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?
- UnknownUnknowns (Data Quality)
- IncMap (Schema Mapping)
- Sherlock (Text Summarization)

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

DBPal (NL Analysis)

Vizdom (Visual Analysis)

IDEA (Interactive Data Exploration Accelerator)

NAM-DB Compute (CPU, GPU, FPGA)

Modern Networks (RDMA, SDNs)

NAM-DB Storage (Main Memory)

User Interfaces

Interactive Execution

Scalable DBMS for OLAP, OLTP, and ML

Data Ingest
DARMSTADT DATA ANALYSIS STACK

DBPal (NL Analysis)

Vizdom (Visual Analysis)

IDEA (Interactive Data Exploration Accelerator)

NAM-DB Compute (CPU, GPU, FPGA)

Modern Networks (RDMA, SDNs)

NAM-DB Storage (Main Memory)

User Interfaces

Interactive Execution

Scalable DBMS for OLAP, OLTP, and ML

Data Ingest
vizdom

Interactive Analytics through Pen and Touch

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1</td>
</tr>
<tr>
<td>CPU</td>
<td>1</td>
</tr>
<tr>
<td>CPU</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>3</td>
</tr>
<tr>
<td>CPU</td>
<td>4</td>
</tr>
<tr>
<td>Disk</td>
<td>1</td>
</tr>
<tr>
<td>Disk</td>
<td>2</td>
</tr>
<tr>
<td>Monitor</td>
<td>1</td>
</tr>
</tbody>
</table>

Sales-Sample

<table>
<thead>
<tr>
<th>Product</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1</td>
</tr>
<tr>
<td>CPU</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>3</td>
</tr>
<tr>
<td>Disk</td>
<td>2</td>
</tr>
</tbody>
</table>

Sampling (Online OR Offline)

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

Select `AVG(salary)`
From `census`
Group By `gender`

<table>
<thead>
<tr>
<th>gender</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>66.66k</td>
</tr>
<tr>
<td>m</td>
<td>100k</td>
</tr>
</tbody>
</table>

**Intermediate Results**

<table>
<thead>
<tr>
<th>gender</th>
<th>SUM</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>200k</td>
<td>3</td>
</tr>
<tr>
<td>m</td>
<td>300k</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gender</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>100k</td>
</tr>
<tr>
<td>m</td>
<td>90k</td>
</tr>
<tr>
<td>f</td>
<td>40k</td>
</tr>
<tr>
<td>m</td>
<td>110k</td>
</tr>
<tr>
<td>m</td>
<td>100k</td>
</tr>
<tr>
<td>f</td>
<td>60k</td>
</tr>
</tbody>
</table>

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

Executed Interactions

Result Cache (Random Variables)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{\text{male}} )</td>
<td>( {0.70, \varepsilon_1} )</td>
</tr>
<tr>
<td>( P_{\text{female}} )</td>
<td>( {0.30, \varepsilon_2} )</td>
</tr>
<tr>
<td>( P_{\text{high}} )</td>
<td>( {0.20, \varepsilon_3} )</td>
</tr>
<tr>
<td>( P_{\text{low}} )</td>
<td>( {0.80, \varepsilon_4} )</td>
</tr>
<tr>
<td>( P_{\text{low}</td>
<td>\text{male}} )</td>
</tr>
<tr>
<td>( P_{\text{low}</td>
<td>\text{female}} )</td>
</tr>
<tr>
<td>( P_{\text{high}</td>
<td>\text{male}} )</td>
</tr>
<tr>
<td>( P_{\text{low}</td>
<td>\text{female}} )</td>
</tr>
</tbody>
</table>
IDEA: RESULT REUSE

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

Rewrites for Reuse:
- Bayes Theorem
- Law of Total Probabilities
- Inclusion/exclusion Principle

Bayes’ Theorem:

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

\[
\hat{P}_{\text{male}|\text{high}} = \frac{P_{\text{high}|\text{male}} \cdot P_{\text{male}}}{P_{\text{high}}} \approx 0.88
\]
IDEA: EXPERIMENTAL EVALUATION

