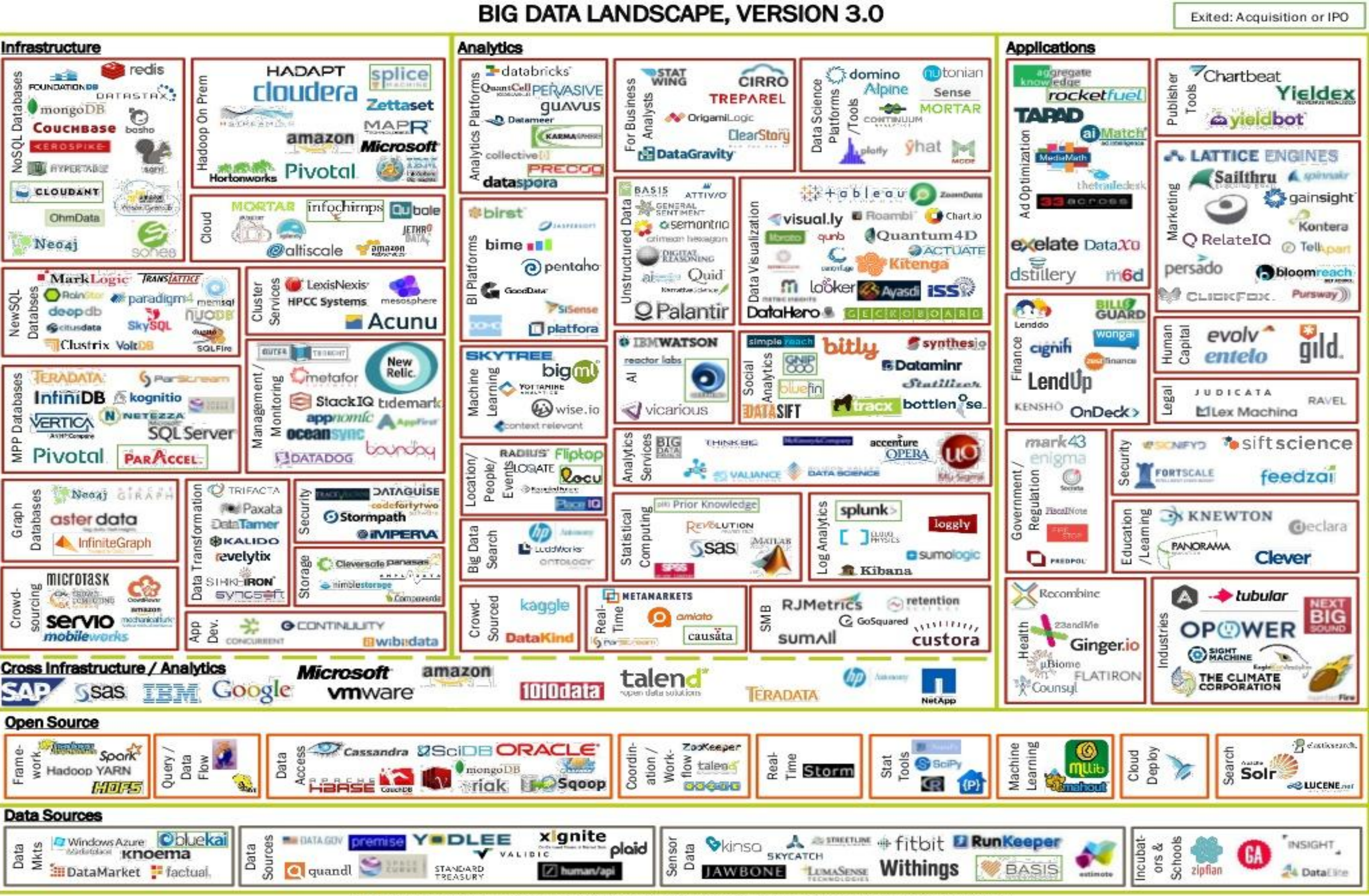
What are there alternatives to Hadoop?

What is so special about Hadoop?

