Hardware Acceleration for Database Systems using Content Addressable Memories

Nagender Bandi, Sam Schneider, Divyakant Agrawal, Amr El Abbadi
University of California, Santa Barbara

DaMoN 2005

Overview

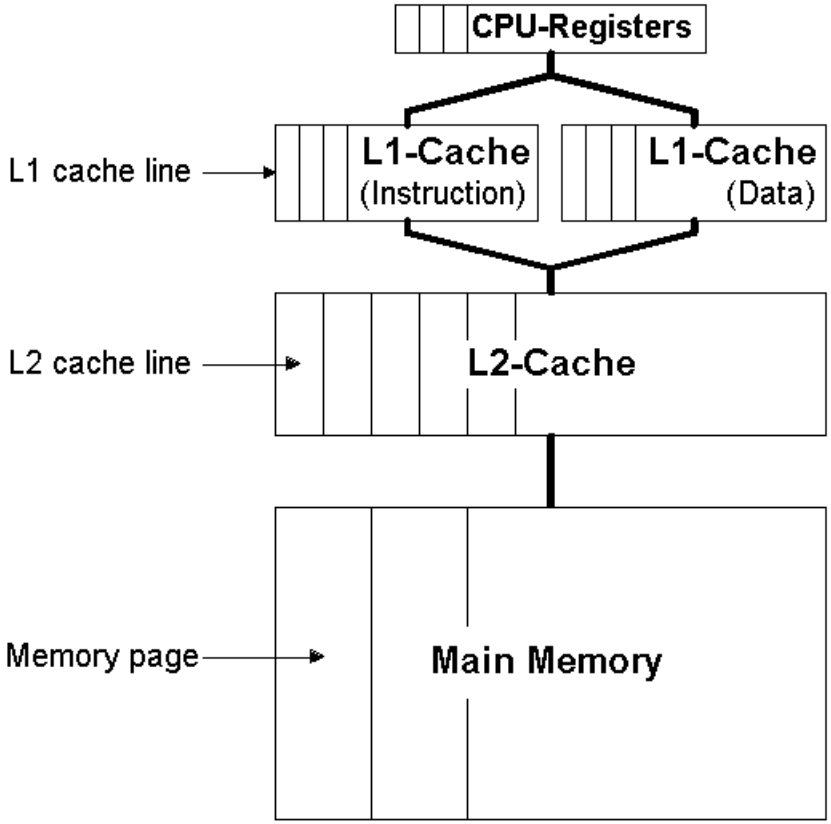
- The “Memory Bottleneck”
  - What is it?
  - How does the Memory Bottleneck effect Database Performance?
- Networking hardware - Can CAM-technology help?
  - Content Addressable Memories (CAMs)
- CAM-Cache Architecture
  - Database equi-join
- Concluding Remarks

The "Memory Bottleneck"

- CPU speed increases about 70% each year
- Memory speed has increased about 50% over the last decade

Memory is increasingly becoming a performance bottleneck

- Solution: "Memory Hierarchy"
  - Faster, smaller and more expensive memories (caches)
    - Data kept near CPU
    - Smaller latency



The “Memory Bottleneck”

- But..

  ..caches improve latency only, if requested data is found in cache.

- Some quotes:
  - “Unfortunately, recent studies report that faster processors do not improve database system performance to the same extent as scientific workloads.” “On the average, half the execution time is spent on stalls.” [Ailamaki et al. VLDB 1999]

  - “..unless special care is taken, a database server running even a simple sequential scan on a table will spend 95% of its cycles waiting for memory to be accessed.”

    “..we must draw the sad conclusion that if no attention is paid in query processing to data locality, all advances in CPU power are neutralized due to the memory access bottleneck.”

    “ This trend of improvement in bandwidth and standstill in latency is expected to continue, with no real solutions in sight.”
    [Boncz et al. VLDB 1999]

