Challenges for data intensive services by recent hardware developments

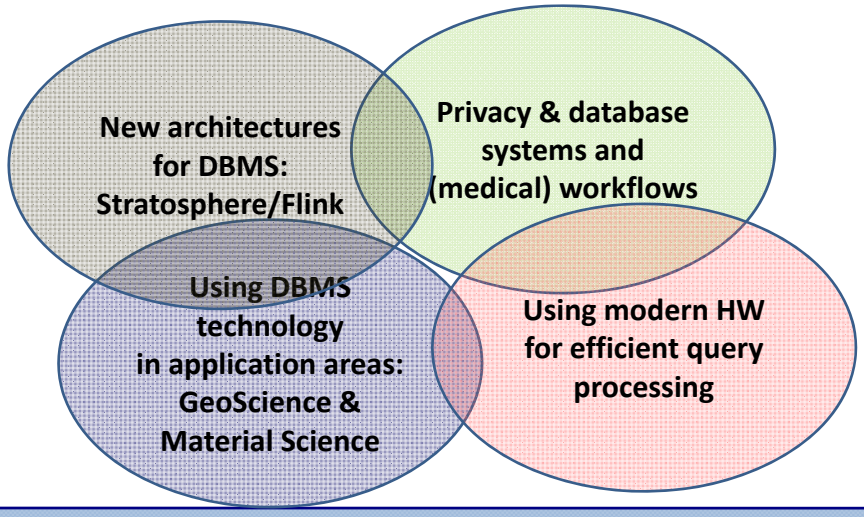
Prof. Johann Christoph Freytag, Ph.D.
Institut für Informatik, Humboldt-Universität zu Berlin

Mainly based on PhD thesis by Steffen Zeuch (April 2018)

Talk @ SummerSoc, Crete, June 2018

Humboldt- Universität zu Berlin

Current Research Topics



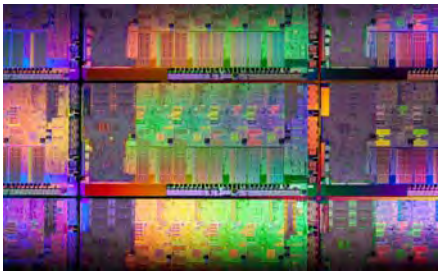
… and now back to the talk

The changes in the world of HW

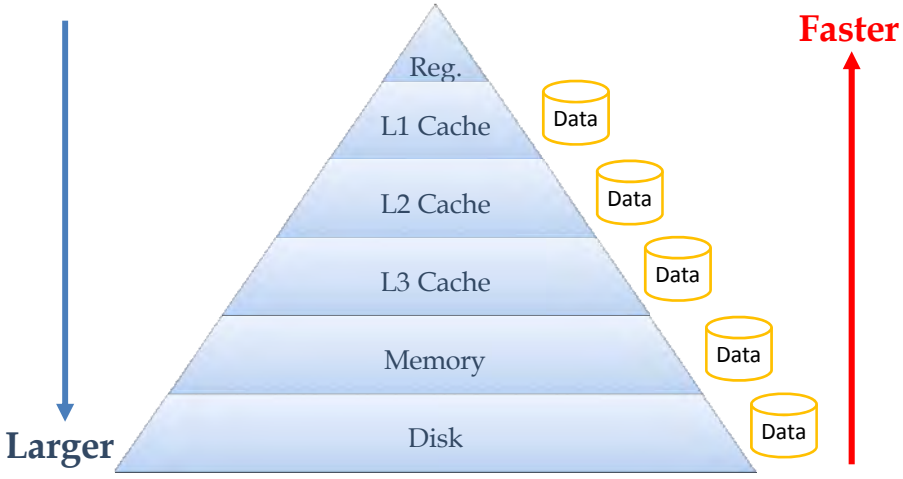
During the last 15 to 20 years….

Modern CPUs – Multi-Core & more

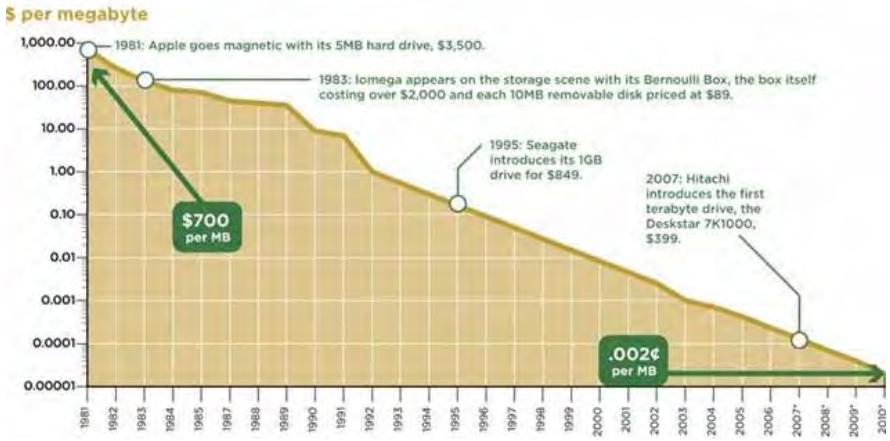
- Features: pipelining, superscalar, branch prediction, prefetching
- Multi-core and simultaneous multi-threading (SMT, hyper-threading)
- SIMD vector instructions (MMX/SSE/AVX)
- Multi-Level cache hierarchy



Memory Hierarchy



Changes in memory cost



Quelle: www.deepspar.com/images/Storage_Cost.jpg

Changing technology: Flash disk

- Characteristics
  - Today: 1 TB - Cost about $300 (Samsung)
  - !! Less power consumption !!

Device	Sequential	Random 8KB	Price $	Power	iops/$	iops/watt
SCSI 15k rpm	75 MBps	200 iops	500$	15 watt	0.5	13
SATA 10k rpm	60 MBps	100 iops	150$	8 watt	0.7	12
Flash- read	53 MBps	2,800 iops	400$	0.9 watt	7.0	3,100
Flash - write	36 MBps	27 iops	400$	0.9 watt	0.07	30

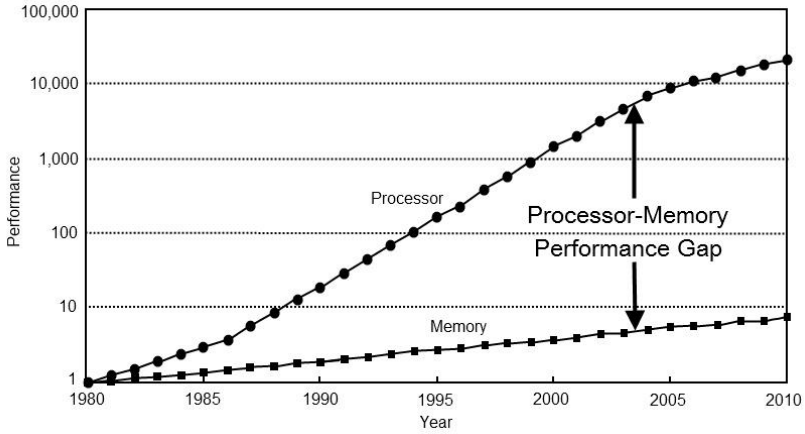
Gray, J., & Fitzgerald, B. (2007), FLASH Disk Opportunity for Server-Applications, from http://research.microsoft.com/~Gray/papers/FlashDiskPublic.doc; Jan 2007; Retrieved March 8, 2007

Changing technology: CPU farms, Clusters & BigData Centers

- Compute Container
  - > 1200 CPUs
  - > 22000 cores
  - > 5.4 TB Main memory
  - > 7.0 PBytes Disk storage
  - Only Need power & Internet access & water
- Big Data Center (Google & others)

