Principles of Distributed Database Systems: spotlight on NewSQL

Patrick Valduriez

Outline

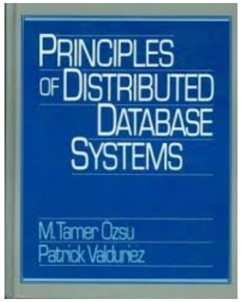
- Distributed database systems
- NoSQL
- Polystores
- Spotlight on NewSQL
- Taxonomy of NewSQL systems
- Current trends

Principles of Distributed Database Systems

Tamer Özsu & Patrick Valduriez

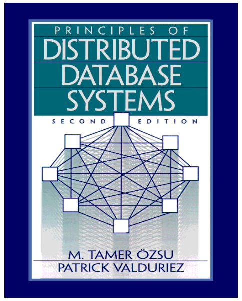
The Story of the Book

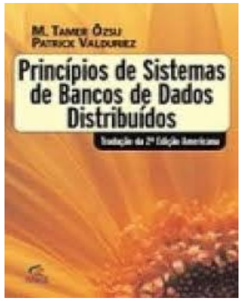
1991

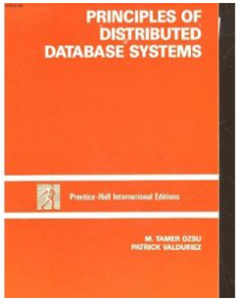


Relational databases.
"In the following 10 years, centralized DBMSs would be an antique curiosity and most organizations would move towards distributed DBMSs." M. Stonebraker (1988)

1999

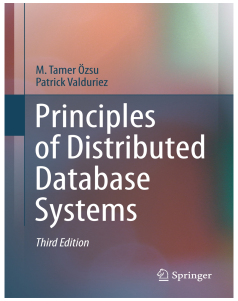


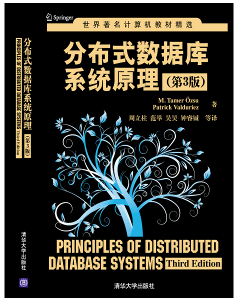




Advanced transaction models, query optimization, object data management, parallel DBMSs.

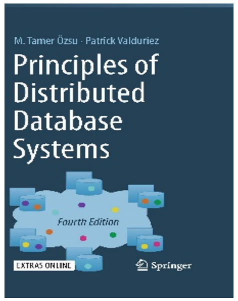
2011





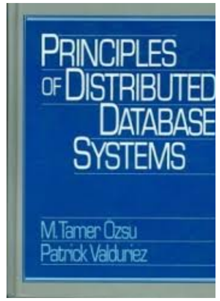
Data replication, database clusters, web data integration, P2P, cloud.

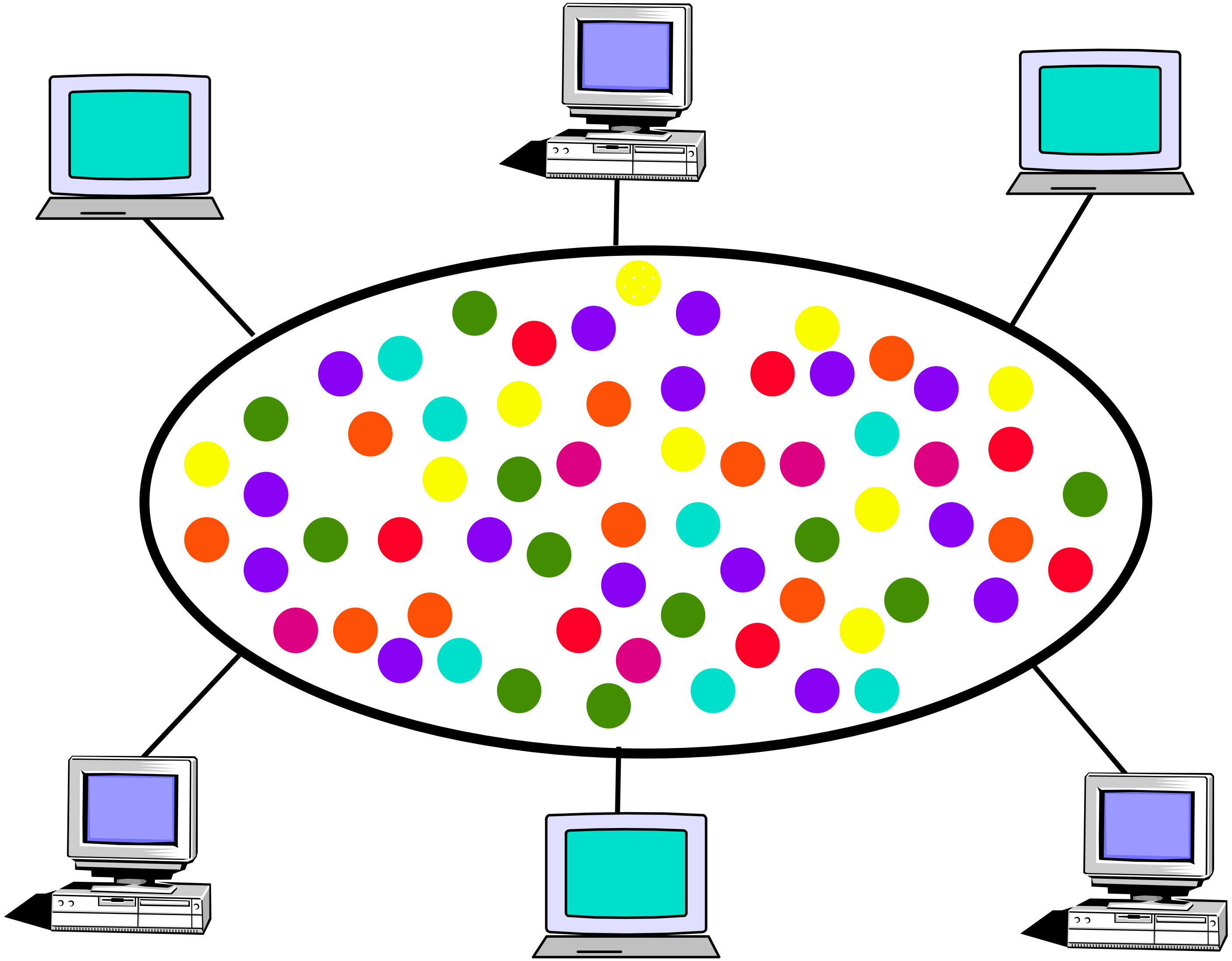
2020



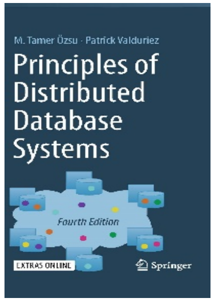
Blockchain, big data, data streaming, graph data analytics, NoSQL, NewSQL, polystores.

