2. Towards Future DBS Architectures

Theo Härder
www.haerder.de

Distribution and parallelism

Classical requirements: performance, availability, extensibility, …

Processing concepts

Future requirements

New DBMS data types

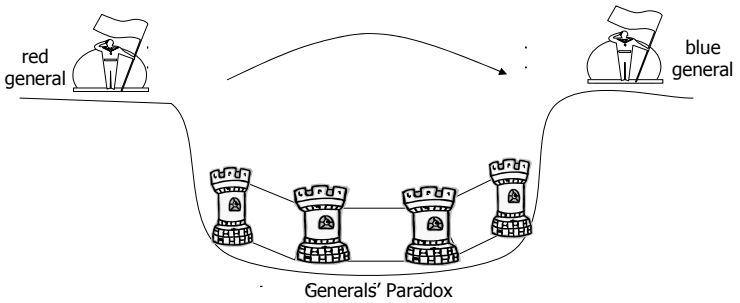
Architectural issues

Overflow

Main references:
Jim Gray: The Next Database Revolution, SIGMOD Conference 2004: 1-4
Erhard Rahm: Mehrrechner-Datenbanksysteme, Addison-Wesley, 1994, Kap. 16

Distribution and Parallelism

- **Distributed system**
  - consists of autonomous subsystems which frequently are allocated far away from each other. They collaborate in a coordinated way to fulfill a common task
- **Analogy: "Coordinated Attack" Problem**



➔ The characteristic core problem for distributed systems is the **lack of global (centralized) knowledge**

Distribution and Parallelism

- **Parallel system**

  consists of a (large) number of uniform subsystems (components) which are allocated locally to each other and which exhibit only a low degree of autonomy.
  A characteristic feature is the close and highly parallel processing of single user tasks (intra-transaction parallelism)
  - decomposition of the task for multiple processors (not multiple cores of a single processor)
  - message exchange for coordination and result composition
  - often use of shared resources (data)

- **Distinguish**
  - parallel processing → ~ 10^4 - 10^5 computing units
  - distributed processing → uncertainty in case of global decisions

Distribution and Parallelism

- **Typical performance criteria for sequential processing:**
  - disk 10 MB/s
  - searching (table scan) 1 MB/s
  - sorting 0.1 MB/s
  - join ?

  ➔ processing time for a 10 TB database ?

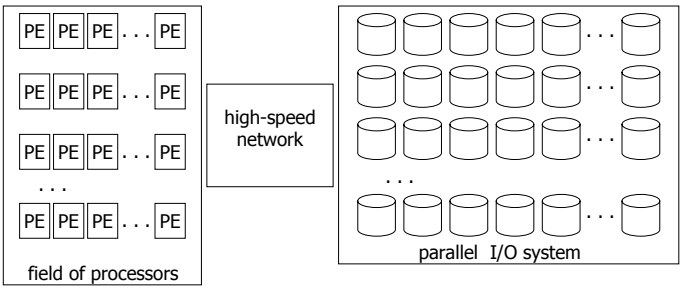
- **Intra-transaction parallelism:**
  use of parallelism within transactions
  - operations on large tables: scan, join computation, sorting, creation of indexes, etc.
  - full-text search in digital libraries
  - multimedia applications
  - complex logic programs, • • •

  ➔ **short response times for data- and/or computation-intensive DB requests**

- **Inter-transaction parallelism:**
  - high transaction rates for OLTP
  - linear growth of throughput

Parallel Database Systems

- **Premise for parallel processing**
  - decomposition of a large problem into smaller subtasks which can be solved in parallel
- **Architecture**
  - parallel computer with a high number of (micro) processors
  - local distribution
  - scalable high-speed network **and** I/O parallelism



- **Special type of MC-DBMS with main properties**
  - high performance, availability
  - scalability, cost effectiveness

MC-DBMS: Requirements

- **Goal**

  use of several computers/DBMS for coordinated processing of DB operations

- **Requirements**
  - high performance (high transaction rates with short response times)
  - high availability / fault tolerance
  - modular extensibility (vertical and horizontal growth)
  - distribution transparency for DB users (for application programs resp. end users)
  - coordinated access to heterogeneous databases
  - support of geographically distributed databases (preservation of high node autonomy)
  - high cost effectiveness (use of powerful microprocessors, but also mainframe clusters, etc.)
  - simple manageability / administration

Requirements: High Performance

- **High transaction rates (throughput)**
  - >> 100,000 tpmC (2006: 3,210,540 tpmC of type 'order processing')
- **Short response times**
  - acceptability of interactive mode
  - despite higher throughput / communication delays
  - parallelization of complex queries
- **Permanently increasing performance needs**
  - growing number of users / clients
  - new applications / transaction types
  - permanent growth of databases
  - processing of more complex tasks and integrity constraints
  - use of higher programming languages
  - more comfortable user interfaces
  - more and more multi-step dialogues

➔ use of MC-DBMS required

Examples of High Performance Needs

1. Banking applications/ reservation systems
   debit/credit TAs or seat reservations should be processed achieving throughput
   - of several 10,000 TPS and
   - with response time < 2 sec
2. Dial switching
   a user profile has to be read from the DB and a accounting record has to be written per telephone call
   in case of high-traffic situations:
   - > 50,000 of such transactions per second
   - < 0.2 sec response time
3. Decision support/ data warehousing
   On a 5 TB DB, complex ad-hoc queries are to be processed, which require a complete scan of the DB in the worst case. To be achieved:
   - throughput of 10 TPS and
   - a response time < 30 sec

… E-Commerce / digital libraries / geo-information systems, ...

Requirements: High Availability

- **Goal: continuous operation**
  - at least tolerance against single failures
  - system must not (completely) fail, because restart is too expensive (network)
  - dynamic reorganization, extensibility, etc. of all data structures
  - dynamic maintenance: replacement of HW, SW versions, etc. during run time
- **preconditions:**
  - redundant system components (fault tolerance)
    - HW and SW components
    - data (log, mirrored disks, replicated DB)
  - automatic error detection and correction
  - system reconfiguration under continuous operation
- **MTBF (mean-time between failures):**

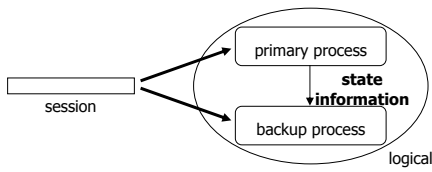
  conventional systems: > 10 days (several months)

  fault-tolerant systems: > 10 years

Requirements: High Availability (2)

- Availability = MTBF / (MTBF + MTTR) = 99.999… %
  - MTTR (mean-time to repair)
  - why must the total system not fail?

    After failure detection:
    - operator interaction / dump?
    - warm start of OS
    - restart of DBS
    - restart of communication system
    - opening of files and sessions (>10^4 users)
- Helpful concepts for availability
  - process pairs and transaction concept to mask failures
  - active primary process and passive backup process
  - update of the backup by periodic checkpoint messages (frequent checkpointing to enable forward recovery)
  - shadow processes: fast restart after rollback with persistent input msg



Analysis of Failure Causes

- Classification of failure causes[1]:

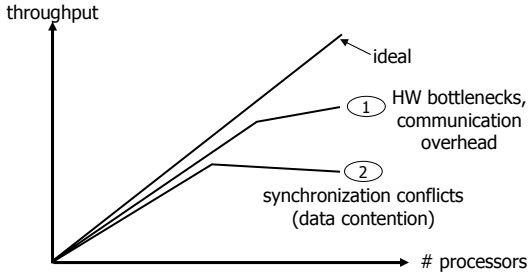
cause	1985	1989	today
software	33 %	62 %	>> %
hardware	29 %	7 %	<< %
maintenance	19 %	5 %	<< %
operations	9 %	15 %	>> %
environment	6 %	6 %	<< %
system MTBF	8 years	21 years	? years

- „Under-reporting" for operating, application SW
- SW errors are dominating cause of failure
  - fast growing amount of application and system SW
  - increasing SW complexity
- Strong improvement for HW and maintenance
  - increasing density of integration (VLSI, ULSI), chunkier disks, communication via fiber
  - HW redundancy masks majority of all failures
  - MTBF for disks improved from 8K to >100K hours!
- Larger degree of automation required for operating

1 J. Gary: A Census of Tandem System Availability Between 1985 and 1990: in IEEE Trans. on Reliability 39 (4), 1990, 409-418. Recent figures are not available. The principal statements are still valid.

