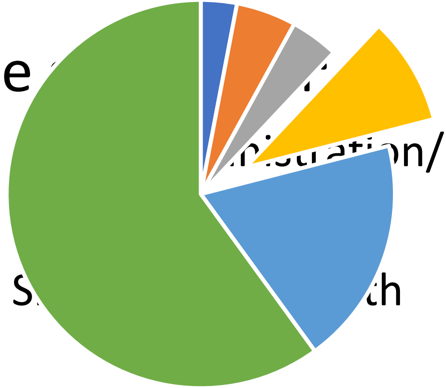
Nothing is for granted:
Making wise decisions using real-time intelligence

Anastasia Ailamaki

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

EPFL

Catching up With an Evolving Landscape



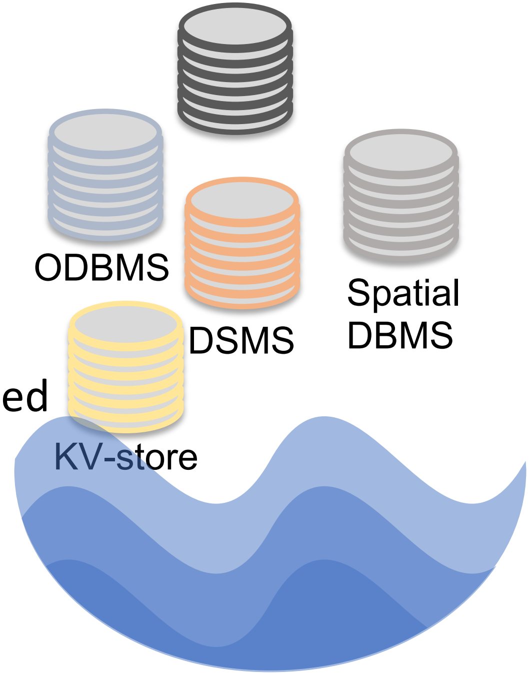
“Useful” time: 9%
(mining data for patterns)
Source: Forbes

One

- nistration/development
-
- S th

Modern requirements:

- OLAP, OLTP, Streams, federated data, ML, FaaS, …., mixed
- Custom & domain-specific types/operations
- Diverse requirements and priorities



Treat heterogeneity as a first-class citizen

Change Means Trouble

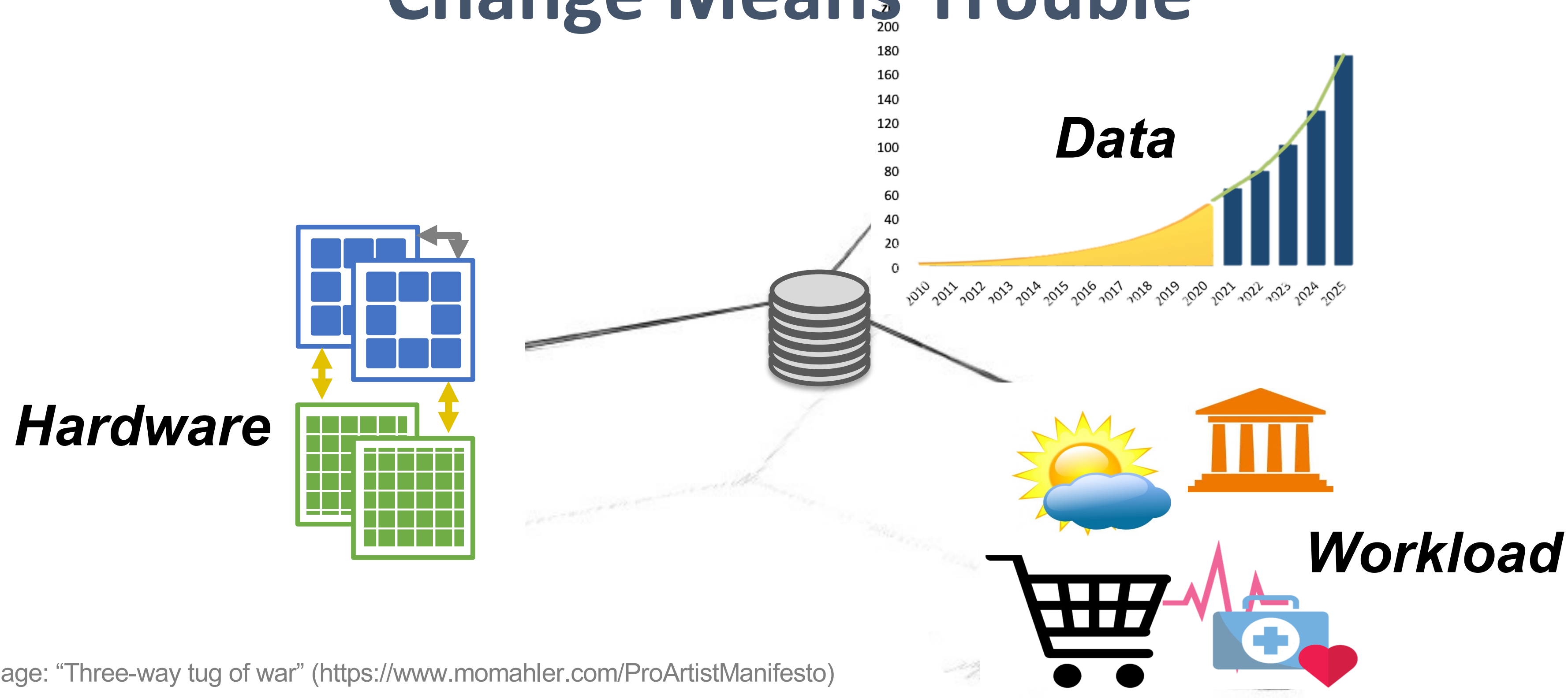


Image: “Three-way tug of war” (https://www.momahler.com/ProArtistManifesto)