Workload: Exploration Sessions (User Study)

<table>
<thead>
<tr>
<th>#</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>sex</td>
</tr>
<tr>
<td>#2</td>
<td>education</td>
</tr>
<tr>
<td>#3</td>
<td>education WHERE sex='Female'</td>
</tr>
<tr>
<td>#4</td>
<td>education WHERE sex='Male'</td>
</tr>
<tr>
<td>#5</td>
<td>sex, education</td>
</tr>
<tr>
<td>#6</td>
<td>sex WHERE education='PhD'</td>
</tr>
<tr>
<td>#7</td>
<td>salary</td>
</tr>
<tr>
<td>#8</td>
<td>salary WHERE education='PhD'</td>
</tr>
<tr>
<td>#9</td>
<td>sex, salary</td>
</tr>
<tr>
<td>#10</td>
<td>salary WHERE sex='Female'</td>
</tr>
<tr>
<td>#11</td>
<td>salary</td>
</tr>
<tr>
<td>#12</td>
<td>salary WHERE sex='Female'</td>
</tr>
<tr>
<td>#13</td>
<td>salary WHERE sex&lt;&gt;='Female'</td>
</tr>
<tr>
<td>#14</td>
<td>salary WHERE sex='Female' AND education='PhD', salary WHERE sex&lt;&gt;='Female' AND education='PhD'</td>
</tr>
<tr>
<td>#15</td>
<td>age</td>
</tr>
<tr>
<td>#16</td>
<td>salary WHERE 20&lt;=age&lt;40 AND sex='Female' AND education='PhD', salary WHERE 20&lt;=age&lt;40 AND sex&lt;&gt;='Female' AND education='PhD'</td>
</tr>
</tbody>
</table>

Evaluated Systems:

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

Data: 500M tuples

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

DBPal (NL Analysis) → IDEA (Interactive Data Exploration Accelerator) → NAM-DB Compute (CPU, GPU, FPA)

NAM-DB Compute (CPU, GPU, FPA) → Modern Networks (RDMA, SDNs) → NAM-DB Storage (Main Memory, NVM)

DBPal (NL Analysis) → IDEA (Interactive Data Exploration Accelerator) → User Interfaces

User Interfaces

Interactive Execution

Scalable DBMS for OLAP, OLTP, and ML

Data Ingest
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

Natural Language → ? → SQL

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)
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

Input

DB Schema

Cover variety of SQL

Generate NL/SQL pairs using templates

SQL / NL Pairs

Automatically augment NL/SQL pairs

Augmented NL/SQL Pairs

Output

DBPAL: GENERATING TRAINING DATA

Input
DB Schema
Generate NL/SQL pairs using templates

Generate NL/SQL pairs

SQL / NL Pairs
Automatically augment NL/SQL pairs

Output
Augmented NL/SQL Pairs

Cover variety of SQL
Cover variety of NL

Template

NL/ SQL Pair

Augmentation
Paraphrasing
Show me the names of patients diagnosed fever?

Noising
Show the names of patients with diagnosed fever?

Input

DB Schema

Generate NL/SQL pairs using templates

Cover variety of SQL

Output

Augmented NL/SQL Pairs

Cover variety of NL

Template

NL/ SQL Pair

Augmentation
Paraphrasing
Show me the names of patients diagnosed fever?

Noising
Show the names of patients with diagnosed fever?

Patient Database

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>diagnoses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carsten</td>
<td>39</td>
<td>fever</td>
</tr>
<tr>
<td>Emilie</td>
<td>8</td>
<td>flu</td>
</tr>
<tr>
<td>Frederik</td>
<td>4</td>
<td>fever</td>
</tr>
</tbody>
</table>

