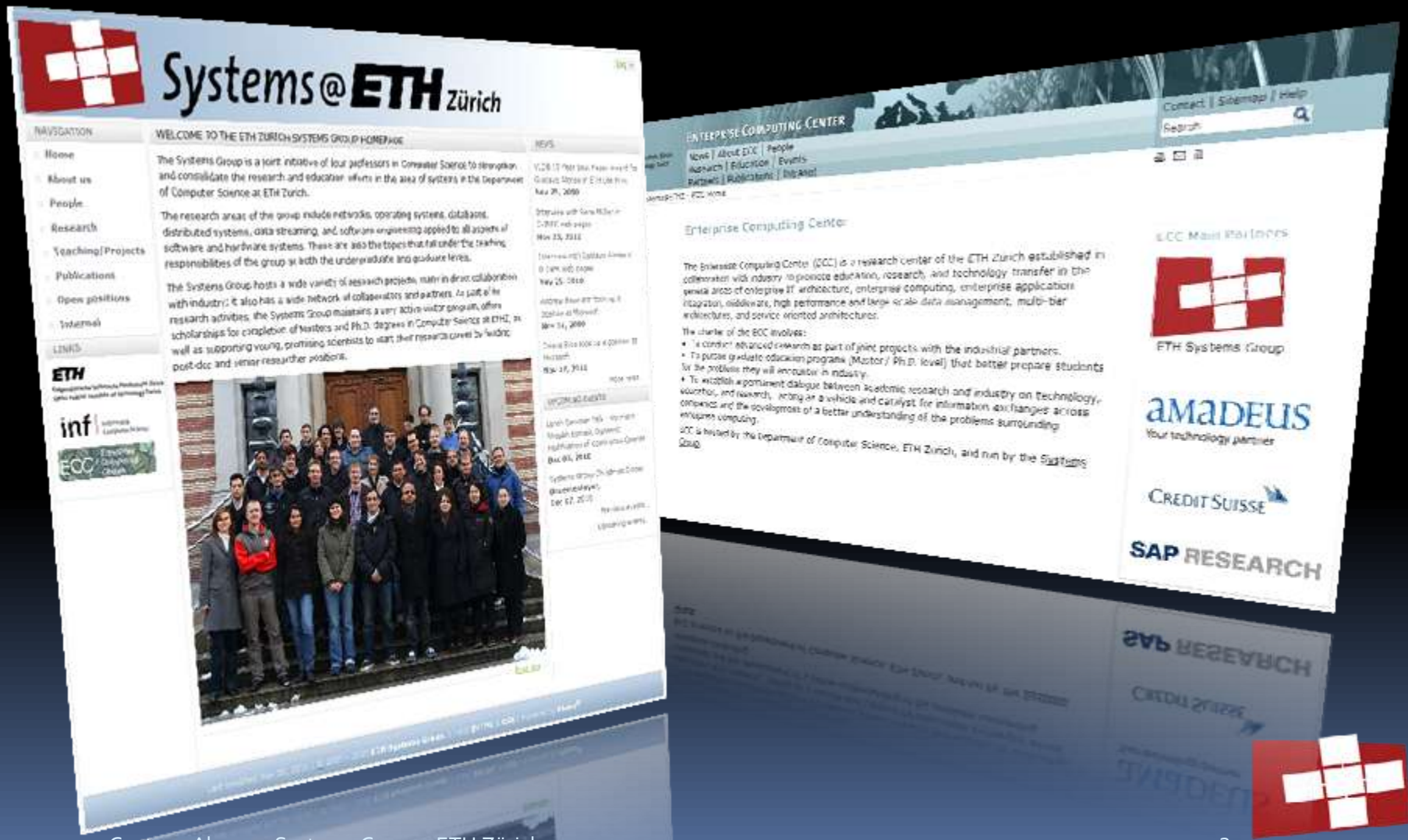
MULTICORE IN DATA APPLIANCES

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

Systems Group = www.systems.ethz.ch
Enterprise Computing Center = www.ecc.ethz.ch



The SwissBox project

- Build an open source data appliance
  - Hardware
  - Software

Goals

- Robust by design
- Scalable by design
- Fully predictable
- Behavior immune to peak loads (read or write)
- Efficient use of modern hardware
  - Cost efficiency
  - Power/space/management complexity

DATA APPLIANCE

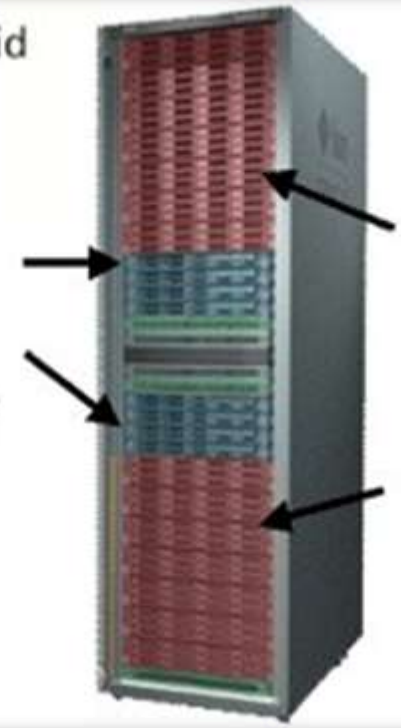
- Database in a box
  - Funny database
  - Funny box
- Examples:
  - Exascale (Oracle)
  - Twin-Fin (Netezza – IBM)
  - NewDB (SAP)
  - Teradata

