NEW ARCHITECTURES FOR DATABASES

Gustavo Alonso
Systems Group
Dept. of Computer Science
ETH Zürich, Switzerland

Systems Group = www.systems.ethz.ch

Enterprise Computing Center = www.ecc.ethz.ch

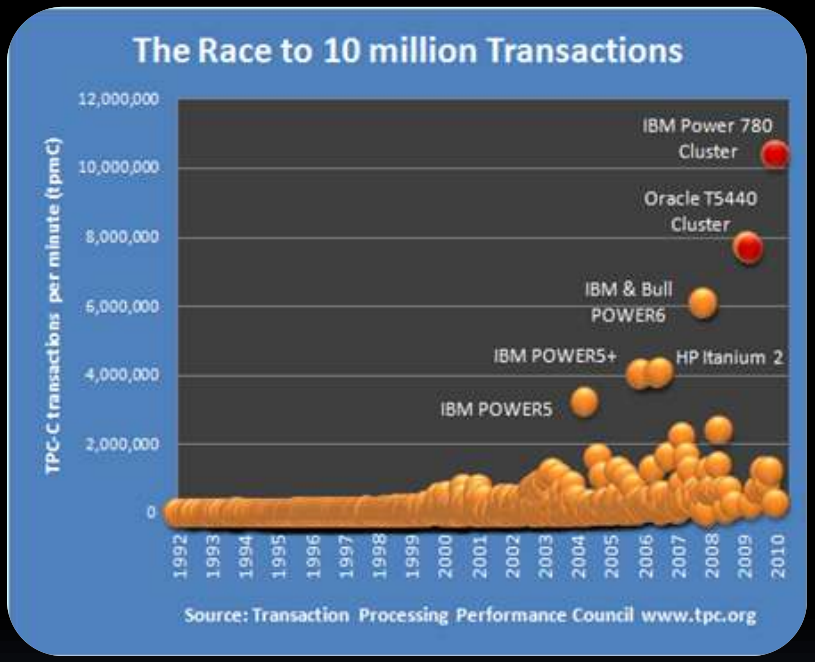
Systems@ETH Zürich

DATABASES:
The world is changing

Differentiate Relational Model from implementations of the relational model

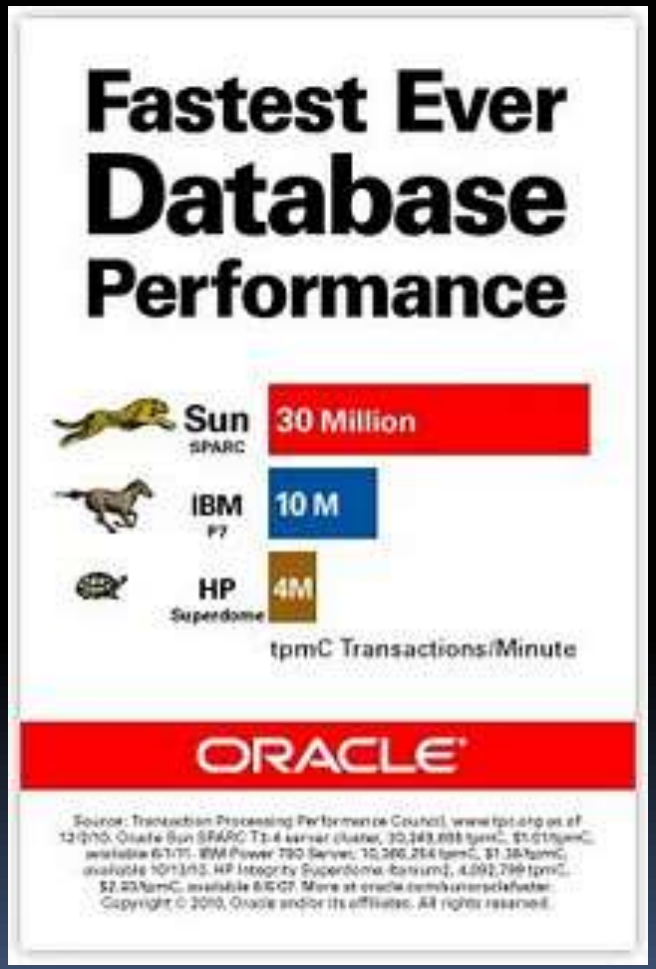
Relational Database technology

- Powerful relational database engines
  - Transactional guarantees
  - Recovery guarantees
  - Existing investments
  - Constant need for larger and larger deployments to cope with modern loads



Systems@ETH zurich

The problem is somewhere else



- **Peak load provisioning** = license a huge database engine to cope with peak loads
- **Scalability** = relational engines scale but only at an exponential cost in licenses, hardware, and administrative overhead
- **Complexity** = Huge costs for tuning, maintenance and administration
- **Price** = exorbitant costs for the performance

Systems@ETH zurich

Data management today

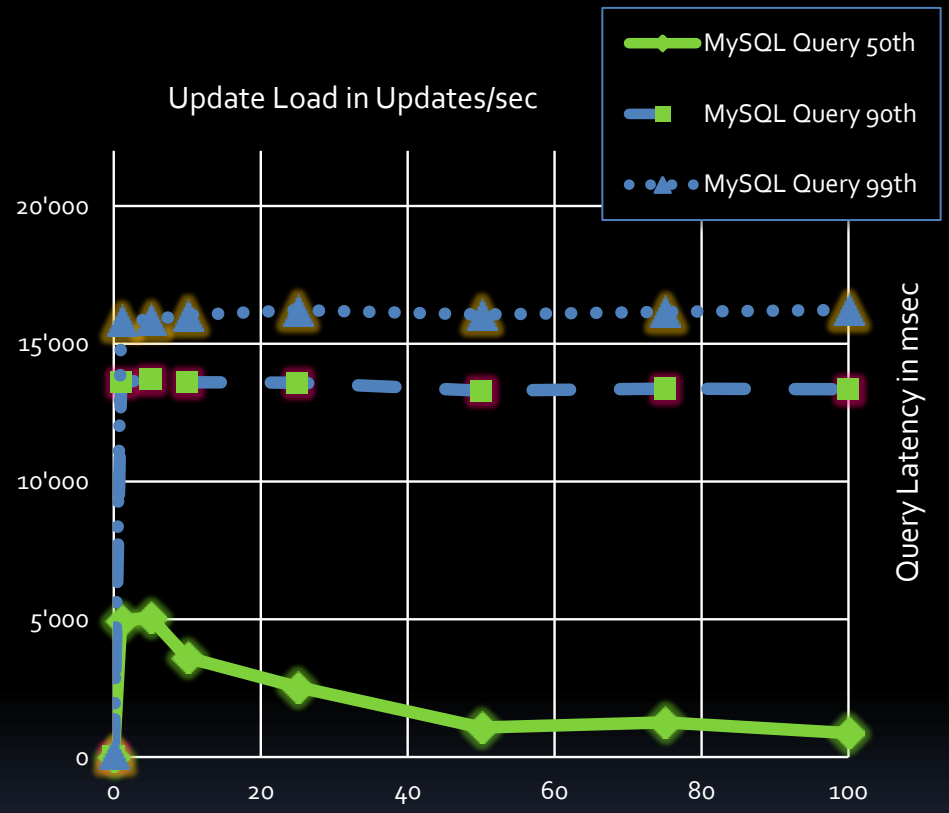
- One size does not fit all
- Reduced capital expense
- Multitude of use cases
  - Data types (e.g., graphs)
  - Queries (e.g., key value)
  - Hardware
  - Software stacks
- NoSQL is only a very small part of the picture