Requirements: Extensibility

- **Goal: modular (horizontal) growth**
  - additional computers, disks, etc.
  - linear increase of throughput (preserving short response times)
  - complex queries: response time reduction through parallelism proportional to the number of processors



Requirements: Extensibility (2)

- **TPC benchmark workloads represent ideal case for scalability**
  - DB grows proportional to number of processors (throughput)
  - ideal partitioning of DB and workload possible (➔ minimal communication overhead)
- **"Real" workloads**
  - limited means of partitioning (growing msg frequency per transaction with growing #processors)
  - difficult load balancing
  - potentially increasing lock conflicts
  - dynamic load control needed (DBAW reminder!)
    - **use of conflict rate in case of lock protocols**[1]
    - conflict rate = # locks granted / # locks held by non-blocked transactions
    - **critical value:** ca. 1.3 (experimentally determined)
    - admission of new TAs only, if critical value is not reached
    - otherwise: abort of TAs

1 G. Weikum et al.: The Comfort Automatic Tuning Project, in: Information Systems 19:5, 1994, 381-432

Relative Reference Matrix (Practical Example)

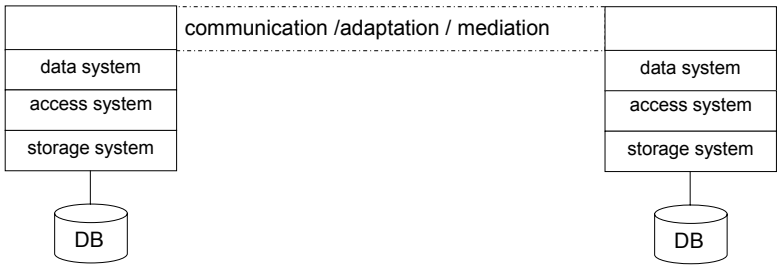
ca. 17 500 transactions, 1 million page references to ca. 66 000 different pages

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	Total
TT1	9.1	3.5	3.3		5.0	0.9	0.4	0.1				0.0		22.3
TT2	7.5	6.9	0.4	2.6	0.0	0.5	0.8	1.0	0.3	0.2	0.0			20.3
TT3	6.4	1.3	2.8	0.0	2.6	0.2	0.7	0.1	1.1	0.4		0.0	0.0	15.6
TT4	0.0	3.4	0.3	6.8			0.6	0.4			0.0			11.6
TT5	3.1	4.1	0.4		0.0		0.5	0.0						8.2
TT6	2.4	2.5	0.6		0.7		0.9	0.3						7.4
TT7	1.3		2.6			2.3	0.1							6.2
TT8	0.3	2.3	0.2		0.0		0.1							2.9
TT9	0.0	1.4	0.0					1.1						2.6
TT10	0.3	0.1	0.3			1.0	0.1					0.0		1.8
TT11		0.9						0.2						1.1
TT12		0.1												0.1
Total	30.3	26.6	11.0	9.4	8.3	4.9	4.1	3.3	1.4	0.6	0.0	0.0	0.0	100.0
partition size (%)	31.3	6.3	8.3	17.8	1.0	20.8	2.6	7.3	2.6	1.3	0.8	0.0	0.0	100.0
% refe-renced	11.1	16.6	8.0	2.5	18.1	1.5	9.5	4.4	5.2	2.7	0.2	13.5	5.0	6.9

➔ This is not uniform!

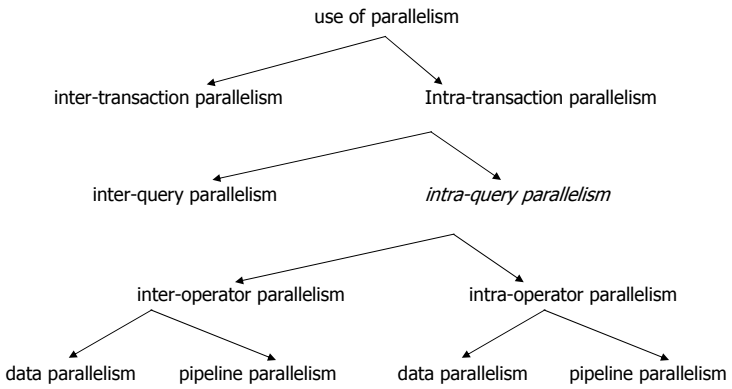
Layer Model for Distributed/Parallel DBMS Processing

- Each node
  - runs a copy of the DBMS
    - typically, a process is unit of scheduling, addressing, and protection
    - single/multi process
    - singe/multi tasking within a process
  - various models of data access
    - shared nothing, shared disk, shared everything
    - SN and SD architectures are discussed later

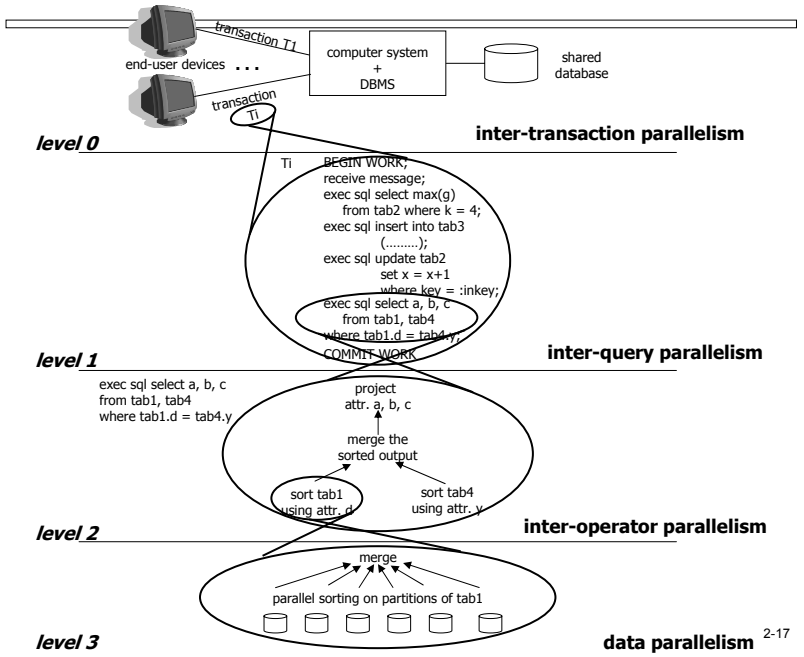


Kinds of Parallelism

- Classification aspects
  - granularity of parallel processing steps (transaction, query, operator)
  - data parallelism vs. function parallelism (pipeline parallelism)
  - processing parallelism vs. I/O parallelism
- Classification

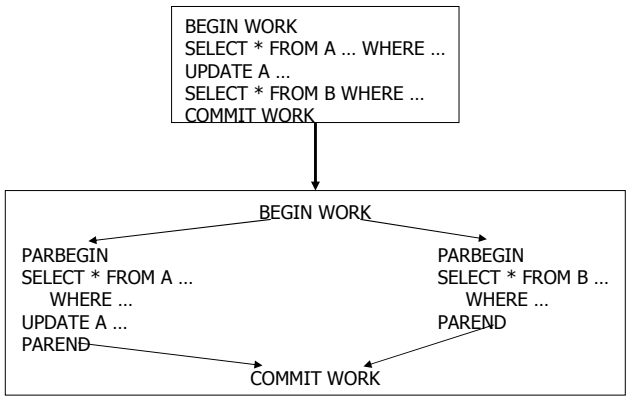


Simultaneous Use of Multiple Kinds of Parallelism



Inter-Query Parallelism

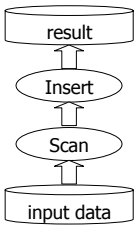
- **Parallel processing of independent DB operations (queries)** of a transaction program
  - programmer must specify potential parallelism



  - only limited degree of parallelism possible

Pipeline Parallelism vs. Data Parallelism

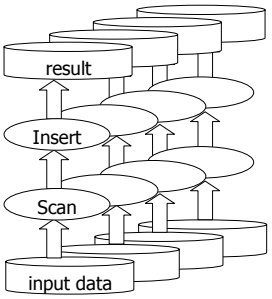
pipeline parallelism



- **Pipeline parallelism**
  - data flow principle for data exchange between operators /sub-operators
  - early transfer of records in intermediate results
  - in particular, use for inter-operator parallelism
  - pipeline break for operators requiring complete input for the result generation: sorting, duplicate elimination, grouping (GROUP BY), aggregate functions, etc.
  - pipelines often very short (≤ 10 operators)