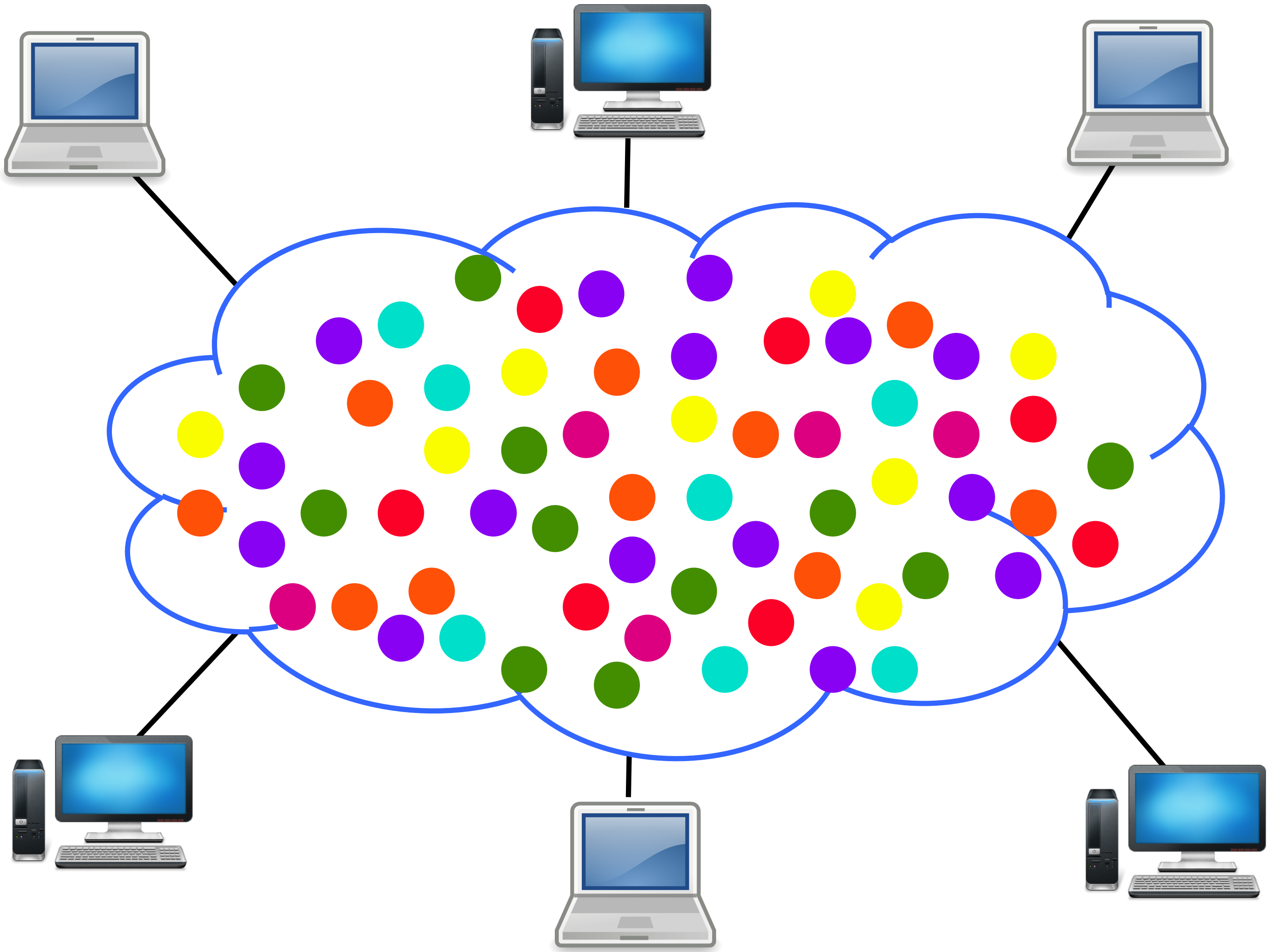
Distributed Database - User View 1991



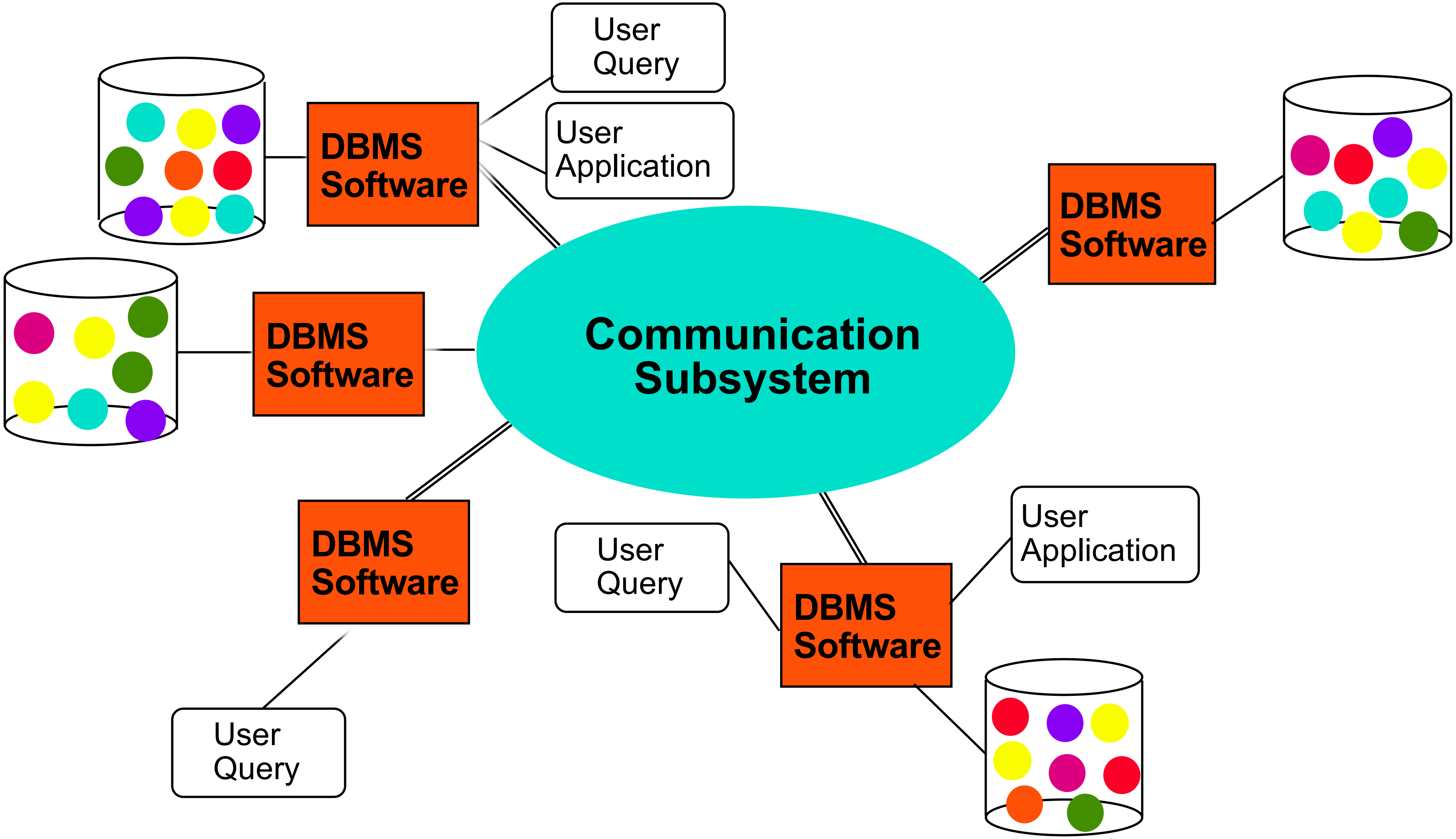


Distributed Database - User View 2020





Distributed DBMS – Reality



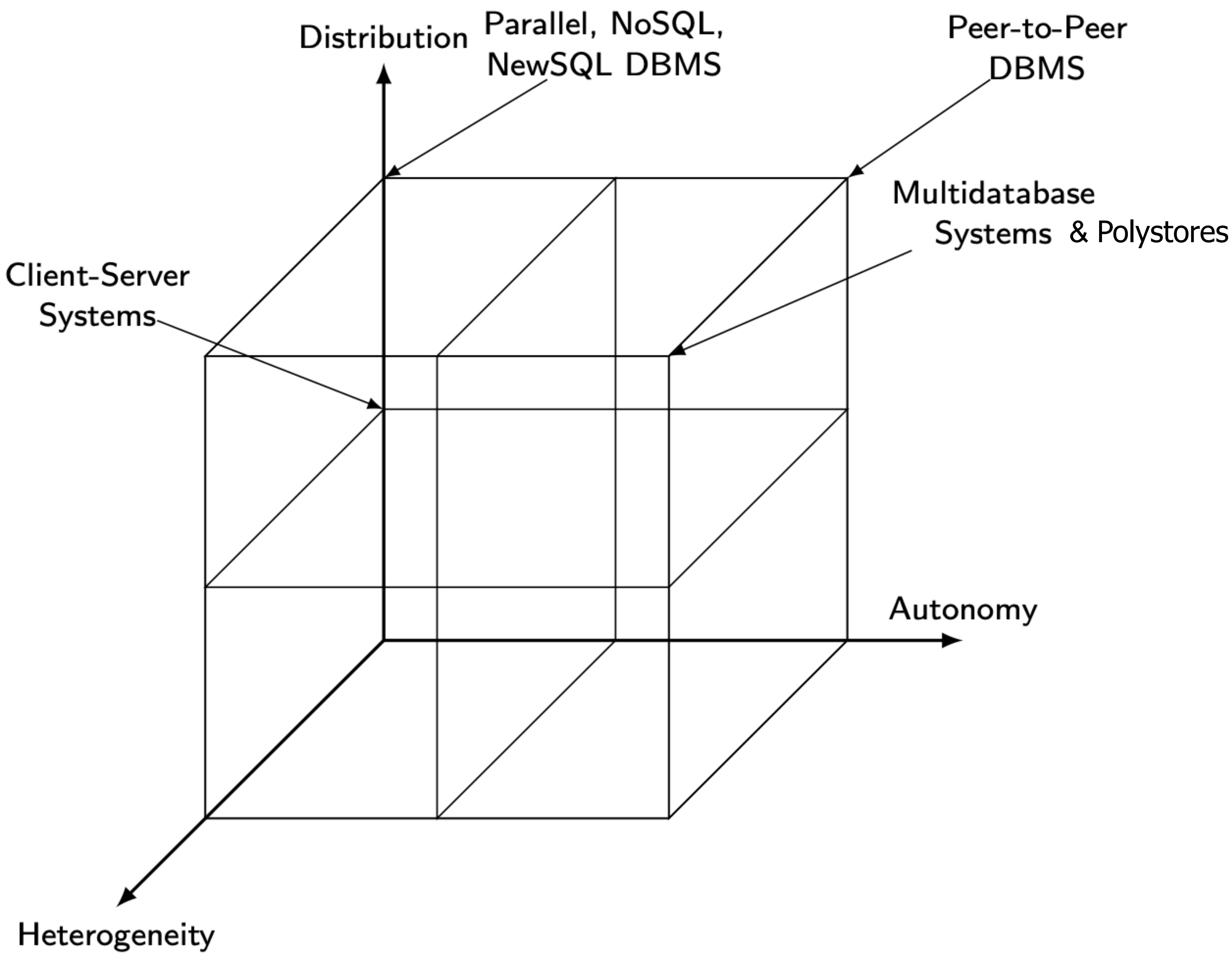
Definitions

- A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network
  - WAN, LAN, cluster interconnect
- A distributed database system (distributed DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users

Promises of Distributed DBMSs

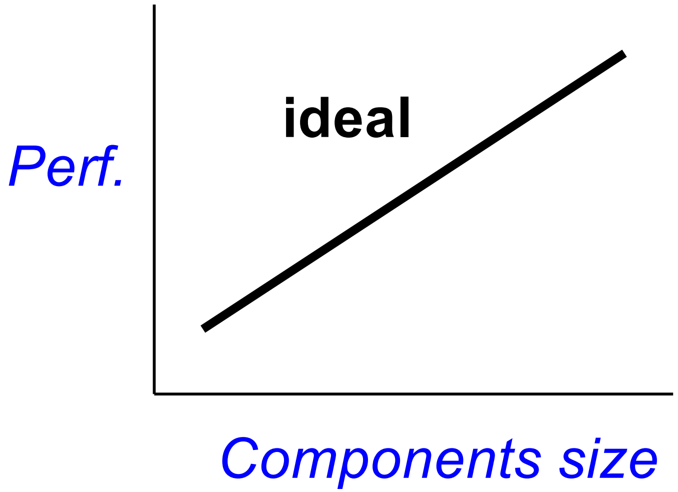
- Transparent management of distributed, fragmented, and replicated data
  - Declarative query language (SQL)
- Improved reliability/availability through replication and distributed transactions
  - Strong ACID consistency
  - Failover and online recovery
- Improved performance
  - Proximity of data to its points of use
