Mammals Flourished Long Before Dinosaurs Became Extinct

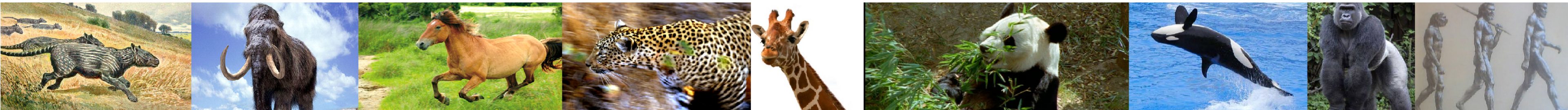
VLDB 2009 Lyon - Ten Year Award

"Database Architecture Optimized For The New Bottleneck: Memory Access" (VLDB 1999)

Stefan Manegold (manegold@cwi.nl)

Peter Boncz (boncz@cwi.nl)

Martin Kersten (mk@cwi.nl)



MICHAEL STONEBRA

… What I see happen … data-base vendors, wh … ants, are selling a one- … old architect … newhere in the late 1

EVOLUTION

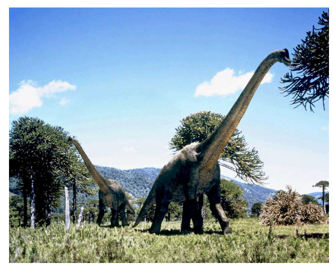
ACM QUEUE May/June 2007


Mammals
200 MYA
Reptiles
Dinosaur

Mammals

Reptiles


Mamals
200 MYA
Reptiles
Dinosaur



Mammals

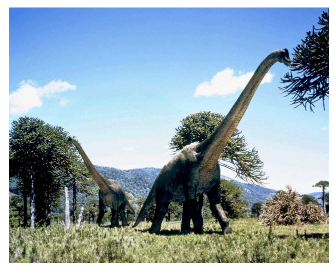
Reptiles

Mammels

200 MYA

Reptiles

Dinosaur



CWI

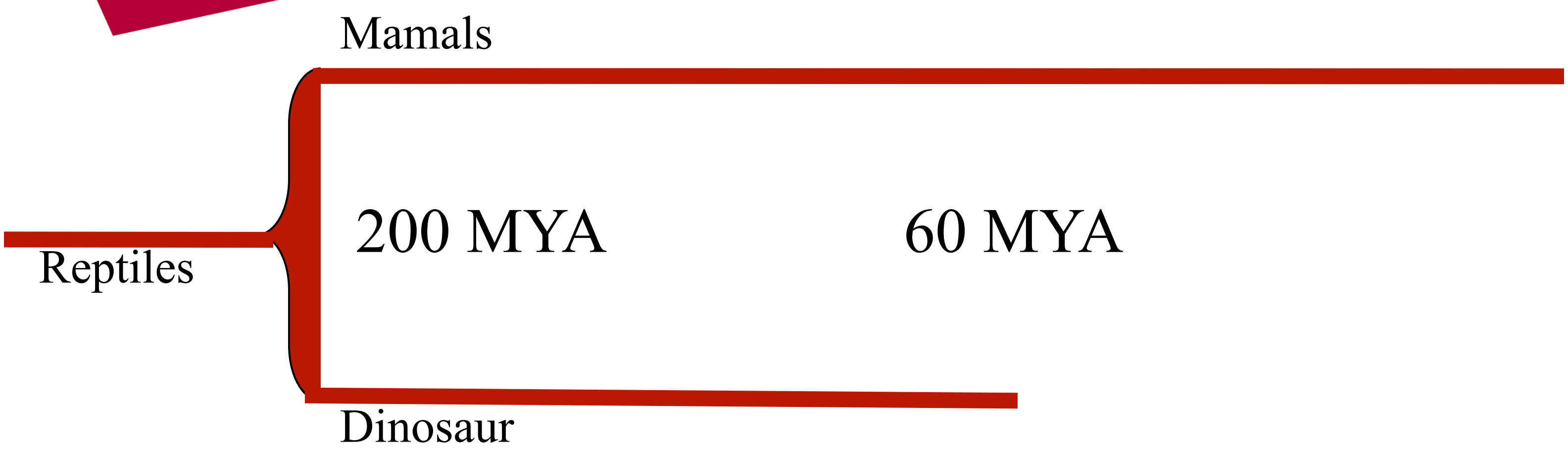
Mamals

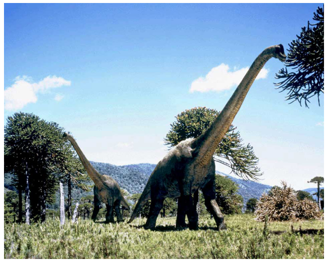
Reptiles

200

Dino

CWI





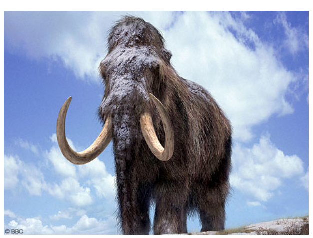
CWI

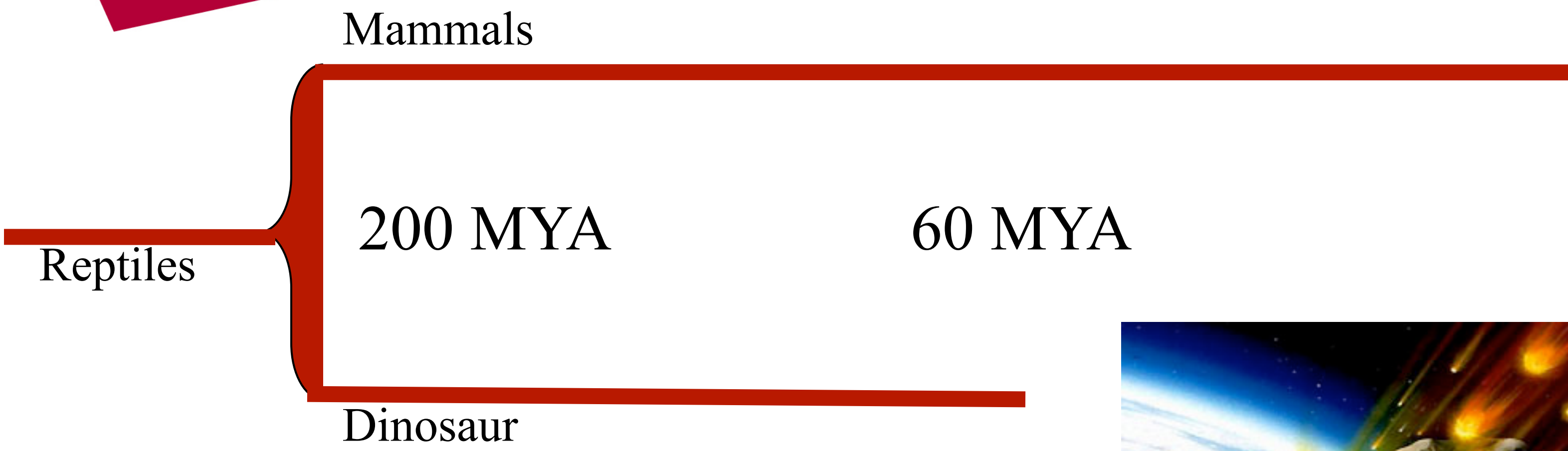
Mamals

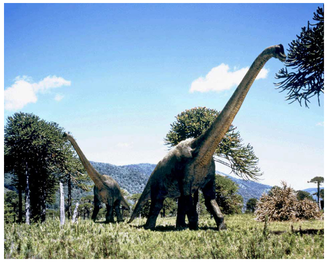
200

Reptiles

Dinos




Mammals
200 MYA
60 MYA
Reptiles
Dinosaur



Large mammals once dined on dinosaurs

Repenomamus giganticus

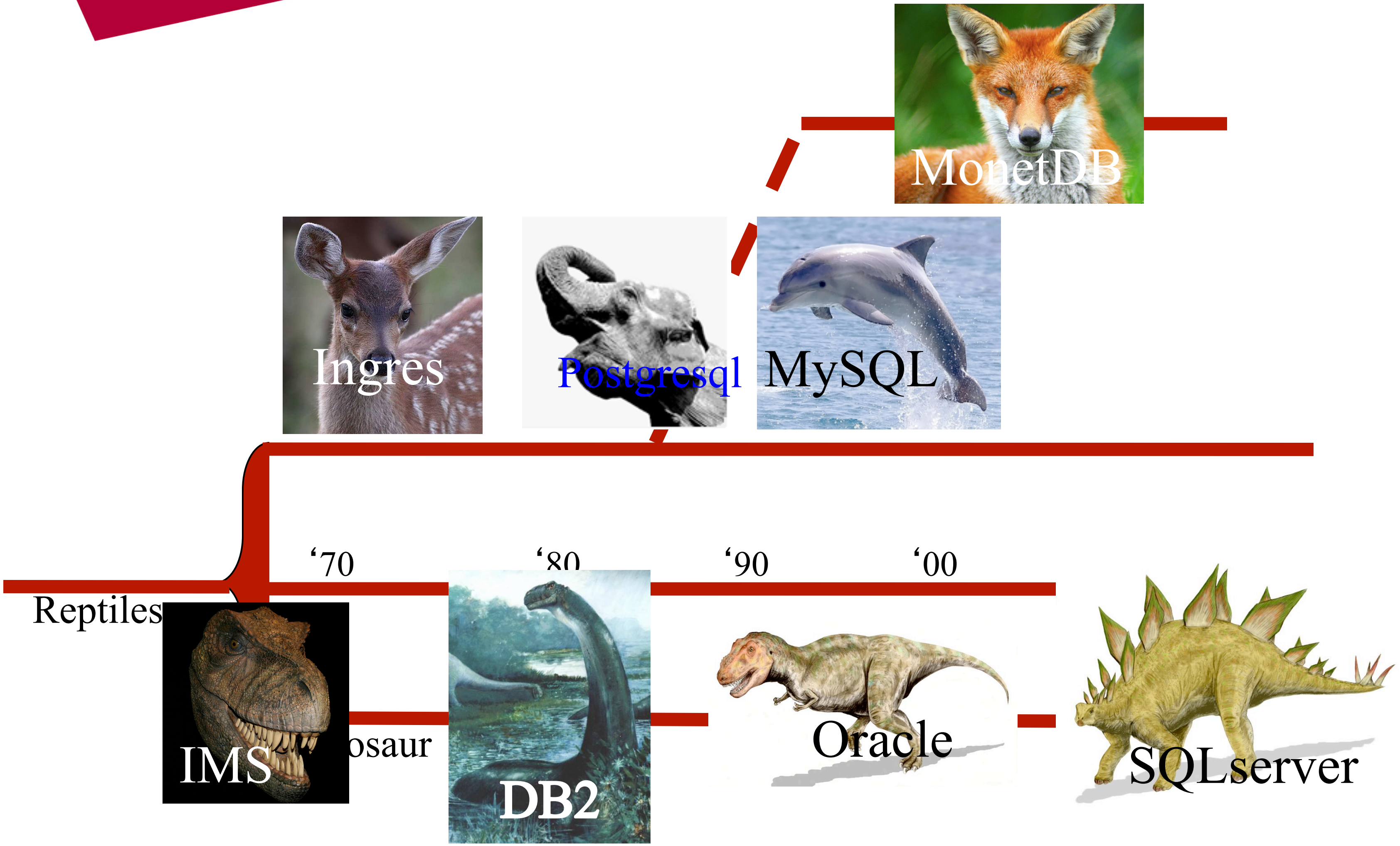
Repenomamus robustus

Yaoming Hu, Jin Meng, Yuanqing Wang and Chuankui Li149 *Large Mesozoic mammals fed on young dinosaurs* Nature (vol 433, p 149, 2005)

Evolution

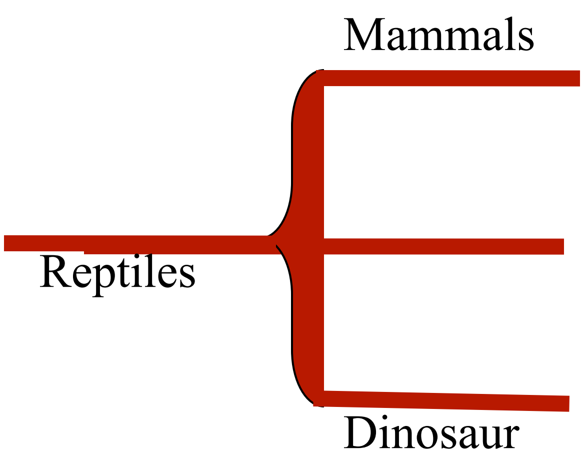
It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change.