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

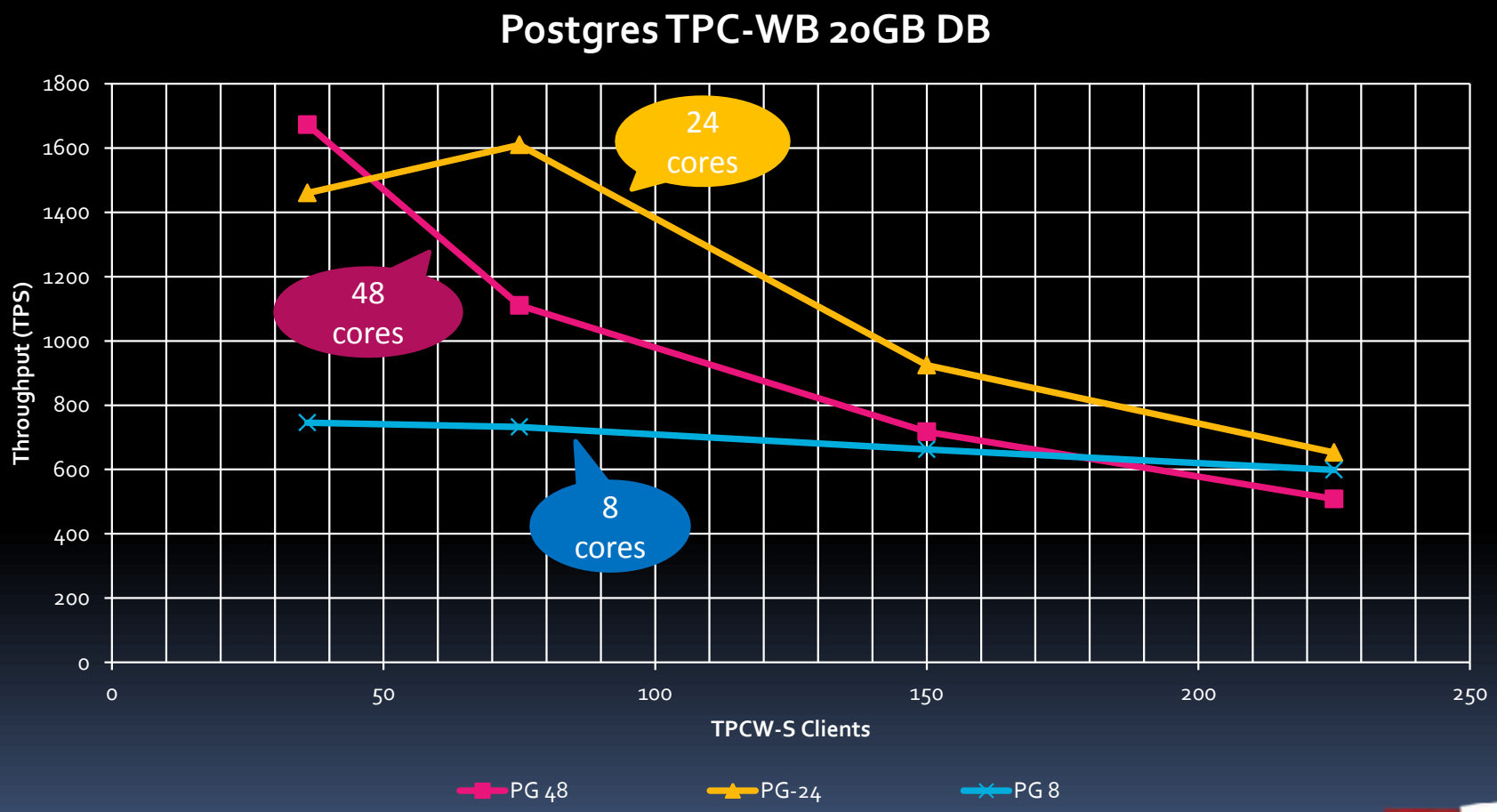
- Intelligent storage manager
- Massive caching
- RAC based architecture
- Fast network interconnect

The Multicore Challenge

Data parallelism

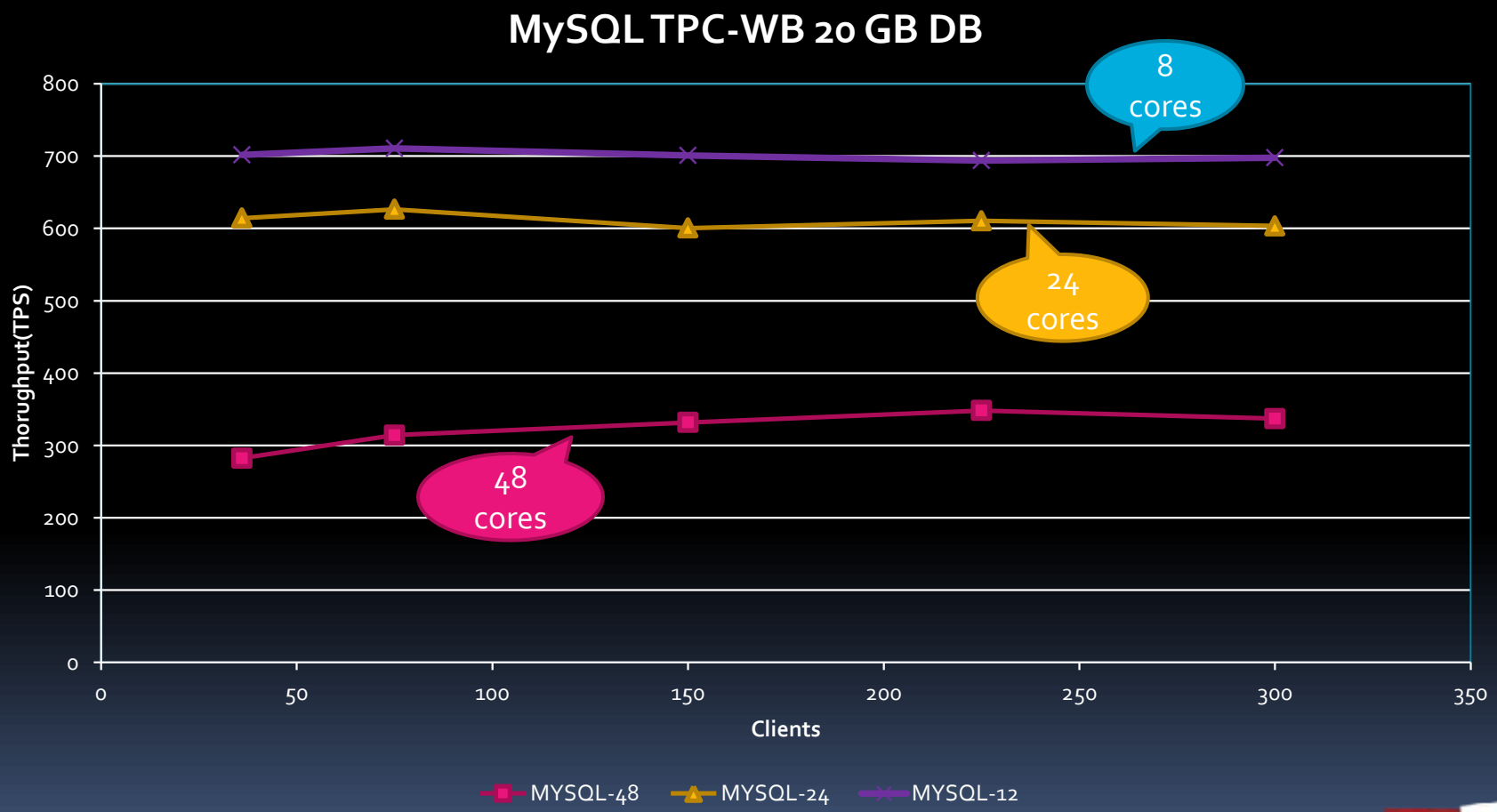
- Relational model is highly parallel
  - Independent tables
  - Orthogonal operators
  - Intra- and Inter-query parallelism
  - Most successful commercial parallel systems
- And yet …