Next generation systems must adapt

Change-driven Architectures

Data & operations hint to optimal architecture

Allow operations to shape DBMS architecture to their size

Reduce uncertain, preemptive work and fixed components

Capabilities

Composability

Modularity

Runtime

Specialization

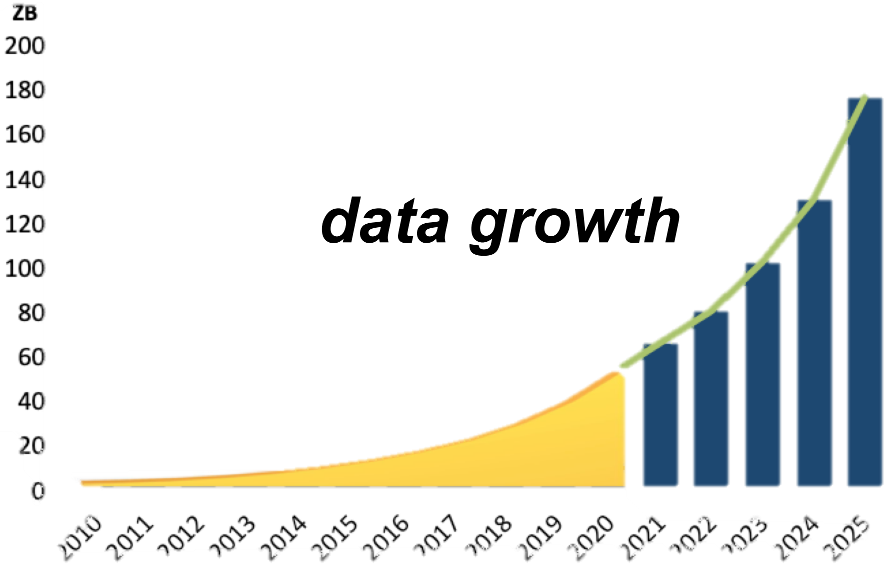
Each workload runs on its own custom DBMS

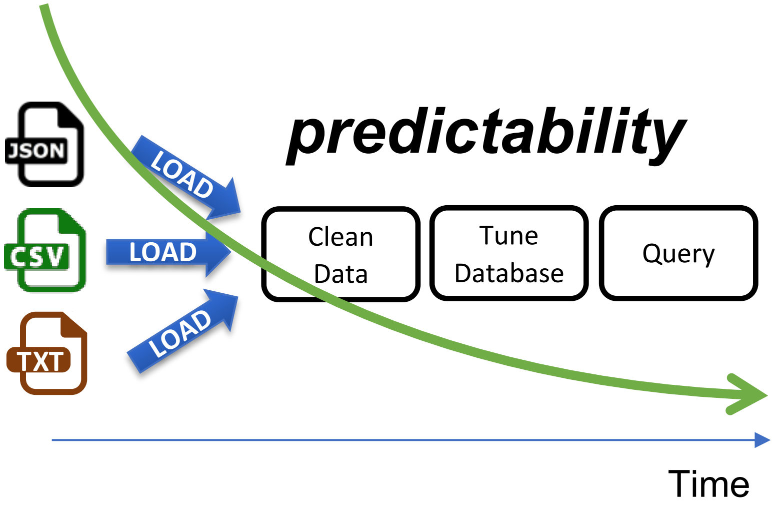
Runtime specialization embraces heterogeneity

Heterogeneous

Data	Unstandardized	Inconsistencies	Skewness
Workload	Uncertainty	Complexity	Volume
Hardware	Underutilization	Boundaries	Interference

