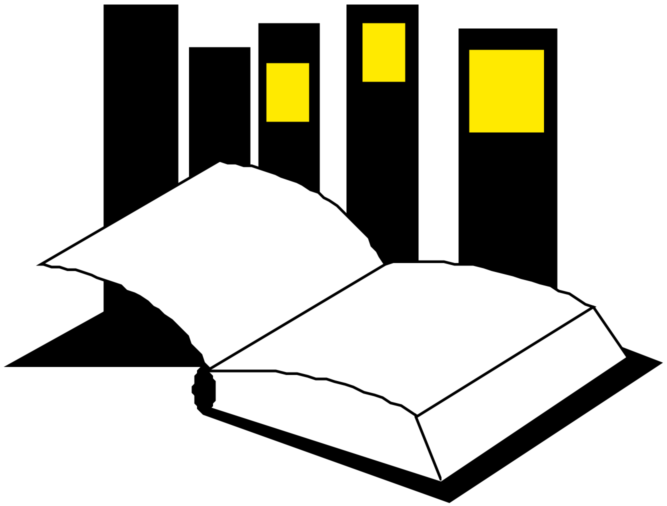
0. Course Overview

- Course Goal
- Course Content
- Course Structure



Distributed Database Systems

Tsinghua University

Digital Living

Users

Enterprise

SMB

Vertical Markets

Prosumer

Teen/Youth

Machines

Applications

Wireless Broadband

M-commerce

e-Learning

e-Health

Multimedia Broadcasting

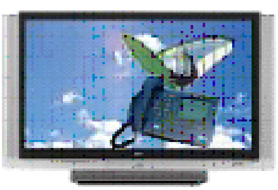
Vertical-specific:

Telematics;

RFID

Location-based

Digital possibilities
Digital living room Digital home Digital shop
Digital store Digital stories Digital story teller
Digital agencies Digital consulting
Digital copy right Digital communication
Digital media Digital broadcast
Digital signal processing,
Digital TV is coming to your mobile phone
Digital Voice recorder Digital iPod Digital TiVo
Digital Podcasters
Top 4 digital trading are
“cars, homes, appliances and dating”
Other top digital trading are
furniture, bedding, home entertainment, shoes,
travel and cool ware



Smart Phone

Network computers

System on Chip (SOC)

HDTV

Robots

Next Generation PC

Devices

IP Core

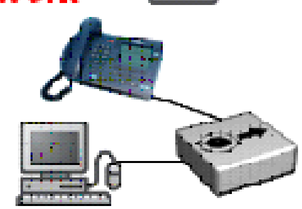
Fiber to the Premises

“Wireless Fabric”

Home Network

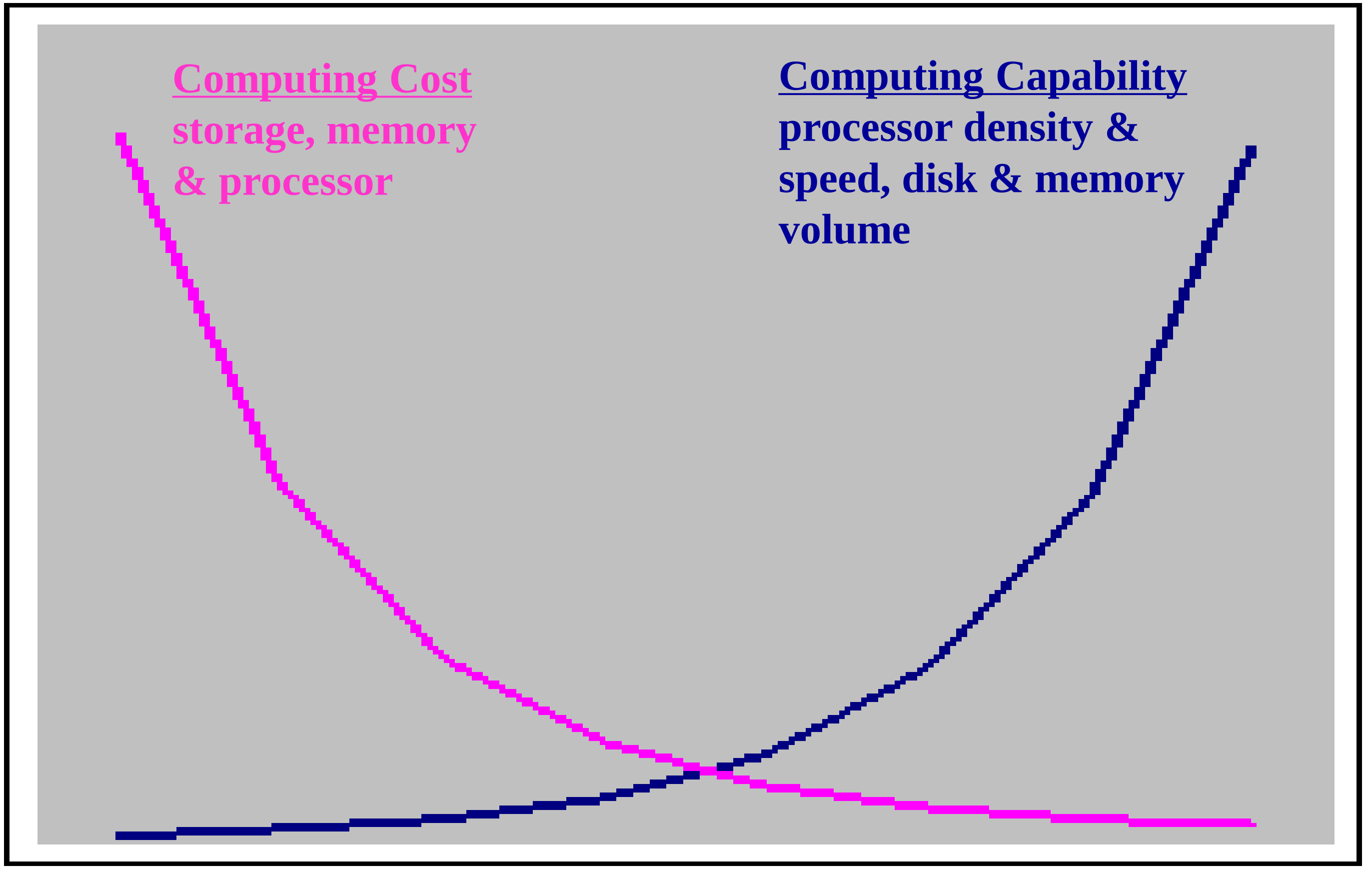
3G WWAN

Sensor Networks (RFID)



Networks

Moore's Law



ubiquitous computing & communication
huge volumes of digital data

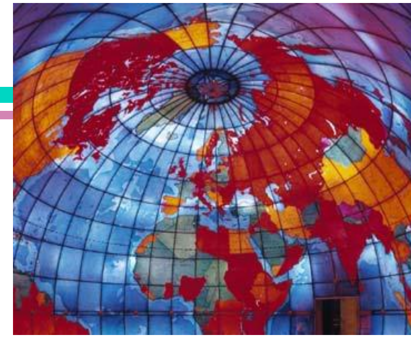
Society
3D People

Freedom

Expression

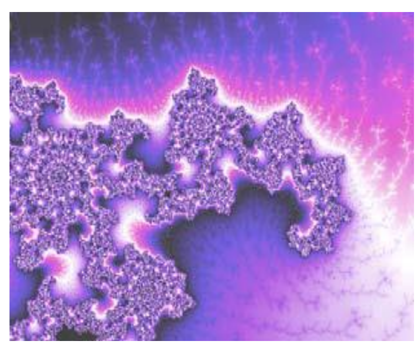
Care

Economy
3D Business



Globalization

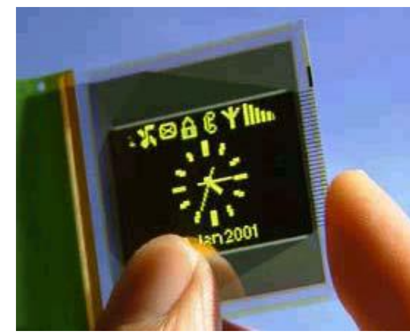
Experience Economy



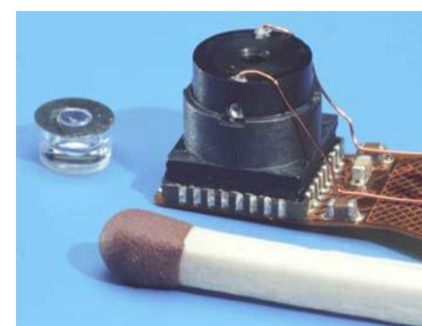
Creative Industry

Technology
3D Moore

More Moore



Large Area Electronics



Systems in Package

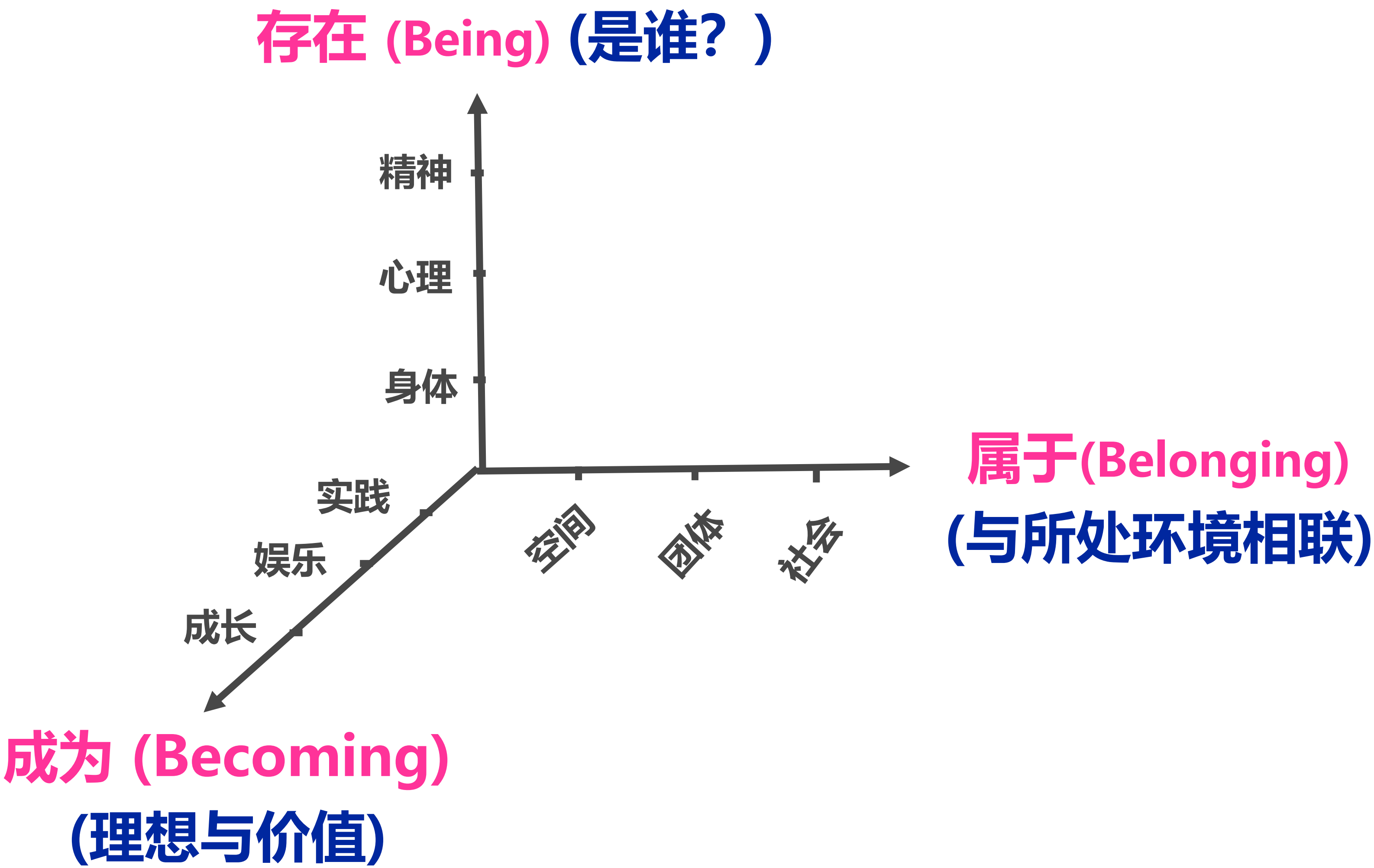
Ambient Intelligence

- Ambient intelligent environments
  - sensitive to people's needs
  - personalized to their requests
  - anticipatory of their behavior
  - responsive to their presence

- Emphasis
  - greater user-friendliness
  - more efficient services support
  - user-empowerment

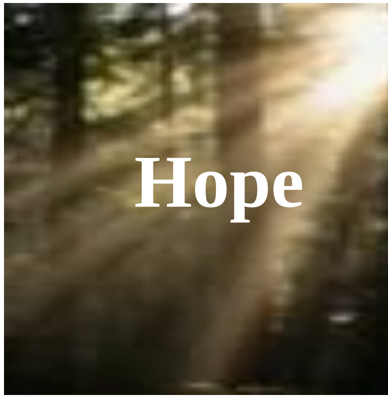
Improve Quality of Life!

