Adaptive Architectures for Robust Data Management Systems

Tiemo Bang[1]

Abstract: "Form follows function" is a well-known expression by the architect Sullivan asserting that the architecture of a building should follow its function. "Adaptive Architectures for Robust Data Management Systems" is a dissertation asserting that DBMS architectures should follow changing workload and hardware to robustly achieve high DBMS performance. The dissertation first evaluates how workload and hardware affect the performance of DBMSs with static architectures. This evaluation concludes that static DBMS architectures degrade DBMS performance under changing workload and hardware, and hence the DBMS architecture has to become adaptive. Subsequently, adaptation concepts for the architecture of single-server and multi-server DBMSs are proposed. These concepts focus fine-grained adaptation of DBMS architectures and are realized through asynchronous programming models. These programming models decouple the implementation of DBMS components from fine-grained architectural optimization. Thereby, optimizers can derive novel architectures better fitting individual DBMS components, leading to high and robust DBMS performance under changing conditions.

Keywords: Database Management System; Architecture; Adaptation; Scale-Up; Scale-Out

Summary of "Adaptive Architectures for Robust Data Management Systems"

"Unceasingly the essence of things is taking shape in the matter of things. [. . .] It is the pervading law of all things organic and inorganic, of all things physical and metaphysical, [. . .] that form ever follows function. This is the law. Shall we, then, daily violate this law in our art?" — Sullivan, 1896 in *The Tall Office Building Artistically Considered*

Sullivan asserts that the architecture of a building should follow from its function. The dissertation "Adaptive Architectures for Robust Data Management Systems" [Ba22a] asserts the same for the architecture of a database management system (DBMS). It particularly asserts that changing workload and hardware for DBMSs necessitate changing (adaptive) DBMS architectures. The dissertation contributes to adaptive DBMS architectures through publications on the performance of DBMSs with static architectures [Ba22b, Ba20a] and concepts for the adaptation of DBMS architectures for single-server DBMSs [Ba20b] and multi-server DBMSs [Ba21]. The following provides a brief summary of the overall work.

[1] University of California Berkeley, EECS, 1860 Le Roy Ave., CA 94709 Berkeley, United States of America
tbang@berkeley.edu

doi:10.18420/BTW2023-33

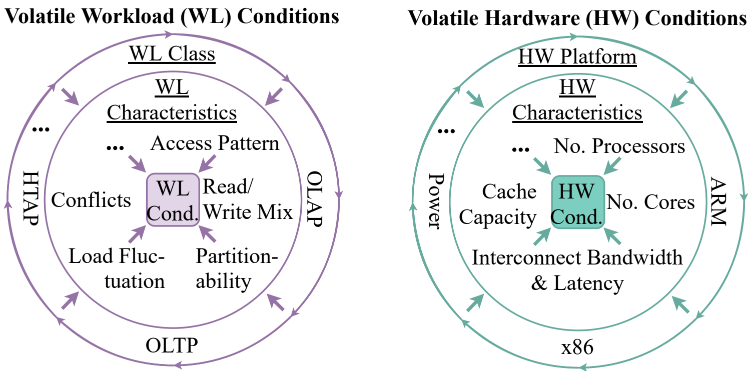

Fig. 1: Overview of the volatile conditions challenging DBMS performance. DBMSs are exposed to a range of workload classes and hardware platforms. These influence the displayed workload and hardware characteristics and lead to challenging volatile conditions under which DBMSs still need to perform well. The DBMS design has to align with these changing characteristics, otherwise performance can degrade significantly.

DBMSs are challenged by changing workloads and hardware. Today's DBMSs handle various types of workloads for online commerce, banking, and fraud detection. These workloads differ in key characteristics like the read-write pattern and further fluctuate depending on the popularity of individual data items. Similarly, modern DBMSs are typically compatible with a wide range of hardware platforms which still have different characteristics like different types or numbers of processors. This presents DBMSs with the challenge to perform well despite workload and hardware characteristics that widely vary, possibly at runtime.

The performance evaluation of this dissertation, along with a review of related work, identifies complex effects of various workload and hardware characteristics on the performance of DBMSs with static architectures, cf. [Ba22b]. As summarized in Figure 1, the different workload classes and hardware platforms lead to a wide range of characteristics that make for volatile workload and hardware conditions a DBMS design has to cope with. The evaluation surfaces a complex interaction between these characteristics with which the DBMS components and the surrounding DBMS architecture have to align precisely, otherwise performance can degrade sharply. Accordingly, the conclusion is that robust performance under changing conditions requires precise adaptation of the entire DBMS design, including the DBMS architecture.