From Data to Information Veracity



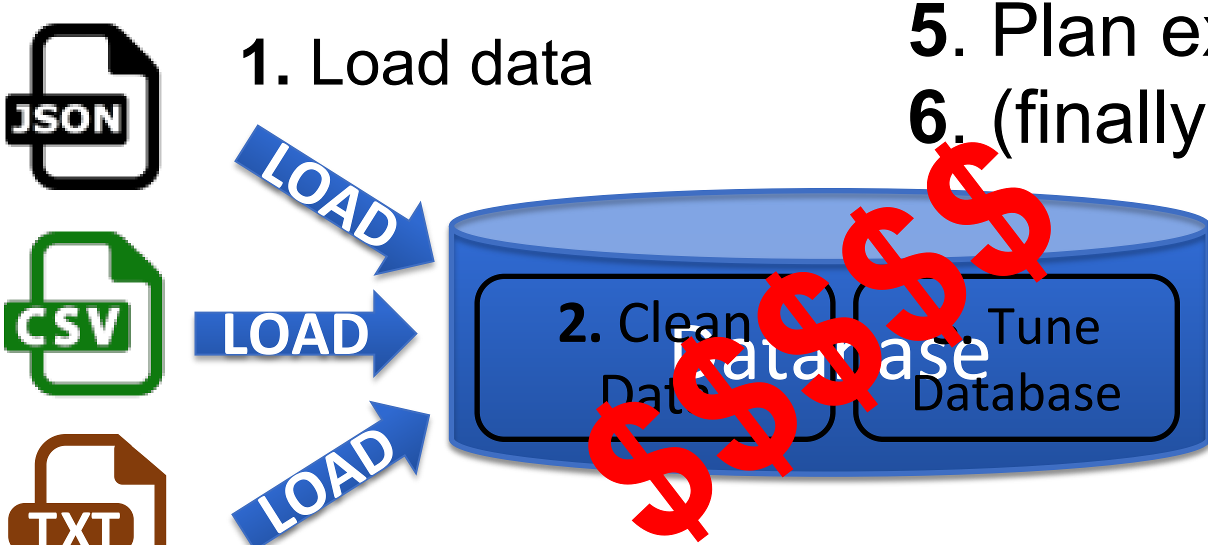


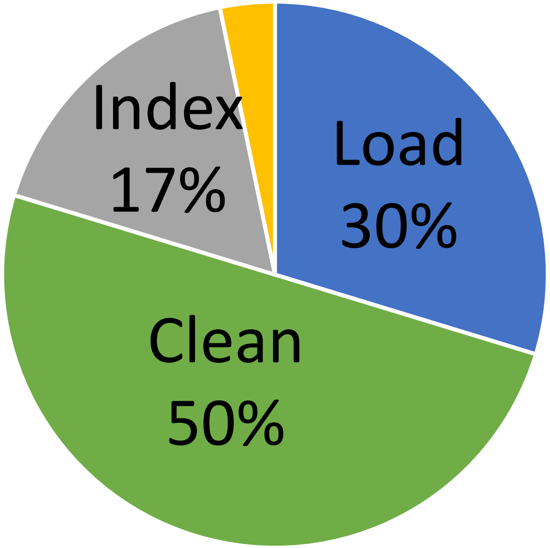
Need timely & informed data preparation

Preparation Kills Discovery

1. Load data

4. Ask analytical question
5. Plan e
6. (finally





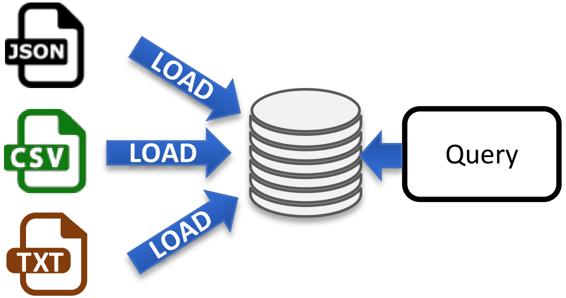
Cost grows with *owned* – not used! – data

Planning is expensive, often even wrong

Heterogeneous Data: Convert the Engine

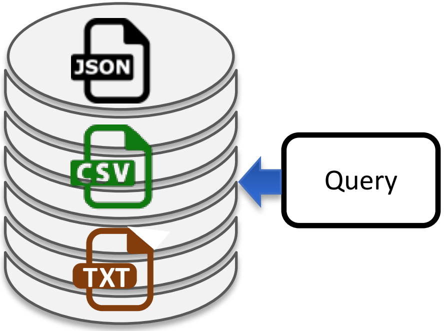
Conservative: pre-load & convert data

- Time-consuming data adaptation to engine
- Transformation: pre-determined order wrt execution
- Wasteful loading of write-only data



Real-time: customize access paths at runtime

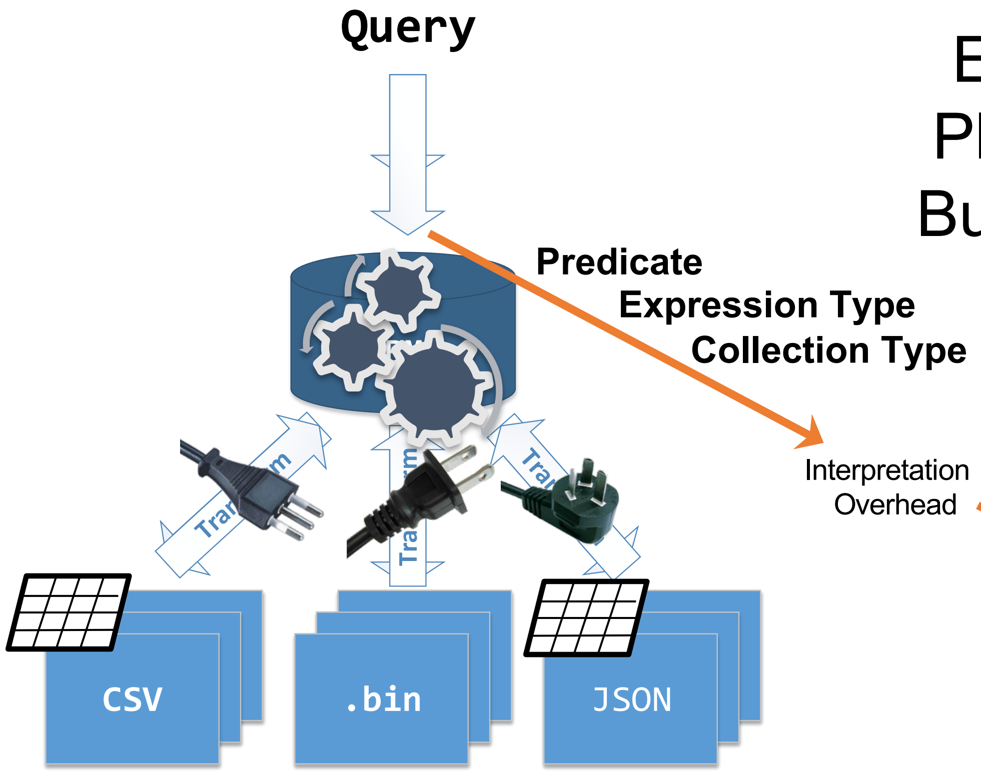
- Virtualize data; specialize access paths to queries+data
- On-the-fly access paths and positional caches
- Pay-as-you-go heterogeneous data accesses



Real-time adaptation to data formats → flexibility

Proteus: Customized Data Access

Engine adapts to data
Plug-in per data source
Build auxiliary structures



Generate ‡
Operators
LLVM

- ✓ Reduce branches
- ✓ Minimize function calls
- ✓ Pipelining

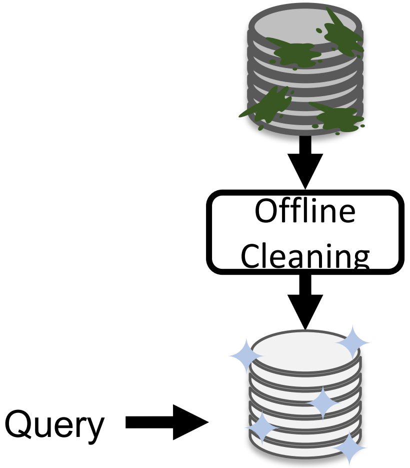
‡ Krikellas [ICDE2010], Neumann [VLDB2011]

Each source as native format, generate special engine/query

Full Cleaning: A Dirty Solution

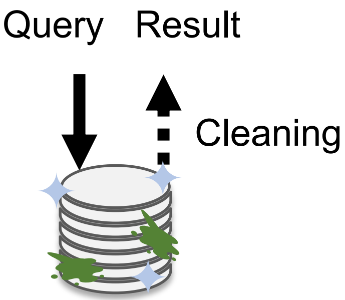
Conservative: query cleaned database copy