  - Data-based parallelism
  - Query optimization
- Easier and more economical system expansion
  - Elasticity in the cloud

Implementation Alternatives



Speed Up

- Ideal: linear speed-up
  - Linear increase of performance by growing the components
    - For a fixed database size and load



Speed-up Limits

- Hardware/software
  - As we add more resources, arbitration conflicts increase
    - E.g. Access to the bus by processors, latching, …
- Application
  - Only part of a program can be parallelized
  - Recall: Amdahl's law that gives the maximum speed-up
    - *Seq* = fraction of code that cannot be parallelized

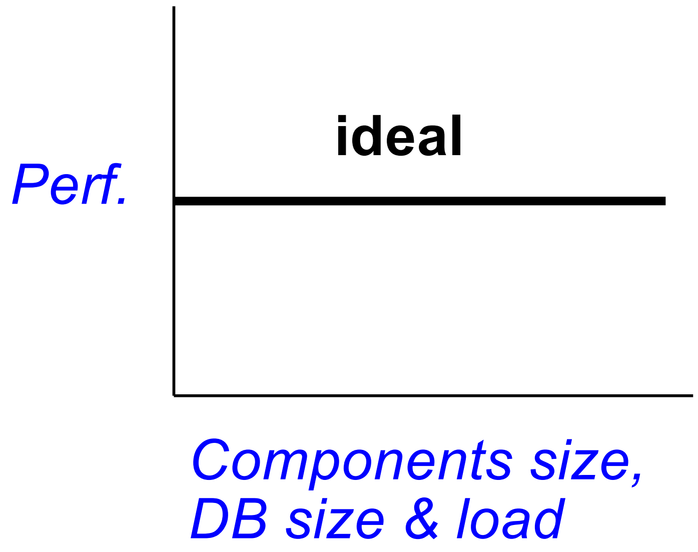
$$\frac{1}{Seq + \dfrac{1 - Seq}{NbProc}}$$

Examples

- Seq=0, NbProc=4 => speed-up= 4
- Seq=30%, NbProc=4 => speed-up= 2,1
- Seq=30%, NbProc=8 => speed-up= 2,5