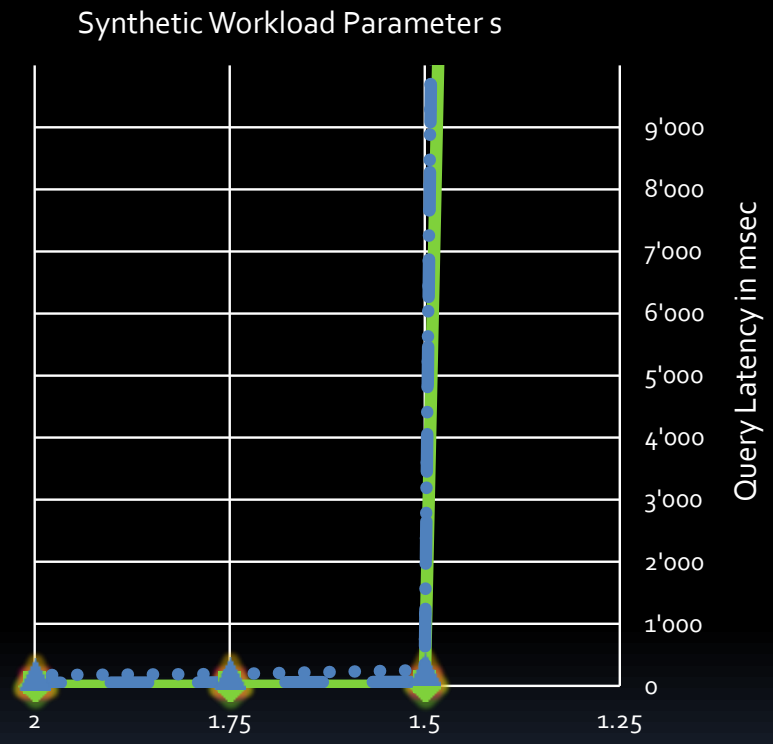
AMADEUS:
an interesting use case

Amadeus Workload

- Passenger-Booking Database
  - ~ 600 GB of raw data (two years of bookings)
  - single table, denormalized
  - ~ 50 attributes: flight-no, name, date, …, many flags
- Query Workload
  - up to 4000 queries / second
  - latency guarantees: 2 seconds
  - today: only pre-canned queries allowed
- Update Workload
  - avg. 600 updates per second (1 update per GB per sec)
  - peak of 12000 updates per second
  - data freshness guarantee: 2 seconds
- Problems with State-of-the Art
  - Simple queries work only because of mat. views
    - multi-month project to implement new query / process
  - Complex queries do not work at all

Traditional engines break down





- Performance depends on workload parameters
  - changes in load (updates, columns accessed) -> huge variance
  - Unpredictable performance, impossible to tune correctly

Systems@ETH zürich

Many problems

- Exhaustive benchmarking and analysis shows:
  - Lack of scalability with number of cores
  - Problems with I/O with number of cores
  - Load interaction problems
  - Unpredictable performance
  - Increasingly expensive tuning

Amadeus requirements

- Predictable (= constant) Performance
- Meet SLAs on latency and data freshness
- Affordable Cost
  - compared to mainframe / current license
- Maintain Consistency
  - monotonic reads (ACID not needed)
- Suitable for modern hardware
  - main-memory, NUMA, large data centers

SWISSBOX

The SwissBox project

- Build an open source data appliance
  - Hardware
  - Software

DATA APPLIANCE

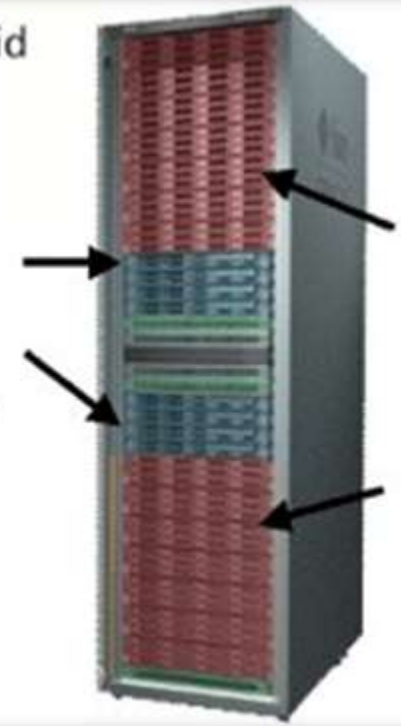
- Database in a box
  - Funny database
  - Funny box

ORACLE EXADATA

Oracle Database Server Grid
- 8 compute servers
- 64 Intel Cores
- 578 GB DRAM

InfiniBand Network
- 40 Gbit/sec. unified server and storage network
- Fault Tolerant