Pipeline Parallelism vs. Data Parallelism (2)

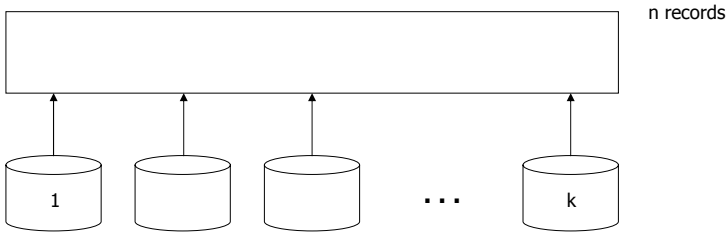
Data parallelism and pipeline parallelism



- **Data parallelism**
  - based on broad (horizontal) data distribution (declustering)
  - parallel processing of sub-operations on disjoint data sets
  - degree of parallelism can be increased with growing amount of data

I/O Parallelism

- **Precondition: declustering of files across several disks**
- **Intra-I/O parallelism (access parallelism, for example, RAID)**



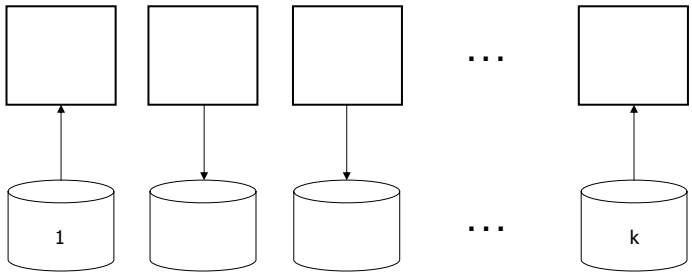
Parallel execution of an I/O task (➔ data parallelism)

sequential: $t = n * (t_{mav} + t_r + t_{tr})$

parallel: $t = t_{mav} + t_r + n/k * t_{tr}$

I/O Parallelism (2)

- **Inter-I/O parallelism (task parallelism)**

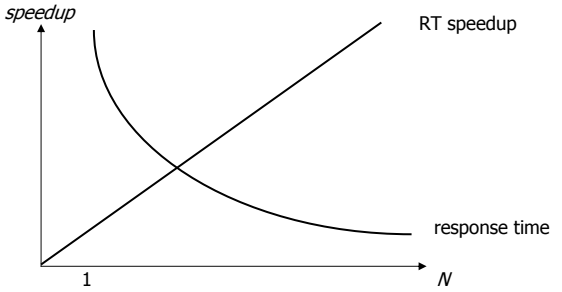


Parallel execution of independent I/O tasks of separate disks (➔ inter-transaction parallelism, inter-query parallelism)

Performance Measures for Parallel Processing: Speedup

- **Assumption: constant DB size**
- **Response time (RT) speedup** (batch speedup)
  measures reduction of response time for complex operations using parallel processing

$$\text{RT speedup (N)} = \frac{\text{RT using 1 computing unit}}{\text{RT using N computing units}}$$



- **Goal:**
  linear reduction of response time with growing number of computing units by use of intra-transaction parallelism

Performance Measures for Parallel Processing: Speedup (2)

- **Amdahl's Law**
  - Speedup is limited by non-optimized (sequential) components of response time

$$\text{RT speedup} = \frac{1}{(1 - F_{opt}) + \frac{F_{opt}}{S_{opt}}}$$

F_{opt} = share of optimized RT components ($0 \le F_{opt} \le 1$)

S_{opt} = Speedup for optimized RT share

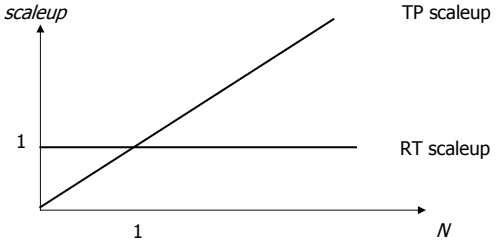
  - example: 5% sequential share $\rightarrow F_{opt} = 0.95$, $S_{opt} = 10$

RT speedup = $1/(0.05 + 0.95/10) = 10/(0.5 + 0.95) \sim 6.89$

Performance Measures for Parallel Processing: Scaleup

- **Assumption: DB size grows linearly with # of computing units**
- **Throughput scaleup (TP)**
  (OLTP scaleup) measures relative throughput growth
  (for given response time restrictions)

$$\text{TP scaleup (N)} = \frac{\text{transaction rate using N computing units}}{\text{transaction rate using 1 computing unit}}$$



➔ goal: linear growth by using inter-transaction parallelism

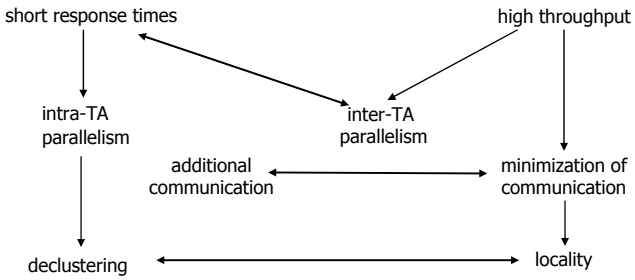
Performance Measures for Parallel Processing: Scaleup (2)

- **Response time scaleup** (batch scaleup)
  measures change of response time for complex operations
  on growing data sets

$$\text{RT scaleup (N)} = \frac{\text{RT using N computing units}}{\text{RT using 1 computing unit}}$$

➔ goal: constant response time despite growing DB
by intra-transaction parallelism

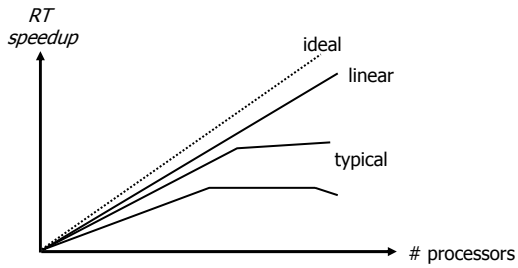
Performance Tradeoffs



- **Desirable: simultaneous processing of short OLTP transactions and complex ad-hoc queries on the same data set**
  - high throughput for short TA
  - short response times for ad-hoc queries
- **Problems:**
  - determination of a suitable data distribution for both workload types
  - synchronization problems for long-running queries
  - high resource needs for complex queries

Limits of Scalability

- **Characterization of response time improvement:**
  The real use of parallelism is by far not "ideal", in general!

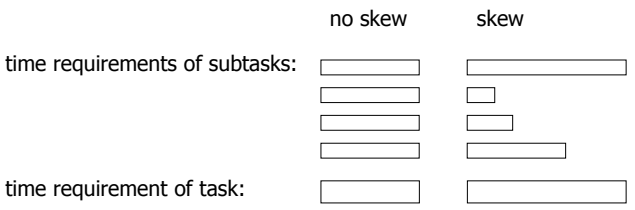


Is super-linear RT speedup possible?

Problems in typical situations:
- synchronization vs. load control, communication overhead
- resource bottlenecks and resource thrashing
- deadlocks and aborts, repetition of work

Limits of Scalability (2)

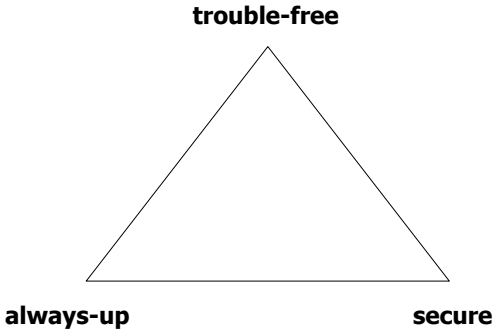
- maximal inherent parallelism is limited
- startup- and termination overhead
- interferences of physical and logical resources
- **variance (skew)** in the execution times of subtasks



➔ **Amdahl's law** limits response time speedup!