Charles Darwin (1809 - 1882)


MonetDB
Ingres
Postgresql
MySQL
‘70
‘80
‘90
‘00
Reptiles
IMS
osaur
DB2
Oracle
SQLserver

CWI

The genes of a species



-SQL86, SQL92,SQL99,SQL03

-n-ary storage scheme

-relational algebra + DDL

-5+ way indexing schemes

-slotted pages of records

-Volcano-style computation

CWI

The evolution of the Fox

1979-1985

Troll a relational engine to simplify relational database programming

SWI Prolog made a much better relational engine then my first system and Ingres, Oracle…

The evolution of the Fox

1979-1985

Troll a relational engine to simplify relational database programming

SWI Prolog made a much better relational engine then my first system and Ingres, Oracle…

Hector Garcia-Molina, Richard J. Lipton, Jacobo Valdes: A Massive Memory Machine. IEEE Trans. Computers 33(5): 391-399 (1984)

David J. DeWitt, Randy H. Katz, Frank Olken, Leonard D. Shapiro, Michael Stonebraker, David A. Wood: Implementation Techniques for Main Memory Database Systems. SIGMOD Conference 1984: 1-8

The evolution of the Fox

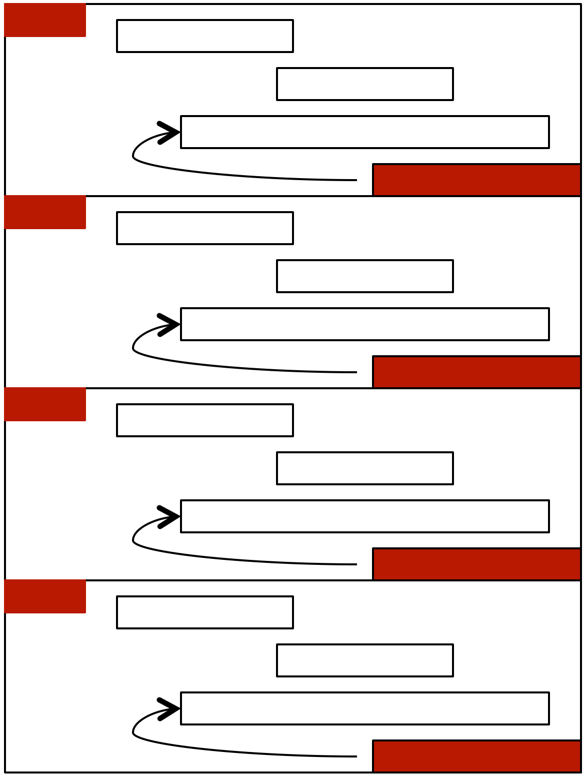
1979-1985

Troll a relational engine to simplify relational database programming

Non-first-normal-form disease

Object-orientation religion

IO pages size increase



512 bytes

The evolution of the Fox

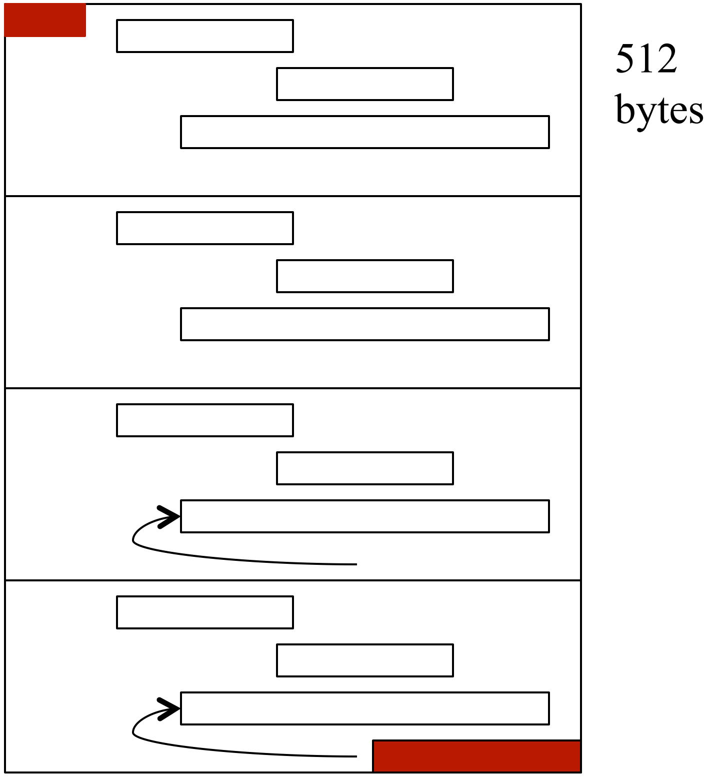
1979-1985

Troll a relational engine to simplify relational database programming

Non-first-normal-form disease

Object-orientation religion

IO pages size increase



The evolution of the Fox

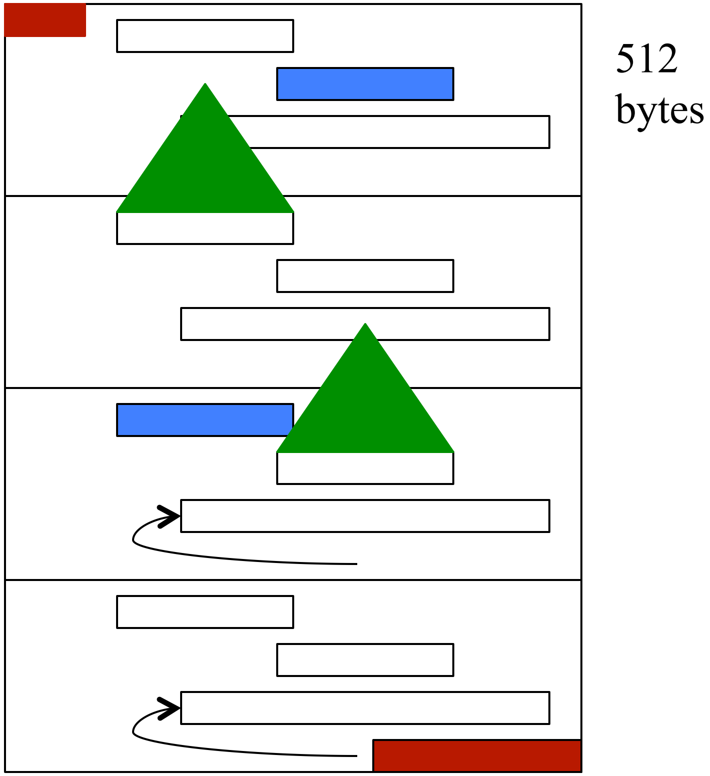
1979-1985

Troll a relational engine to simplify relational database programming

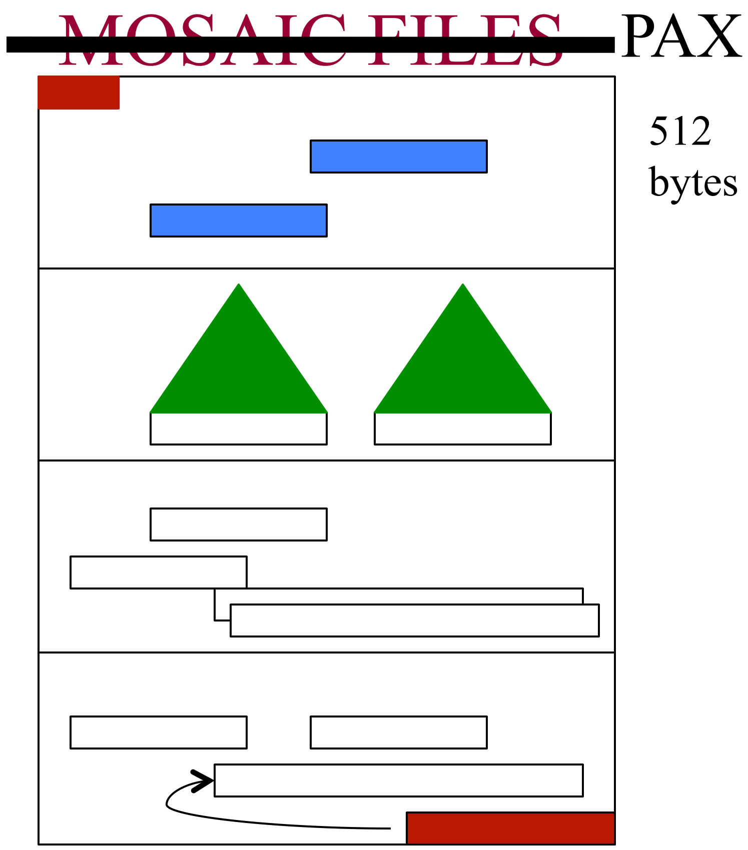
Non-first-normal-form disease

Object-orientation religion

IO pages size increase



The evolution of the Fox



1979-1985

Troll a relational engine to simplify relational database programming

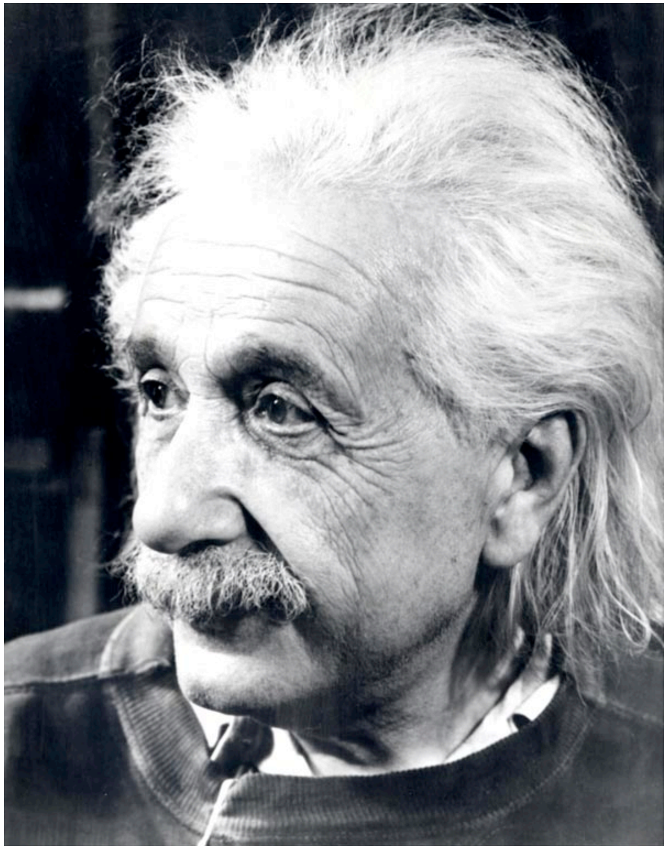
Non-first-normal-form disease

Object-orientation religion

IO pages size increase

Albert Einstein

"We can't solve problems by using the same kind of thinking we used when we created them."



CWI

A DECOMPOSITION STORAGE MODEL

SIGMOD 1985

George P Copeland
Setrag N Khoshafian

2 1 Support Of Multivalued Attributes

A more comprehensive data model than normalized relations might allow multivalued

2 2 Support Of Entities

A more comprehensive data model than the original relational model might support the notion

2 3 Support Of Multiple Parent Relations

A data model with more generality than relations might allow multiple parent relations, where a single record can have more than one parent

2 4 Support Of Heterogeneous Records

A data model with more generality than relations might allow heterogeneous records, where records of a single relation can have different

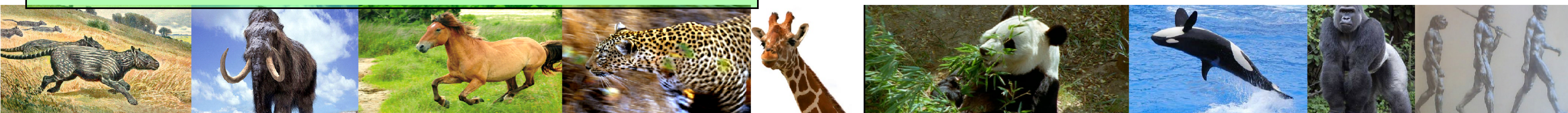
2 5 Support Of Directed Graphs

A data model with more generality than relations might allow a directed graph structure,

The DSM offers simplicity Simple systems have several major advantages over complex systems One advantage is that a set of fewer and simpler functions, given fixed development resources, can be either further tuned in software or pushed further into hardware to improve performance This is similar to the RISC (Patterson and Ditzel 1980) approach in general purpose architectures A second advantage is that many alternative cases with different processing strategies can less often be exploited, since the cases are not always recognized

studies have compared the performance of transposed storage models with the NSM (Hoffer 1976, Batory 1979, March and Severance 1977, March and Scudder 1984) In this report, we describe the advantages of a fully decomposed storage model (DSM), which is a transposed storage model with surrogates included The DSM pairs each attribute value with the surrogate of its conceptual schema record in a binary relation For example, the above relation would be stored as

a1	sur	val	a2	sur	val	a3	sur	val
	s1	v11		s1	v21		s1	v31
	s2	v12		s2	v22		s2	v32
	s3	v13		s3	v23		s3	v33



CWI

The genes of a new species

-SQL86, SQL92,SQL99,SQL03
-n-ary storage scheme
-relational algebra + DDL
-5+ way indexing schemes
-slotted pages of records
-Volcano-style computation

The genes of a new species

-SQL86, SQL92,SQL99,SQL03

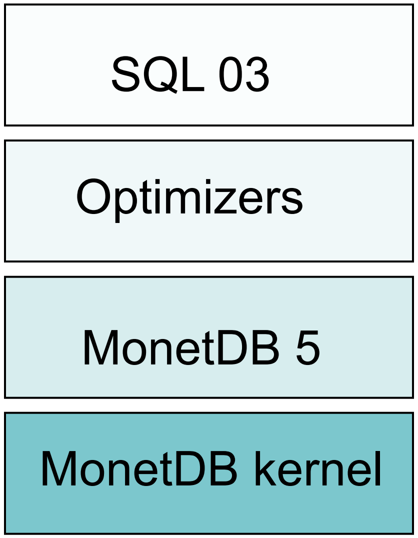
Binary Association Tables — [redacted] storage scheme

