Bachelor-/Master-Forum 2020

Utilizing Emergent Technologies for Big Data Analytics

Institut für Informationssysteme (IFIS)

Arbeitsgruppe Groppe

Professor Dr. rer. nat. habil. Sven Groppe

https://www.ifis.uni-luebeck.de/index.php?id=groppe

Big Data and the Jungle of possibilities for Datamanagement

Zoo of Data Formats, for example:

- relational data
  - in relational databases
- XML
  - for exchange
- JSON
  - web data
- Resource Descr. Framework (RDF)
  - Semantic Web
- graph data
  - from social networks
- unstructured data
  - of social media like wikis

➡ Parallel use of different Data Models for storing and processing

Relational:

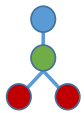
Primary		Secondary	Primary	

XML:

```
<root>
  <child>
    <first>hello</first>
    <sibling>sibling</sibling>
  <child>
</root>
```

JSON:

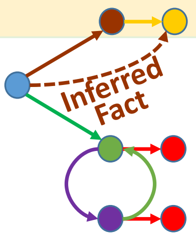
```
{root:{
      child:{
         first:hello,
         sibling:sibling
      }
}}
```



RDF/Graph Data:

Ontology of Semantic Web

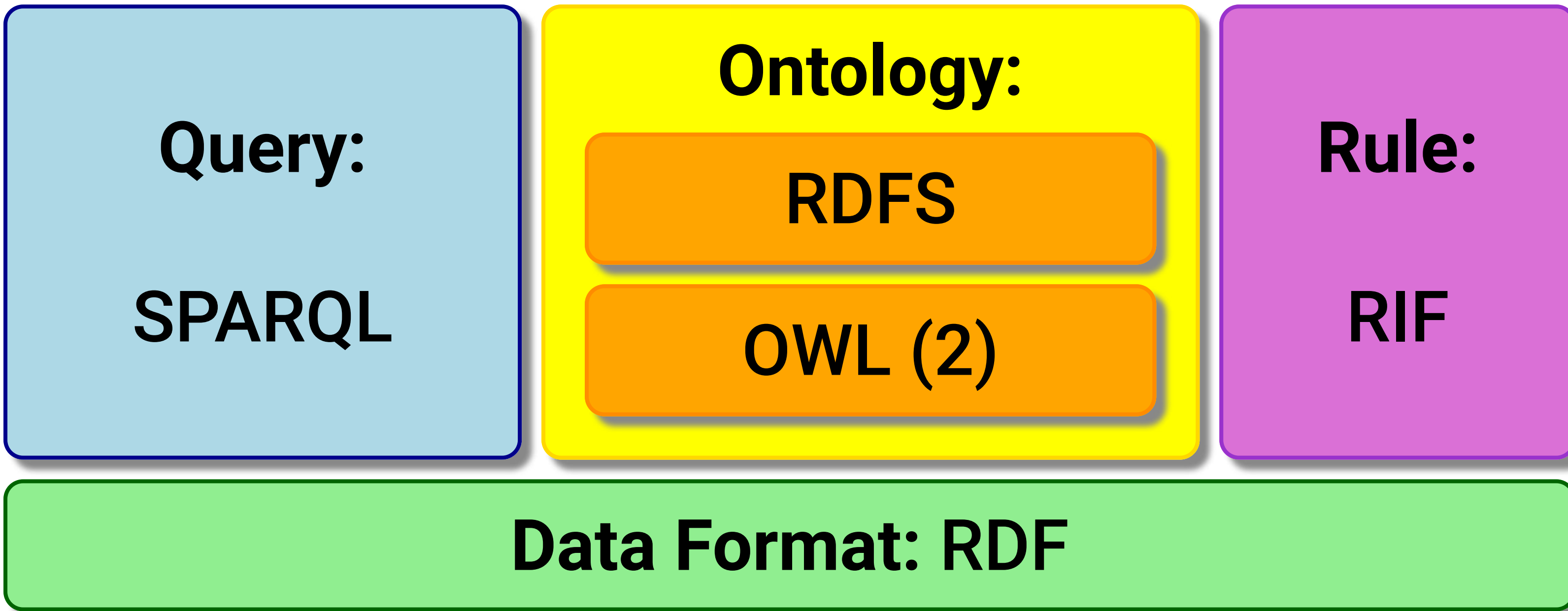
```
:article   rdfs:subclassOf bench:doc
:article1  rdf:type        :article .
:article1  dc:creator      :person1 .
:person1   foaf:name       'Martin' .
:person1   :likes          :person2 .
:person2   foaf:name       'Jennifer' .
:person2   :hates          :person1 .
```



Unstructured Data:

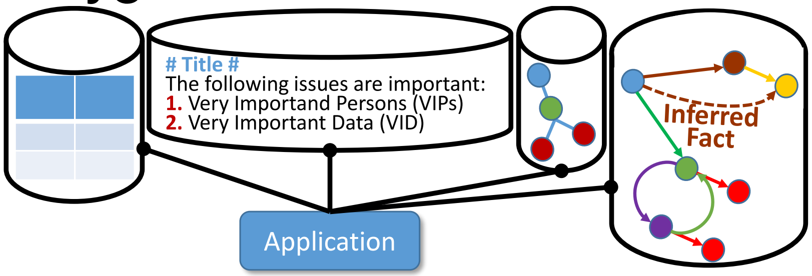
Title
The following issues are important:
1. Very Importand Persons (VIPs)
2. Very Important Data (VID)

Semantic Web (Core) "Standards"



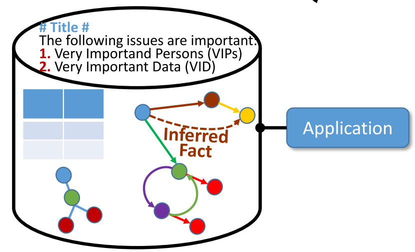
- Every data model (here Semantic Web) has its own set of languages (data, query, rule, ...)