Quality of Life

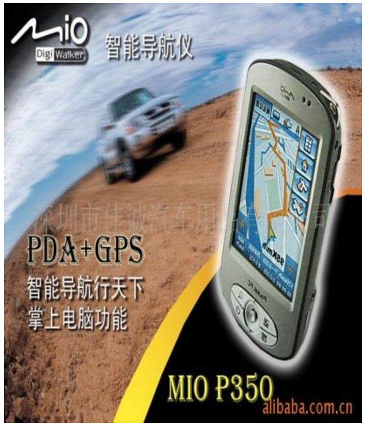


Ambient Environment

- Satisfy basic needs
- Provide opportunities
- Choice and initiative right



Smart Living



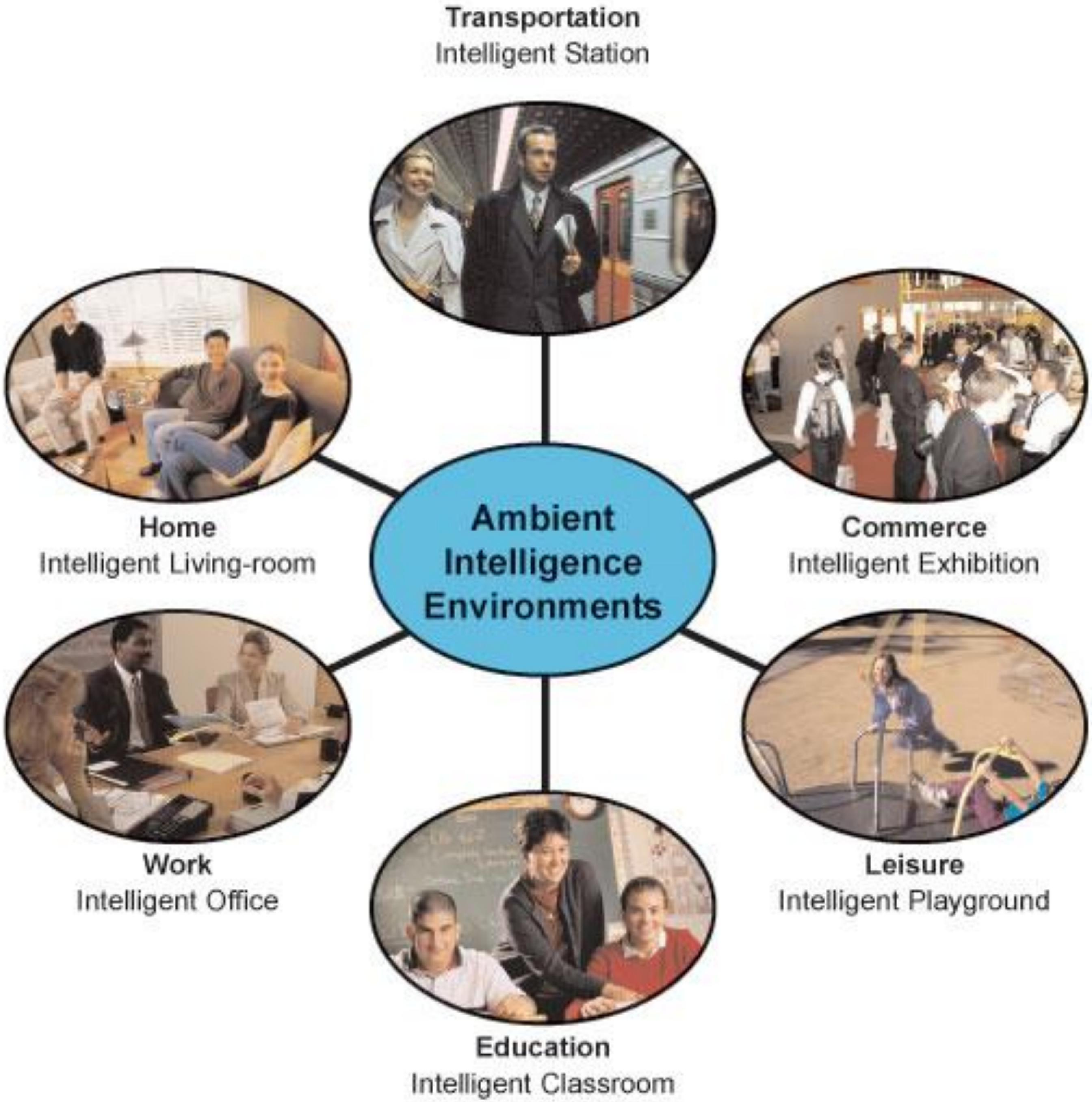
(From Philips)

Smart Kitchen

- The kitchen desktop is a projection display that can recognize food put on it, suggest what to cook, and play tutorial videos
- The table can weigh, time, and cook
- The waste bin can scan and identify the materials. The recyclable garbage is crushed, sorted, and disinfected. After that, it is vacuum-sealed and labeled for future use.
- The sink has two holes, one side keeps the relatively clean water for watering flowers and flushing the toilet; the other side throws away "black water" that cannot be used any more.

Smart Refrigerator

- The refrigerator knows the food inside. When the inventory is reduced to a certain extent, the refrigerator orders food from the online store, set by the user.
- The refrigerator has an electronic touch screen, which can display the calendar, weather, schedule, and phone number when the phone rings.


Transportation
Intelligent Station
Home
Intelligent Living-room
Ambient Intelligence Environments
Commerce
Intelligent Exhibition
Work
Intelligent Office
Leisure
Intelligent Playground
Education
Intelligent Classroom

Internet of Things

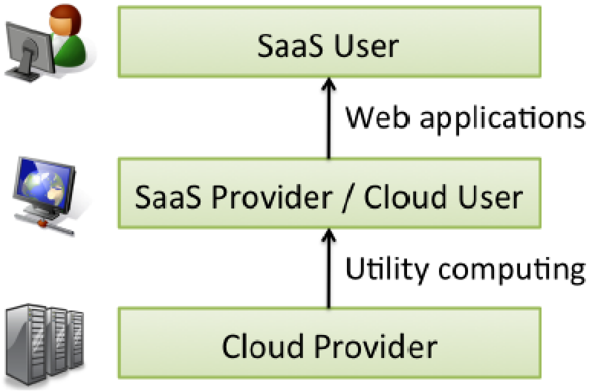
- The **Internet** of **Things** (IoT) is the network of physical objects or "**things**" embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

Interactive Connectivity

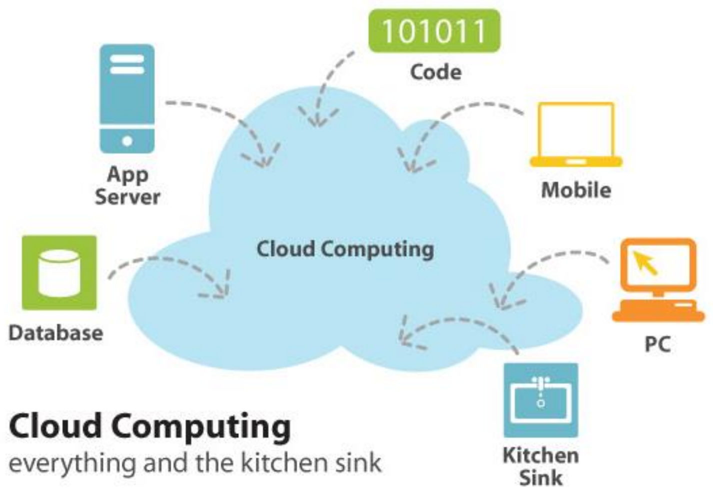
- Creating and Consuming web contents
- Creating value by making connections