Quelle: http://technoblimp.com/2013/06/24/where-the-internet-lives-a-look-inside-googles-data-centers/

Challenge: The "Memory Wall"

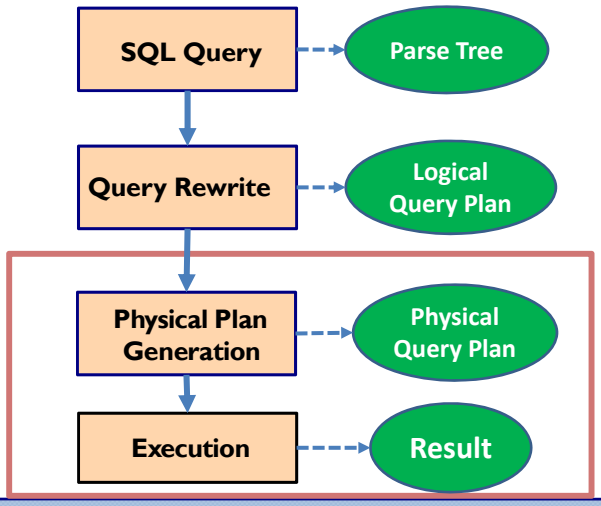


https://www.extremetech.com/computing/261792-what-is-speculative-execution

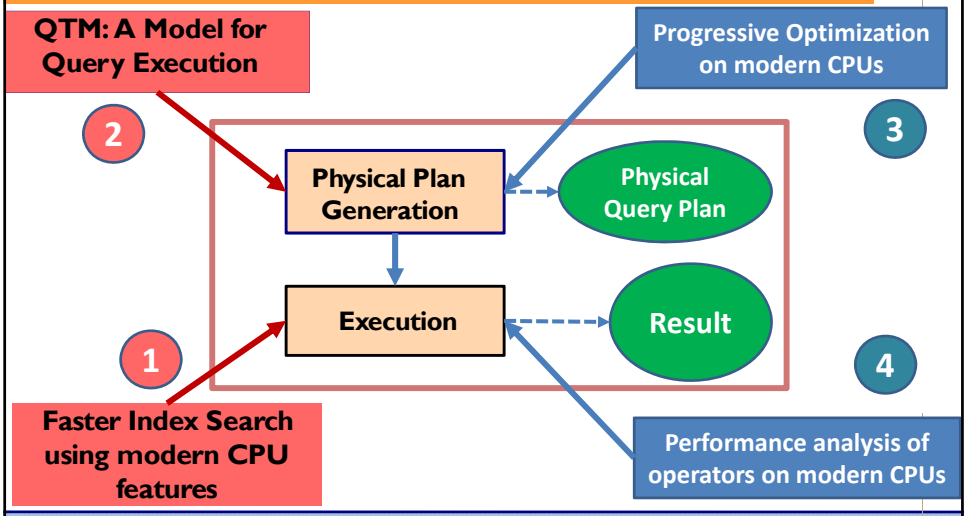
What does it mean for DBMS

- The world of HW has changed
- The world of data management has changed
  - MapReduce Systems
    - 1st generation: Hadoop, Google proprietary, ….
    - 2nd generation: Apache Flink (Berlin), Apache Spark (Berkeley)
- Relational DBMS technology ***has not*** changed
  - Architecture is 40+ years old
  - Needs rethinking with new HW/OS opportunities
  - Architecture & HW impact
    - From static (2 phase) to dynamic (decide during execution)
    - Include new HW (GPUs, FPGAs, Cluster, …)

Database Query Execution



Focus of our work so far

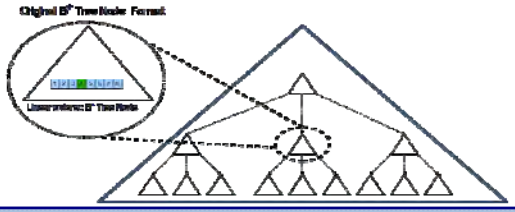


Part 1:
Faster index search using modern CPU features

Motivation

- **B^+-Tree:** commonly used index structure
- Finding a key in $O(\log_b n)$
- Common node-internal search algorithm:
  - Binary search in $O(\log_2 n)$

Can we do better?
Yes with SIMD!



Outline Part 1

a. Binary Search & SIMD

b. K-ary Search

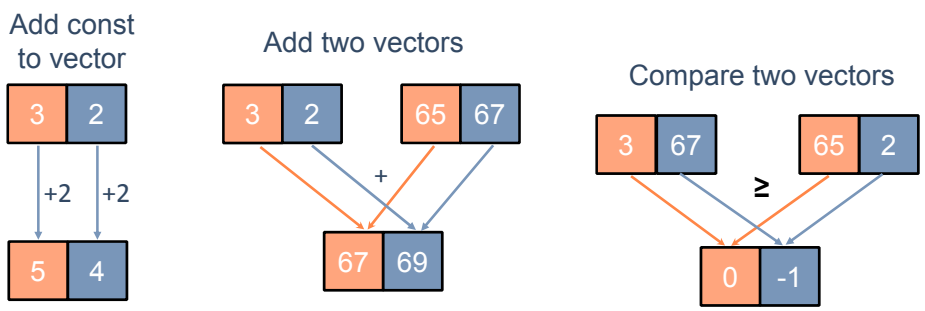
c. Segmented Tree

d. Segmented Trie

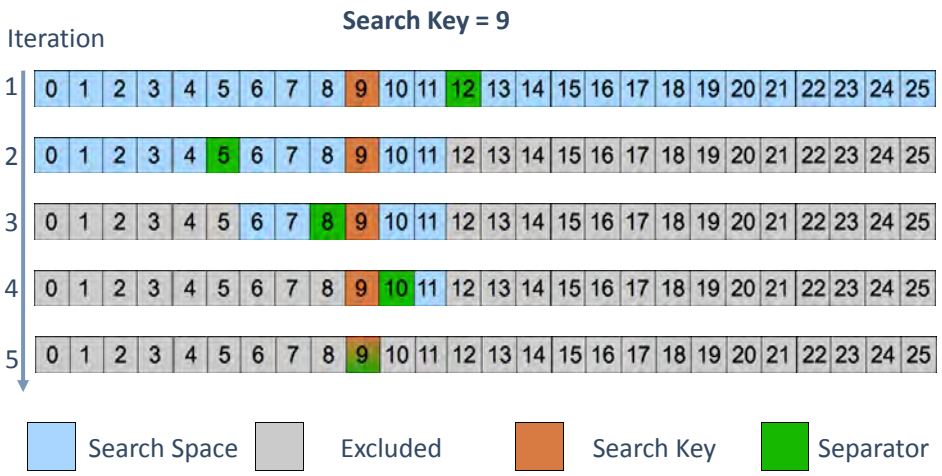
e. Evaluation & Contribution

SIMD

- Single Instruction Multiple Data:
  - Available on CPU and GPU
  - Arithmetical, comparison, conversion, logical



Binary Search



Outline

a. Background ✓
b. K-ary Search
c. Segmented Tree
d. Segmented Trie
e. Evaluation
f. Conclusion Part 1

Binary Search - two Separator

Search Key = 9

Iteration

1 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

2 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

3 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Search Space | Excluded | Search Key | Separator

