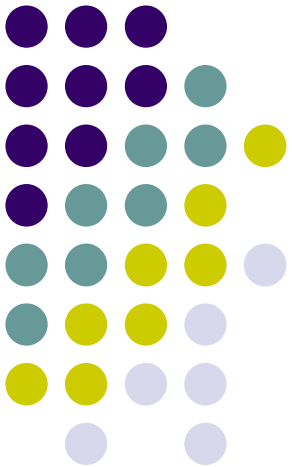
"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Cost models for Main-Memory database systems

Peter Boncz

Stefan Manegold

CWI (Amsterdam)



"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Contents

- Cost Components
- Disk-based vs. Main-Memory: Physical Costs
- Memory Access Pattern
- Conclusions

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Introduction: Purposes of Cost Models

estimate / predict performance in order to

- understand and analyze algorithms
- tune algorithms
- optimize QEPs

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Introduction: Cost Components

- **Logical Costs / Data Volume:**
  - estimate (intermediate) result sizes
  - "purely algebraic", independent of algorithms
  - require statistics about the stored data
  - synopses, histograms, wavelets, sampling, etc.
- **Algorithmic Costs:**
  - complexity of algorithms
  - O-classes
- **Physical Costs:**
  - execution time, memory consumption, etc.
  - require knowledge about hardware

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Physical Costs: Disk-based DBMS

- I/O costs (disk access) dominate execution
- CPU costs are negligible

  $\Longrightarrow \quad T = \#_{I/O} * t_{I/O}$

- estimation of $\#_{I/O}$ is easy due to
- full control over buffer management:
  - known replacement strategy, pinning, exclusive access
  - full associativity $\Rightarrow$ no conflicts

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Physical Costs: Main-Memory DBMS

- no I/O
- only CPU costs and memory access costs

  $\Longrightarrow \quad T = T_{CPU} + \#_{mem} * t_{mem}$
- estimation of $\#_{mem}$ is difficult due to
- no control over cache management:

  - unknown replacement strategy, no pinning, shared caches

  - limited associativity $\Rightarrow$ conflicts
- analytical modeling of CPU costs is virtually impossible:

  coding style, compiler optimization, runtime optimization (CPU), etc.

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Memory Access Costs

- From Radix-work:
  - $T = T_{CPU} + M_{L1} * l_{L1} + M_{L2} * l_{L2} + M_{TLB} * l_{TLB}$
  - T_{CPU} is calibrated per operator
  - cache/memory characteristics are measured per system (The Calibrator)
- Problem:
  - hand-crafting the cost functions that estimate the cache miss rates is too expensive and not intuitive
  - we need a more generic solution

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Memory Access Pattern: The Idea

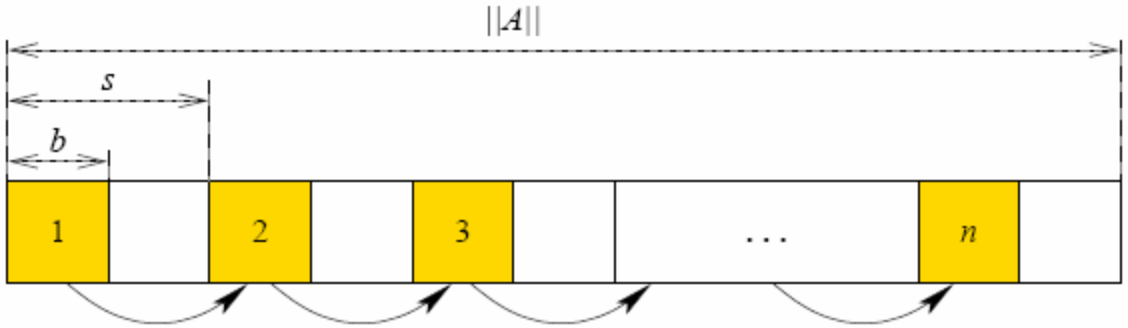
- specify basic memory access pattern and cost functions that calculate their cache miss rates
- use basic memory access pattern as building blocks to describe the memory access behavior of DB operators
  $\Longrightarrow$ compound memory access pattern
- provide rules to calculate the cache miss rates of compound memory access pattern using the cost functions of their building blocks

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

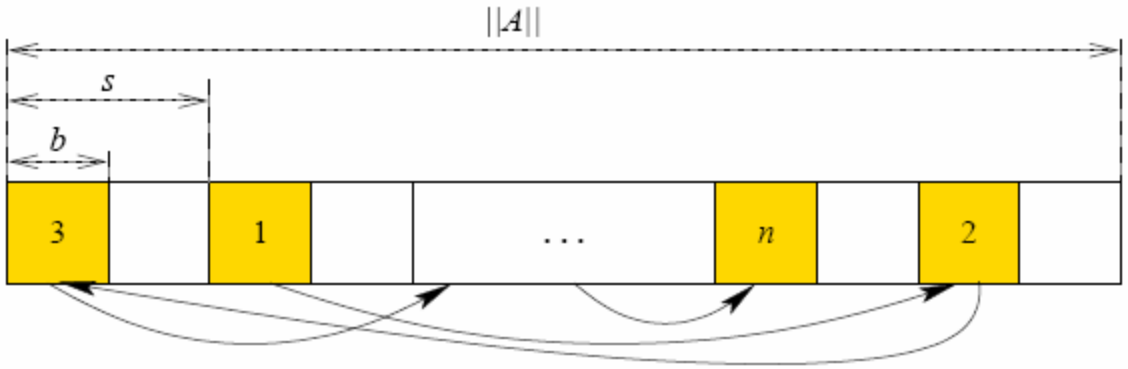
Simple Memory Access Pattern: $sMAP = (A, o, n, s, b)$