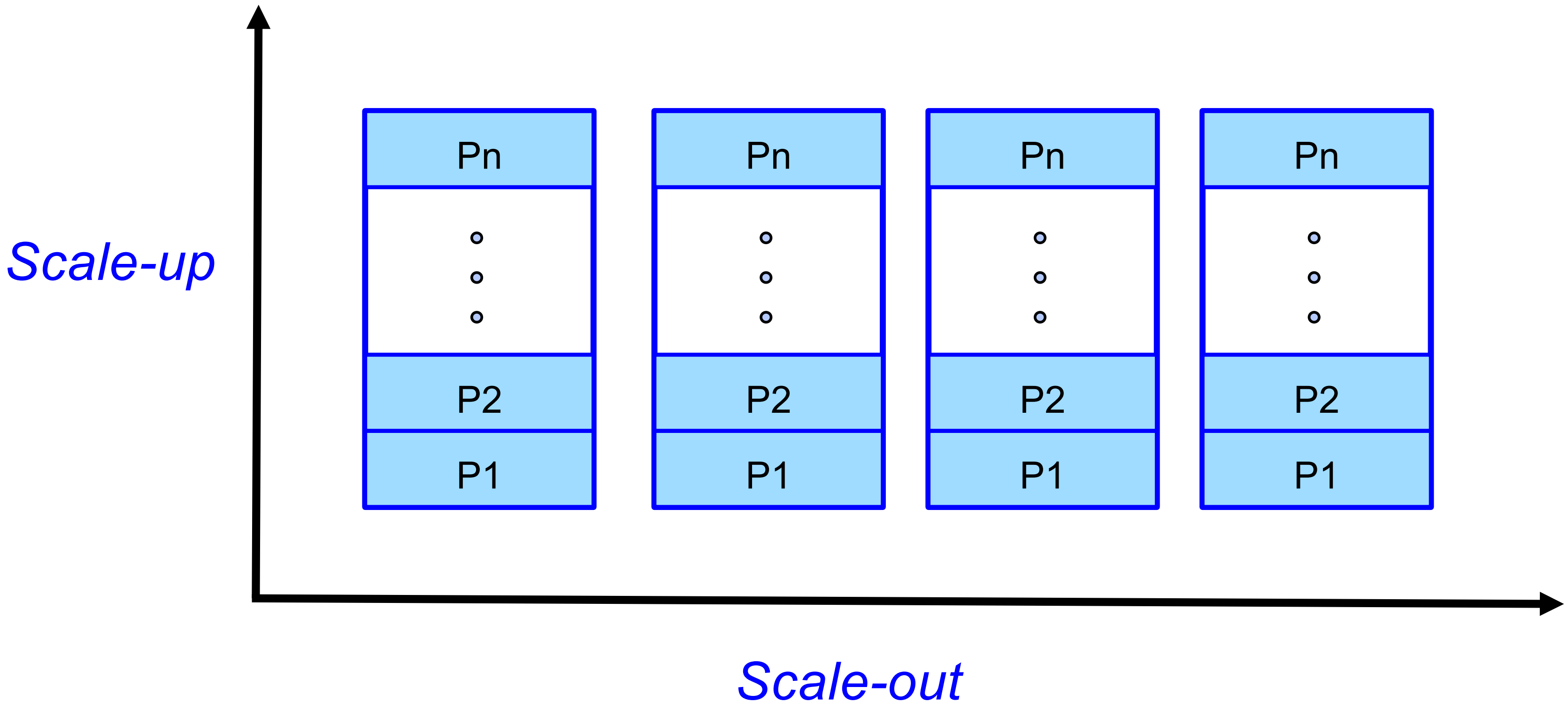
Scalability

- Ideal: linear scale-up
  - Sustained performance for a linear increase of database size and load
  - By proportional increase of components



