



TECHNISCHE
UNIVERSITÄT
DARMSTADT

TOWARDS INTERACTIVE DATA ANALYSIS

(A System's Guy Perspective)

CARSTEN BINNIG

DATA MANAGEMENT LAB

**Intuitive Interfaces
for End-Users**

**Fast & Complex
Analytical Operations**

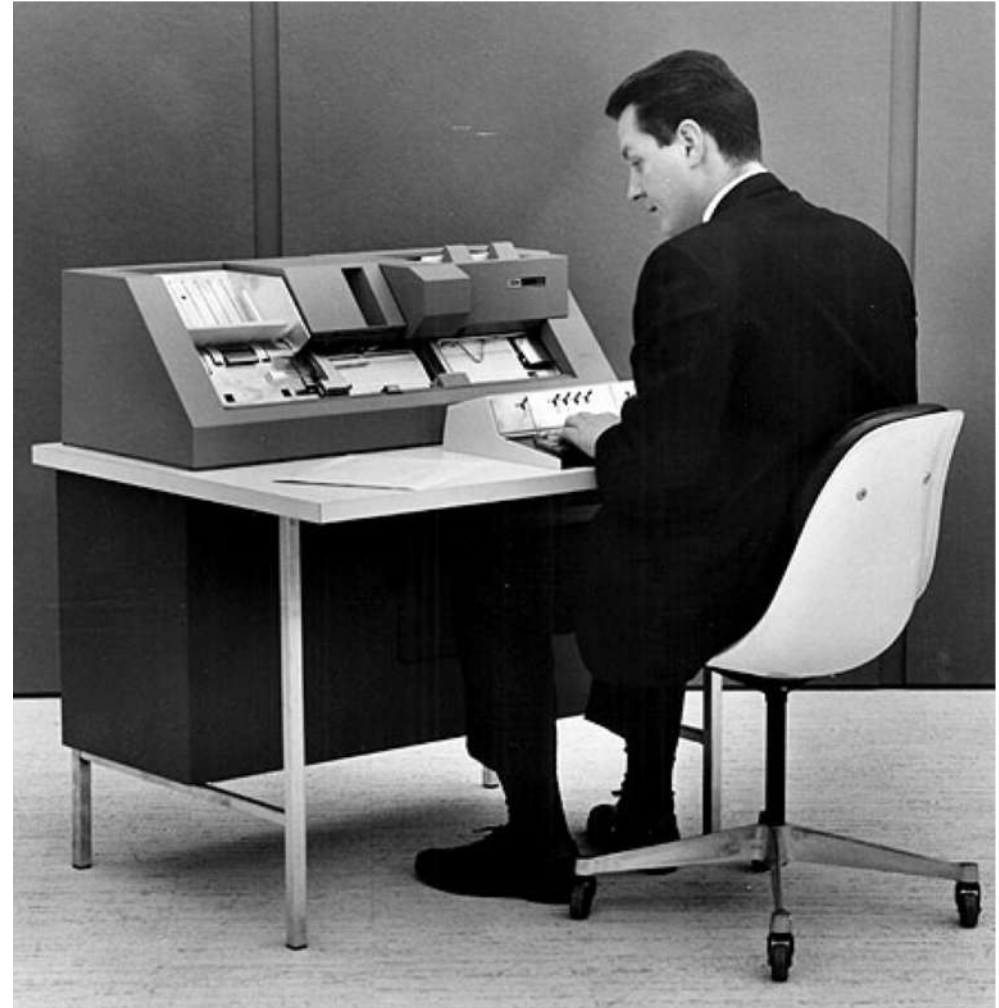
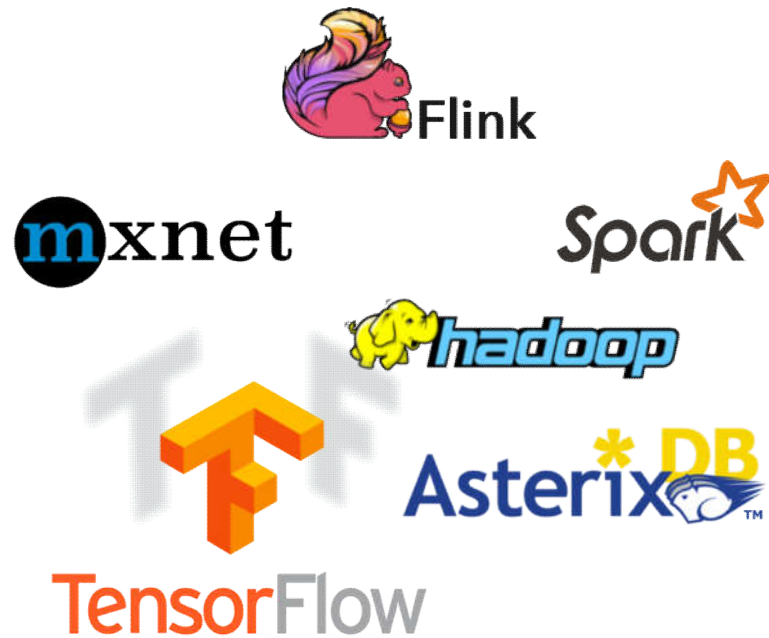
**Large and
Heterogenous Data**

VISION: INTERACTIVE DATA ANALYTICS



TODAY'S USER INTERFACES

... AND THE BIG DATA SYSTEMS?



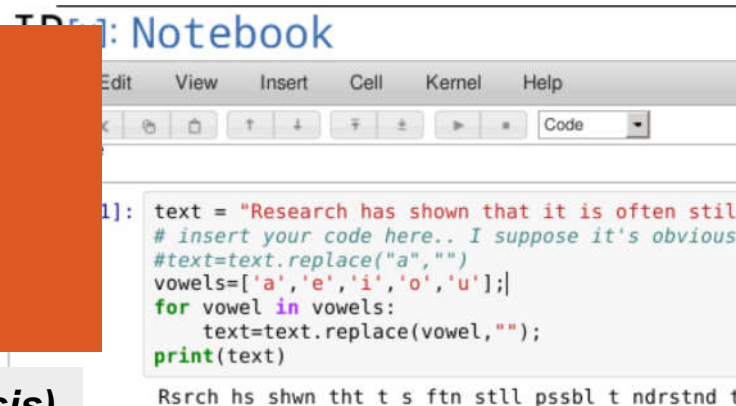
A TYPICAL DATA ANALYSIS PIPELINE

How do analytics interfaces need to change?

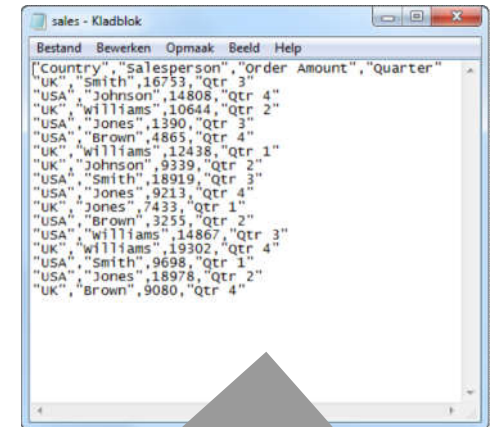
- **Vizdom (Visual Analysis)**
- **DBPal (NL Interface)**

How do we reduce data cleaning and transformation costs?

- **UnkownUnkowns (Data Quality)**
- **IncMap (Schema Mapping)**
- **Sherlock (Text Summarization)**



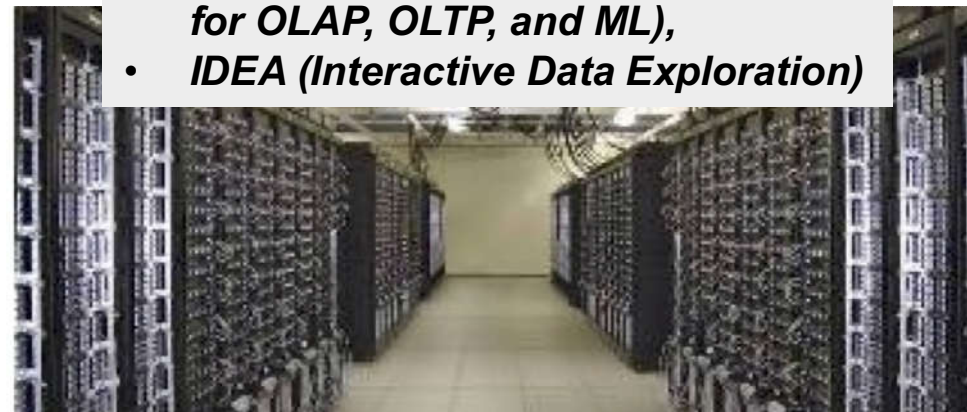
```
1): text = "Research has shown that it is often still  
# insert your code here.. I suppose it's obvious  
#text=text.replace("a", "")  
vowels=['a', 'e', 'i', 'o', 'u'];  
for vowel in vowels:  
    text=text.replace(vowel, "");  
print(text)  
  
Rsrch hs shwn tht t s ftn still psbl t ndrstdn x
```



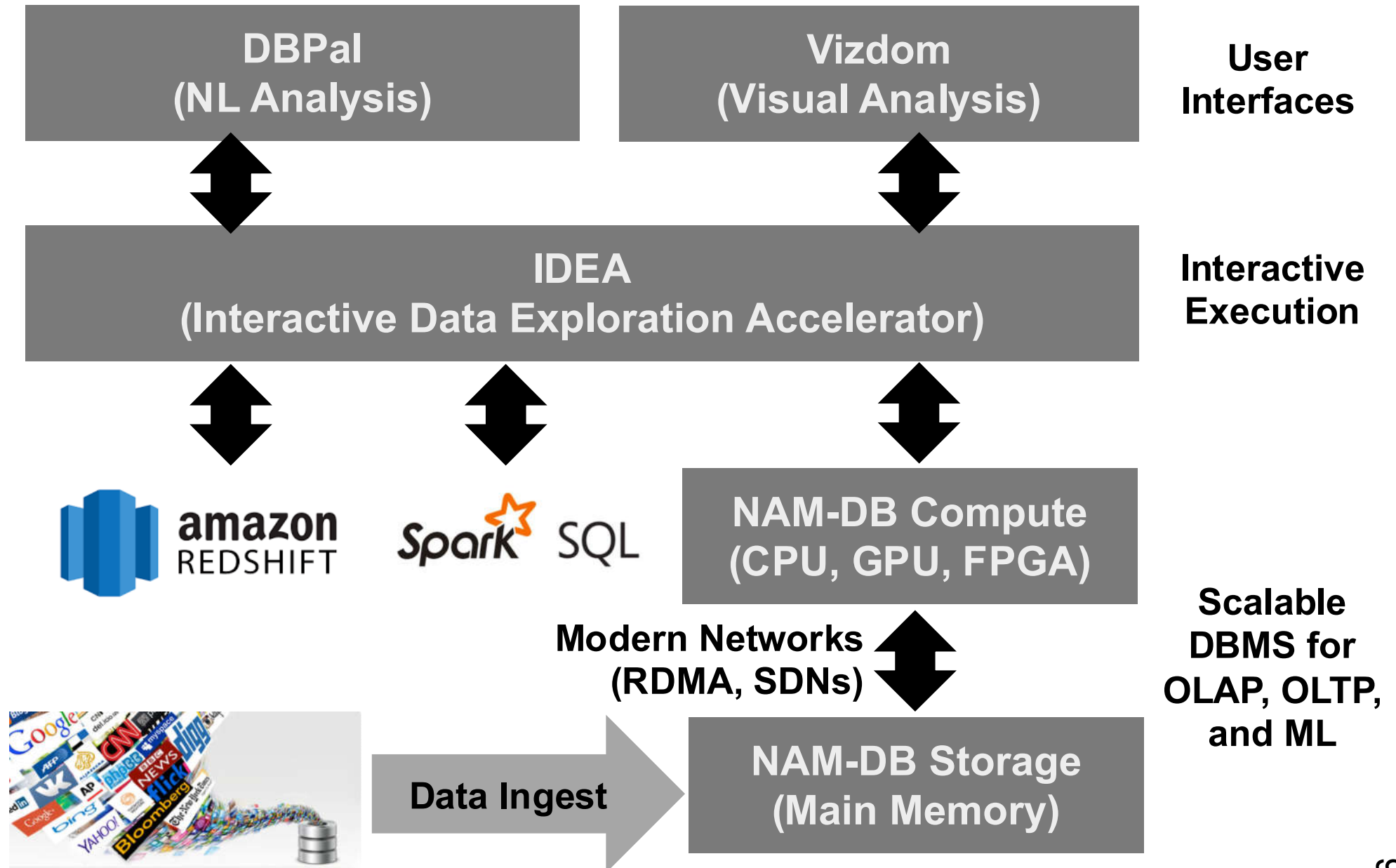
```
sales - Kladblok  
Bestand Bewerken Opmaak Beeld Help  
["Country", "Salesperson", "Order Amount", "Quarter"  
"UK", "Smith", 16753, "Qtr 3"  
"USA", "Johnson", 14808, "Qtr 4"  
"UK", "Williams", 10644, "Qtr 2"  
"USA", "Jones", 1390, "Qtr 3"  
"USA", "Brown", 4865, "Qtr 4"  
"UK", "Williams", 12438, "Qtr 1"  
"UK", "Johnson", 9339, "Qtr 2"  
"USA", "Smith", 18919, "Qtr 3"  
"USA", "Jones", 9213, "Qtr 4"  
"UK", "Jones", 7433, "Qtr 1"  
"USA", "Brown", 3255, "Qtr 2"  
"USA", "Williams", 14867, "Qtr 3"  
"UK", "Williams", 19302, "Qtr 4"  
"USA", "Smith", 9698, "Qtr 1"  
"USA", "Jones", 18978, "Qtr 2"  
"UK", "Brown", 9080, "Qtr 4"]
```

How do we enable high-speed complex analytics on large data?