DBMSs remain tethered to their static architecture. Serving changing workloads and supporting diverse hardware platforms is non-trivial for DBMSs. The design of the DBMS components and the DBMS architecture have to fit the conditions. DBMS components, like the query execution engine, indeed have developed adaptation mechanisms for that reason.

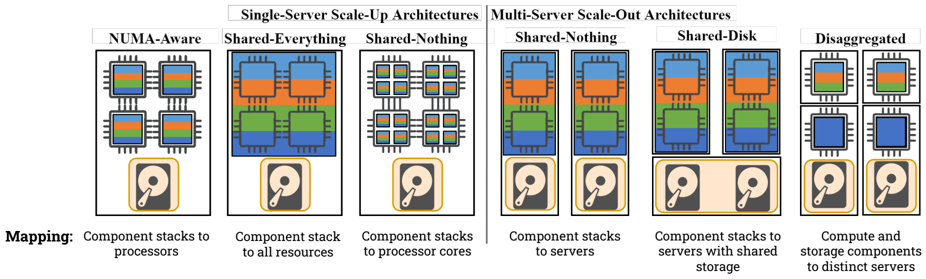
Single-Server Scale-Up Architectures
Multi-Server Scale-Out Architectures
NUMA-Aware
Shared-Everything
Shared-Nothing
Shared-Nothing
Shared-Disk
Disaggregated
Mapping:
Component stacks to processors
Component stack to all resources
Component stacks to processor cores
Component stacks to servers
Component stacks to servers with shared storage
Compute and storage components to distinct servers

Fig. 2: Overview of the established DBMS architectures for single-server and multi-server DBMSs with static deployment strategies, indicated by the colored boxes. Each architectures follows a static deployment strategy that determines a resource partitioning (shown by the sizes of the boxes) and deployment of DBMS components (shown by the colors the boxes). Most architectures uniformly deploy a stack of each DBMS component onto the resource partitions.

Mechanisms like just-in-time query compilation, for example, allow the query execution engine to flexibly specialize its operators to different workload or hardware. However, such adaptation is limited to the internals of these DBMS components. There is no adaptation outside of or across DBMS components, as the surrounding DBMS architecture is inflexible.

Today DBMS designers select from a handful of static architectures with rigid deployment strategies. As shown in Figure 2, these static architectures predetermine a fixed resource partitioning. The DBMS components are then deployed onto these resource partitions, as stacks containing an instance of each component. For example, the NUMA-aware architecture accounts for multi-processor hardware through resource partitions per processor onto which the DBMS components are deployed. Such tailoring, however, is a static design decision. This decision at design-time leads to an architecture baked into the DBMS implementation that only fits predetermined workload and hardware. Moreover, this manual decision also causes oversimplification. That is, all the static architectures uniformly deploy DBMS components onto coarse-grained resource partitions ignoring the unique workload and hardware effects within individual DBMS components. As a result, current static DBMS architectures are not flexible and lack sophistication, such that these architectures become unfit and degrade DBMS performance.

Adaptive DBMS Architectures

The overall idea for the adaptation of DBMS architectures is to flexibly compose fine-grained "building blocks" of the DBMS to a best-fit architecture. This dissertation emphasizes fine-grained adaptation of the architecture to suit the individual internal functions of DBMS components. Besides a relief from the re-implementation effort, the goal is to create a navigable optimization space for DBMS architectures, enabling optimizers to flexibly mimic any existing architecture and more importantly to derive entirely new architectures.