Utility Computing – Cloud

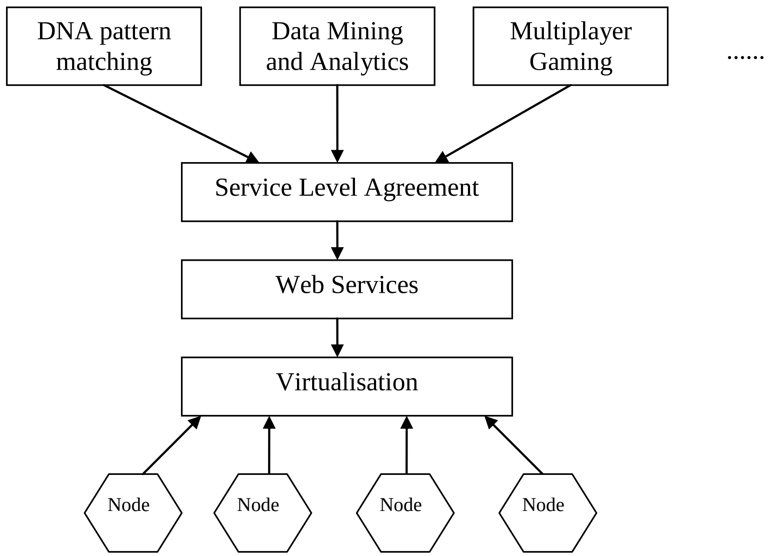
- Service
- Platform
- Utility

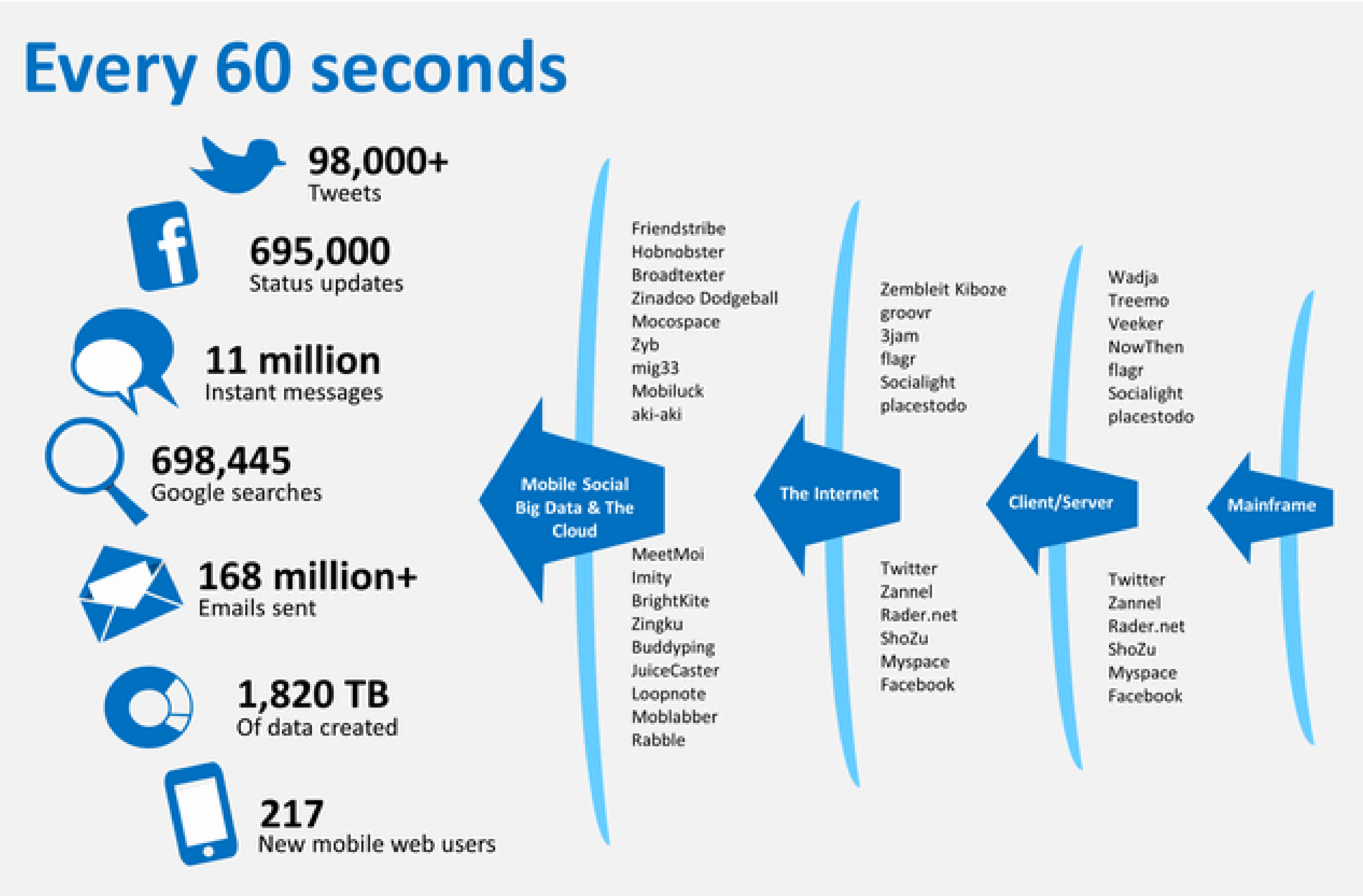


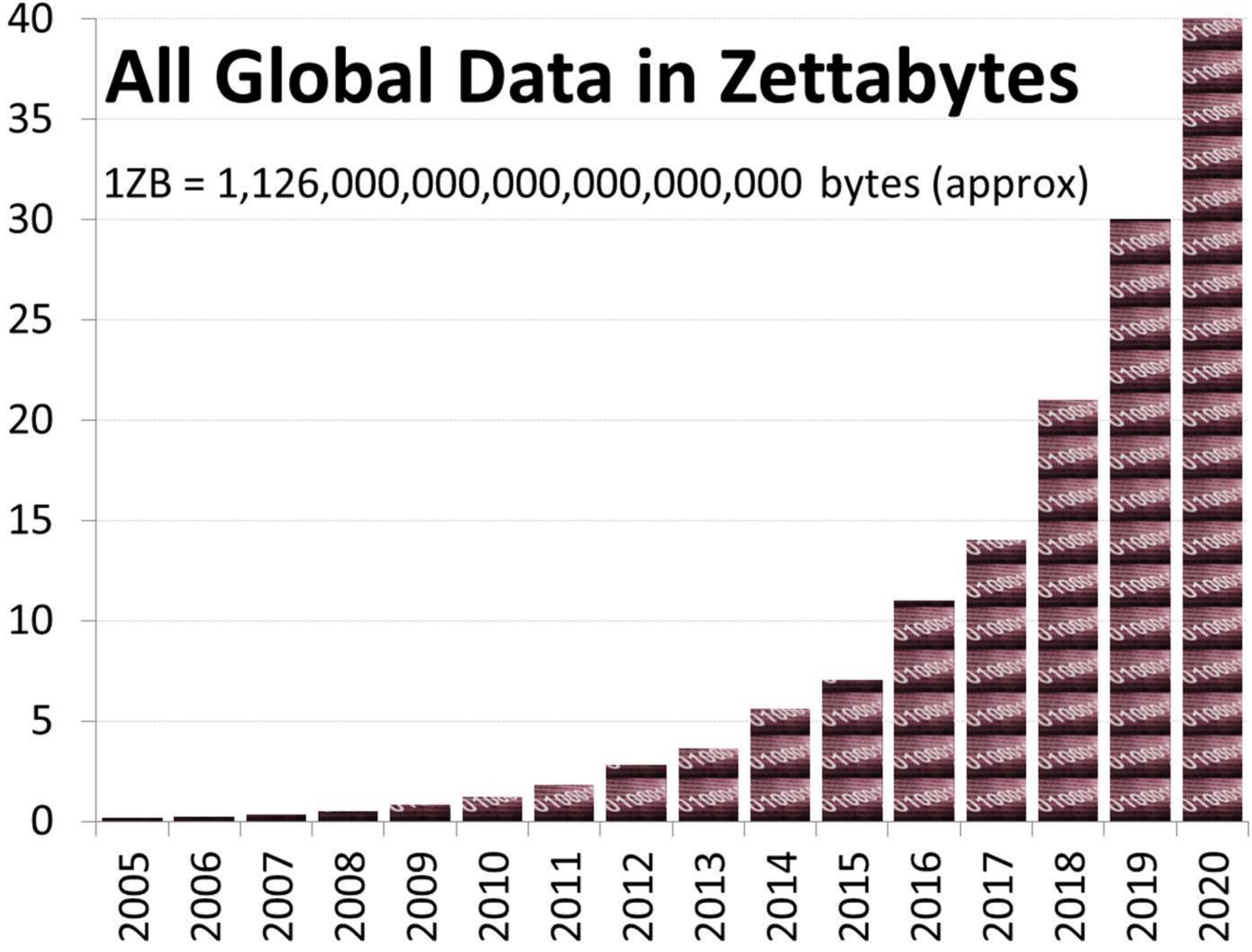
Above the Clouds: A Berkeley View of Cloud Computing 2009



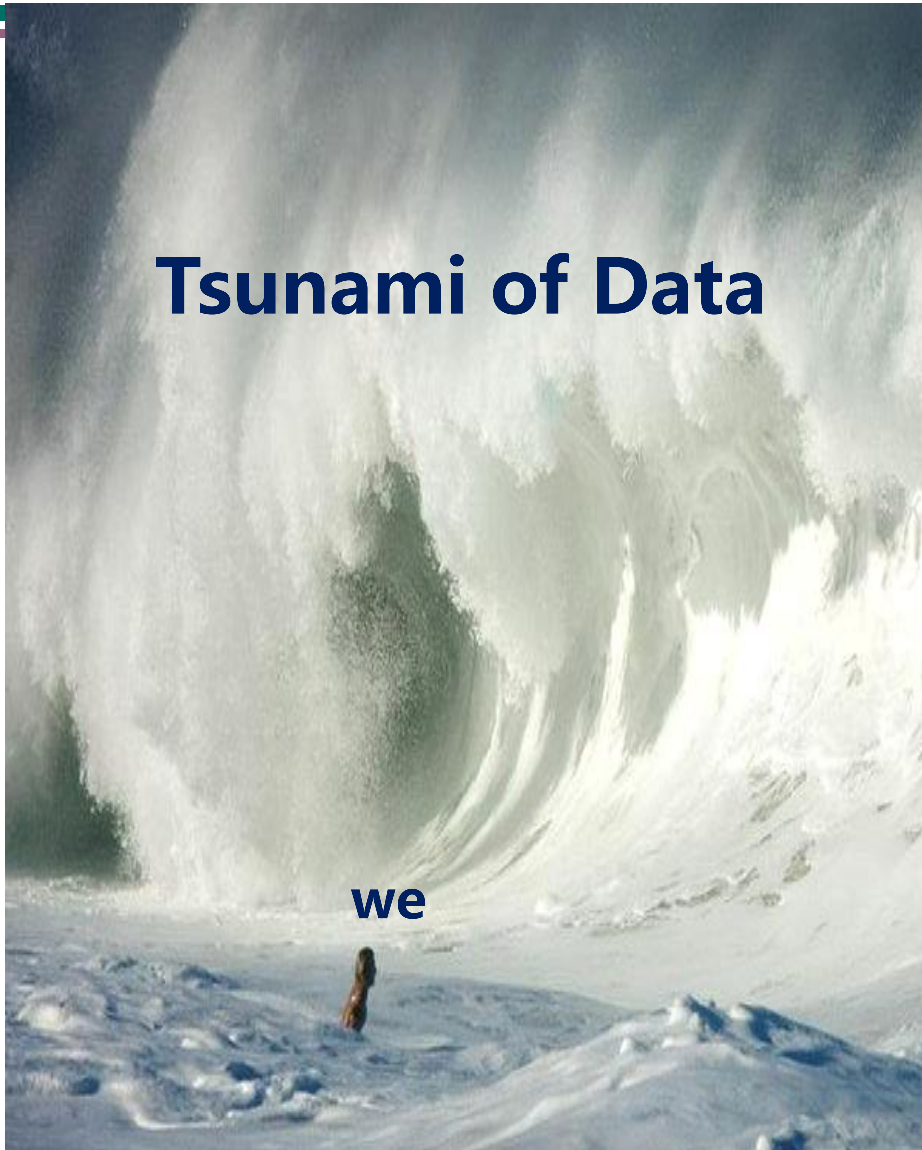
http://lonewolflibrarian.files.wordpress.com/2009/02/cloud-computing-kitchen-sink.jpg?w=510&h=364




Every 60 seconds
98,000+
Tweets
695,000
Status updates
11 million
Instant messages
698,445
Google searches
168 million+
Emails sent
1,820 TB
Of data created
217
New mobile web users
Mobile Social Big Data & The Cloud
Friendstribe
Hobnobster
Broadtexter
Zinadoo Dodgeball
Mocospace
Zyb
mig33
Mobiluck
aki-aki
MeetMoi
Imity
BrightKite
Zingku
Buddyping
JuiceCaster
Loopnote
Moblabber
Rabble
The Internet
Zembleit Kiboze
groovr
3jam
flagr
Socialight
placestodo
Twitter
Zannel
Rader.net
ShoZu
Myspace
Facebook
Client/Server
Wadja
Treemo
Veeker
NowThen
flagr
Socialight
placestodo
Twitter
Zannel
Rader.net
ShoZu
Myspace
Facebook
Mainframe


All Global Data in Zettabytes
1ZB = 1,126,000,000,000,000,000,000 bytes (approx)
40
35
30
25
20
15
10
5
0
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