New Future Requirements

- Additional properties needed (Jim Gray)



Approaches to Adaptivity of System Behavior

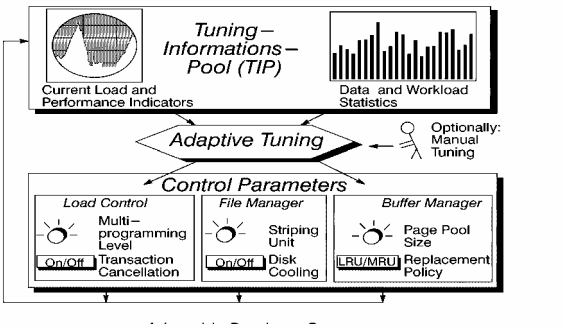
- Choose default values for tuning: rules of thumb
  - OK for workload-independent parameters: page size, striping unit, minimal buffer size
  - insufficient for load balancing aspects: MPL limit, etc.
- Hardware is cheap: **the KIWI principle**
  - OK if applied with care
  - however, it often implies a waste of resources

> **An engineer is someone who can do for a dime what any fool can do for a dollar.**

- Autonomic computing: **online feedback control loop**
  - OK when it converges under stationary workload
  - susceptible to instability

Automate some Tasks of the DBA

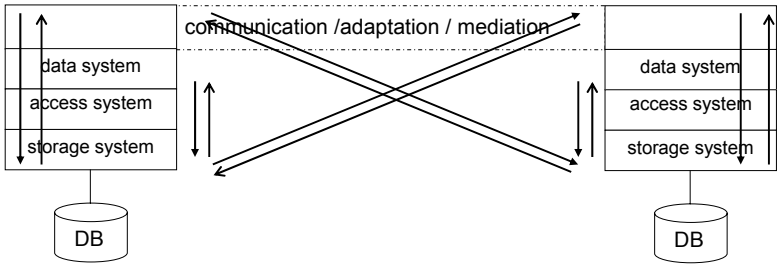
- Manual performance tuning: observe – predict – react



- Process the loop automatically
  - monitor – analyze – plan – react
  - prediction needs quantitative models!
  - additional information flow within / between layers

What does Adaptivity Mean for the Architectural Model?

- Architectural model is suitable for
  - for relational data types
  - XML data types
  - ...?
- Additional information flow across nodes for
  - enhanced adaptivity
  - other self-* properties



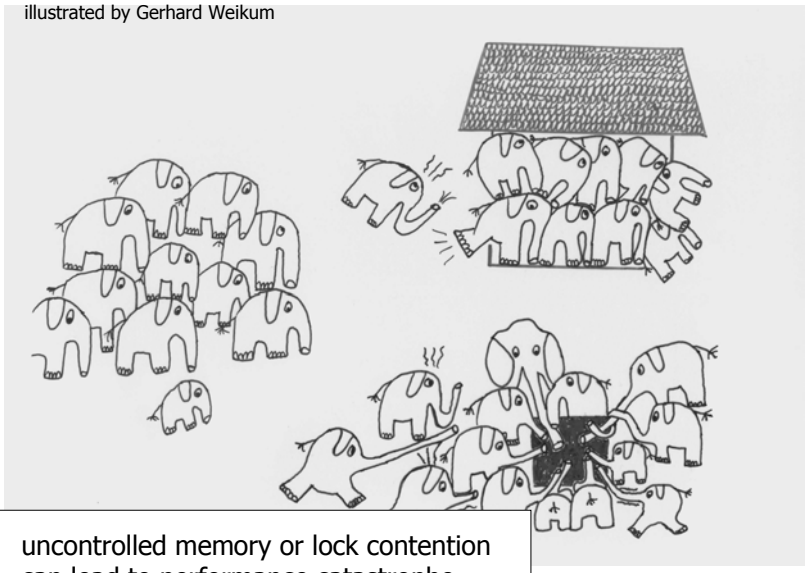
New Tasks to be Solved in the Kernel/Across Nodes

	Transaction Mgmt	Query Processing		
C	Consistency Control	Compilation Optimization	L5	Data System
	Transaction Services	Path Processing Algorithms	L4	
I				Access System
	Concurrency Control	Document Representation Labeling, Indexing	L3	
A	Logging / Recovery	Buffer Mgmt	L2	Storage System
D		Propagation Control File Services	L1	

Examples:
buffer management (L2 – L5)
index selection (L3 – L5)
load balancing (L3 – L5)

Multi-User Processing – Various Forms of Adaptivity Required

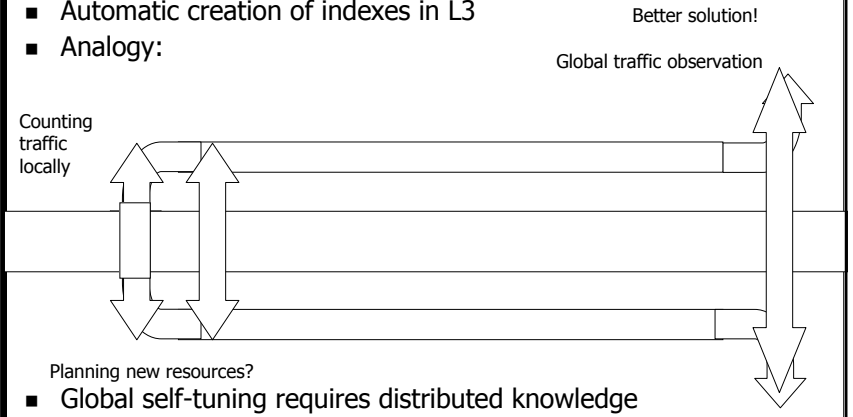
illustrated by Gerhard Weikum



uncontrolled memory or lock contention can lead to performance catastrophe

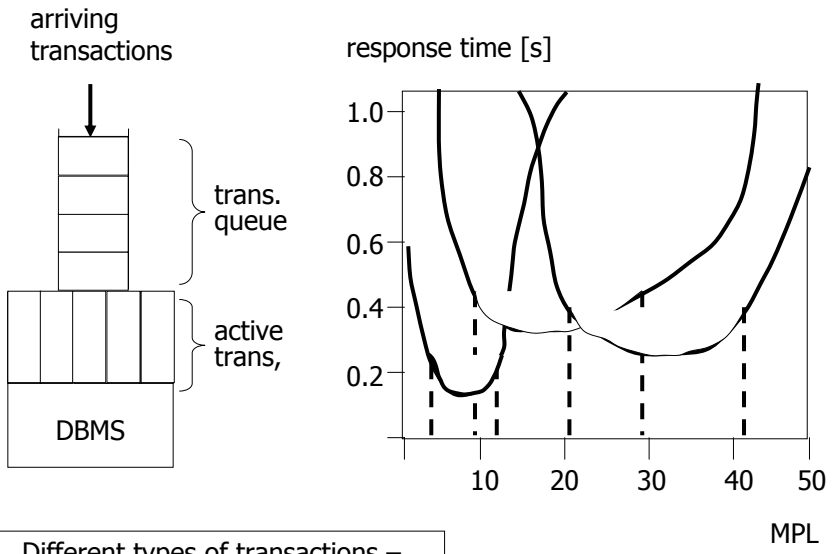
Local Self-Tuning – Index Selection

- Automatic creation of indexes in L3
- Analogy:



- Global self-tuning requires distributed knowledge
  - workload statistics collected in L5
  - use of path processing algorithms in L4
  - availability of alternative indexes in L3

Load Control for Locking – How difficult can this be?



Different types of transactions – conflicting performance goals

Adaptivity at the Interface: Google vs SQL

- **Google**
  - text oriented
  - IR and ranking-based search
  - link-based ranking
  - documents and links
  - atom search
  - heterogeneous, distributed collection
  - harvesting
  - freshness
  - embarrassingly parallel
  - sub-second response time
  - monolithic application (does not do anything else)
- **Database**
  - categorical typed information
  - query-oriented search
  - schema driven
  - attributes, rows, tables, and relations
  - molecular search
  - homogeneous, centralized collection
  - transactional updates
  - consistency
  - monolithic software
  - RT is query dependent
  - multiple functionality (user-defined functions, support for Business Intelligence, etc.)

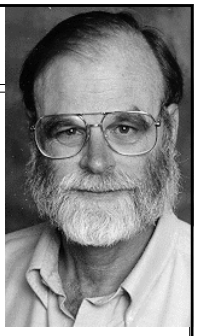
Important Aspects of Dependability

- *Dependability* of a computing system is the ability to deliver a service that can be justifiably trusted.