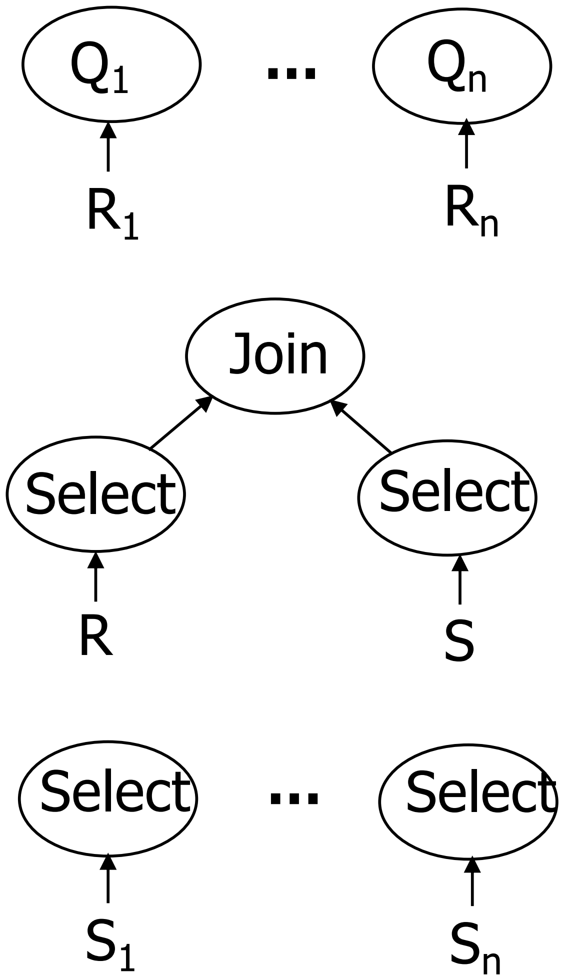
Vertical vs Horizontal Scaleup

- Typically in a computer cluster

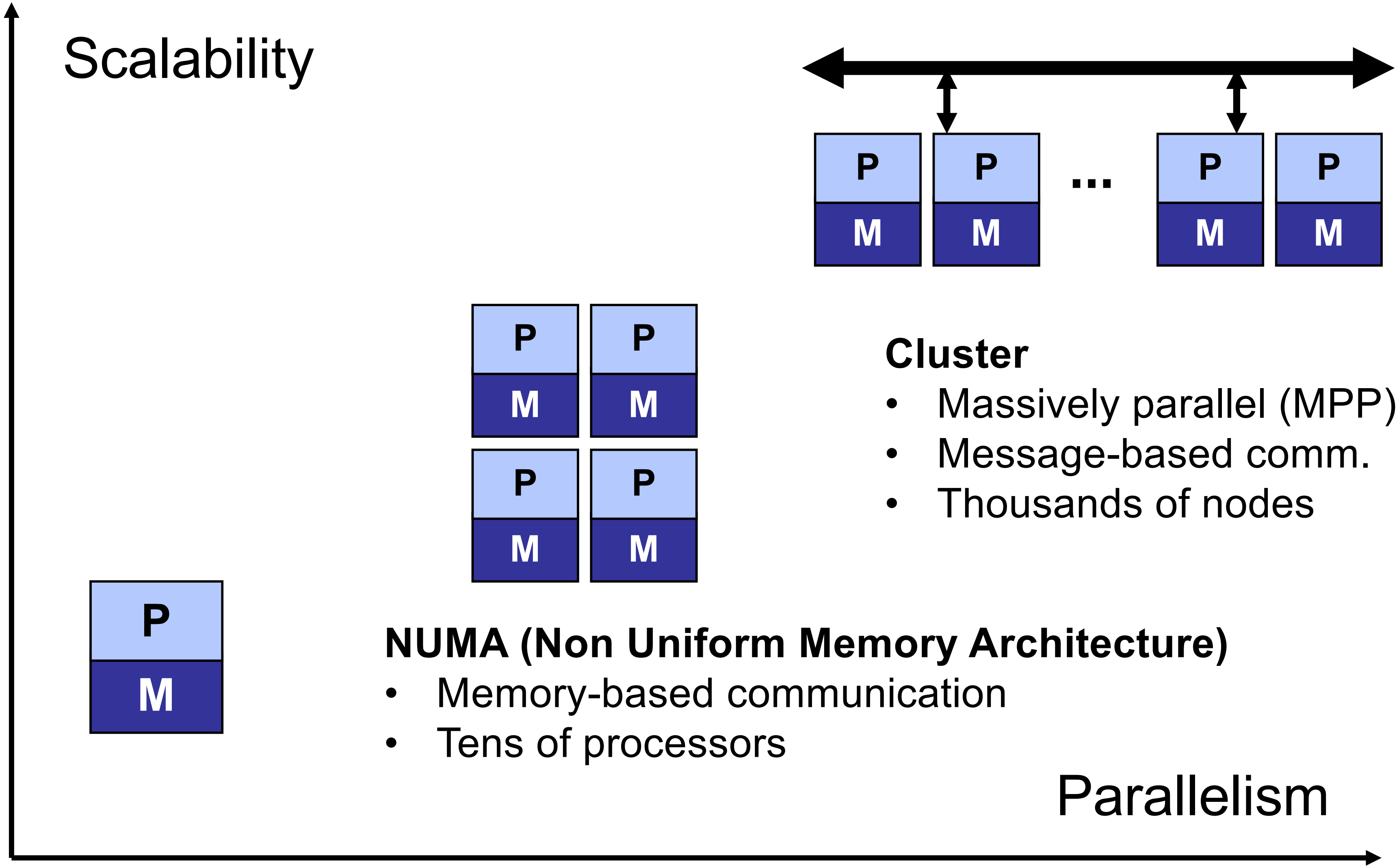


Query Parallelism

- Inter-query
  - Different queries on the same data
  - For concurrent queries
- Intra-query inter-operator
  - Different operations of the same query on different data
  - For complex queries
- Intra-query intra-operator
  - The same operation on different data
  - For large queries



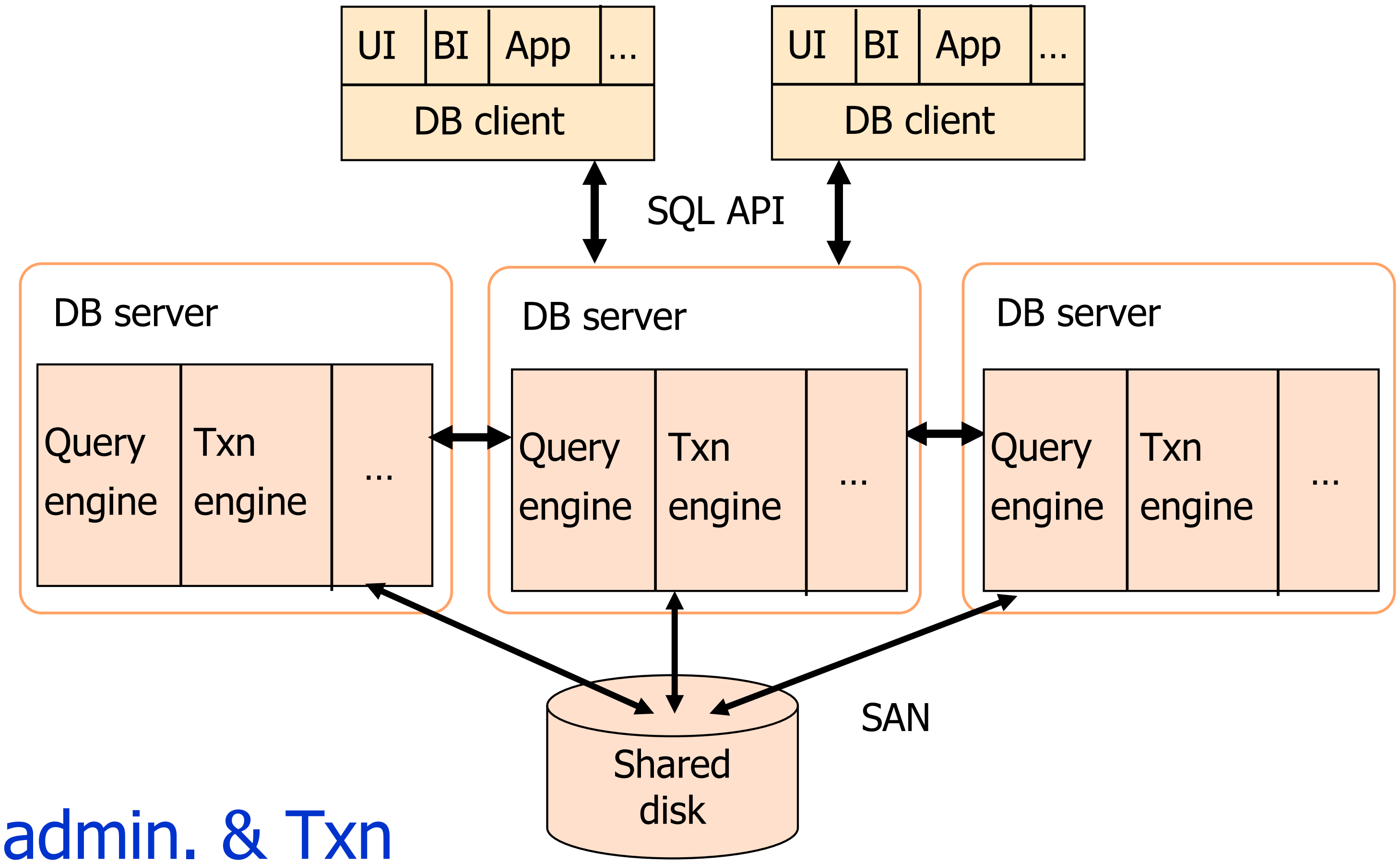
New Computing Hierarchy



Cluster
- Massively parallel (MPP)
- Message-based comm.
- Thousands of nodes

NUMA (Non Uniform Memory Architecture)
- Memory-based communication
- Tens of processors