- Time-consuming transformation: Dirty=>Clean DB
- Analysis dependent
- Wasteful effort on unnecessary data



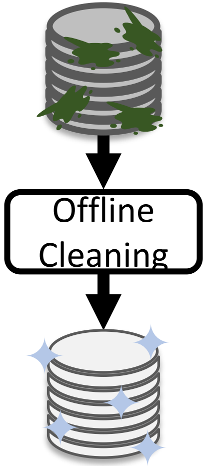
Real-time: Sanitize only interesting data

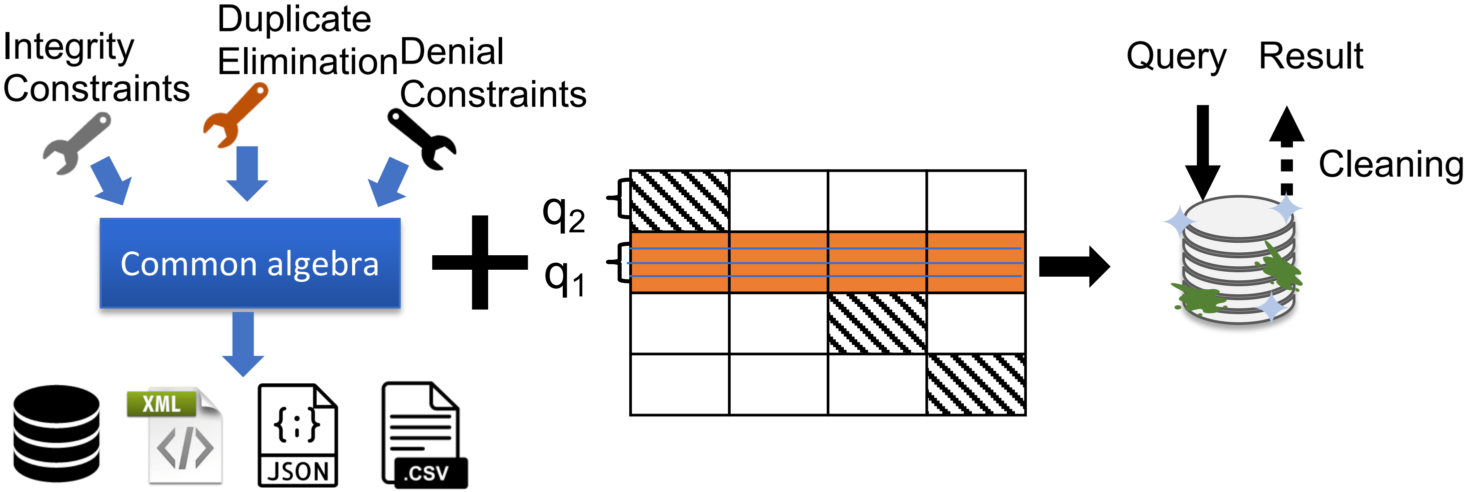
- Virtualized clean DB
- Query relaxation to guarantee correctness
- Clean touched data as-you-go



Focus on cleaning useful data

Query-Driven Data Sanitization



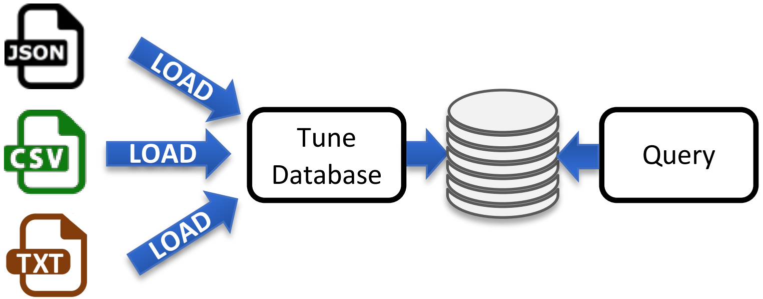


Clean only useful data with probabilistic fixes

Indexing Decisions: It's All About Accesses

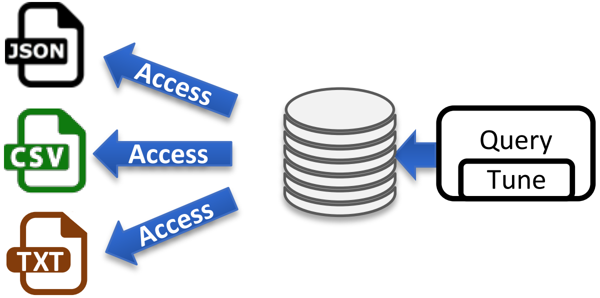
Conservative: parse before tune

- Based on workload-expectations
- Data duplication
- Coarse-grained access path selection



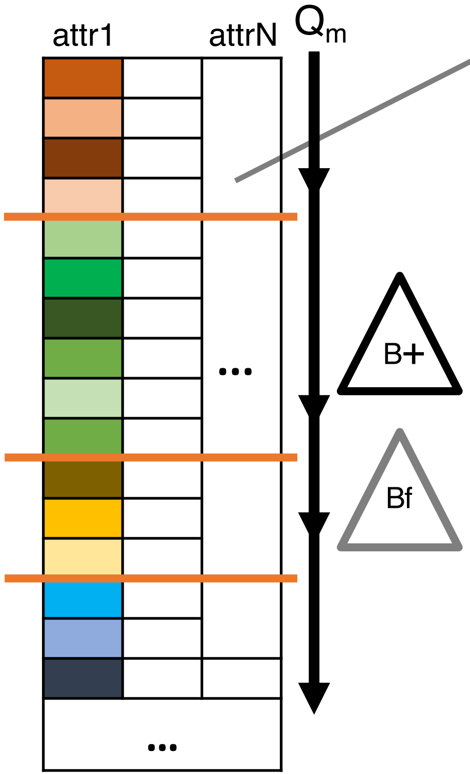
Real-time: on-the-fly indexing of raw data

- Virtualized (logical) partitioning & online index tuning
- Reuse raw data
- Fine-grained tuning, as-you-go

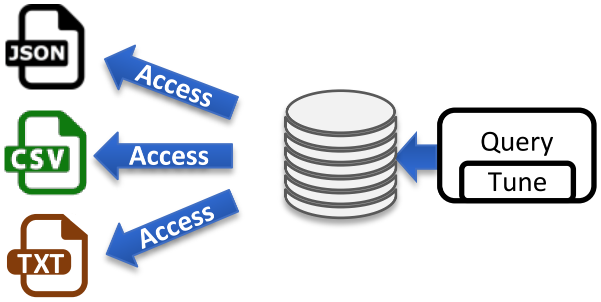


Online partial indexing, specialized to access pattern

Evolving Indexes



costs vs. gains
Should I build or not?