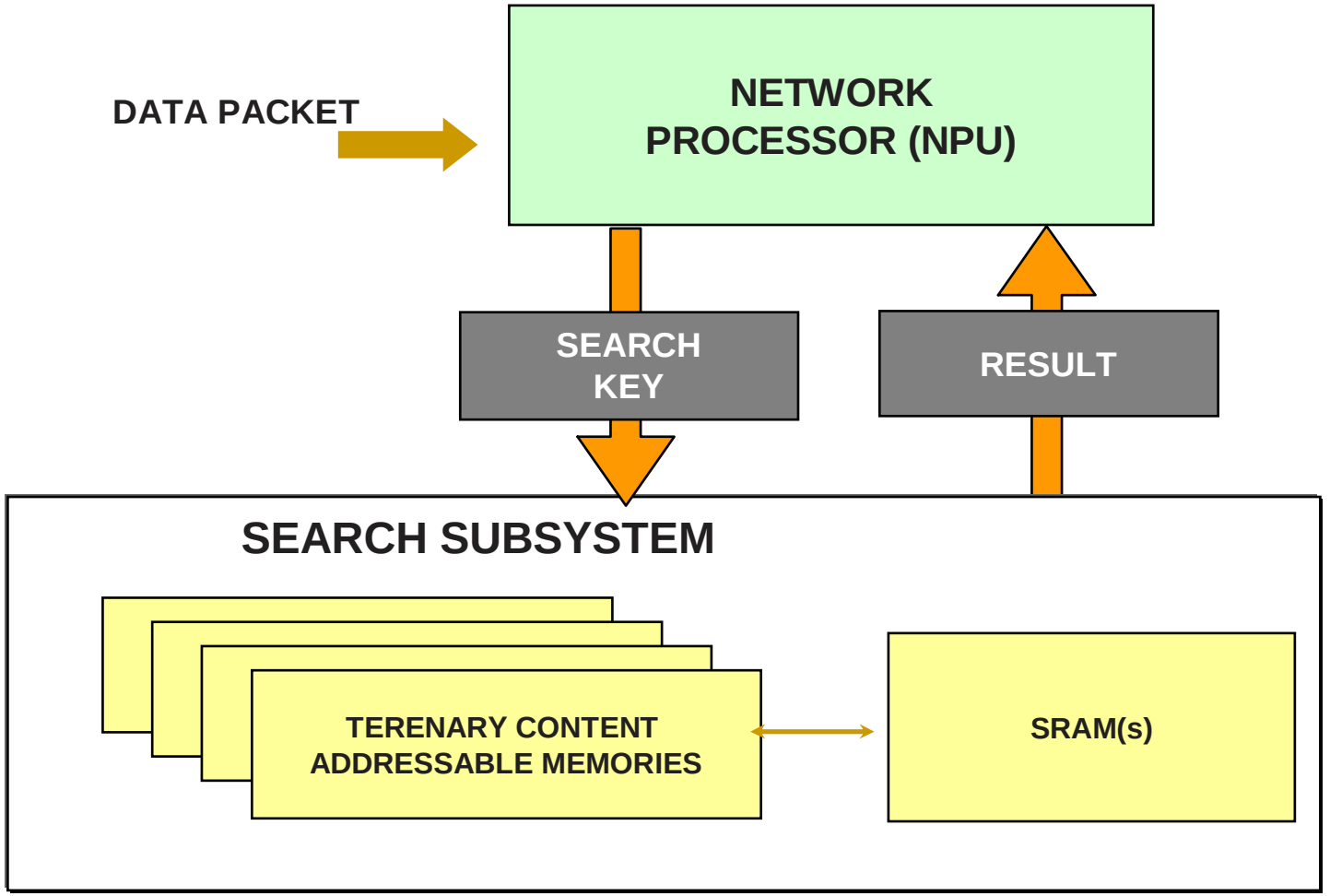
Optimization Algorithms

- Data layout techniques
  - Decomposition Storage Model [Copeland et al. 1985]
  - PAX Model [Ailamaki et al. 2001]
- Cache line extraction of relevant data [Shatdal et al. 1994]
- Compression of tuples [Manegold et al. 1999]
- Pre-fetching-based optimizations

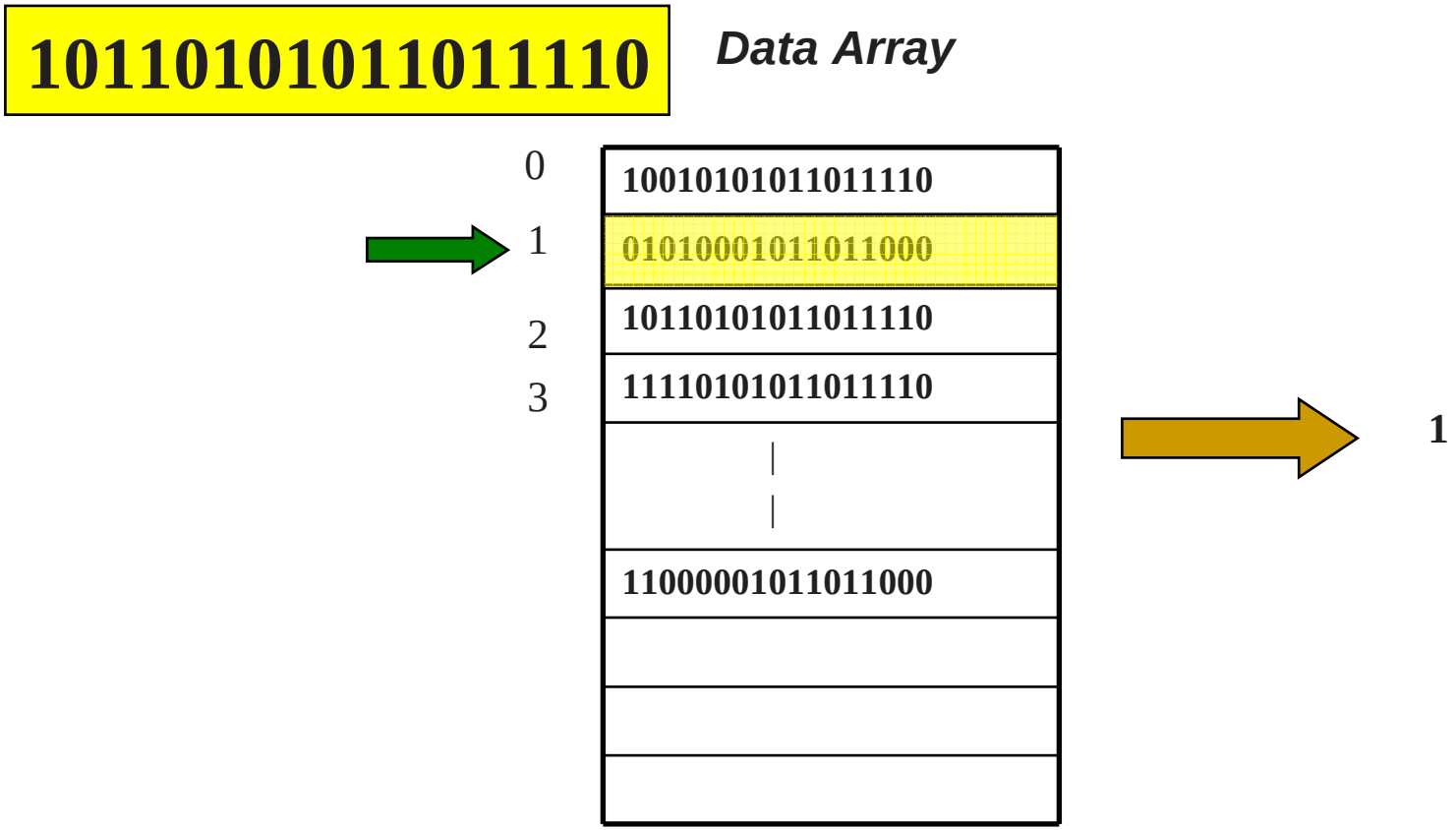
Hardware-based Approaches

- Graphics Processors
  - Hardware Acceleration of Spatial Database Operations [Sun et al. SIGMOD 2003]
  - Fast Computation of Database Operations using Graphics Processor [Govindaraju et al. SIGMOD 2004]
  - Hardware Acceleration in Commercial Databases : A Case Study of Spatial Operations [Bandi et al. VLDB 2004]
  - Fast and Approximate Stream Mining of Quantiles and Frequencies Using Graphics Processors [Govindaraju et al. SIGMOD 2005]