Millions of different NL/SQL pairs
### DBPAL: EXPERIMENTAL RESULTS

#### Patient and Geo Benchmark

<table>
<thead>
<tr>
<th></th>
<th>Patients</th>
<th>GeoQuery</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaLIR (w/o feedback)</td>
<td>15.60%</td>
<td>7.14%</td>
</tr>
<tr>
<td>NaLIR (w feedback)</td>
<td>21.42%</td>
<td>N/A</td>
</tr>
<tr>
<td>NSP++</td>
<td>N/A</td>
<td>83.9%</td>
</tr>
<tr>
<td>NSP (template only)</td>
<td>10.60%</td>
<td>5.0%</td>
</tr>
<tr>
<td>DBPal (w/o augmentation)</td>
<td>74.80%</td>
<td>38.60%</td>
</tr>
<tr>
<td>DBPal (full pipeline)</td>
<td>75.93%</td>
<td>55.40%</td>
</tr>
</tbody>
</table>

#### Patient Benchmark (Breakdown per Linguistic Category)

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

#### 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)
nathaniel@titanx:~$ ./interactive.sh

Loading model...
indexing database...
select distinct first_name from patients
select distinct last_name from patients
select distinct gender from patients
select distinct diagnosis from patients
preparing lemmatizer...
type ":q" to exit
nl query: 

1


DARMSTADT DATA ANALYSIS STACK

IDEA
(Interactive Data Exploration Accelerator)

DBPal
(NL Analysis)

Vizdom
(Visual Analysis)

User Interfaces

Interactive Execution

NAM-DB Storage
(Main Memory, NVM)

NAM-DB Compute
(CPU, GPU, FPA)

Modern Networks
(RDMA, SDNs)

Data Ingest

Scalable DBMS for OLAP, OLTP, and ML
Network Communication is evil: Must be avoided at all cost

### 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, ...)

<table>
<thead>
<tr>
<th>RAM</th>
<th>Network 1Gbps</th>
<th>Net/RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency, Random 1KB (μs)</td>
<td>0.1</td>
<td>100</td>
</tr>
<tr>
<td>Throughput (GB/s)</td>
<td>51.2 (4 channels)</td>
<td>0.125</td>
</tr>
</tbody>
</table>
MODERN HIGH-SPEED NETWORKS

Bandwidth (GB/s)

<table>
<thead>
<tr>
<th>InfiniBand</th>
<th>QDR</th>
<th>FDR-10</th>
<th>FDR</th>
<th>EDR</th>
<th>DDR3</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x, 4x, 12x</td>
<td>1x, 4x, 12x</td>
<td>1x, 4x, 12x</td>
<td>1x, 4x, 12x</td>
<td>1x, 4x, 12x</td>
<td>1333, 1600, 1866, 2133</td>
<td></td>
</tr>
</tbody>
</table>

(one channel)
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

![Graph showing performance comparison between different network architectures.]

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)
Network-Attached Memory (NAM) Architecture:

**Network-Attached Memory (NAM) Architecture:**

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

**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)

**Execution:**
OLAP, OLTP, and ML, …

**Shared State:**
Versioned Tables, Indexes, …

*Project: NAM-DB*
*Binnig et al.: The End of Slow Networks: It's Time for a Redesign. PVLDB’16*
NAM-DB: DIFFERENT INSTANTIATIONS

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

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

- **Coarse-grained Distribution**
  - Server 1: 0-99
  - Server 2: 100-199
  - Server 3: 200-299

- **Fine-grained Distribution**
  - Server 1: Remote Pointers
  - Server 2: Remote Pointers
  - Server 3: Remote Pointers

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

- One-Sided RDMA: Memory-based (READ / WRITE)
- Two-Sided RDMA: RPC-based (SEND / RCV)
The “Design Matrix” for RDMA-based Indexes:

Index Distribution

Coarse-grained Distribution

Server 1: 0-99
Server 2: 100-199
Server 3: 200-299

Fine-grained Distribution

Server 1
Server 2
Server 3
Remote Pointers

Index Access

Two-Sided  ✓

One-Sided

Strictly worse than two-sided

No benefits over one-sided*

*Assuming that each RDMA access needs to visit a different server
DESIGN 1: COARSE-GRAINED / 2-SIDED