Enterprise Linux

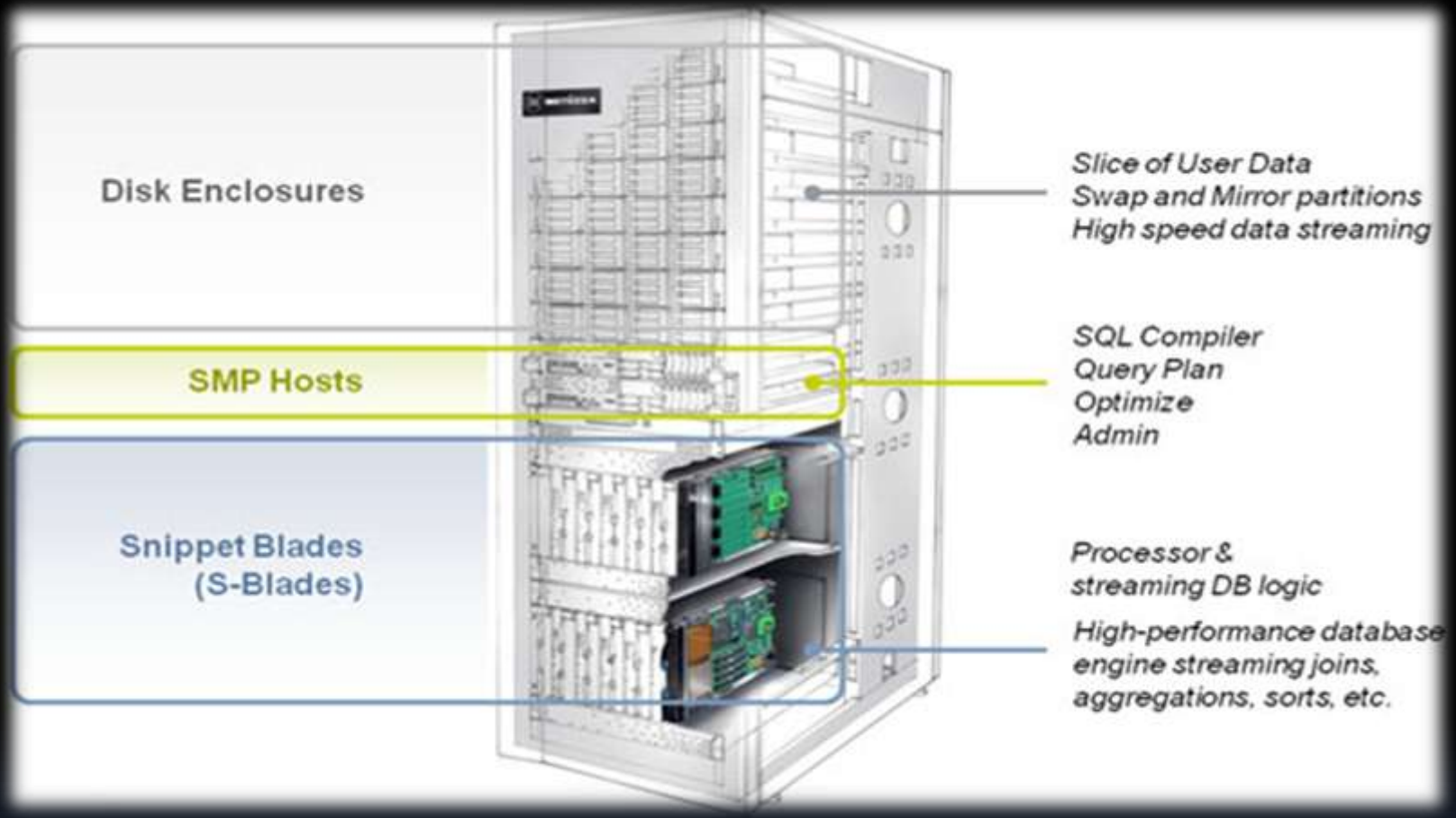


Exadata Storage Server Grid
- 14 storage servers
  168 Platten/112 Intel Cores
- 100 TB raw SAS disk storage
  or
  338 TB raw SATA disk storage
- 5 TB flash storage!

21 GB/sec. IO-Datendurchsatz

- Intelligent storage manager
- Massive caching
- RAC based architecture
- Fast network interconnect

NETEZZA (IBM) TWINFIN



- No storage manager
- Distributed disks (per node)
- FPGA processing
- No indexing

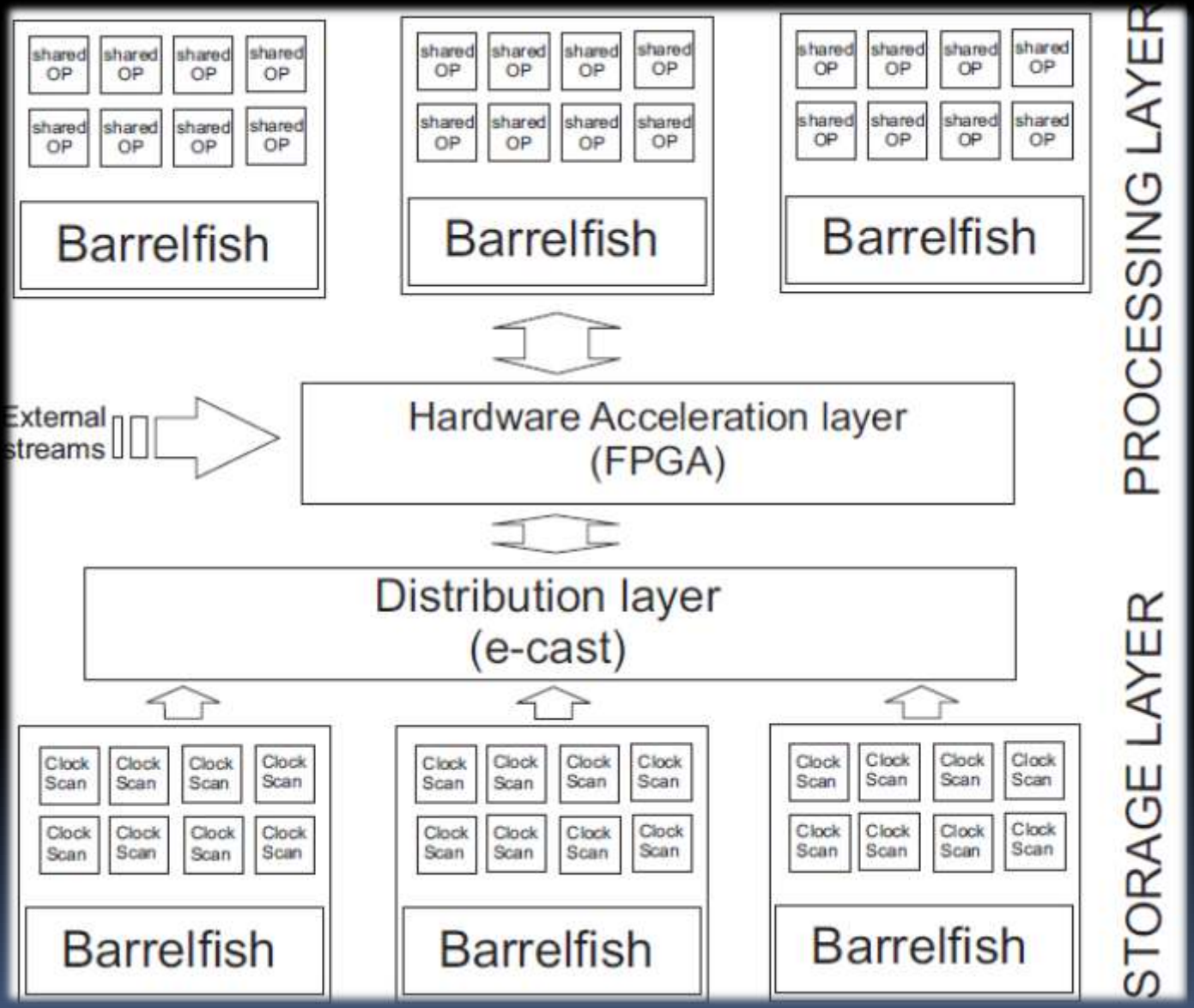
SwissBox: the project

- Great opportunity for research
  - Rethink the entire system software stack
  - Redesign the operating system, database, and storage system architecture
  - Software – hardware co-design