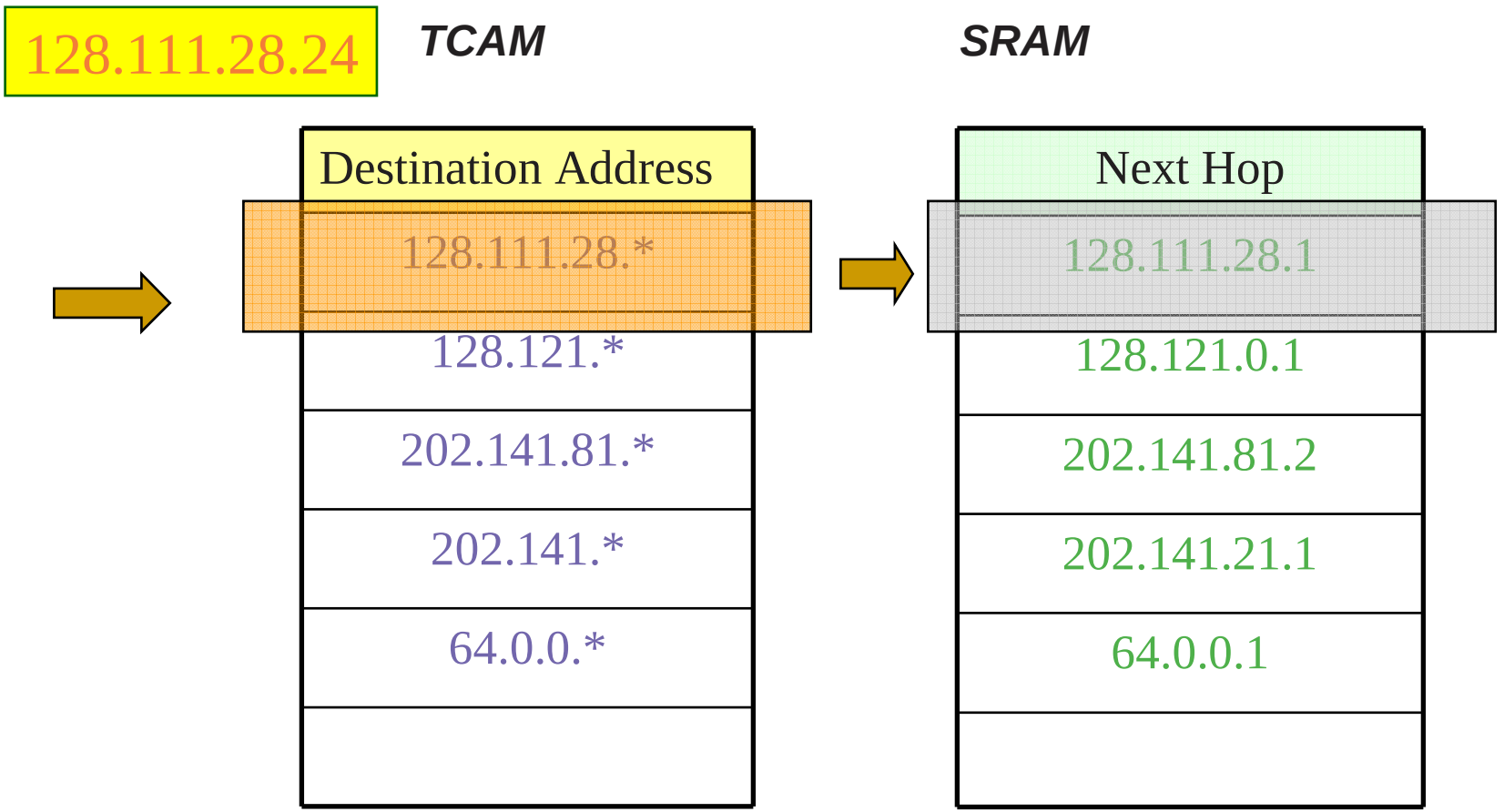
Networking Hardware : Content Addressable Memories



Ternary Content-Addressable Memory (Network Search Engine)



