Rack-scale Data Processing System

Jana Giceva, Darko Makreshanski, Claude Barthels, Alessandro Dovis, Gustavo Alonso

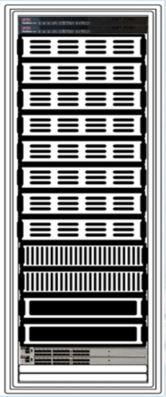
Systems Group, Department of Computer Science, ETH Zurich

Rack-scale Data Processing System

Jana Giceva, Darko Makreshanski, Claude Barthels, Alessandro Dovis, Gustavo Alonso

Systems Group, Department of Computer Science, ETH Zurich

Application's perspective of a rack



FruitBox – a data processing system

Transactional QP

Graph processing

Analytical QP

Machine learning

Ad-hoc BI QP

MULTI FLAVOR DATA PROCESSING

ETH zürich

FruitBox

- Building a system for multi-flavor data processing:

1. Hardware that meets the resource demand.
2. System architecture to support workload heterogeneity.
3. Aim for 10s-100s millions of requests per second.
4. Efficient resource utilization.

MULTI FLAVOR DATA PROCESSING

Graph processing

Machine learning

Transactional QP

Analytical QP

Ad-hoc BI QP

FruitBox – a rack-scale data processing system

- Which box could run such a heterogeneous WL?
  - A multicore is not enough
- A rack-scale system:
  - More resources
  - Better isolation
  - Blurring the machine-cluster boundaries

MULTI FLAVOR DATA PROCESSING

Graph processing

Machine learning

Transactional QP

Analytical QP

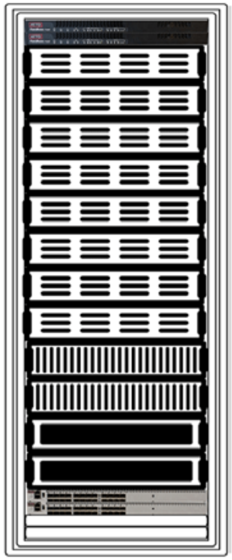
Ad-hoc BI QP

RACK-SCALE SYSTEM

1000s of cores

TBs of RAM

InfiniBand



Rack-scale data processing system

Custom build a rack-scale system for data processing?

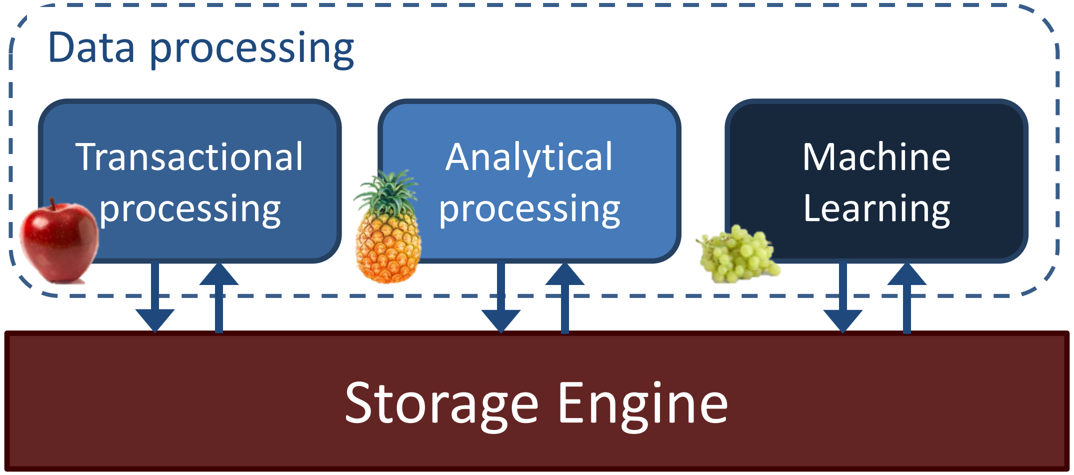
Many such commercial systems exists – **Data Appliances**