Data skipping

Fine-grained access path selection

Choose what to build & when

- Value-Existence (i.e., Bloom filters)
- Value-Position (i.e., B+ Trees)

Build / drop based on budget

On-the-fly virtual indexes to invest on important data

Runtime specialization embraces heterogeneity

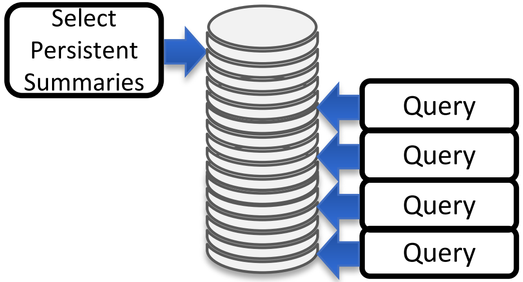
Heterogeneous

Data	Unstandardized	Inconsistencies	Skewness
Workload	Uncertainty	Complexity	Volume
Hardware	Underutilization	Boundaries	Interference

AQP: From Expectations to Adaptation

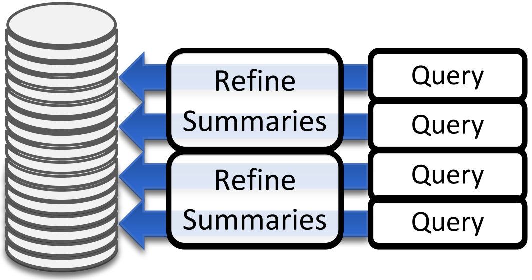
Conservative: offline vs online AQP

- Tradeoff between performance and flexibility
- Preprocessing cost or reduced gains
- Static sampling user-driven tactic



Real-time: Workload-driven data summarization

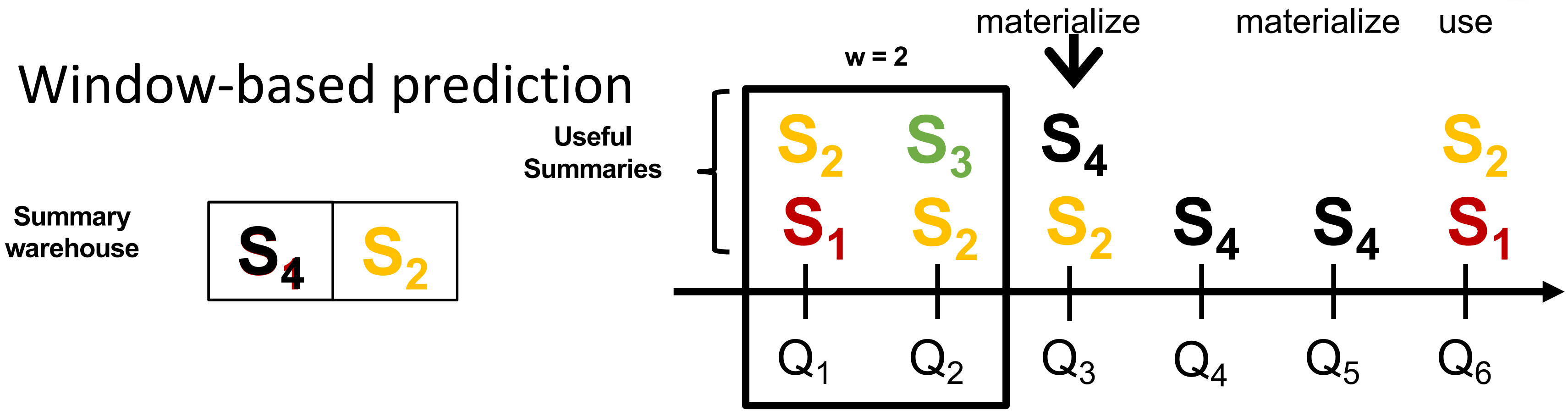
- Maintain sample/sketch set specialized to workload history
- Materialize summaries based on gain estimates
- Pay-as-you-go summary creation and storage overheads



Adapt summaries based on workload patterns

Tuning Materialized Summaries

Window-based prediction



Estimate prospective gains vs. storage cost

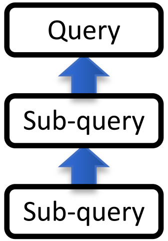
Adapt window size based on quality of predictions

Materialize summaries maximizing gain on past workload

Bypassing Intra-Query Dependencies

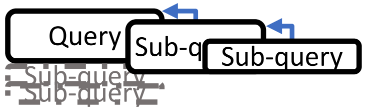
Conservative: Query unnesting

- Staged query execution: waiting for unnecessary details
- Sharing prohibited across barriers
- Blocking & limited parallelism dues to dependencies