SwissBox: the product

- Direct collaboration and input from industry
- Great demand for tailored systems
  - Big data
  - Highly demanding applications
  - Low power / high efficiency

SWISSBOX



Systems@ETH Zurich

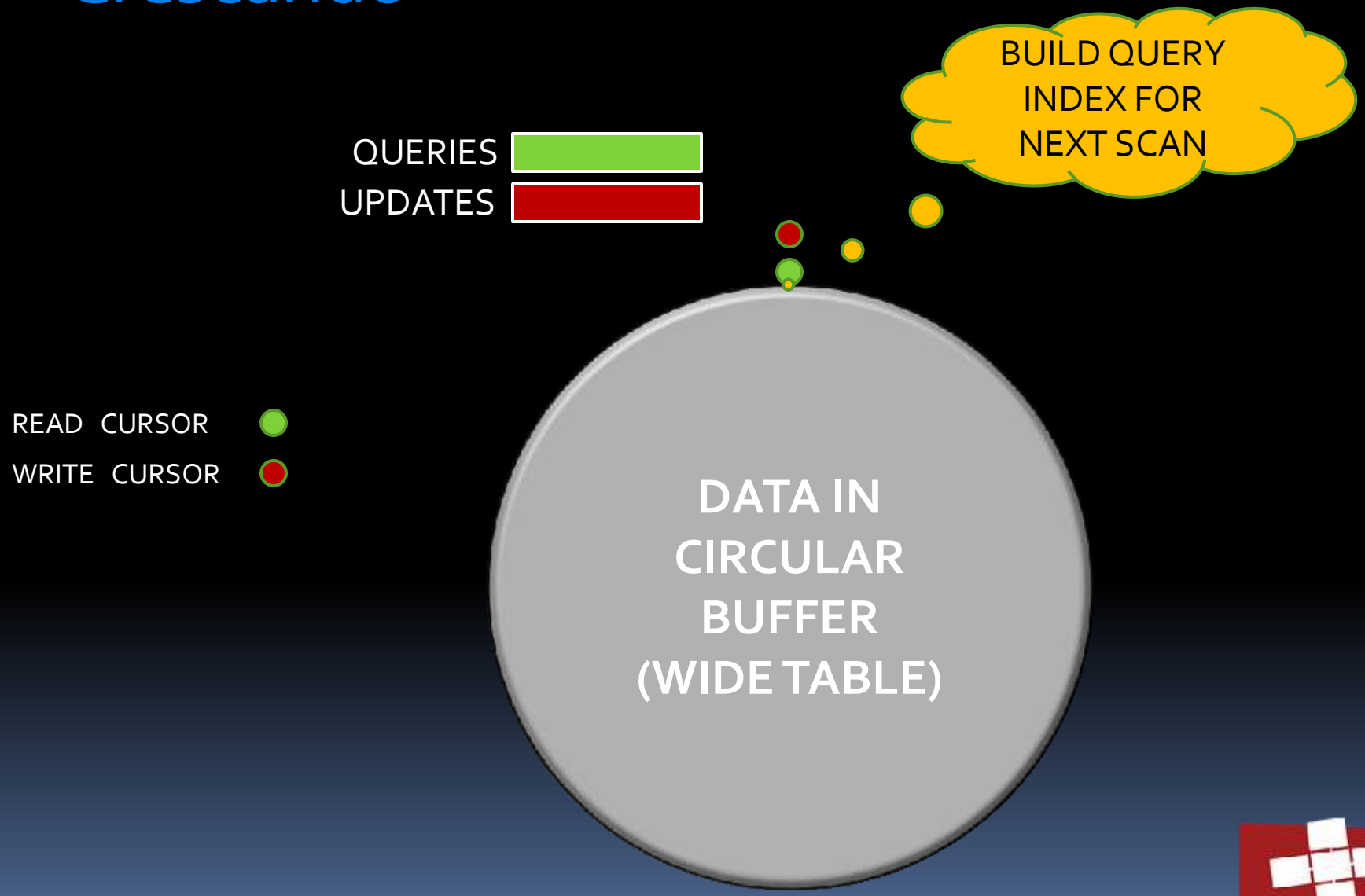
SwissBox: four of many ideas

- Database – Operating System codesign
- Hardware acceleration (FPGAs)
- Main memory, multicore storage engines
- New database engine architectures

SwissBox: Storage engine

- Cluster based – multi core - Main memory
- Shared scans / searches
- Heterogenous engines
  - Crescando: main memory storage manager
  - Key value store
- Interface no longer page/block