Database engines and multicore



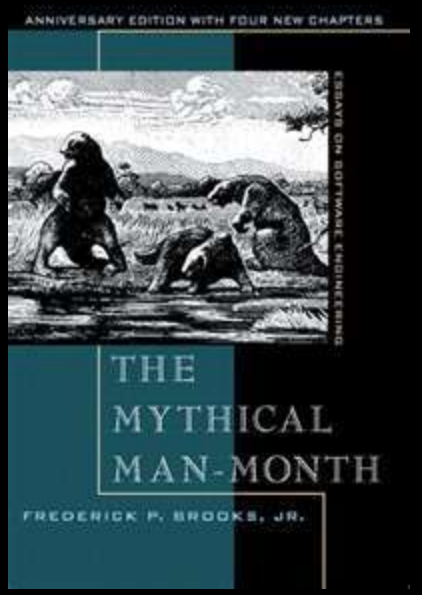
Salomie, Subasu, Giceva, Alonso, EuroSys 2011

Database engines and multicore



Salomie, Subasu, Giceva, Alonso, EuroSys 2011

CLAIM #1



Adding resources to a troubled application does not necessarily lead to improvements

Size matters

- The challenge of appliances is the unprecedented power available
  - 64 cores AMD, 256 GB memory, 10 Gb network + 3 TB NAS: ~14k CHF
  - Imagine a rack full of those

Is your job large enough?

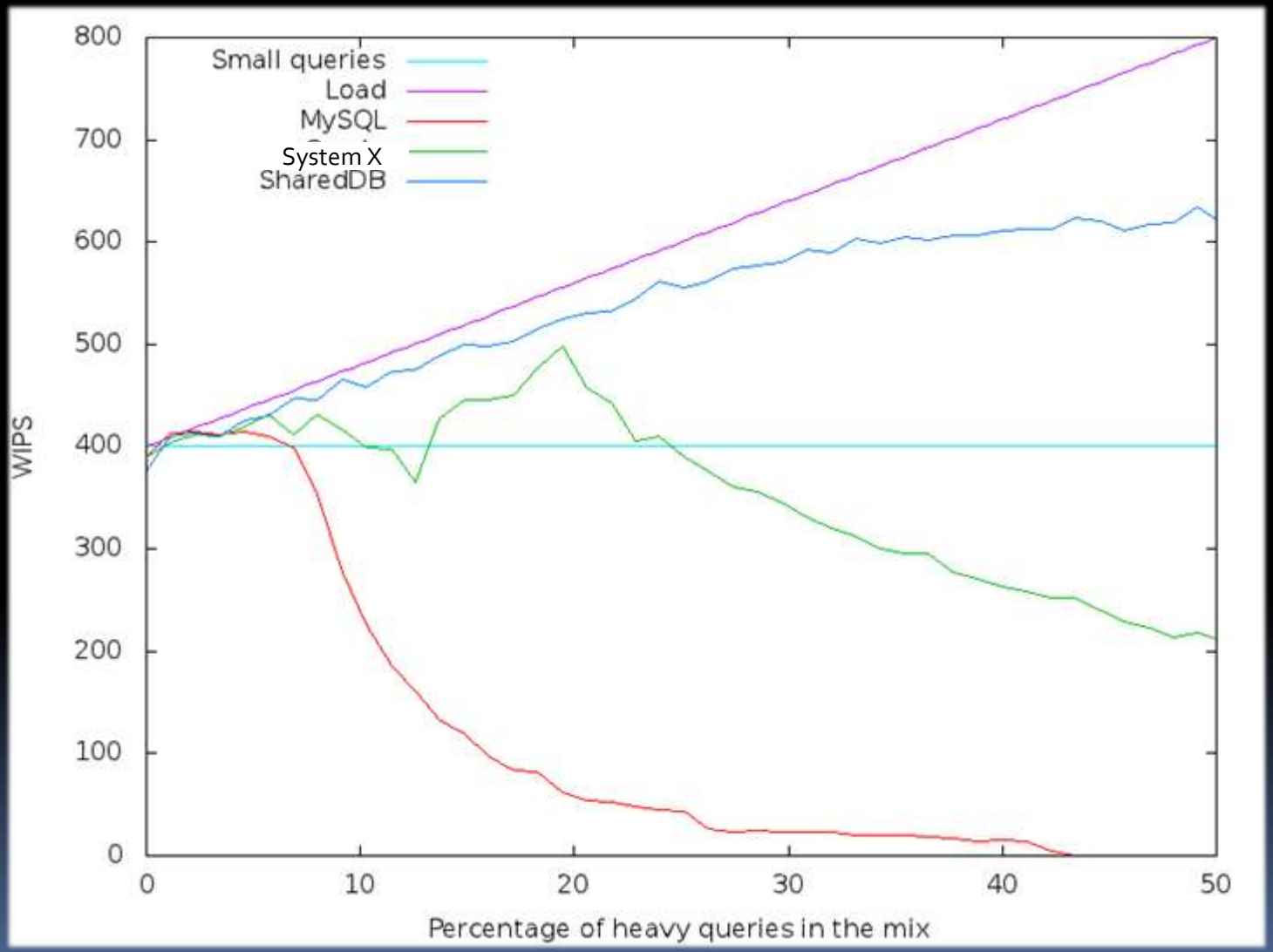
CLAIM #2

Large scale parallelism sensible only when looking at aggregated loads

Load interaction

- In a highly parallel system, multiple jobs will get on each other's way:
  - Synchronization
  - Data movement
  - Resource arbitrage
  - Resource capping
  - Heavy vs. light jobs
  - Management and coordination