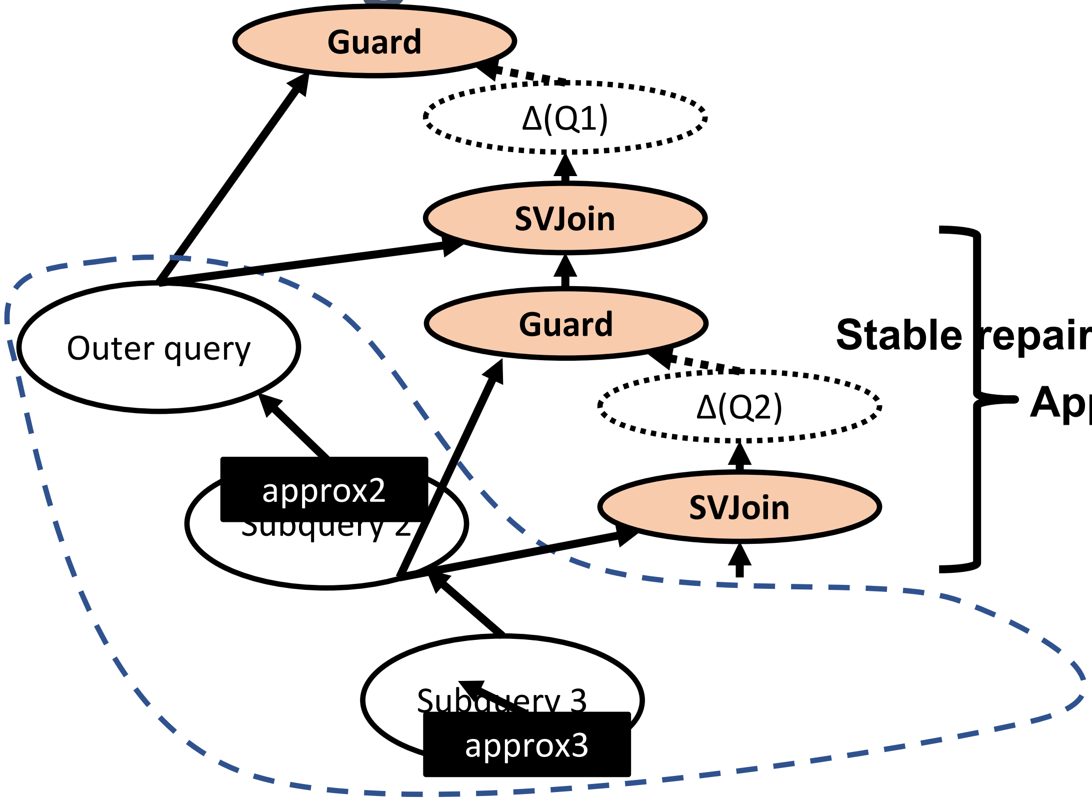
Real-time: Speculate intra-query dependencies

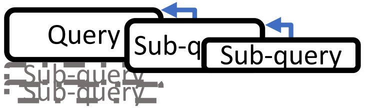
- Virtualize query execution by speculating and repairing
- Exposed sharing and parallelism across barrier
- Verify & repair query results as-you-go



Expect, predict & proceed: specialize to uncertainty

Chocking Points Reduce Parallelism





Stable repair propagation

Apply recursively

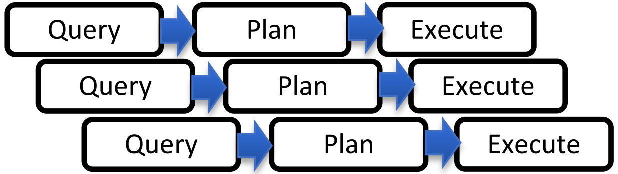
Task parallelism and sharing opportunities

Subquery speculation to create task-independence

Planning in Multi-Query Execution

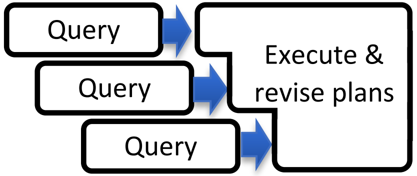
Conservative: Opportunistic sharing

- Missed sharing opportunities, to avoid optimization time
- (Multi-) Query & statistic dependent
- Sensitive to query order, plans, correlations



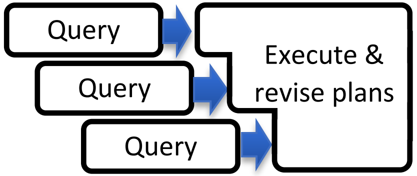
Real-time: Adaptive multi-query optimization

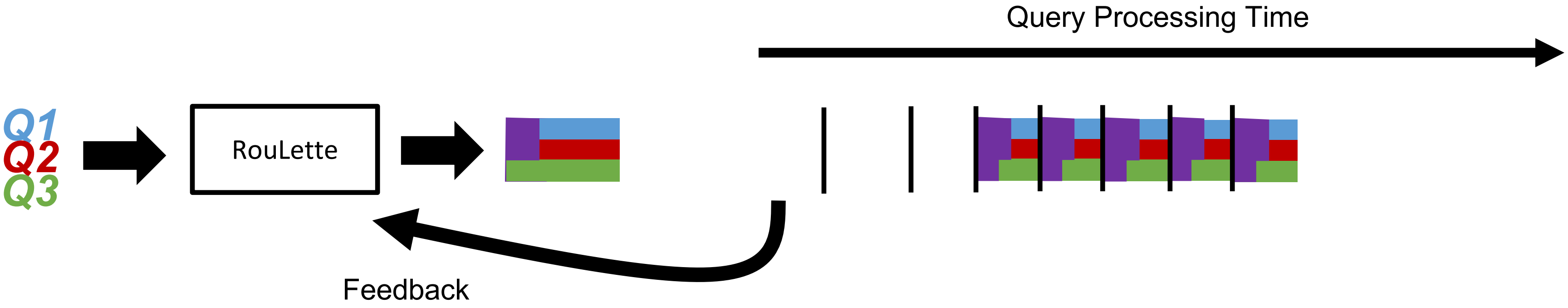
- Specialize to running queries- & data-at-hand
- Inspect actual statistics, observed by minibatches
- Reconfigure execution for pay-as-you-go plans



Learn & explore using running Qs → fine granularity reopt

Concurrent Queries: From Complexity to Shareability





Plan efficiency → **Reinforcement learning**

Execution Efficiency → **Low-overhead adaptation**

Query-interoperability-based, on-the-fly batch-reoptimization

Runtime specialization embraces heterogeneity

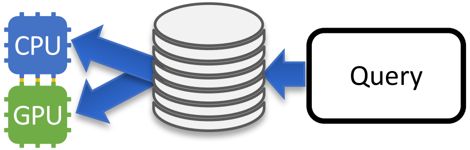
Heterogeneous

Data	Unstandardized	Inconsistencies	Skewness
Workload	Uncertainty	Complexity	Volume
Hardware	Underutilization	Boundaries	Interference