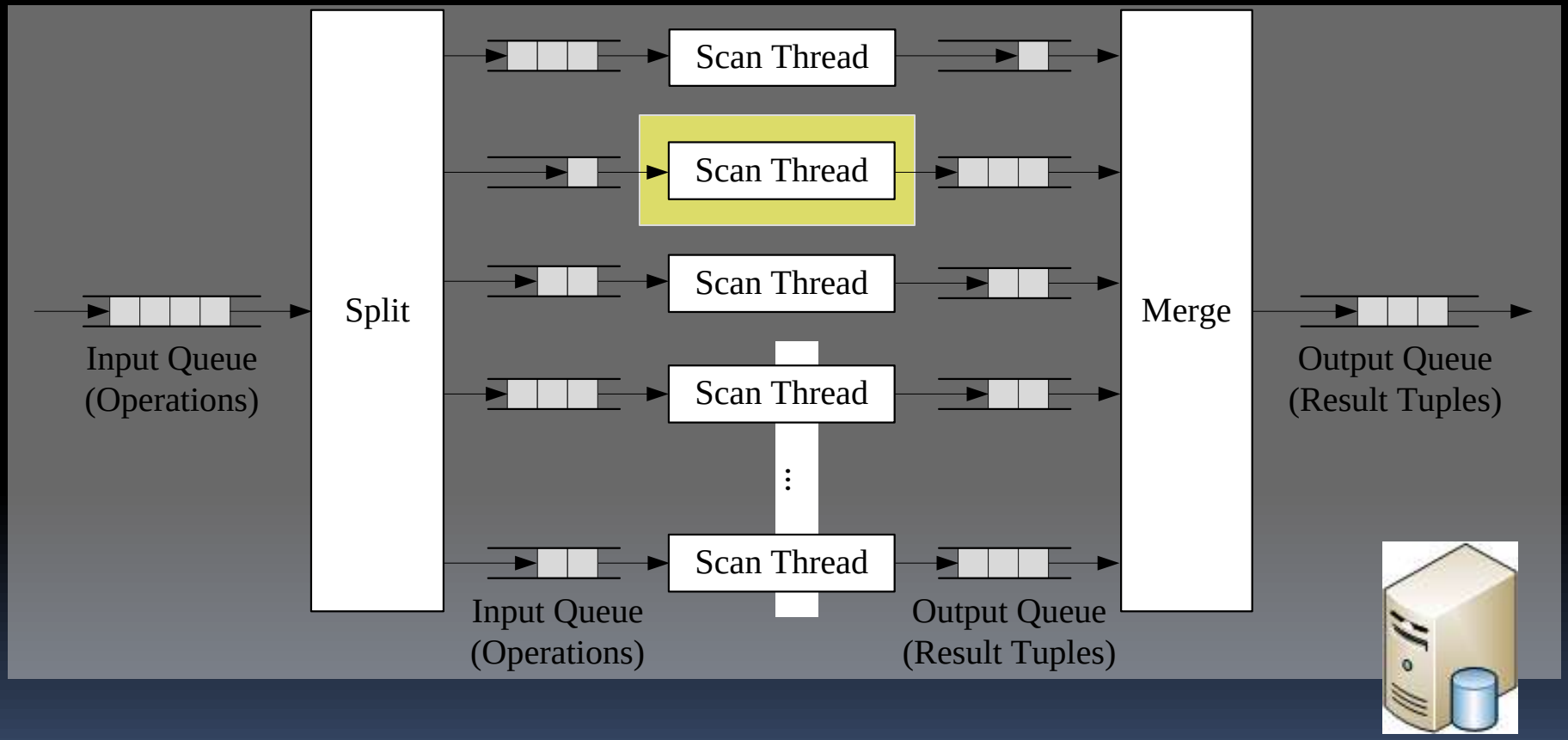
Crescando



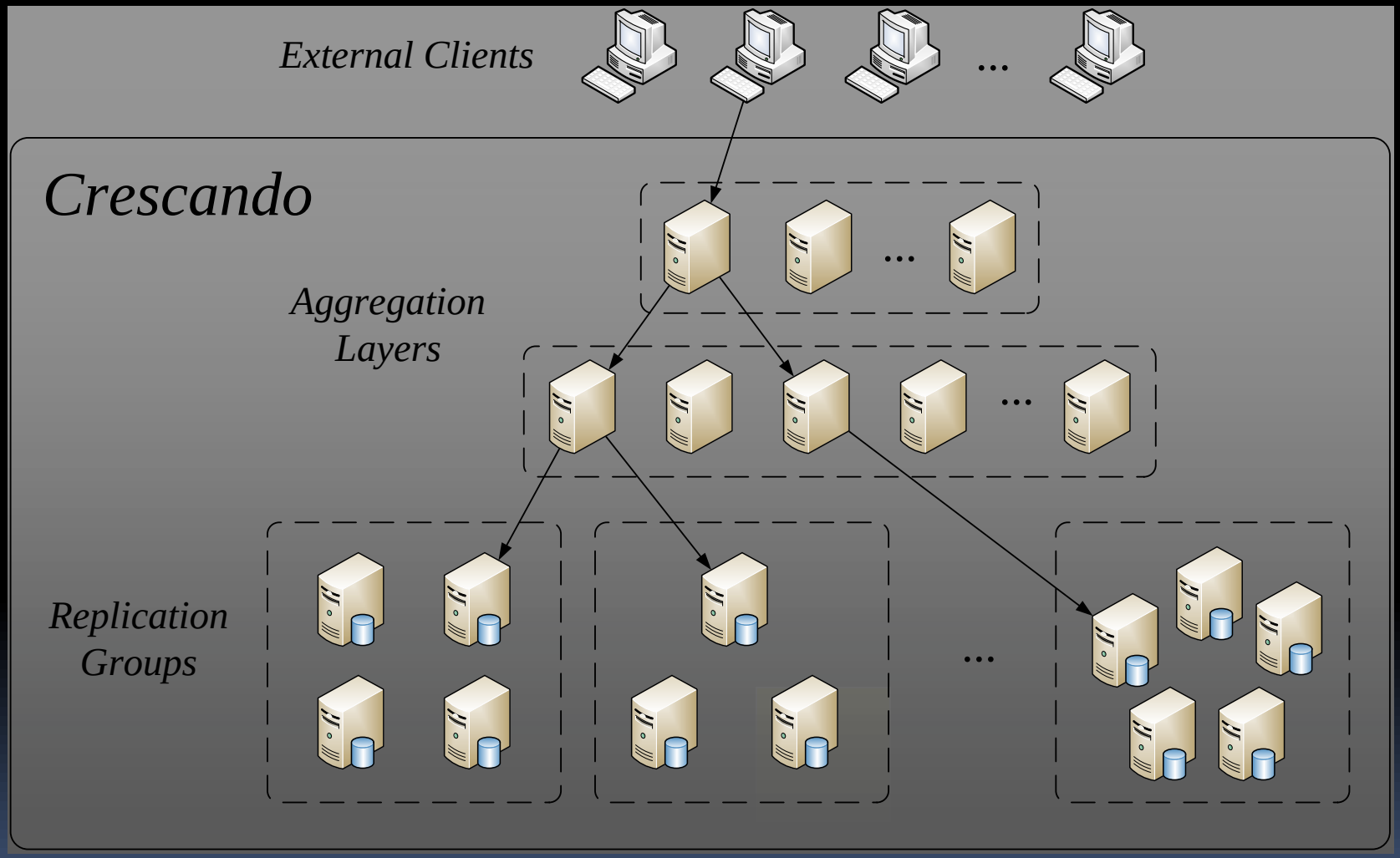
Why?