-relational algebra + DDL

Self managing — [redacted] indexing schemes

Arrays — [redacted] of records

Materialize operator — [redacted] computation

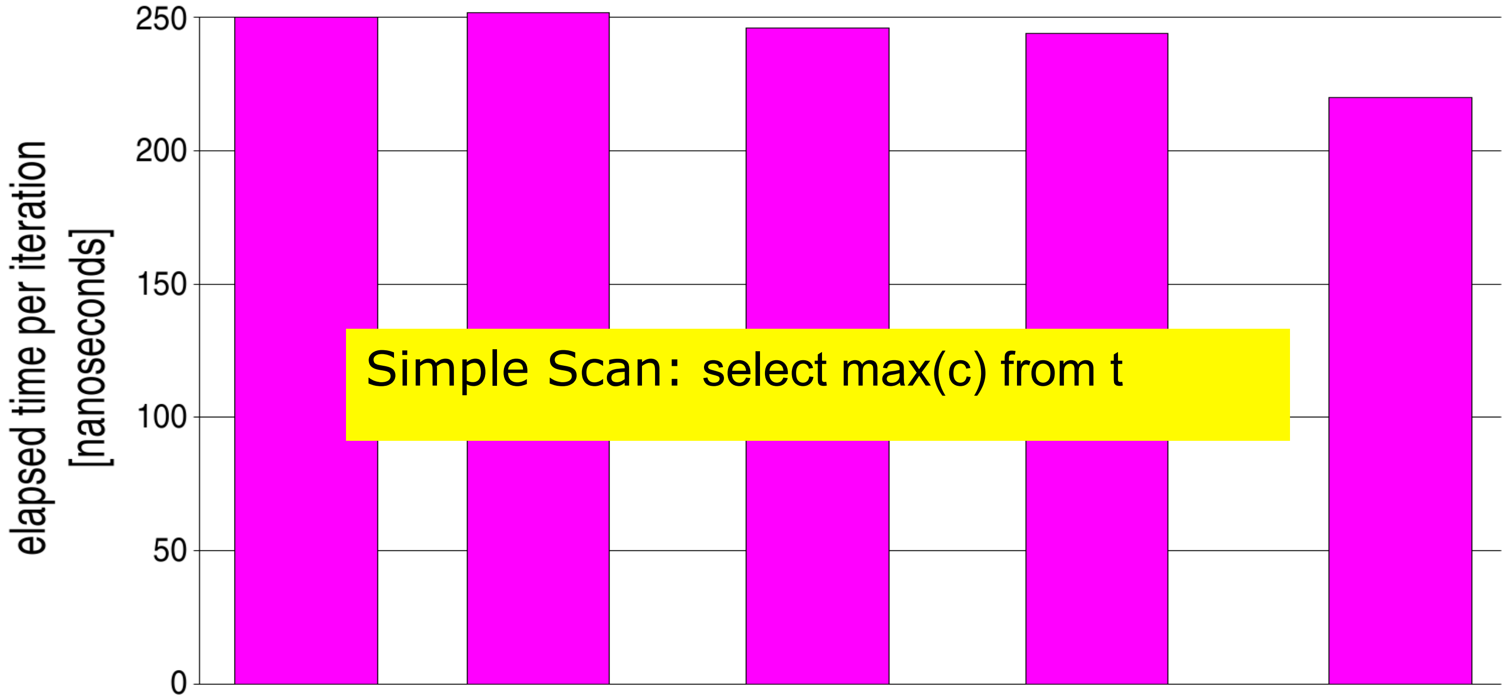


The MonetDB Software Stack

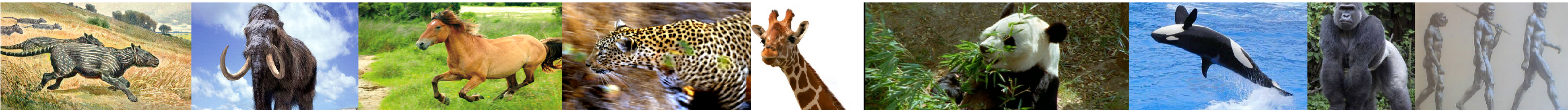
Cache-Conscious Query Processing in MonetDB

Stefan Manegold (manegold@cwi.nl)
Peter Boncz (boncz@cwi.nl)
Martin Kersten (mk@cwi.nl)

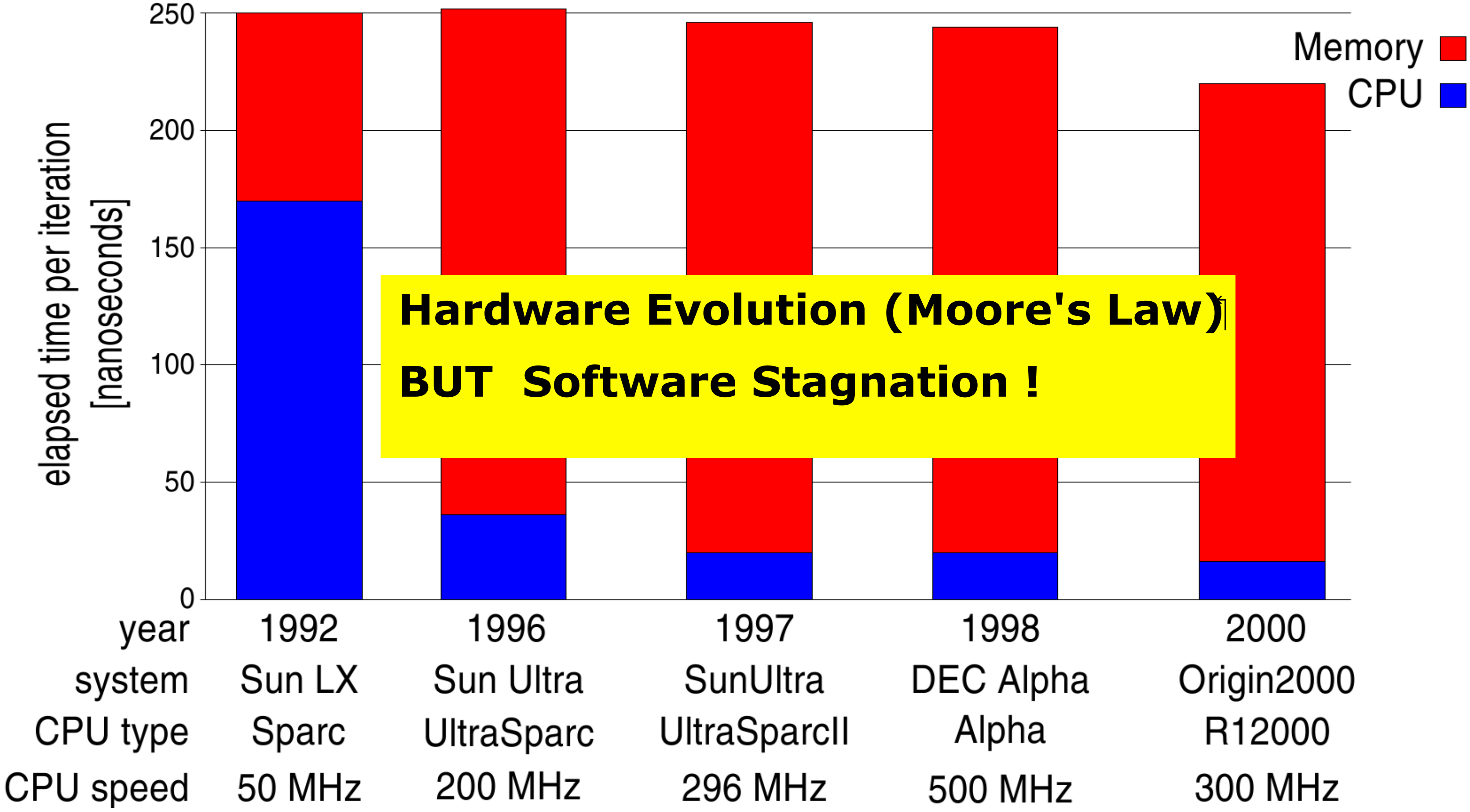
Evolution == Progress?



year	1992	1996	1997	1998	2000
system	Sun LX	Sun Ultra	SunUltra	DEC Alpha	Origin2000
CPU type	Sparc	UltraSparc	UltraSparcII	Alpha	R12000
CPU speed	50 MHz	200 MHz	296 MHz	500 MHz	300 MHz

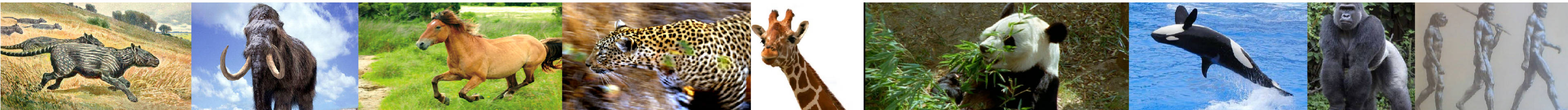


Evolution == Progress?



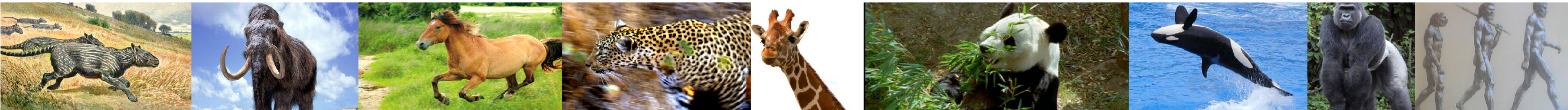
Hardware Evolution (Moore's Law)

BUT Software Stagnation !

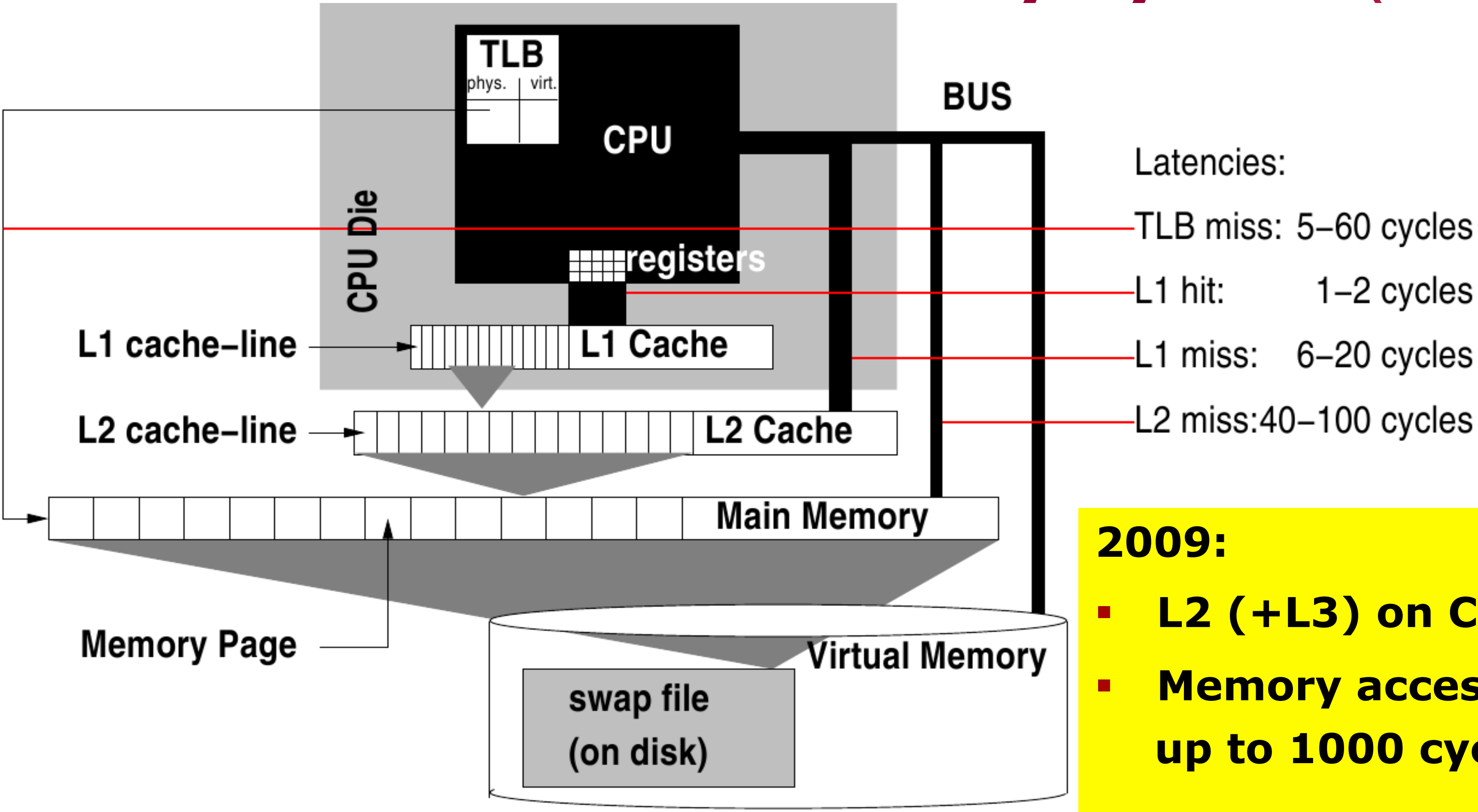


Databases hit The Memory Wall

- Detailed and exhaustive analysis for different workloads using 4 RDBMSs by **Anastassia Ailamaki** et al. in "*DBMSs On A Modern Processor: Where Does Time Go?*" (VLDB 1999)
- CPU is 50%-90% idle, waiting for memory:
  - L1 data stalls
  - L1 instruction stalls
  - L2 data stalls
  - TLB stalls
  - Branch mispredictions
  - Resource stalls

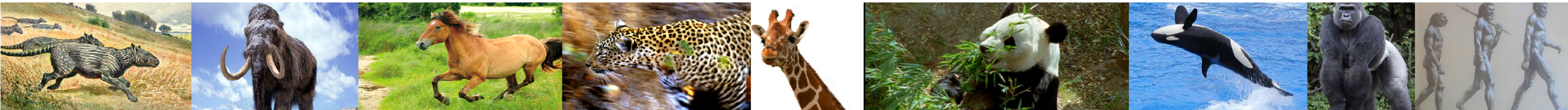


CPU & Hierarchical Memory System (1999)