Binary Search + SIMD

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

SIMD Register A: 8 17

SIMD Register B: 9 9

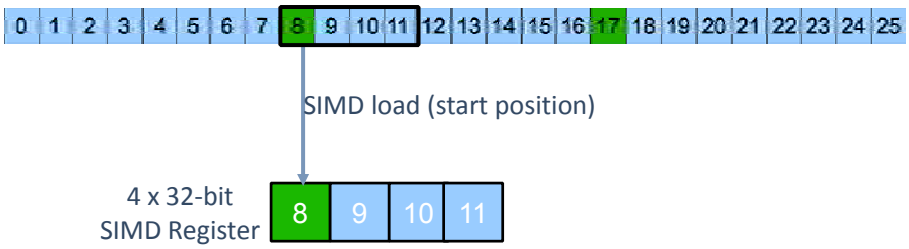
>=

SIMD Register C: 0 -1

Search Space | Excluded | Search Key | Separator

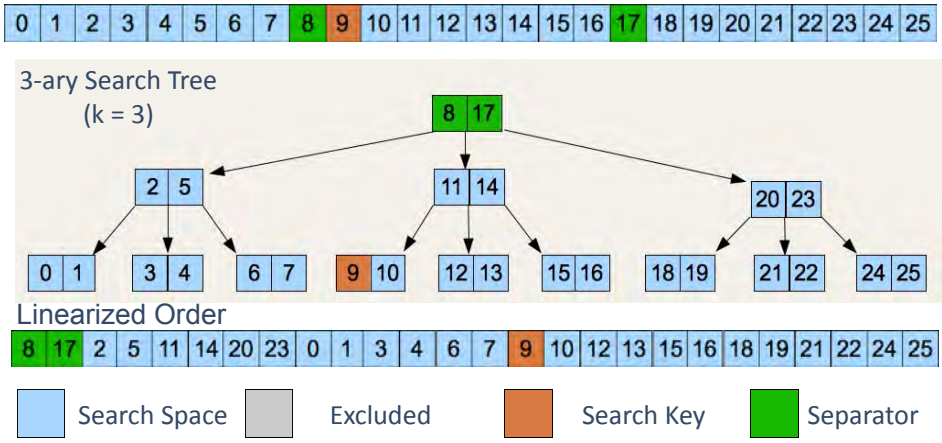
Problem: SIMD on CPU

SIMD on CPU does not support scatter and gather functionality.

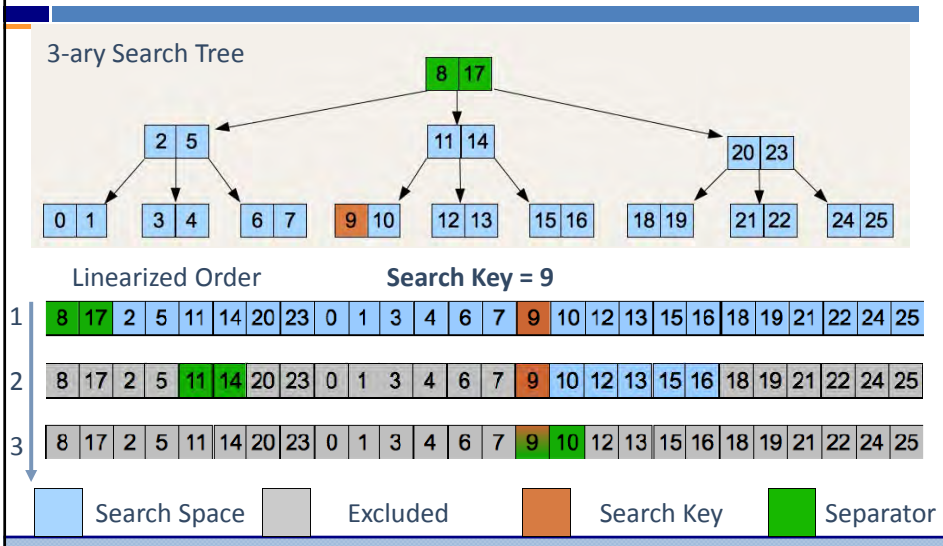


Solution: K-ary Search by Schlegel et al.

Search Key = 9



Applied K-ary Search



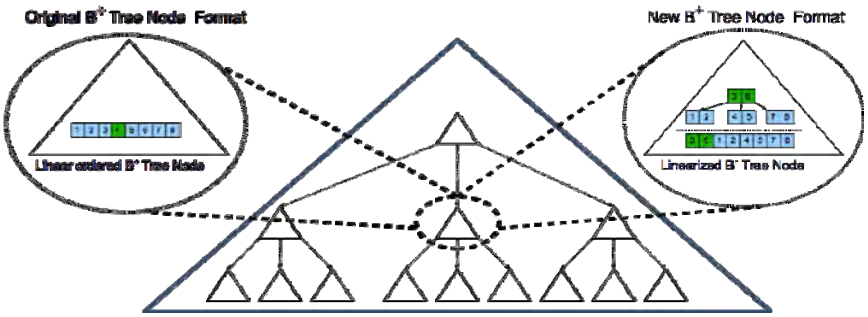
Degree of Parallelism

SIMD Bandwidth	Search Method	Data Type	Parallel Comparisons
128-bit	K-ary Search	8-bit	16
		16-bit	8
		32-bit	4
		64-bit	2
	Binary Search	All	1

Outline Part 1

a. Binary Search & SIMD ✓
b. K-ary Search ✓
c. Segmented Tree
d. Segmented Trie
e. Evaluation
f. Conclusion Part 1

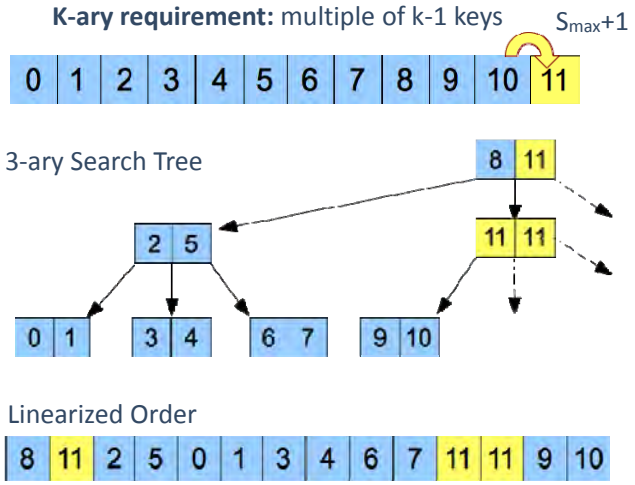
Segmented Tree



Change inner-node search algorithm from commonly binary search to k-ary search.

Problem: Unfilled Nodes

K-ary requirement: multiple of k-1 keys



Insert implies Reordering

- Inserting a new key requires a reordering if it is inserted between two existing keys:
  - Sorting → Inserting → Linearizing
- **Not necessary if:**
  - Empty Node
  - Key is greater than the largest existing key

Segmented Tree

Advantages:

- High resource utilization
- Less iterations required
  - Binary Search: $\log_2 n$ vs. k-ary Search $\log_k n$

Disadvantages:

- Reordering overhead
- Large data types decrease performance

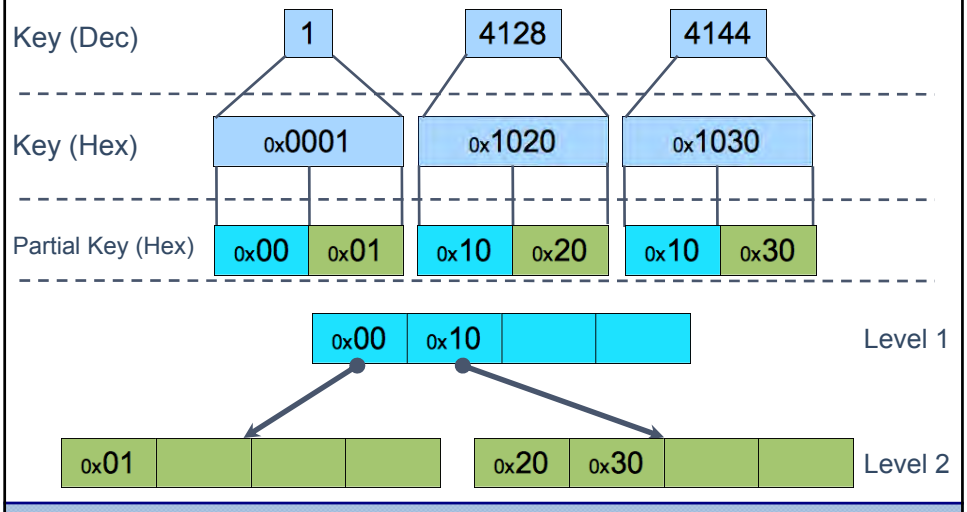
Open/Challenge:

- Updates

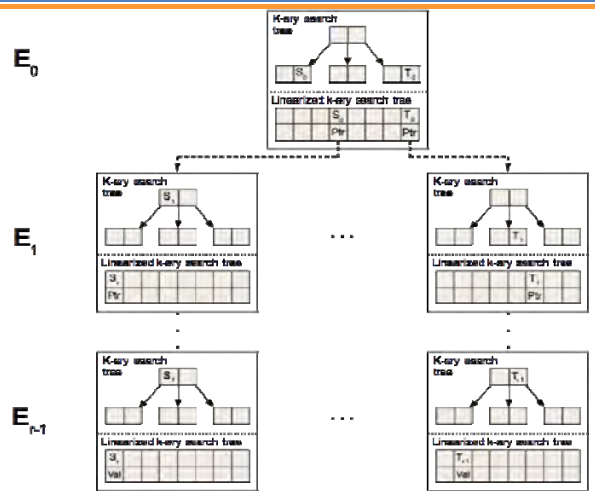
Outline Part 1

a. Binary Search & SIMD ✓
b. K-ary Search ✓
c. Segmented Tree ✓
d. Segmented Trie
e. Evaluation
f. Conclusion Part 1

Segmented Trie (SegTrie)



Segmented Trie (SegTrie)



Segmented Trie (SegTrie)

Advantages:

- High SIMD search performance
- Prefix compression
- Early termination

Disadvantages:

- Fix level count
- Reordering overhead

Upen/Challenge:

- Updates

SegmentedTree vs. Segmented Trie

	SegTree	SegTrie
Derived From	B^+-Tree	Prefix B-Tree
Number of Iterations	Tree Height	Max. #Level (Early termination)
Number of Level	Dynamic	Static (Pre-defined)
DOP	Depends on Data Type	16 (8-bit)

Outline Part 1

a. Binary Search & SIMD ✓
b. K-ary Search ✓
c. Segmented Tree ✓
d. Segmented Trie ✓
e. Evaluation
f. Conclusion Part 1

Test Setup

HW/SW Configuration:

- **CPU:** Intel Xeon 5520, 4 x 2,26 GHz
- **L1:** 32KB, **L2:** 256 KB, **L3:** 8 MB, **MM:** 8 GB
- **Cacheline:** 128 Byte, **SIMD bandwidth:** 128 Bit
- Windows 7 64-bit Professional

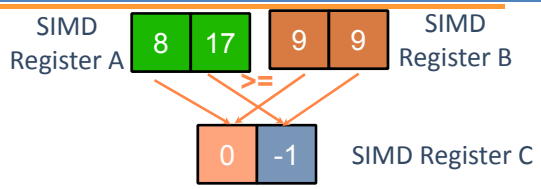
Test Dataset:

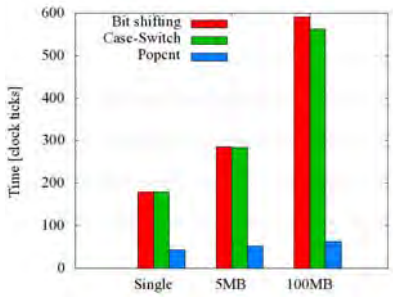
- Synthetically generated, ascending, starting at 0

Evaluation: Bitmask

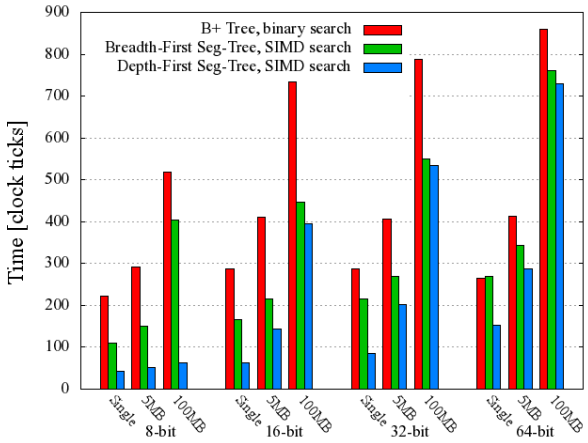
Three Algorithms:

1. Bit Shifting
2. Case-Switch
3. PopCnt





Evaluation: SegTree



Single = 4k page

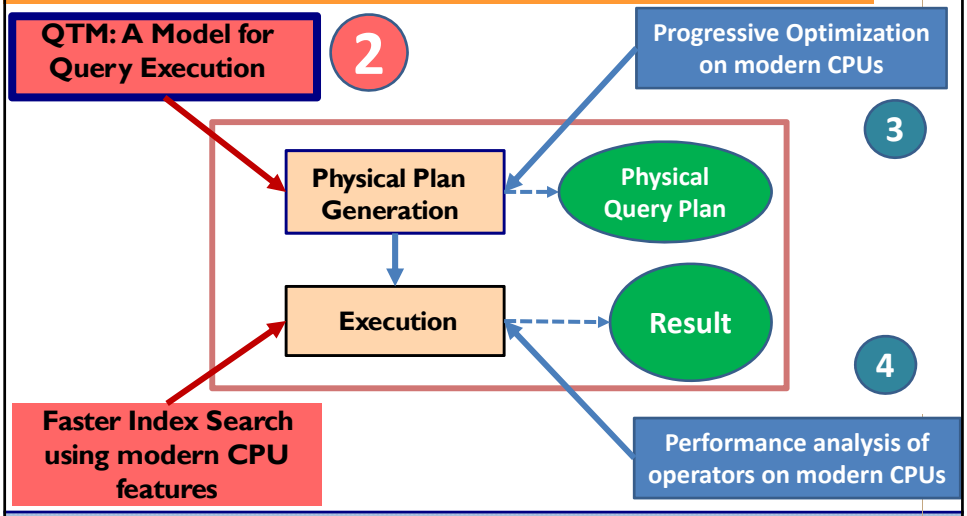
Outline Part 1

a. Binary Search & SIMD ✓
b. K-ary Search ✓
c. Segmented Tree ✓
d. Segmented Trie ✓
e. Evaluation ✓
f. Conclusion Part 1

Our Contributions – Part 1

- Using SIMD
  - B+-Tree -> Segmented Tree
  - Prefix B-Tree -> Segmented Trie
- Transformation and search algorithm using breadth-first and depth-first data layout
- Three algorithms for interpreting SIMD comparison results
- Generalization for an arbitrary key count ("Filling up")

Part 2: QTM: A Model for Query Execution



Motivation

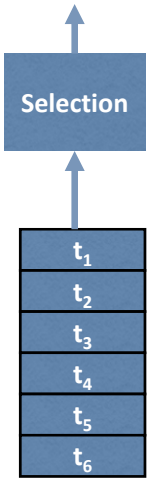
- **Different DBMS execute the same QEP using different schedules**
  - Run-time execution not query optimization
  - No uniform scheduling format
  - Query execution in different DBMS are not comparable
- **Major differences between DBMS:**
  - **Chunk Size:** Size of operator's input
  - **Scheduling Strategy:** Execution model vs. run-time scheduler

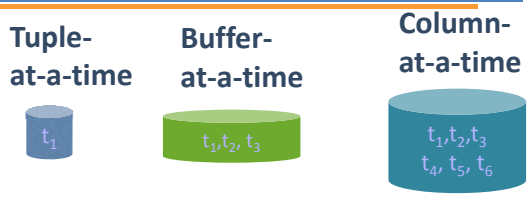
How to make different schedules comparable to explain why one schedule performs better than another?