Nature（2008年9月3日）

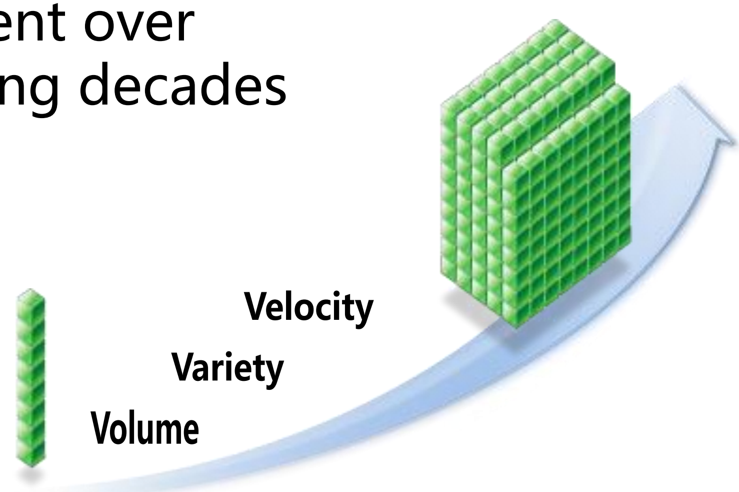


Entering Big Data Era

44 x

as much data and content over coming decades

2020
35 zettabytes



2009
800, 000 petabytes

Over 80%

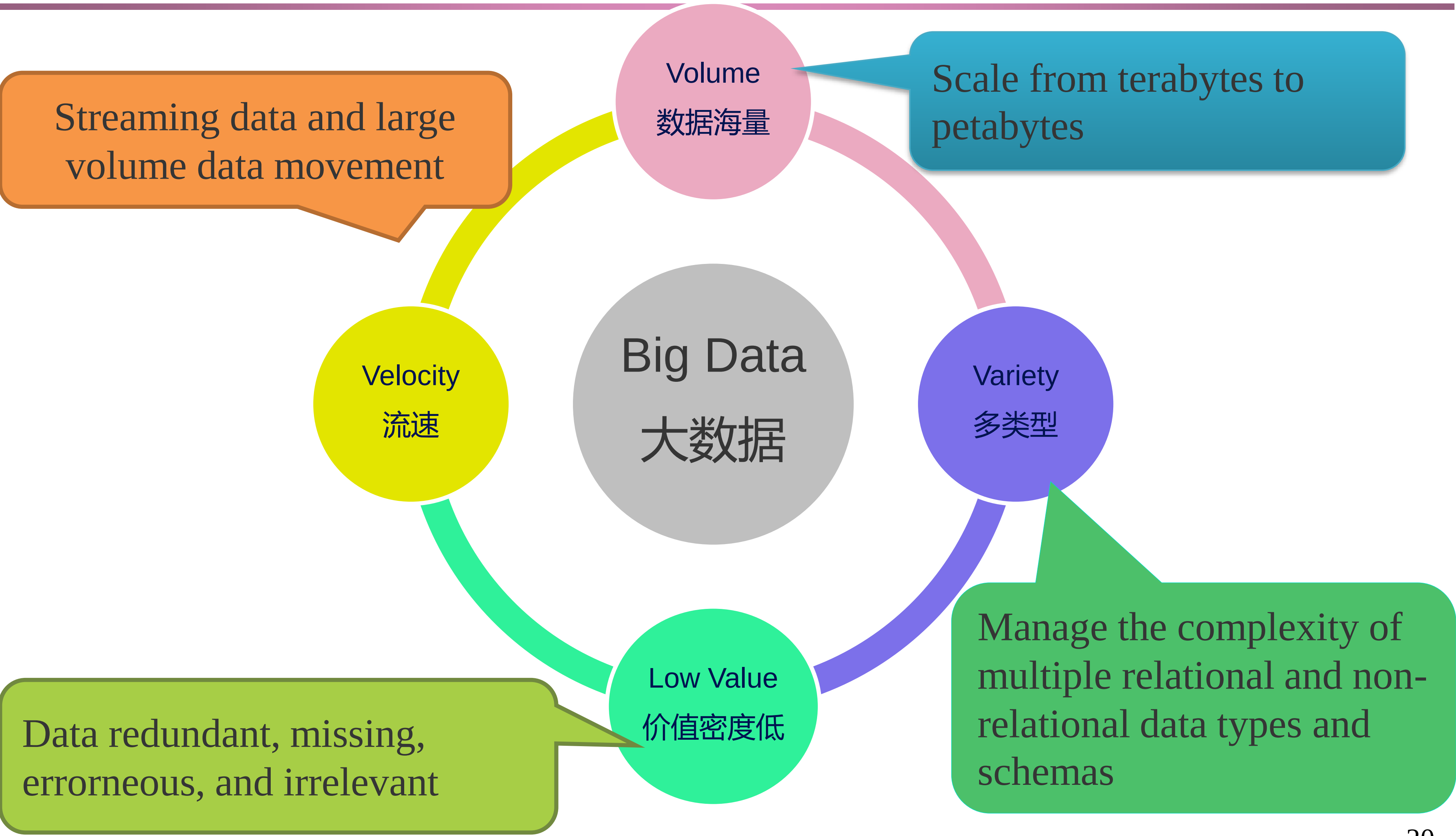
Unstructured data

Definition of Big Data

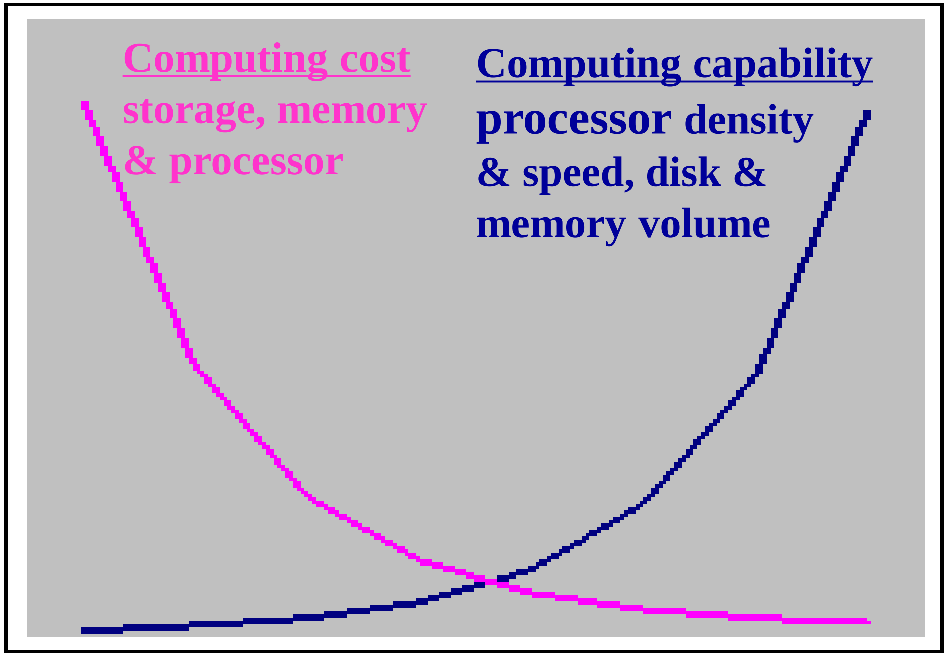
Big data **is an all-encompassing term for any collection of** ***data sets*** **so** ***large and complex*** **that it becomes** ***difficult*** **to process using** ***traditional*** **data processing applications**.

-- http://en.wikipedia.org/wiki/Big_data

Characteristics of Big Data



Moore Law' Exception



key problem in CS&T

Data Overload → Lack of Insight

1/2 say they feel overwhelmed by the amount of data their companies manage

1/3 business leaders make decision based on information they don’ t trust, or don’ t have.

60% say they need to do a better job capturing and understanding information rapidly.

83% cite “BI & Analytics” as part of their visionary plans to enhance competitiveness.

Wind Power's Prediction Accuracy

GE imagination at work

Vestas

VS

金风科技
GOLDWIND

95%

85%

Big Data Landscape

Vertical Apps
Predictive Policing
bloomreach
Myrrix

Log Data Apps
splunk
loggly
sumologic

Ad/Media Apps
rocketfuel
collective[i]
bluefin
Recorded Future
LuckySort
Media Science
Turn
DataXu

Data As A Service
kaggle
factual.
knoema
GNIP
DataSift
Windows Azure Marketplace
INRIX
LexisNexis
Space Curve
Loqate

Business Intelligence
ORACLE | Hyperion
SAP Business Objects
RJMetrics
Microsoft | Business Intelligence
IBM
Cognos
birst
Autonomy
MicroStrategy
QlikView
bime
Domo
Chart.io
GoodData

Analytics and Visualization
tableau
Palantir
Opera
metaLayer
Metamarkets
dataspora
centrifuge
Teradata Aster
sas
TIBCO
Karmasphere
panopticon
pentaho
Datameer
platfora
ClearStory
Cirro
alteryx
visual.ly
Ayata

Analytics Infrastructure
Hortonworks
Vertica
MapR
cloudera
Infobright
ParAccel
EMC
Greenplum
Netezza
kognitio
DataStax
Exasol
calpont

Operational Infrastructure
Couchbase
10gen
Teradata
Hadapt
Terracotta
VoltDB
MarkLogic
Informatica

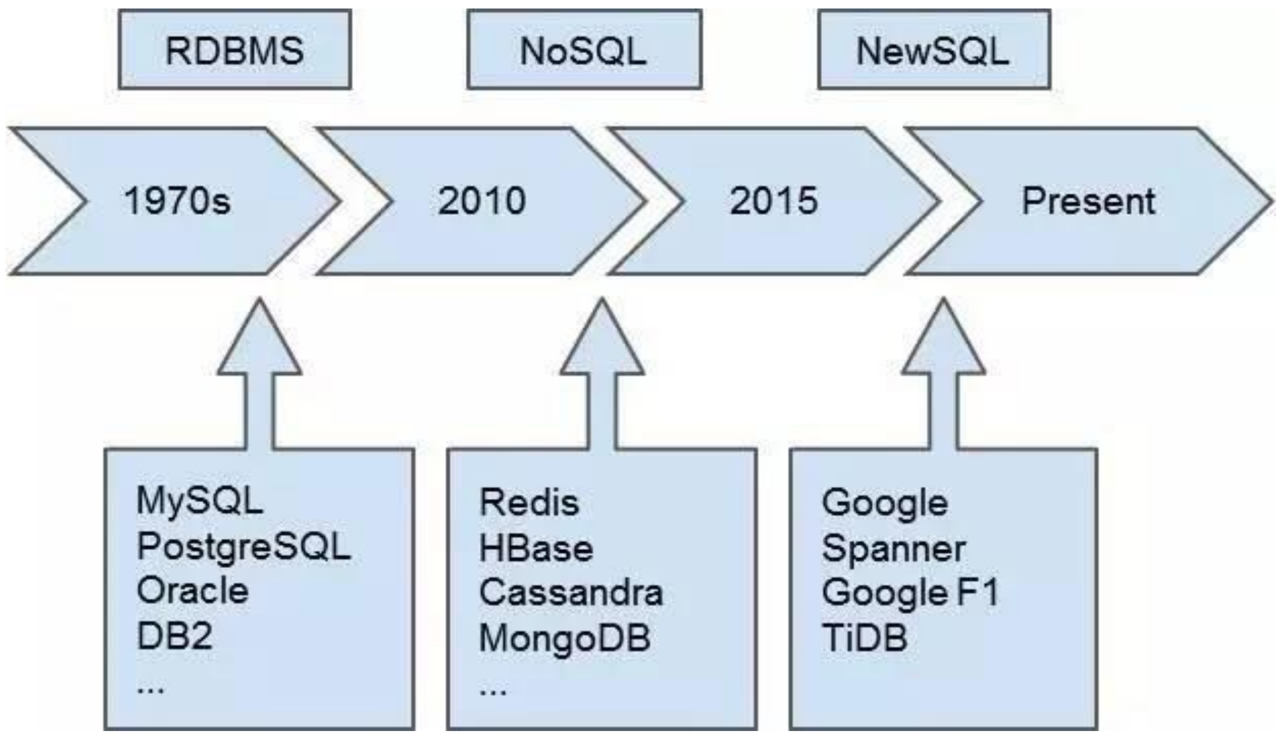
Infrastructure As A Service
amazon web services
Windows Azure
infochimps
Google BigQuery

Structured Databases
ORACLE
MySQL
SQL Server
PostgreSQL
IBM DB2
Sybase
memsql

Technologies
hadoop
hadoop MapReduce
mahout
Apache HBase
Cassandra

dave@vcdave.com
blogs.forbes.com/davefeinleib

DBMS Evolution ……



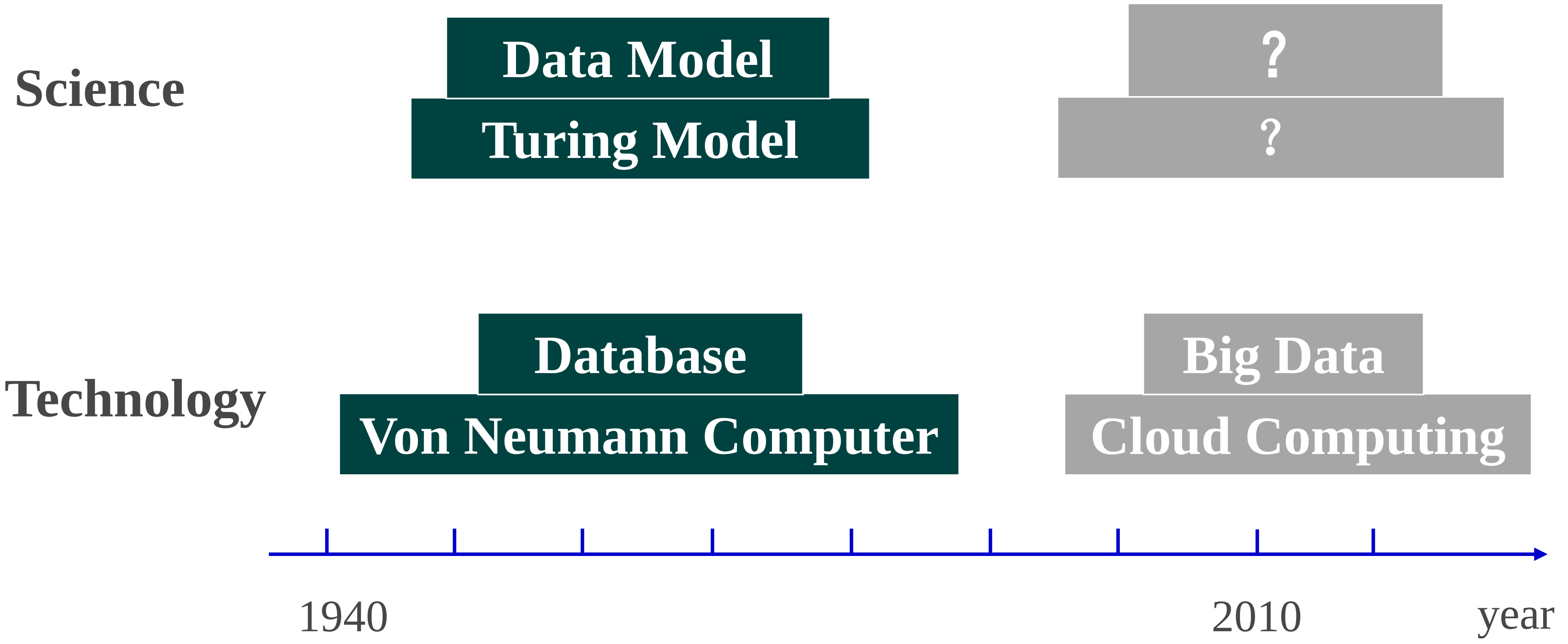
Describe/store/
process data
Consistency