2009:

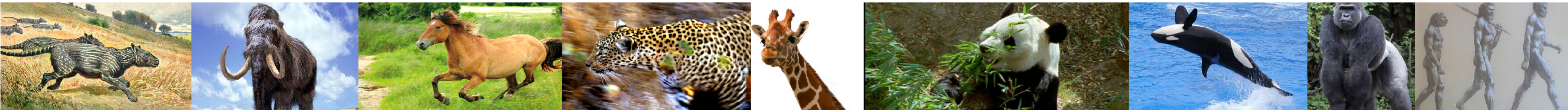
- **L2 (+L3) on CPU die**
- **Memory access: up to 1000 cycles**



Required DBMS Evolution

- Memory access has become a significant cost factor
- Database algorithms suffer particularly from latency (due to random access patterns)

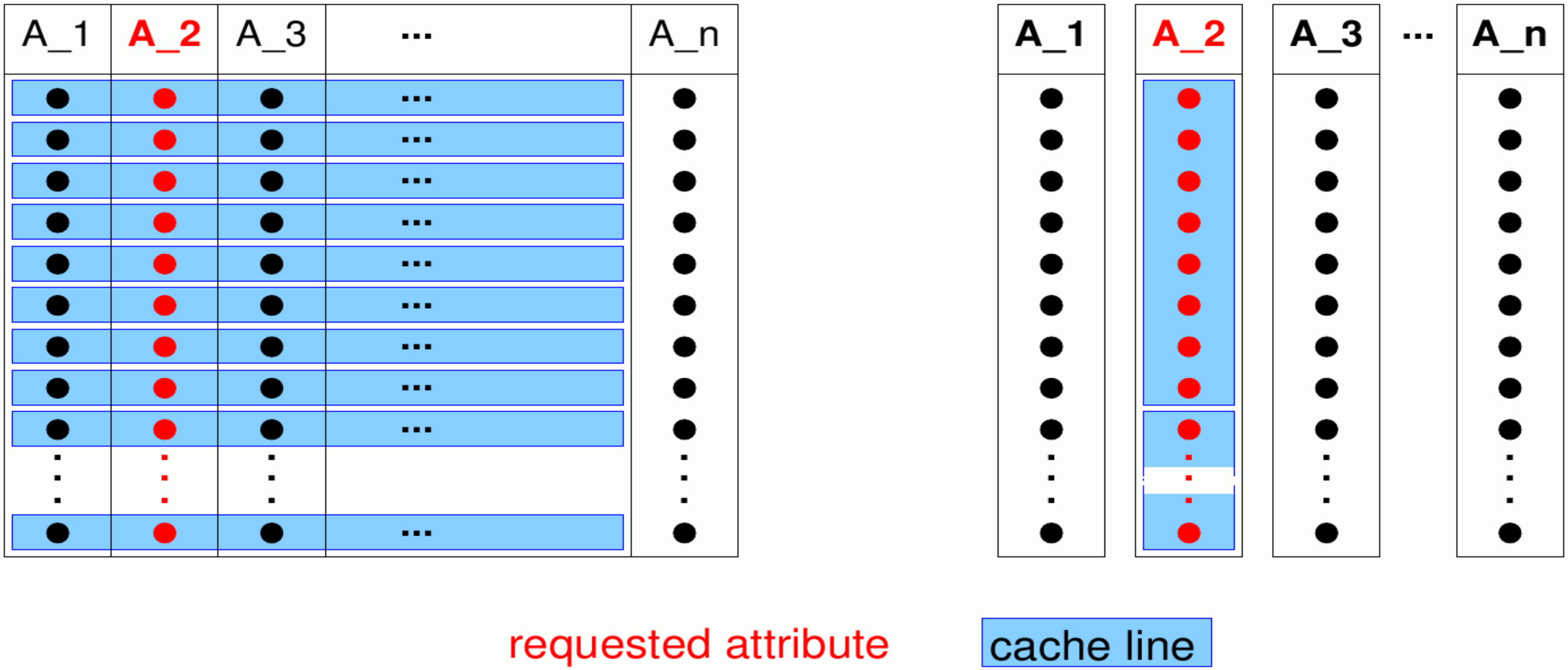
Goal		Optimize
• Use cache lines fully	⇒	Data structures
• Prevent cache & TLB misses	⇒	Memory access / algorithms
• Prevent CPU stalls	⇒	Implementation techniques
• Exploit CPU-inherent parallelism	⇒	Implementation techniques

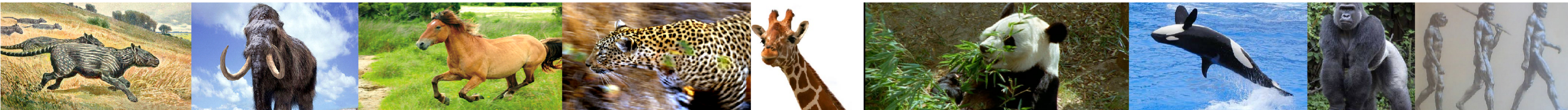


Data Structure Evolution

Row-storage
wastes bandwidth

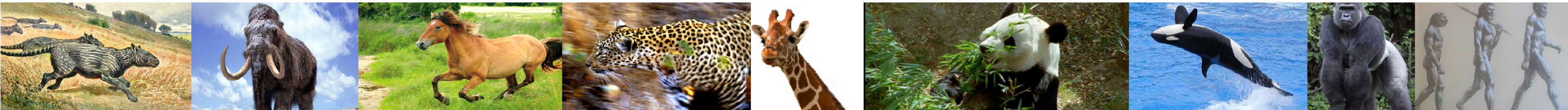
Column-storage
exploits full bandwidth





Algorithm Evolution: Joins

- Nested-loop:
  + sequential access to both inner & outer input
  -- *quadratic complexity*
- Sort-merge:
  + single sequential scan during merge ("benefit")
  -- *random access during sort ("investment")*
- Hash-join:
  + sequential scan over both inputs
  -- *random access to hash table (build & probe)*



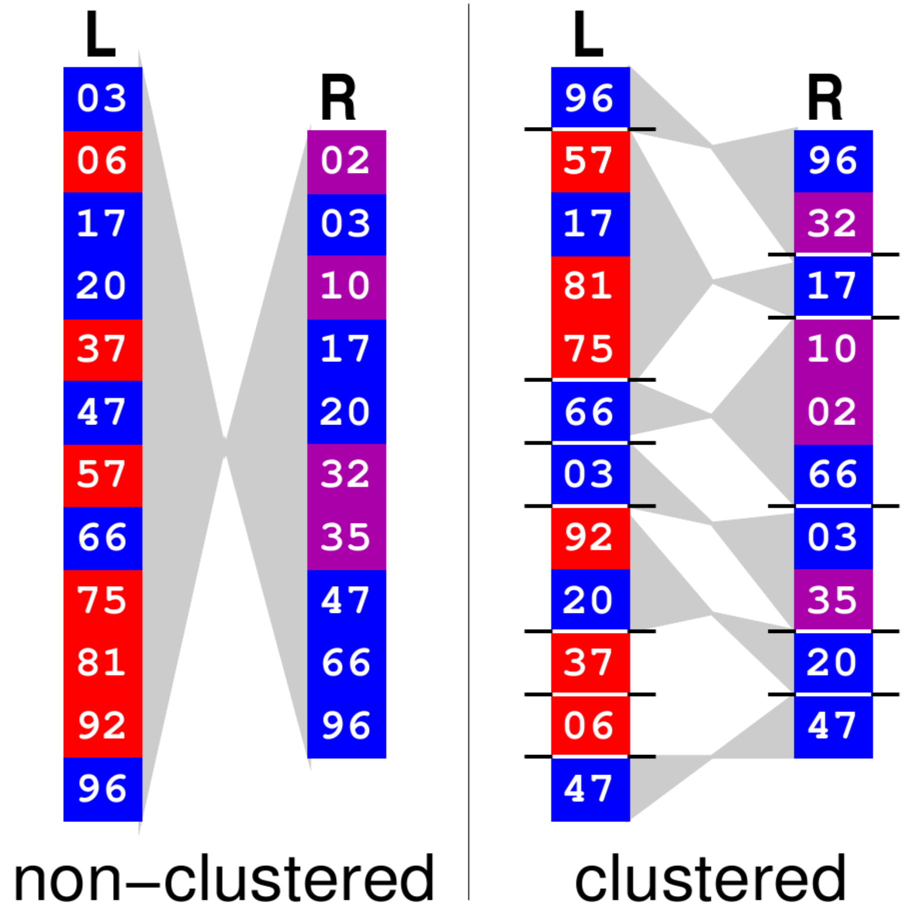
Algorithm Evolution: Partitioned Hash-Joins

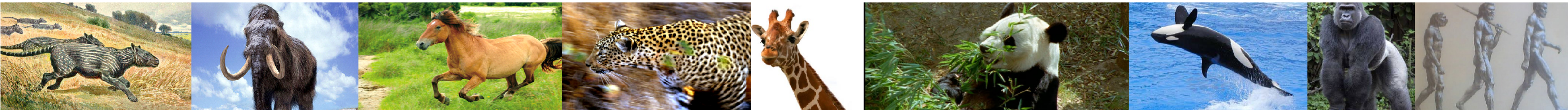
Phase 1:

- Cluster both input relations
- Create clusters that fit in CPU cache
- Restrict random data access to (smallest) cache
- Avoid cache capacity misses

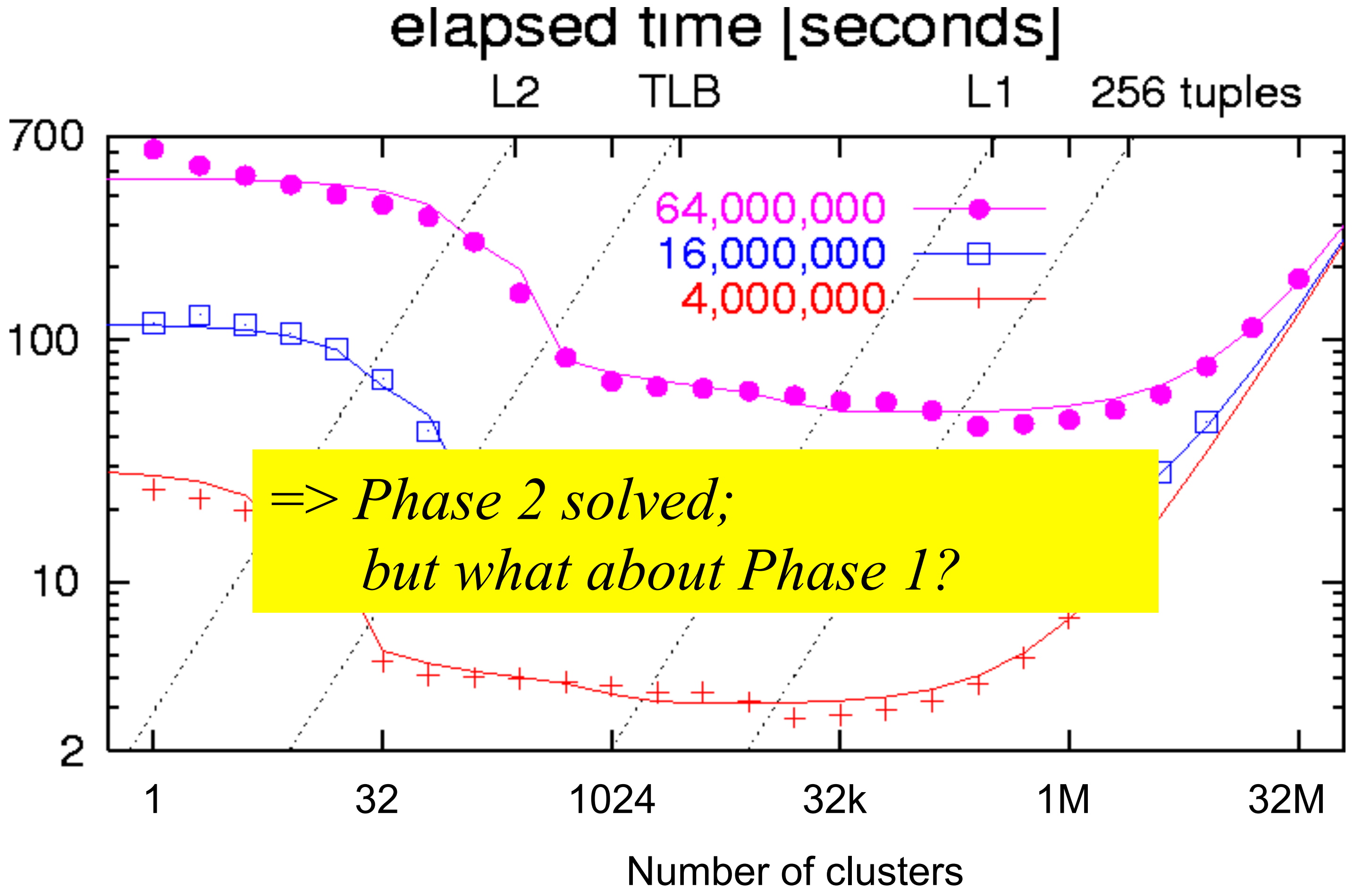
Phase 2:

- Join matching clusters



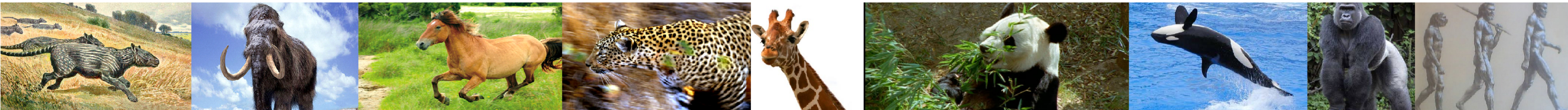


Partitioned Hash-Join: Joining (Phase 2)



