Beyond the Wall: Near-Data Processing for Databases

Sam Xi, Ore Babarinsa, Manos Athanassoulis, Stratos Idreos
Memory Wall
Memory Wall

Performance vs. Time Chart:
- **CPU**: ~20-25% perf. increase annually
- **DRAM**: ~2-11% perf. increase annually
Memory-optimized data systems
Data access *remains* the bottleneck
We are not the first to visit this pyramid!
Near-data processing

Intelligent RAM

DIVA

Logic-in-memory

RADram

Terasys
Why did NDP not take off?

<table>
<thead>
<tr>
<th>Leakage</th>
<th>DRAM</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Switching speed</th>
<th>DRAM</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td></td>
<td>Fast</td>
</tr>
</tbody>
</table>

Fabrication processes are **incompatible**
Moore’s Law + Dennard scaling provided consistent performance scaling for years

Moore’s Law.

Dennard scaling.

Not the case anymore!
Our approach

HARP
Q100
Widx

Ibex

Secondary storage

DRAM
L3
L2
L1
CPU
Outline

Intro

NDP for data systems: Past and present

The architecture of JAFAR

Experimental results

Conclusion
Opportunity for NDP

Host server

Lots of data

Query

Database

Many rows fail the query predicate and are discarded.

Filter data before it is sent to CPU.
JAFAR: “Just” A Filtering Accelerator on Relations
JAFAR: Overall design

```
JAFAR

CPU  CPU  CPU  CPU

Last level cache

System bus + memory controller

DRAM
```
JAFAR context

From CPU

Memory access arbiter

RAS

CAS

Bank 0

Sense Amps

JAFAR

IO buffer
JAFAR architecture

From IO buffer

Data latch

Comparison is true?

ALU

ALU

Opcode

Left

Right

Opcode

page offset bitmask

Page offset counter

write enable

Output buffer
int errno = select_jafar(
    void* col_data,
    int range_low,
    int range_high,
    uint8_t* out_buf,
    size_t num_input_rows,
    size_t* num_output_rows);
Handling multiple modules

- CPU
- CPU
- CPU
- CPU
- Last level cache
- System bus + memory controller
- JAFAR
- DRAM
Handling multiple modules
Fill up each module first

CPU
CPU
CPU
CPU

Last level cache

System bus + memory controller

JAFAR
JAFAR

DRAM
Handling multiple modules
Interleave data across modules
Coordinating memory access

The CPU and JAFAR cannot simultaneously attempt to access memory.

CPU grants JAFAR ownership to a DRAM rank for a period of time.

Possible mechanism: DRAM mode registers
Experimental setup
Simulation framework

Out-of-order CPU

Classic cache model

SimpleDRAM

gem5

ALADDIN
Experimental setup
Queries, input data, and database

```
select * from table where column < n;
```

In-house column store database
4 million rows of unsorted integers
Experimental results

![Bar chart showing speedup vs. selectivity (%)]
Memory contention

Scheduling of ownership transfers will be important

What would JAFAR’s performance look like without a scheduler?
Memory contention

CPU

Memory requests

Memory requests

Idle period

JAFAR can execute
Idle periods on TPC-H

![Bar chart showing idle periods for different TPC-H queries.](image)
JAFAR as a framework
More operators

- Aggregations ✔
- Projections ✔
- Sort ✔
- Joins ?
JAFAR as a framework
Data types and layouts

Row-stores and hybrids
  *Multiple filters per row*
  *Efficient projections*

Variable length datatypes
  *Process on CPU?*
NDP is an exciting opportunity for innovation in data systems
NDP is a promising solution to the memory wall for data systems.

JAFAR provides up to 9x speedup on simple select queries.

JAFAR is built on an extensible framework for accelerating data systems.
Thank you