Scale-up
Available

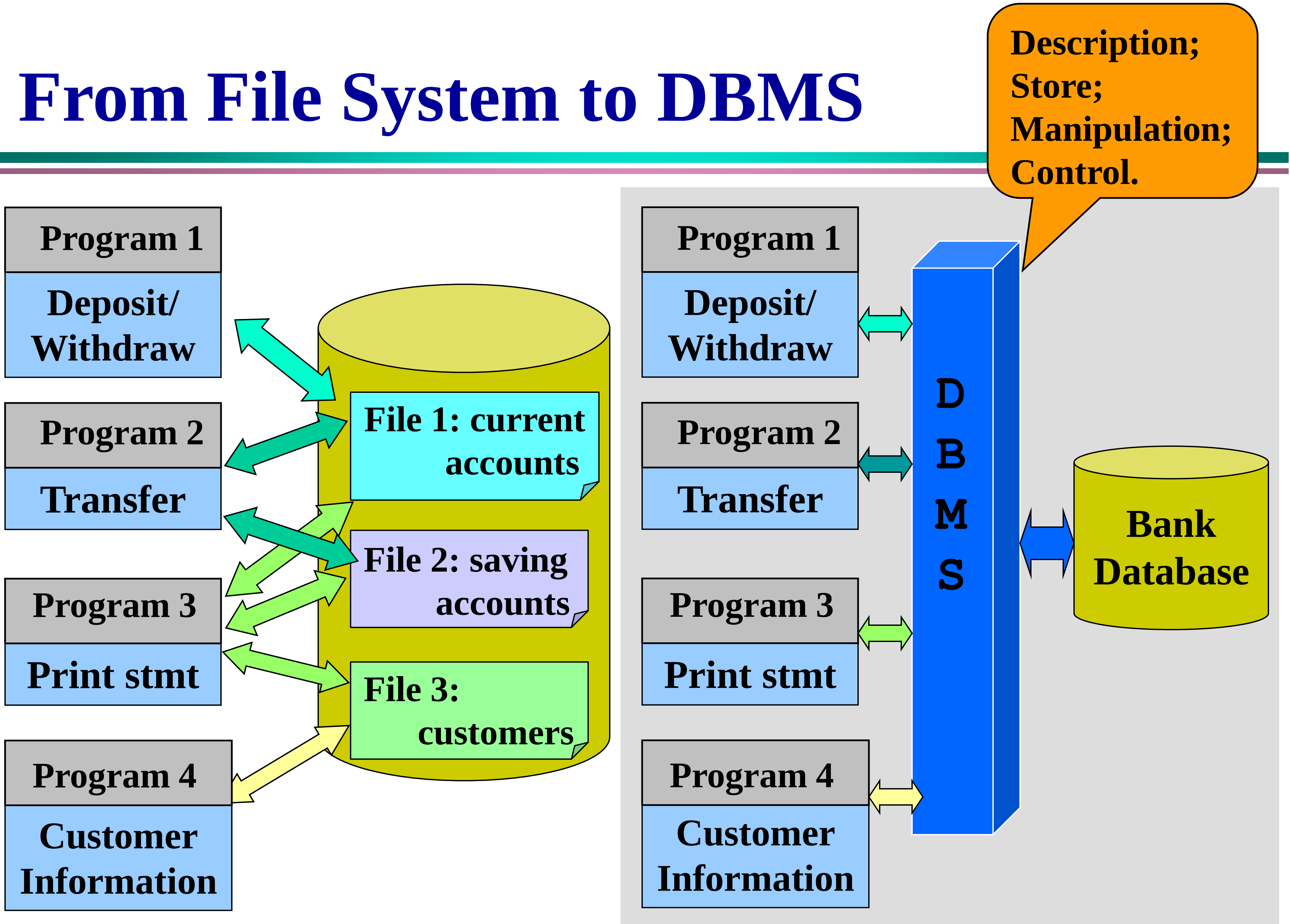
Both

Some Promising Solutions

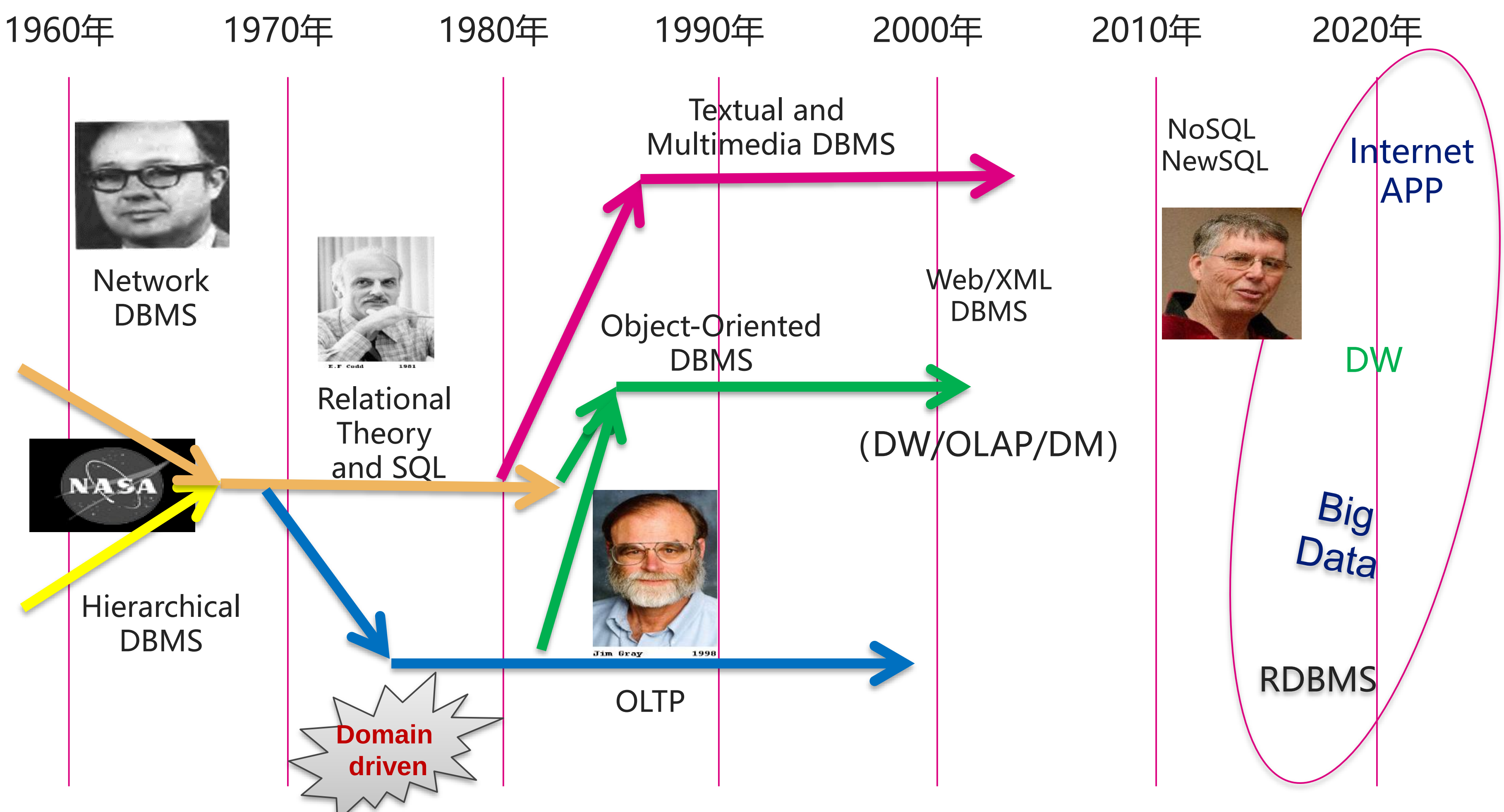
1. **On the Cloud**
2. **Multi-tenants**
3. **Combine OLAP and OLTP**
4. **Failure recovery re-usable and autonomous**


Science
Data Model
Turing Model
?
?
Technology
Database
Von Neumann Computer
Big Data
Cloud Computing
1940
2010
year

From File System to DBMS



Evolution of Data Management



From “One size fits all”
To “One size fits none”

Course Goal (1)

- to enhance the previous knowledge of database systems by deepening the understanding of the theoretical and practical aspects of the database technologies;
- to show the need for distributed database technology to tackle deficiencies of centralized database systems;
- to introduce basic principles and implementation techniques of distributed database systems, including distributed database design and architecture, query processing and optimization, transaction management, recovery, and reliability protocols;

Course Goal (2)

- to expose active and emerging research issues in distributed database systems and application development, including parallel and streaming data management, data warehousing, NoSQL and NewSQL big data management on the cloud;
- to apply theory to practice by building an application running in a distributed environment, serving at least millions of users.

Course Content

1. Theoretical study of distributed database systems

It covers the core of principles of distributed database management systems, including distributed database design and architecture, query processing and optimization, transaction management, concurrency control, failure recovery, and reliability. Latest developed data management technologies including parallel and streaming data management, data warehousing, NoSQL, and NewSQL database on the cloud will also be addressed.

2. Practice

Students are organized in teams to design and implement an application, running in a distributed environment, and serving multiple users.

Course Content - Theory

- Distributed database architecture
- Distributed database design
  (horizontal and vertical partitioning)
- Distributed database query processing and optimization
- Distributed database transaction management
- Distributed concurrency control
- Distributed reliability protocols
- Beyond traditional database technologies - big data on the cloud (SQL, NoSQL and NewSQL)
- Parallel and streaming data management
- Data warehousing and OLAP

Course Content - Practice

- Teamwork of 2 members
- Design and implement an application system running in a distributed environment
- Demonstration, documentation, and presentation

Course Assessment

1. Three Individual Assignments (30%)

- distributed database architecture and design
- distributed database querying
- distributed transaction management and concurrency control

2. 20-minute presentation on an advanced topic (20%)

- data storage, data migration, data backup, cache, multi-tenant database on the cloud, multiple-query processing, adaptive query optimization, indexing, natural language query interface, OLAP querying, continuous querying, failure recovery, long-life transaction management, etc.

3. Group Project (50%)

- Start-up Presentation (10%)
- Project Report (20%)
- Final presentation + Demo (20%) (this part is evaluated and given by students)

Project Evaluation Forms

Group Project Start-up Presentation (10%)

Evaluation Criteria	Problem Motivation	Literature Study of State-of-art solutions	Problem Definitio
Marking Scale	(8-10) Very Interesting (6-8) Interesting (4-6) Neural (2-4) Just-so-so (0-2) Boring	(8-10) Very Comprehensive (6-8) Comprehensive (4-6) Neural (2-4) Weak (0-2) Very Weak	(8-10) Very Clear (6-8) Clear (4-6) Neural (2-4) Vague (0-2) Very Vague
Your Mark			

Group Project Final Presentation + Demo (20%)

Evaluation Criteria	Methods	System Implementation	Experimental Setting for Evaluation
Marking Scale	(8-10) Very Good (6-8) Good (4-6) Neural (2-4) Weak (0-2) Very Weak	(8-10) Very Good (6-8) Good (4-6) Neural (2-4) Weak (0-2) Very Weak	(8-10) Very Clear (6-8) Clear (4-6) Neural (2-4) Vague (0-2)Very Vague
Your Mark			

Scientific Fostering

- basic concepts
- problem formulation
- identification of key technical issues and scientific challenges
- possible solutions
- algorithm presentation
- system level design and implementation
- testing
- maintenance

Prerequisites

- Can know or learn the knowledge of Database and Computer Network in the next two weeks.

- Can think, write, compile, run, and debu computer programs.
- Have the time and the will to work hard for a semester to achieve a worthy goal.

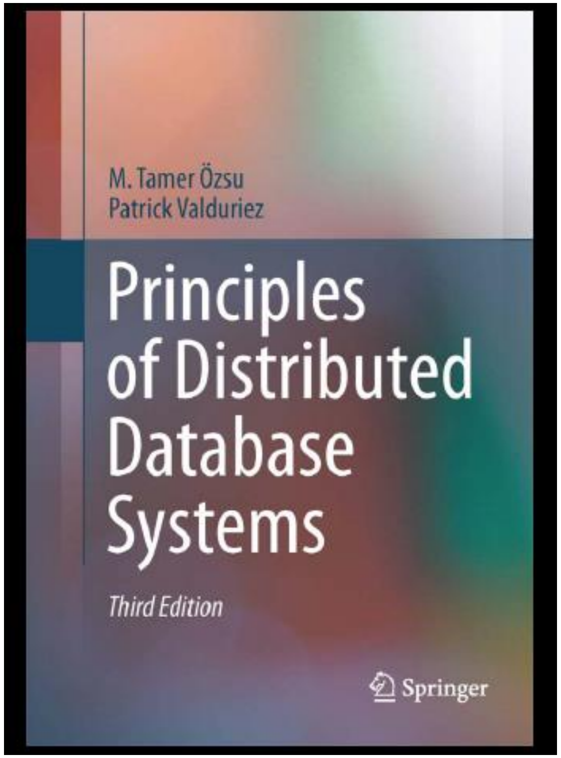
After the Course

- get familiar with the currently available models, technologies for and approaches to building distributed database systems;
- have developed practical skills in the use of these models and approaches to be able to select and apply the appropriate methods for a particular case;
- be aware of the current research directions in the field and their possible outcomes;
- be able to carry out research on a relevant topic, identify primary references, analyse them, and come up with meaningful conclusions;
- be able to apply learned skills to solving practical database related tasks.

Main Textbook

Principles of Distributed Database Systems

M. Tame Özsu
Patrick Valduriez
Prentice-Hall, 2011



Course Lecturer

- Ling Feng (冯铃)
  - Office: East Main Building #10-208
  - Phone: 62773581
  - Email: fengling@tsinghua.edu.cn

How to Succeed?

- Where there is a will, there is a "good"!
- Lecture overheads are designed to convey information, not to be mechanically memorized.
- LISTEN to what is said.
- THINK about what is said.
- ASK questions when you have doubts.
- WRITE down only what is important.
- PRACTICE what has learned.

Tip #1

Don't wait until the last minute to get help.

Tip #2

Bad things happen while learning a new skill. You will probably crash and burn on some programs. Start early; give yourself time for mistakes.

Tip #3

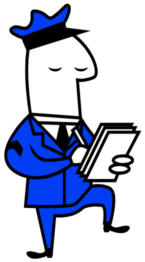
Don't be too ambitious with your course load. You CANNOT slack off in this class, even for a few days.

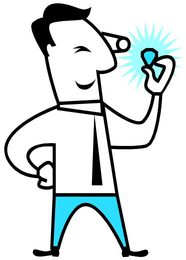
Tip #4

- Critical Thinking
- Research
- Problem Identification and Solving
- Speaking
- Writing
- Work Ethic
- Number Crunching
- Physical Performance
- Influencing People
- Teamwork

Ten things employers want you to learn in college.

Tip #5





Learn **Question** **Practice** **Enjoy**

Late Policy

- No delay penalty in emergent cases.
- You will be allowed 2 total late days without penalty for the entire semester. That is, you may be late by 1 day on two different deliverables or late by 2 days on one deliverable. Once those days are used, you will be penalized according to the following policy:
  - The deliverable is worth full credit on the due date.
  - It is worth half credit for the next 48 hours.
  - It is worth zero credit after that.