Outline Part 2

a. Parallel Query Execution
b. QTM: Query Task Model
c. Evaluation
d. Outlook

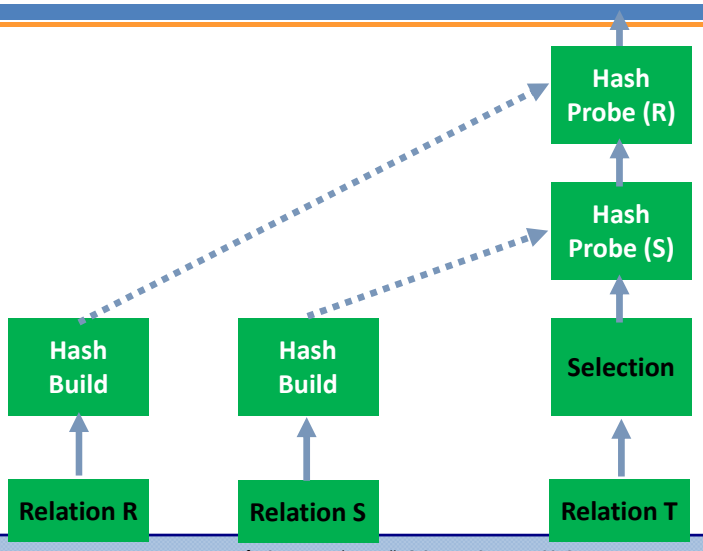
Chunk Size



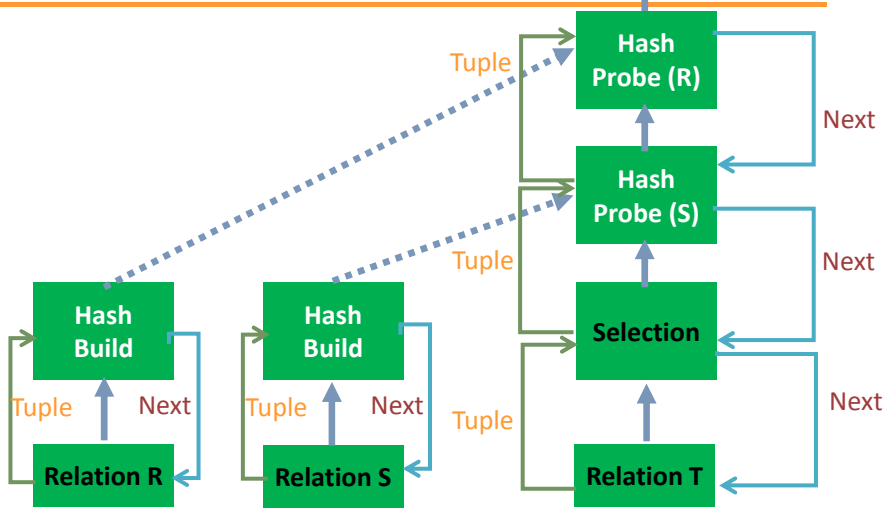


Chunk Size	DBMS
1 Tuple	System R, MySQL, (PostgreSQL)
"Fit into Cache"	Monet X100, DB2 with BLU
Fix number of tuples	Hyper
Fixed Block Size	C-Store
Column	MonetDB MIL

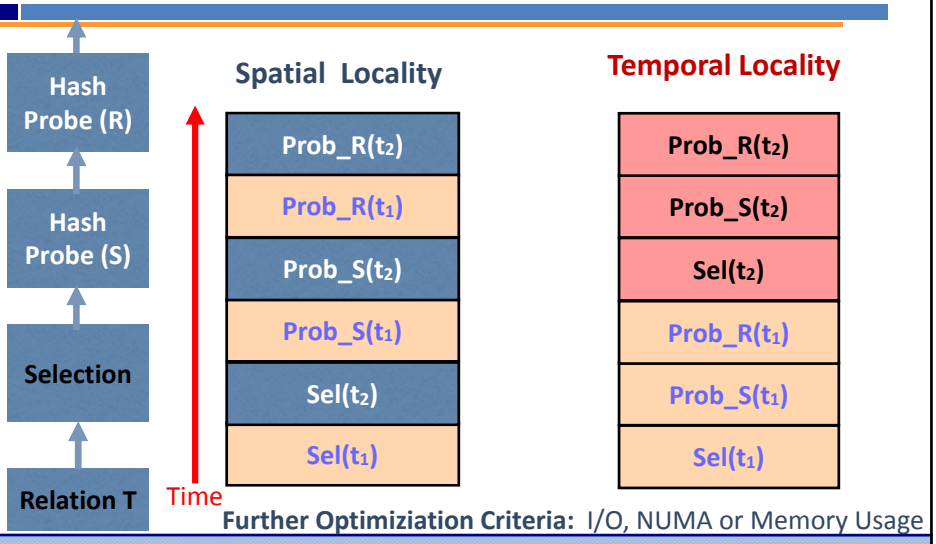
Scheduling Strategie



Volcano Execution Model (Open-Next-Close Iterator)

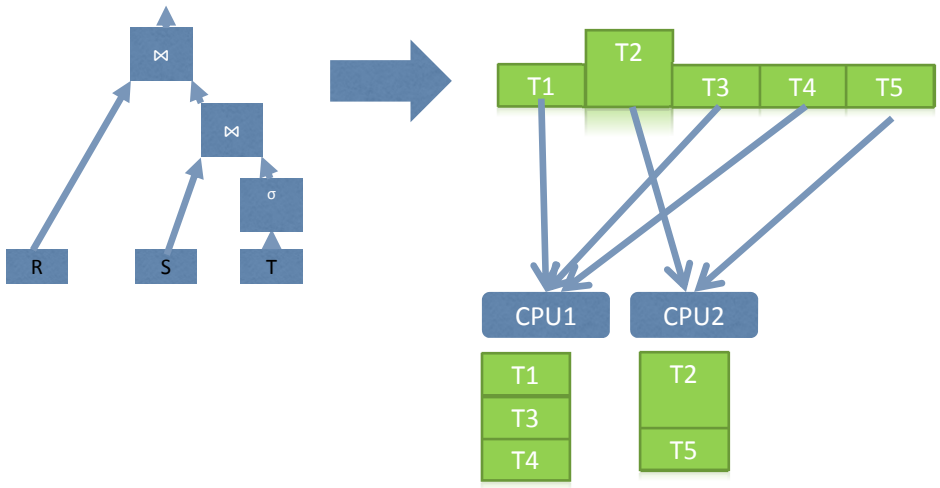


(Run-time) Scheduler

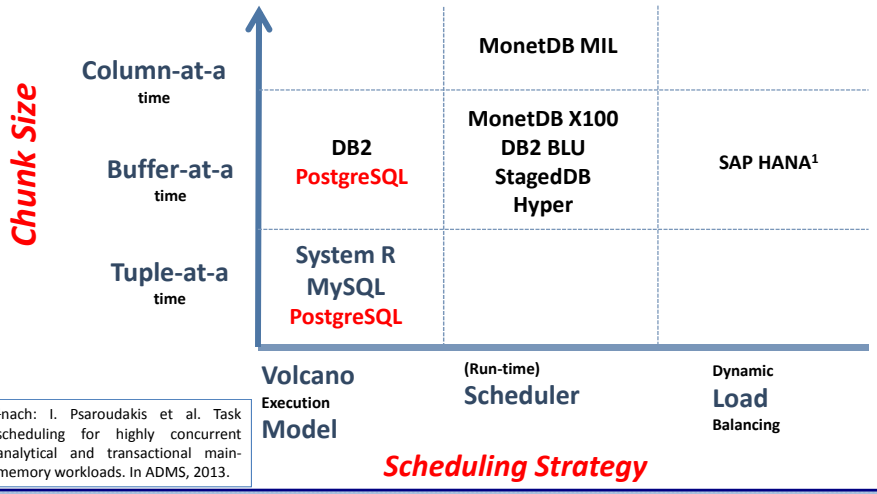


Further Optimiziation Criteria: I/O, NUMA or Memory Usage

Dynamic Load Balancing



DBMS Landscape



[1]nach: I. Psaroudakis et al. Task scheduling for highly concurrent analytical and transactional main-memory workloads. In ADMS, 2013.