Load interaction in practice



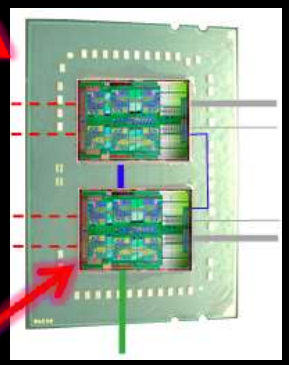
Giannikis, Alono, Kossmann, PVLDB 2012

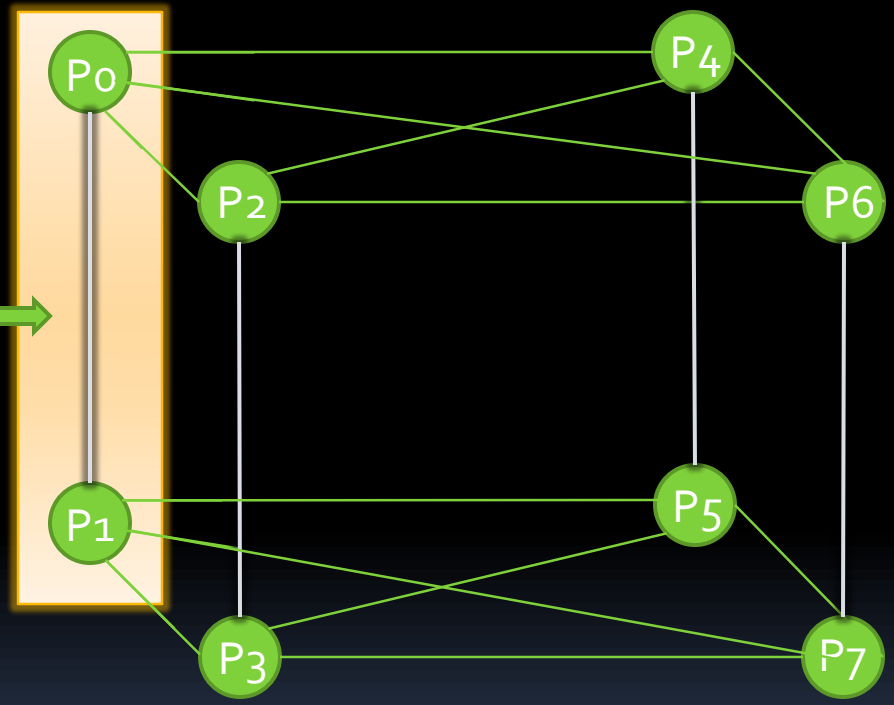
CLAIM #3

Robustness and performance can only be obtained by minimizing interaction

Locality in the XXIst century

Each package has:
- 12 cores
- 4HT ports
- 4 memory channels



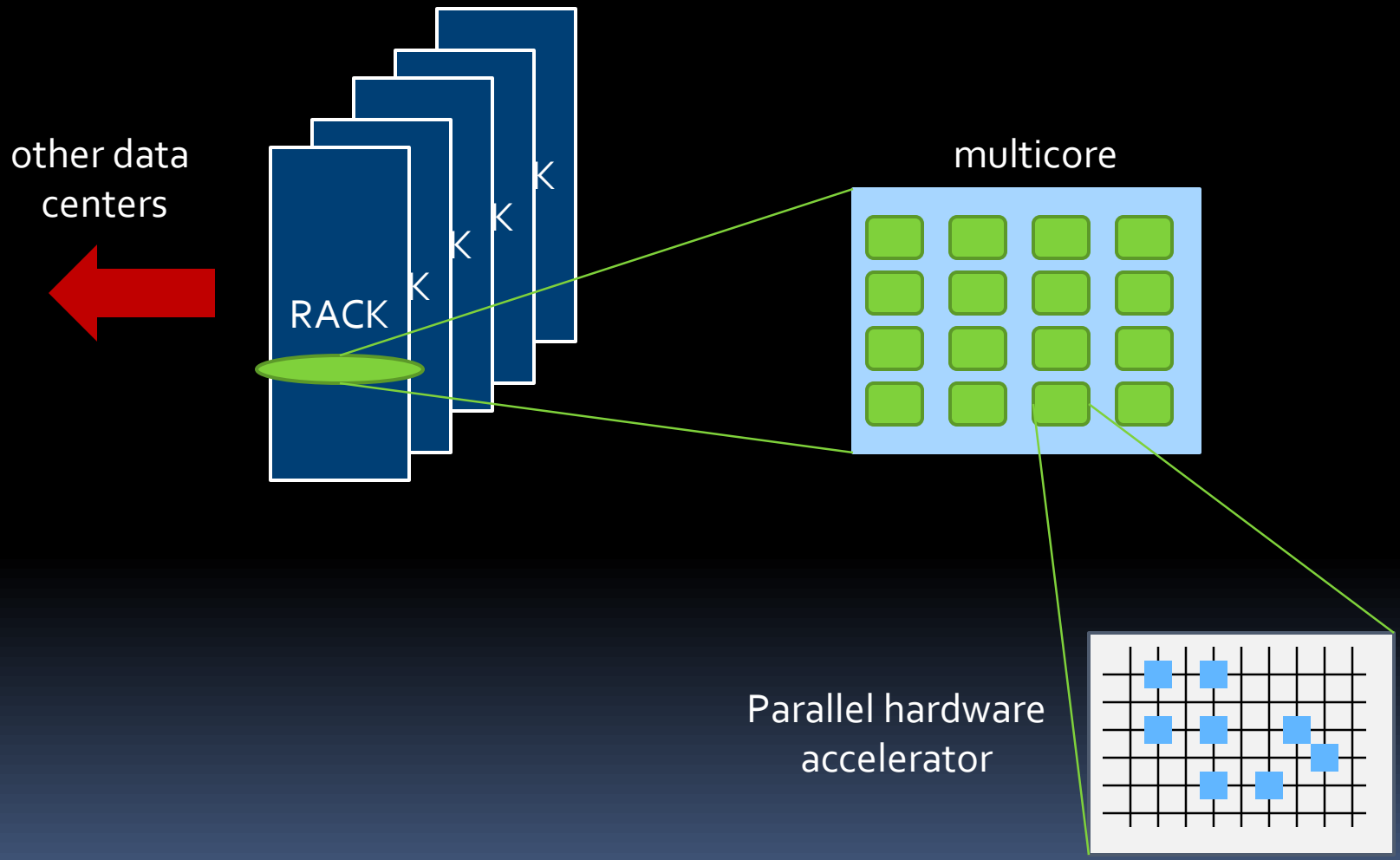


Each die has:
- 6 cores
- 4HT ports
- 2 memory channels

CLAIM #4

Robustness and definitely performance can only be achieved on fixed data paths

Architecture



CLAIM #5

Traditional layered architectures (HW/OS/VM/App) do not work in these environments