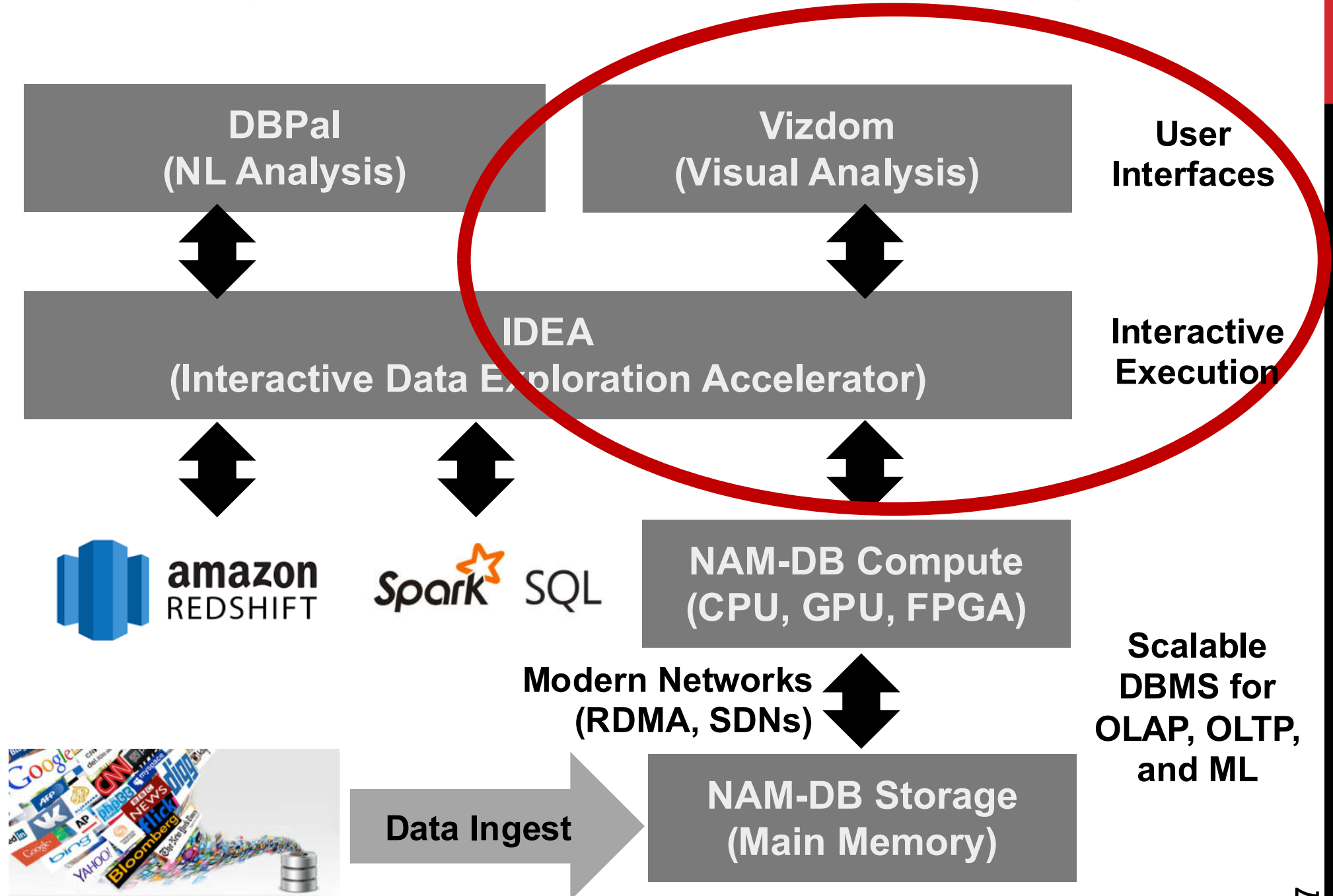
- **NAM-DB (Scalable Databases for OLAP, OLTP, and ML),**
- **IDEA (Interactive Data Exploration)**



DARMSTADT DATA ANALYSIS STACK



DARMSTADT DATA ANALYSIS STACK





Interactive Analytics through Pen and Touch

Andrew Crotty, Alex Galakatos, Emanuel Zraggen, Carsten Binnig, Tim Kraska



CHALLENGE: INTERACTIVE RESPONSES

Response time higher than 500 ms already limit the exploration space and productivity of users

The Effects of Interactive Latency on Exploratory Visual Analysis

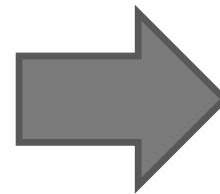
Zhicheng Liu and Jeffrey Heer

In this research, we have found that interactive latency can play an important role in shaping user behavior and impacts the outcomes of exploratory visual analysis. **Delays of 500ms incurred significant costs, decreasing user activity and data set coverage while reducing rates of observation, generalization and hypothesis.** Moreover, initial exposure to higher latency interactions resulted in reduced rates of observation and generalization during subsequent analysis sessions in which full system performance was restored.

BASIC IDEA: AQP FROM THE 90'S

Sales

Product	Amount
CPU	1
CPU	1
CPU	2
CPU	3
CPU	4
Disk	1
Disk	2
Monitor	1



**Sampling
(Online OR Offline)**

Sales-Sample

Product	Amount
CPU	1
CPU	2
CPU	3
Disk	2

```
SELECT SUM(Amount) FROM Sales WHERE Product = 'CPU'
```

Exact Answer:

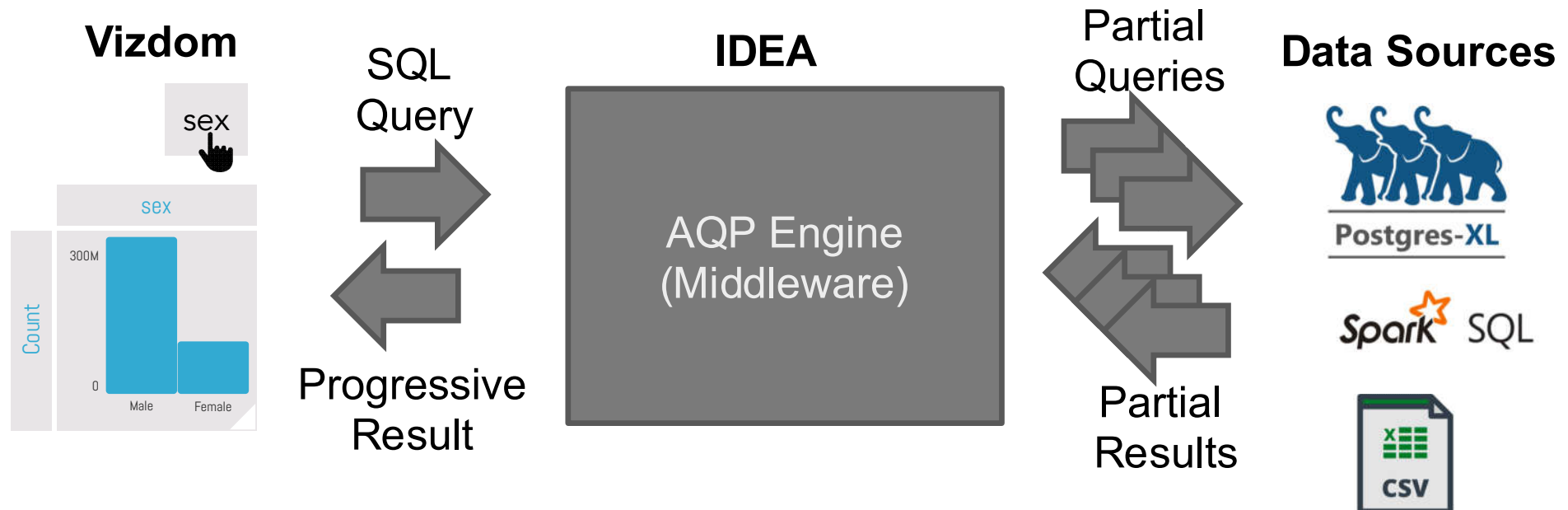
$$1+1+2+3+4 = 11$$

Approx. Answer:

$$(1+2+3)*2 = 12$$

PROJECT: IDEA

Crotty et al: The case for interactive data exploration accelerators (IDEA). HILDA@SIGMOD'16



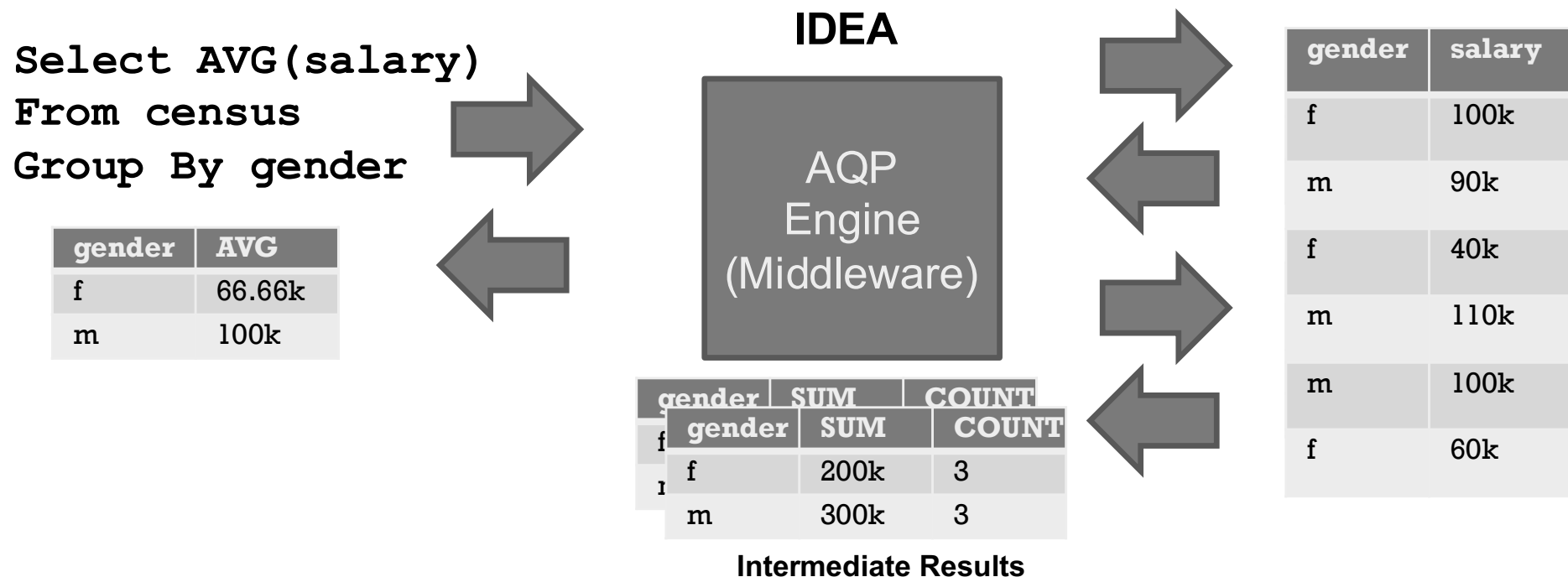
IDEA is a middleware for AQP on top of (Big) Data engines

- Can connect to a variety of engines or other data sources (CSV, ...)
- Provides interactive (progressive) query answering on top of those engines

IDEA: AQP IN THE MIDDLEWARE

Basic Idea:

- **Offline:** data in sources is prepared for progressive AQP (i.e., tables are split into smaller chunks of fixed size)
- **Online:** Incoming SQL queries are split into multiple “smaller” SQL queries and results are merged in middleware



Additional optimizations: Caching and reuse of approximate results to answer subsequent queries

IDEA: RESULT CACHING

Executed Interactions

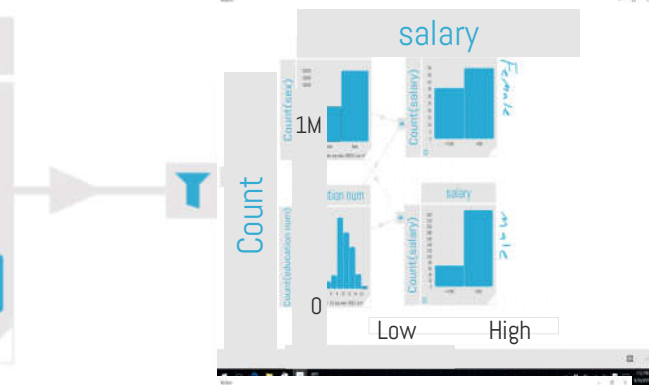
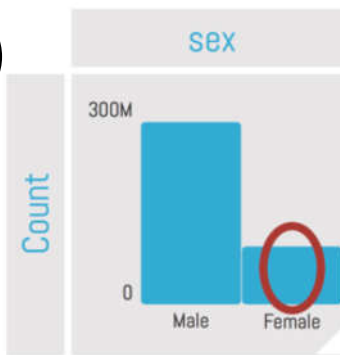
1



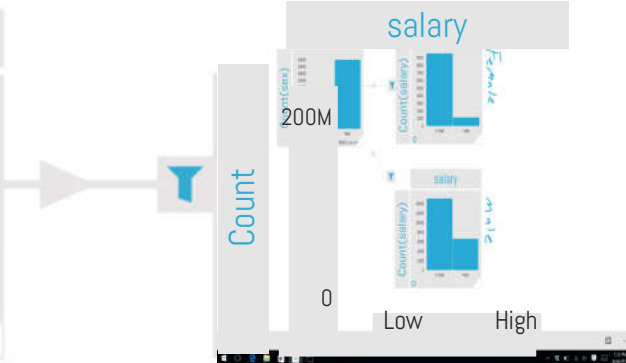
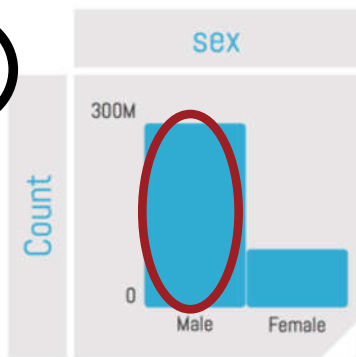
2



3



4

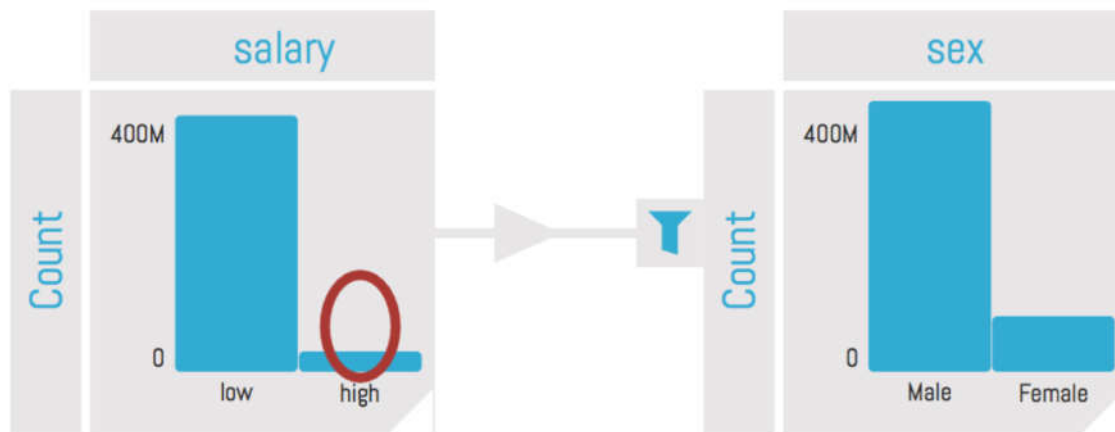


Result Cache (Random Variables)

P_{male}	$\{0.70, \varepsilon_1\}$
P_{female}	$\{0.30, \varepsilon_2\}$
P_{high}	$\{0.20, \varepsilon_3\}$
P_{low}	$\{0.80, \varepsilon_4\}$
$P_{\text{low} \text{male}}$	$\{0.75, \varepsilon_5\}$
$P_{\text{low} \text{female}}$	$\{0.92, \varepsilon_6\}$
$P_{\text{high} \text{male}}$	$\{0.25, \varepsilon_7\}$
$P_{\text{low} \text{female}}$	$\{0.08, \varepsilon_8\}$

IDEA: RESULT REUSE

Galakatos et al.: Revisiting Reuse for Approximate Query Processing. PVLDB '17



Result Cache (Random Variables)

P_{male}	$\{0.70, \epsilon_1\}$
P_{female}	$\{0.30, \epsilon_2\}$
P_{high}	$\{0.20, \epsilon_3\}$
P_{low}	$\{0.80, \epsilon_4\}$
$P_{\text{low} \text{male}}$	$\{0.75, \epsilon_5\}$
$P_{\text{low} \text{female}}$	$\{0.92, \epsilon_6\}$
$P_{\text{high} \text{male}}$	$\{0.25, \epsilon_7\}$
$P_{\text{high} \text{female}}$	$\{0.08, \epsilon_8\}$

Rewrites for Reuse:

- Bayes Theorem
- Law of Total Probabilities
- Inclusion/exclusion Principle
-

Bayes' Theorem:

$$P(A | B) = \frac{P(B | A) \cdot P(A)}{P(B)}$$

$$\hat{P}_{\text{male}|\text{high}} = \frac{P_{\text{high}|\text{male}} * P_{\text{male}}}{P_{\text{high}}} \approx 0.88$$

IDEA: EXPERIMENTAL EVALUATION

Workload: Exploration Sessions (User Study)

#1	sex
#2	education
#3	education WHERE sex='Female'
#4	education WHERE sex='Male'
#5	sex, education
#6	sex WHERE education='PhD'
#7	salary
#8	salary WHERE education='PhD'
#9	sex, salary
#10	salary WHERE sex='Female'
#11	salary
#12	salary WHERE sex='Female'
#13	salary WHERE sex<>'Female'
#14	salary WHERE sex='Female' AND education='PhD', salary WHERE sex<>'Female' AND education='PhD'
#15	age
#16	salary WHERE 20<=age<40 AND sex='Female' AND education='PhD', salary WHERE 20<=age<40 AND sex<>'Female' AND education='PhD'

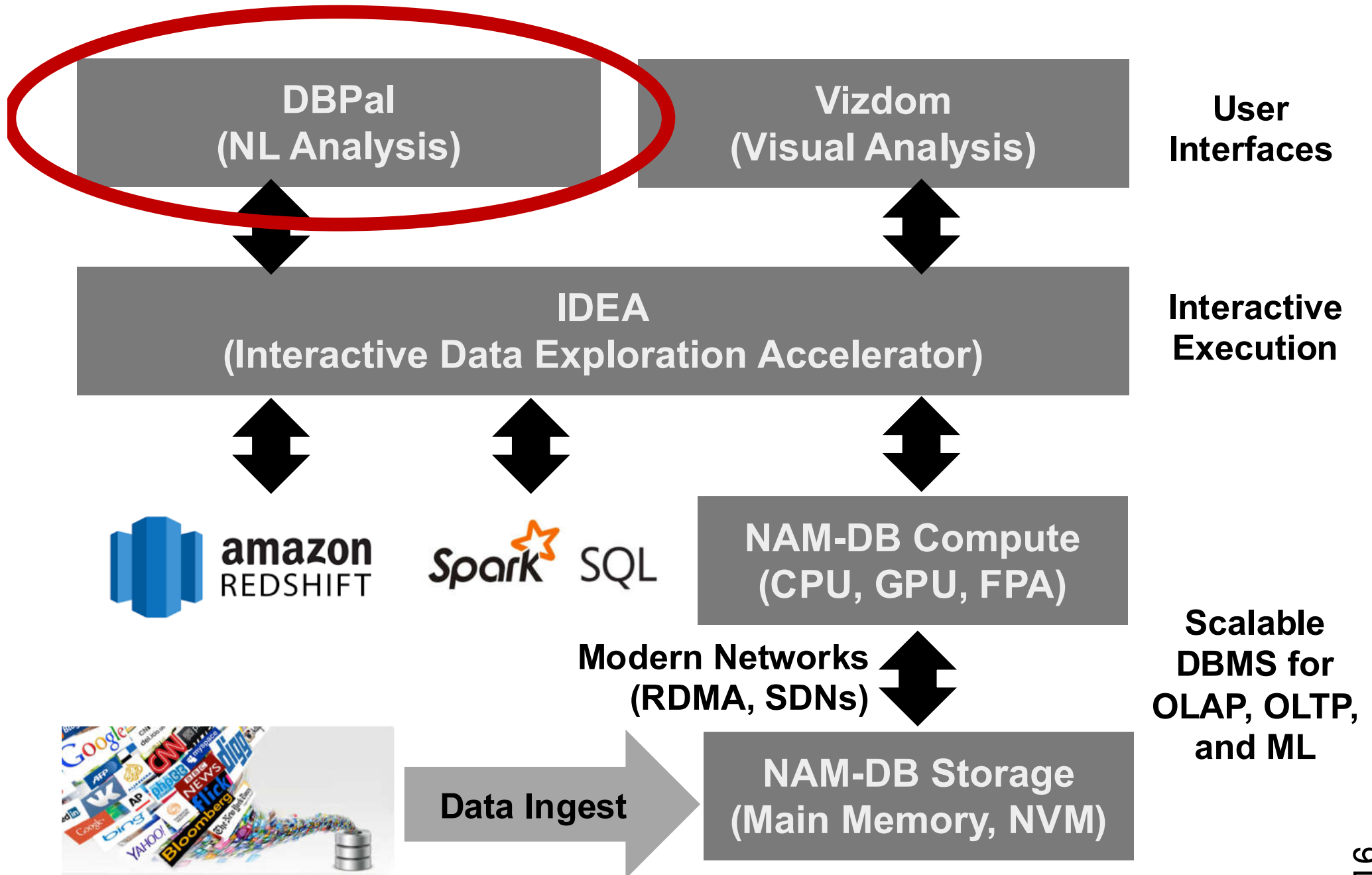
Evaluated Systems:

- **MonetDB**: Analytical Column-Store
- **Online Aggregation** (Hellerstein. 90's)
- **IDEA**: on top of raw CSV files

Data: 500M tuples

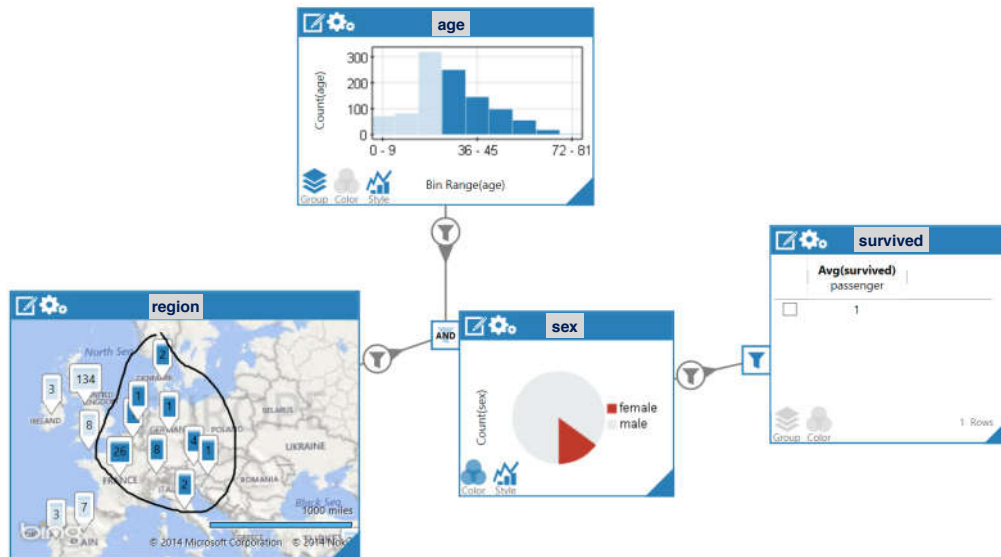
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16
Census MonetDB	0.34	0.39	5.40	8.70	0.48	1.20	1.20	0.91	0.53	4.80	0.42	4.70	1.10	5.60	1.60	7.10
Online Agg	0.05	0.24	0.78	0.59	0.24	0.46	0.04	0.48	0.07	0.11	0.04	0.11	0.08	7.53	0.29	24.3
IDEA	0.09	0.29	0.42	0.00	0.00	0.00	0.09	0.12	0.00	0.17	0.00	0.00	0.00	0.48	0.37	2.87

DARMSTADT DATA ANALYSIS STACK



NL INTERFACE FOR DBMS (NLIDB)

Visual Query:



NL Query:

“How many females older than 30 survived the sinking of the Titanic?”

NL interfaces enable **a natural and concise way** to query data

CHALLENGES FOR NLIDBS

Paraphrased Queries:

- “Show me the patients diagnosed with fever?”
- “What are the patients with a diagnosis fever?”

Incomplete Queries:

- “What are the patients with fever?”
- “Fever patients?”

Ambiguous Queries:

- What are neighbors of New York? (city or state?)

DBPAL: DEEP NL2SQL TRANSLATION

Language Translation Model



How to get training data for each database?



TODAY'S DEEP NLP RECIPE

RECIPE FOR DEEP LEARNING

1. Pick task & domain

2. Manually create training data
(e.g., using crowd)

3. Train translation model

(Repeat for every new task & domain)

... APPLIED FOR NL2SQL

RECIPE FOR DEEP LEARNING

1. Pick task & domain

(DATABASE SCHEMA)

2. Manually create training data
(e.g., using crowd)

(NL-SQL PAIRS)

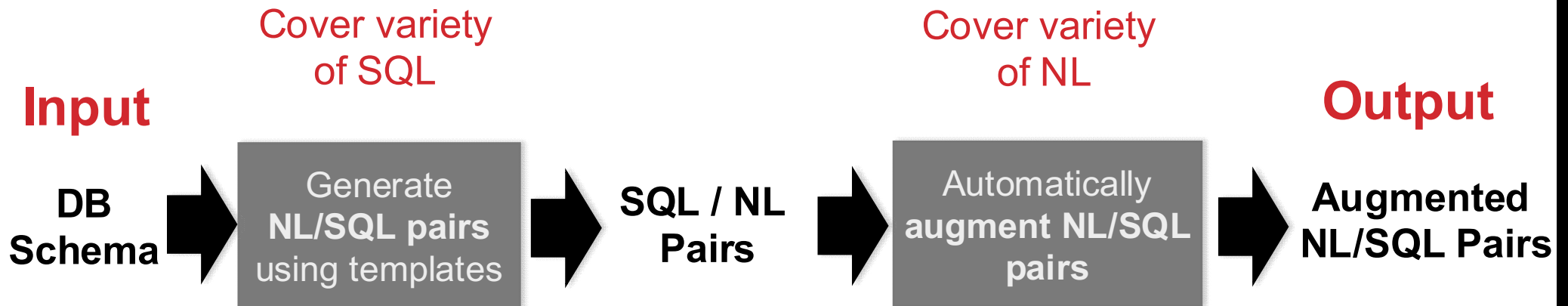
3. Train translation model

(SEQ2SEQ)

(Repeat for every new task & domain)

DBPAL: GENERATING TRAINING DATA

Main Idea: Weak Supervision to Generate Training Data

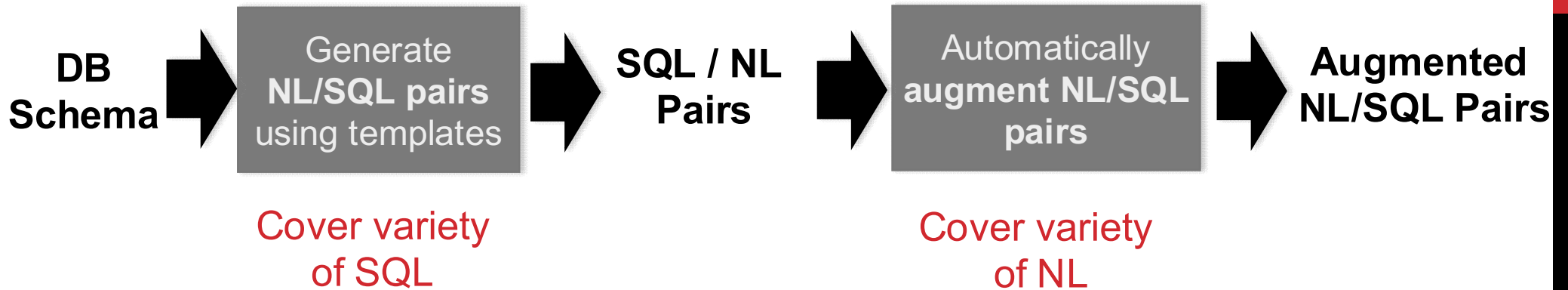


Prasetya Utama, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Çetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, Arif Usta: An End-to-end Neural Natural Language Interface for Databases. CoRR abs/1804.00401 (2018)

DBPAL: GENERATING TRAINING DATA

Input

Output



Template

NL/ SQL Pair

Augmentation

```

SELECT <att>
FROM <table>
WHERE <filter>

Show me the <att>s of
<table>s with <filter>?
  
```

```

SELECT name
FROM patient
WHERE diagnoses = fever

Show me the names of
patients with diagnoses
fever?
  
```

Paraphrasing

Show me the names of patients diagnosed fever?

Noising

Show the names of patients with diagnosed fever?

name	age	diagnoses
Carsten	39	fever
Emilie	8	flu
Frederik	4	fever

Patient Database

... **Millions of different NL/SQL pairs** ...

DBPAL: EXPERIMENTAL RESULTS

Patient and Geo Benchmark

	Patients	GeoQuery
NaLIR (w/o feedback)	15.60%	7.14%
NaLIR (w feedback)	21.42%	N/A
NSP++	N/A	83.9%
NSP (template only)	10.60%	5.0%
DBPal (w/o augmentation)	74.80%	38.60%
DBPal (full pipeline)	75.93%	55.40%

Patient Benchmark (Breakdown per Linguistic Category)

	Naive	Syntactic	Lexical	Morphological	Semantic	Missing	Mixed
NaLIR (w/o feedback)	19.29%	28.07%	14.03%	17.54%	7.01%	5.77%	17.54%
NaLIR (w feedback)	21.05%	38.59%	14.03%	19.29%	7.01%	5.77%	22.80%
NSP (template only)	19.29%	7.01%	5.20%	17.54%	12.96%	3.50%	8.70%
DBPal (full pipeline)	96.49%	94.7%	75.43%	85.96%	57.89%	36.84%	84.20%

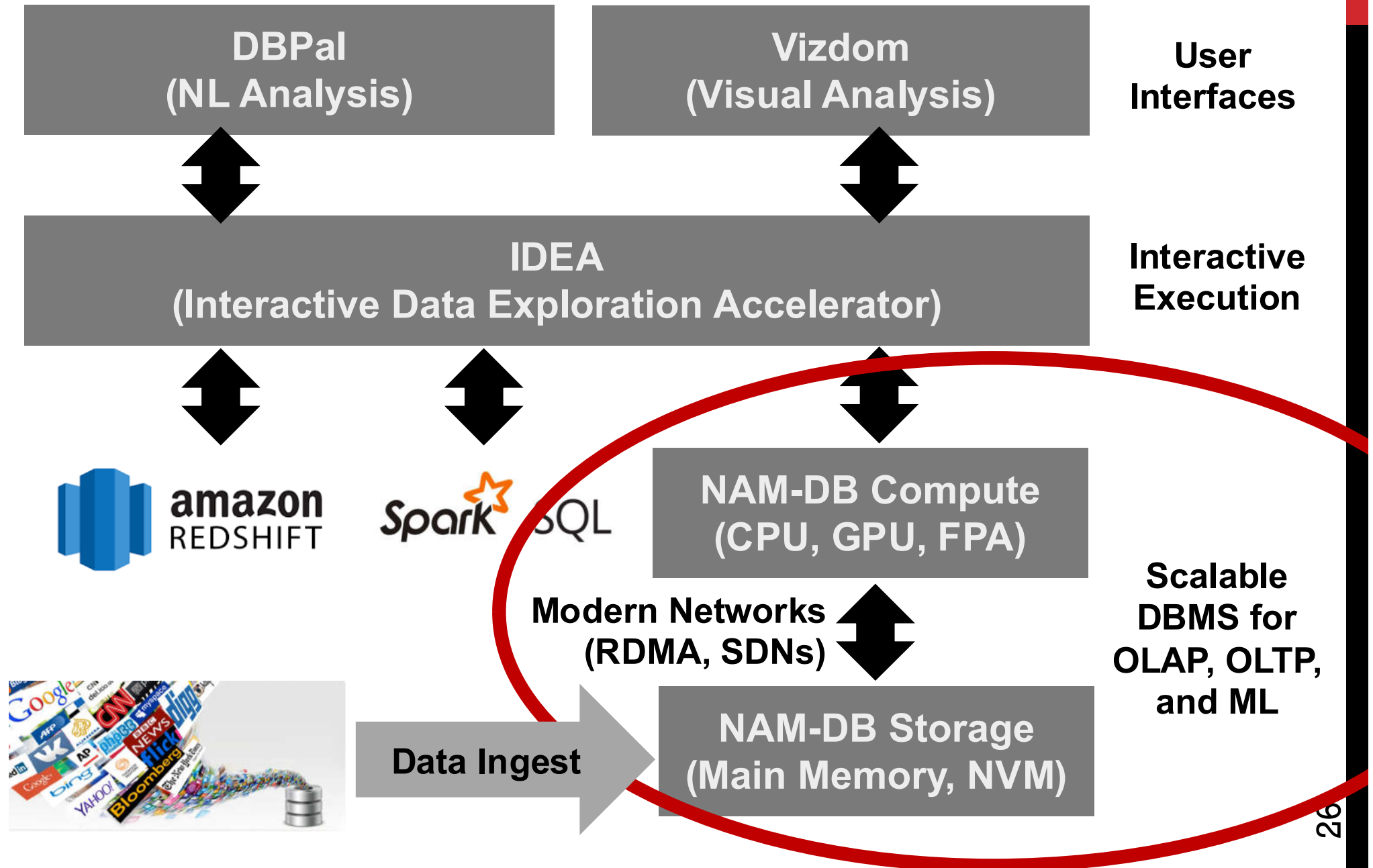
Benchmarks:

- Patient (simple schema, 400 queries in diff. linguistic variations)
- Geo (complex schema, 280 queries)

Baselines

- Traditional: NaLIR (rule-based)
- Deep Model: NSP and NSP++ (manually created training data)

DARMSTADT DATA ANALYSIS STACK



SCALABLE DBMS: PAST WISDOM



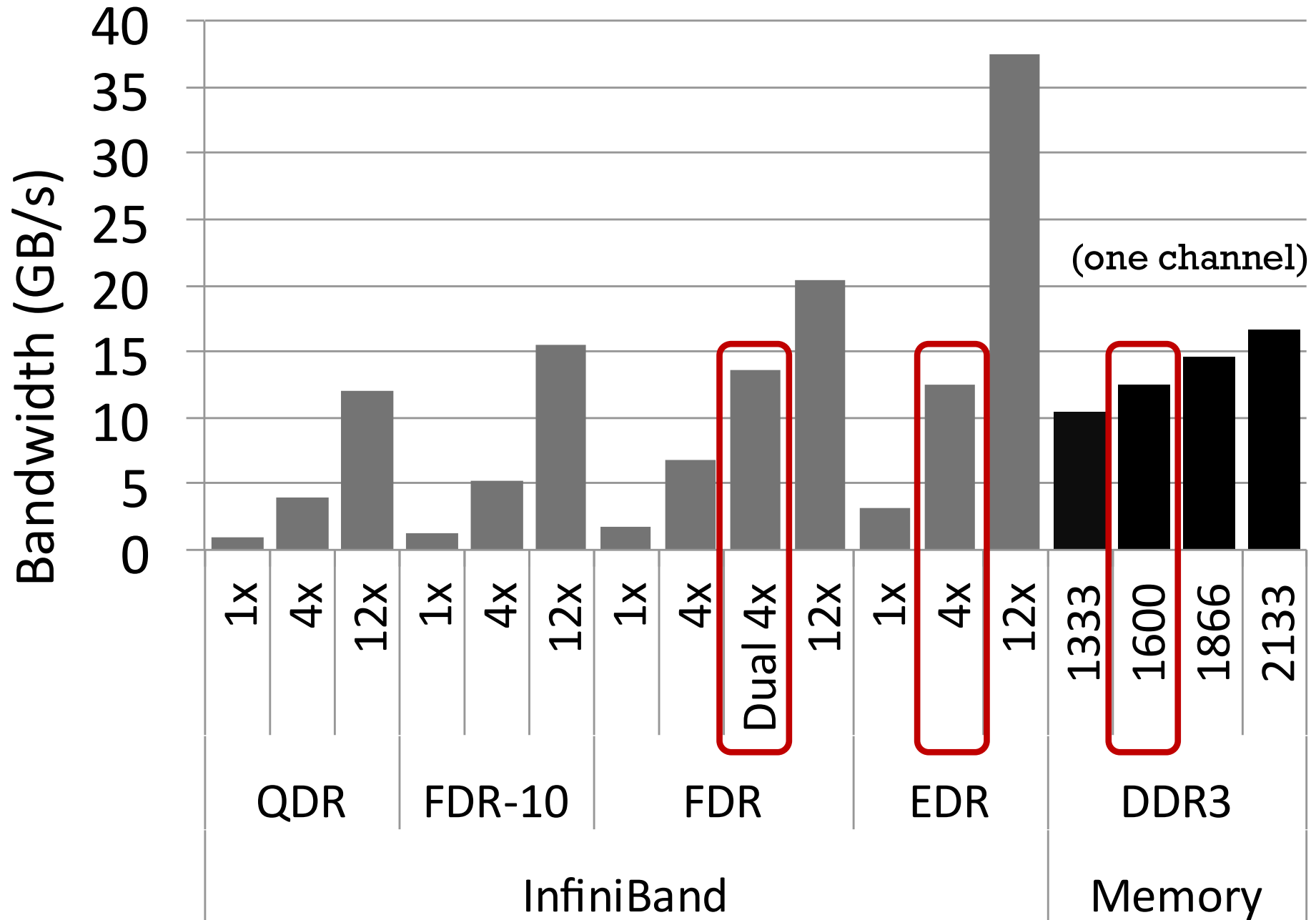
Network Communication is evil: Must be avoided at all cost

	RAM	Network 1Gbps	Net/RAM
Latency, Random 1KB (μ s)	0.1	100	1000
Throughput (GB/s)	51.2 (4 channels)	0.125	~400

Distributed DBMS Mantra: Locality-first!

- **Complex partitioning schemes** to provide data-locality (e.g., Schism, Ref-Partitioning,)
- **Complex computation schemes** to reduce data transfers (e.g. Semi-join reducers, Relaxed consistency protocols, ...)

MODERN HIGH-SPEED NETWORKS

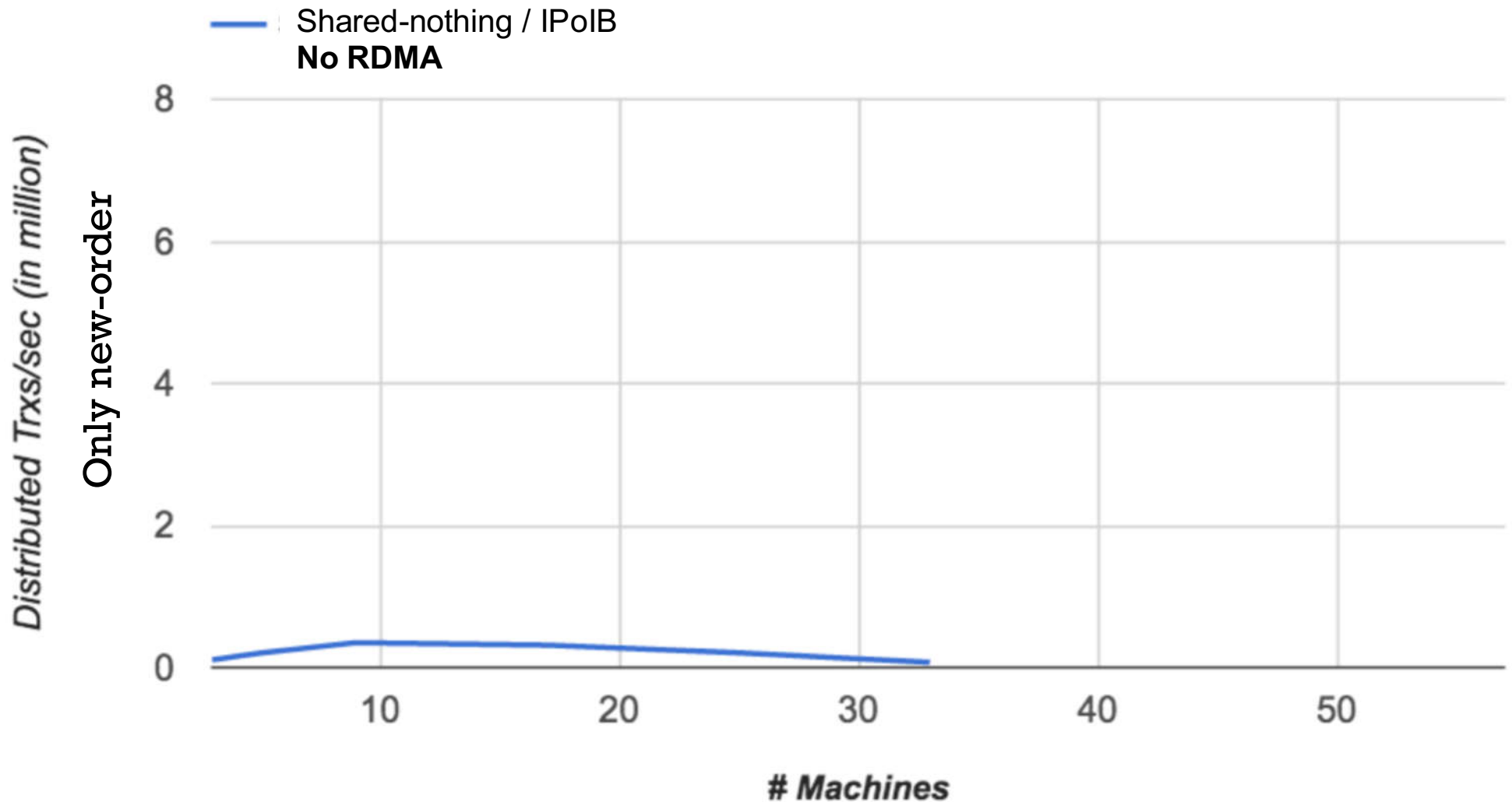


JUST UPGRADE THE NETWORK?



Scale Out Experiment on TPC-C

Binnig et al.: The End of Slow Networks: It's Time for a Redesign. VLDB 2016



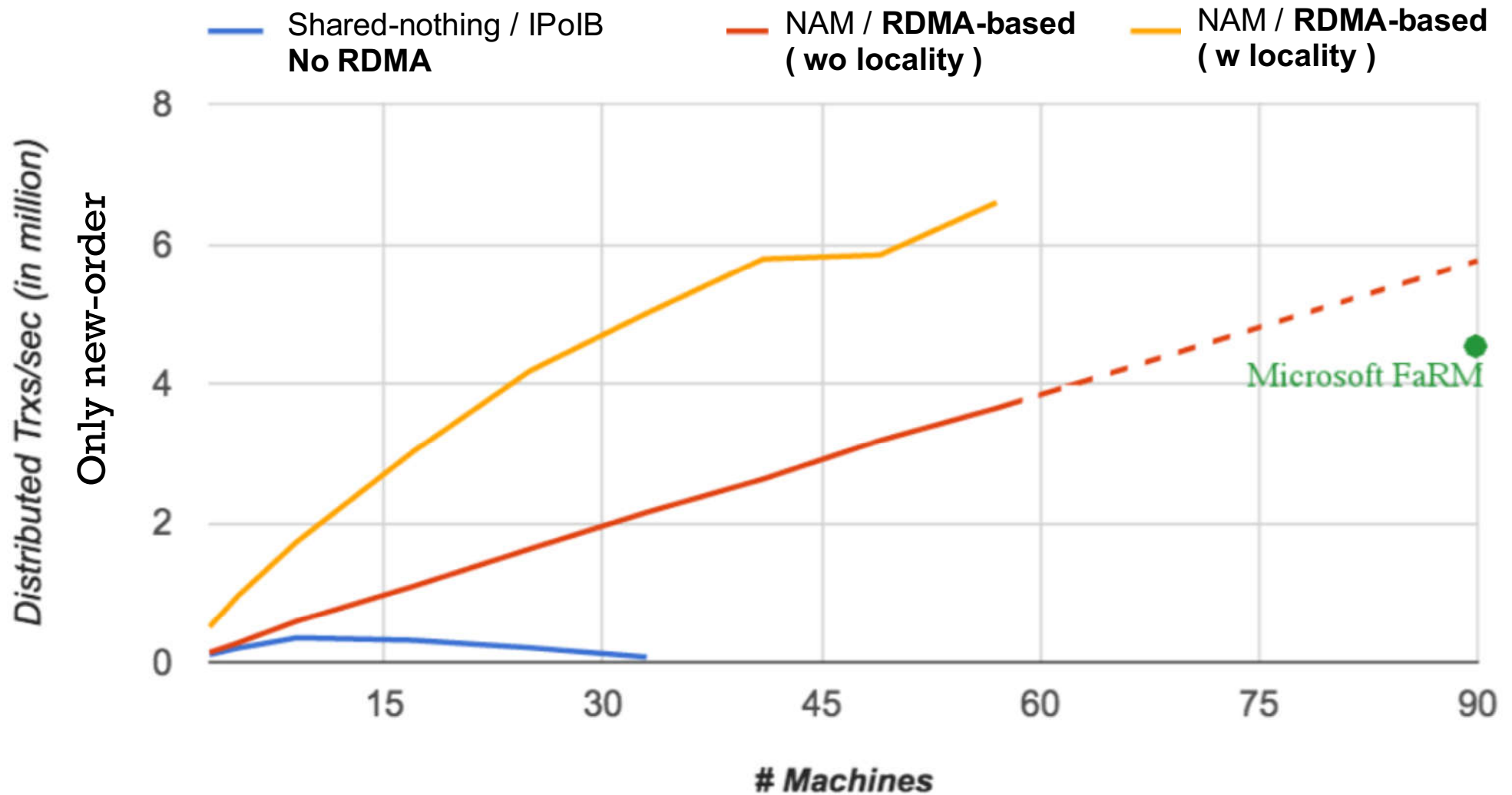
Workload: standard TPC-C, with 50 warehouses per server.

27 machines of type: Two Xeon E7-4820 processors (each with 8 cores), 128 GB RAM

28 machines of type: Two Xeon E5-2660 processors (each with 8 cores), 256 GB RAM

THE CASE FOR A REDESIGN

Binnig et al.: The End of Slow Networks: It's Time for a Redesign. VLDB 2016



Workload: standard TPC-C, with 50 warehouses per server.