Polyglot Persistence



- data sources: integration at application level
- performance of data processing cannot be fully optimized
- fault-tolerance cannot be transparently offered across the different databases
- zoo of query languages
- features of different types of databases can be used

Multi-Model DBMS (MM-DBMS)

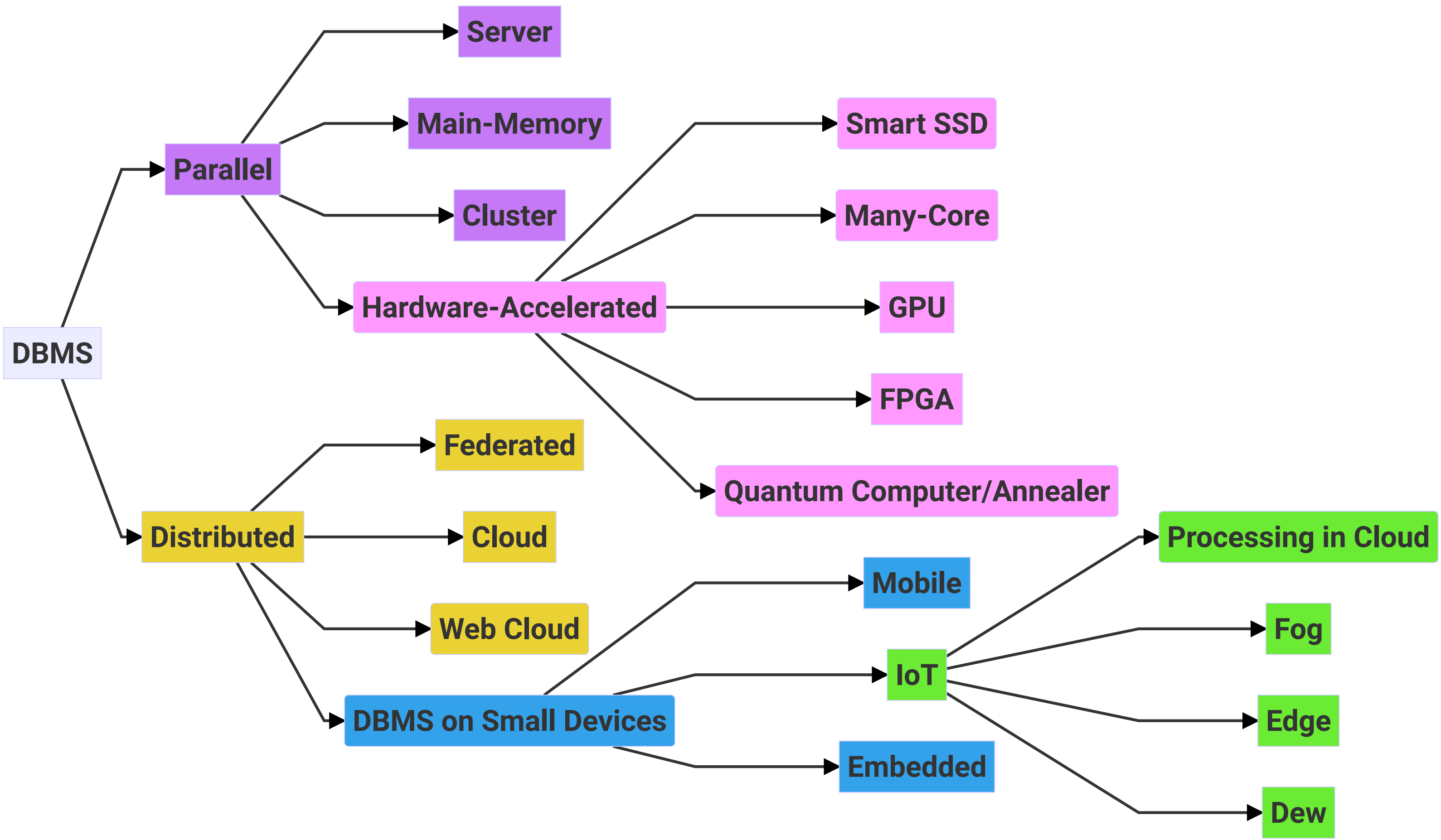


- full and uniform data integration at database level
- performance: fully optimized across different data models
- transparent fault-tolerance
- SQL standards:

relational ('87), XML ('03), temporal ('11), JSON ('16), Multi-dimensional Arrays ('19), schemaless ('19), streams ('20?), property graphs ('21?)