CLAIM #6

Strong notions of consistency (serializability) and atomicity of complex programs no longer feasible

SWISSBOX

Alonso, Kossmann, Roscoe, CIDR 2011

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

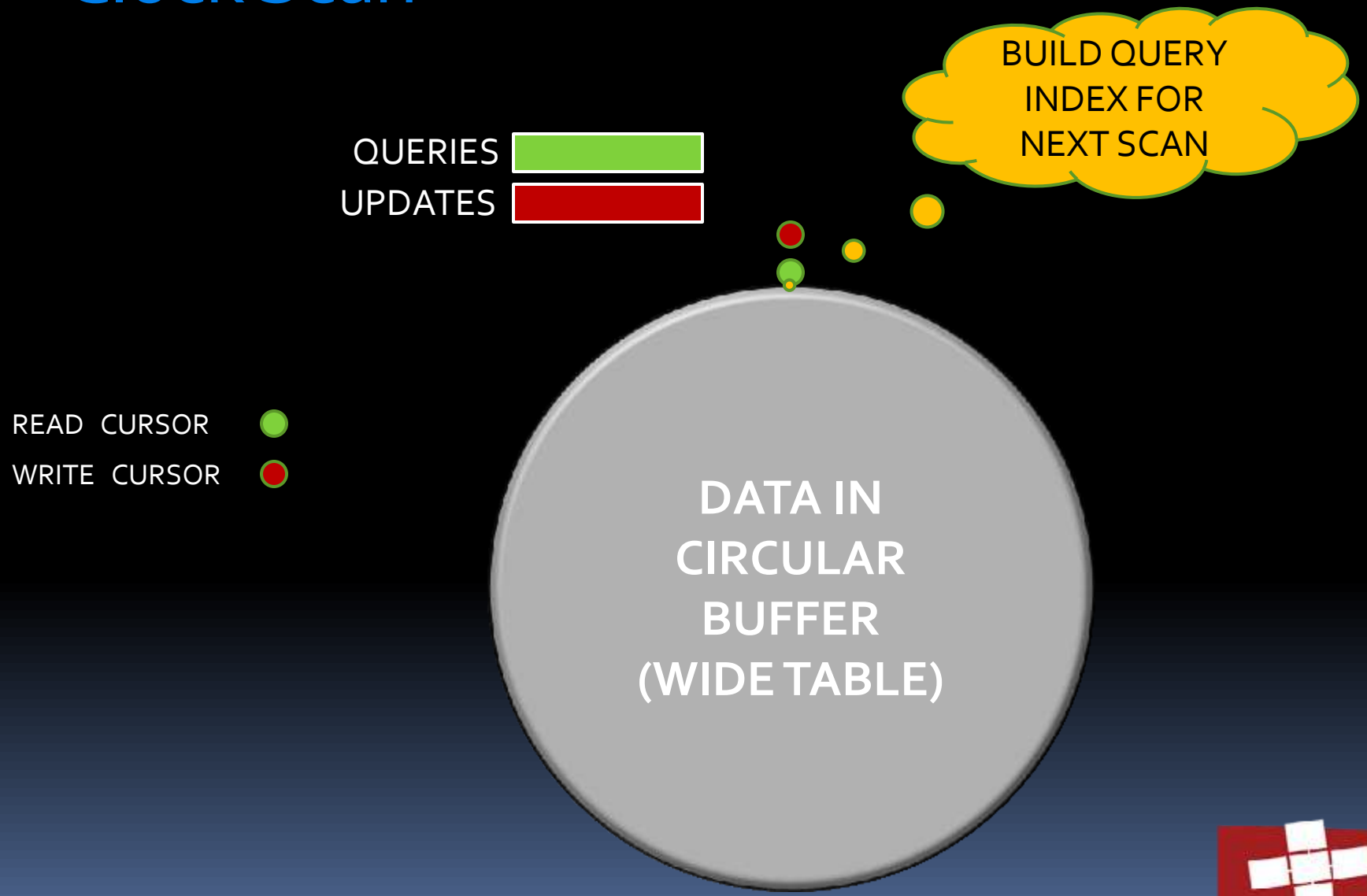
Claim # 1 => Deterministic behavior

Adding resources to an application does not necessarily lead to a performance improvement

System performance completely determined at design time through simple parameters