ORACLE Exadata | Netezza (IBM) TwinFin | HUAWEI SAP HANA

and many more …

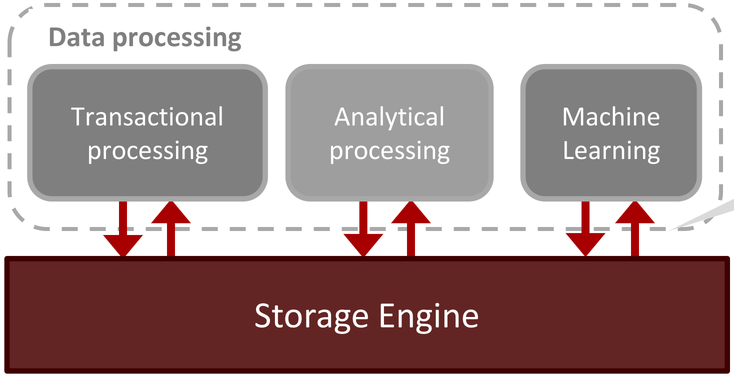
System design for Multi-flavor data processing

- Separate data-storage from data-processing



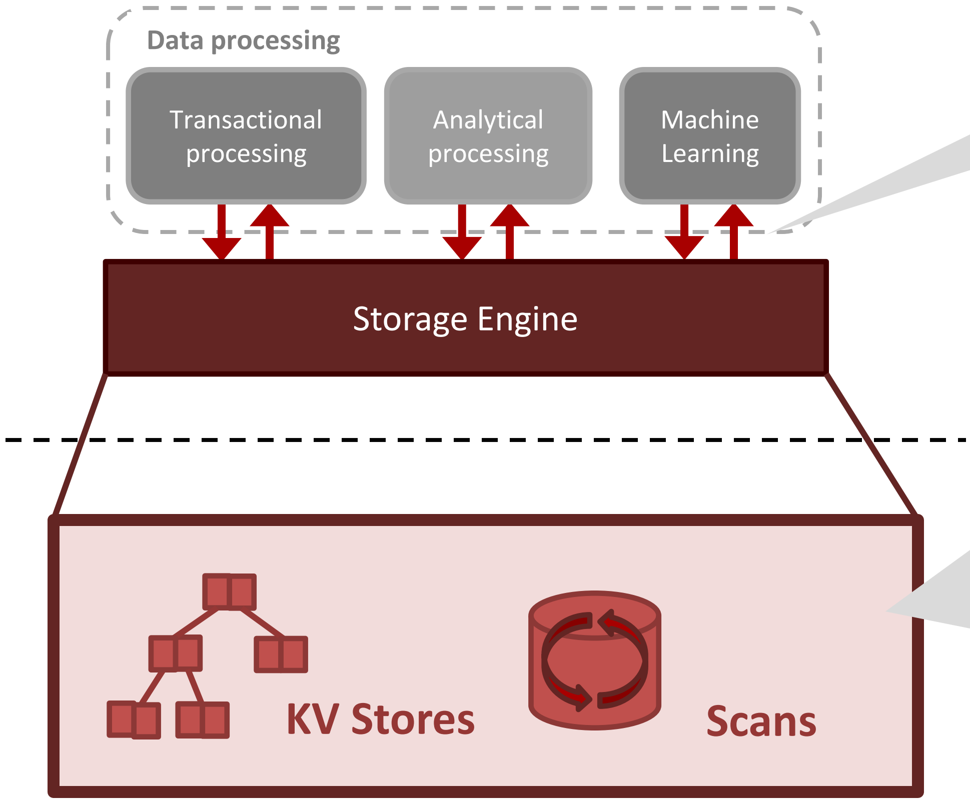
- Achieve both *physical* and *logical* data independence

Storage Engine



Tuple- and *batch-based* interface to the storage engine.