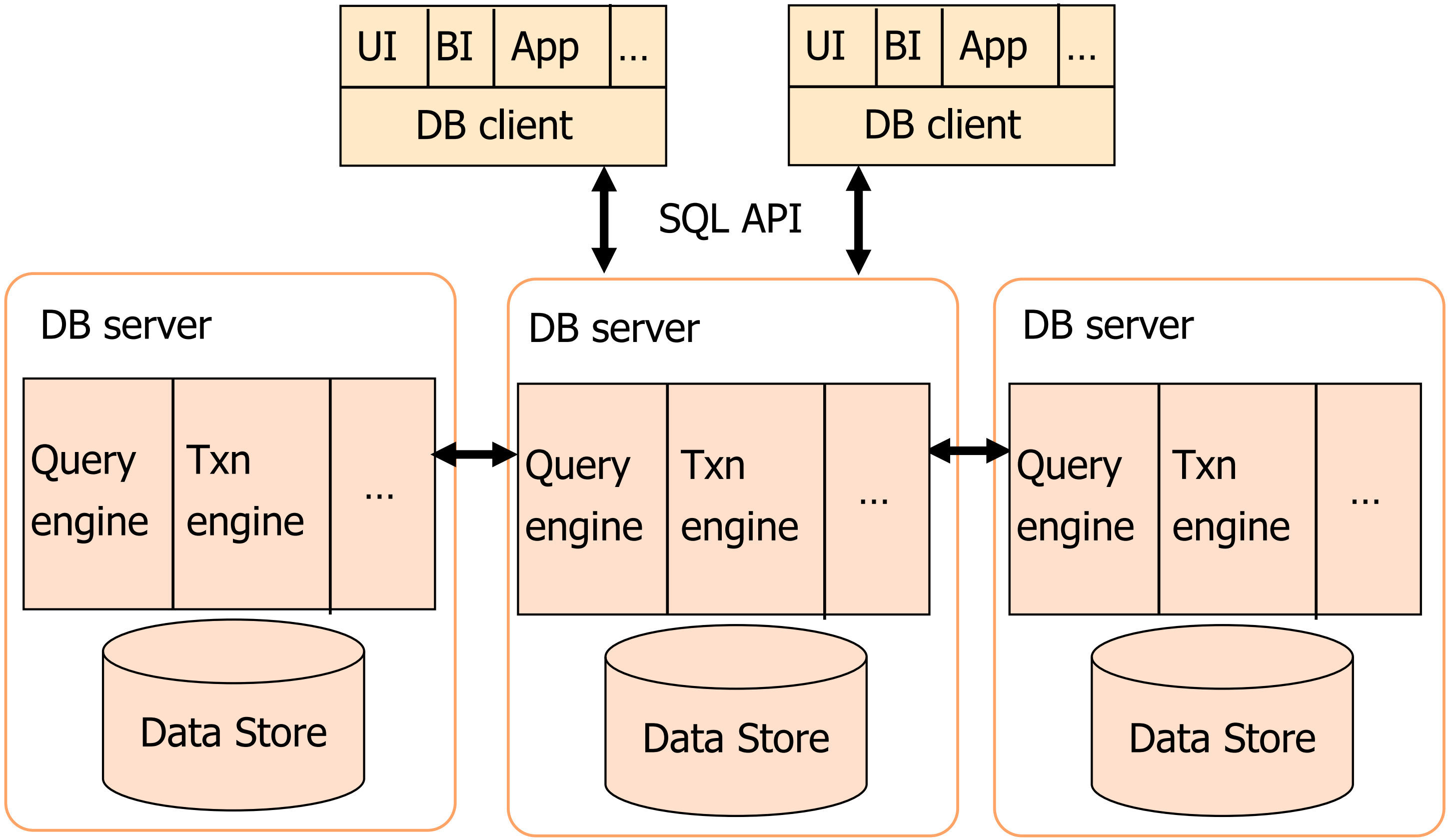
Shared-disk Architecture



+ Simple admin. & Txn
- Scalability (cache coherency)
- Cost of SAN

Best for OLTP

Shared-nothing Architecture



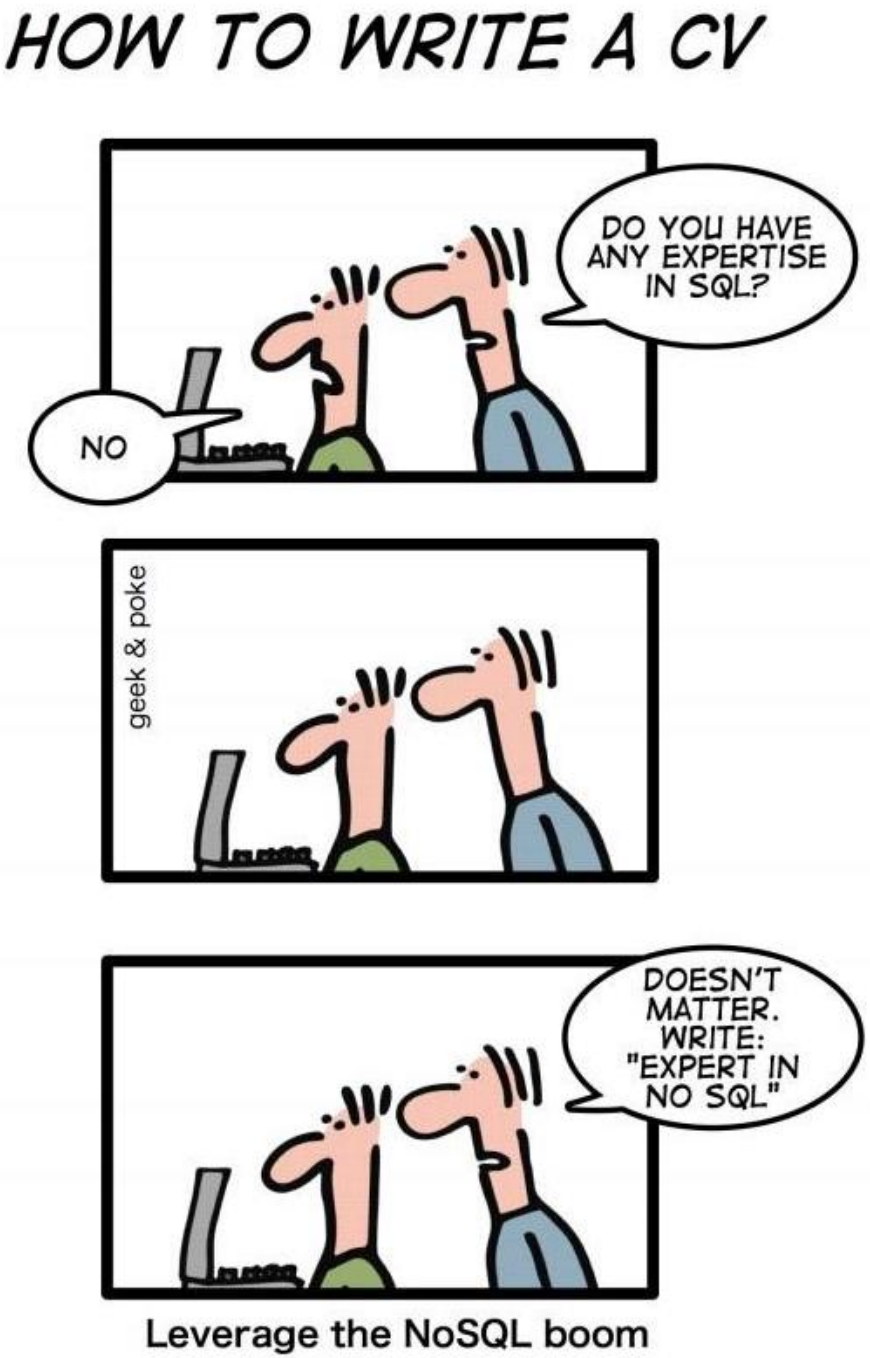
+ Scalability, cost
- Distributed updates

Perfect match for OLAP and big data (read intensive)

NoSQL

Not Only SQL

Why NoSQL?


HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

Why NoSQL?

- Trends
  - Big data
    - Unstructured data
  - Data interconnection
    - Hyperlinks, tags, blogs, etc.
  - Very high scalability
    - Data size, data rates, concurrent users, etc.
- Limits of SQL systems (in fact RDBMSs)
  - Need for skilled DBA, tuning and well-defined schemas
  - Full SQL complex
  - Hard to make updates scalable
    - Parallel RDBMS use a shared-disk for OLTP, which is hard to scale