Clock Scan

Unterbrunner, Giannikis, Alonso, Kossmann, PVLDB 2009

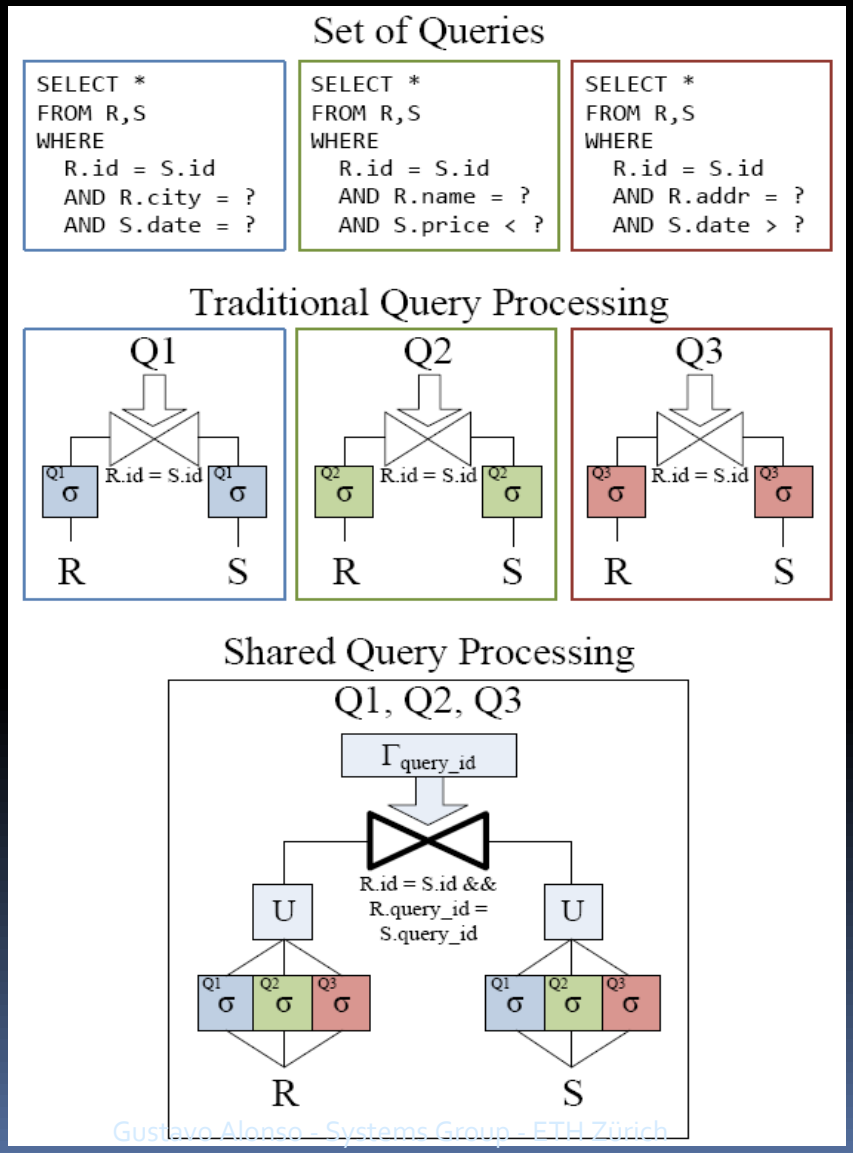


Claim # 2 => Batch processing

Large scale parallelism makes sense only when considering aggregated loads

Execution proceeds in batches (1000's of queries per batch)