TCAM and Associated SRAM



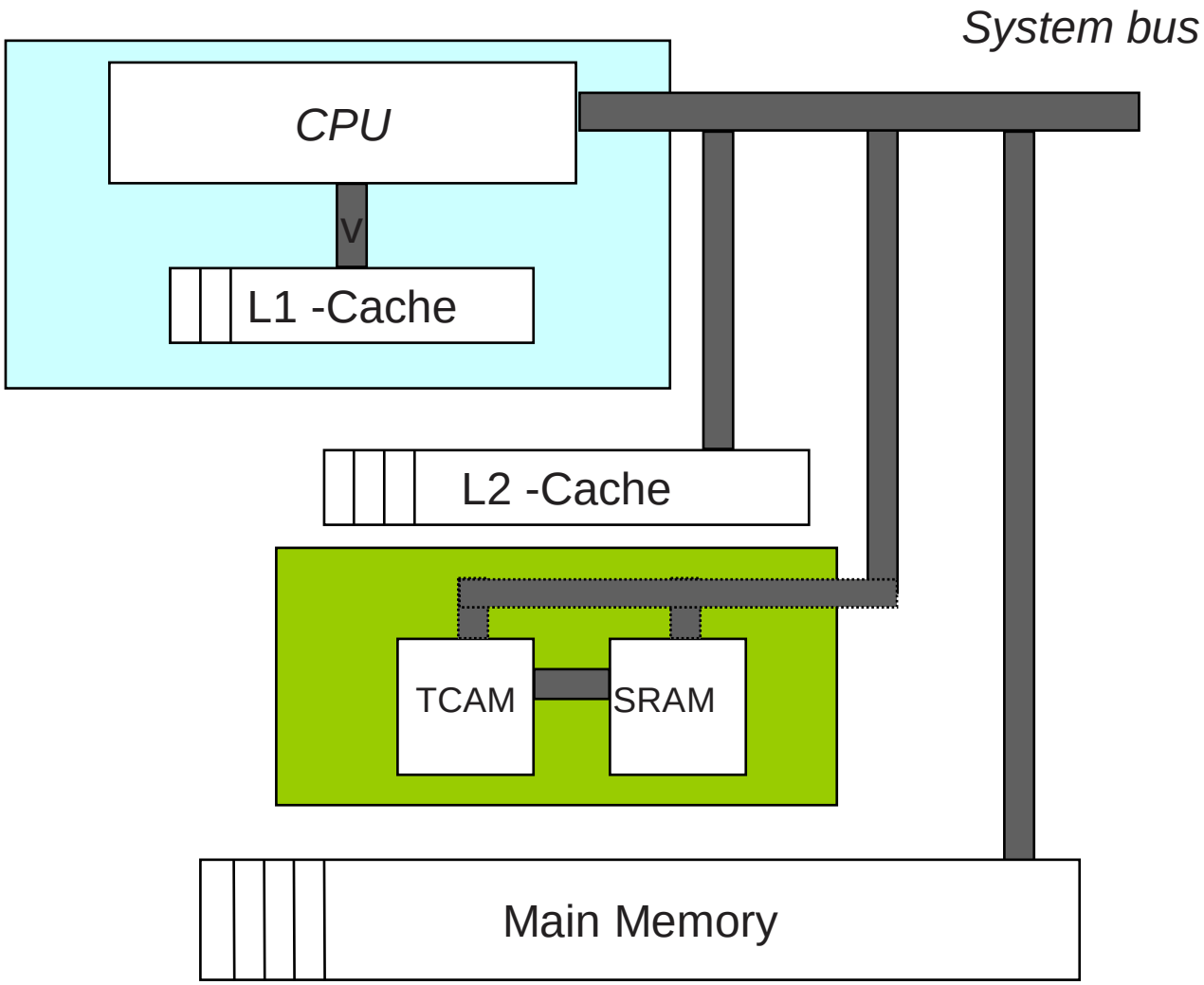
TCAMs and Applications

- Packet Classification
  - Fast and scalable layer four switching [**SIGCOMM98**]
- Intrusion Detection
  - Efficient Multi-Match Packet Classification with TCAM [**HOTI'04**]
- Pattern Matching
  - Gigabit Rate Packet Pattern-Matching Using TCAM [**ICNP'04**]

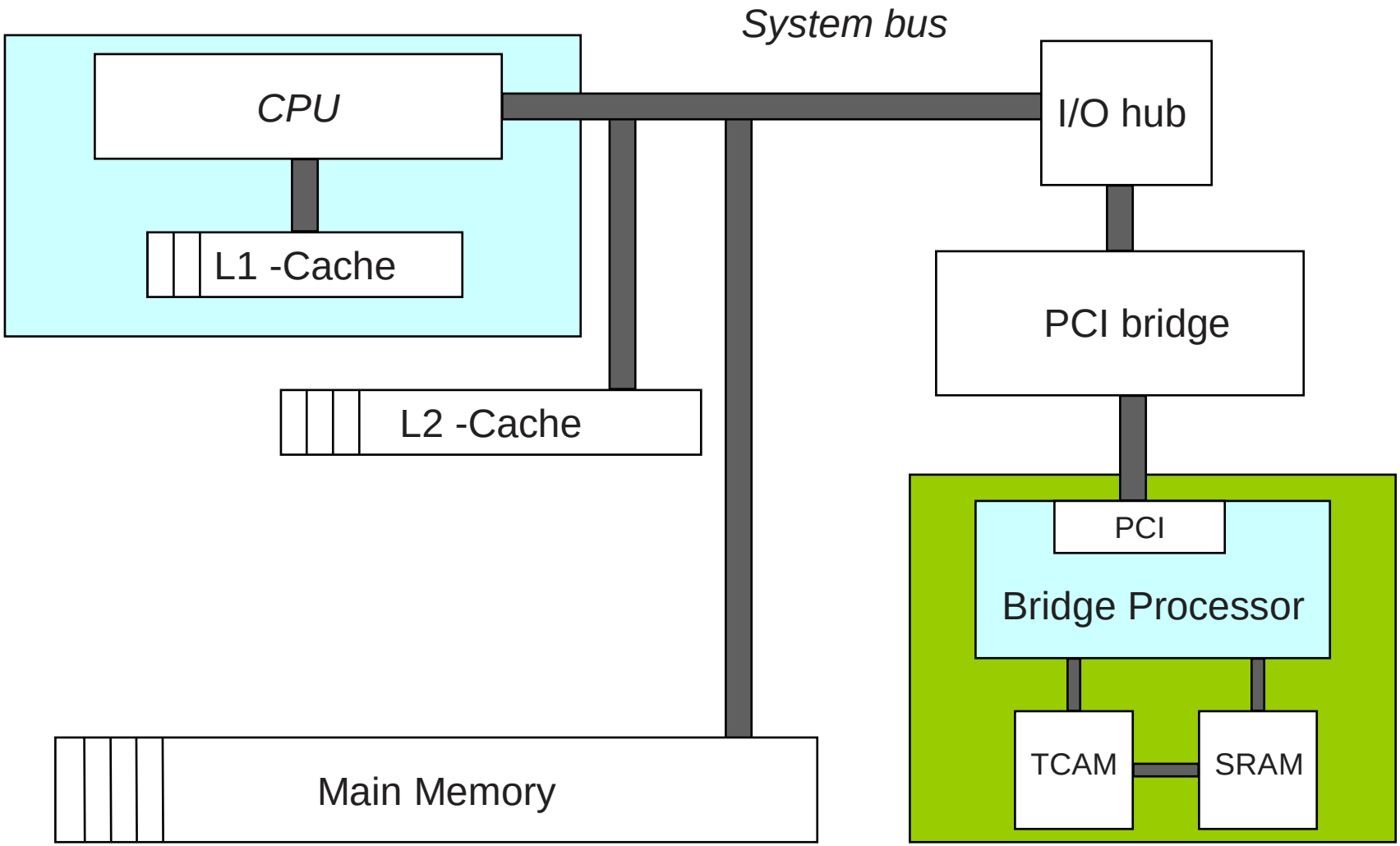
Databases and TCAMs

- TCAMs provide very fast data-centric searches
- Earlier works proposed using CAMs
  - A Data Management System Utilizing an Associative Memory [**DeFiore73**]
  - A content-addressed memory design for data base applications [**Kain76**]
- Why did not they work?
  - Hardware technology was in its infancy [**Grosspietsch92**]
    - Largest CAM sizes were typically hundred words
    - Very expensive to build
  - Scientific computing was the main driving force and not databases
- What changes it now?
  - Industry standard TCAMs (25600 entries (72 – bit) size and 133 Million searches per second)
  - Can be cascaded to provide storage of 18Million entries without losing any search throughput