Storage Engine



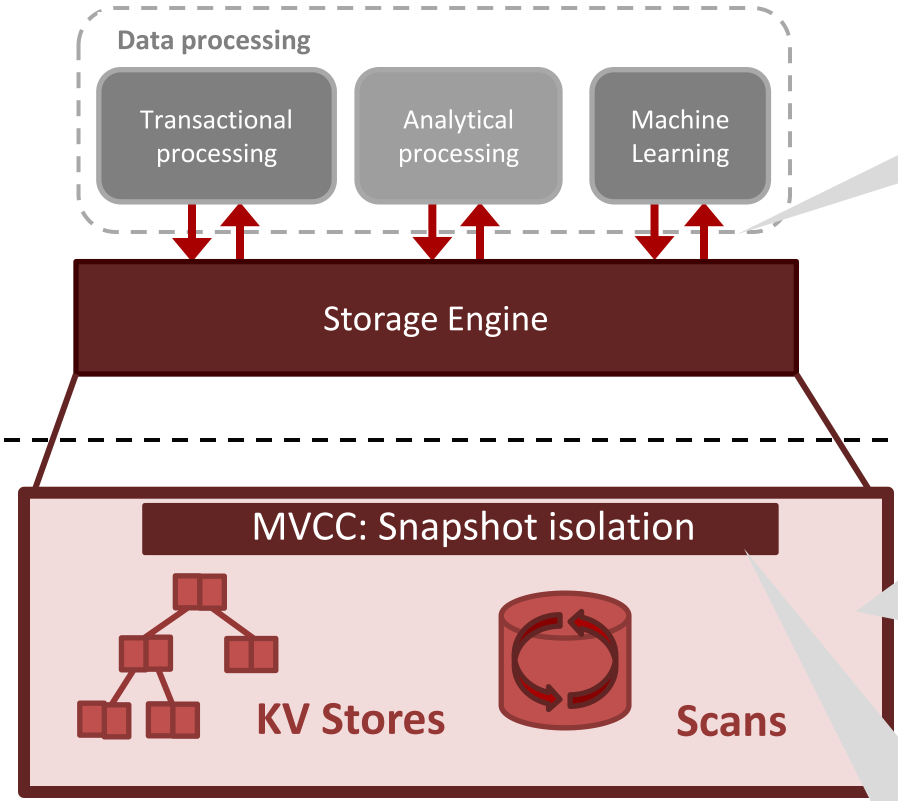
Tuple- and *batch-based* interface to the storage engine.

Storage engine components:

- KV Stores (B-tree)
- Crescando Scans

KV Stores → transactional
Scans → analytical

Storage Engine



Tuple- and *batch-based* interface to the storage engine.

Storage engine components:

- KV Stores (B-tree)
- Crescando Scans

KV Stores → transactional
Scans → analytical

Transaction logic separated from query processing.

Hyder [SIGMOD'15], HyPer[VLDB'11], Hekaton [SIGMOD'14],
SharedDB [Giannikis PhD'14], Tell [SIGMOD'15], Multimed [Eurosys'11]

Handling millions of requests/second

- It makes no sense to process them individually
  if they access the same data.
- Why should each query scan a TB of data?

vs

- *Batch requests* – share data, computation, bandwidth

… for higher throughput and predictable performance
trading off a bit of latency.

IBM Blink, MonetDB/X100[VLDB'07], CJOIN [VLDB'09], Crescando[VLDB'09], SharedDB [VLDB'12,'14],