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

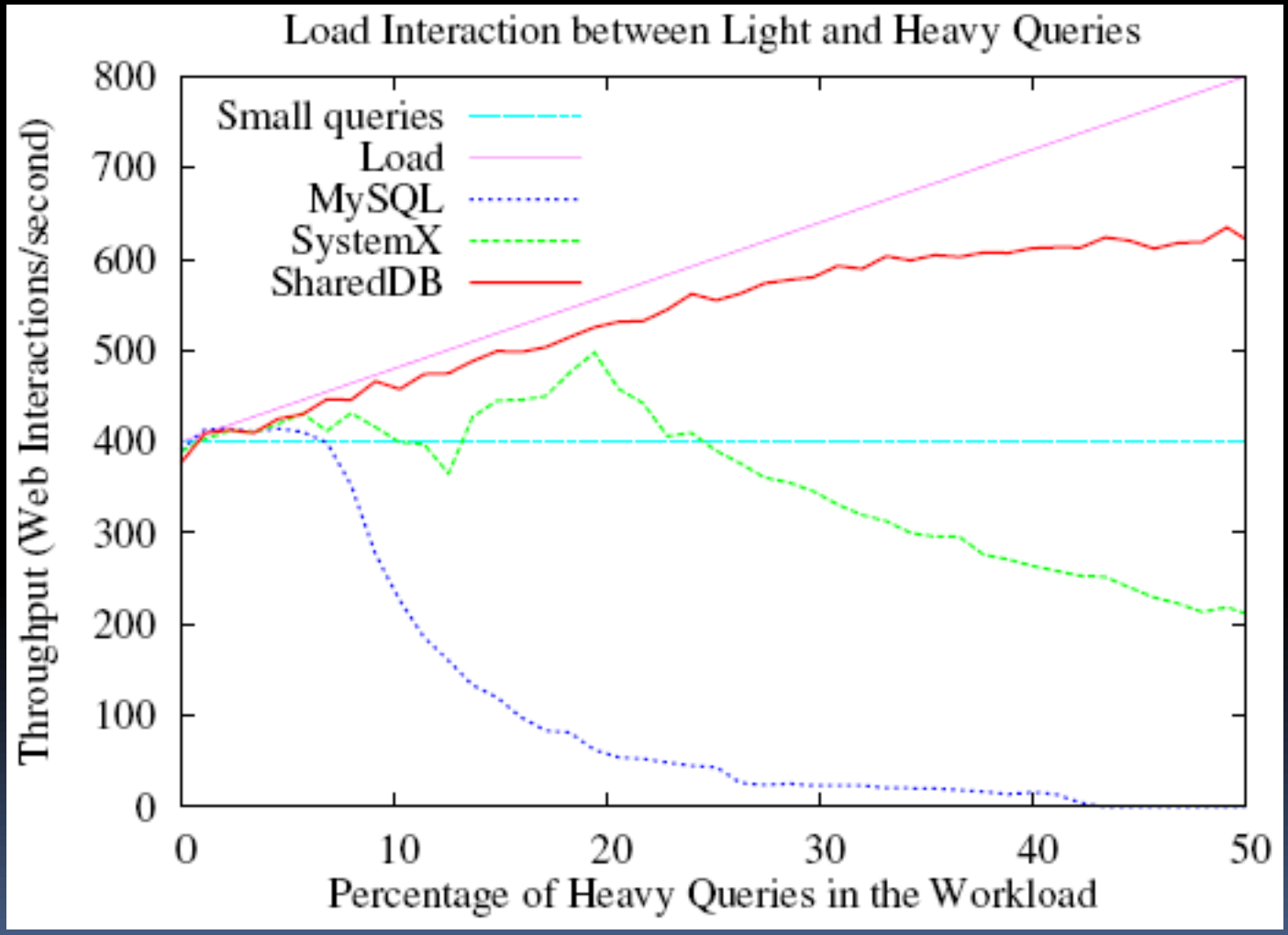
Giannikis, Alono, Kossmann, PVLDB 2012

Claim # 3 => No load interaction

Robustness and performance can only be obtained by minimizing interaction

Operators are orthogonal and work on clearly delimited resources

Predictability, robustness



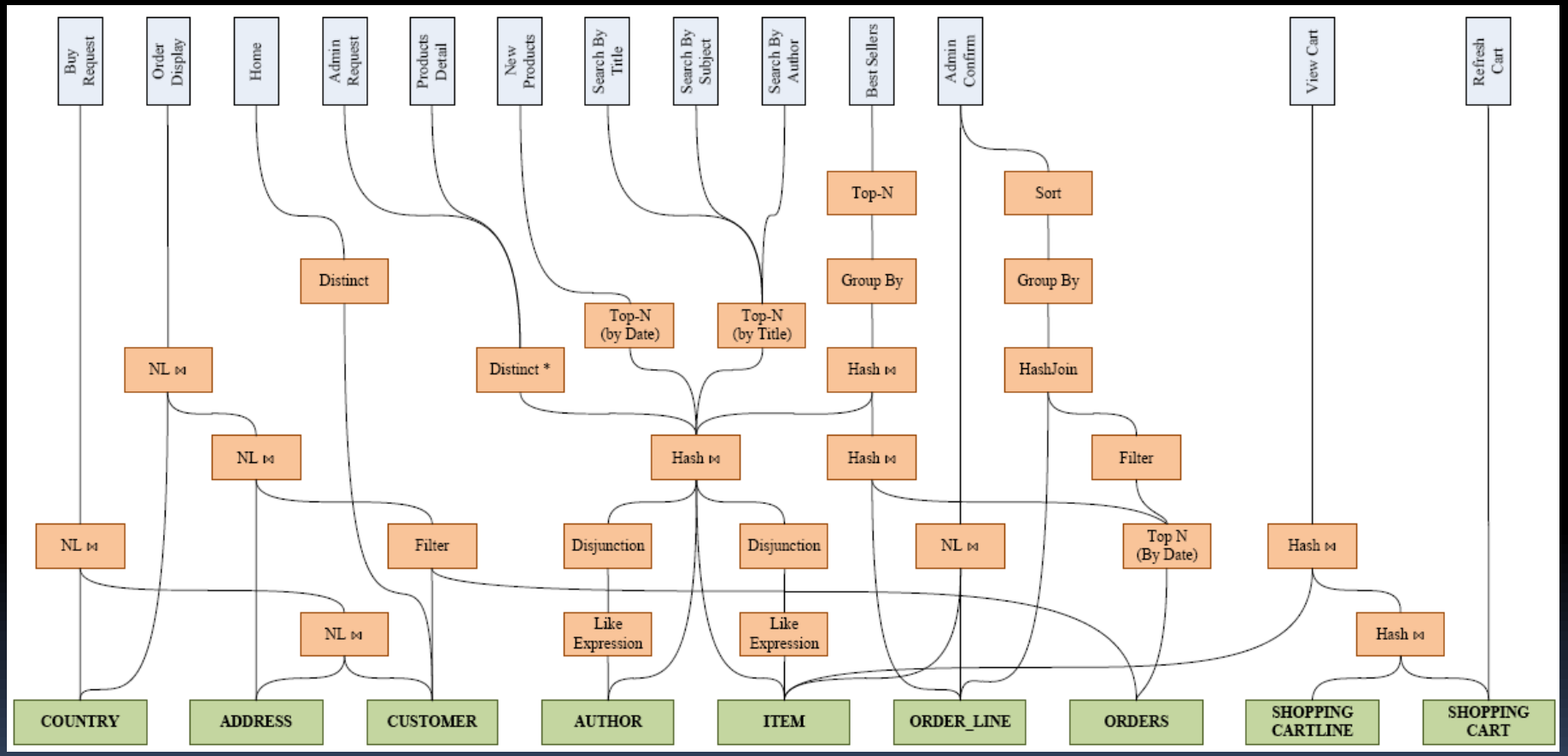
Giannikis, Alono, Kossmann, PVLDB 2012

Claim # 4 => No dynamic scheduling

Robustness (and definitely performance) can only be achieved on fixed data paths

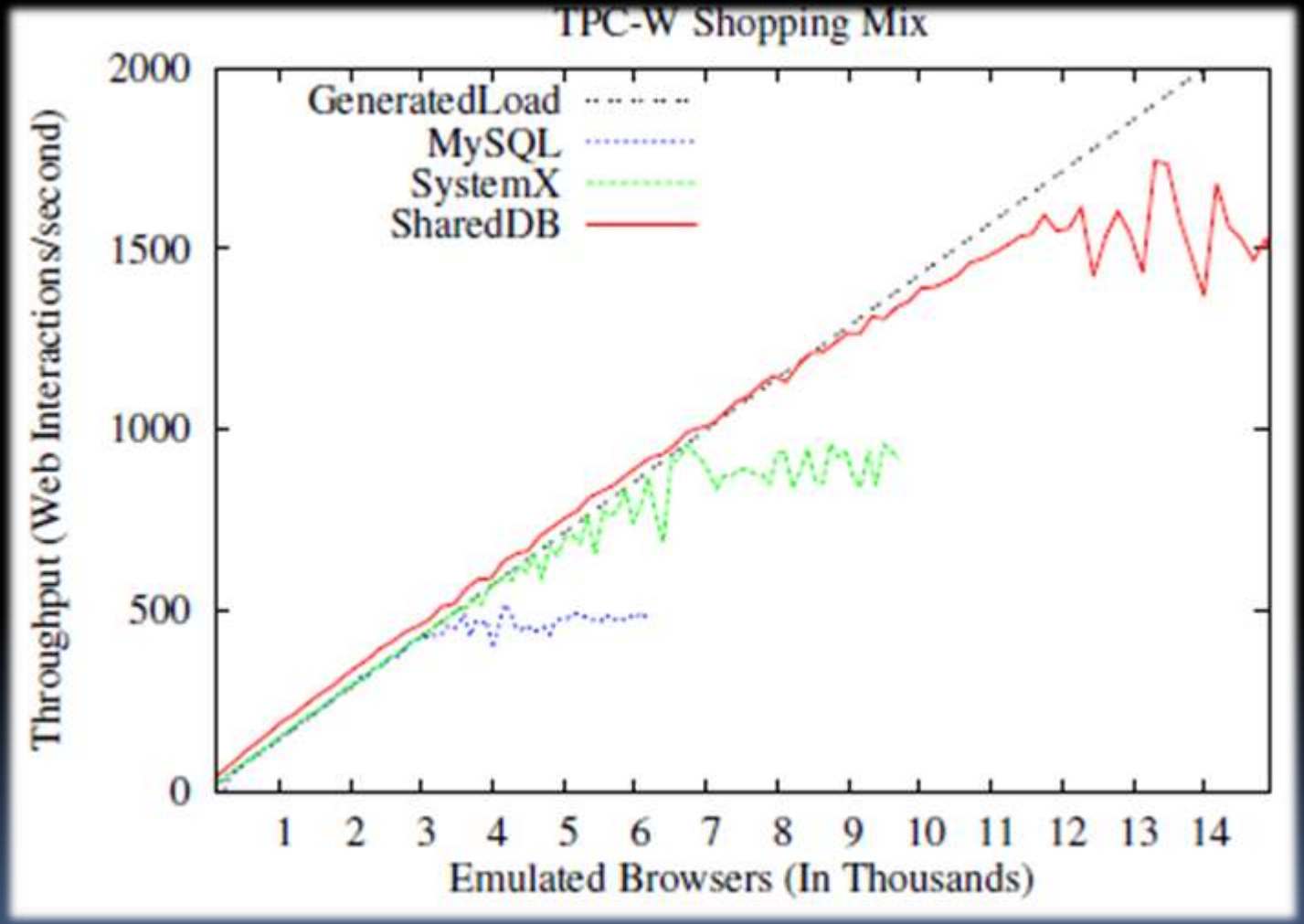
No dynamic scheduling, operators are always on and at fixed locations