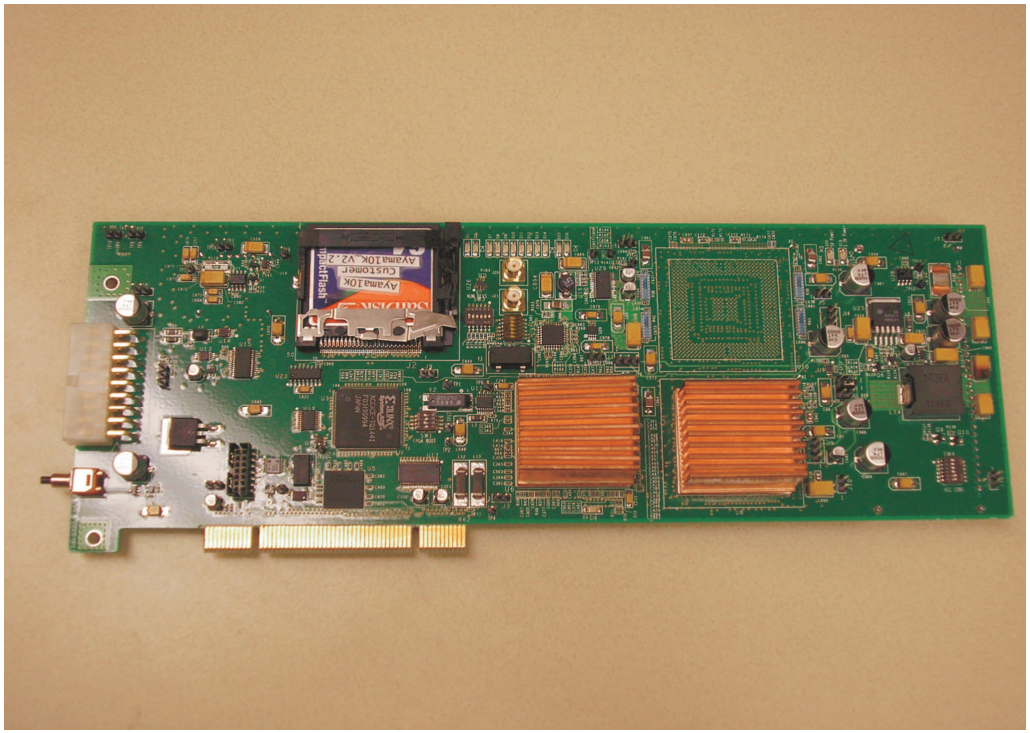
CAM-Cache : Ideal Architecture



CAM-Cache : Prototype architecture



CAM-Cache prototype

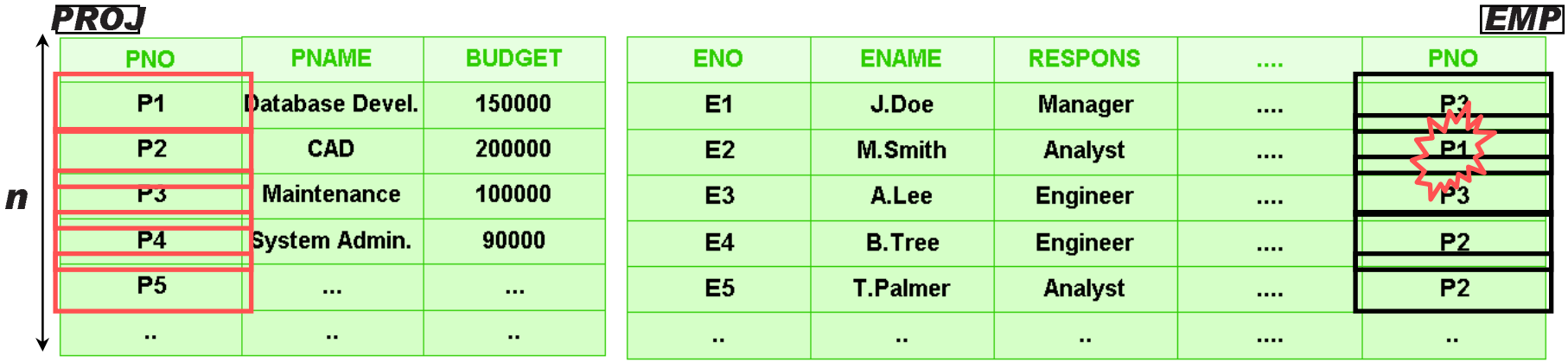


Courtesy : Cypress Semiconductor Inc.