- features of different types of databases can**not** be used

Platform-specific types of DBMS

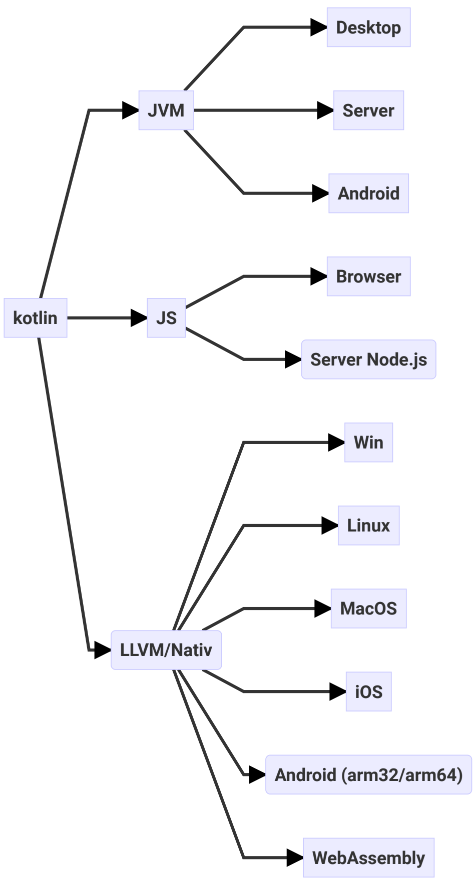


Multi-Platform Development of DBMS

- Native Binaries via C/C++
  - support of a new platform: porting code is necessary
  - code close to hardware, fast execution
  - direct access to native libraries
  - doesn't run in browser
  - most server DBMS: C/C++ code
- Java/Java Virtual Machine (JVM)
  - runs on many platforms (without porting code)
  - interpreted bytecode, via Just-In-Time compilation comparable speed to native execution
  - no direct access to native libraries
  - doesn't run on iPhone, in browser
  - many NoSQL/NewSQL/Cloud DBMS: Java (or JVM language like Scala) code
- Code generation for query processing via C/C++ or Janino-Compiler (JVM)

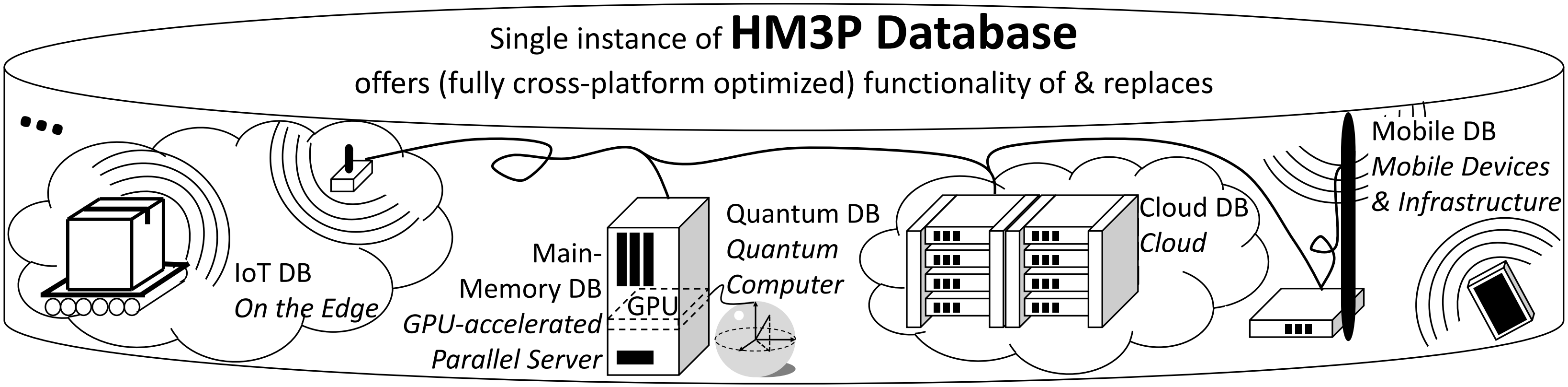
Multi-Platform Development with Kotlin

Targets:



- Most target platforms are supported
- Splitting the project in platform-independent and platform-dependent code
  - Platform-dependent code can be partly coded in the programming language of the target platform (e.g., Java for JVM, JS for Web)
- Enables one code repository for various target platforms
  - Sharing of code between server & (various) clients
- Avoids efforts to port code (into other programming languages)

Hybrid Multi-Model Multi-Platform (HM3P) Database

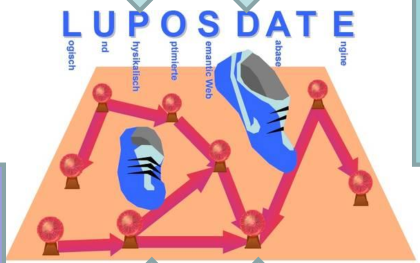


- full and uniform data integration at database level
- performance: fully optimized across different data models
- transparent fault-tolerance
- SQL standards: relational ('87), XML ('03), temporal ('11), JSON ('16), Multi-dimensional Arrays ('19), schemaless ('19), streams ('20?), property graphs ('21?)
- **features of different types of databases running on different platforms can be used**

Port of LUPOSDATE (Java) to LUPOSDATE3000 (Kotlin)

Support of:
- SPARQL Queries
- RIF Rules
- RDF Schema
- OWL (via OWL2RL in RIF)

Indexing:
- Stream Processing
- Main memory for small datasets
- Disk-based for large datasets
    - RDF3X
- Cloud: HBase
- P2P