Efficient resource utilization

MULTI FLAVOR DATA PROCESSING

Graph processing

Machine learning

Transactional QP

Analytical QP

Ad-hoc BI QP

- Noisy system environment
- Load interaction

☹ Unpredictable performance
☹ Not meeting SLAs

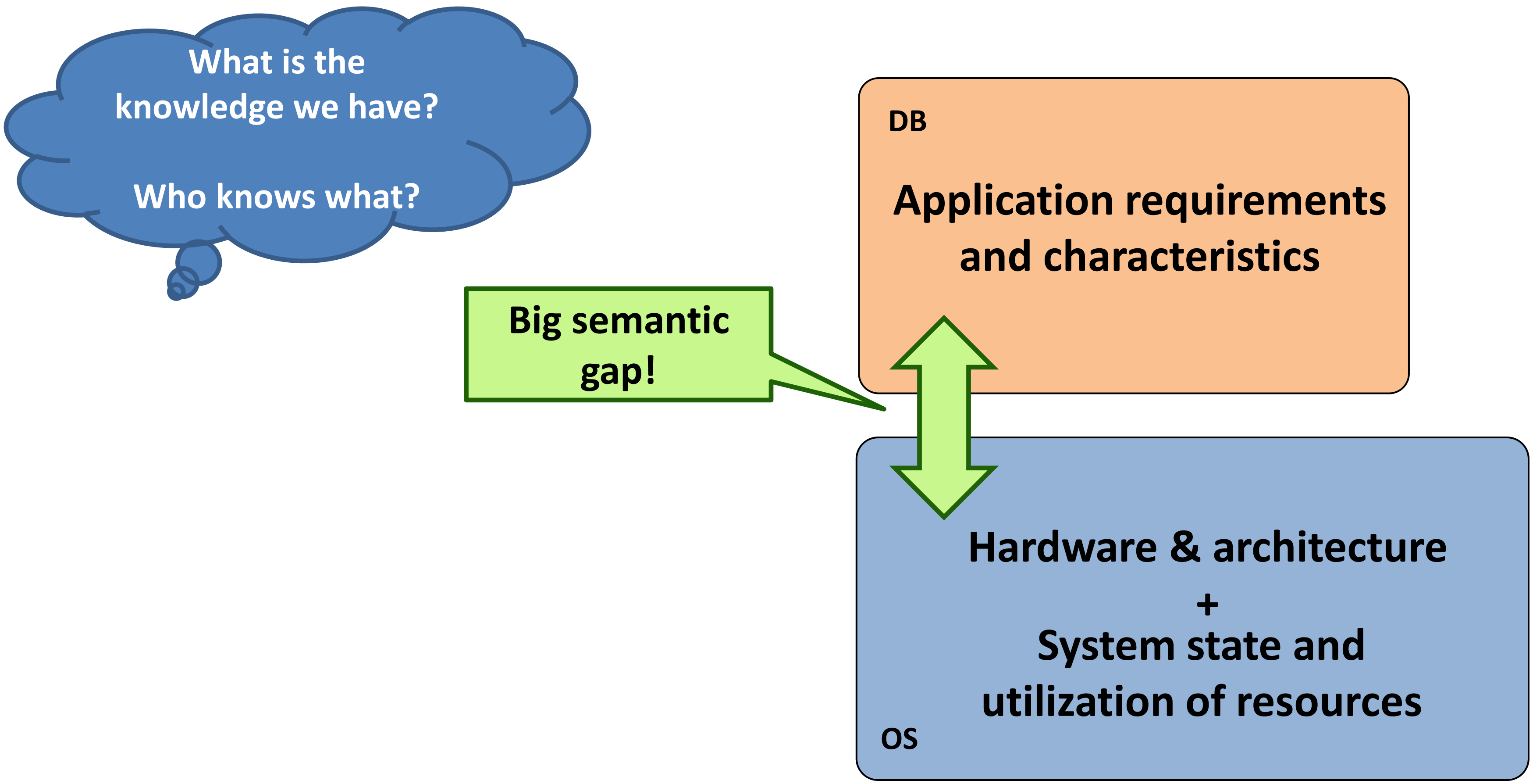
- Resource overprovisioning

☹ Inefficiency and higher cost

- Getting the most out of such a complex system requires cross-layer optimization.
  - e.g. DB/OS co-design

- Already some work on multicore systems.