27 machines of type: Two Xeon E7-4820 processors (each with 8 cores), 128 GB RAM

28 machines of type: Two Xeon E5-2660 processors (each with 8 cores), 256 GB RAM

FaRM: From the paper "No compromises: distributed transactions with consistency, availability, and performance"

RDMA IN A NUTSHELL

RDMA = Remote Direct Memory Access

Bypasses the OS (i.e., zero-copy data transfer)

RDMA verbs

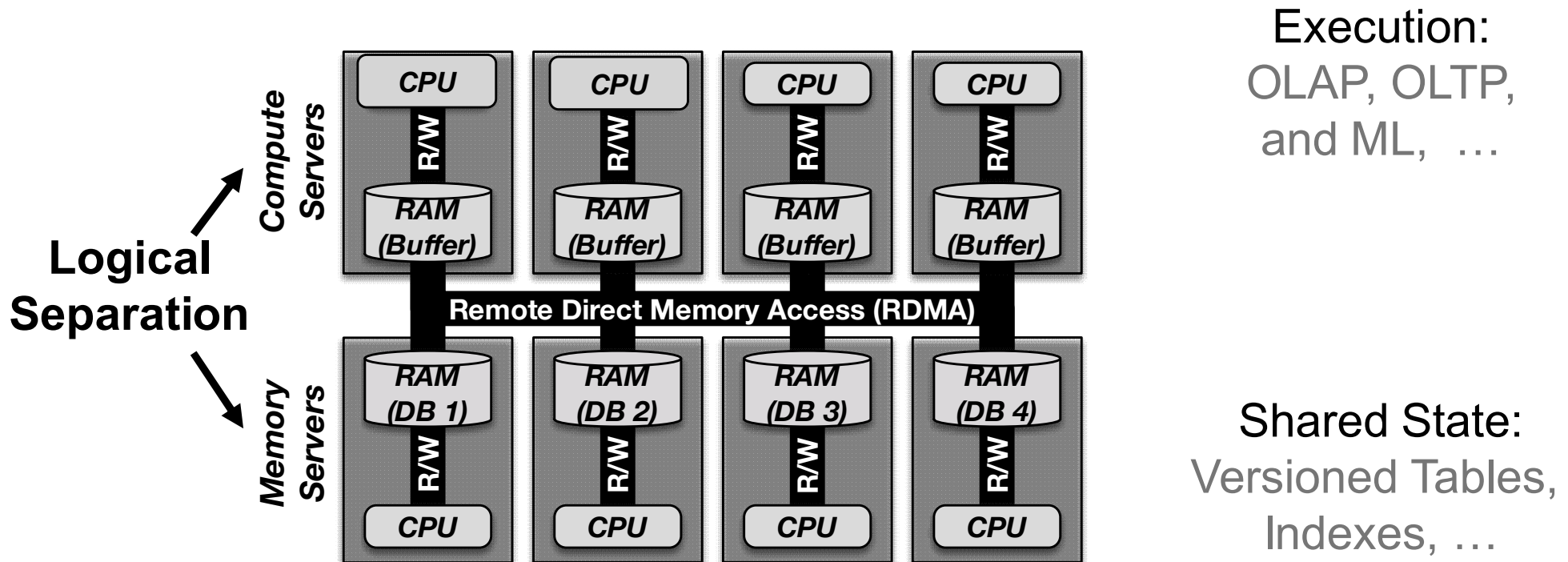
- **One-sided:** READ/WRITE (Remote CPU not involved)
- **Two-sided:** SEND/RECEIVE (Remote CPU involved)

Processing of verbs is offloaded to RDMA NIC (RNIC)

PROJECT: NAM-DB

Binnig et al.: The End of Slow Networks: It's Time for a Redesign. PVLDB'16

Network-Attached Memory (NAM) Architecture:

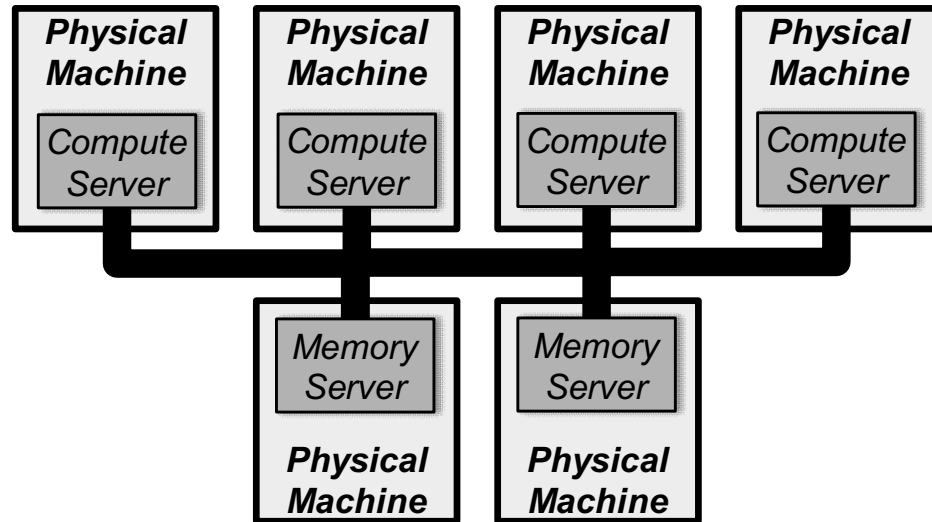


NAM-Architecture: Illusion of one “large” machine

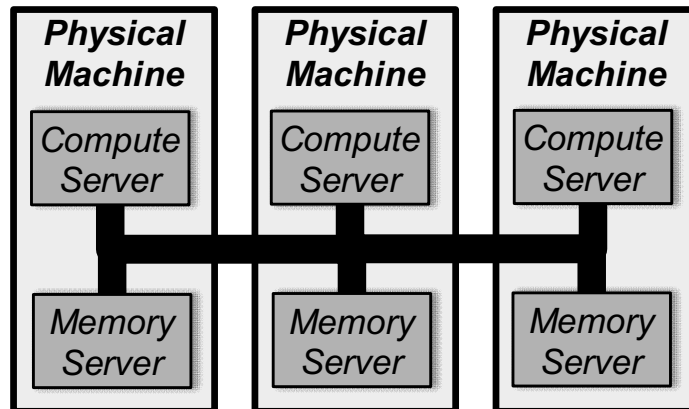
- **Memory Servers:** Expose distributed shared memory pool
- **Compute Servers:** Execute workload -> read/write data via RDMA

Goal: Scalable support for a wide variety of workloads (OLTP, OLAP, ML)

NAM-DB: DIFFERENT INSTANTIATIONS



***Compute-Intensive Workloads
(e.g., Deep Learning)***

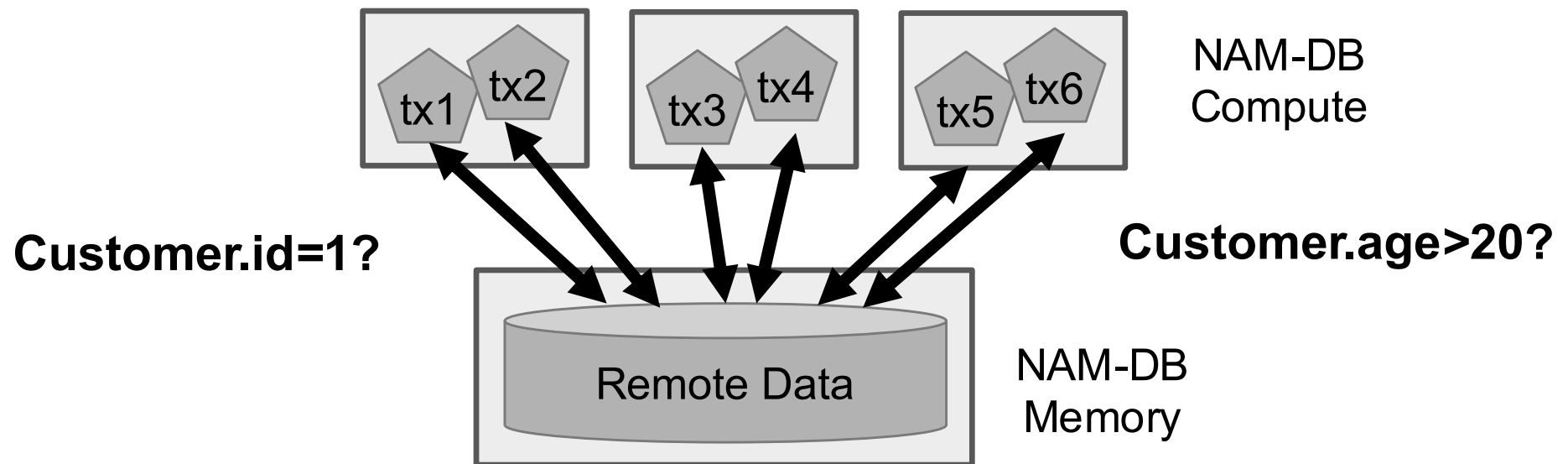


***Memory-Intensive Workloads
(e.g., OLTP and OLAP)***

NAM-DB: REMOTE TABLE ACCESS

Ziegler et al.: Designing Distributed Tree-based Indexes for RDMA. SIGMOD'19

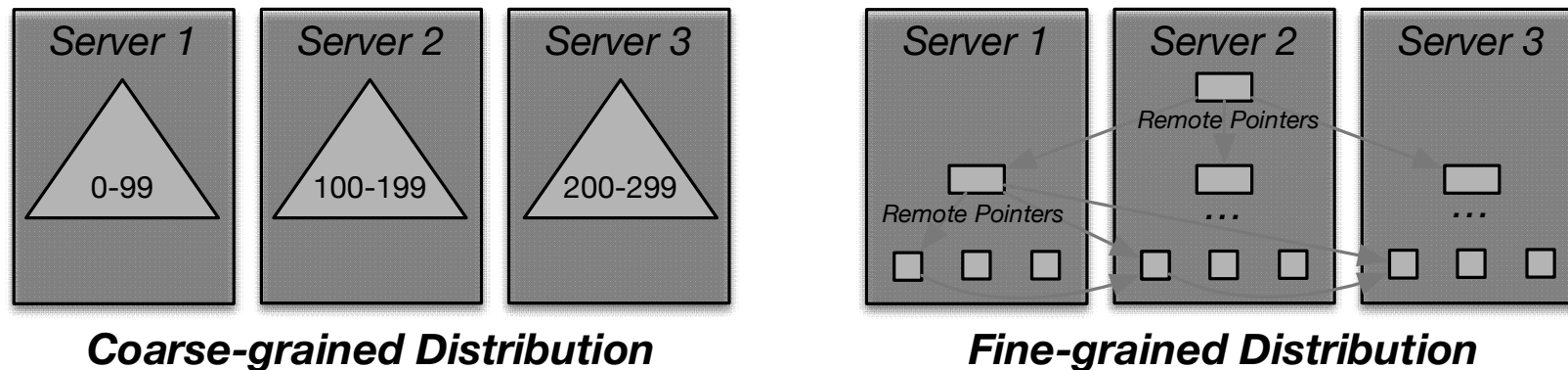
How to enable efficient remote access of remote tables (key and range lookups) on memory servers?



Key Question: How to design of tree-based indexes (i.e., B-tree like indexes) for RDMA?

NAM-DB: INDEX DESIGN SPACE

Index Distribution: How to distribute remote indexes across memory servers?

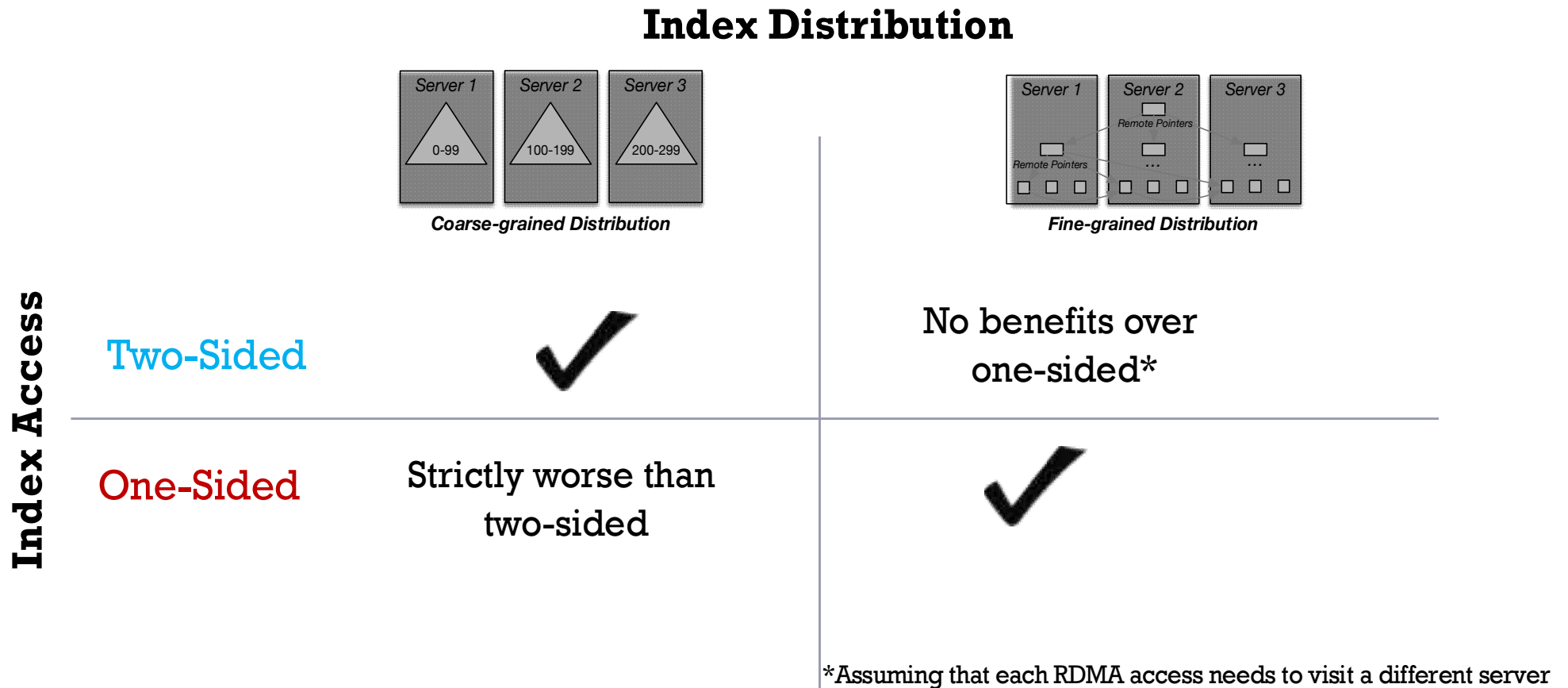


Index Access: How to implement index accesses from compute servers?