CAM-Cache Prototype

- Communicates with the CPU over PCI bus
  - 32-bit 33MHz
- Bridge processor
  - FPGA Implementation
- One 256K TCAM from Cypress Semiconductor Inc.
  - 133 million searches per second capability
- Current prototype's search through-put
  - 33000 searches per second
  - PCI communication is the bottleneck

Software-based Nested Loop Join



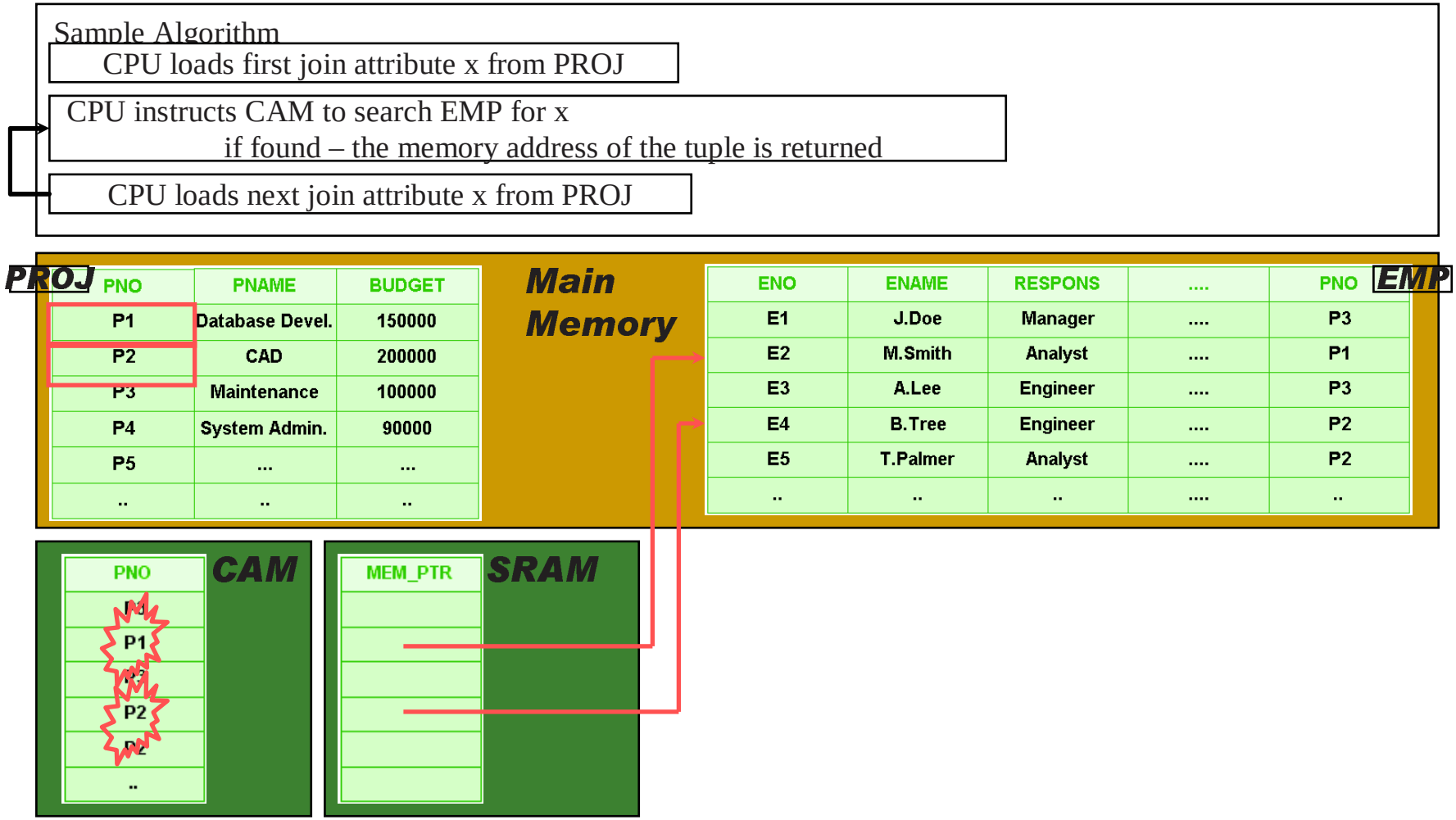
PNO	PNAME	BUDGET
P1	Database Devel.	150000
P2	CAD	200000
P3	Maintenance	100000
P4	System Admin.	90000
P5	...	...
..	..	..

ENO	ENAME	RESPONS	....	PNO
E1	J.Doe	Manager	....	P3
E2	M.Smith	Analyst	....	P1
E3	A.Lee	Engineer	....	P3
E4	B.Tree	Engineer	....	P2
E5	T.Palmer	Analyst	....	P2
..	..	..	....	..

PNO	ENAME
P1	M.Smith
P2	B.Tree
P2	T.Palmer
P3	J.Doe
P3	A.Lee
..	..

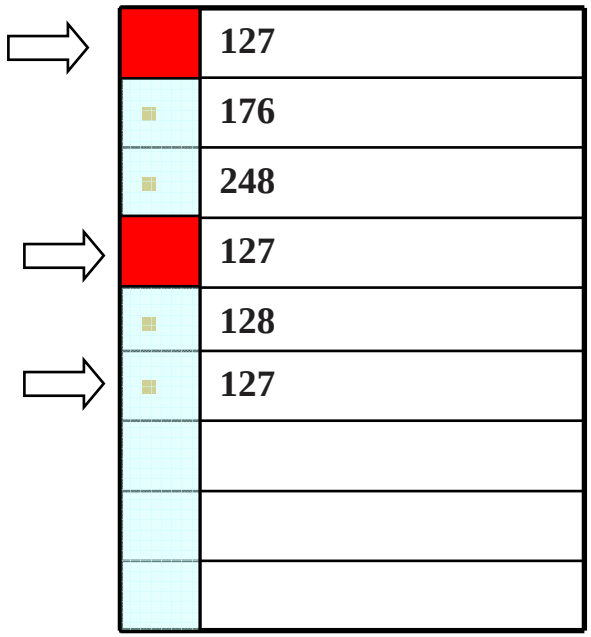
FOR each tuple p in PROJ DO
    FOR each tuple e in EMP DO
        IF p and e join to make a tuple t
        THEN output t;

CAM-based equi-join (CJ)