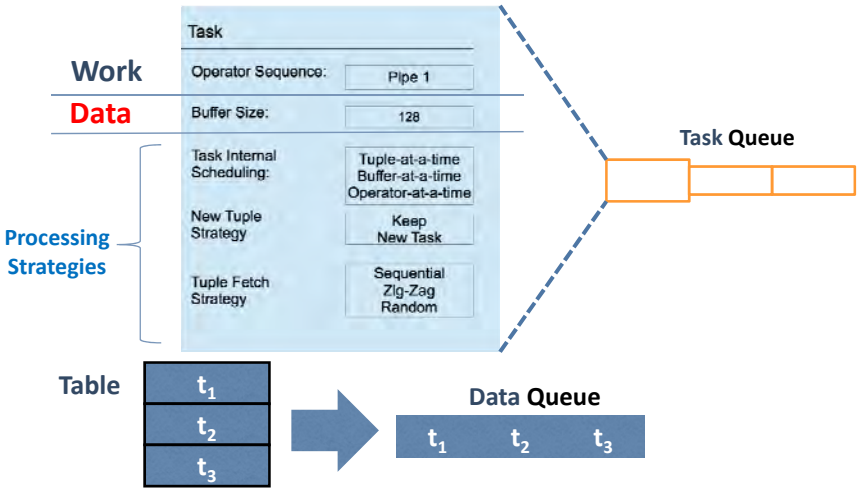
Outline Part 2

a. Parallel Query Execution ✓
b. QTM: Query Task Model
c. Evaluation
d. Outlook

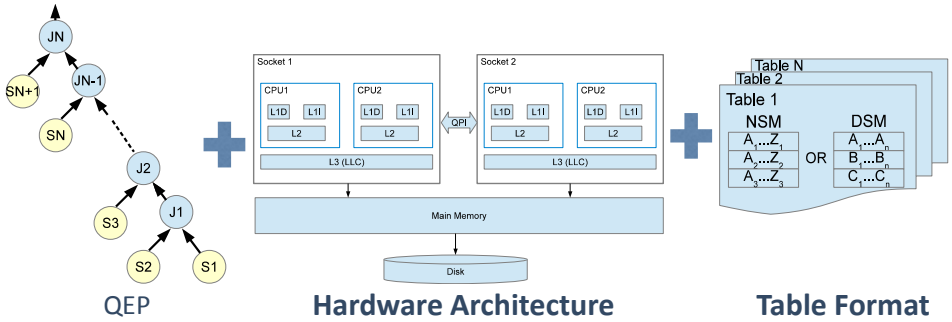
QTM: Query Task Model

- **Idea: A model that describes parallel query execution with tasks**
  - **QEP →** Queue of tasks
  - **Task:** Encapsulates a *piece of work on some data*
- **Goal:**
  - Open a design space for DBMS schedules
  - Make main aspects of query scheduling comparable:
    Execution order, degree of parallelism and thread coordination, and partitioning

Query Task Model



QTM Transformation: Input



QEP

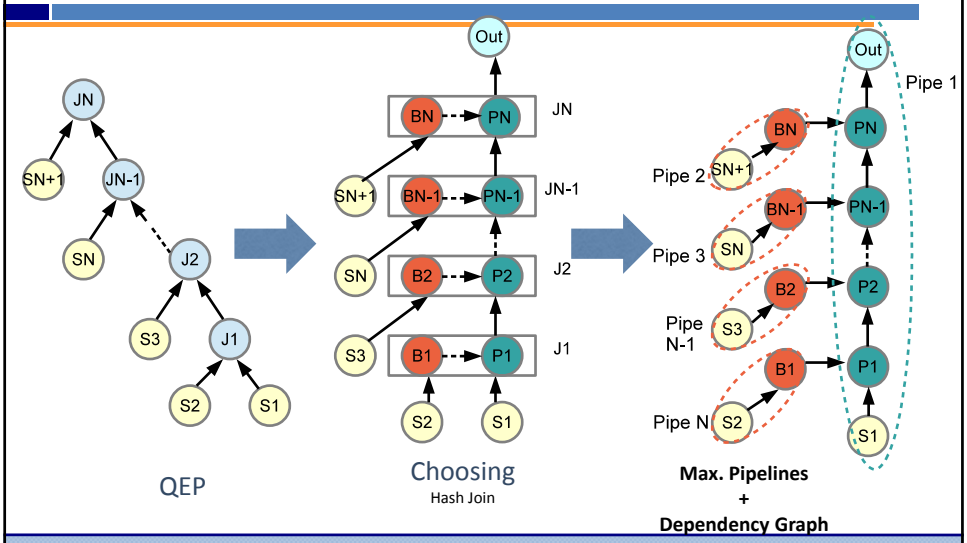
Hardware Architecture

Table Format

NSM = "row oriented"

DSM = "column oriented"