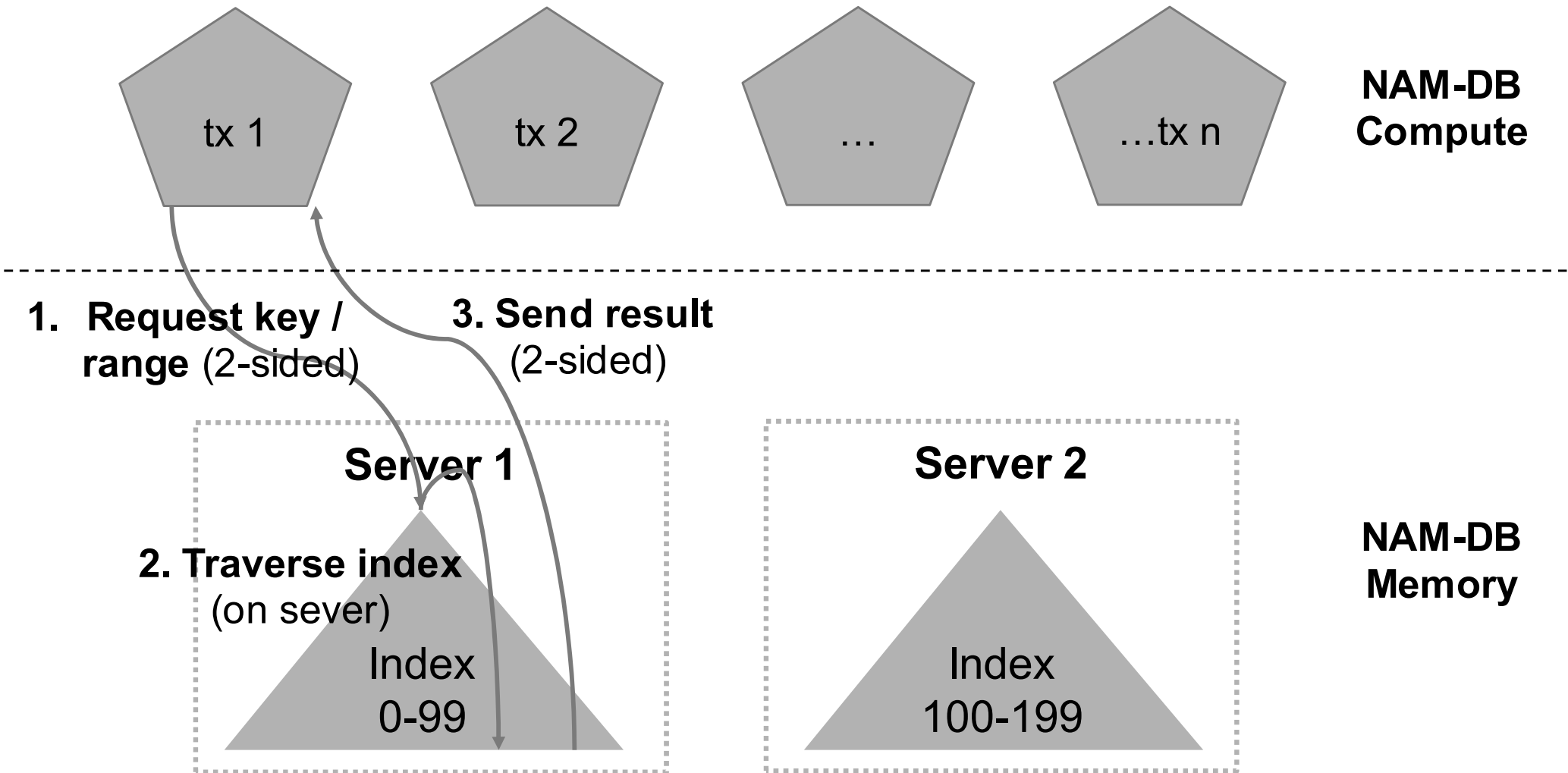
- One-Sided RDMA: Memory-based (READ / WRITE)
- Two-Sided RDMA: RPC-based (SEND / RCV)

NAM-DB: INDEX DESIGN SPACE

The “Design Matrix” for RDMA-based Indexes:

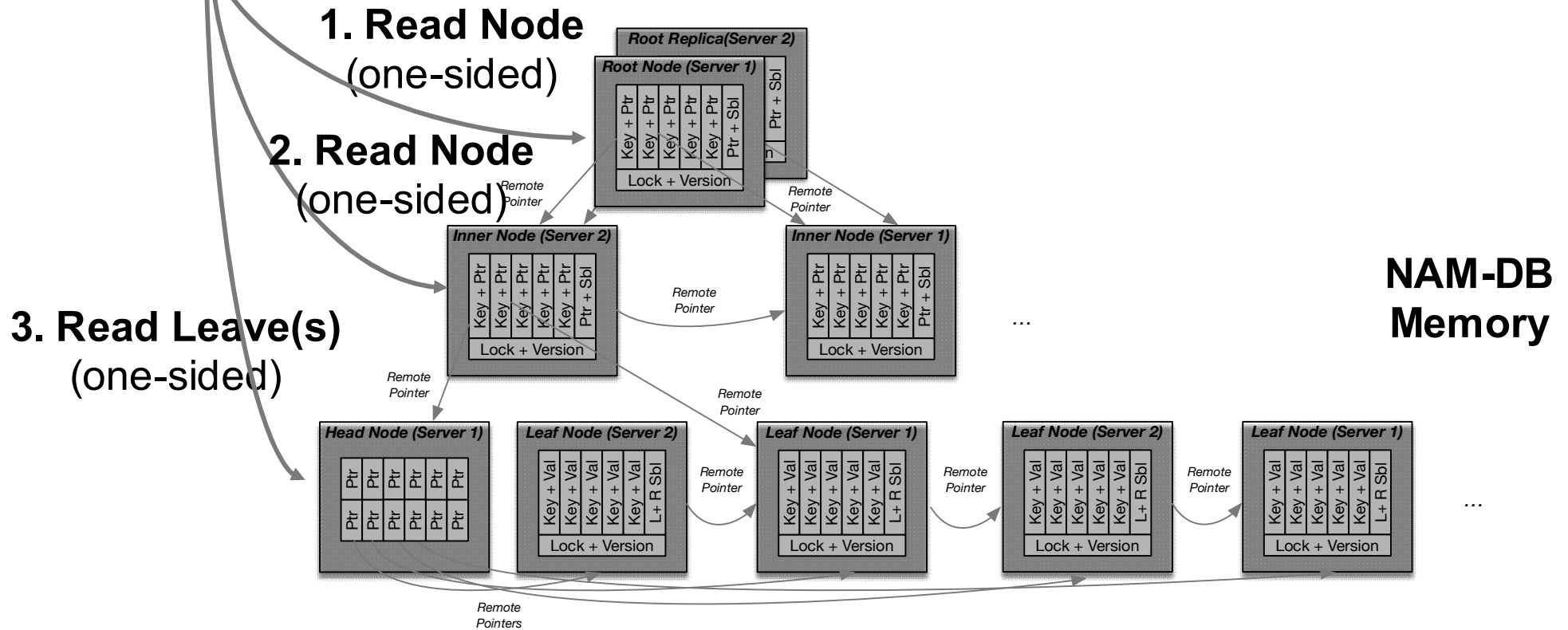
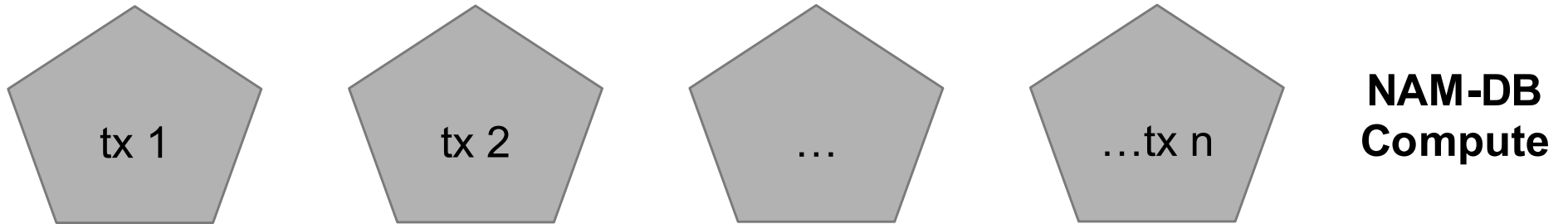


DESIGN 1: COARSE-GRAINED / 2-SIDED



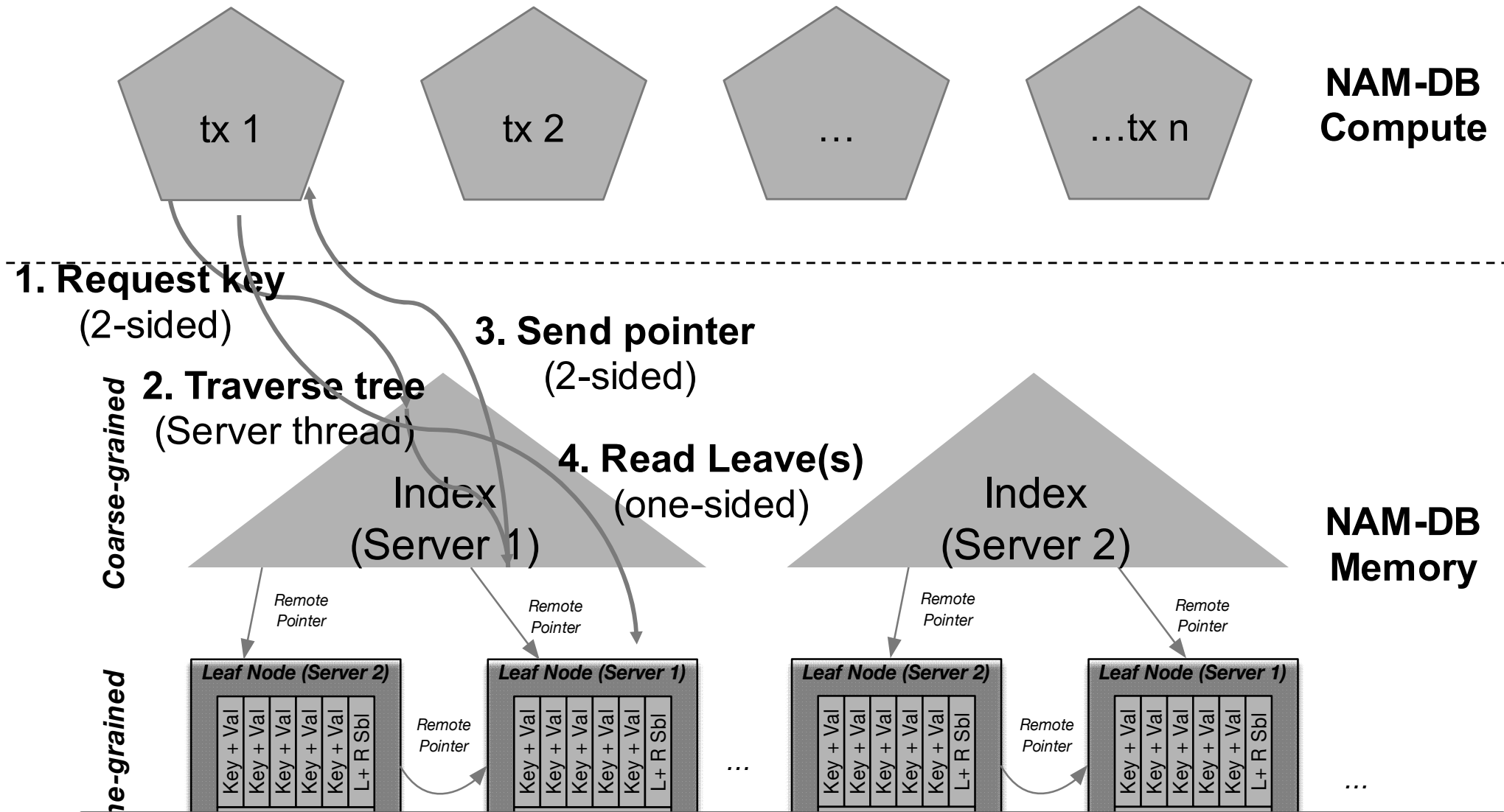
Only one roundtrip BUT sensitive to skewness

DESIGN 2: FINE-GRAINED / 1-SIDED



Multiple roundtrips BUT better load balancing

DESIGN 3: HYBRID (FINE/COARSE)



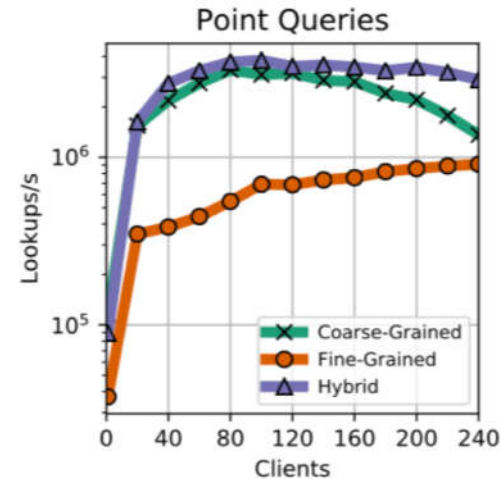
One roundtrip for index traversal +
Multiple reads for data but better load balancing

NAM-DB: EVALUATION (INDEXES)

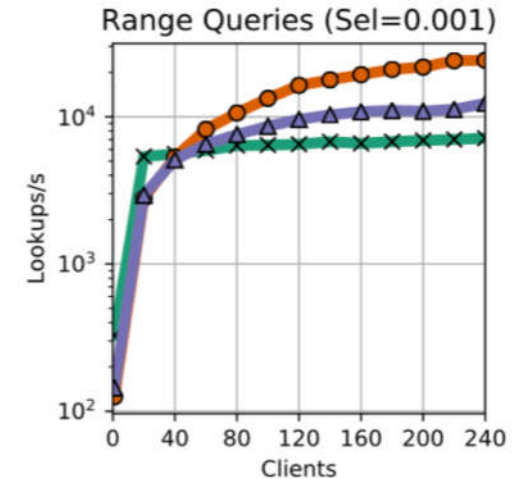
Index Workloads:

Workload	Point Queries	Range Queries (sel=s)	Inserts
A	100%		
B		100%	
C	95%		5%
D	50%		50%

Throughput (Workload A+B, Skewed):



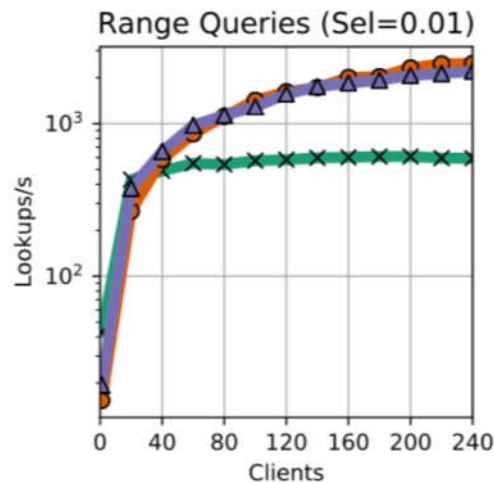
(a) Point Query



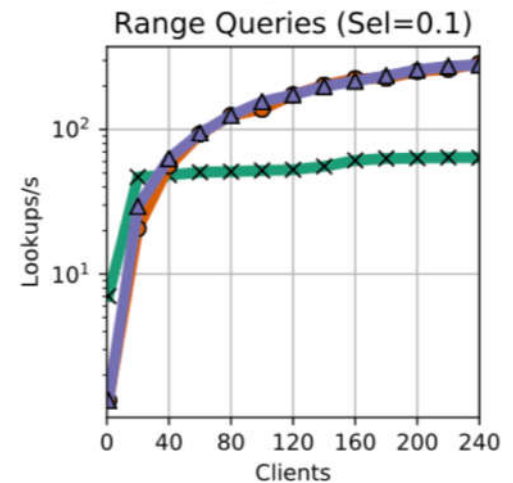
(b) Range Query (sel=0.001)

Setup:

- 4 Memory Servers
- 6 Compute Servers
- No co-location
- Data 100M unique keys

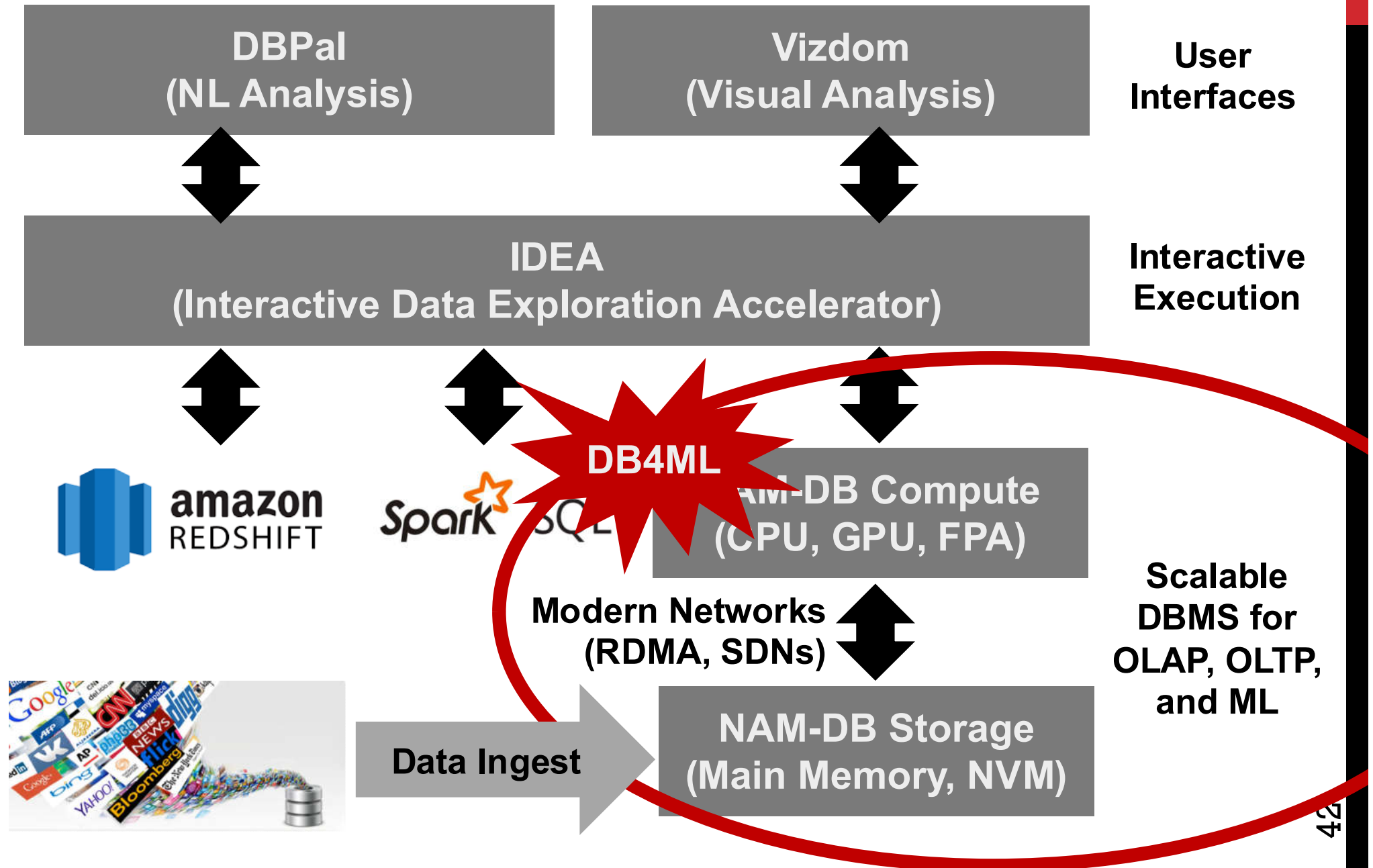


(c) Range Query (sel=0.01)



(d) Range Query (sel=0.1)

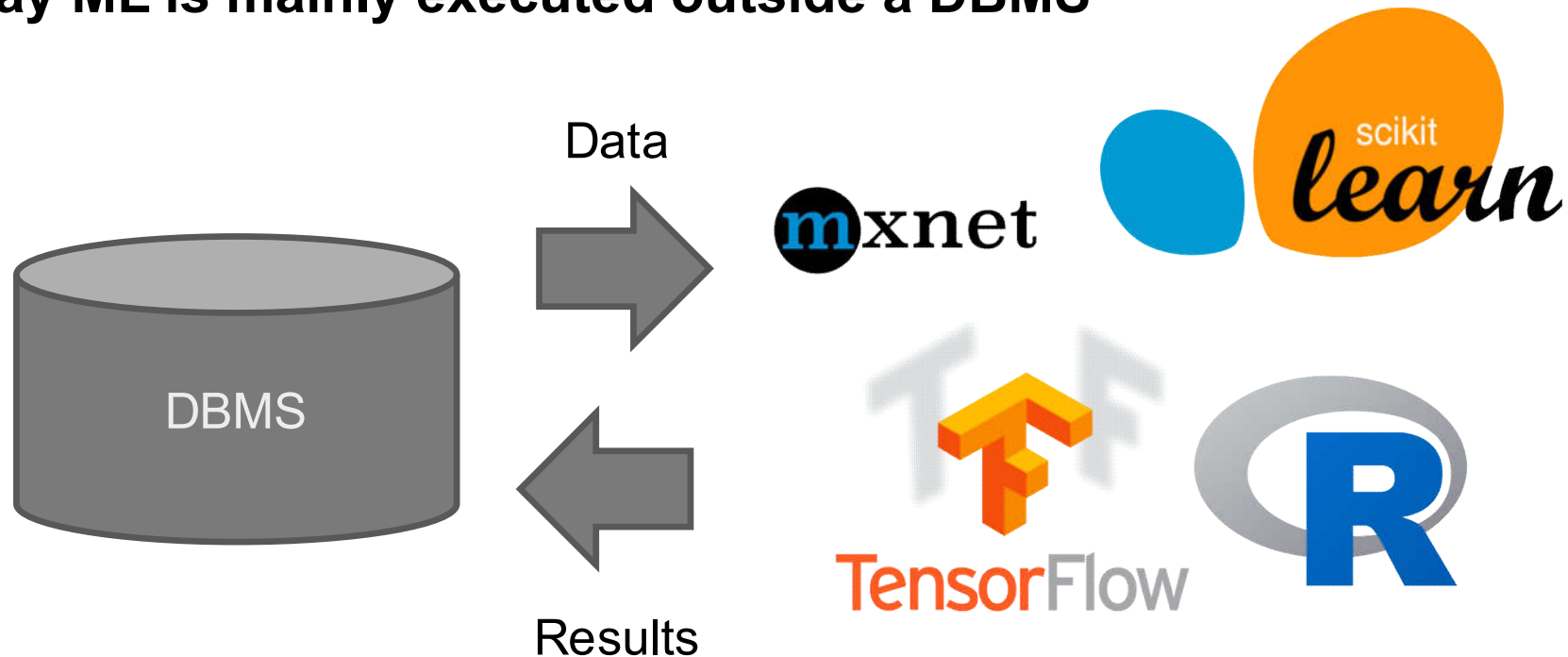
DARMSTADT DATA ANALYSIS STACK



PROJECT: DB4ML

Ziegler et al.: DB4ML – Distributed In-DBMS Machine Learning. To be submitted.

Today ML is mainly executed outside a DBMS



But most business data resides in DBMSs and thus expensive data transfers between DBMS and ML ecosystems are required

Goal of DB4ML: Enable ML inside a scalable DBMS (NAM-DB)

DB4ML: MAIN IDEA

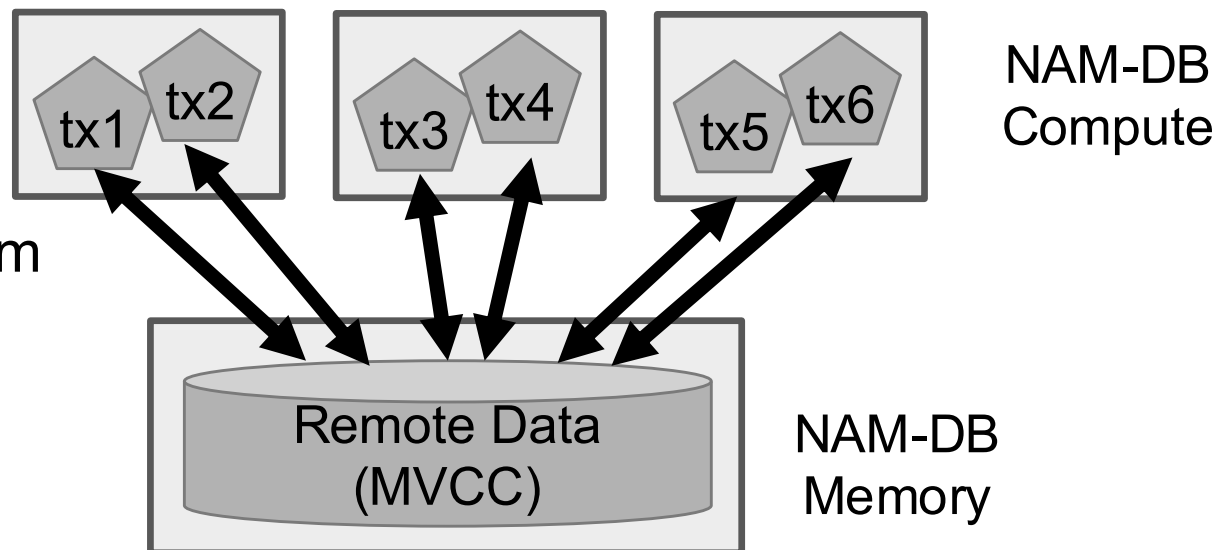
Existing approaches: integrate ML into DBMS extend query processing layer (e.g., MADLib)

However, modern ML algorithms make use of

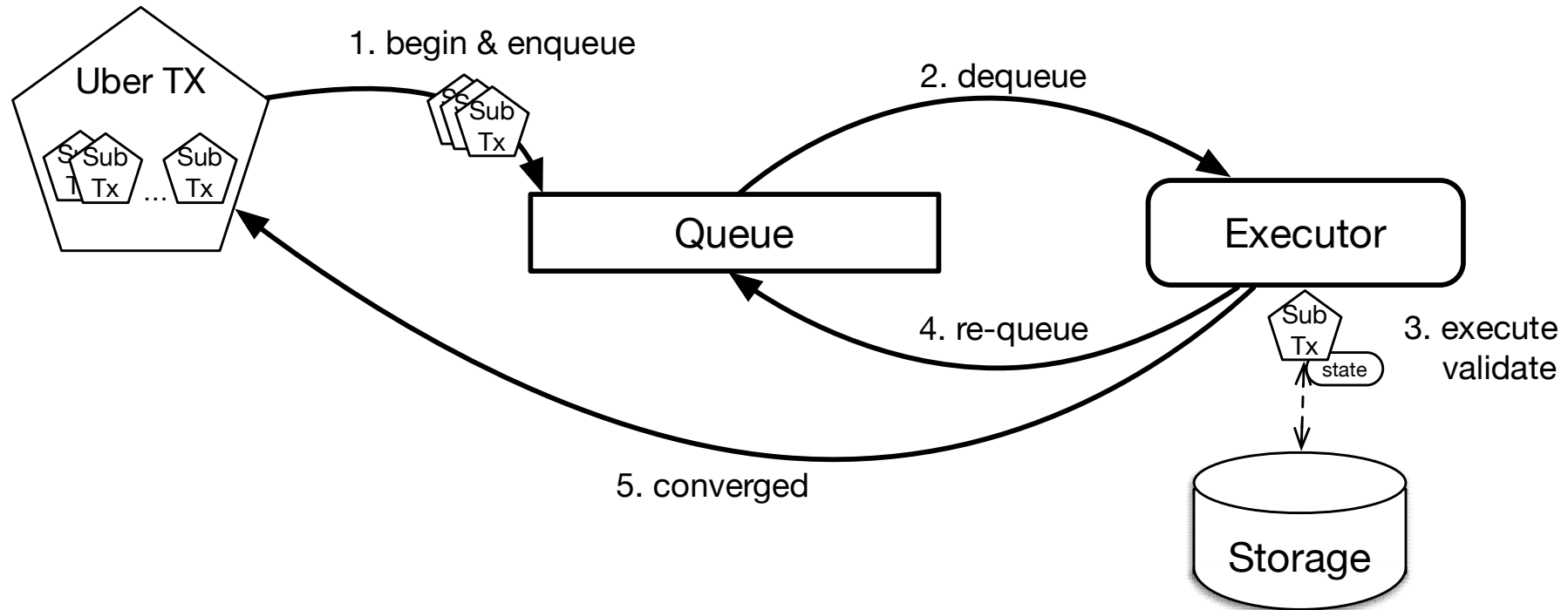
- fine-grained parallel execution and
- relaxed consistency to update shared state (e.g., bounded-staleness)

Main idea of DB4ML:

- Use transactions
-> fine-grain parallelism
- Use MVCC
-> ML consistency



DB4ML: EXECUTION MODEL



Transaction model

- **Uber transaction** “coordinates” many small sub-transactions
- **Sub-transactions** iteratively update shared state until convergence

Storage model

- Multi-version concurrency
- **Isolation levels:** Sync, Bounded-Staleness, Async

EXAMPLE: PARALLEL SGD

Uber-transaction

Algorithm 3: SGD – Uber-Transaction

```
1 BEGIN TRANSACTION
2   #rows = SELECT COUNT(*) FROM GlobalParameter
3   #subtxs = #cpu_cores
4   numEpochs = 20
5   batchSize = 2000
6   learnRate = 1.0
7   SET SUB-TX ISOLATION LEVEL
   {SYNC|ASYNC|BOUNDED-STALENESS}
8   for i = 0...#subtxs do
9     startKey = i * (#rows/#subtxs)
10    endKey = lowKey + (#rows/#subtxs) - 1
11    sub_tx = new sub_tx()
12    sub_tx.begin(numEpochs, batchSize, learnRate,
    startKey, endKey))
13  end
14  WAIT //until all sub_tx converged
15  COMMIT
16 END
```

Sub-transaction

Algorithm 4: SGD – Iterative Sub-Transaction

```
1 begin (T_State initial_state)
2   tx_state.currentEpoch = 0
   // Init local variables -> executed once
9 execute ()
10  localParamVector = read_parameters()
11  mini-batch = randomSamples(tx_state.lowKey,
    tx_state.highKey, tx_state.batchSize)
12  gradient = sgd(mini-batch,localParamVector)//Eq. (2)
   // Update global state -> executed iteratively
15  tx_state.currentEpoch++
16
17 validate () /After each execute() call
18  if tx_state.numberEpochs reached then
19    | return DONE //Finished all iterations
20  else
21    | return COMMIT //Commit one iteration
22  end
```

DB4ML: EVALUATION (PARALLEL SGD)

Baseline: HogWild! (Parallel SGD with only limited coordination)