Design Challenges for equi-join

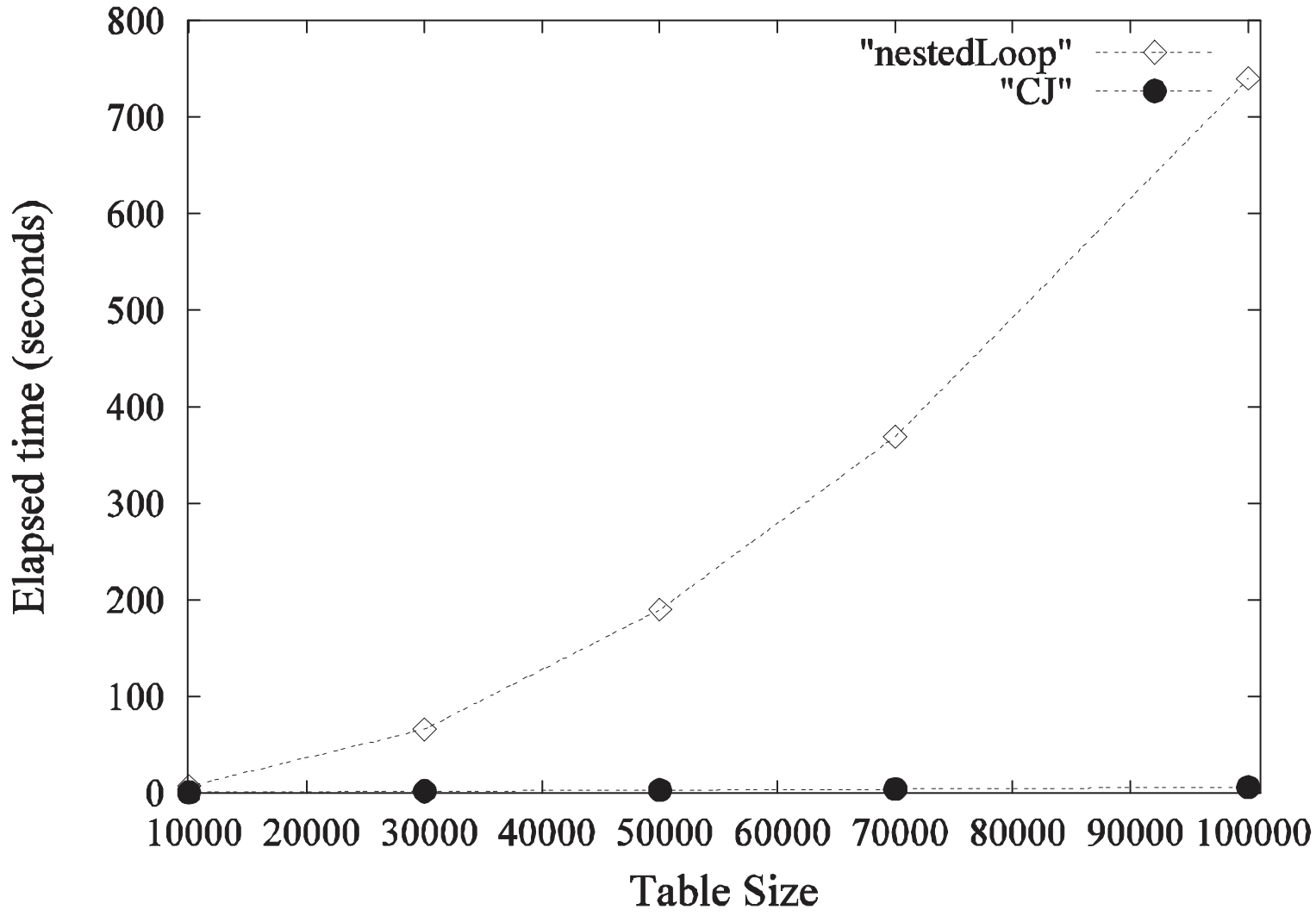
- Multiple matches not supported
  - Joins can have a tuple matching multiple tuples
- Multi-hit flag
  - For each match null the valid bit
  - This return the next top result



Experimental Setup

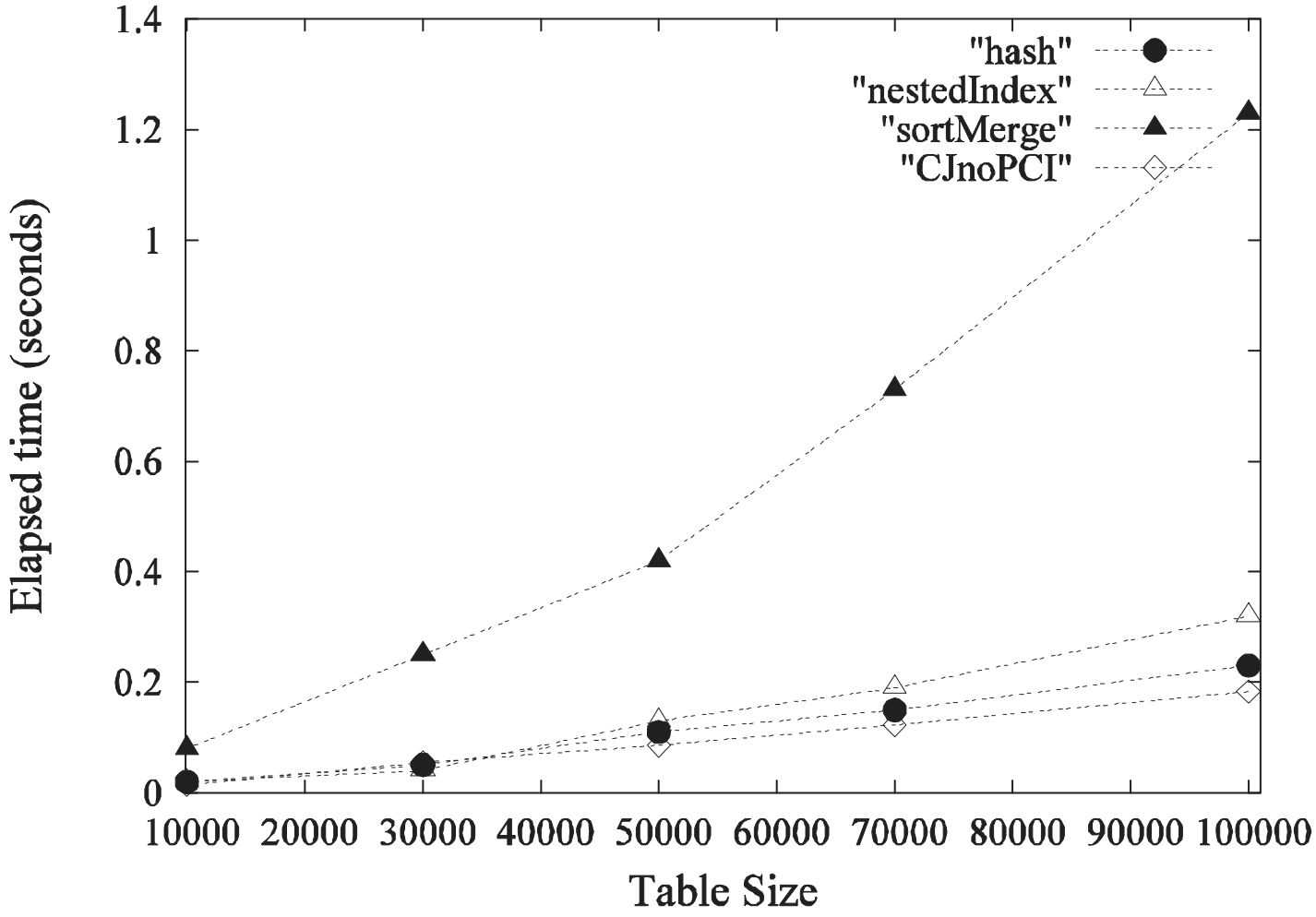
- Integration into a commercial database
  - Allows comparison with highly optimized techniques
- Comparison
  - CAM-Cache based join
  - Nested Loop Join, Hash Join, Nested Indexed Join, Sort-Merge Join
- Dataset
  - Uniformly distributed numerical dataset
- Results Shown
  - **CJ** (PCI overhead inclusive)
  - **CJnoPCI** (PCI overhead excluded)