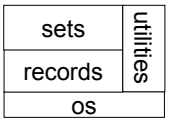
⇨ **dependability even more complex and fuzzy!**

- Dependability attributes:
  - *availability*: readiness for correct service
  - *reliability*: continuity of correct service
  - *safety*: absence of catastrophic consequences for the user(s) or the environment
  - *confidentiality*: absence of unauthorized disclosure of information
  - *integrity*: absence of improper system state alterations
  - *maintainability*: ability to undergo repairs and modifications

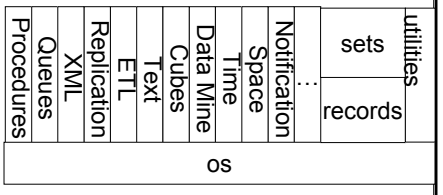
Classic DBMS Architecture



- The classic architectural model of DBMS
  - Worked, but applications wanted to query other data types
  - added:



+Text, Time, Space
+ Cubes, Data mining
+ XML, XQuery
+ Programming Languages
+ Triggers and queues
+ Replication, Pub/sub
+ Extract-Transform-Load
+ Many more extensions coming

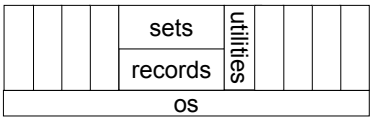


⇨ **A Mess?**

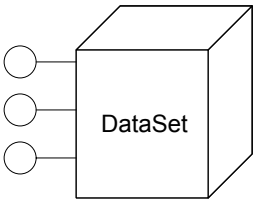
Thanks to Jim Gray: The section on "New DBMS data types" was adopted from his talk on The Next Database Revolution, SIGMOD Conference 2004

DB Systems evolved to be **containers for information services** develop, deploy, and execution environment

- Classic ++
  - + Programming Languages
  - + Triggers and queues
  - + Replication, Pub/sub
  - + Extract-Transform-Load
  - + Text, Time, Space
  - + Cubes, Data mining
  - + XML, XQuery
  - + Many more extensions coming



- **Vision:** DBMS is an ecosystem
  OO is the key structuring strategy:
  - Everything is a class
  - Database is a complex object
  - Core object is DataSet
  - Classes publish/consume them
  - Depends on strong Object Model

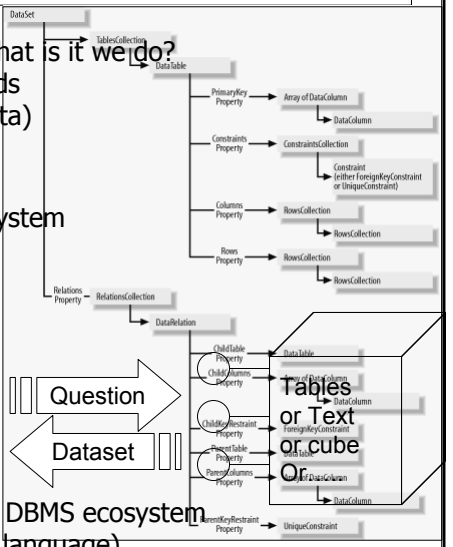


- Many of the DBMS concepts are now mainstream

Ask not "How to add objects to databases?", *Ask* "What kind of object is a database?"

- Q: Given an object model, what is it we do?
  A: DataSet class and methods
  (nested relation with metadata)

- This is the basis for the ecosystem
  Distributed DB
  Extensible DB
  Interoperable DB
  ...



- This was implicit in ODBC,
  but is now explicit within the DBMS ecosystem
  Input: Command (any language)
  Output: Dataset

Code and Data: Separated at Birth

COBOL

- ~~IDENTIFICATION: document~~

AUTHOR, PROGRAM-ID, INSTALLATION,
SOURCE-COMPUTER, OBJECT-COMPUTER,
SPECIAL-NAMES, FILE-CONTROL, I-O-CONTROL,
DATE-WRITTEN, DATE-COMPILED,
SECURITY.

- ~~ENVIRONMENT: OS~~

CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.

- ~~DATA: Files/Records~~

FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
REPORT SECTION.
SCREEN SECTION.

- ~~PROCEDURE: code~~

"them"

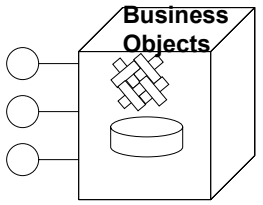
"us"

CODASYL - DBTG
COnference on DAta SYstems Languages
Data Base Task Group
Defined DDL for a network data model
Set-Relationship semantics
Cursor Verbs
Isolated from procedures.
No encapsulation

The Object-Relational World

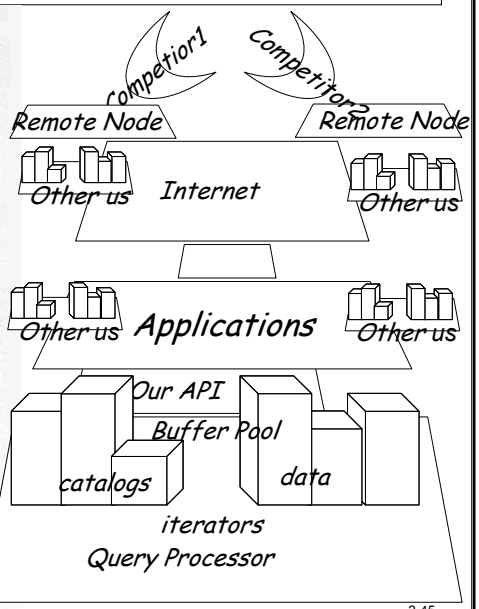
Algorithms + Data Structures = Programs (N. Wirth)

- Marriage of programming languages and DBMSs
  - stored procedures evolve to "real" languages Java, C#,.. with real object models.
  - data encapsulated: a class with methods
  - classes may be persistent
  - tables are enumerable & indexable record sets with foreign keys
  - records are vectors of objects
  - opaque or transparent types
  - set operators on transparent classes
  - transactions:
    - Preserve invariants
    - A composition strategy
    - An exception strategy



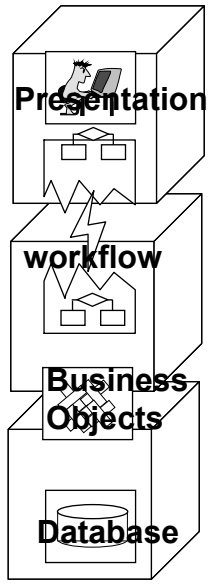
- **Ends Inside-DB Outside-DB dichotomy**

What's Outside?



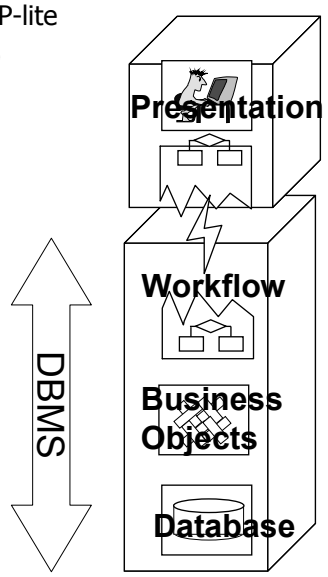
Classic: What's Outside?

- **Three-tier computing**
  - clients do presentation, gather input
  - do some workflow (script)
  - send high-level requests to ORB (Object Request Broker)
  - ORB dispatches workflows and business objects -- proxies for client, orchestrate flows & queues
  - server-side workflow invokes distributed business objects to execute task
  - business object read/write database



DBMS is Web Service!

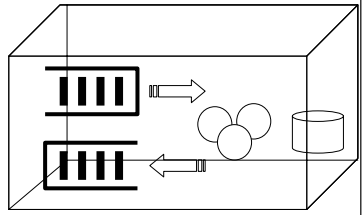
- Client/server is back; the revenge of TP-lite
- **Web servers** and runtimes (Apache, IIS, J2EE, .NET) displaced TP monitors & ORBS
  - Give persistent objects
  - Holistic programming model & environment
- **Web services** (soap, wsdl, xml) are displacing current brokers
- DBMS listening to Port 80 publishing WSDL, DISCO, Servicing SOAP calls.
  DBMS is a web service
- Basis for distributed systems.
- A **consequence of OR DBMS**



Queues & Workflows

- Properties
  - apps are loosely connected via queued messages
  - queues are databases.
  - basis for workflow
  - queues: the first class to add to an OR DBMS
  - queues fire triggers. active databases
  - synergy with DBMS
    security, naming, persistence, types, query,…

Workflow:
Script
Execute
Administer &
Expedite
all built on queues



Text, Temporal, and Spatial Data Access

- **Q**: What comes after queues?