- Fully predictable performance
  - Accurate analytical model
  - Easily tunable / scalable
- Tolerates high peaks of reads and updates without compromising SLA
- Serves as intelligent storage engine

Crescando on 1 Machine (N Cores)



Crescando in a Data Center (N Machines)



Systems@ETH Zurich

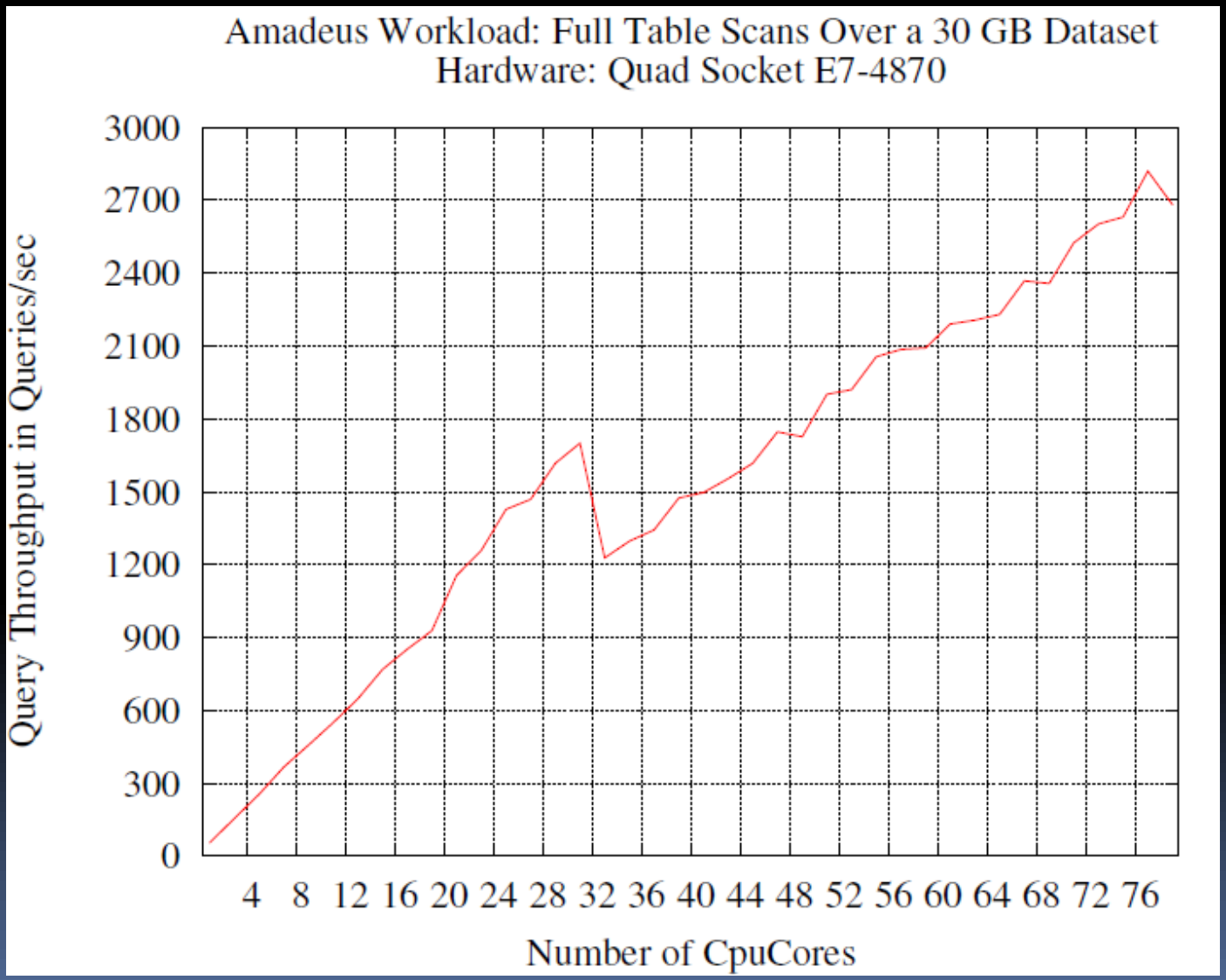
Implementation Details

- Optimization
  - decide for batch of queries which indexes to build
  - runs once every second (must be fast)
- Query + update indexes
  - different indexes for different kinds of predicates
  - e.g., hash tables, R-trees, tries, …
  - must fit in L2 cache (better L1 cache)
- Probe indexes
  - Updates in right order, queries in any order
- Persistence & Recovery
  - Log updates / inserts to disk (not a bottleneck)

What is different?

- No threads (work unit = core)
- No synchronization across work units
- Data partitioned across work units
- Work units across cores and machines
- No indexes on data, no materialized views
- Constant performance
- Performance determined by design
- Dynamic scalability / elasticity