1. Request key / range (2-sided)
2. Traverse index (on server)
3. Send result (2-sided)

Only one roundtrip BUT sensitive to skewness
DESIGN 2: FINE-GRAINED / 1-SIDED

1. Read Node (one-sided)
2. Read Node (one-sided)
3. Read Leave(s) (one-sided)

Multiple roundtrips BUT better load balancing
DESIGN 3: HYBRID (FINE/COARSE)

1. Request key (2-sided)
2. Traverse tree (Server thread)
3. Send pointer (2-sided)
4. Read Leave(s) (one-sided)

One roundtrip for index traversal + Multiple reads for data but better load balancing
## NAM-DB: EVALUATION (INDEXES)

### Index Workloads:

<table>
<thead>
<tr>
<th>Workload</th>
<th>Point Queries</th>
<th>Range Queries (sel=s)</th>
<th>Inserts</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100%</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>95%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>50%</td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Throughput (Workload A+B, Skewed):

Setup:
- 4 Memory Servers
- 6 Compute Servers
- No co-location
- Data 100M unique keys
DARMSTADT DATA ANALYSIS STACK

DBPal
(NL Analysis)

Vizdom
(Visual Analysis)

IDEA
(Interactive Data Exploration Accelerator)

DB4ML

NAM-DB Compute
(CPU, GPU, FPA)

Modern Networks
(RDMA, SDNs)

NAM-DB Storage
(Main Memory, NVM)

Data Ingest

User Interfaces

Interactive Execution

Scalable DBMS for
OLAP, OLTP, and ML
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
**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,
13                  startKey, endKey))
14 end
15 WAIT //until all sub tx converged
16 COMMIT
17 END

Sub-transaction

Algorithm 4: SGD – Iterative Sub-Transaction

1 begin (T State initial_state)
2     tx_state.currentEpoch = 0
3     // Init local variables -> executed once
4 end
5 execute ()
6     localParamVector = read_parameters()
7     mini-batch = randomSamples(tx_state.lowKey,
8                              tx_state.highKey, tx_state.batchSize)
9     gradient = sgd(mini-batch, localParamVector)//Eq. (2)
10     // Update sgd global state -> executed iteratively
11     tx_state.currentEpoch++
12     if tx_state.numberEpochs reached then
13         return DONE  //Finished all iterations
14     else
15         return COMMIT //Commit one iteration
16     end
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**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Training set</th>
<th>Test set</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcv1.binary</td>
<td>2</td>
<td>677,399</td>
<td>20,242</td>
<td>47236</td>
</tr>
<tr>
<td>susy</td>
<td>2</td>
<td>4,500,000</td>
<td>500,000</td>
<td>18</td>
</tr>
<tr>
<td>epsilon</td>
<td>2</td>
<td>400,000</td>
<td>100,000</td>
<td>2000</td>
</tr>
</tbody>
</table>

**Hardware:**

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

**Runtime & Speedup:**

![Runtime and Speedup Graphs](image)
DARMSTADT DATA ANALYSIS STACK

DBPal (NL Analysis)

Vizdom (Visual Analysis)

IDEA (Interactive Data Exploration Accelerator)

DB4ML

NAM-DB Compute (CPU, GPU, FPA)

Modern Networks (RDMA, SDNs)

NAM-DB Storage (Main Memory, NVM)

User Interfaces

Interactive Execution

Scalable DBMS for OLAP, OLTP, and ML

Data Ingest
THE DATA QUALITY PROBLEM

Data Sources (Crowd, Web, …)

Data Quality Problems

Cleansing = High upfront Cost

Integrated Database

SELECT SUM(employees) FROM TECH_COMP

No Cleansing = High Risk

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

1. Estimate COUNT (e.g. Chao84)

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

$$\text{Count}_{\text{est}} = \text{Count}_{\text{obs}} + \frac{f_1^2}{2 \cdot f_2}$$
$$= 6 + \frac{16}{4} = 10$$

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

Naïve over-estimates (due to publicity-bias)

Bucketized approach (robust against bias)
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 $\rightarrow$ GPUs and RDMA, …)

• …
COLLABORATORS