Points measured

Lines modeled [VLDB2002]



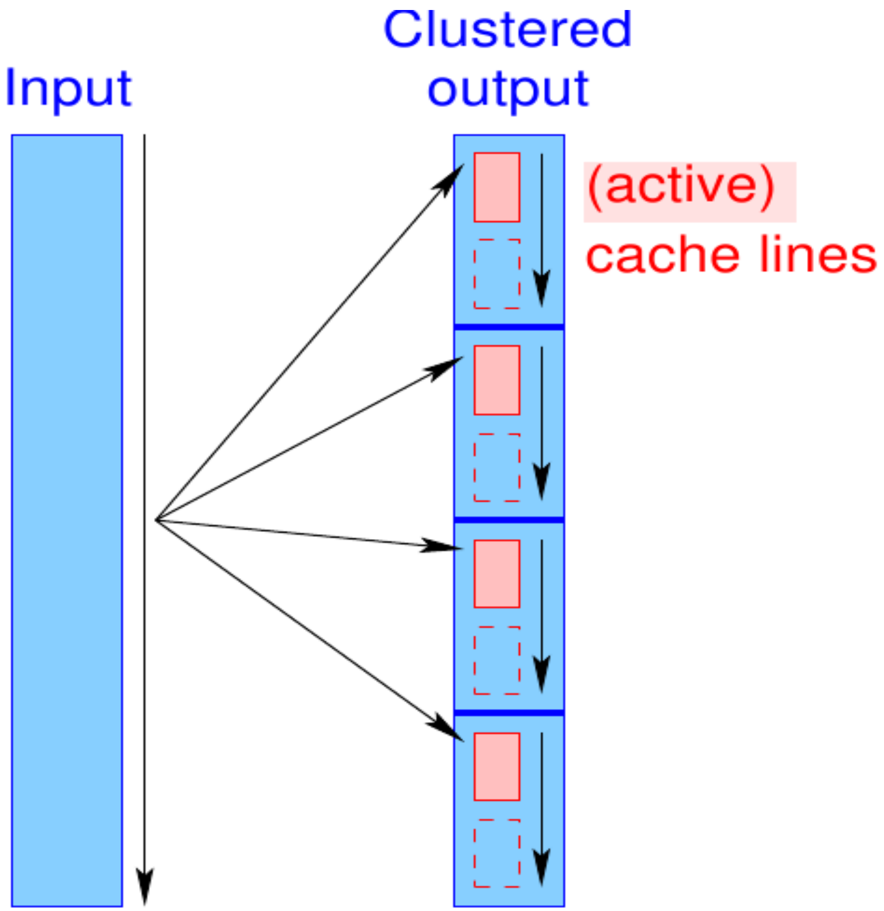
Algorithm Evolution: Clustering

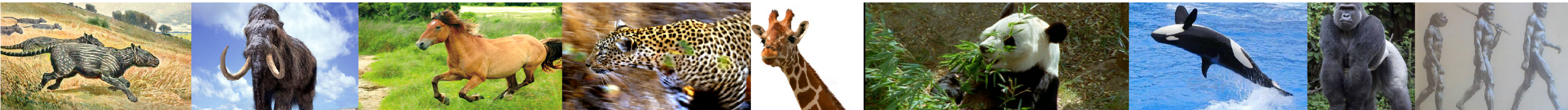
Problem:

- Number of clusters exceeds number of cache lines / TLB entries
- => cache / TLB thrashing

Solution:

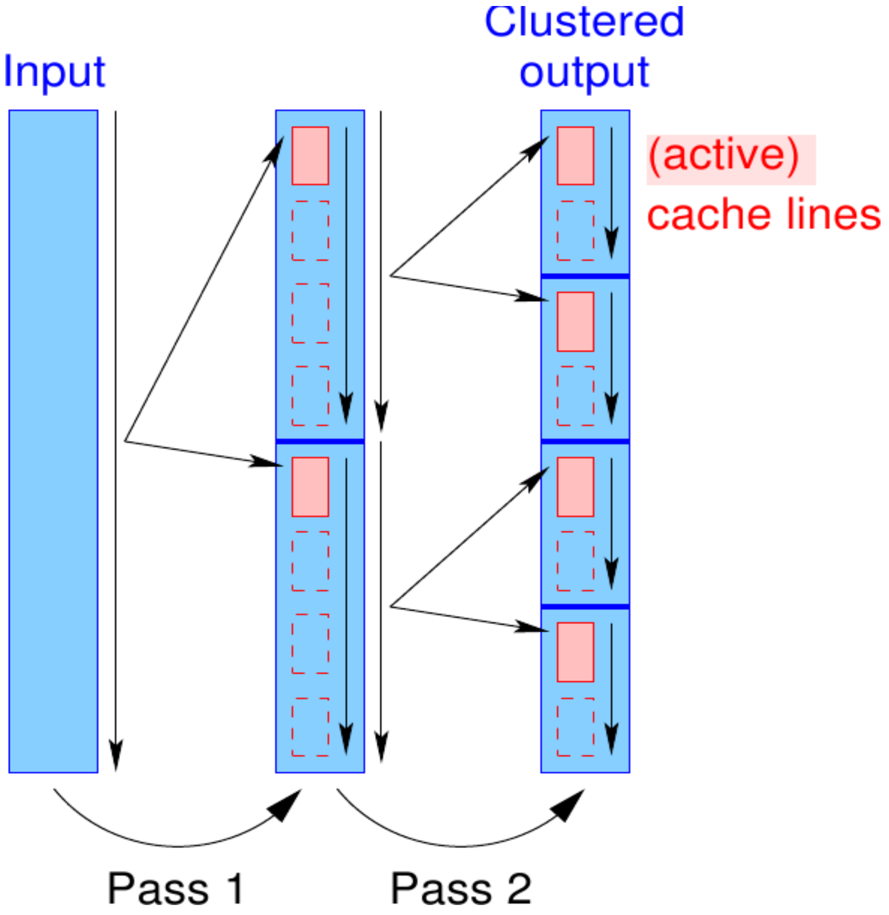
- Multi-pass clustering

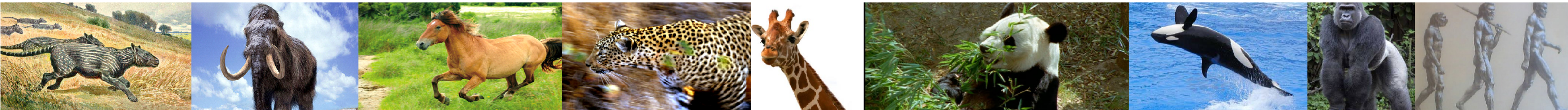




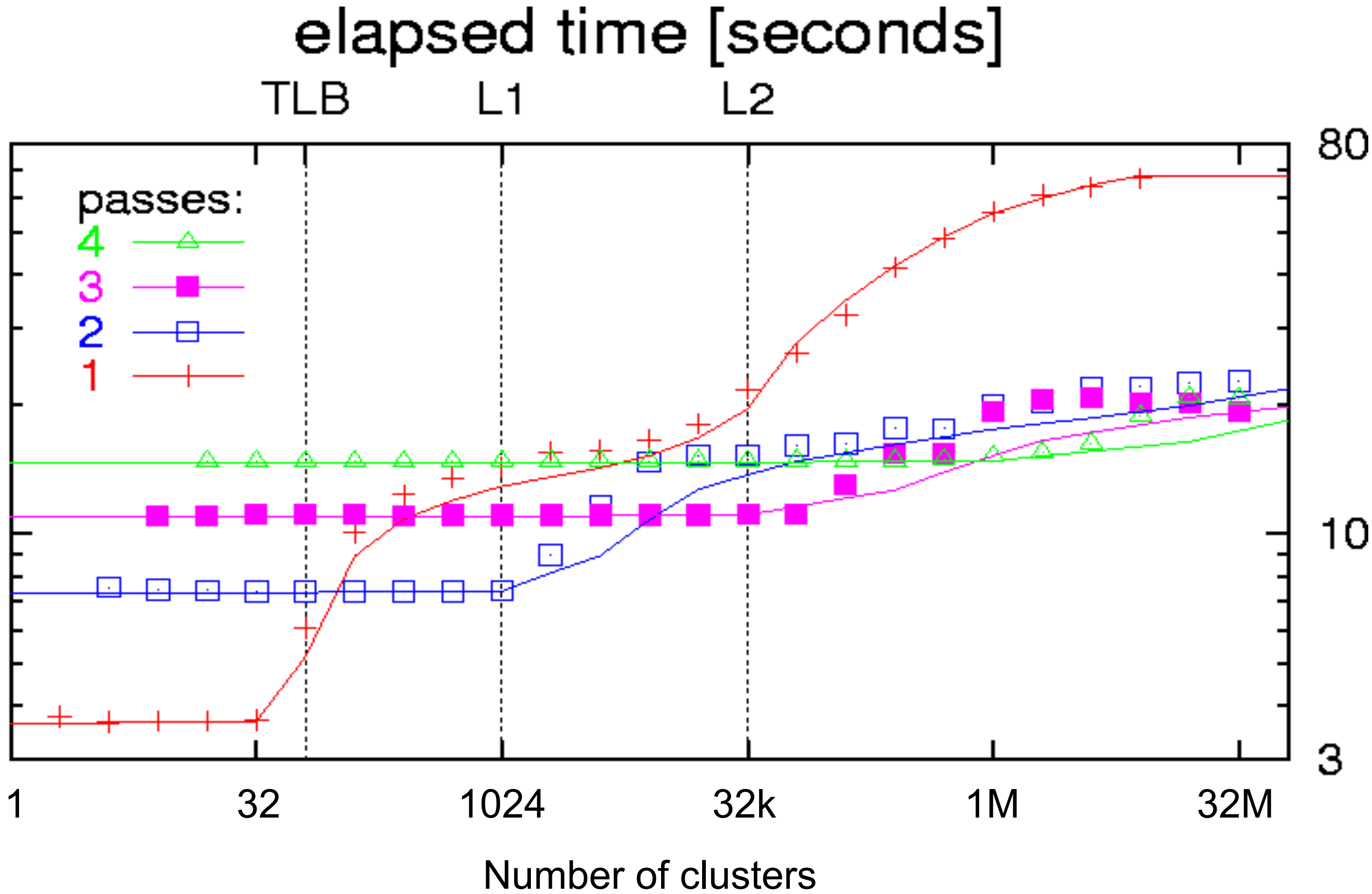
Algorithm Evolution: Multi-Pass Clustering

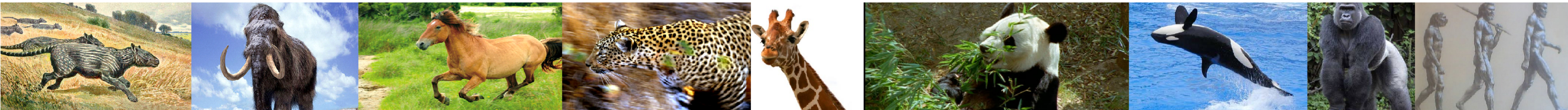
- Limit number of clusters per pass
- Avoid cache / TLB thrashing
- Trade memory cost for CPU cost



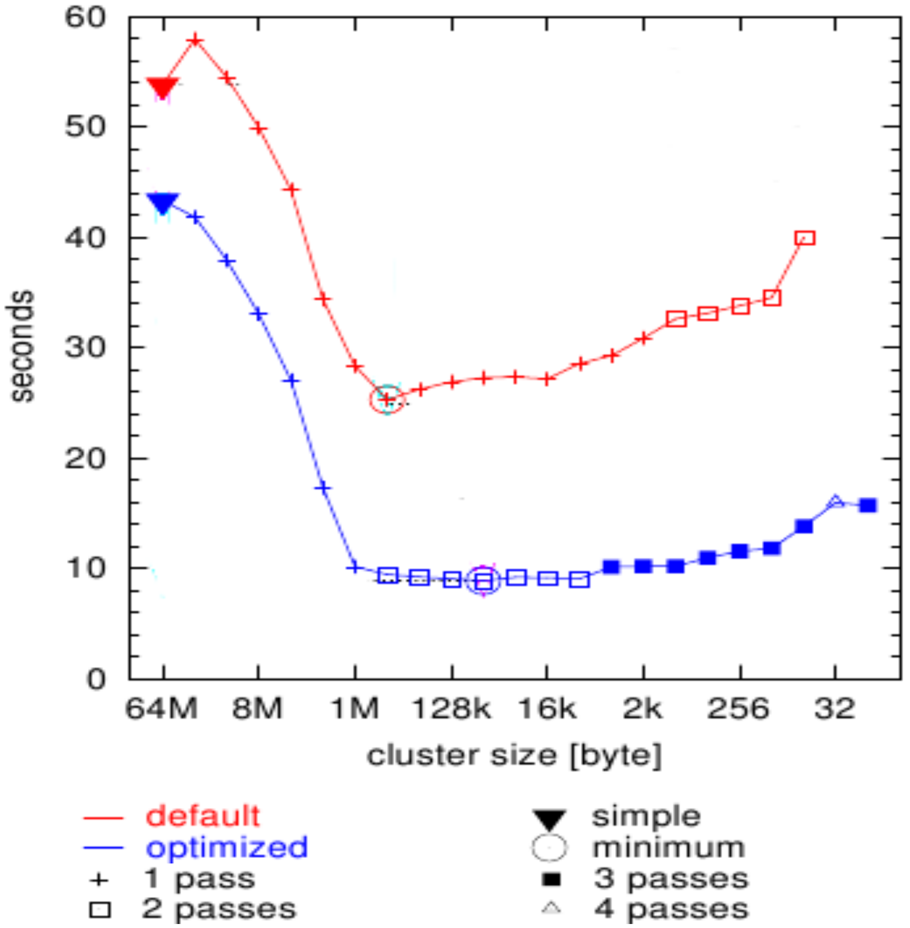


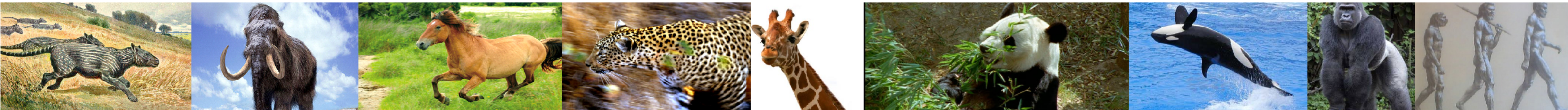
Partitioned Hash-Join: Multi-Pass Clustering