  A: Basic types: text, time, space,…
  - great application of OR technology
  - **key idea**: **table-valued functions == indexes**
    - an index is a table, organized differently
    - query executor uses index to map: key → set (aka sequence of rows)
  - table-valued function can do this map, optimizer can use it.
  - +extras: cost function, cardinality,…

- **Big deal approximate answers – rank and support**

```
select Title, Abstract, Rank
from Books join
   FreeTextTable(Title, Abstract,
            'XML semistructured') T
on  BookID = T.Key
```

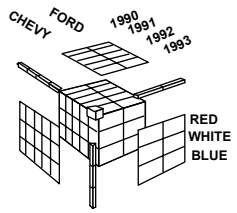
```
select store, holiday, sum(sales)
from Sales join
     HolidayDates(2004) T
on Sales.day = T.day
group by store, holiday
```

```
select galaxy, distance
from GetNearbyObjEQ(22,37)
```


Cubes

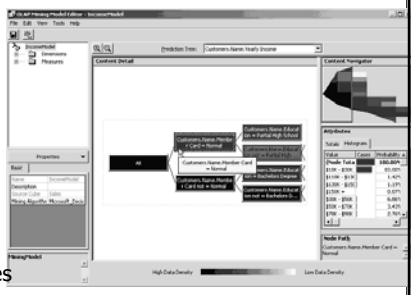
- Data cubes now standard
  - MDX is very powerful
    (**M**ulti-**D**imensional e**X**pressions)
  - Dimension, Measure, Operator concepts highly evolved beyond snowflake schema
  - Cube stores cohabit with row stores
    ROLAP + MOLAP + (∀x xOLAP)
    (relational +multidimensional online analytic processing)
  - Very sophisticated algorithms
  - A big part of the ecosystem

SELECT <axis_spec>
FROM <cube_spec>
WHERE <slicer_spec>



Data Mining and Machine Learning

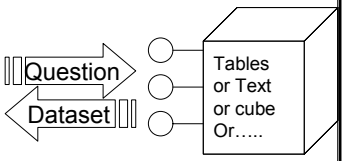
- Tasks: classification, association, prediction
- Tools: decision trees, Bayes, apriori, clustering, regression, neural net,…
- Now unified with DBs
  - create table T (x,y,z,u,v,w)
    learn "x,y,z" from "u,v,w"
    using <algorithm>
  - train T with data
  - then can ask:
    - probability x,y,z,u,v,w
    - what are the u,v,w probabilities given x,y,z
  - example: learn height from age.



- Anyone with a data mining algorithm has full access to the DBMS infrastructure
- Challenge: better learning algorithms

What's new here?

- DBMS have tight-integration with language classes (Java, C#, VB, …)
  - the DB is a class



- you can add classes to DB.
- adding indices is "easy"
  if you have a new idea.

- now have solid Queue systems
  adding workflow is "easy"
  if you have a new idea.

- this is a vehicle for publishing data on the Web.

DM – DB Synergy

Create the model:

```
CREATE MINING MODEL  HeightFromAgeSex
                ( ID long key,
                  Gender  text discrete,
                  Age long continuous,
                  Height long continuous PREDICT)
                  USING Decision_Trees
```

learn height from Gender + Age

Train a data mining model:

```
INSERT INTO Height
        SELECT  ID, Gender, Age, Height
        FROM People
```

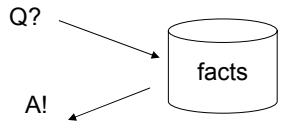
DB verbs to drive Modeler

Predict height from model:

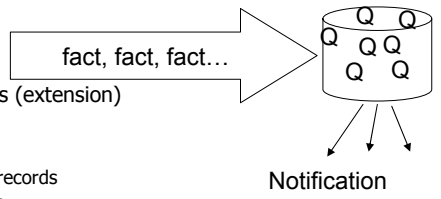
```
SELECT height,
        PredictProbability(height)
FROM Height PREDICTION JOIN New
        ON   New.Gender = Height.Gender
        AND New.Age      = Height.Age
```

Probabilistic Reasoning

Notification, Stream Processing, and Sensor Processing

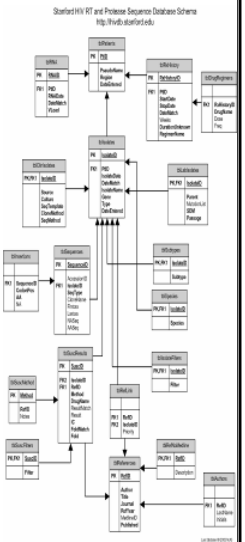


- **traditionally:**
  query billions of facts
- **streams:**
  millions of queries one new fact
  - new protein compare to all DNA
  - change in price or time
- **implications**
  - new aggregation operators (extension)
  - new programming style
  - streams in products:
    - queries represented as records
    - new query optimizations
- **sensor networks**
  - push queries out to sensors.
  - simpler programming model
  - optimizes power & bandwidth



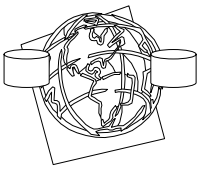
Semi-Structured Data

- DB people are a "strong schema" community
- that has pros-and-cons
- files <stuff/> and XML <<foo/> <bar/>> are here to stay. *Get over it!*
- *"everyone starts with the same schema: <stuff/>."* then they refine it." J. Widom
- file directories are becoming databases;
  - pivot on any attribute
  - folders are standing queries
  - freetext + schema search (better precision/recall)
- XSD (xml schema) and XQuery are transitional; but we have to do them to get to the real answer
- cohabit with row-stores
- challenge: figure out what comes after XSD+xQuery



Publish-Subscribe, Replication, Extract-Transform-Load

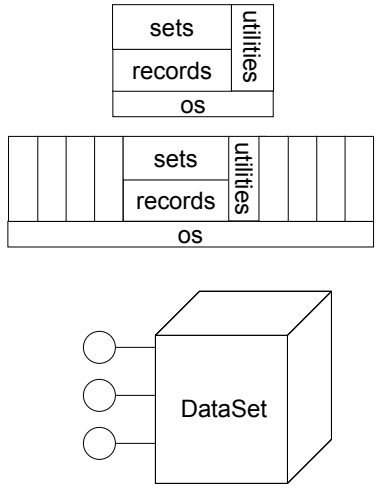
- data has many users
- replicas for availability and/or performance (e.g. directories.)
- mobile users do local updates synchronize later.
- classic warehouse
  - replicate to data warehouse
  - data marts subscribe to publications



- disaster recovery geoplex
- many different algorithms:
  - transactions, 1-safe, snapshot, merge, log ship,…
  - each algorithm seems to be best for something.
- extract-transform-load (ETL) is a major application & component
  - data loading
  - data scrubbing
  - publish/subscribe workflows.
- all use procedures for reconciliation, scrubbing,…

Restatement: DB Systems evolved to be containers for information services develop, deploy, and execution environment

- DBMS is an ecosystem – key structuring strategy:
  - everything is a class
  - database is a complex object
  - core object is DataSet
- This architecture uses many of your ideas
- The architecture lets you add your new ideas.



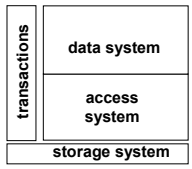
How can such a Wish List be Accomplished?