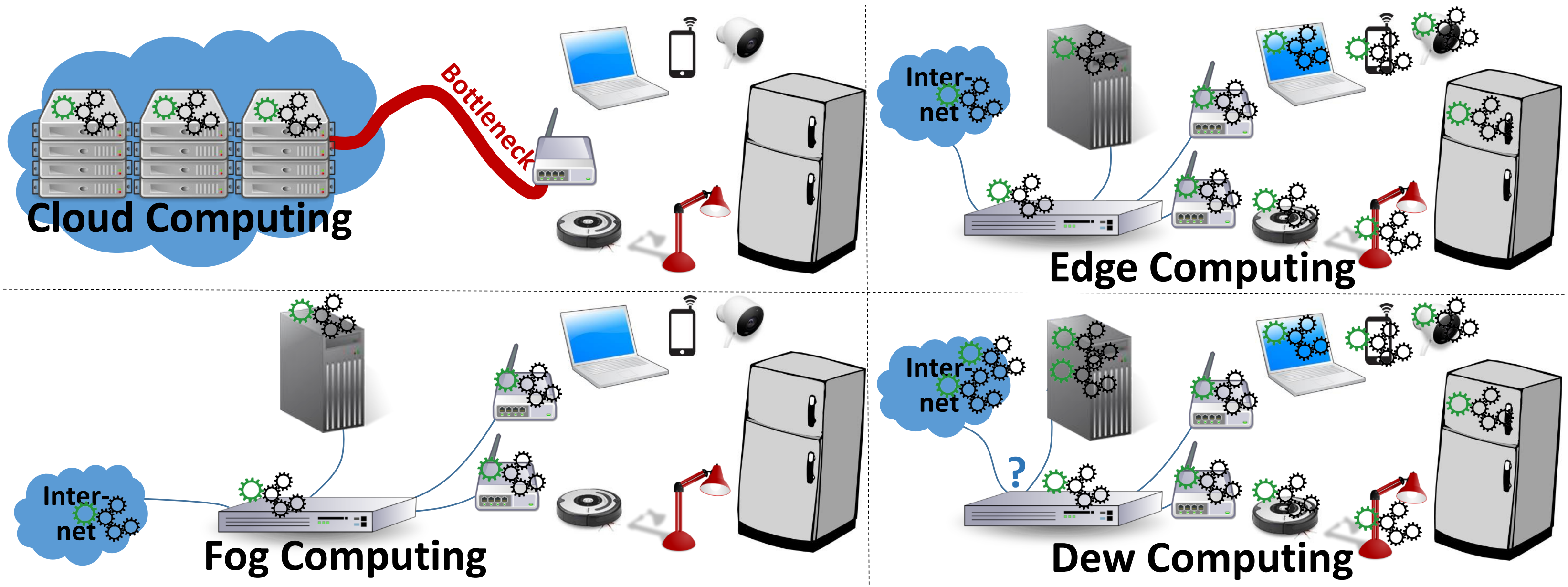
Visualizations:
- Visual Editor
    - Queries (SPARQL)
    - Rules (RIF)
    - Data (RDF) in
        - 2D and
        - 3D
    - Logical Optimization Rules
- Summaries of RDF Data
- Operator graph
- Processing of Queries and Rules
- Optimization Steps

Extra:
- Parallel Processing
- Distributed Processing
- Cloud Computing
- Mobile Computing
- P2P for Internet of Things
- Compression of RDF Data
- Embedding of SW Languages in Programming Languages
- Speeding up by FPGAs

What is working...

- RDF data import
- SPARQL query processing
  - local/parallel
    - column-based query processing
  - distributed in cluster
- Multiple targets (JVM/JS/native/...)

IoT Architectures

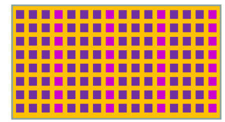
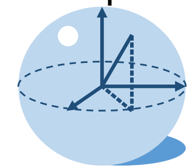


- Enable LUPOSDATE (not only) for IoT architectures

Topics of thesis in this context

- Code Generation
- Machine Learning
  - Query Optimization in Distributed Environments
- Continuous Queries
  - Redundant query processing
  - efficient recovery after crashes
- Distributed RDFS Inference
- Concurrency Control
- Hybrid Multiplatform Approaches
- Parser-Generator for parsing Big Data
  - inlining, avoiding unnecessary object creation, reusing objects

Architectures of Emergent Hardware

	Multi-Core CPU	Many-Core CPU	Graphics Processing Unit (GPU)	Field Programmable Gate Arrays (FPGA)	Quantum Computer/Annealer
					
Cores	~10	~100	~1000	~100 000	~50 / ~2000 qubits
Core Complexity	Complex (optimized for single thread performance)			Simple	Complex
Computational Model	MIMD + SIMD		SIMD	Data-Flow	Universal/Adiabatic Quantum computing
Parallelism	Thread and Data Parallel		Data Parallel	Arbitrary	Quantum Logic Gates/ Fluctuation (glob. Min.)
Memory Model	Shared	Distributed			Quantum Superposition
Power	150 W	200 W	250 W	50 W	25 KW
Database Op.	Query Optimization (Enumeration of Plans), Concurrency Control				
	Efficient Processing Of				
	Query Processing				

Legend: Computational Unit, Execution Controller, Interconnection Network, On-Chip Memory