Partitioned Hash-Join





Joins in Column-Stores: Handling Payload

<u>Problem:</u>

- Join result: pairs of tuple IDs; *Out-of order*
- => random access during projection / tuple-reconstruction

<u>Solutions:</u>

- Jive-Join (Li, Ross; VLDB-Journal 1998)
- Flash-Join (Tsirogiannis, Harizopoulos, Shah, Wiener, Graefe; SIGMOD 2009)
- Radix-Decluster (Boncz, Manegold, Kersten; VLDB 2004)
- (Sideways Cracking (Idreos, Kersten, Manegold; SIGMOD 2009))

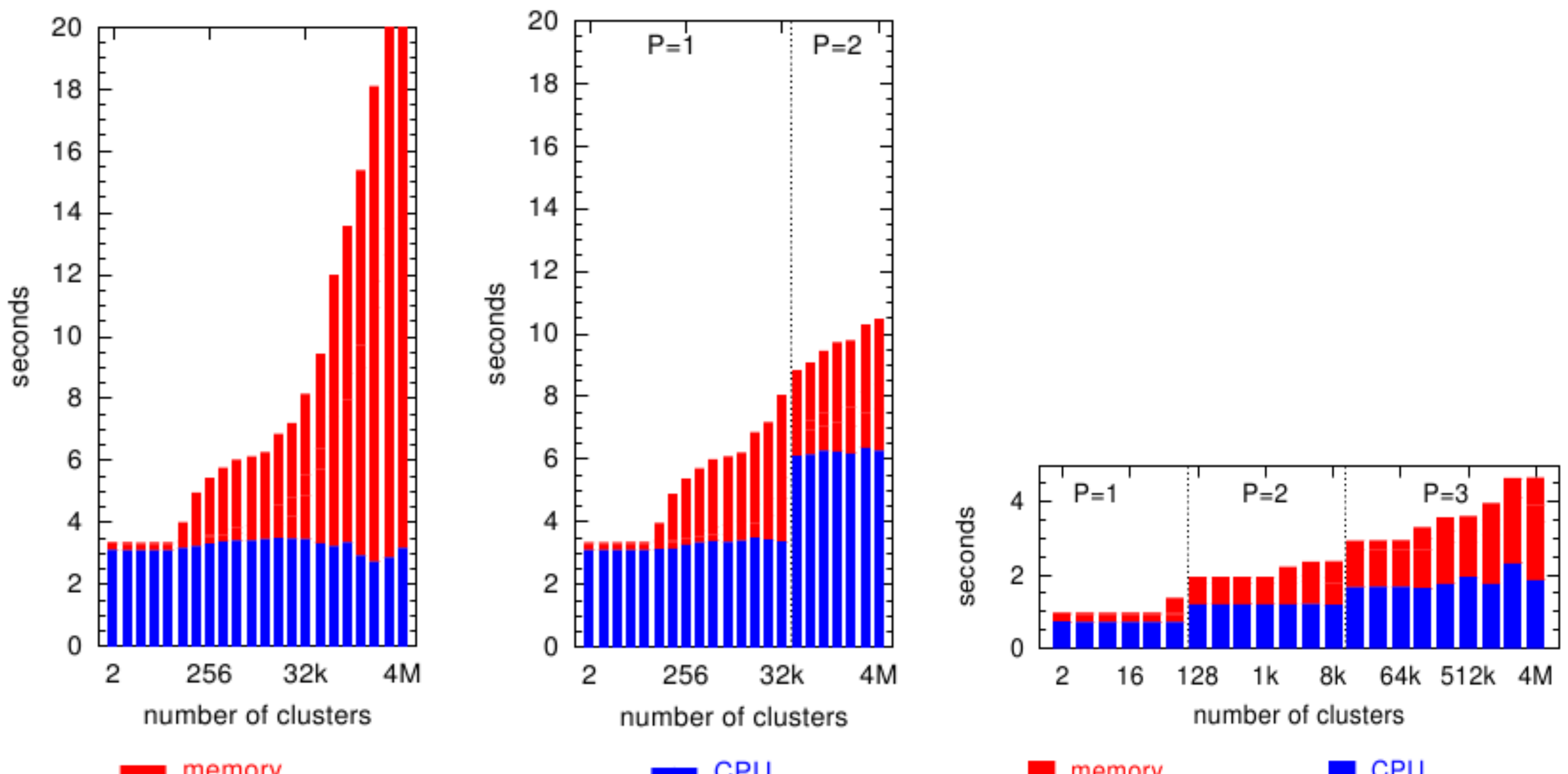
=> post-projection / late materialization

Algorithm Evolution: Multi-pass Clustering

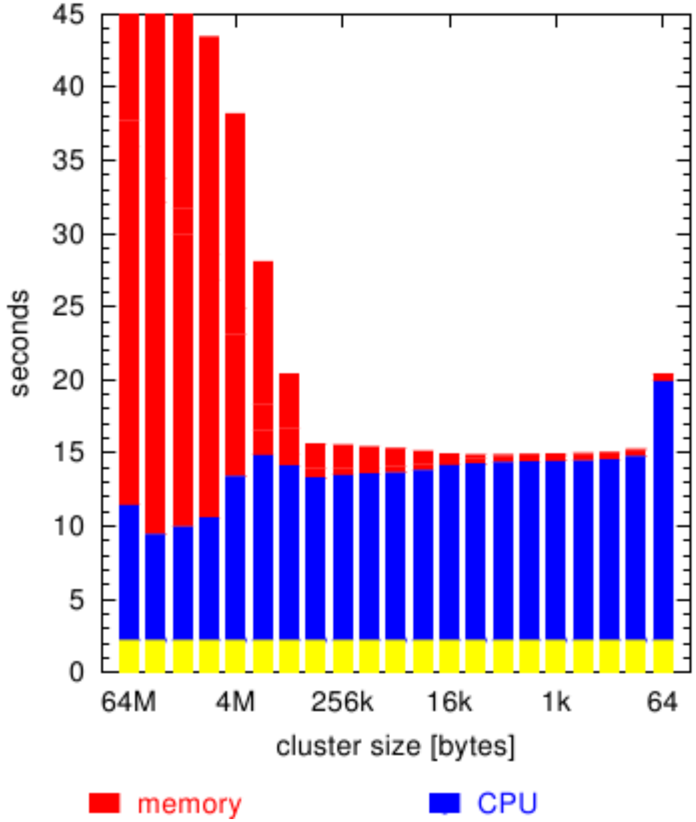
1 pass

P passes

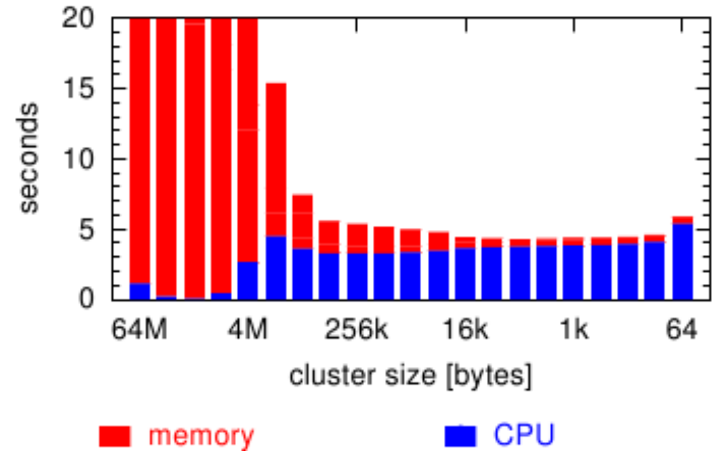
P passes, CPU optimized

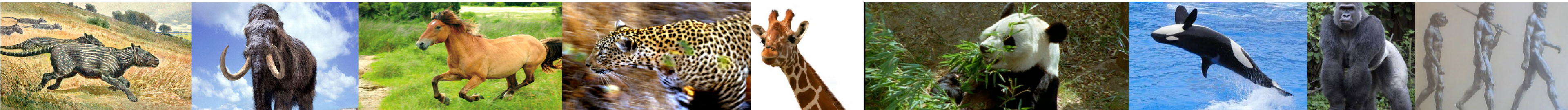


Algorithm Evolution: Partitioned Hash-Join



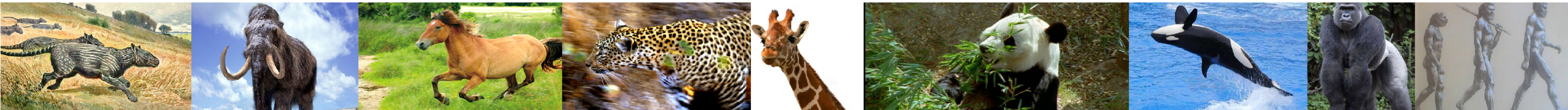
CPU optimized





Cost Model Evolution: Data Access

- Total data access cost is sum over all cache/memory levels
- Cost per level is number of cache misses scored by latency
- Simple tool to measure latency per cache level ("*The Calibrator*")
- few simple *basic access patterns* "*sequential*", "*random*", ...
- *compound access patterns:* combinations of basic access patterns
- *basic cost functions:* estimate number of cache misses of basic access patterns
- Rules how to create *compound cost functions* using basic cost functions
- Describe data access of algorithms using access patterns

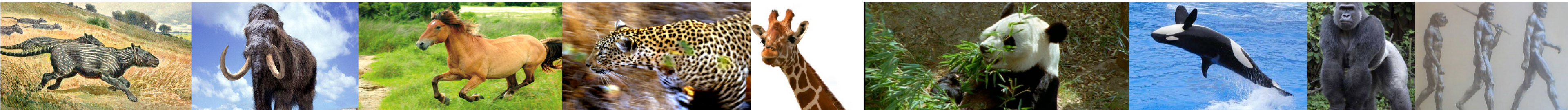


The Bigger Picture:
Evolving Columnar Database Architecture

Stefan Manegold (manegold@cwi.nl)

Peter Boncz (boncz@cwi.nl)

Martin Kersten (mk@cwi.nl)



RISC Relational Algebra

CPU ☺? Give it "nice" code !

- few dependencies (control,data)
- CPU gets out-of-order execution
- compiler can e.g. generate SIMD

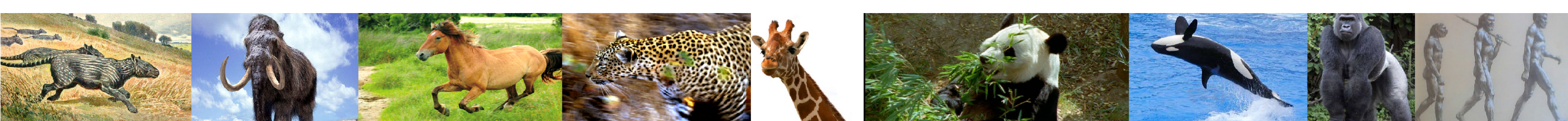
One loop for an entire column

- no per-tuple interpretation
- arrays: no record navigation
- better instruction cache locality

```
{
    for(i=0; i<n; i++)
        res[i] = col[i] - val;
}
```

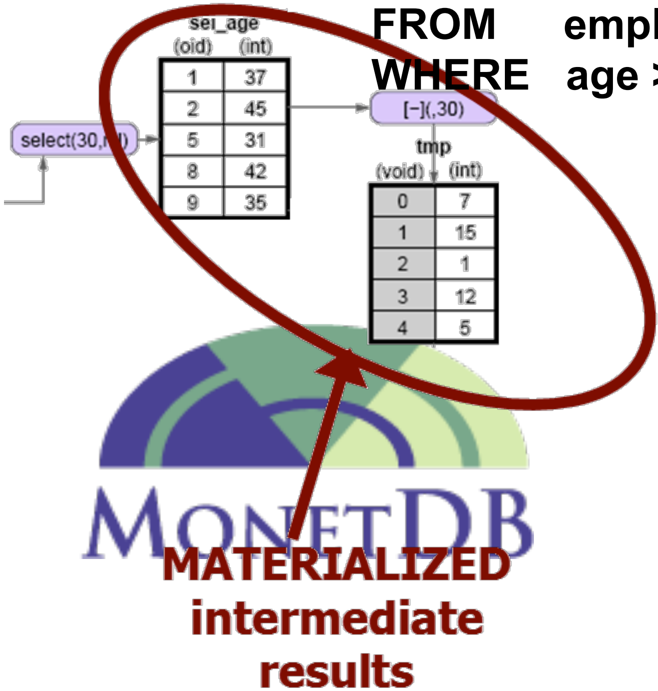
Simple, hard-coded semantics in operators