Step 1: QTM Transformation

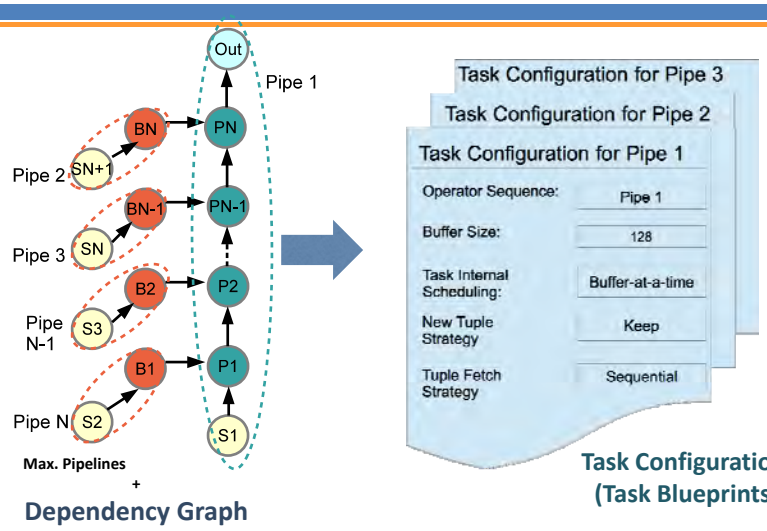


QEP

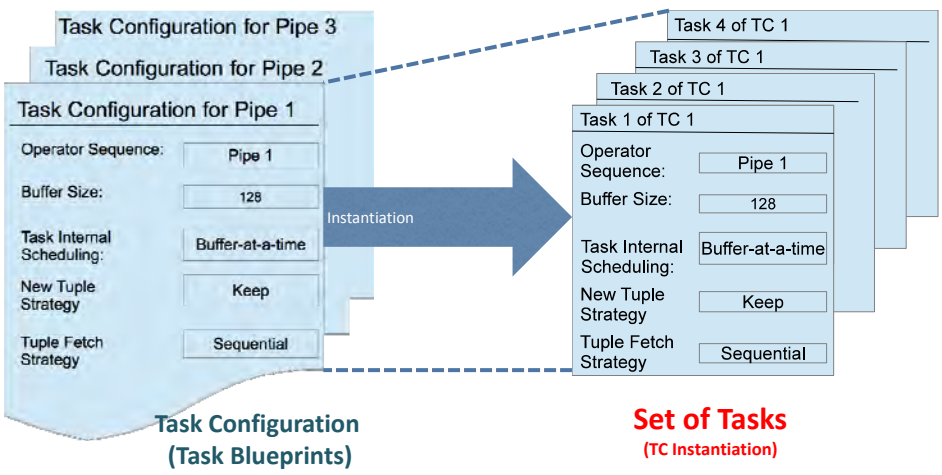
Choosing
Hash Join

Max. Pipelines
+
Dependency Graph

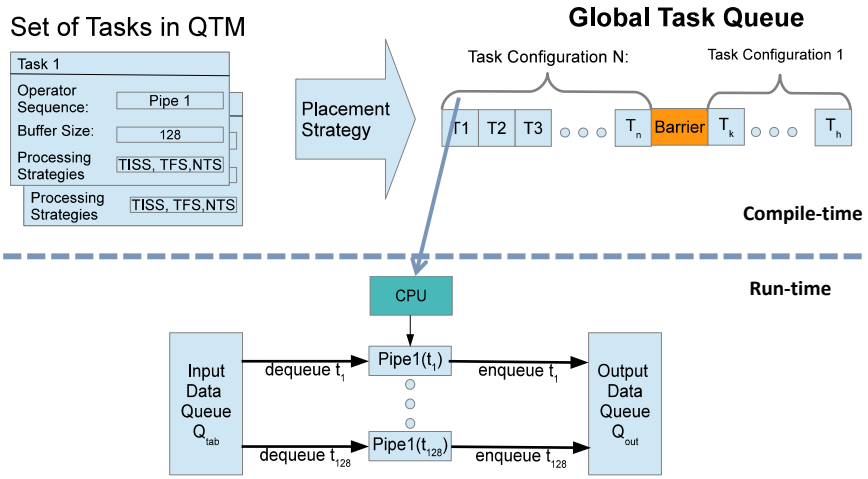
Step 2: QTM Task Configuration



Step 3: QTM Task Generation



Step 4. QTM Task Execution



Outline

a. Parallel Query Execution ✓
b. QTM: Query Task Model ✓
c. Evaluation
d. Outlook

Evaluation: Scenario

1) Tup - Pipe
B1 | B2 | Bar | P1

2) Tup - Mat
B1 | B2 | Bar | Sel | Mat | P1 | Mat | P2

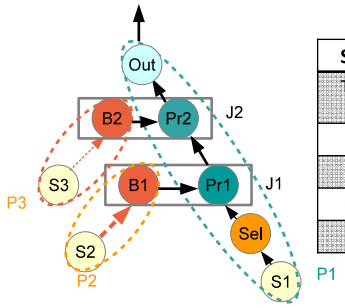
3) Tup - Seq
Sel | Mat | B1 | Bar | P1 | Mat | B2 | Bar | P2

4) Buf - CL & 5) L1 & 6) L2 & 7) L3
B1 | B2 | Bar | P1

8) Op - Mat
B1 | B2 | Bar | Sel | Mat | P1 | Mat | P2

9) Op - Seq
Sel | Mat | B1 | Bar | P1 | Mat | B2 | Bar | P2

Legend
Operator (TC) | Materialization | Barrier



Schedule	Workload
Tuples per Relation	30M
Selection	< 25M
S1 Values	0,1,2 ...
S2 Values	0,2,4,...
S3 Values	0,4,8,...

Evaluation: Configuration

1) Tup - Pipe
B1 | B2 | Bar | P1

2) Tup - Mat
B1 | B2 | Bar | Sel | Mat | P1 | Mat | P2

3) Tup - Seq
Sel | Mat | B1 | Bar | P1 | Mat | B2 | Bar | P2

4) Buf - CL & 5) L1 & 6) L2 & 7) L3
B1 | B2 | Bar | P1

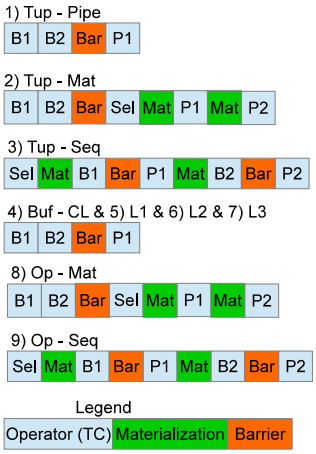
8) Op - Mat
B1 | B2 | Bar | Sel | Mat | P1 | Mat | P2

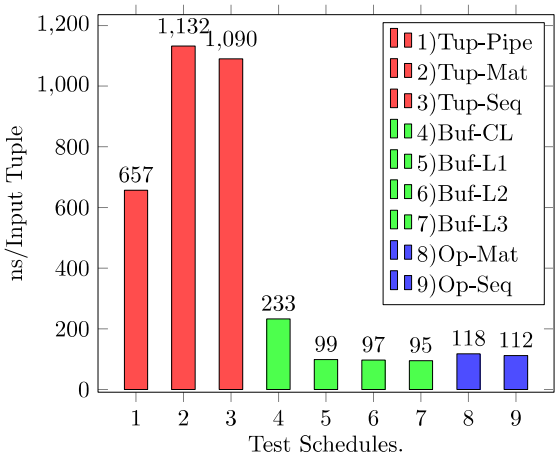
9) Op - Seq
Sel | Mat | B1 | Bar | P1 | Mat | B2 | Bar | P2

Legend
Operator (TC) | Materialization | Barrier

Schedule	Buffer Size	Tasks per Op	Total Tasks
1) Tup – Pipe	1	30M	90M
2) Tup – Mat	1	30M	150M
3) Tup – Seq	1	30M	150M
4) Buf - CL	4	7.5M	22.5M
5) Buf – L1	2,048	14,649	43,947
6) Buf – L2	16,384	1,832	5,496
7) Buf – L3	491,520	62	186
8) Op - Mat	7.5M	4	20
9) Op - Seq	7.5M	4	20

Evaluation: Runtimes

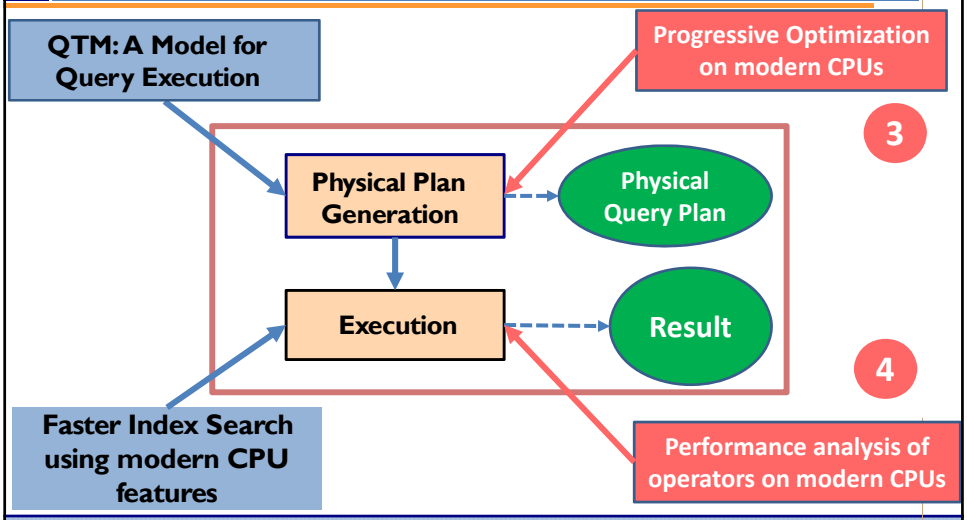




Evaluation: Insights

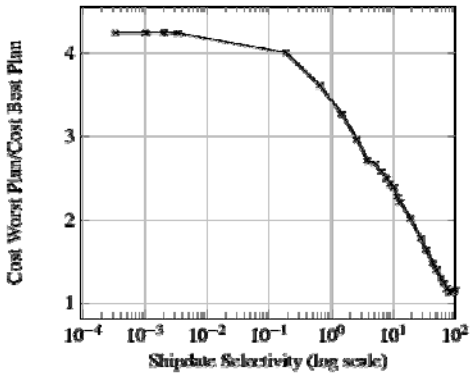
- Tradeoff between data and instruction cache performance
- **Medium sized tasks are data-efficient**:
  - **Pros**: Buffer fits entirely into cache, high data locality
  - **Cons**: High number of tasks and instructions
- **Large tasks are instruction-efficient**:
  - **Pros**: Decrease number of instructions and tasks, high instruction locality
  - **Cons**: More data cache misses if cache size is exceeded