Layers of Players

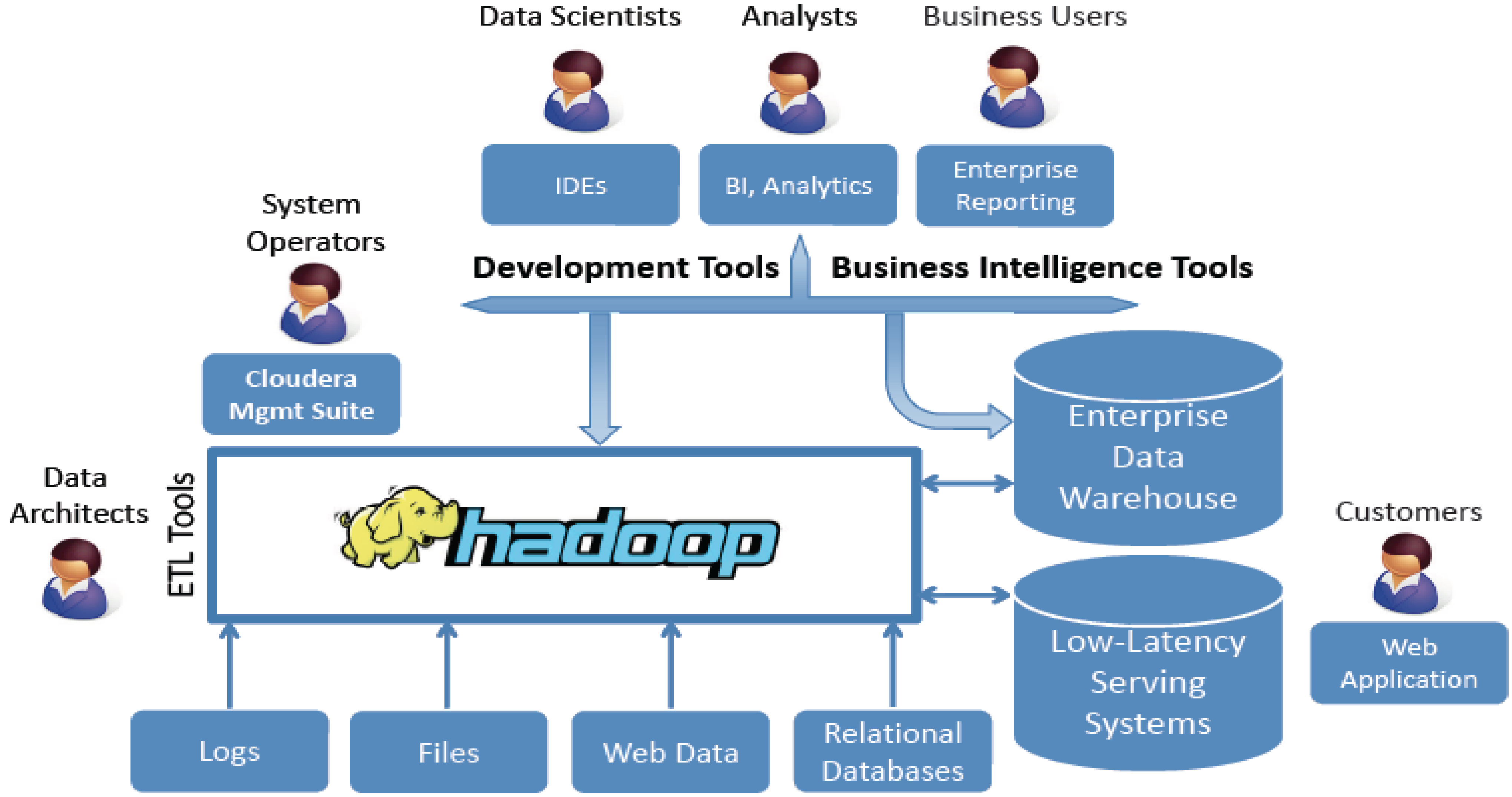


BIG DATA LANDSCAPE, VERSION 3.0



© Matt Turck (@mattturck), Sutian Dong (@sutiandong) & FirstMark Capital (@firstmarkcap)

Big Data – Enterprise Roles



International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

Web: http://www.insofe.edu.in

Facebook: https://www.facebook.com/insofe

Twitter: https://twitter.com/Insofeedu

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE

LinkedIn: http://www.linkedin.com/company/international-school-of-engineering