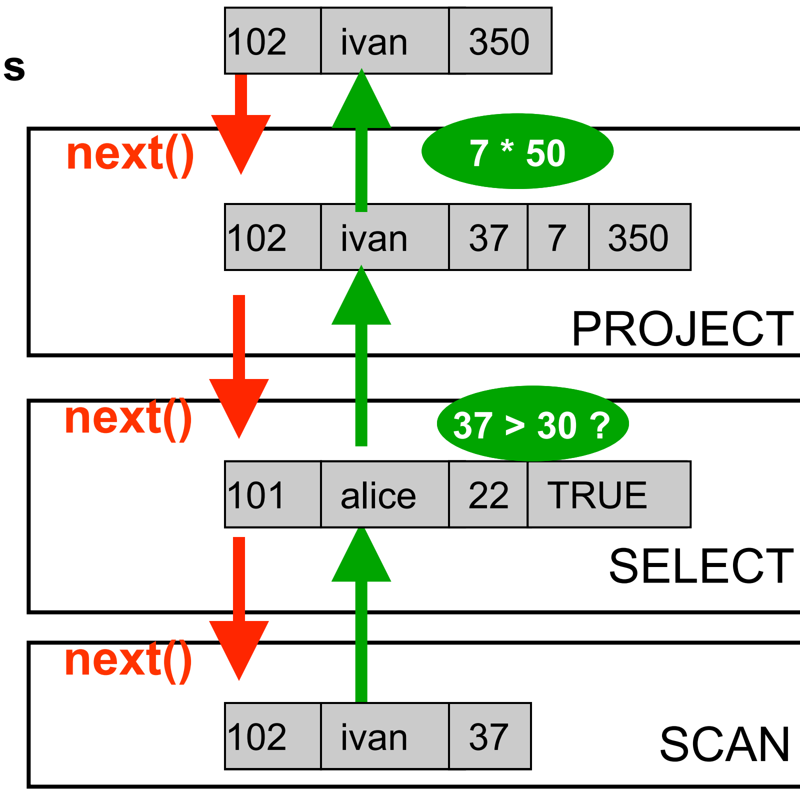
MATERIALIZED intermediate results

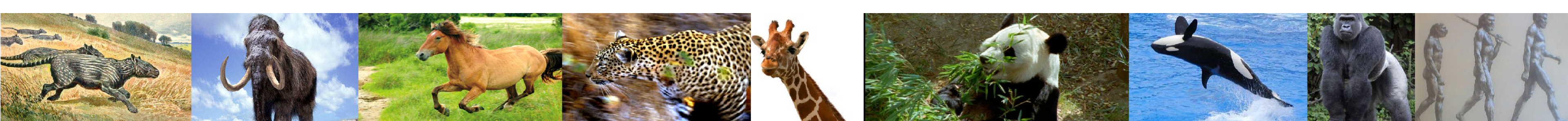


Materialization vs Pipelining

```
SELECT   id, name
         (age-30)*50 AS bonus
FROM     employee
WHERE    age > 30
```

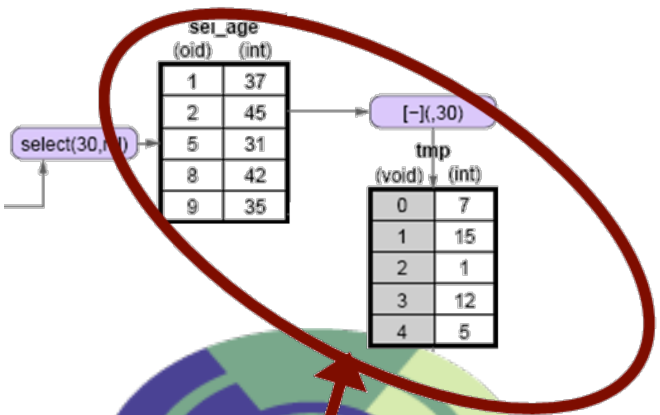




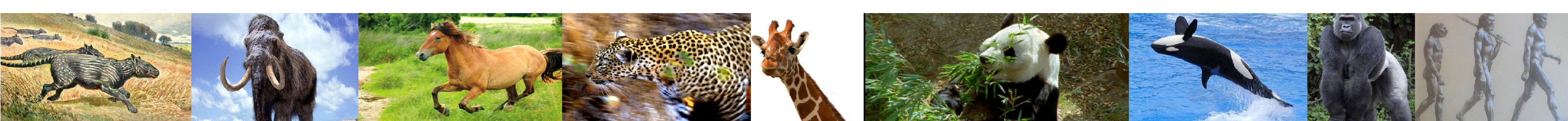


MonetDB spin-off: vectorwise

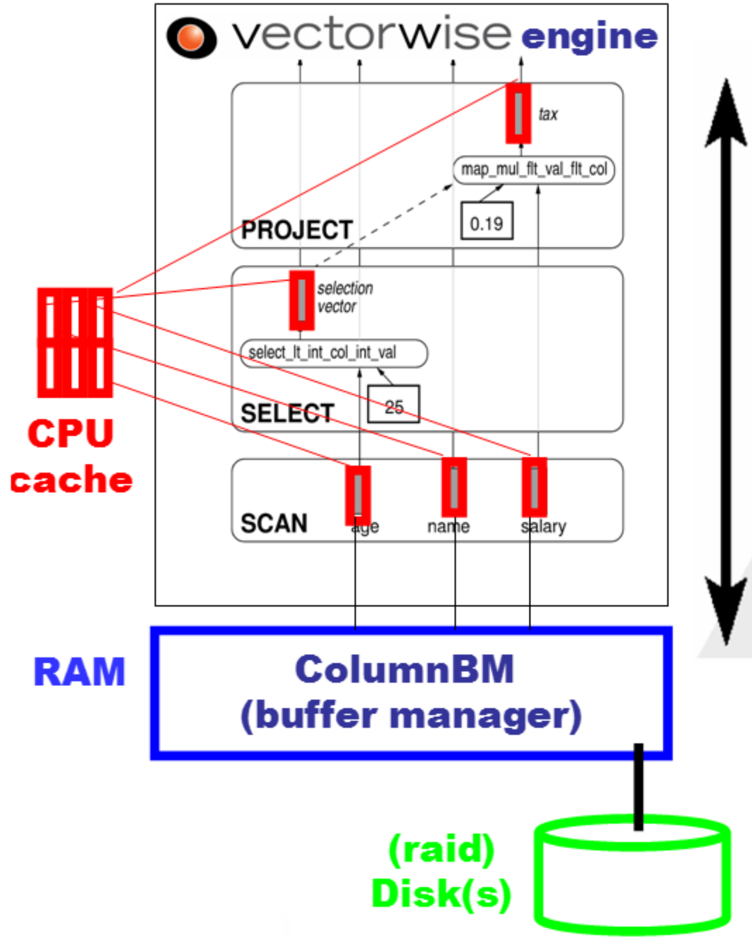
Materialization vs Pipelining

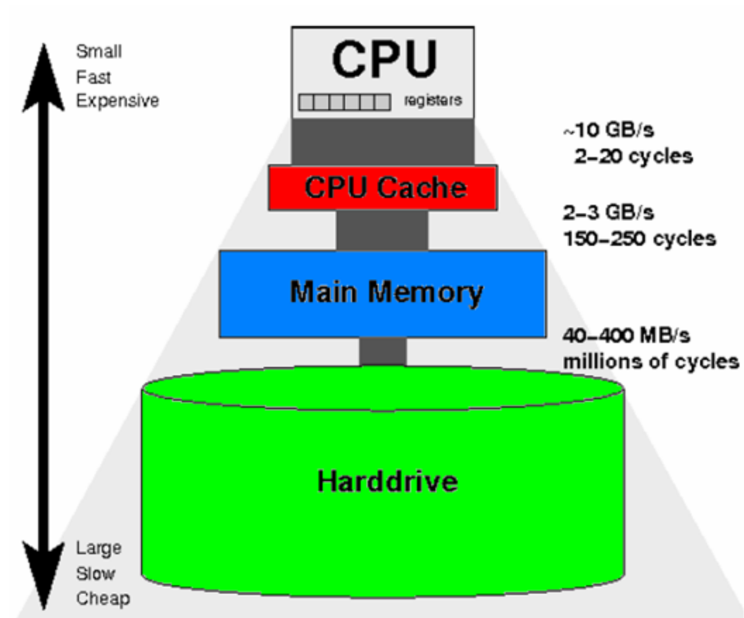


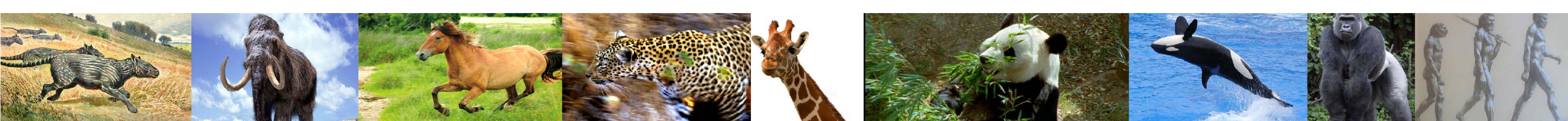
MATERIALIZED intermediate results



vectorwise Memory Hierarchy





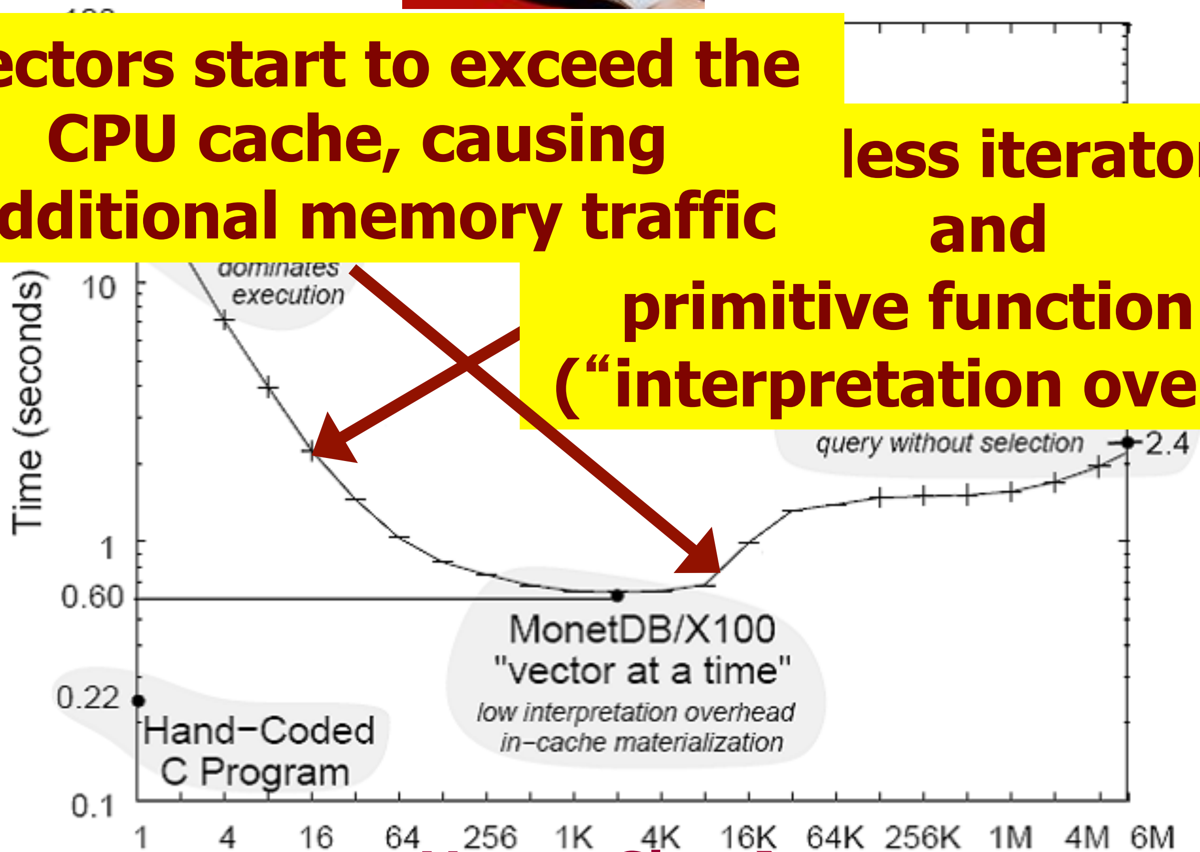


The optimal  diet?

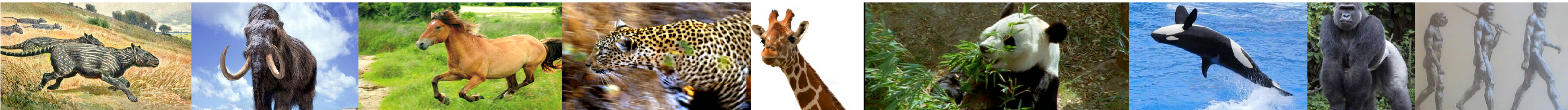
Vectors start to exceed the CPU cache, causing additional memory traffic

less iterator.next() and primitive function calls (“interpretation overhead”)

TPCH Q1



Vector Size →



More on vectorwise

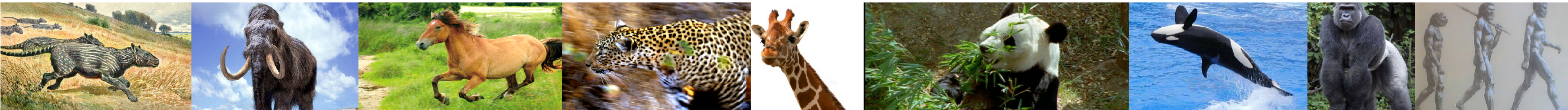
And much more..