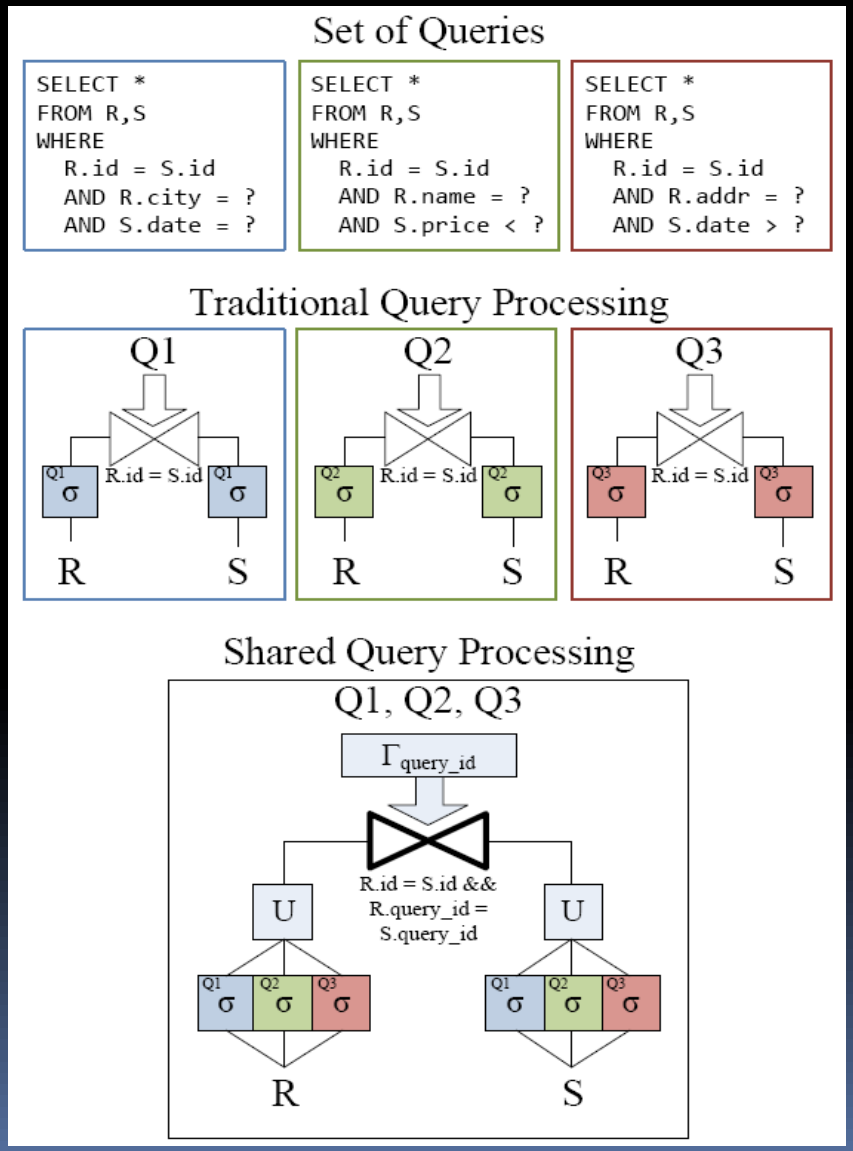
Linear scalability on modern hardware



SwissBox: data processing

- Cluster based – multicore – main memory
- Batch query processing (1000's)
- Shared operators (all operators)
- One single plan for everything
- Fully predictable performance
  - By design
  - Easy tuning

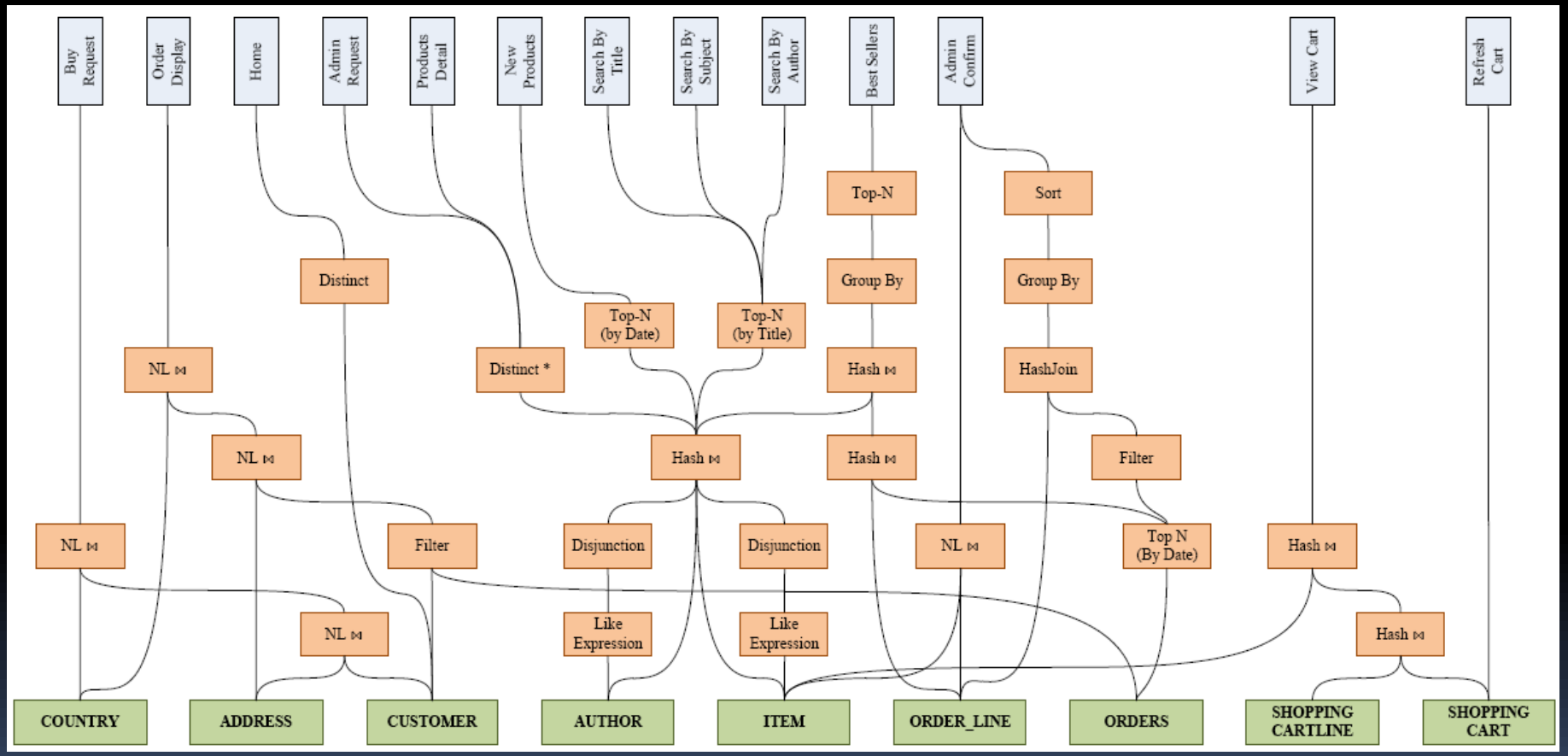
Shared join



- Crescando runs selection and projections in one set of cores
- SharedDB runs joins on the streams from Crescando, thousands of queries at a time