Single plan: operator per core



Systems@ETH Zürich

Raw performance



Giannikis, Alono, Kossmann, PVLDB 2012

Claim # 5 => Open stack

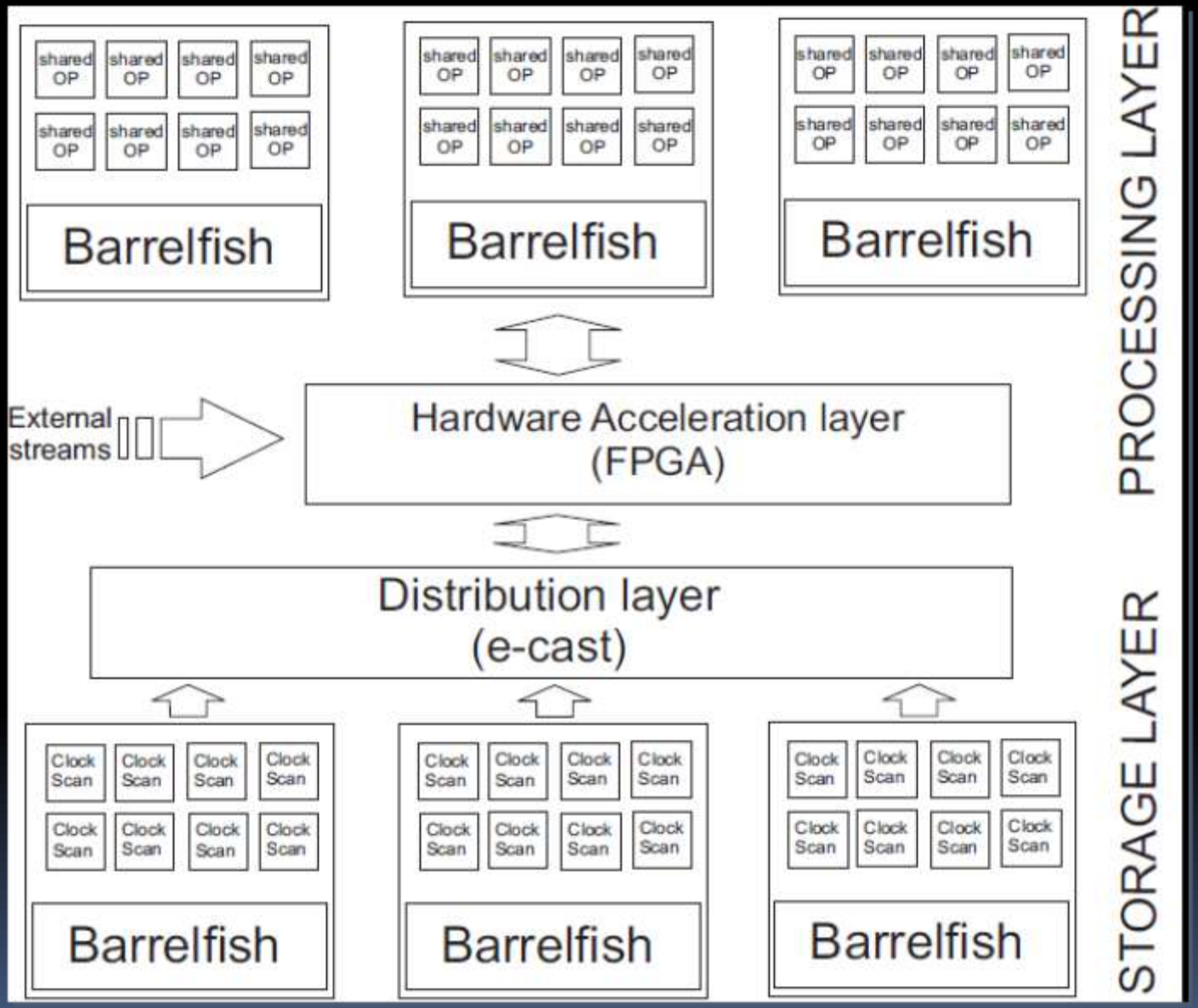
Traditional layered architectures (HW/OS/VM/App) do not work in these environments

Operating System / Database co-design

Claim # 6 => Consistency

Strong notions of consistency (serializability) and atomicity of complex programs no longer feasible

Snapshot isolation, multiversions, eventual consistency



SWISSBOX

Systems@ETH Zürich

Where are we?

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

Next steps

- Hardware acceleration
- Query optimizer
- Parallel operators
- OS / DB interfaces

In the future

- 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

SwissBox in a nutshell

- A new way to process data
  - Parallel, predictable by design
  - Not optimal but good enough
  - Co-design at all levels
- Great opportunity for research
  - Redo everything from scratch