Part 2: QTM: A Model for Query Execution



Motivation: TPC-H Q6 Example

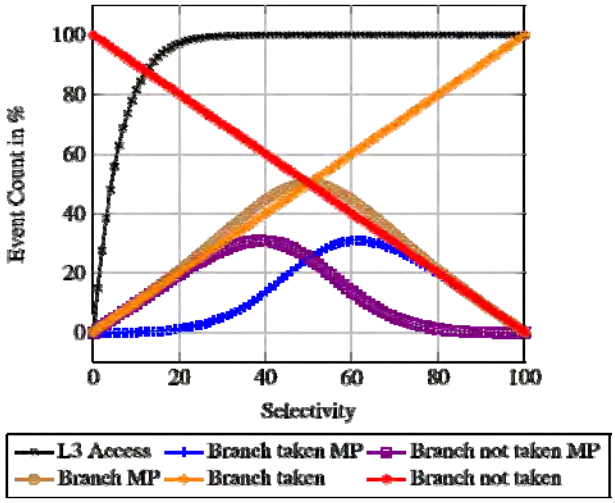
SELECT sum(l_extendedprice * l_discount) as revenue FROM lineitem
WHERE l_shipdate <=VALUE and l_quantity < 24 and
l_discount between 0.06 - 0.01 and 0.06 + 0.01



Approach: Steps Taken (Selections)

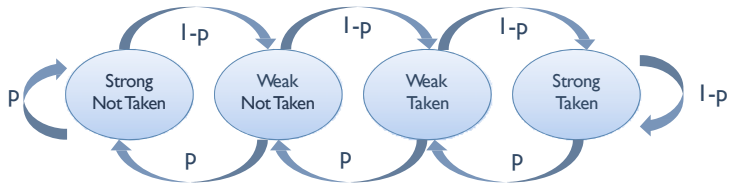
1. Use CPU counters
2. Thorough analysis (understanding) of operators with respect to CPU counters
3. From data to knowledge:
   - Interpret counters in the context of selections
4. Build model
   - Compare actual counters vs. predicted (by model)
   - Validate model extensively
5. Optimization algorithm by continuous observation (actual) and adjustment (predicted vs. actual)

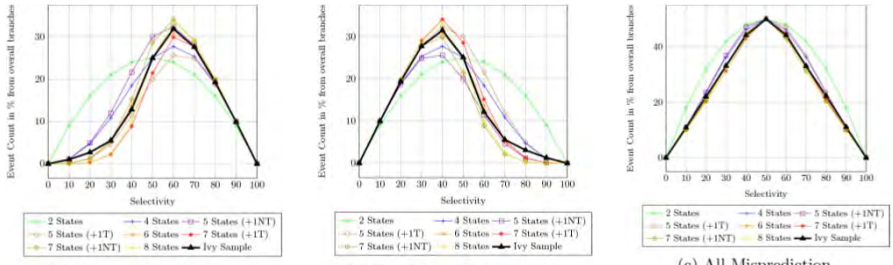
Cost Model for Selection



Non-Invasive Progressive Optimization for In-Memory Databases. PVLDB 9(14):

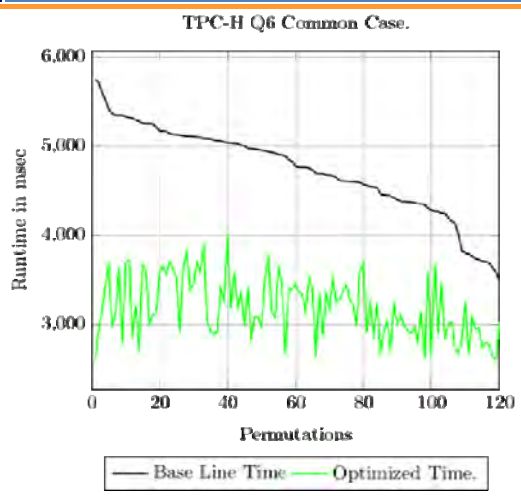
Branch Prediction (Markov Model)

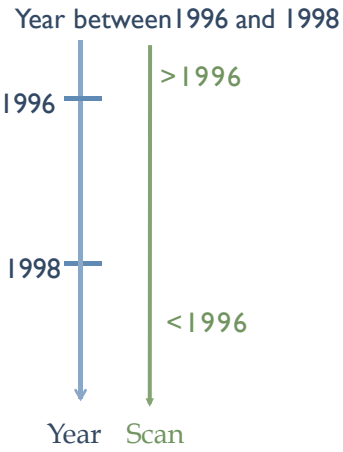




(a) Taken Misprediction. (b) Not Taken Misprediction. (c) All Misprediction.

TPC-H Query 6





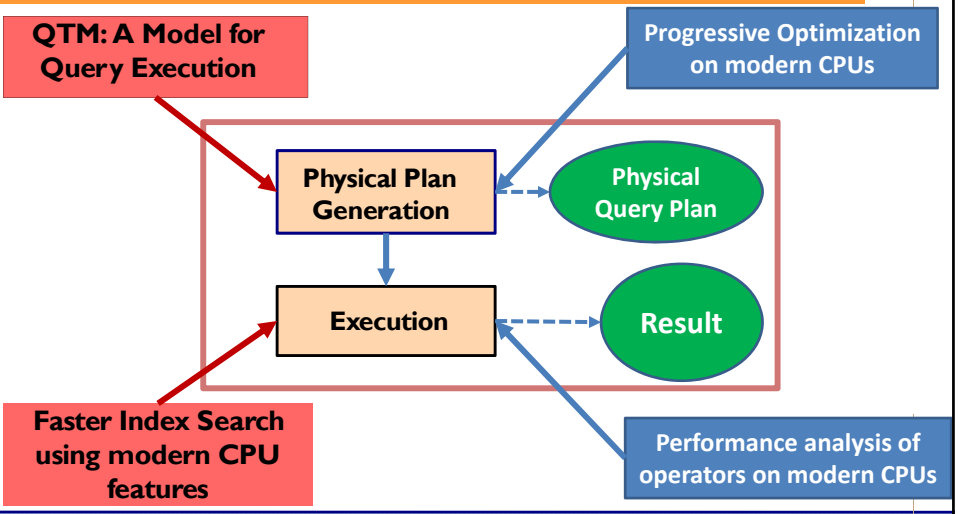
Contribution

- Markov model for branch prediction
- From Data to Knowledge: Interpreting counters for need to optimize
- Algorithm for continuous optimization of a sequence of selections
- **Future**: extend approach for joins (partially addressed in paper)

Publications

- QTM: Modelling Query Execution with Tasks, ADMS 2014
- Adapting Tree Structures for Processing with SIMD Instructions, EDBT 2014
- Selection on Modern CPUs, IMDM 2015
- Non-Invasive Progressive Optimization for In-Memory Databases, PVLDB 9(14) 2016

Summary



Questions ??