14:00-17:00 Tutorial “Column-Oriented DB Systems”

+ Daniel Abadi (Yale) and Stavros Harizopoulos (HP Labs)

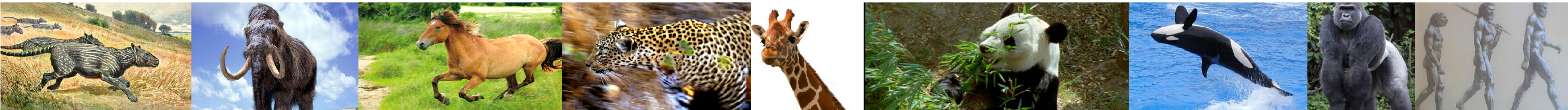
MonetDB Highlights

- **Architecture-Conscious Query Processing**
  - Data layout, algorithms, cost models
- **Multi-Model: ODMG, SQL, XQuery, .. SPARQL**
  - Columns as the building block for complex data structures
- **RISC Relational Algebra (vs CISC)**
  - Faster through simplicity: no tuple expression interpreter
- **Decoupling of Transactions from Execution/Buffering**
  - ACID, but not ARIES..   Pay as you need transaction overhead.
  - differential, lazy, optimistic, snapshot
- **Run-Time Indexing and Query Optimization**
  - Extensible Optimizer Framework
  - cracking, recycling, sampling-based runtime optimization

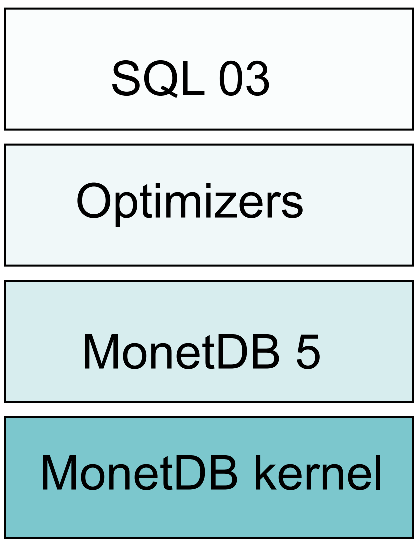


MonetDB vs Traditional Architecture

- Architecture-Conscious Query Processing
  - vs Magnetic disk I/O conscious processing
- Multi-Model: ODMG, SQL, XQuery, .. SPARQL
  - vs Relational with Bolt-on Subsystems
- RISC Relational Algebra
  - vs Tuple-at-a-time Iterator Model
- Decoupling of Transactions from Execution/Buffering
  - vs ARIES integrated into Execution/Buffering/Indexing
- Run-Time Indexing and Query Optimization
  - vs Static DBA/Workload-driven Optimization & Indexing



The MonetDB Software Stack

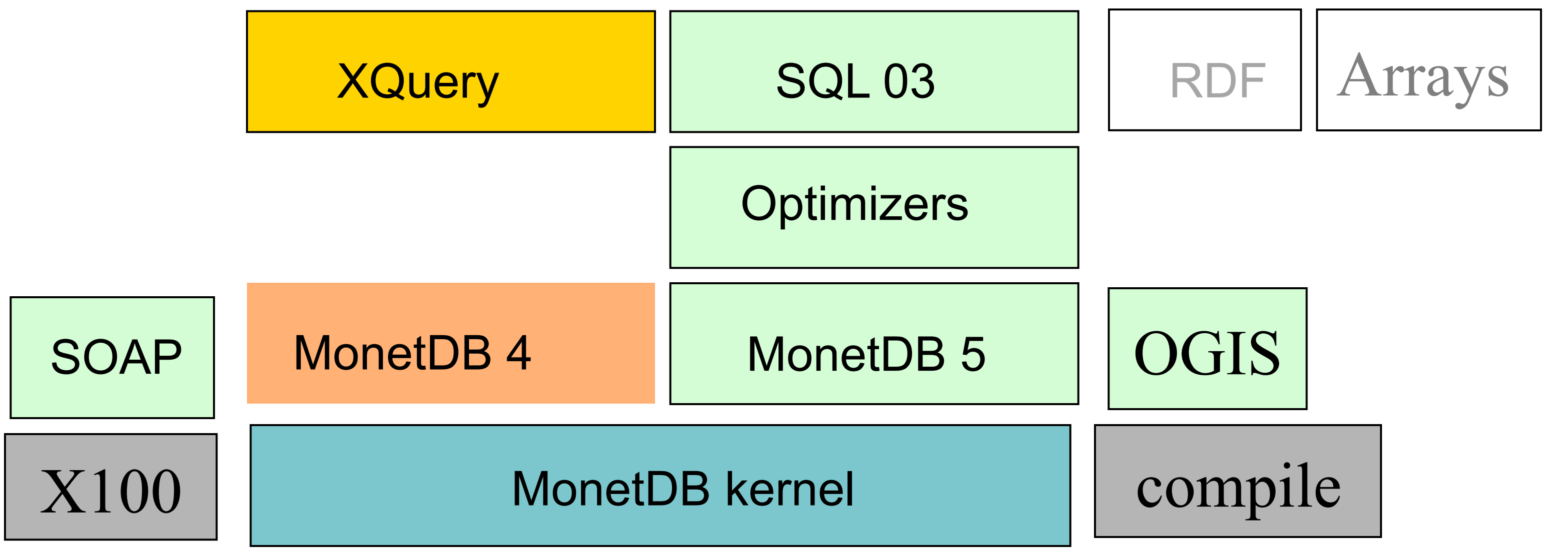


Orthogonal extension of SQL03

Clear computational semantics

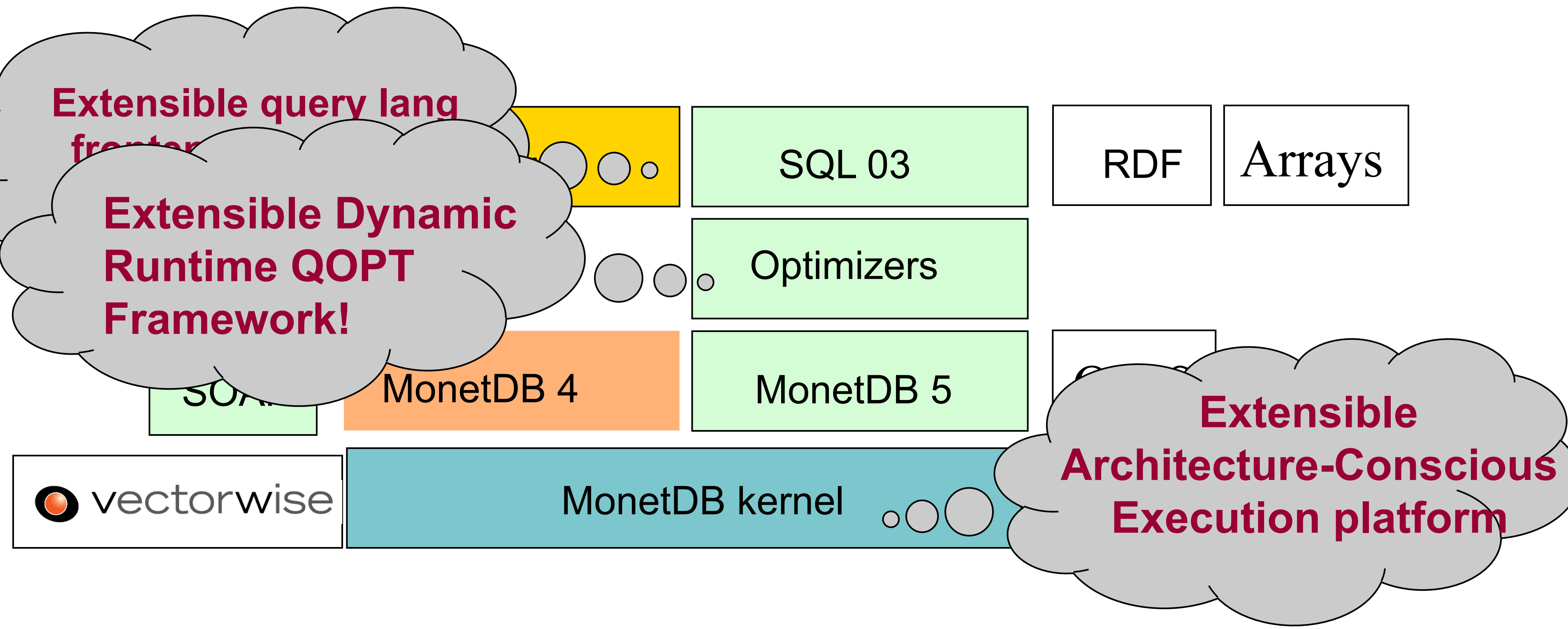
Minimal extension to MonetDB

The MonetDB Software Stack



CWI

The MonetDB Software Stack



CWI

Farming new species

MonetDB

vectorwise

Cyclotron
Romulo Gonçalves

Data cell
Erietta Liarou

Sky server
Milena Ivanova

Armada
Fabian Groffen

Cracking
Stratos Idreos

XRPC
Jenny Zhang

XML pattern search
Nan Tang

RDF Graphs
Lefteris Sidirourgos

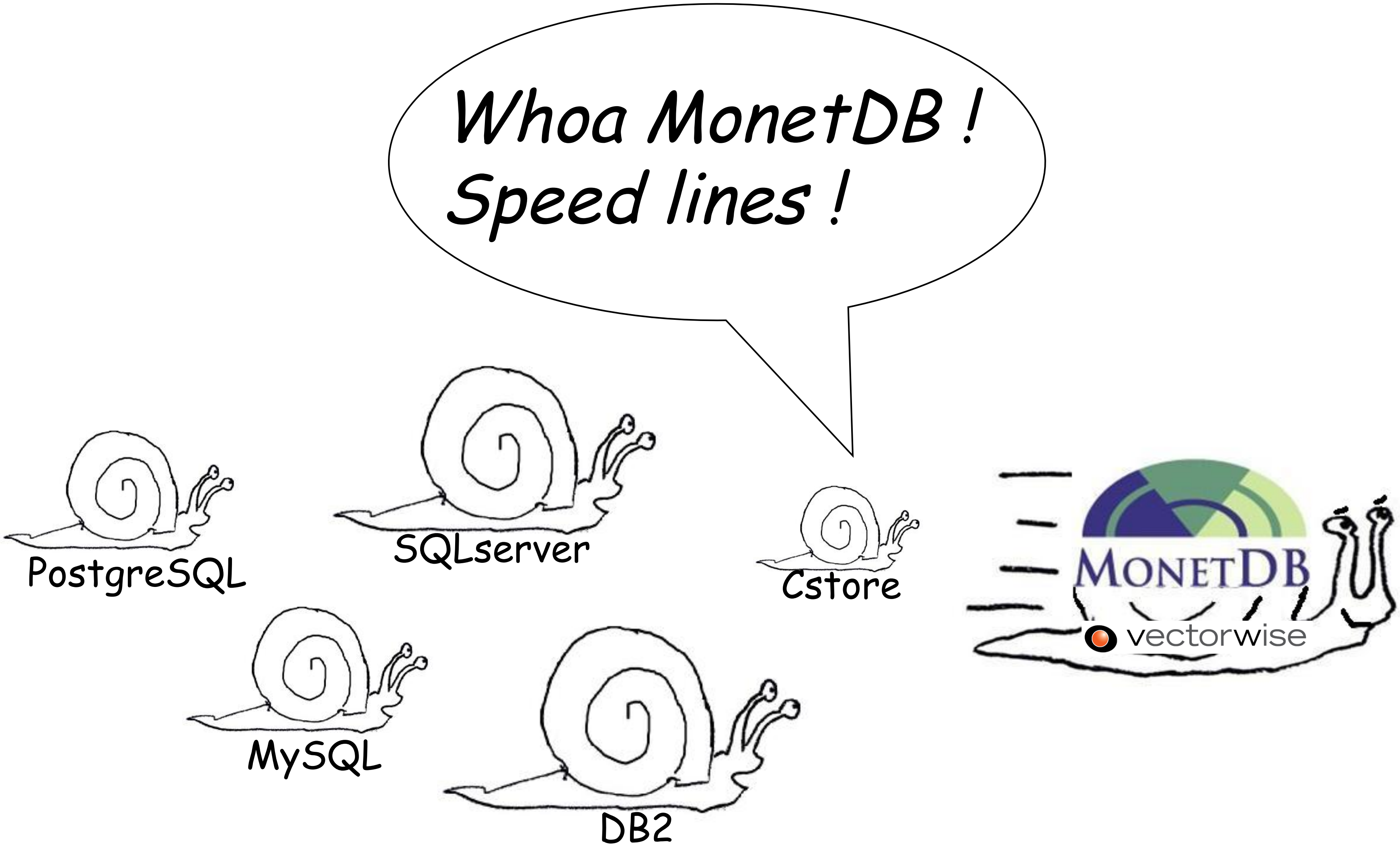
Acknowledgements

Martin Kersten
Peter Boncz
Niels Nes
Stefan Manegold
Fabian Groffen
Sjoerd Mullender
Steffen Goeldner
Arjen de Vries
Menzo Windhouwer
Tim Ruhl
Romulo Goncalves

Alex van Ballegooij
Johan List
Georgina Ramirez
Marcin Zukowski
Roberto Cornacchia
Sandor Heman
Torsten Grust
Jens Teubner
Maurice van Keulen
Jan Flokstra
Milena Ivanova
Lefteris Sidirourgos

Jan Rittinger
Wouter Alink
Jennie Zhang
Stratos Idreos
Erietta Liarou
Lefteris Sidirourgos
Florian Waas
Albrecht Schmidt
Jonas Karlsson
Martin van Dinther
Peter Bosch
Carel van den Berg
Wilco Quak

CWI


Whoa MonetDB !
Speed lines !
SQLserver
PostgreSQL
Cstore
MonetDB
vectorwise
MySQL
DB2