High-End Parallel GPU System: DGX-2

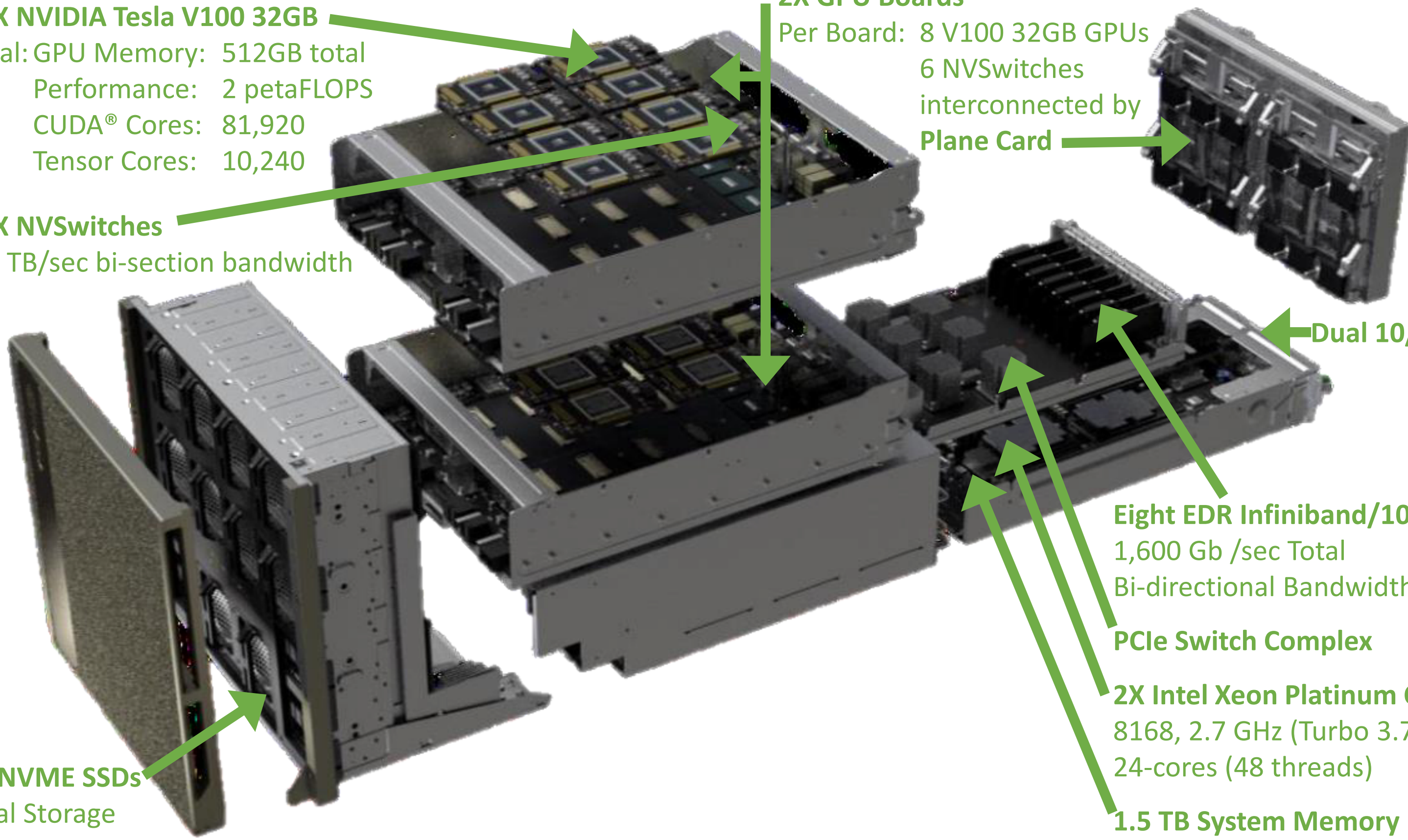
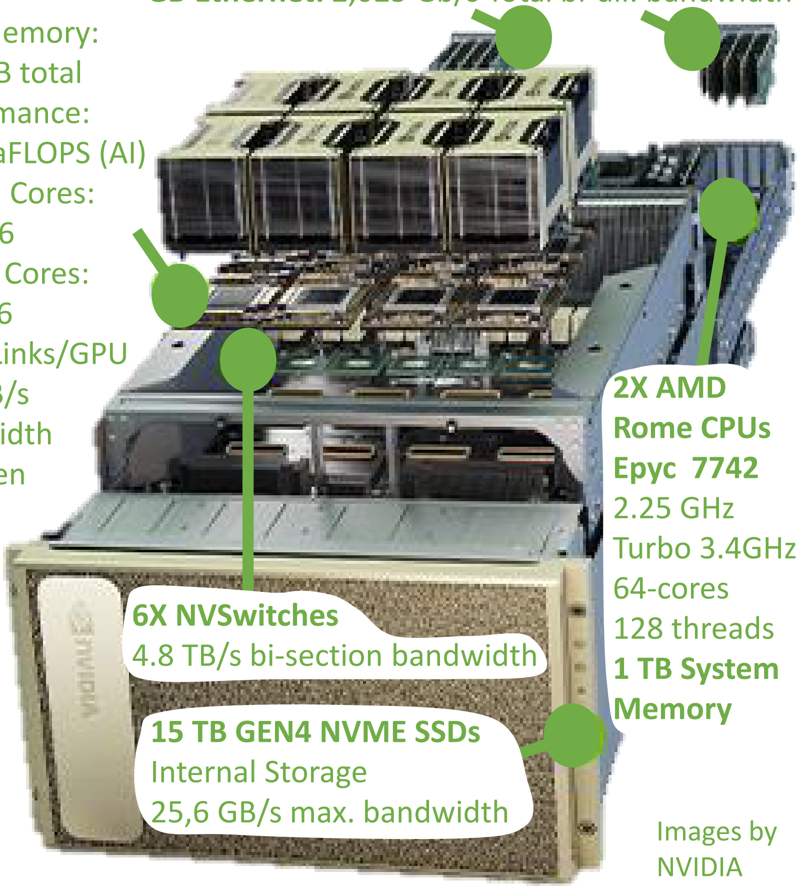