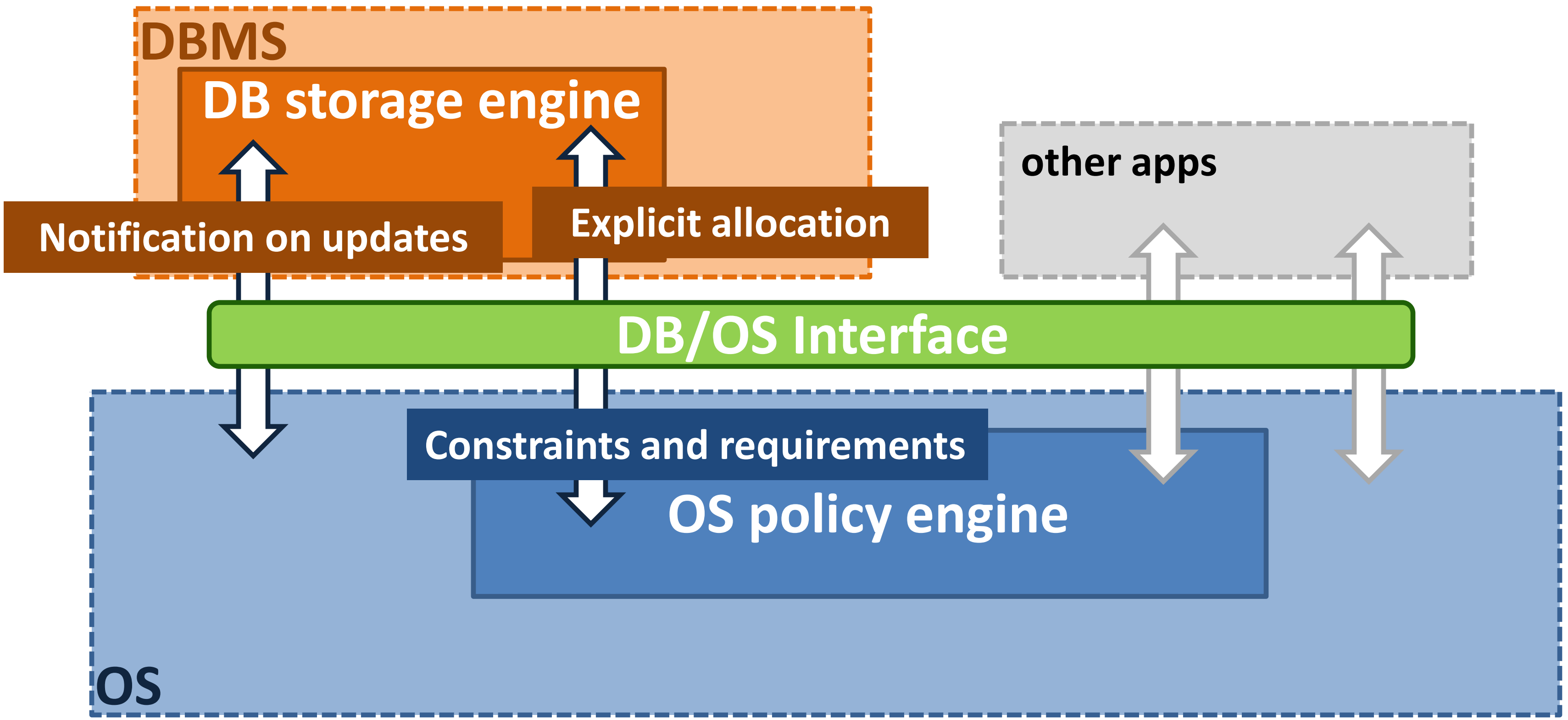
COD: DB/OS co-design