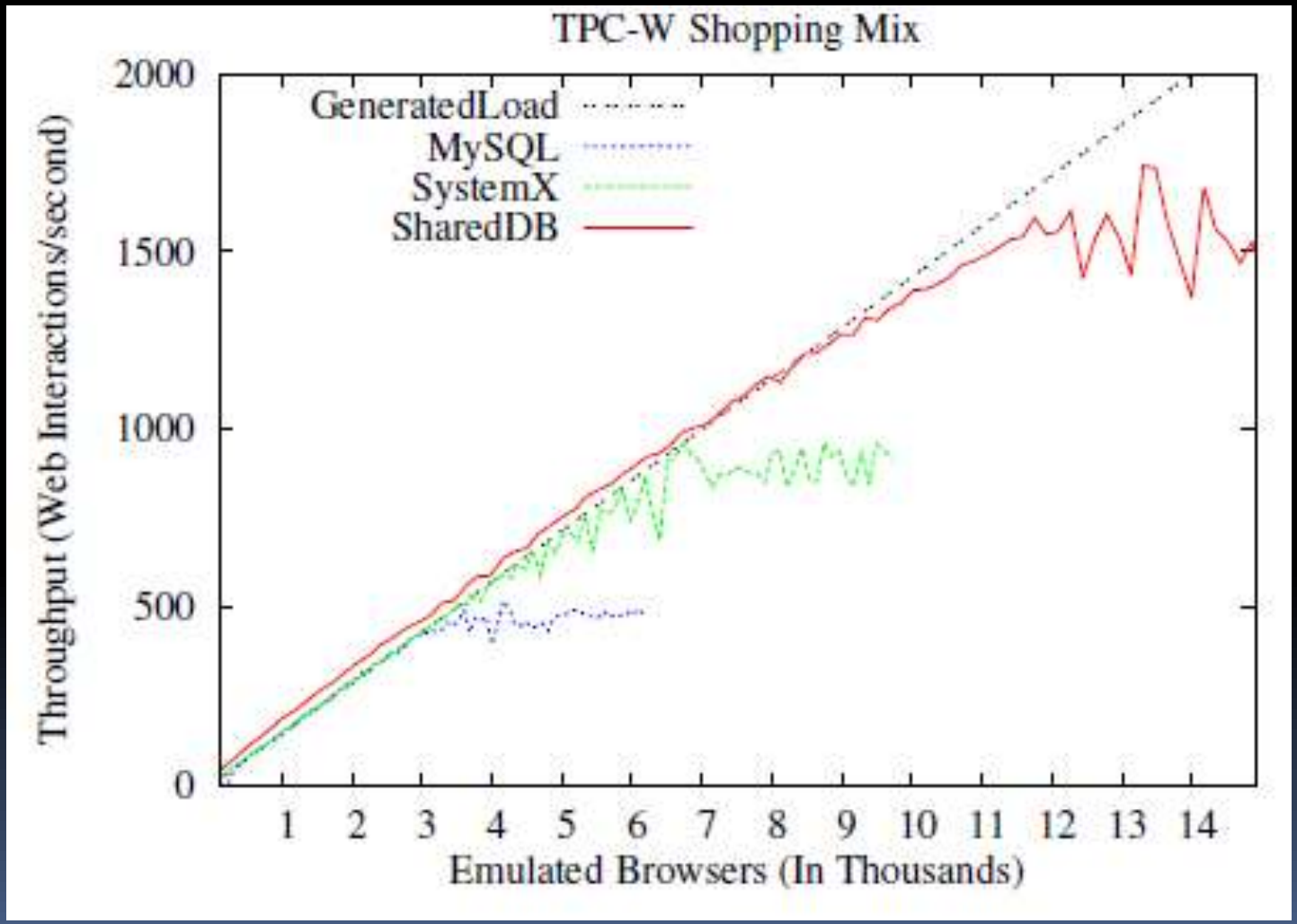
Systems@ETH zürich

Shared DB can run TPC-W (!)

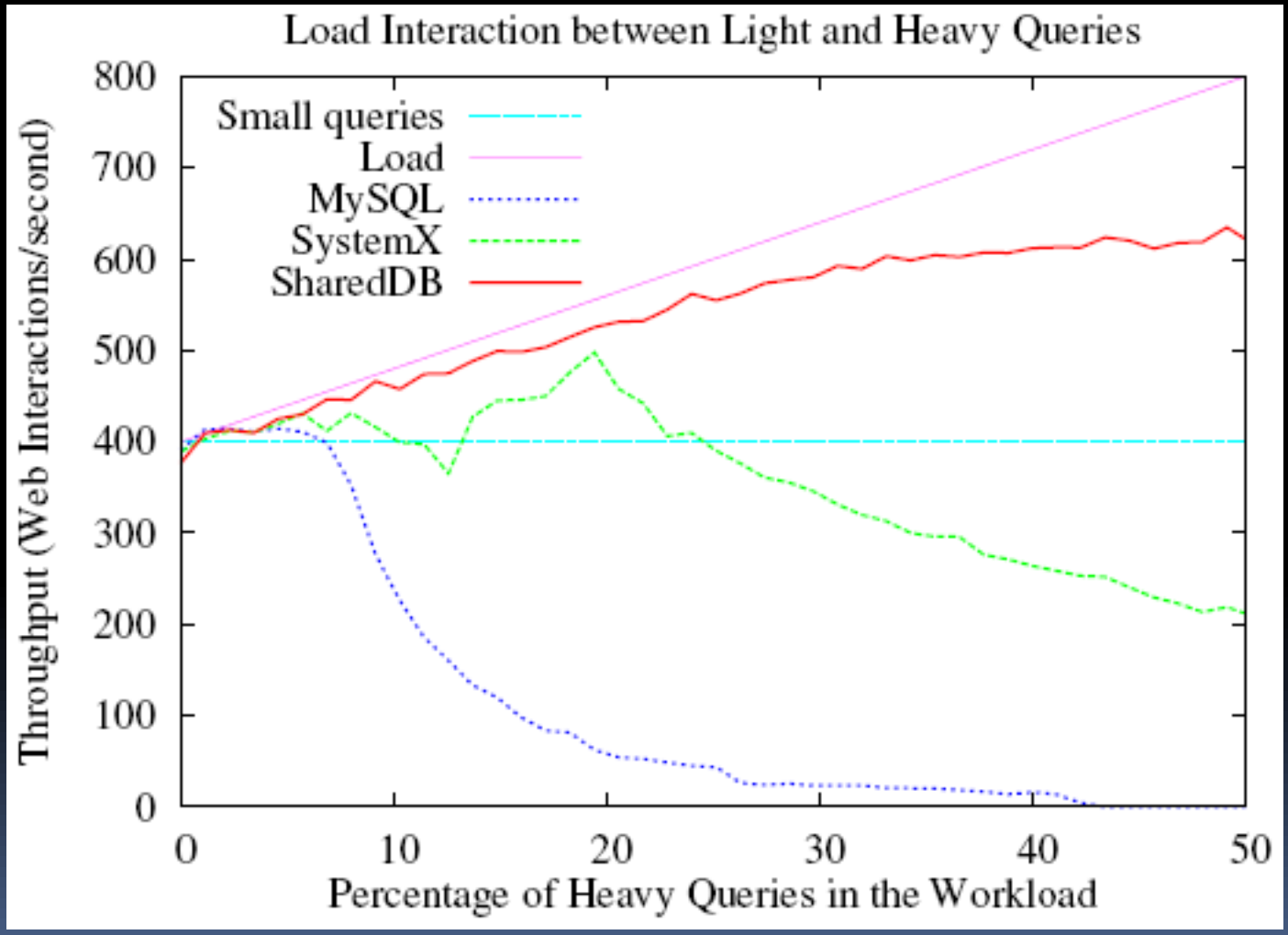


Systems@ETH zurich

Raw performance



Predictability, robustness



Many optimization options

- Virtualized operators
- Flexible, elastic deployment through OS interaction
- Scalability through operator/plan replication across cores and machines
- Hardware acceleration by operator offloading and in-network data processing
- Operator parallelism

CONCLUSIONS

Big data = big demand

- Increasing amount of systems where, due to their cost, a tailored solution is possible
  - Competitive advantage
  - New Services
- Expect to see many more such systems
  - NoSQL might not even be the most interesting one …