Image by NVIDIA

High-End Parallel GPU System: DGX A100

8X NVIDIA A100
Total:
GPU Memory:
320GB total
Performance:
5 petaFLOPS (AI)
CUDA® Cores:
55,296
Tensor Cores:
3,456
12 NVLinks/GPU
600 GB/s
bandwidth
between
GPUs

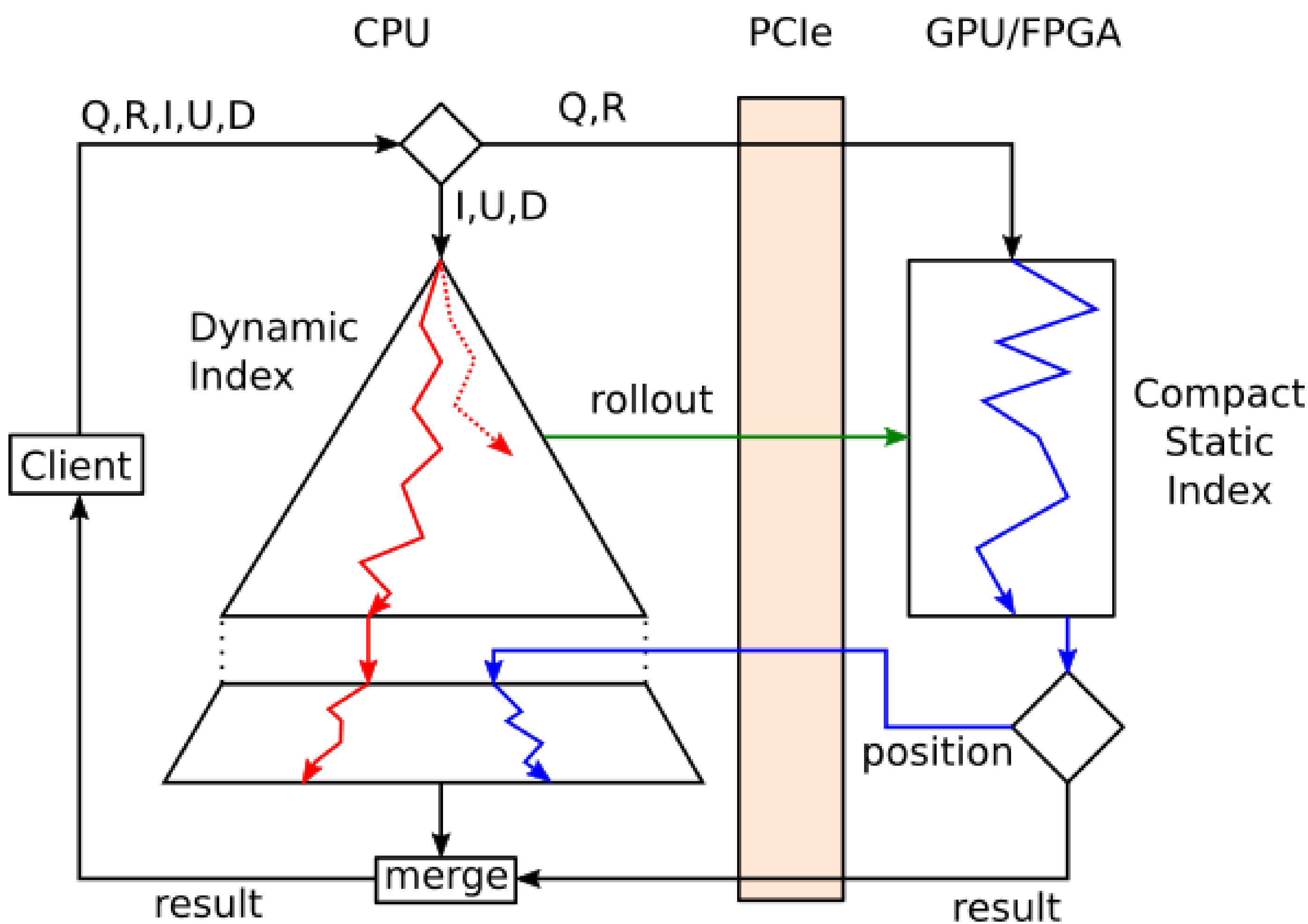
9 Mellanox ConnectX-6 VPI HDR InfiniBand/200 GB Ethernet: 2,025 Gb/s Total bi-dir. bandwidth



2X AMD Rome CPUs Epyc 7742
2.25 GHz
Turbo 3.4GHz
64-cores
128 threads
1 TB System Memory

Images by NVIDIA

Architectures for Hybrid Indices (Variant 1)



Architectures for Hybrid Indices (Variant 2)

Good News

- Framework for Hybrid Indices including Benchmark Suite available
  $\rightsquigarrow$ Tobias
- Offered topics deal with
  - integrating suitable/develop new indices in this hybrid approached