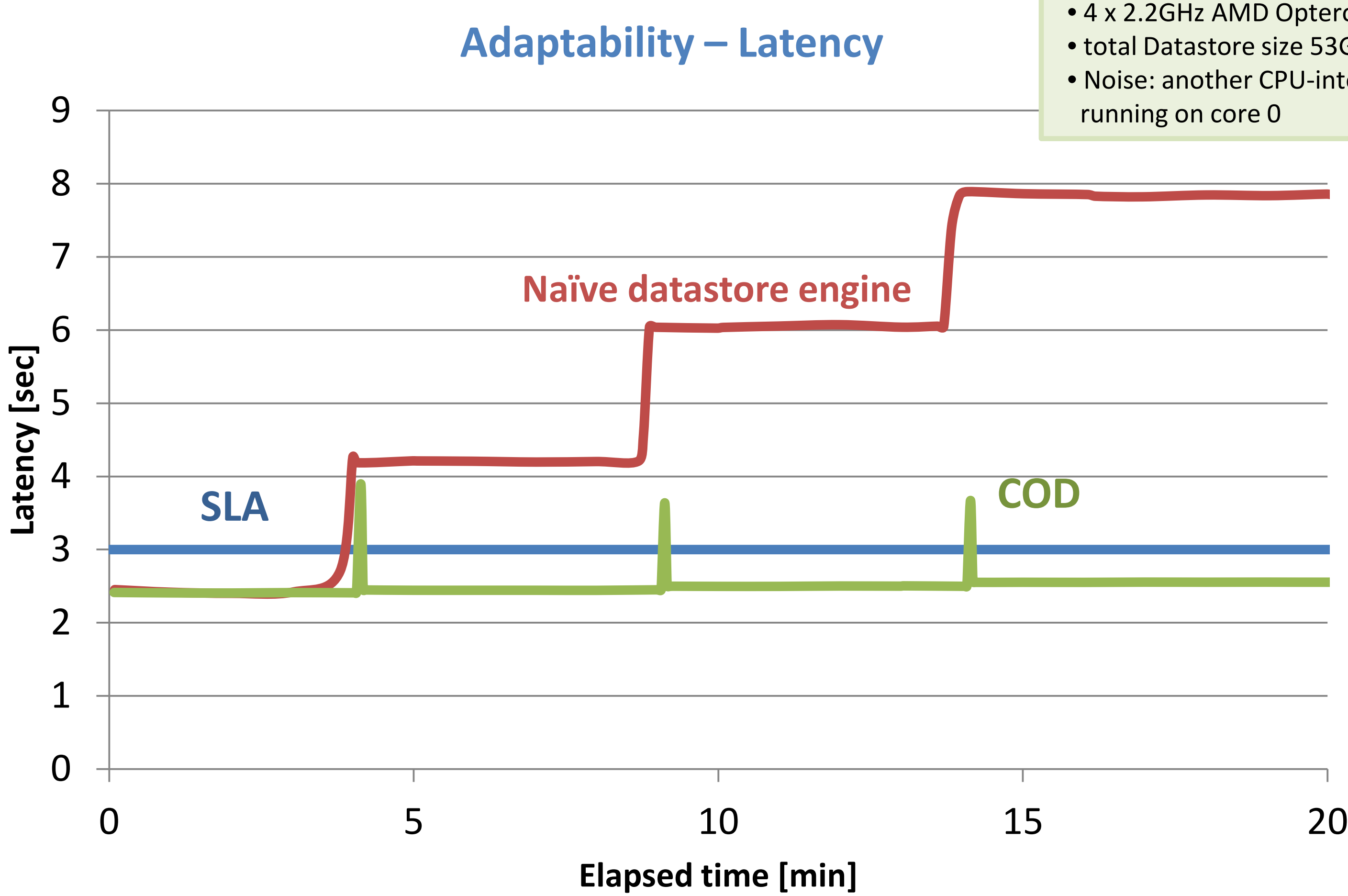
COD: Database/Operating System co-design [CIDR’12]