Hardware Acceleration: A Balancing Game

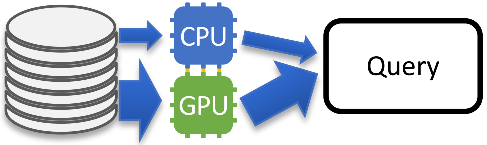
Conservative: Performance – portability tradeoff across devices

- Device specific operator implementations
- Limited or expectation-based load-balancing
- Inefficient hardware use



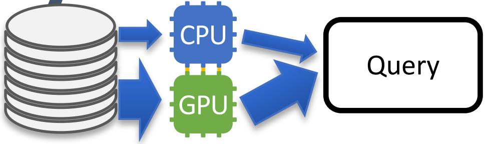
Real-time: Synergistic CPU-GPU execution

- Virtualize hardware & generate hardware-specific code
- Throughput-based data-flow load balancing
- Exploit Accelerator-Level Parallelism



Use devices based on their relative performance

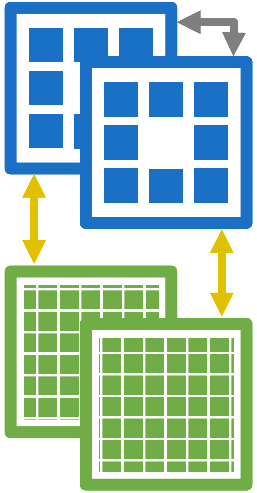
Hardware Heterogeneity: Let the Data Flow

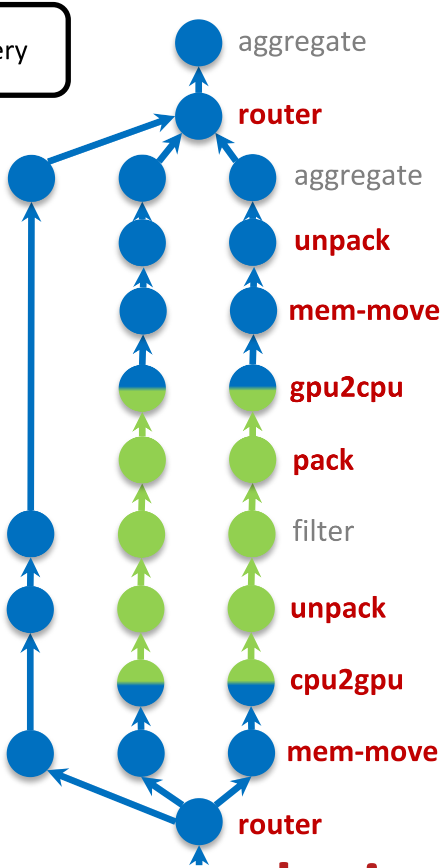


Decouple data- from control-flow

Encapsulate trait conversions into operators

Inspect flows to load-balance



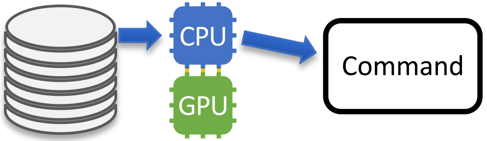


Flow inspection to load balance across heterogeneous devices

Hardware Boundaries → Isolation Mechanisms

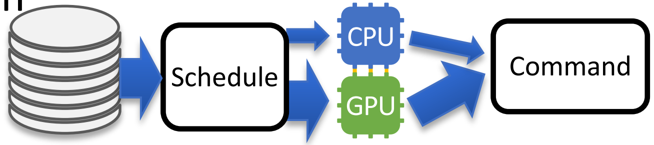
Conservative: Device-collocated OLAP & OLTP

- Wasted parallelism and throughput
- Static hardware preferences
- Destructive interference across workloads



Real-time: Dynamic task assignment minimizes interference

- ALP & hardware boundaries as an isolation mechanism
- Fresh-data-rate- & isolation-driven task assignment
- Pay-as-you-access-fresh-data interference



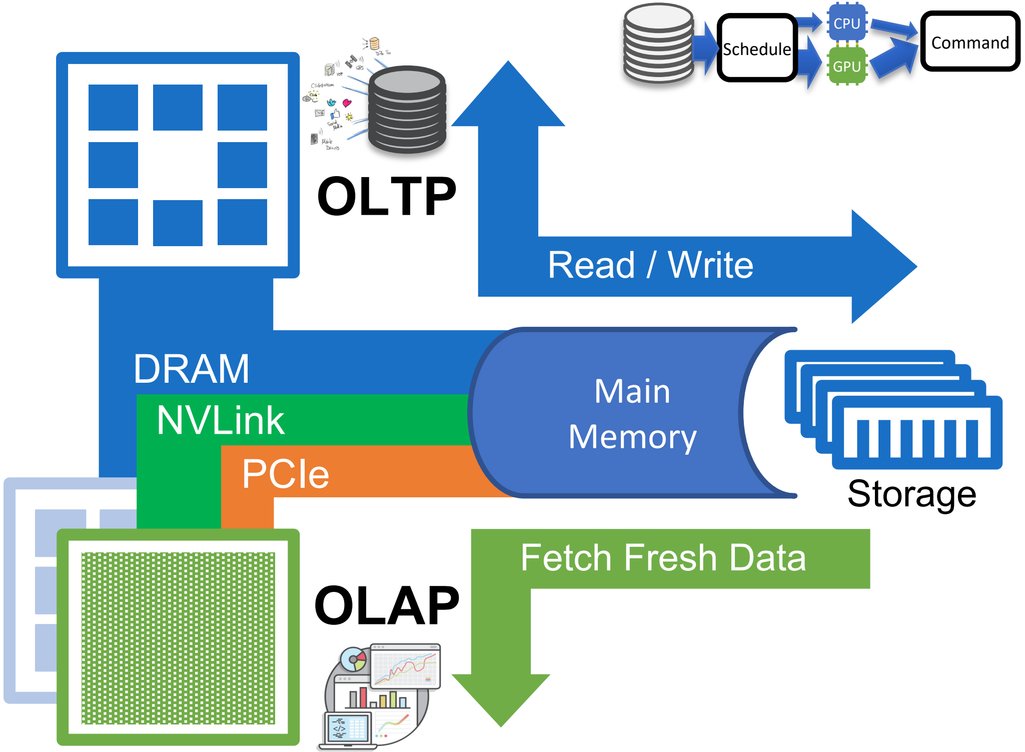
Align isolation requirements with hardware boundaries