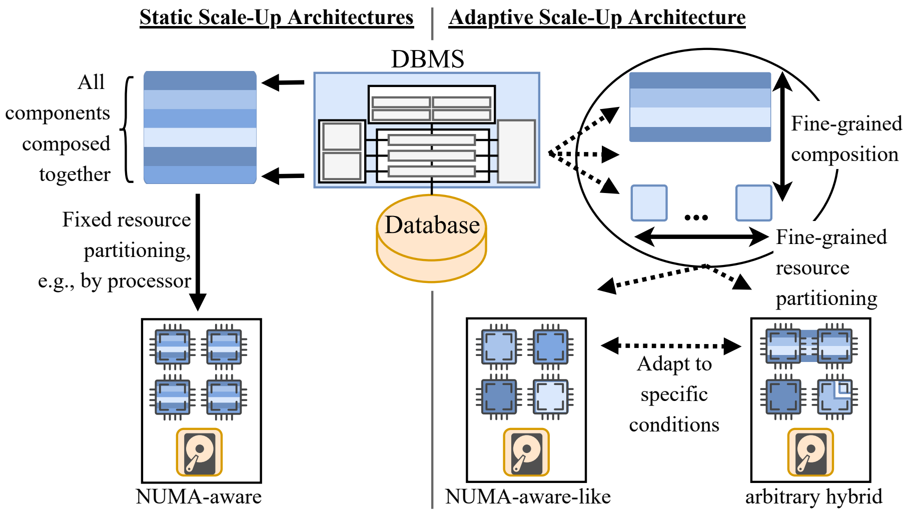
Static Scale-Up Architectures
Adaptive Scale-Up Architecture
DBMS
All components composed together
Fixed resource partitioning, e.g., by processor
Database
Fine-grained composition
Fine-grained resource partitioning
Adapt to specific conditions
NUMA-aware
NUMA-aware-like
arbitrary hybrid

Fig. 3: Designing static scale-up architectures (left) versus proposed fine-grained adaptation (right). Left: Current architectures are statically designed with static deployment strategies for specific hardware/workload and treat all DBMS components uniformly, e.g., the NUMA-aware architecture dictates the resources partitioning by processor which each contain the complete stack of components. Right: The proposed adaptive scale-up architecture enables fine-grained arbitrary resource partitioning for arbitrary compositions of DBMS components, facilitating best-fit architectures.

Adaptive Single-Server DBMS Architecture. The adaptive single-server architecture introduces flexible configuration for DBMSs on shared-memory hardware, cf. [Ba20b]. It addresses the distinct adaptation demands on a single server, orthogonal to adaptation across multiple servers.

The key factors for single-server DBMS architectures to address are shared-memory data structures and the constraint to fixed resources. Single-server DBMS architectures must carefully organize these limited resources, especially in regard to efficient concurrent execution on shared data structures. For example, the static NUMA-aware architecture employs processor-sized resource partitions to enforce processor-local concurrent execution of DBMS components and explicit coordination of DBMS components across processors. Generally, single-server architectures strive for a resource partitioning that effectively balances partition-local concurrent execution on shared data structures within DBMS components and the explicit coordination of components across resources partitions. Static DBMS architectures strike this balance only for specific predetermined workloads and hardware, but likely become unfit under changing conditions. Instead, the proposed adaptive architecture enables flexible adaptation at the granularity of individual data structures of DBMS components, for best-fit single-server architectures across changing workloads and hardware.

The proposed adaptive single-server architecture introduces abstractions for the generic implementation of DBMS components and the fine-grained configuration of the architecture. It introduces a programming model of asynchronous data-aware tasks for DBMS components. These data-aware tasks divide DBMS components into units of execution on individual data structures, enabling the fine-grained configuration of the DBMS architecture. As shown in Figure 3, this configuration enables optimizers to flexibly configure arbitrary architectures with heterogeneous resource partitions tailored to individual data structures (or composites thereof). In contrast to the static "prepackaged" architectures, optimizers thereby can form novel architectures best-fit for each data structure under changing workload and hardware. The benefits of this flexible configuration are demonstrated through an optimizer for automatic throughput maximization.

Adaptive Multi-Server DBMS Architecture. The adaptive multi-server architecture offers flexible orchestration of the DBMS across elastic network-connected resources, cf. [Ba21]. It complements the above adaptation within a single server.

The key factors for multi-server architectures to address are the network communication between resources and the flexible addition/removal (scaling) of resources. Especially in the cloud, multi-server DBMSs can utilize an elastic resource pool to maintain high performance when facing heterogeneous and fluctuating workloads. Yet, at the same time, the network communication across this resource pool also poses a significant challenge. Multi-server architectures therefore aim to balance resource load and network communication overhead when orchestrating the DBMS across this resource pool. Static multi-server architectures, however, strike this balance only for specific workloads, e.g., the shared-nothing architecture for partitionable workloads. Changing or mixed workloads and the properties of individual DBMS components are not well reflected. For example, analytical queries and the query executor component can generally utilize more resources than transactional queries and the transaction manager components.