Online Resources

- *"Always search for yourself!"*
- OpenCL
  - Matthew Scarpino: A Gentle Introduction to OpenCL (Dr.Dobb's)
  - David Gohara: Episode 1 - Introduction to OpenCL (YouTube)
- Filter based on Hashing
  - Bloom: Paper Tutorial Code
  - Cuckoo: Paper Code 1 Code 2
  - Morton: Paper Code
  - Vacuum: Paper Code
- Search Trees
  - B/B+-tree: Paper Code
  - CSB/CSB+: Paper Code
  - B^{ed}-Tree: Paper Code
  - Masstree: Paper Code
- Learned Index: Paper Code
- Hybrid Index: Paper

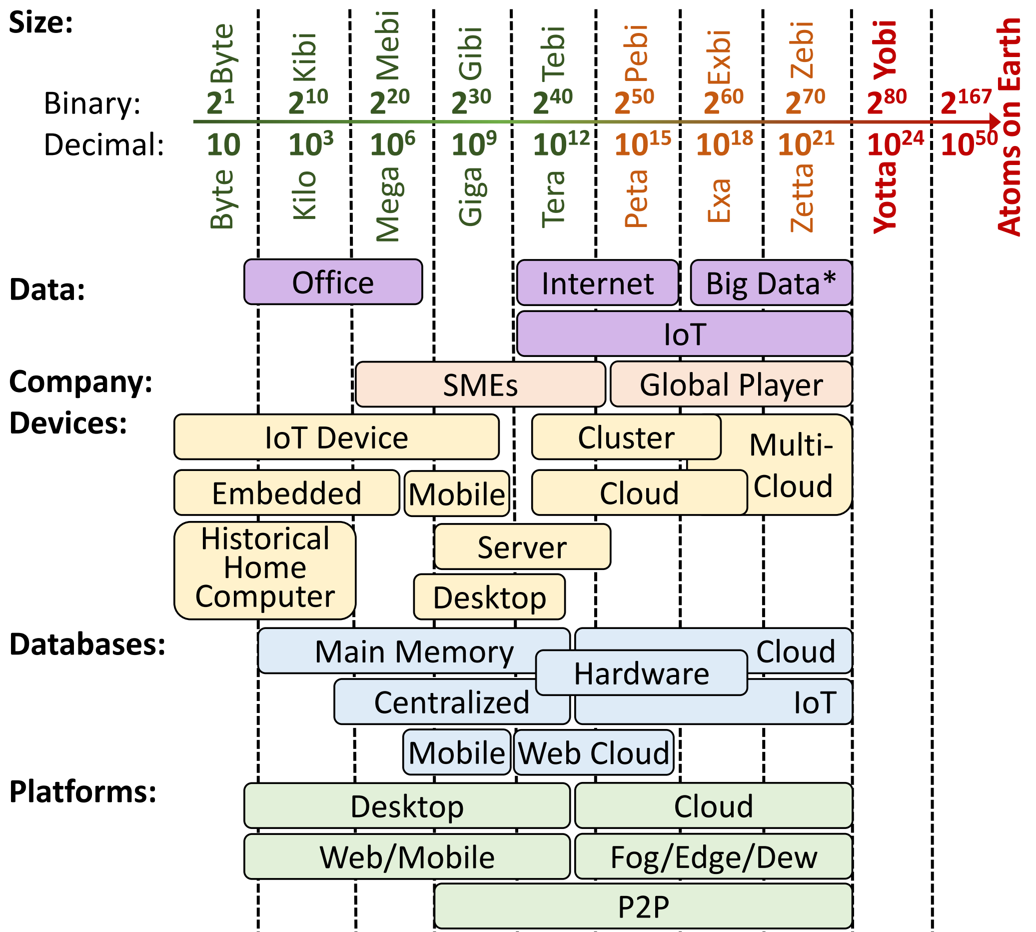
Concurrency Control for several 10,000 cores

- Paper Code
- Approach e.g. for enumerating parallel transaction schedules

Quantum Computing

- Programming Languages and Tools are now
  - on the move "to be used by computer scientists (rather than by physician)"
  - follow more the computer scientists' style of developing software
- More details: Our Paper Presentation
- Offered Topics, e.g.
  - Concurrency Control
  - Query Optimization