Experimental evaluation : Nested loop join



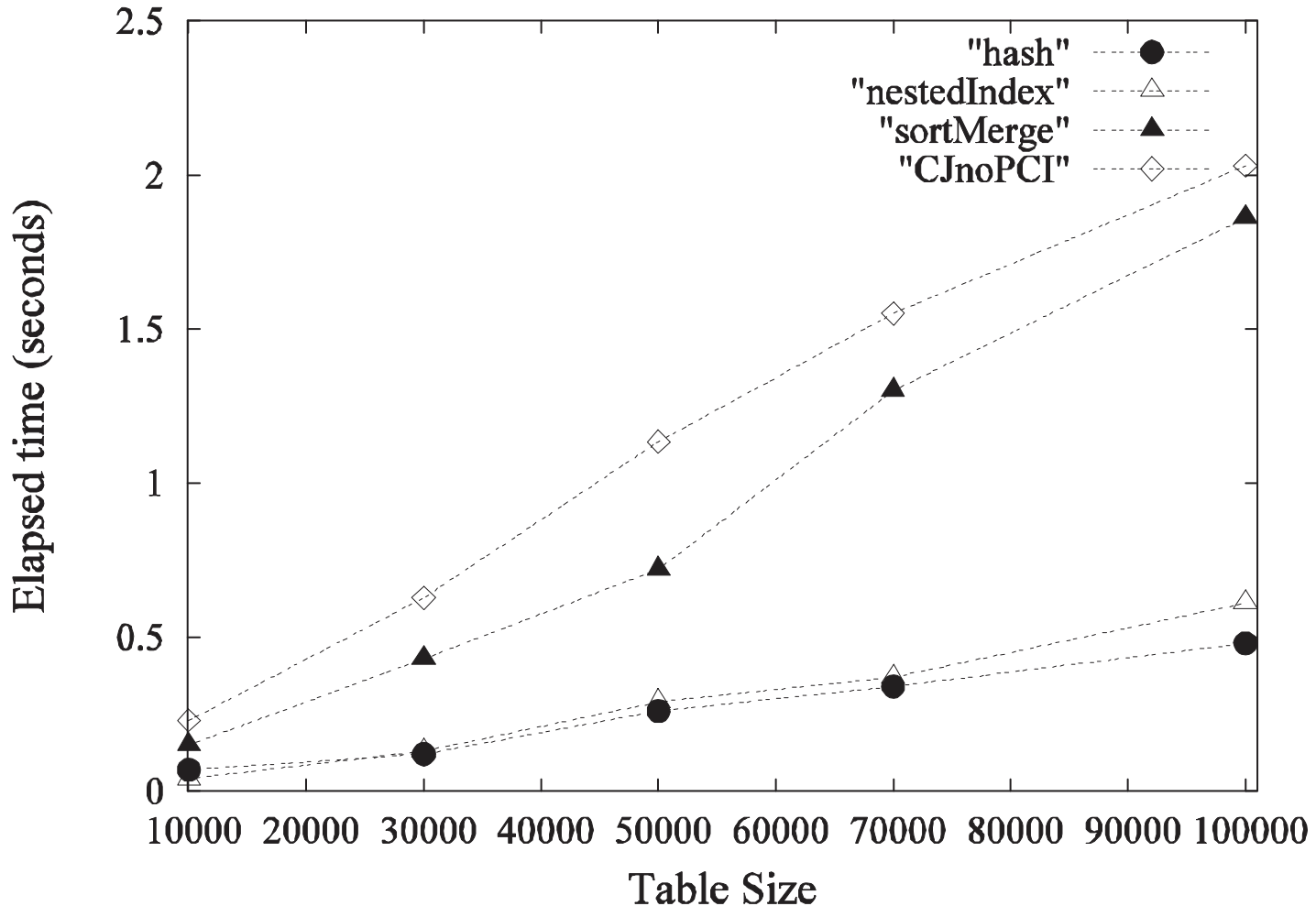
Join Factor = 1

Comparison with efficient techniques



Join Factor = 1

Comparison with efficient techniques



Join Factor = 12

Concluding Remarks

- Prototype improvement
  - PCI-X bus
  - Driver optimization
- Other database applications
  - Fast O(n) complexity sorting algorithms
  - Fast solution for general conditional joins
    - Supports multi-matches without any write overhead
  - String Matching.