How to survive the Data Deluge: Petabyte scale Cloud Computing

Gianmarco De Francisci Morales
IMT Institute for Advanced Studies Lucca
CSE PhD XXIV Cycle

18 Jan 2010

Outline

- Part 1: Introduction
  - What, Why and History
- Part 2: Technology overview
  - Current systems and comparison
- Part 3: Research directions
  - Ideas for future improvements

Part I
Introduction

How would you sort...

- ... 1GB of data?
- ... 100GB of data?
- ... 10TB of data?
- Scale matters!
  - Because More Isn't Just More, More Is Different

The Petabyte Age

What is scalability?

- The ability for a system to accept increased volume without impacting the profits
- Scale-free systems
- Scale-up vs *Scale-out*
- Types of parallel architectures:
  - Shared memory, Shared disk, *Shared nothing*

What if you need...

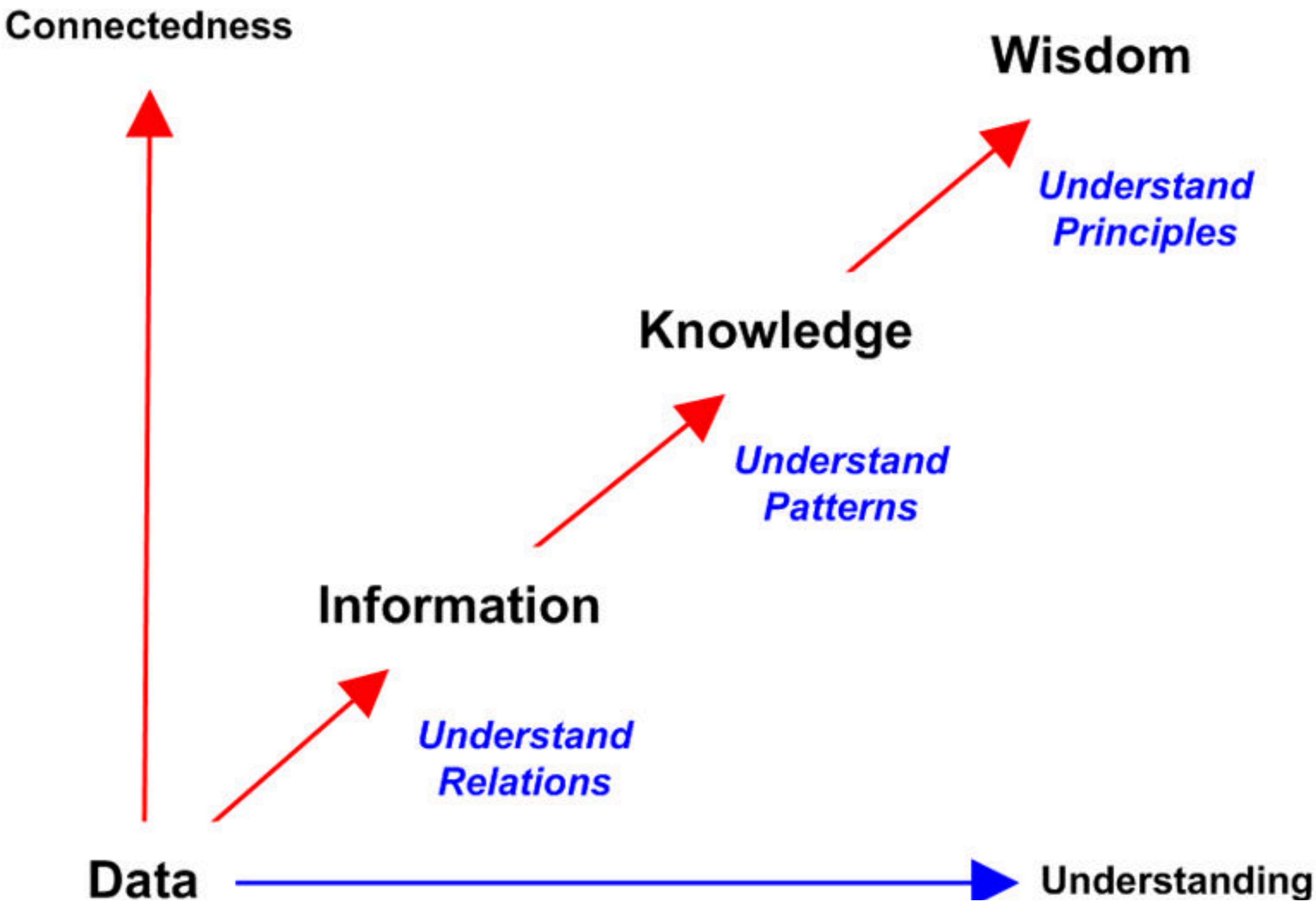
- ... to store and analyze 10TB of data per day?
  - Parallel is a must, but not enough
- Usual approaches fail at this scale because of secondary effects
  - Operational costs
  - Faults

What is fault tolerance?

- System operates properly in spite of the failure of some of its components
- High Availability
- Real world need
  - Software has bugs
  - Hardware fails

Why data?

- The world is drowning in data: Data Deluge
- Data sources:
  - Web 2.0 (user generated content)
  - Scientific experiments
    - Physics (particle accelerators)
      Astronomy (satellite images)
      Biology (genomic maps)
  - Can you think of others?



"Data is not information,
information is not knowledge,
knowledge is not wisdom."
Clifford Stoll

DBMS evolution

- ‘60s CODASYL
- ‘70s Relational DBMS
- ‘80s Object-Oriented DBMS (Back to navigation)
- ‘80s & ‘90s Parallel DBMS
- Not much has happened since the ‘70s
    - The fundamental model and the code lines are still the same

DBMS yesterday

- Business transaction processing (OLTP)
- Relational model
- SQL

DBMS today

- Different markets (OLTP, OLAP, Stream, etc..)
- Stored Procedures & User Defined Functions
- Parallel DBMS (Teradata, Vertica, etc..)
  - Not enough flexibility
  - Limited fault-tolerance and scalability

Why cloud?