NoSQL (Not Only SQL) Definition

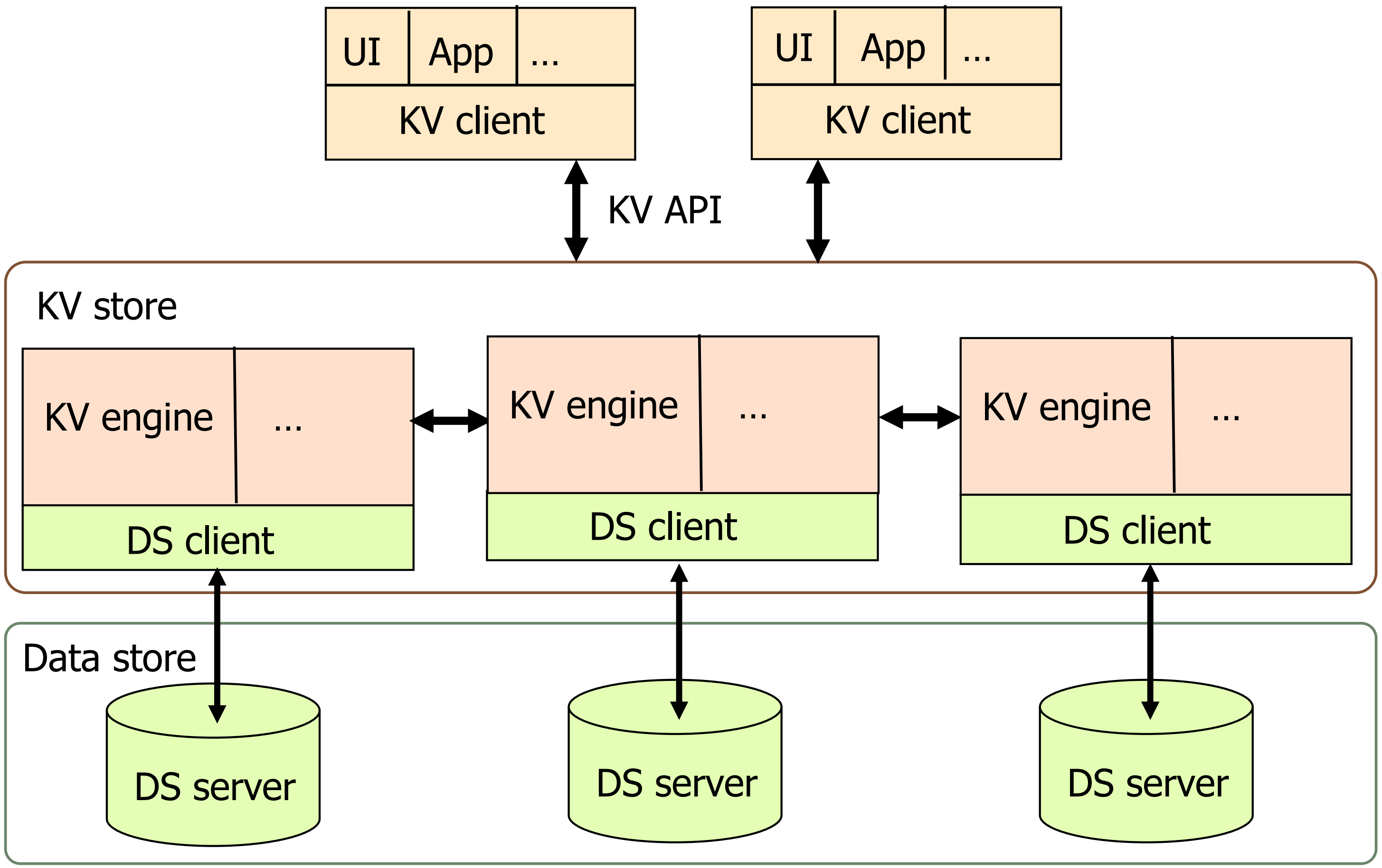
- Specific DBMS, typically for web-based data
  - Specialized data model
    - Key-value, table, document, graph
  - Trade relational DBMS properties
    - Full SQL, ACID transactions, data independence
  - For
    - Simplicity (schema, basic API)
    - Horizontal scalability and performance
    - Flexibility for the programmer (integration with programming language)

NoSQL Approaches

- Characterized by the data model, in increasing order of complexity:
  1. Key-value: DynamoDB, RockDB, Redis
  2. Tabular: Hbase, Bigtable, Cassandra
  3. Document (JSON): MongoDB, Coubase, CouchDB, Expresso
  4. Graph: Neo4J, AllegroGraph, MarkLogic, RedisGraph
  5. Multimodel: OrientDB, ArangoDB

- What about object DBMSs or XML DBMSs?
  - Were there much before NoSQL
  - Sometimes presented as NoSQL
  - But no horizontal scalability

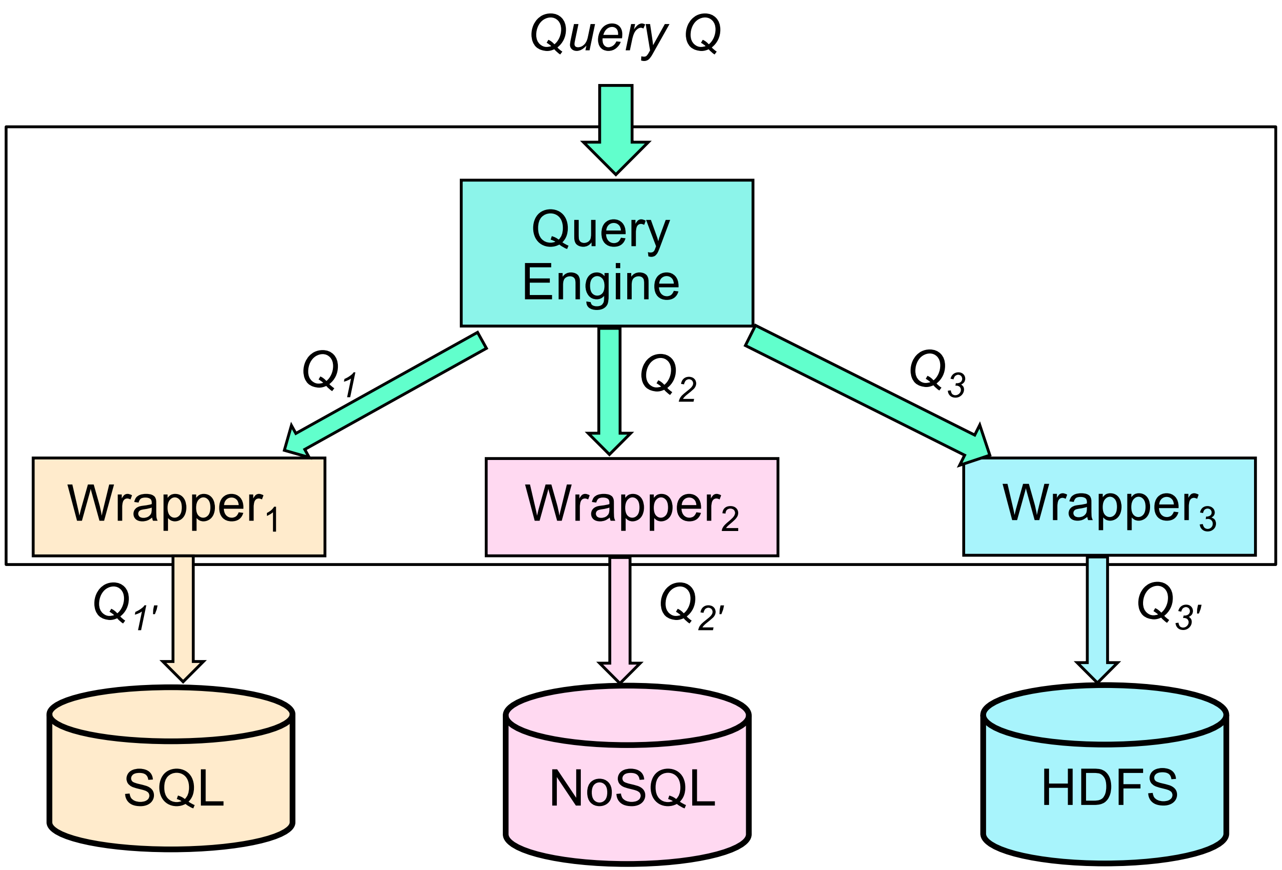
Key-value Store Distributed Architecture



Main NoSQL Systems

Vendor	Product	Category	Comments
Amazon	DynamoDB	KV	Proprietary
Apache	Cassandra Accumulo	KV Tabular	Open source, Orig. Facebook Open source, Orig. NSA
Couchbase	Couchbase	KV, document	Origin: MemBase
Google	Bigtable	Tabular	Proprietary, patents
FaceBook	RocksDB	KV	Open source
Hadoop	Hbase	Tabular	Open source, Orig. Yahoo
LinkedIn	Voldemort Expresso	KV Document	Open source ACID transactions
10gen	MongoDB	Document	Open source
Oracle	NoSQL	KV	Based on BerkeleyDB
OrientDB	OrientDB	Graph, KV, document	Open source, ACID transactions
Neo4J.org	Neo4J	Graph	Open source, ACID transactions
Ubuntu	CouchDB	Document	Open source

Polystores


Query Q
Query Engine
Q1
Q2
Q3
Wrapper1
Wrapper2
Wrapper3
Q1'
Q2'
Q3'
SQL
NoSQL
HDFS

Polystores*

- Also called *Multistores*
- Provide integrated access to multiple cloud data stores such as NoSQL, HDFS and RDBMS
- Great for integrating structured (relational) data and big data
- A major area of research & development

*Michael Stonebraker.The Case for Polystores. ACM blog, July 2015.