1. Introduction

Chapter 1

Introduction

Outline

- What is a distributed database system?
- Promises of DDBSs
- Complicating Factors
- Problem Areas

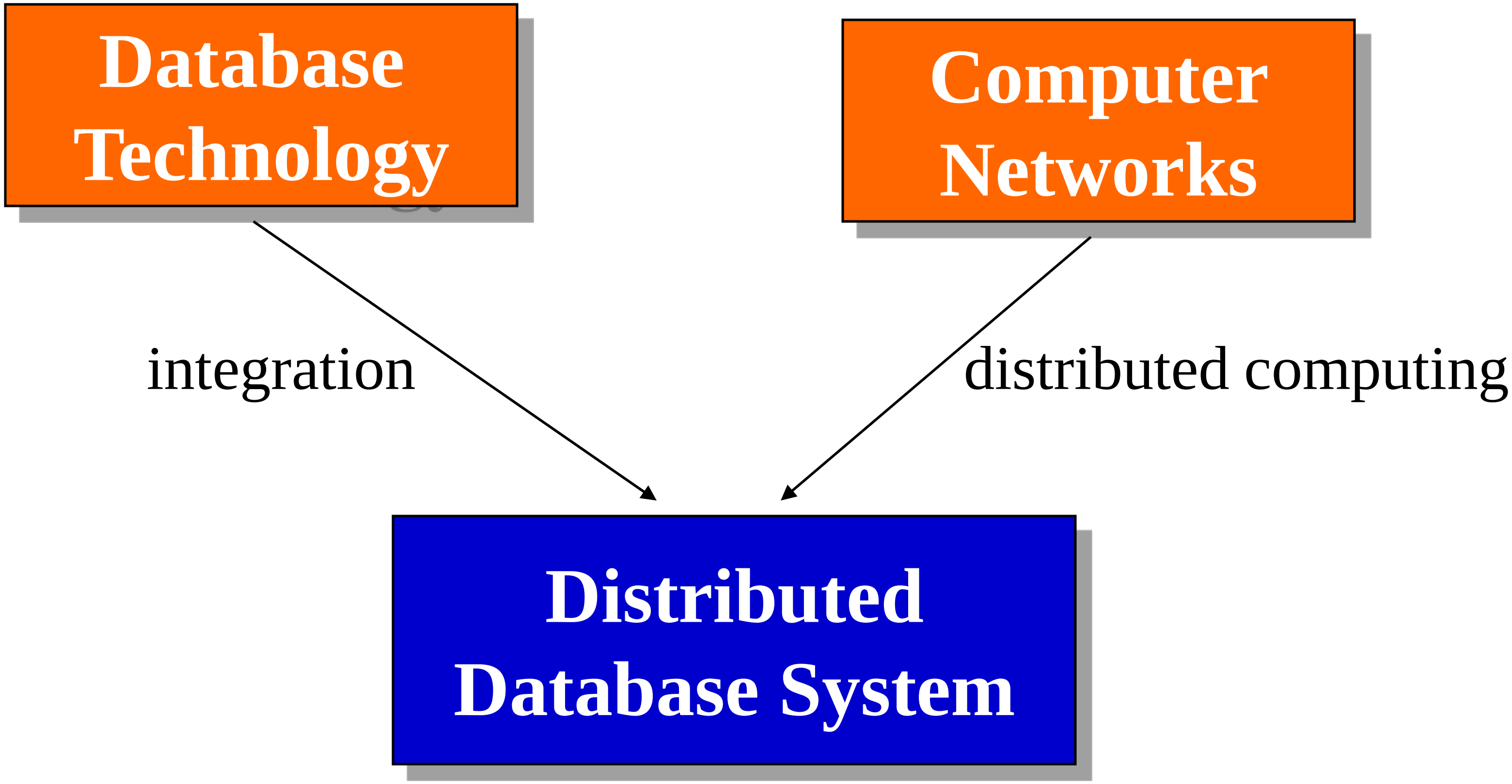
Motivating Example

- Multinational manufacturing company:
  - headquarters in New York
  - manufacturing plants in Chicago and Montreal
  - warehouses in Phoenix and Edmonton
  - R&D facilities in San Francisco
- Data and Information:
  - employee records (working location)
  - projects (R&D)
  - engineering data (manufacturing plants, R&D)
  - inventory (manufacturing, warehouse)

Features

- Data are distributed over sites (e.g. employee, inventory)
- Queries (e.g. “get employees who are younger than 45”) involve more than one site.

Distributed Database System Technology



- The key is integration, not centralization
- Distributed database technology attempts to achieve integration without centralization.

DDBS = Database + Network

- Distributed database system technology is the union of what appear to be diametrically opposed approaches to data processing
  - DDBS = database + network
  - Database integrates operational data of an enterprise to provide a centralized and controlled access to that data.
  - Computer network promotes a work mode that goes against all centralization efforts and facilitates distributed computing.

Distributed Computing

- A distributed computing system consists of
  - a number of autonomous processing elements (not necessarily homogeneous), which
    - are interconnected by a computer network
    - cooperate in performing their assigned tasks

- What is distributed?
  - Processing Logic
  - Function
  - Data
  - Control

All these are necessary and important for distributed database technology.

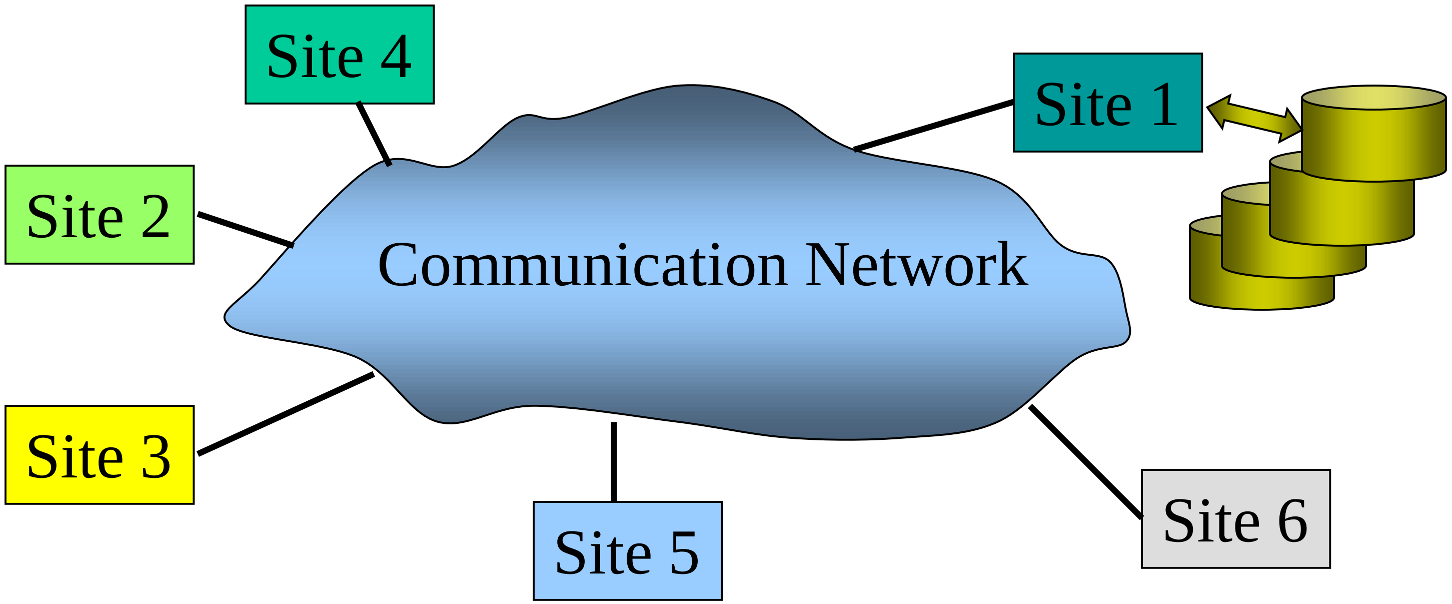
What is a Distributed Database System?

- A distributed database (DDB) is a collection of multiple, *logically interrelated* databases, distributed over a computer network
  - i.e., storing data on multiple computers (nodes) over the network
- A distributed database management system (DDBMS) is the software that
  - manages the DDB;
  - provides an access mechanism that makes this distribution transparent to the users.
- Distributed database system (DDBS):
  - **DDBS = DDB + DDBMS**

What is not a DDBS?

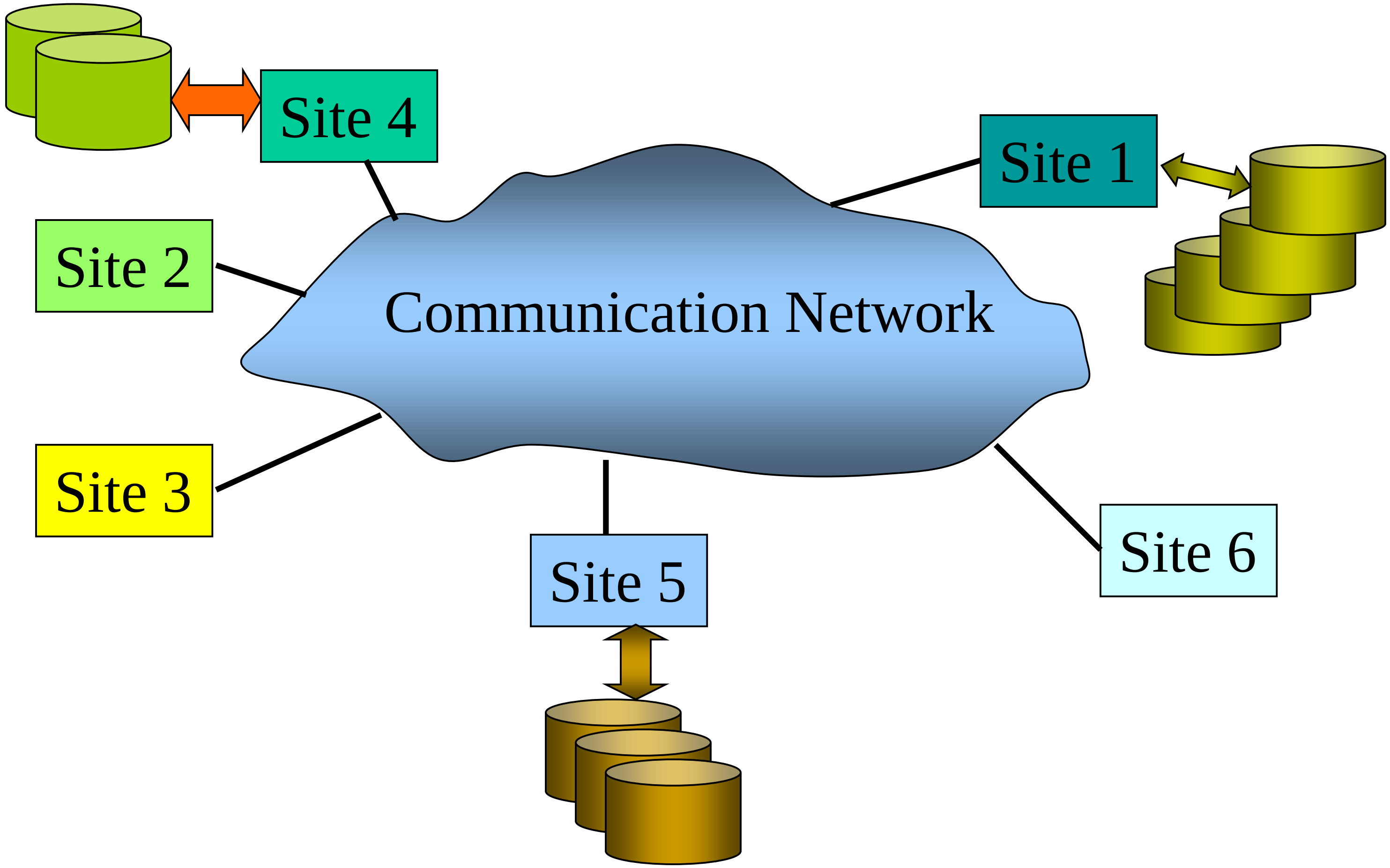
- A timesharing computing system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers
  - This is a centralized database on a network node

Centralized DBMS on a Network



- Data resides only at one node.
- Database management is the same as in a centralized DBMS.
- Remote processing, single-server multiple-clients

Distributed DBMS Environment



Applications of DDBMS

- Multi-plant manufacturing
- Military command and control
- Airlines
- Hotel chains
- Any organization which has a decentralized organization structure
- Big data era

Why DDBS?