ETH zürich

COD’s interface



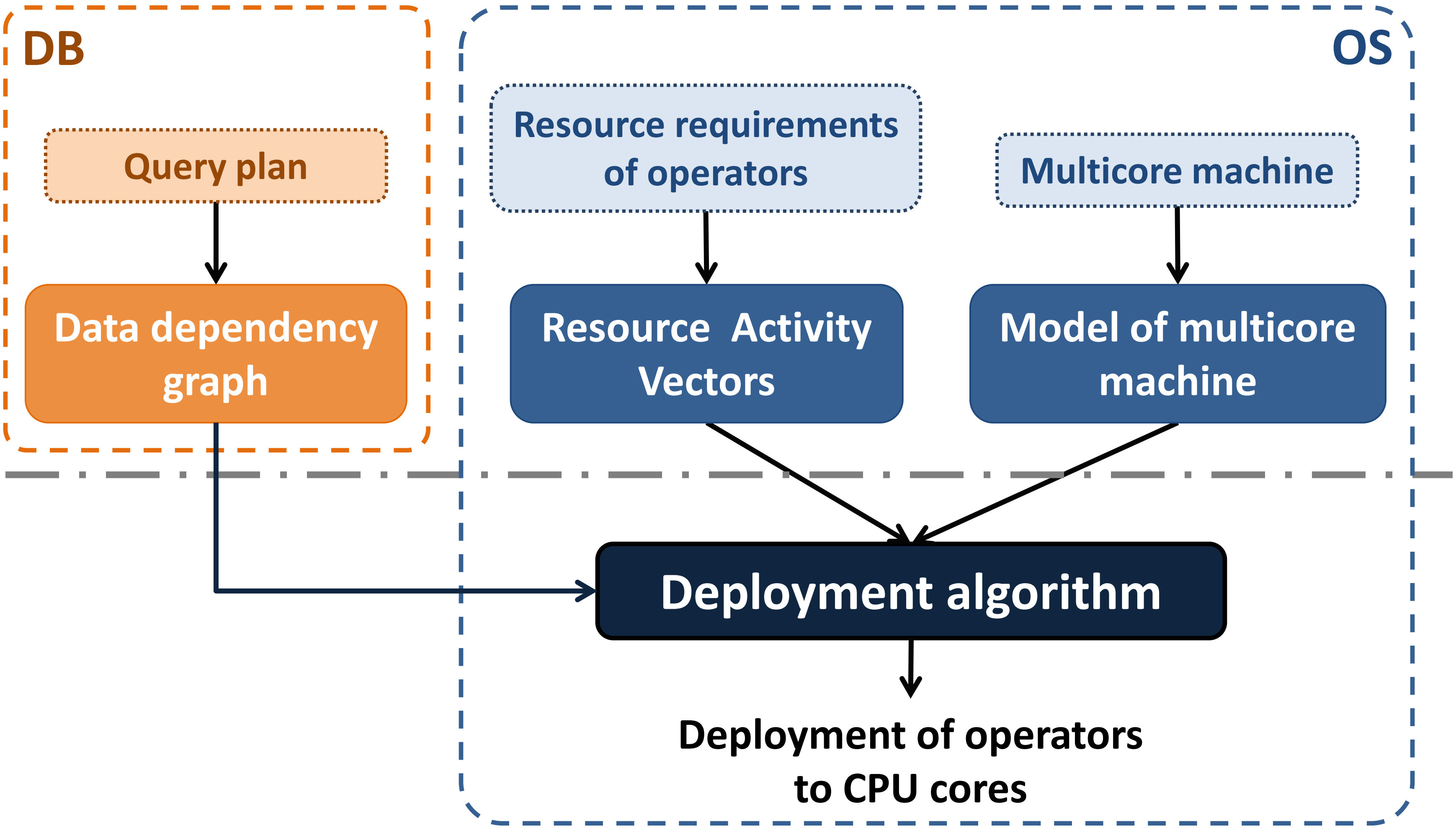
Adaptability to dynamic system state



Experiment setup

- AMD MagnyCours
- 4 x 2.2GHz AMD Opteron 6174 processors
- total Datastore size 53GB
- Noise: another CPU-intensive task running on core 0

Resource efficient deployment

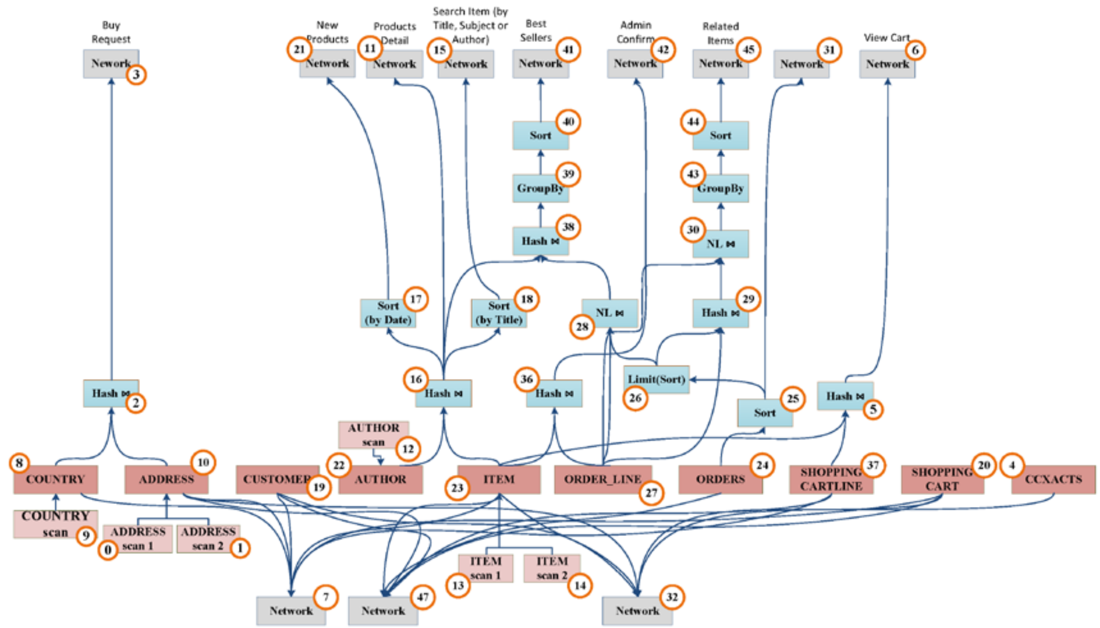


Deployment of query plans on multicores [VLDB'15]