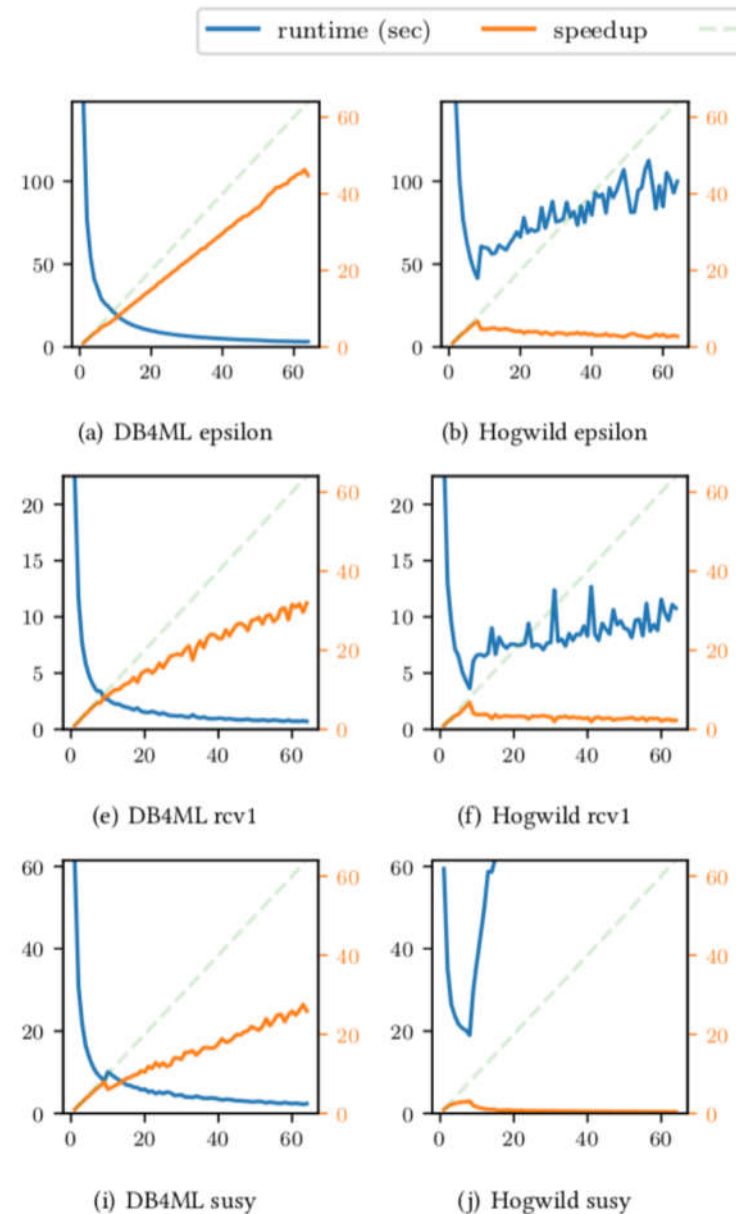
Data Sets

Dataset	Classes	Training set	Test set	Features
rcv1.binary	2	677,399	20,242	47236
susy	2	4,500,000	500,000	18
epsilon	2	400,000	100,000	2000

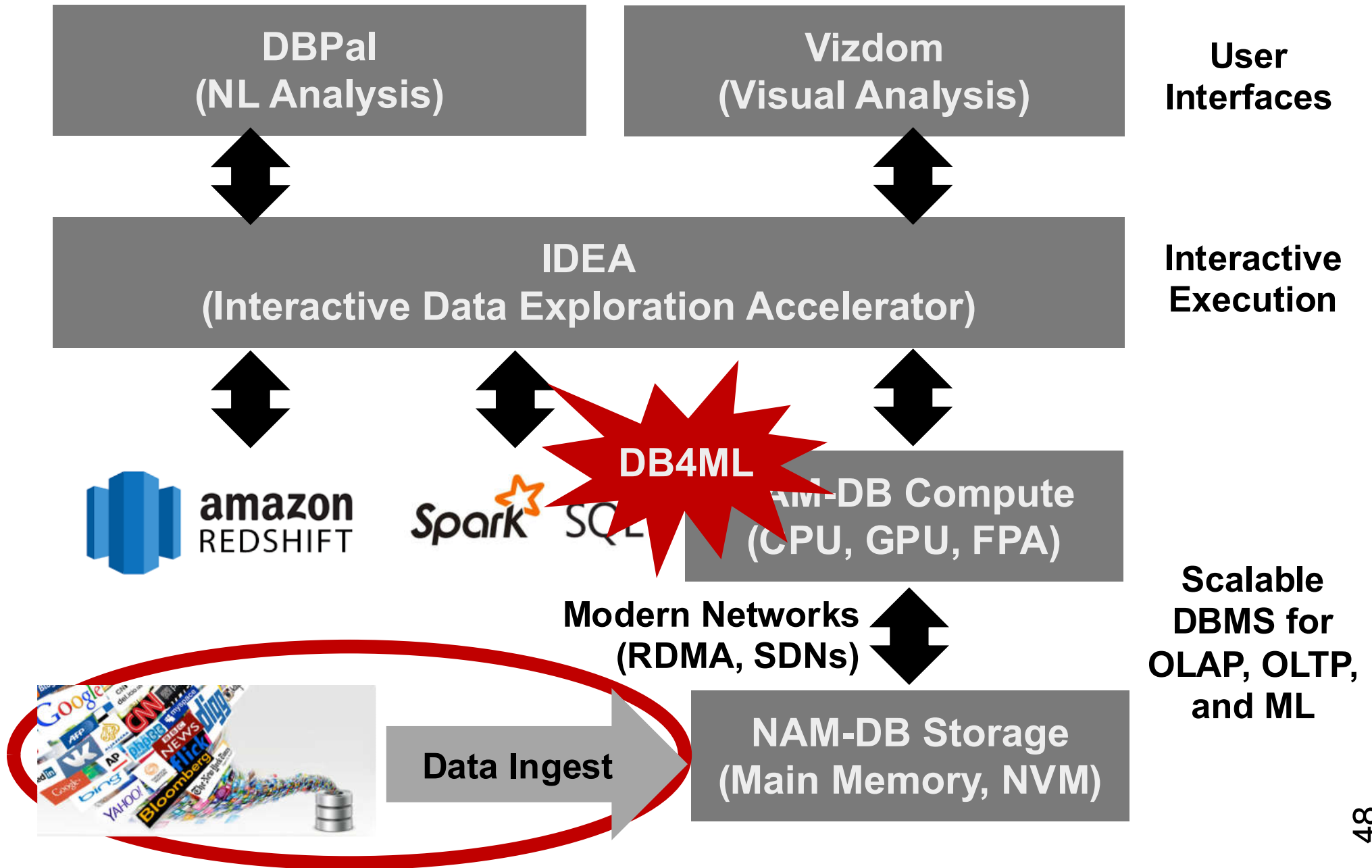
Hardware:

- Simulated distributed setup
- Single machine with 64 cores in 8 NUMA regions

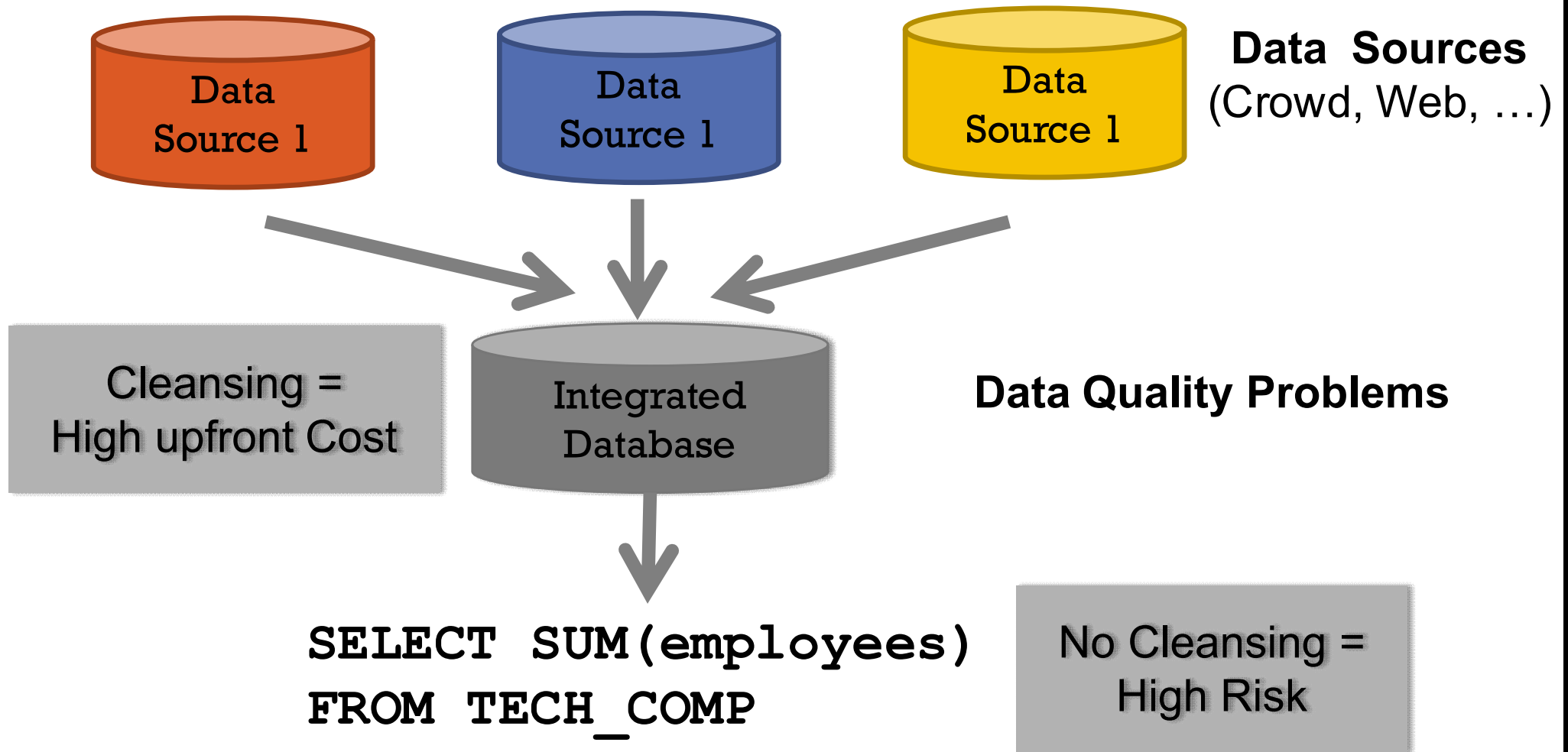
Runtime & Speedup:



DARMSTADT DATA ANALYSIS STACK



THE DATA QUALITY PROBLEM



Goal: Automatic Detection of Errors and Cleaning of Data

THE DATA QUALITY PROBLEM

Data-level Problems:

- Data Formatting Errors
- Data duplicates
- Missing values
- **Missing tuples**
- ...

Schema-level Problems :

- Naming Conflicts
- Structural Conflicts
- ...

PROBLEM: MISSING TUPLES

Idea: Estimate Impact of Missing Tuples + Correct Result

DB1

Company	Employees
Google	30K
Facebook	10K
Spotify	1K

DB2

Company	Employees
Google	30K
Microsoft	50K

DB3

Company	Employees
Facebook	10K
Snowflake	100
FutureLive	50

**SELECT SUM(employees)
FROM TECH_COMP**

1. Estimate COUNT (e.g. Chao84)

Statistics:

- Singletons: $f_1=4$
- Doubletons $f_2=2$

$$\begin{aligned}\text{Count}_{\text{est}} &= \text{Count}_{\text{obs}} + f_1^2 / 2 \cdot f_2 \\ &= 6 + 16/4 = 10\end{aligned}$$

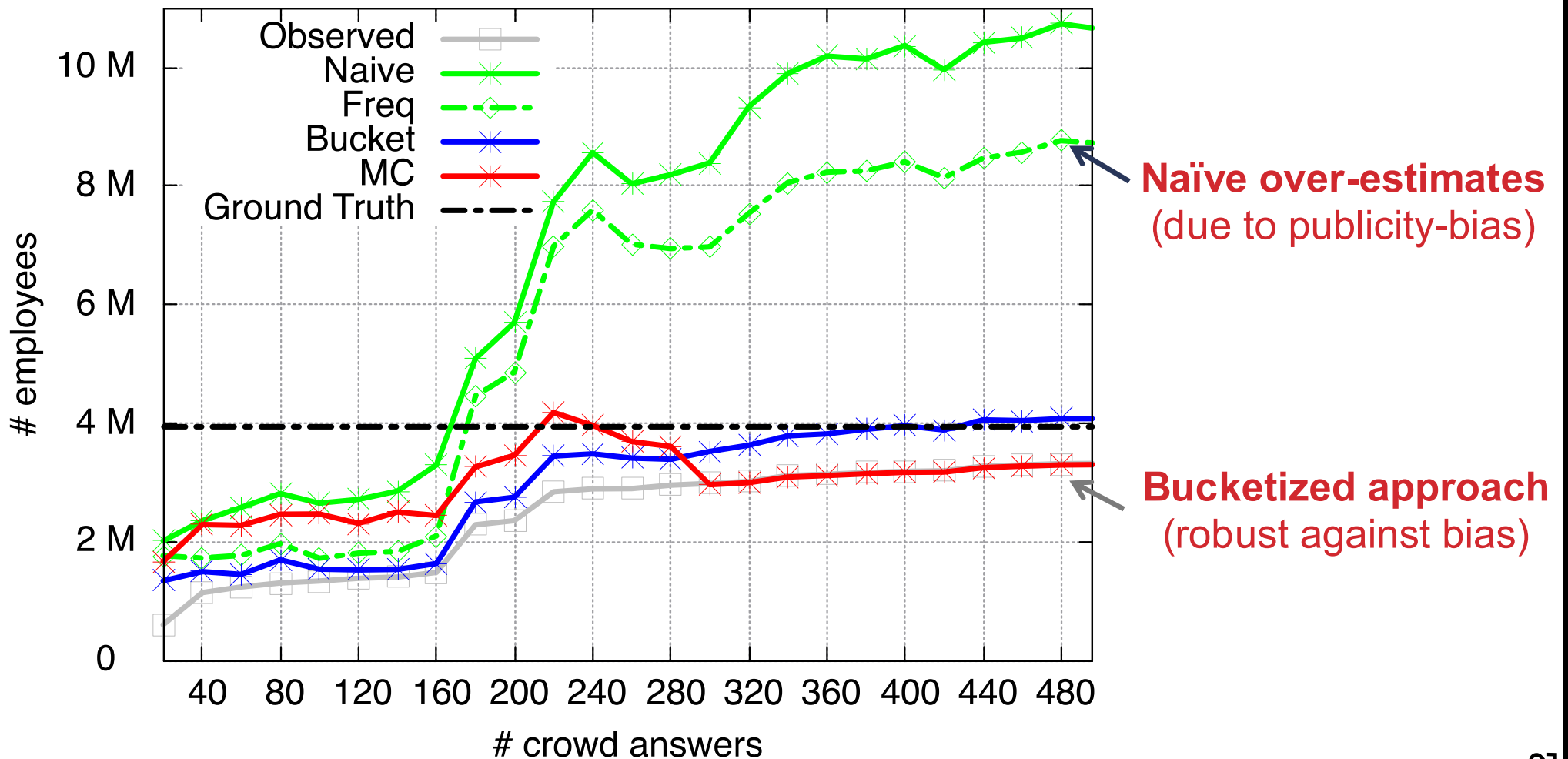
2. Correct Query Results

$$\text{Sum}_{\text{est}} = \text{Count}_{\text{est}} \cdot \text{Avg}_{\text{obs}}$$

EVALUATION: REAL WORLD DATA

Chung et al.: Estimating the Impact of Unknown Unknowns on Aggregate Query Results.
SIGMOD'16

Number of employees in US tech companies



CURRENT AND FUTURE DIRECTIONS

Vision: Support Non-ML Experts to interactively curate End-to-end ML Pipelines

Different directions of my group:

- **Conversational Natural Language Interfaces**
(Chatbot-like Interfaces for Users and Machine)
- **Interactive Machine Learning for Non-ML Experts**
(Combine AutoML and user feedback optimally)
- **Scalable Heterogeneous Computing**
(Distributed Deep Learning → GPUs and RDMA, ...)
- ...

COLLABORATORS