Differences with Federated Databases

- Multidatabase systems
  - A few databases (e.g. less than 10)
    - Corporate DBs
  - Powerful queries (with updates and transactions)
- Web data integration systems
  - Many data sources (e.g. 1000's)
    - DBs or files behind a web server
  - Simple queries (read-only)
- In the cloud, more opportunities for an efficient architecture
  - Less restriction to where mediator and wrapper components need be installed

Classification of Polystores*

- We divide polystores based on the level of coupling with the underlying data stores
  - Loosely-coupled
  - Tightly-coupled
  - Hybrid

*C. Bondiombouy, P. Valduriez. Query Processing in Cloud Multistore Systems: an overview. *Int. Journal of Cloud Computing*, 5(4): 309-346, 2016.

Loosely-Coupled Polystores

- Reminiscent of multidatabase systems
  - Mediator-wrapper architecture
  - Deal with autonomous data stores
  - One common interface to all data stores
  - Common interface translated to local API
- Examples
  - BigIntegrator (Uppsala University)
  - Forward (UC San Diego)
  - QoX (HP Labs)

Tightly-Coupled Polystores

- Use the local interfaces of the data stores
- Use a single query language for data integration in the query processor
- Allow data movement across data stores
- Optimize queries using materialized views or indexes
- Examples
  - Polybase (Microsoft Research, Madison)
  - HadoopDB (Yale Univ. & Brown Univ.)
  - Estocada (Inria)

Hybrid Polystores

- Support data source autonomy as in loosely-coupled systems
- Exploit the local data source interface as in tightly-coupled systems
- Examples
  - SparkSQL (Databricks & UC Berkeley)
  - BigDaWG (MIT, U. Chicago & Intel)
  - CloudMdsQL (Inria & LeanXcale)

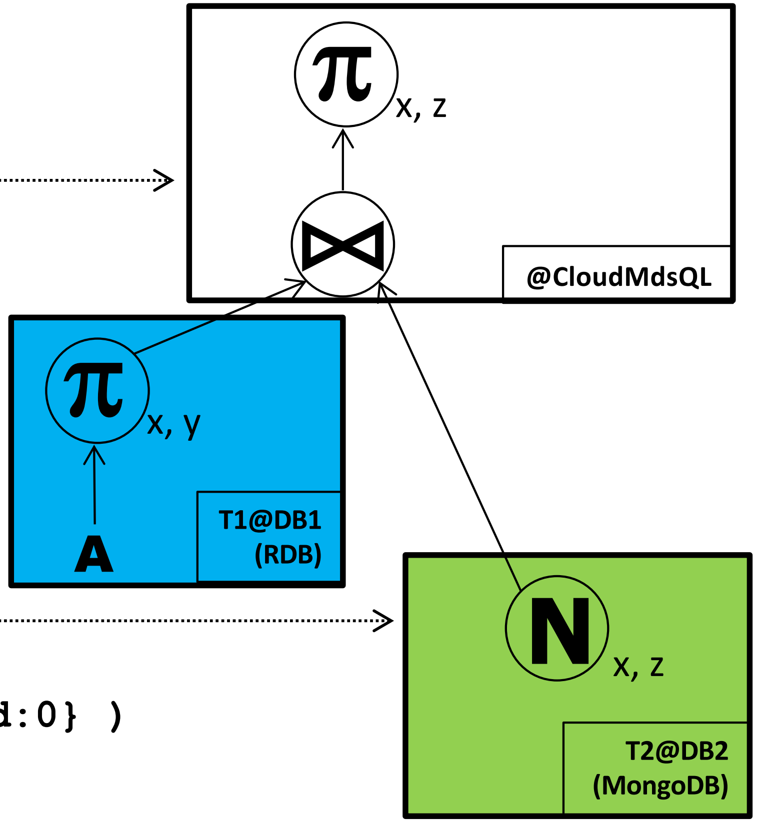
Polyglot Query Example

- A query in CloudMdsQL* that integrates data from
  - DB1 – relational (RDB)
  - DB2 – document (MongoDB)
    - Expressed as a Native subquery

```
/* Integration */
SELECT T1.x, T2.z
FROM T1 JOIN T2
  ON T1.x = T2.x

/* SQL sub-query */
T1(x int, y int)@DB1 =
( SELECT x, y FROM A )

/* Native sub-query */
T2(x int, z string)@DB2 = {*
  db.B.find( {$lt: {x, 10}}, {x:1, z:1, _id:0} )
*}
```



*B. Kolev, C. Bondiombouy, P. Valduriez, R. Jiménez-Peris, R. Pau, J. Pereira. The CloudMdsQL Multistore System. SIGMOD 2016.

Comparisons: functionality

Polystore	Objective	Data model	Query language	Data stores
Loosely-coupled				
BigIntegrator	Querying relational and cloud data	Relational	SQL-like	BigTable,RDBMS
Forward	Unyfing relational and NoSQL	JSON-based	SQL++	RDBMS, NoSQL
QoX	Analytic data flows	Graph	XML based	RDBMS, ETL
Tightly-coupled				
Polybase	Querying Hadoop from RDBMS	Relational	SQL	HDFS, RDBMS
HadoopDB	Querying RDBMS from Hadoop	Relational	SQL-live (HiveQL)	HDFS, RDBMS
Estocada	Self-tuning	No common model	Native QL	RDBMS, NoSQL
Hybrid				
SparkSQL	SQL on top of Spark	Nested	SQL-like	HDFS, RDBMS
BigDAWG	Unifying relational and NoSQL	No common model	Island query languages	RDBMS, NoSQL, Array DBMS, DSMSs
CloudMdsQL	Querying relational and NoSQL	JSON-based	SQL-like with native subqueries	RDBMS, NoSQL, HDFS

Comparisons: implementation techniques

Polystore	Objective	Data model	Query language	Data stores
Loosely-coupled				
BigIntegrator	Importer,absorber, finalizer	LAV	Access filters	Heuristics
Forward	Query processor	GAV	Data store capabilities	Cost-based
QoX	Dataflow engine	No	Data/function shipping	Cost-based
Tightly-coupled				
Polybase	HDFS bridge	GAV	Query splitting	Cost-based
HadoopDB	SMS planer, db conector	GAV	Query splitting	Heuristics
Estocada	Storage advisor	Materialzed views	Query rewriting	Cost-based
Hybrid				
SparkSQL	Dataframes	Nested	In-memory caching	Cost-based
BigDAWG	Island query	GAV within islands	Function/datashipping	Heuristics
CloudMdsQL	Query planner	No	Bind join	Cost-based