- Several factors leading to the development of DDBS
  - Distributed nature of some database applications
  - Increased reliability and availability
  - Allowing data sharing while maintaining some measure of local control
  - Improved performance
  - Achieve scalability

Implicit Assumptions

- Data stored at a number of sites
  - Each site has processing power
- Processors at different sites are interconnected by a computer network
- Distributed database is a database, not a collection of files
  - Data logically related as exhibited in the users' access patterns (e.g., relational data model)
- DDBMS is a fully-fledged DBMS
  - Not remote file systems

Design Issues of Distributed Systems

- Transparent data distribution
- Reliable
  - Design should not require the simultaneous functioning of a substantial number of critical components
  - More redundancy, greater availability, and greater consistency
  - Fault tolerance, the ability to mask failures from the user
- Good performance
  - Important (the rest are useless without this)
  - Balance number of messages and grain size of distributed computations
- Scalable
  - A maximum for developing distributed systems
  - Avoid centralized components, tables, and algorithms
  - Only decentralized algorithms should be used

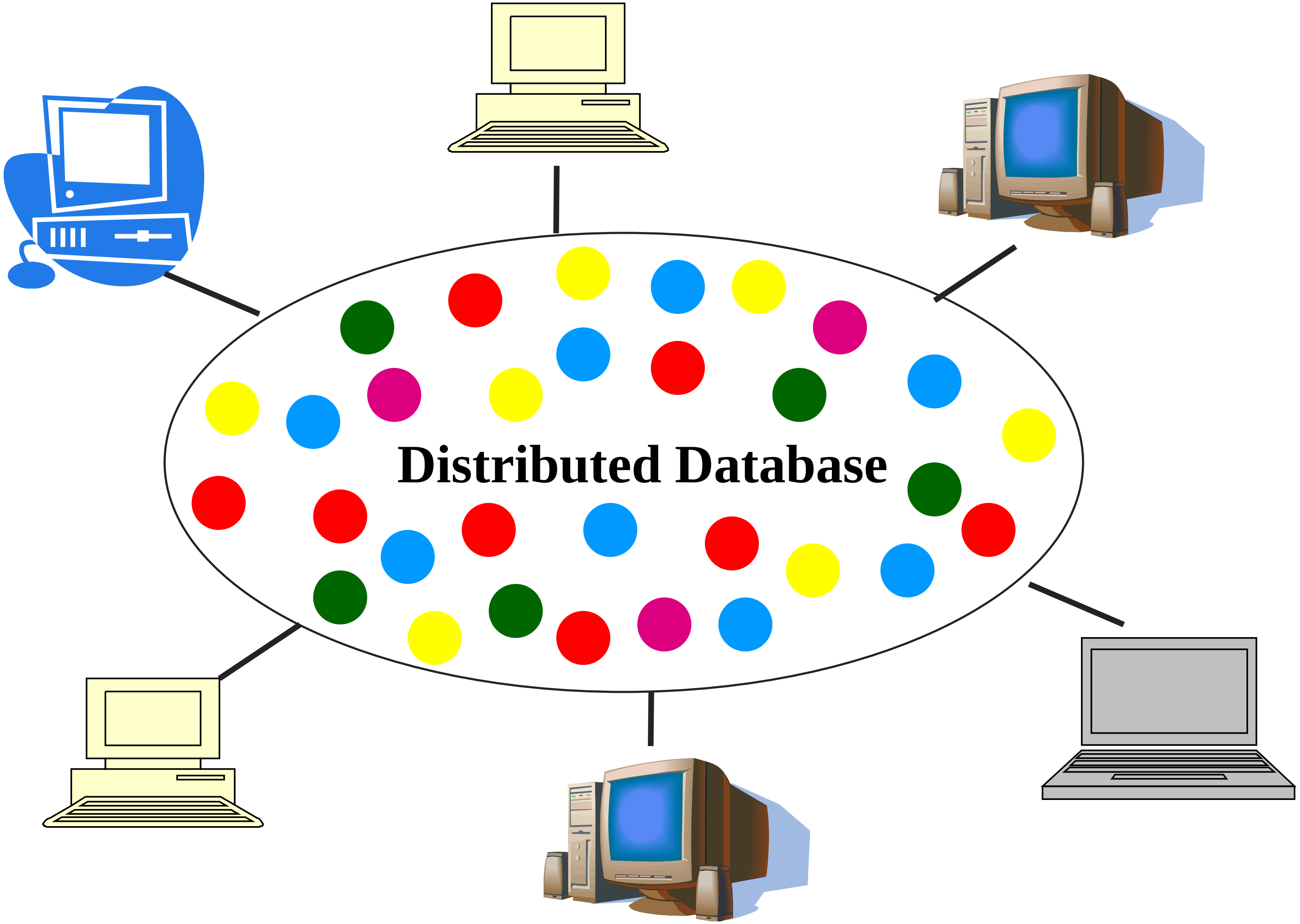
Promises of DDBSs

- Transparent management of distributed, partitioned/fragmented, and replicated data
- Improved reliability and availability through distributed transactions
- Improved performance
- Easier and more economical system expansion

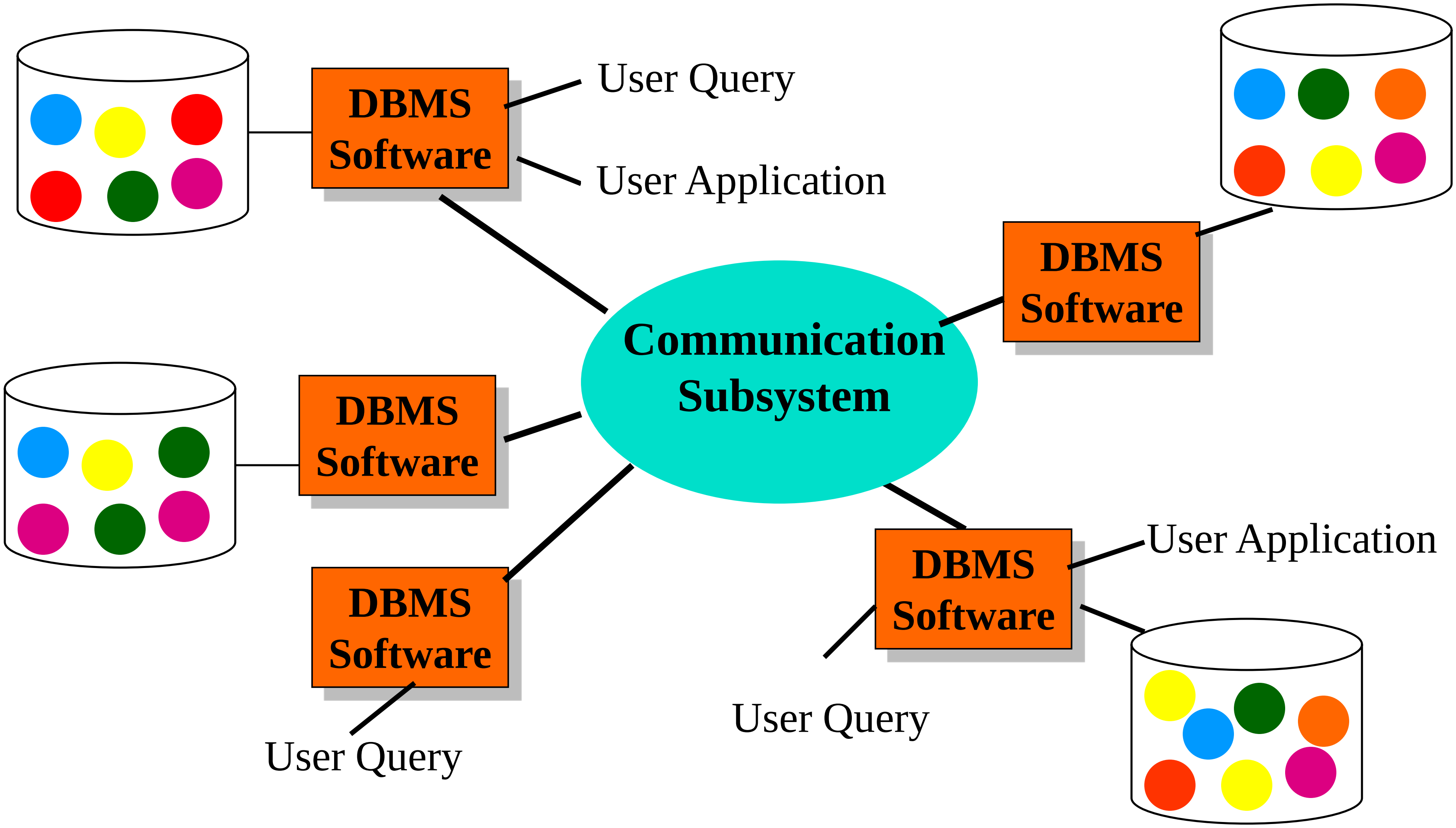
Transparency

- Transparency refers to separation of the high-level semantics of a system from low-level implementation details.
- Fundamental issue is to provide data independence in the distributed environment
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency
    - Horizontal fragmentation: selection
    - Vertical fragmentation: projection
    - hybrid

Distributed Database – User View



Distributed DBMS – Reality



Example Relations

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Programmer	48
E4	P2	Manager	18
E5	P2	Manager	24
E6	P4	Engineer	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

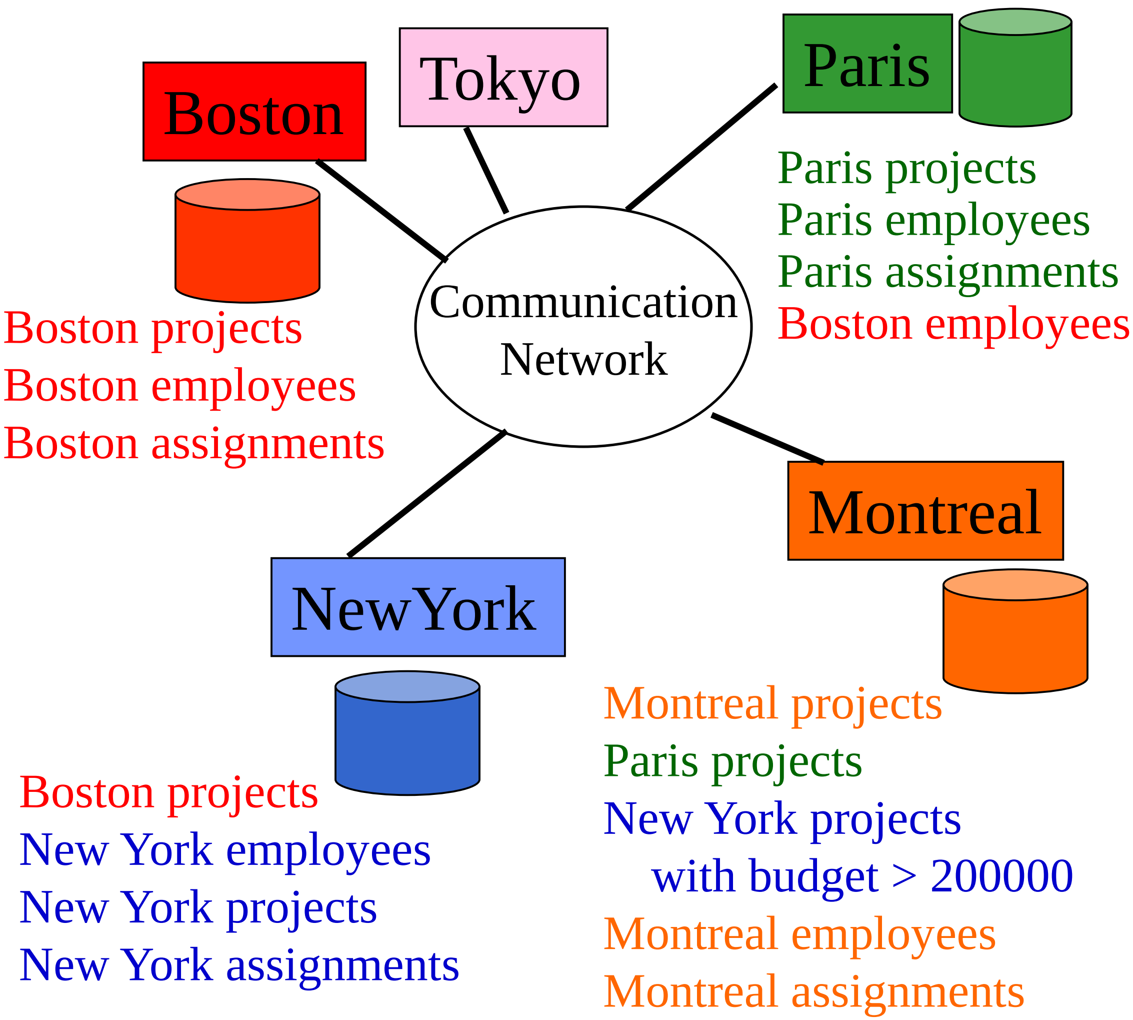
PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Transparent Access



Find the names and salaries of employees who are assigned to projects for over 12 weeks.

```
SELECT  ENAME, SAL
FROM    EMP, ASG, PAY
WHERE   DUR > 12
AND     EMP.ENO = ASG.ENO
AND     PAY.TITLE = EMP.TITLE
```

Improved Performance

- Parallelism in execution
  - inter-query parallelism
  - intra-query parallelism
- Since each site handles only a portion of a database, the contention for CPU and I/O resources is not that severe.
- Data localization reduces communication overheads.
  - Proximity of data to its points of use requires some support from fragmentation and replication

Parallelism Requirements

- Have as much of the data required by *each* application at the site where the application executes
  - Full replication

- How about updates?
  - Updates to replicated data requires implementation of distributed concurrency control and commit protocol

Improved Reliability

- Distributed DBMS can use replicated components to eliminate single point failure.
- Users can still access part of the distributed database with "proper care" even though some of the data is unreachable.
- Distributed transactions facilitate maintenance of consistent database state even when failures occur.