Data Sizes

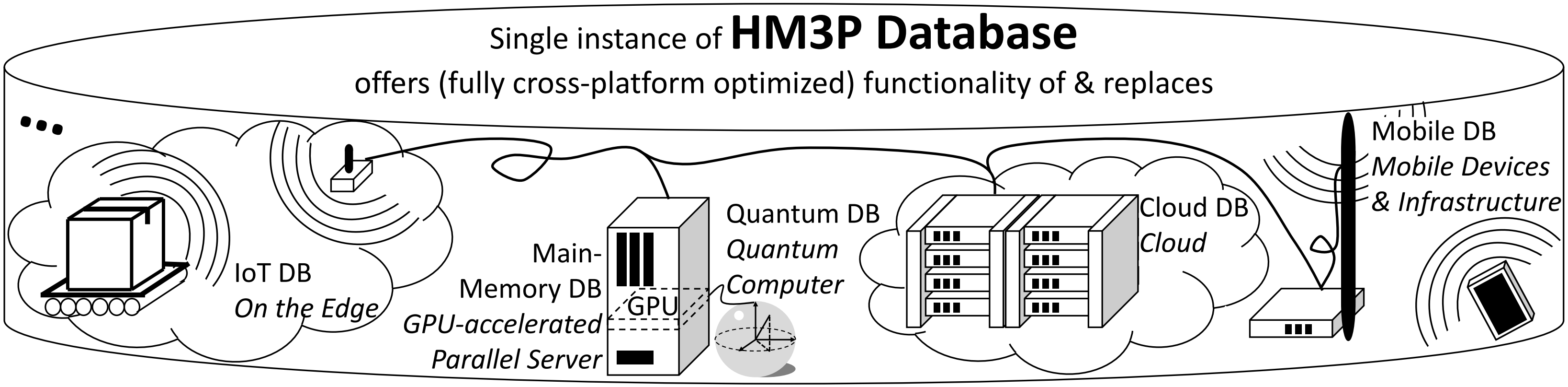


SMEs: Small and medium-sized enterprises * social media, search engines

Features of different types of databases

Feature \ DBMS	Main Memory	Parallel	Distributed	Federated	Cloud	Web Cloud	Mobile	IoT
Scalability	⊖	⊙	⊕	⊕	⊕⊕	⊕⊕⊕	⊕	⊕⊕
Transaction rates	⊕⊕⊕	⊕⊕	⊙/⊕	⊙	⊕⊕	⊕	⊖	⊖⊖
Intra-Transaction Parallelism	⊕⊕⊕	⊕⊕	⊙/⊕	⊖/⊙	⊕	⊙	⊖	⊖
Atomicity	⊕⊕⊕	⊕⊕⊕	⊕⊕	⊕	⊕	⊕	⊕	⊕
Durability	⊕	⊕	⊕⊕⊕	⊕⊕	⊕⊕	⊖	⊙	⊖
Consistency	⊕⊕⊕	⊕⊕⊕	⊕⊕	⊕	⊕	⊕	⊕	⊕
Extensibility	⊖	⊕	⊙/⊕	⊙	⊕⊕	⊕⊕⊕	⊖	⊕⊕⊕
Schemaless	⊖⊖⊖	⊖⊖⊖	⊖⊖⊖	⊖	⊕⊕⊕	⊕⊕⊕	⊕	⊕⊕⊕
Availability	⊕⊕	⊕	⊕	⊖	⊖	⊖⊖⊖	⊖⊖	⊖⊖⊖
Transparency of Distribution	⊕⊕	⊕⊕	⊕	⊙	⊕⊕	⊖	⊖	⊖⊖
Geographical Distribution	⊖⊖	⊖	⊕	⊕	⊕⊕	⊕⊕⊕	⊕⊕	⊕⊕
Mobility	⊖	⊖	⊖	⊙	⊙	⊙	⊕⊕	⊕
Node Autonomy	⊖⊖	⊖	⊙	⊕	⊙	⊖⊖	⊕⊕	⊕
Heterogeneity of DBMS	⊖⊖	⊖	⊖	⊕	⊖	⊖	⊕⊕	⊕⊕⊕
Administration	⊙	⊙	⊖	⊖/⊖⊖	⊖	⊕⊕	⊖⊖	⊖⊖⊖
Hardware Costs	⊖	⊖⊖	⊖	⊖	⊕⊕	⊕⊕⊕	⊖	⊕⊕⊕

Hybrid Multi-Model Multi-Platform (HM3P) Database



- How to integrate the features of different types of databases into one single database running also on different platforms?

Challenges for HM3P Databases 1/2

- developing only one code base for the different platforms, but not introducing performance overhead in comparison to single platform databases
- identifying common properties of several platforms and reusing those approaches (like fault tolerance mechanisms) in different combinations, which are best suitable for these considered platforms
- data distribution among different platforms (applying different data distribution approaches as well)
- data distribution strategies considering overall the different properties of used platforms and models (like fast reads on parallel servers (using relational databases) and fast updates in cloud databases)

Challenges for HM3P Databases 2/2

- query optimization and other database tasks across different platforms, which apply different database approaches
- concurrency control approaches of different type have to be combined and work in cooperation (like 2 phase locking for server platforms and optimistic concurrency control for P2P networks)
- combining different types of databases (on different platforms) to offer the best of these databases and platforms *under one hood* to applications and users transparently or via intelligent integration into query language and API, e.g.,
  - guaranteeing atomicity and isolation in transactions for the data stored on a parallel server, but not for those data in the cloud supporting fast updates

Summary and Conclusions

- Different data models and their special features
  - ➡ Multi-Model Databases
- Different platforms and a need for different types of databases
  - Different features
  - ➡ Multi-Platform Databases
- Databases spanning over different platforms in operation (supporting multiple data models)
  - ➡ Hybrid Multi-Model Multi-Platform (HM3P) Databases
- Emergent Technologies like IoT, GPU, Quantum Computing
- We are offering **many topics for bachelor/master thesis**...
  - Please contact me: groppe@ifis.uni-luebeck.de

Contact Persons IFIS

- Prof. Sven Groppe: Utilizing Emergent Technologies for Big Data Analytics
  - Benjamin Warnke: IoT, Hybrid Multi-Model Multi-Platform Databases, Query Processing, Parser-Generator
  - Tobias Groth: GPU, Hybrid Index in CPU/GPU-Systems
- Other topics (not in this talk!)
  - Prof. Ralf Möller: Dynamic Probabilistic Relational Models, Probabilistic Computing
  - PD Özgür L. Özçep: Logical and mathematical Modeling for Knowledge Representation and Processing and for Machine Learning