GPU Accesses Fresh Data from CPU Memory

OLTP generates fresh data on CPU Memory

Data access protected by concurrency control

OLAP needs to access fresh data

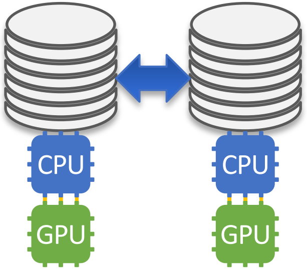


Provide snapshot isolation for OLAP w/o CC overheads

HTAP: Chasing Freshness Locality

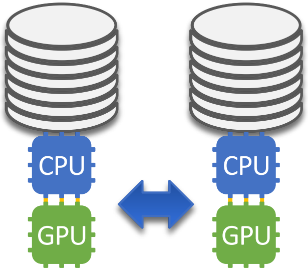
Conservative: Static OLAP-OLTP assignment

- Unnecessary tradeoff between interference and performance
- Pre-determined resource assignment based on workload type
- Wasteful data consolidation and synchronization



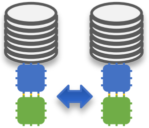
Real-time: Adaptive scheduling of HTAP workloads

- Specialize to requirements and data/freshness-rates
- Workload-based resource assignment
- Pay-as-you-go snapshot updates



Task placement based on resource usage

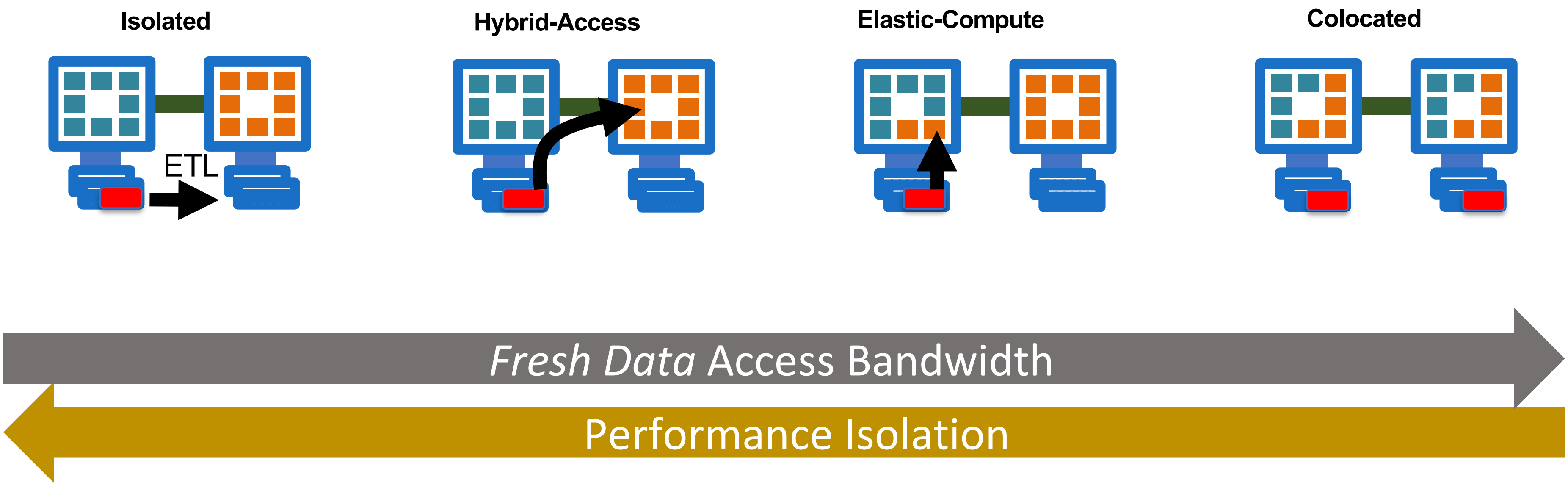
Workload Isolation & Fresh Data Throughput



OLTP

OLAP

Fresh Data



Freshness-based: from destructive to constructive interference

Runtime specialization embraces heterogeneity

Heterogeneous	**Data**	Unstandardized	Inconsistencies	Skewness
	Workload	Uncertainty	Complexity	Volume
	Hardware	Underutilization	Boundaries	Interference
		Inspect & Reduce	**Relax & Repair**	**Track & Adjust**

Runtime specialization embraces heterogeneity

Heterogeneous	**Data**	Unstandardized	Inconsistencies	Skewness
	Queries **~~Workload~~**	Uncertainty	Complexity	Volume
	Hardware	Underutilization	Boundaries	Interference
	Workload			

Application landscape changes data processing

Complexity of Modern Workloads

Diverse modern data problems

- IOT, OCR, ML, NLP, Medical, Mathematics etc…

DBMS catch-up for popular functionality

- Human effort and big delays
- Oblivious to out-of-DBMS workflows

Vast resource of libraries

- Authored by domain experts, used by everybody
- Loose library-to-data-sources integration and optimization

Commercial AI/ML

Augmented analytics

Conversational analytics and NLP

Combination of IoT and analytics

Need for systems that can “learn” new functionality

five old friends revisited

~~Data~~ variety → Operational environment variety

- Unpredictable application requirements

~~Data~~ veracity → Inter-component veracity

- Heterogeneous data & variable importance

~~Data~~ volume → Structural volume

- Multi-layered system architectures

~~Data~~ value → Resource value

- Broader, multi-featured analytics

~~Data~~ velocity → Technological velocity

- Hardware heterogeneity & volatility

Intelligent systems to catch-up with an evolving landscape

Intelligent Real-time Systems

Incorporate change into native design.

Anticipate change and react, learning from errors.

A solution is only as efficient as its least adaptive component.