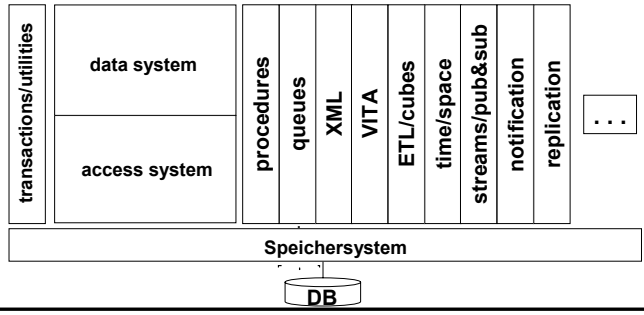
- Several types (e.g. VITA) are "far" away from the processing invariants of the layer architecture
  - other abstraction levels required
  - storage system including external files: least common denominator at best
- DBMS become "Containers for Information Services" (J. Gray)
  - variety of **type-specific** DBMS architectures
  - extensible object-relational systems with **non-procedural operators for sets of objects**
  - each DBMS is a Web service
- What happens with the transactional control in such an ecosystem?

Recommendation: Jim Gray: A Call to Arms. ACM Queue 3:3, 30-38, April 2005

Revolution in the DBMS Architecture?

- (O)RDBMS architecture is exhausted
  - Extenders, DataBlades, …
  - extensibility infrastructure, …
- Despite these architectural enhancements the layer model can not well serve the new data types
  - VITA, streaming, external files, …
  - new data types can not adequately be supported by the classic invariants of DB processing
  - cooperation of individual architectures in a DBMS ecosystem?





Is there Nothing Left for the Plumbers to do?

- **Basic question: detailed architecture of ecosystem members**
  - can everything be done as an extension?
  - not quite….
  - first, there is LOTS of plumbing in extensions
  - doing the OO integration is not trivial
  - better optimizers
  - deal with massive main memory
  - security & privacy still an open problem
  - federation, Distribution, parallelism
  - auto-everything
- **What is an appropriate transaction model for the members in the ecosystem?**

Smart Objects: Databases Everywhere

- **How do we integrate large numbers of small, mobile, heterogeneous, … systems**
  - phones, PDAs, cameras, … have small DBs
  - disk drives have enough cpu, memory to run
    → a full-blown DBMS
  - all these devices want-need to share data
  - they need an Esperanto
  - it is the DBMS ecosystem language
  - needs a simple-but-complete DBMS

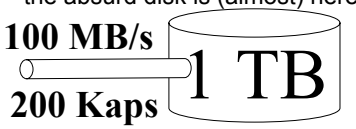
Late Binding in Query Plans

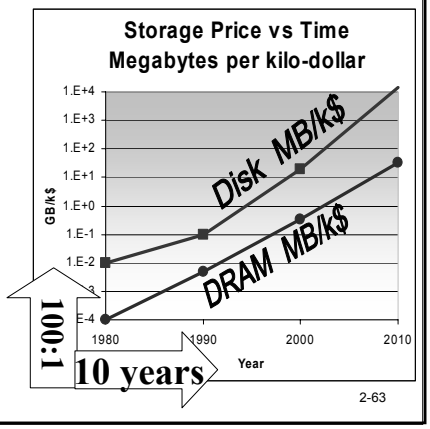
- Cost based query optimizers are great!
  – when they guess right
- But
  - if it guessed 1 minute and the query has been running for a day…
  - if system is busy plan is different
- Better strategy: have query optimizer learn
  - from previous queries
  - from previous instances of this query
  - from this query
  - from environment
- Anyone who has waited days for a query to complete thinks this VERY important (!)

Massive Memory, Massive Latency

- **Some problems**
  - RAM costs
    ~ 100k$...300k$/**TeraByte**
  - 64 bit addressing everywhere
  - latency a problem
  - NUMA latency a problem
  - checkpoint 1TB?
    restart 1TB?
    scan 1TB
  - OK,
    now how about 100TB?
  - challenge: algorithms for massive main memory

the absurd disk is (almost) here



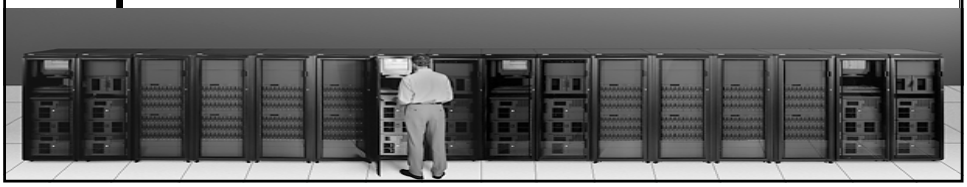


Trends in DBS

Distribution & parallelism
Requirements
Processing concepts
Future requirements
New DBMS data types
Architectural issues
Overflow

Self Managing & Always Up

- **Needs for self-***
  - people costs have always exceeded IT capital
  - but now that hardware is "free" …
  - self-managing, self-configuring, self-healing, self-organizing and … is key
  - no DBAs for cell phones or cameras
  - requires a modular software architecture
    - clear and simple knobs on modules
    - software manages these knobs
- **So, again the class model (interfaces) are key**



Summary

- **DB system properties**
  - high performance (high transaction rates, parallel processing of complex queries)
  - support of very large DBs
  - high availability and fault tolerance for all components
  - modular extensibility, etc.
- **Processing support**
  - diverse kinds of intra-transaction parallelism
  - evaluated by speedup and scaleup metrics
  - adaptivity in and across components
- **OO enables Object-Relational Ecosystem**
  - federate many kinds of data
  - enables DB extensions
- **Yes, there are still plumbing problems left**
  - framework to allow extensions
  - auto-everything.

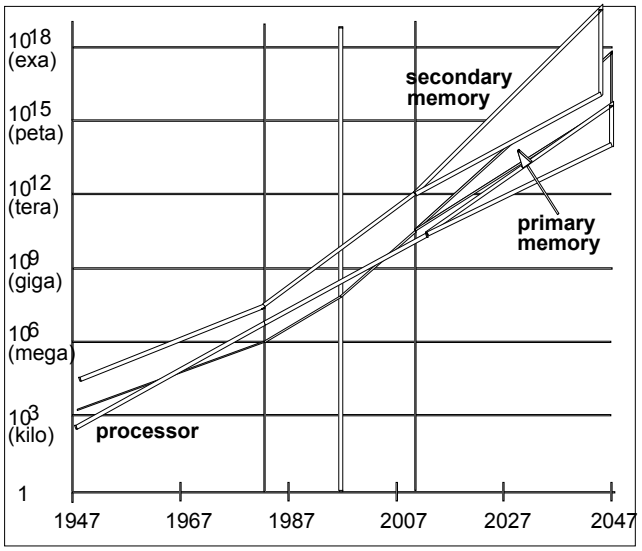
Further References

- *Gray, J.:* What Next?: A Dozen Information-Technology Research Goals. J. ACM 50(1): 41-57 (2003) (Journal Version of the 1999 ACM Turing Award Lecture)
- *Gray, J.:* A Call to Arms. ACM Queue 3:3, 30-38, April 2005
- *Härder, T.:* DBMS Architecture – New Challenges Ahead, Datenbank-Spektrum, dpunkt-Verlag, Heft 14, Aug. 2005, pp. 38–48.
- *Härder, T., Rahm, E.*: Datenbanksysteme: Konzepte und Techniken der Implementierung. 2nd edition, Springer 2001, Kap. 1
- *Rahm, E.:* Mehrrechner-Datenbanksysteme - Grundlagen der verteilten und parallelen Datenbankverarbeitung. Addison-Wesley 1994
- *Weikum, G., Mönkeberg, A., Hasse, C., Zabback, P.:* Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering. VLDB 2002: 20-31

What are the Technological Drivers?

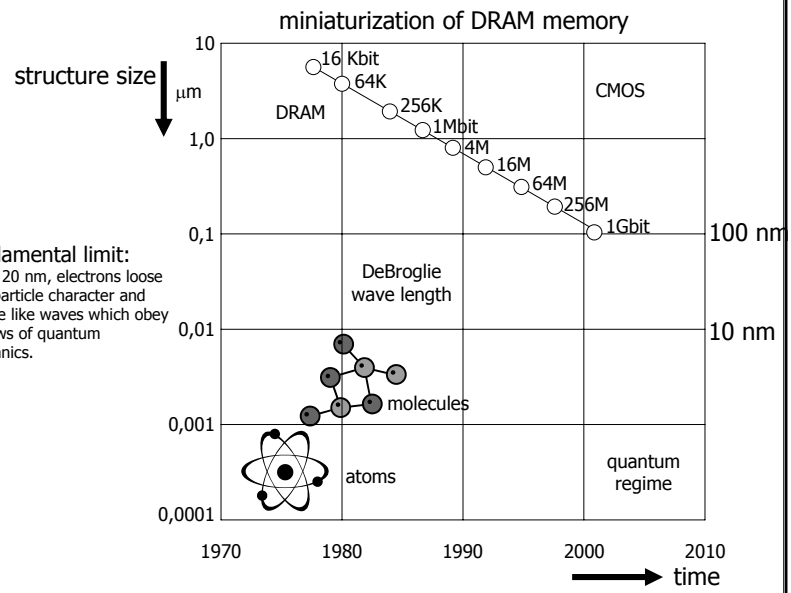
Evolution of the processing speed of computers in instructions per second and primary/secondary memory size in bytes from 1947 to present, with a "surprise-free" projection to 2047.