Evaluation

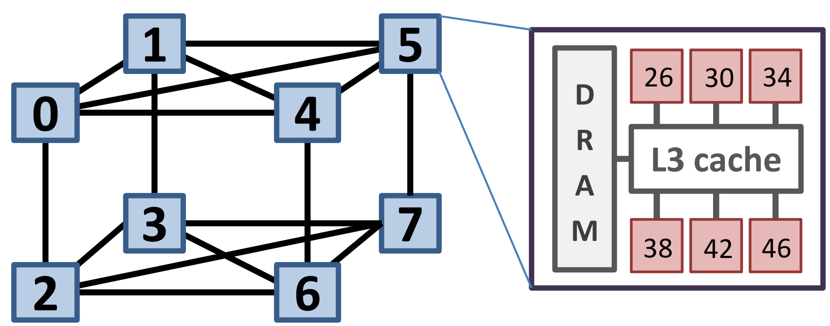
Query plan

- SharedDB's TPC-W [1]
- 11 web-interactions in one query plan
- 44 operators
- 20GB dataset



AMD Magnycours

- 4 x 2 dies:
  - 6 cores
  - 5 MB L3 cache
  - 16 GB NUMA node



[1] SharedDB – Giannikis et al. VLDB'12

Comparison with standard approaches

		Throughput [WIPS]		Response Time [ms]		
Approaches	# cores	Average	Stdev	50^{th}	90^{th}	99^{th}
Default OS	48					
Operator per core	44					
Deployment algorithm						

Comparison with standard approaches

		Throughput [WIPS]		Response Time [ms]		
Approaches	# cores	Average	Stdev	50^{th}	90^{th}	99^{th}
Default OS	48	317.30	31.11	8.22	72.43	82.03
Operator per core	44	425.86	54.34	14.59	22.93	36.08
Deployment algorithm						

Comparison with standard approaches

		Throughput [WIPS]		Response Time [ms]		
Approaches	# cores	Average	Stdev	50^{th}	90^{th}	99^{th}
Default OS	48	317.30	31.11	8.22	72.43	82.03
Operator per core	44	425.86	54.34	14.59	22.93	36.08
Deployment algorithm	6					

Comparison with standard approaches

		Throughput [WIPS]		Response Time [ms]		
Approaches	# cores	Average	Stdev	50th	90th	99th
Default OS	48	317.30	31.11	8.22	72.43	82.03
Operator per core	44	425.86	54.34	14.59	22.93	36.08
Deployment algorithm	6	428.07	32.80	15.36	23.73	36.13

Comparison with standard approaches

		Throughput [WIPS]		Response Time [ms]		
Approaches	# cores	Average	Stdev	50^{th}	90^{th}	99^{th}
Default OS	48	317.30	31.11	8.22	72.43	82.03
Operator per core	44	425.86	54.34	14.59	22.93	36.08
Deployment algorithm	6	428.07	32.80	15.36	23.73	36.13

➡ *Performance / Resource efficiency savings of x7.37*

Conclusion

Multi-flavor data processing system

- We have all the pieces of the puzzle

Separate data-storage from data-processing

Batching as a first class citizen

Efficient resource management

… on a rack-scale system

Putting them together opens a lot of opportunities.

Conclusion

Multi-flavor data processing system

- We have all the pieces of the puzzle

- Intelligent storage engine:
  - Co-processors, active-memory, hardware specialization (FPGAs)
- Optimizing the network stack:
  - … for different memory access patterns
- Extend the cross-layer interface:
  - DB optimizer that is aware of the complexity of the rack
- Rack-scale resource management

Putting them together opens a lot of opportunities.