- Parallel computing is dead
  - Amdahl's law: $SpUp(N) = 1 / ((1-P_a)+P_a/N)$
- Long live parallel computing
  - Gustafson's law: $SpUp(N) = P_G*N + (1-P_G)$
  - Physical limits
  - Manycore
  - Money

Parallel computing evolution

- Parallel (single)
- Cluster (intra-site)
- Grid (inter-site)
- Cloud (scale-free)
- What's next?

Parallel computing yesterday

- CPU bound problems
  - Tightly coupled
- Use of MPI or PVM
  - Move data among computing nodes
- Use of NAS/SAN
  - Expensive and does not scale (shared disk)

Parallel computing today

- I/O bound problems (often)
- Move computing near data
- Focus on scalability and fault tolerance
  - Simple!
  - Shared nothing architecture
    on commodity hardware
  - Data streaming

Wrap-up

- Main motivations
  - Scalability
  - Money
- Focus on BIG data
  - BIG = need to stop & think because of its size
  - Common issues with PDBMS
    (load balancing, data skew)

Part 2
Technology overview

What is Cloud Computing?

- Did anyone notice I skipped the definition?
  - Buzzword!
- IaaS (EC2, S3)
- PaaS (App Engine, Azure Services Platform)
- SaaS (Salesforce, OnLive, virtually any Web App)
- Scale free computing architecture

Who is involved?

Software stacks

	Google	Yahoo	Microsoft	Others
High Level Languages	Sawzall	Pig/Latin	DryadLINQ, Scope	Hive, Cascading
Computation	MapReduce	Hadoop	Dryad	
Data Abstraction	BigTable	HBase, PNUTS		Cassandra, Voldemort
Distributed Data	GFS	HDFS	Cosmos	CloudStore, Dynamo
Coordination	Chubby	Zookeeper		

Comparison with PDBMS

- CAP Theorem
- BASE vs ACID
- Computing on large data vs Handling large data
- OLAP vs OLTP
- User Defined Functions vs Select-Project-Join
- Nested vs Flat data model

Comparison with PDBMS

- Start small (no upfront schema, flexible, agile)
  Grow big (optimize common patterns)
- MapReduce, a major step backwards
  DeWitt, Stonebraker
- "If the only tool you have is a hammer,
  you tend to see every problem as a nail"
  Abraham Maslow
- SQL and Relational Model are not the answer

Wrap-up

- A lot of hype
  - But also activity
  - Industry is leading the trend,
    has cutting edge software
- Different approaches
  - Most focus on MapReduce
  - Shift toward higher level abstractions

Wrap-up

- NoSQL movement
  - No Relational Model
  - No ACID
  - No Join

Part 3
Research Directions

- Extensions
- Models
- High velocity analytics
- Hybrid systems
- Optimizations

Extensions

- Map-Reduce-Merge: simplified relational data processing on large clusters.
  H. Yang, A. Dasdan, R. Hsiao, and D. Parker. In SIGMOD 2007.
- Goal: implement relational operators efficiently
- How: new final phase that merges 2 key-value lists
- Issues: very low level and hard to use
  needs integration into a high level language

Models

- A new computation model for rack-based computing. F. Afrati and J. Ullman. Unpublished.
- Goal: I/O cost characterization
- Issues: only theoretical analysis
  no existing reference system
- Future: best algorithms for the model
  model adaptation to real systems

Models

- A model of computation for MapReduce.
  H. Karloff, S. Suri, and S. Vassilvitskii. In SODA, 2010.
- Goal: theoretical computability characterization of MapReduce algorithms
- Result: algorithmic design technique for MapReduce
- Future: develop algorithms in this class
  find relationships with other classes

High velocity analytics

- Interactive analysis of web-scale data.
  C. Olston, E. Bortnikov, K. Elmeleegy, F. Junqueira, B. Reed. In CIDR, 2009.
- Goal: speed up general queries for big data
- How: pre-computed templates to fill at run-time
- Future: which templates are useful for interactive?
  help the user to formulate templates (sampling?)

High velocity analytics

- MapReduce online.
  T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears. Technical report, University of California, Berkeley, 2009.
- Goal: speed up turnaround of MapReduce jobs
- How: operator pipelining, online aggregation
- Issues: limited inter-job pipelining (data only)
  inter-job aggregation problematic (scratch data)

Hybrid systems

- HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads.
  A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, A. Rasin.
  In VLDB, 2009.
- Goal: advantages of both DB and MapReduce
- How: integrate a DBMS (PostgreSQL) in Hadoop, Hive as interface
- Issues: better reuse principles than technology

Optimizations

- The Curse of Zipf and Limits to Parallelization: A Look at the Stragglers Problem in MapReduce.
  J. Lin. In LSDS-IR, 2009.
- Goal: data distribution effects on MapReduce parallel query/pairwise similarity as case study
- How: balance input data (split long posting lists)
- Issues: very specific for the problem/algorithm

Other ideas

- Sampling and result estimation
  - A good enough result is often acceptable
- Semantic clues
  - Leverage properties of M/R functions (associativity, commutativity)
  - Properties of the input may speed up the computation

Wrap-up

- New and active field
  - Many opportunities for research
- Crossroad of Distributed Systems and Databases
  - Answer the plea not to "reinvent the wheel"

How to survive the Data Deluge: Petabyte scale Cloud Computing

- Integrate DB principles into Cloud systems
- Enable interactive and approximate analytics
- Evolve beyond the MapReduce paradigm

Questions?