Each division represents 3 orders of magnitude and happens roughly in 15-years steps.



Nano-Electronics Comes Closer

Fundamental limit:
Below 20 nm, electrons loose their particle character and behave like waves which obey the laws of quantum mechanics.



Technology Trends

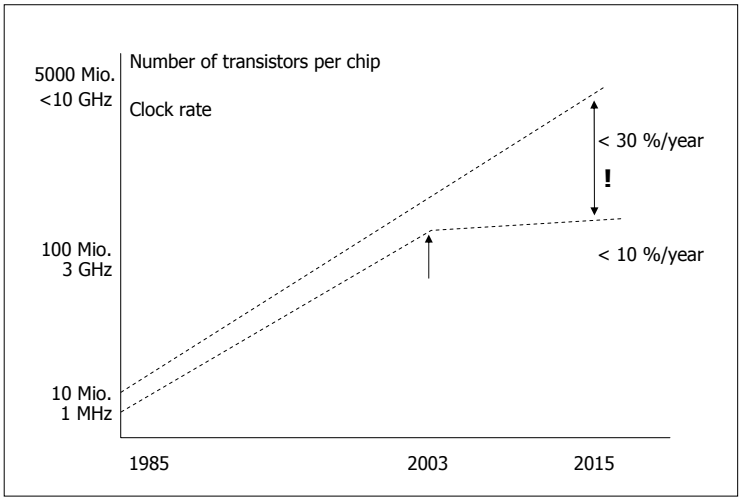
- The technology challenges for all microprocessor manufacturers:
  - leak streams, performance density !!

"Moore's law 2005" : Double ~~speed~~ heat every 18 … 24 months ???

Performance ≠ GHz ???

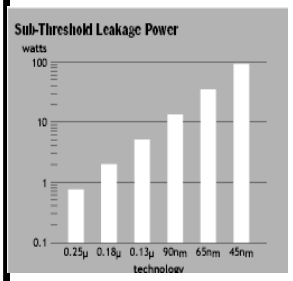
- Consequences, trends
  - not GHz at any price!
    - example Intel : => Pentium-III/-M-type design @ 2…3… GHz
  - revision of design priorities:
    - more performance/chip by more processors/chip ("MultiCoreChips" instead of "UHF"-Single-Core with hardly controllable heat build-up
  - separation of processor lines
    - high-end / high-performance server (e.g., …, Itanium ???)
    - "Performance/Watt" optimized high-volume processors

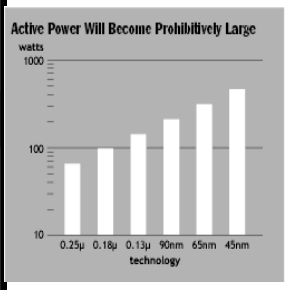
Heat Thwarts Chip Development

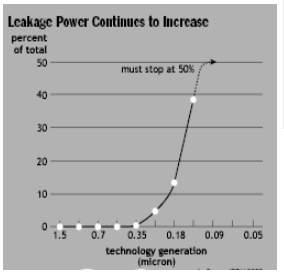


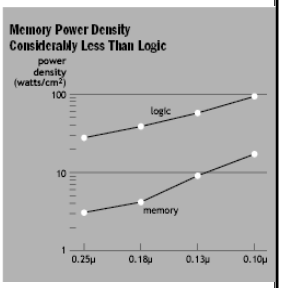
For thermal reasons, the clock rate does not grow anymore as fast as the # of transistors/chip
- technology forecast for transistors per chip and clock rate of processors
- multi-processor systems or multi-kernel systems on a chip

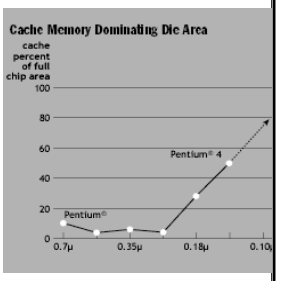
Is Moore's Law Coming to an End?











What is CELL?

- What is CELL?
  - A new family of processors optimized for compute-intensive and broadband rich media applications
  - A standard architecture that is scalable and power-efficient
    - Outstanding Flops, Flops/Watt, and Flops/Area
  - First implementation in 90nm SOI technology and modular System-on-Chip methodology
    - Advantage per chip & per Watt of about 10x to 100x on well behaved applications (graphics, media, DSP, …)
    - Supercomputer like performance on a chip

The First-Generation CELL Processor

- Multi-Core
  - Power Processor Element (PPE)
  - 8 Synergistic Processor Elements (SPE)
- High bandwidth interfaces
  - Element Interconnect EIB
    - Flexible bandwidth, up to 76.8 GB/s
  - Memory interface: dual XDR
    - On-chip-controller, up to 25.6 GB/s
- Implementation
  - Designed for 11 F04 with air-cooled envelope
  - Observed clock speed: > 4 GHz
  - Peak performance
    - Single precision > 256 GFlops
    - Double precision > 26 GFlops

Synergistic Processor Element (SPE)

Provides computational performance of CELL processor
Optimized for media & streaming applications

- Each unit operates on 128b
  - 4-, 8- and 16-way SIMD
- Two instruction per cycle
  - Dual issue to 2 pieces
- Large 128 x 128b register file
  - Shared between all units
  - Enables multiple loop-unrolling
- High frequency design w/o design complexity
  - Software controlled Local-Store instead of Cache
  - Limited OoO to overcome dependencies
  - Optimized ISA
- High bandwidth, DMA interface

SPE – Power Performance Tradeoffs

Vdd (Volts) \ Freq (GHz)	2	2.2	2.4	2.6	2.8	3	3.2	3.4	3.6	3.8	4	4.2	4.4	4.6	4.8	5
1.3	48C 4W	49C 4W	50C 5W	50C 6W	51C 6W	52C 7W	53C 7W	54C 7W	55C 8W	56C 8W	57C 9W	58C 9W	59C 10W	60C 10W	61C 10W	63C 10W	61C
1.2	39C 2W	39C 3W	40C 3W	41C 4W	42C 4W	42C 4W	43C 5W	44C 5W	45C 5W	45C 5W	46C 6W	47C 6W	47C 7W	48C	49C		
1.1	32C 2W	33C 2W	33C 3W	35C 3W	35C 3W	36C 3W	36C 4W	37C 4W	37C 4W	38C 4W	38C 4W	39C	39C				
1	28C 2W	28C 2W	29C 2W	29C 2W	30C 2W	30C 3W	30C 3W	31C 3W	31C 3W	31C 3W	32C						
0.9	25C 1W	26C 1W	26C 1W	26C 2W	27C 2W	27C 2W	27C	Failed									

B. Flachs et al, ISSCC 2005

- Schmoo measured on 1st Hardware: up to **5.6 GHz @ 1.4 V and 56° C**
- Chip for high-volume consumer market
  - Little sorting (buy-all), all systems shipped at the same frequency
  - Air-cooled envelope, low power budget for each SPE
  - →Limited use of "technology tricks" used for conventional high-performance chips (e.g. servers)
- Performance improvements have to come from the design itself

Summary & Outlook

- Cell starts a new era of leading edge processors optimized for digital media and entertainment
- Cell provides a new level of performance and power efficiency
  - Enabling entire new classes of applications
  - Real-time graphics with physical simulations for more realism
    - E.g.: TRE, real-time, ray-tracing …
- Cell Processors can support many systems
  - Game console systems
  - Workstations
  - HDTV, home media servers
  - Supercomputers
  - …
- Wide set of technical specs has been made available to the public www.ibm.com/chips/power/splash/cell