The adaptive multi-server architecture introduces individually optimized architectures for simultaneous queries. Rather than compromising on a common architecture, it simultaneously orchestrates query-optimized architectures across elastic resources. It defines a reactive programming model with generic DBMS actors for that reason. These actors are flexibly instrumented to temporarily enact (parts of) DBMS components and together enact a DBMS architecture. The execution and data flow of the DBMS are essentially broken down into streams of events and data items that can be routed across the resource pool. As illustrated in Figure 4, the routing and interleaving of these event and data streams therefore enables the simultaneous orchestration of multiple query-optimized architectures. The dissertation focuses on exploring the new degrees of freedom offered by this adaptive multi-server architecture.

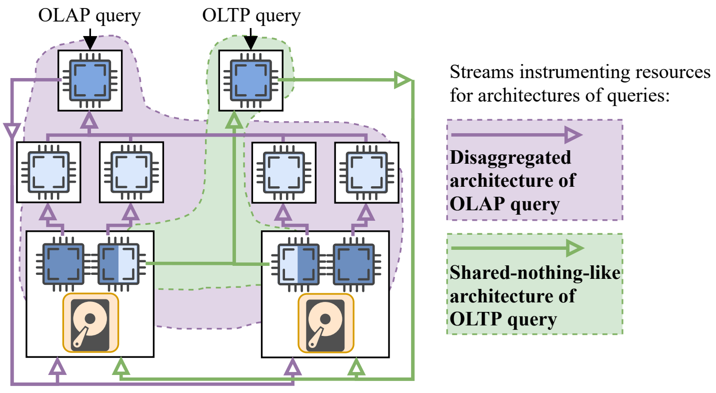
OLAP query
OLTP query
Streams instrumenting resources for architectures of queries:
Disaggregated architecture of OLAP query
Shared-nothing-like architecture of OLTP query

Fig. 4: The adaptive multi-server DBMS architecture for simultaneous optimization to distinct queries. The adaptive architecture instruments DBMS actors on elastic resources via event and data streams to enact DBMS components, allowing to orchestrate query-optimized architectures for the concurrent OLTP and OLAP queries. The purple streams orchestrate a disaggregated architecture for the OLAP query, while the green streams simultaneously orchestrate a shared-nothing architecture for the OLTP query.

Conclusion

The key findings are that both the realized adaptive single-server and multi-server architectures prove effective and efficient. Under changing transactional and mixed workloads, the proposed adaptive architectures generally perform at least on par with the individually best state-of-the-art architecture. Indeed, when adopting novel better-fit architectures, all existing architectures are outperformed, e.g., when partitioning resource at a granularity unlike any of the existing architectures or when distinctly orchestrating architectures for queries of mixed workloads. Overall, the proposed flexible and precise adaptation demonstrates higher and more robust performance.

While the findings exhibit novel better-fit architectures only for a subset of possible workload and hardware conditions, this dissertation overall indicates high potential for adapting architectures with the proposed concepts. As the proposed concepts make a vast optimization space generally navigable, optimizers will be able to adapt DBMS architectures flexibly and more precisely to many workloads and hardware. Instead of fragile static architectures, the proposed adaptive architectures thus provide a necessary element for DBMSs to achieve high and robust performance under changing workload and hardware.

Acknowledgments I would like to thank the many people who provided advise, companionship, and sponsoring. Special thanks go to Carsten Binnig whose steadfast guidance led to this dissertation.

Bibliography

[Ba20a] Bang, Tiemo; May, Norman; Petrov, Ilia; Binnig, Carsten: The Tale of 1000 Cores: An Evaluation of Concurrency Control on Real(ly) Large Multi-Socket Hardware. In: Proceedings of the 16th International Workshop on Data Management on New Hardware. ACM, 2020.

[Ba20b] Bang, Tiemo; Oukid, Ismail; May, Norman; Petrov, Ilia; Binnig, Carsten: Robust Performance of Main Memory Data Structures by Configuration. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20),. ACM, New York, NY, USA, 2020.

[Ba21] Bang, Tiemo; May, Norman; Petrov, Ilia; Binnig, Carsten: AnyDB: An Architecture-less DBMS for Any Workload. In: 11th Annual Conference on Innovative Data Systems Research (CIDR '21). 2021.

[Ba22a] Bang, Tiemo: Adaptive Architectures for Robust Database Management Systems. PhD thesis, Technische Universität, Darmstadt, 2022.

[Ba22b] Bang, Tiemo; May, Norman; Petrov, Ilia; Binnig, Carsten: The Full Story of 1000 Cores: An Examination of Concurrency Control on Real(ly) Large Multi-Socket Hardware. In: The VLDB Journal. 2022.