Easier System Expansion

- Ability to add new sites, data, and users over time without major restructuring.
- Huge centralized database systems (mainframes) are history (almost!).
- Cloud computing will lead to natural distributed processing.
- Some applications (such as large enterprise, social network data) are naturally distributed - centralized systems will just not work.

Disadvantages of DDBSs

- Complexity
  - DDBS problems are inherently more complex than centralized DBMS ones
- Cost
  - More hardware, software, and people costs
- Distribution of control
  - Problems of synchronization and coordination to maintain data consistency
- Security
  - Database security + network security
- Difficult to convert
  - No tools to convert centralized DBMSs to DDBSs

Complicating Factors

- Data may be replicated in a distributed environment, consequently the DDBS is responsible for
  - choosing one of the stored copies of the requested data for access in case of retrievals
  - making sure that the effect of an update is reflected on each and every copy of that data item
- If there is a site/link failure while an update is being executed, the DDBS must make sure that the effect will be reflected on the data residing at the failing or unreachable sites as soon as the system recovers from the failure.

Complicating Factors (cont.)

- Maintaining consistency of distributed/replicated data.
- Since each site cannot have instantaneous information on the actions currently carried out in other sites, the synchronization of transactions at multiple sites is harder than a centralized system.

Distributed vs. Centralized DBS

- Distribution leads to increased complexity in the system design and implementation.
- DDBS must be able to provide additional functions to those of a centralized DBMS.
  - To access remote sites and transmit queries and data among various sites via a communication network
  - To keep track of data distribution and replication in the DDBMS catalog
  - To devise execution strategies for queries and transactions that access data from more than one site
  - To decide on which copy of a replicated data item to access
  - To maintain the consistency of copies of a replicated data item
  - To maintain the global conceptual schema of the distributed database
  - To recover from individual site crashes and from new types of failures such as failure of a communication link

Distributed DBMS Issues

- Distributed Database Design
- Distributed Query Processing
- Distributed Concurrency Control (Distributed Deadlock Management)
- Reliability of Distributed Databases

Distributed Database Design

- The problem is how the database and the applications that run against it should be placed across the sites.
- Two fundamental design issues:
  - fragmentation (the separation of the database into partitions called fragments)
  - allocation (distribution)
    - The optimum distribution of fragments. The general problem is NP-hard.
    - Replicated & non-replicated allocation

Distributed Query Processing

- Query processing deals with designing algorithms that analyze queries and convert them into a series of data manipulation operations.
- The problem is how to decide on strategy for executing each query over the network in the most cost effective way. The objective is to optimize where the inherent parallelism is used to improve the performance of executing the transaction.
  - min {cost = data transmission + local processing + … }

Distributed Directory Management

- A directory contains information (such as descriptions and locations) about data items in the database.
- A directory may be global to the entire DDBS, or local to each site, distributed, multiple copies, etc.

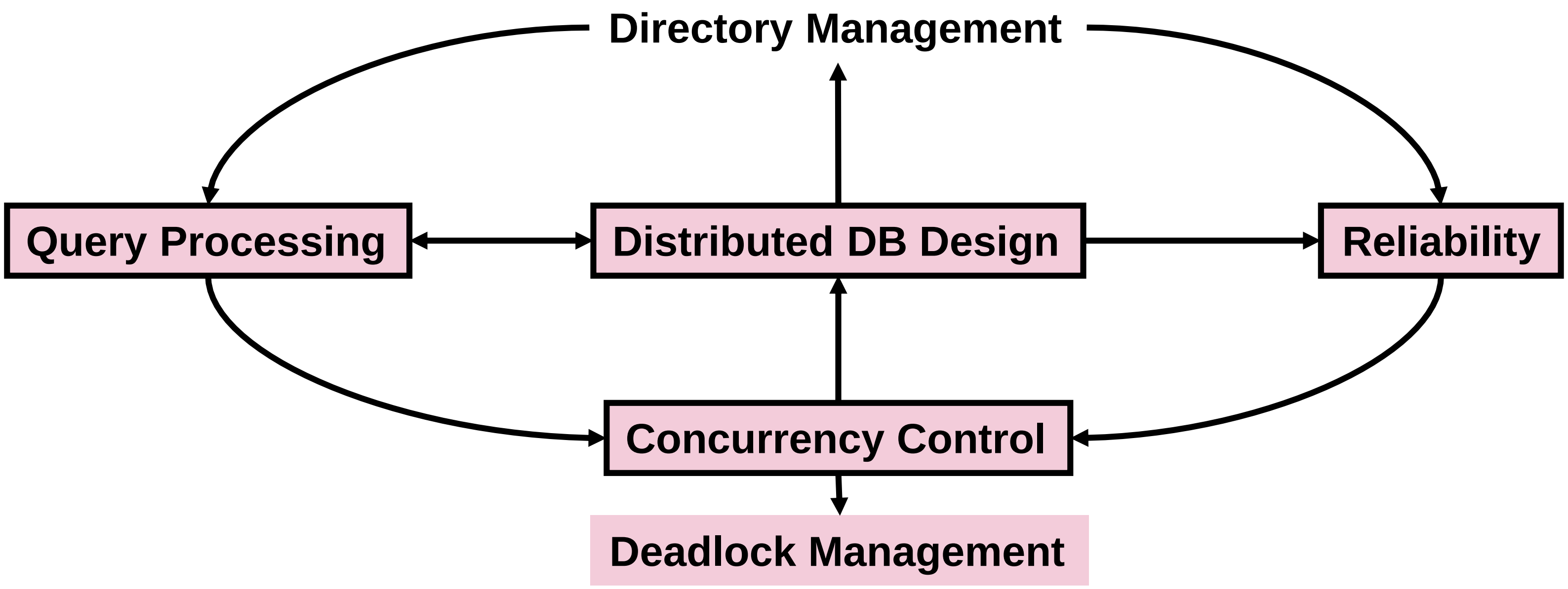
Distributed Concurrency Control

- Concurrency control involves
  - synchronization of accesses to the distributed database, such that the integrity of the database is maintained.
  - Consistency of multiple copies of the database (mutual consistency) and isolation of transactions' effects.
- Deadlock management

Reliability of Distributed DBMS

- It is important that mechanisms be provided to ensure the consistency of the database as well as to detect failures and recover from them.
- This may be extremely difficult in the case of network partitioning, where the sites are divided into two or more groups with no communication among them.

Relationship among Topics



Date's 12 Rules for Distributed RDBMSs

To the user, a distributed system should look exactly like a non-distributed system.

1. Local autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence

Rule 1: Local Autonomy

Sites should be autonomous to the maximum extent possible.

- Local data is locally owned and managed with local accountability
  - security consideration
  - integrity consideration
- Local operations remain purely local
- All operations at a given site are controlled by that site
  - No site X should depend on some other site Y for its successful functioning

Rule 1: Local Autonomy (cont.)

- In some situations, some slight loss of autonomy is inevitable.
  - fragmentation problem - Rule 5
  - replication problem - Rule 6
  - update of replicated relation - Rule 6
  - multiple-site integrity constraint problem - Rule 7
  - a problem of participation in a two-phase commit process - Rule 8

Rule 2: No Reliance on a Central Site

There must not be any reliance on a central "master" site for some central service, such as centralized query processing or centralized transaction management, such that the entire system is dependent on that central site.

- Reliance on a central site would be undesirable for at least the following two reasons:
  - that central site might be a bottleneck
  - the system would be vulnerable

Rule 2: No Reliance on a Central Site (cont.)

- In a distributed system, the following functions (among others) must therefore all be distributed:
  - Dictionary management
  - Query processing
  - Concurrency control
  - Recovery control

Rule 3: Continuous Operation

There should ideally never be any need for a planned entire system shut down.

- Incorporating a new site X into an existing distributed system D should not bring the entire system to a halt.
- Incorporating a new site X into an existing distributed system D should not require any changes to existing user programs or terminal activities.

Rule 3: Continuous Operation (cont.)

- Removing an existing site X from the distributed system should not cause any unnecessary interruptions in service.
- Within the distributed system, it should be possible to create and destroy fragments and replicas of fragments dynamically.
- It should be possible to upgrade the DBMS at any given component site to a newer release without taking the entire system down.

Rule 4: Location Independence (Transparency)

Users should not have to know where data is physically stored, but rather should be able to behave - at least from a logical standpoint - as if the data were all stored at their own local site.

- Simplify user programs and terminal activities.
- Allow data to migrate from site to site.
- It is easier to provide location independence for simple retrieval operations than it is for update operations.
- Distributed data naming scheme and corresponding support from the dictionary subsystem.

Rule 4: Location Independence (Transparency) (cont.)

- User naming scheme
  - User U has to have a valid logon ID at each of multiple sites to operate
  - User profile for each valid logon ID in the dictionary
  - Granting of access privileges at each component site

Rule 5: Fragmentation

- A distributed system supports data fragmentation if a given relation can be divided into pieces or "fragments" for physical storage purposes.
- Fragmentation is desirable for performance reason.
  - **Horizontal** and/or **Vertical** fragmentation

Employee		
EMP#	**DEPT#**	**SALARY**
E1	DX	45K
E2	DY	40K
E3	DZ	50K
E4	DY	63K
E5	DZ	40K

New York Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Physical storage
New York

London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

Physical storage
London

Rule 5: Fragmentation Independence (Transparency)

Users should be able to behave (at least from a logical standpoint) as if the data were in fact not fragmented at all.

- A system that supports data fragmentation should also support fragmentation independence (also known as fragmentation transparency).
- Fragmentation independence (like location independence) is desirable because it simplifies user programs and terminal activities.

Rule 5: Fragmentation Independence (Transparency) (cont.)

- Fragmentation independence implies that users should normally be presented with a view of the data in which the fragments are logically combined together by means of suitable joins and unions.

Rule 6: Replication Independence (Transparency)

- A distributed system supports data replication if a given relation (more generally, a given fragment of a relation) can be represented at the physical level by many distinct stored copies or replicas, at many distinct sites.

- Replication is desirable for at least two reasons
  - Performance
  - Availability

Employee

EMP#	DEPT#	SALARY
E1	DX	45K
E2	DY	40K
E3	DZ	50K
E4	DY	63K
E5	DZ	40K

NewYork Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Replica of London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

Replica of New York Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Rule 6: Replication Independence (Transparency) (cont.)

User should be able to behave as if the data were in fact not replicated at all.

- User should be able to behave as if the data were in fact not replicated at all.
- Replication, like fragmentation, should be "transparent to the user".
- Update propagation problem
- Replication independence (like location and fragmentation independence) is desirable because it simplifies user programs and terminal activities.

Rule 7: Distributed Query Processing

It is crucially important for distributed database systems to choose a good strategy for distributed query processing.

- Query processing in a distributed system involves
  - local CPU and I/O activities at several distinct sites
  - some amount of data communication among those sites
  - energy, cost, …
- Amount of data communication is a major performance factor.
- Query compilation ahead of time

Rule 8: Distributed Transaction Management

Two major aspects of transaction management, concurrency control and recovery control, require extended treatment in a distributed environment.

- In a distributed system, a single transaction can involve the execution of code at multiple sites and can thus involve updates at multiple sites.
- Each transaction is therefore said to consist of multiple "agents," where an agent is the process performed on behalf of a given transaction at a given site.
- Deadlock problem may be incurred.

Rule 9: Hardware Independence (Transparency)

User should be presented with a "single-system image'' regardless of any particular hardware platform.

- It is desirable to be able to run the same DBMS on different hardware systems.
- It is desirable to have those different hardware systems all participate as equal partners (where appropriate) in a distributed system.
- It is assumed that the same DBMS is running on all those different hardware systems.

Rule 10: Operating System Independence

It is obviously desirable, not only to be able to run the same DBMS on different hardware systems, but also to be able to run it on different operating systems - even different operating systems on the same hardware.

- From a commercial point of view, the most important operating system environments, and hence the ones that (at a minimum) the DBMS should support, are probably MVS/XA, MVS/ESA, VM/CMS, VAX/VMS, UNIX (various flavors), OS/2, MS/DOS, Windows,…

Rule 11: Network Independence

It is obviously desirable to be able to support a variety of disparate communication networks.

- From the viewpoint of the distributed DBMS, the network is merely the provider of a reliable message transmission service.
- “Reliable" here means that, if the network accepts a message from site X for delivery to site Y, then it will eventually deliver that message to site Y.
- Messages will not be delivered more than once, and will be delivered in the order sent.
- The network should also be responsible for site authentication.
- Ideally the system should support both local and wide area networks.
- A distributed system should support a variety of different network architectures.

Rule 12: DBMS Independence

Ideally a distributed system should provide DBMS independence (or transparency).