- traverse a memory area A once; $||A|| = n \cdot s \ \wedge \ b \leq s$

$o =$ seq



$o =$ ran



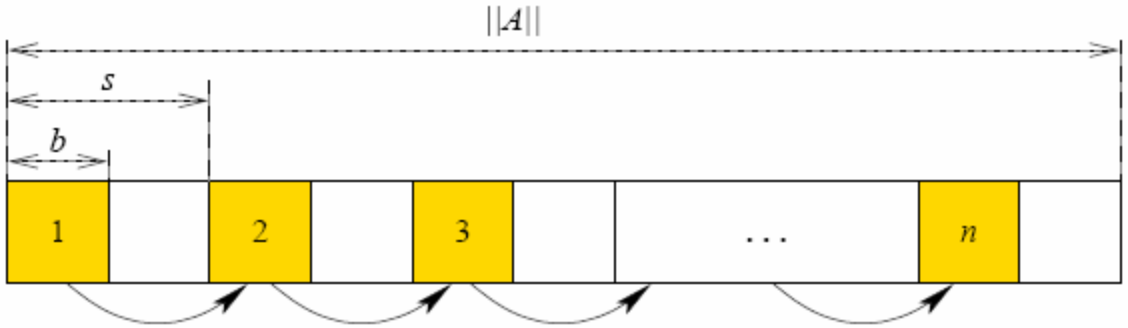
"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Simple Memory Access Pattern: $sMAP = (A, o, n, s, b)$

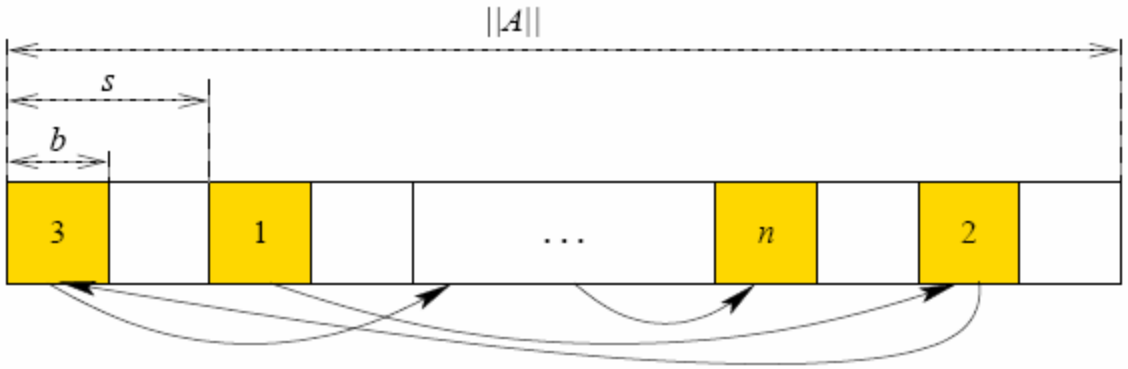
- traverse a memory area A once $||A|| = n \cdot s \ \wedge \ b \leq s$

Region size

$o = \text{seq}$



$o = \text{ran}$



"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Simple Memory Access Pattern: Cache Misses

$$M_{\llcorner i}((A, o, n, s, b)) =$$

$$\begin{cases} n \cdot \left(\left\lceil \frac{b}{LS_{\llcorner i}} \right\rceil + \frac{(b-1) \bmod LS_{\llcorner i}}{LS_{\llcorner i}} \right), & \text{if } s - b \geq LS_{\llcorner i} \quad \text{Skippy Access} \\ |A|_{\llcorner i}, & \text{if } s - b < LS_{\llcorner i} \wedge o = \text{seq} \\ |A|_{\llcorner i} & \\ + (n - \min\{|\llcorner i|_{\llcorner i}, |\llcorner i|_s\}) \cdot \left(1 - \min\left\{1, \frac{||\llcorner i||}{||A||}\right\}\right), & \text{if } s - b < LS_{\llcorner i} \wedge o = \text{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Simple Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((A, o, n, s, b)) =$$

$$\begin{cases} n \cdot \left(\left\lceil \frac{b}{LS_{\mathsf{L}i}} \right\rceil + \frac{(b-1) \bmod LS_{\mathsf{L}i}}{LS_{\mathsf{L}i}} \right), & \text{if } s - b \geq LS_{\mathsf{L}i} \\ |A|_{\mathsf{L}i}, & \text{if } s - b < LS_{\mathsf{L}i} \wedge o = \text{seq} \\ |A|_{\mathsf{L}i} & \\ + (n - \min\{|\mathsf{L}i|_{\mathsf{L}i}, |\mathsf{L}i|_s\}) \cdot \left(1 - \min\left\{1, \frac{||\mathsf{L}i||}{||A||}\right\}\right), & \text{if } s - b < LS_{\mathsf{L}i} \wedge o = \text{ran} \end{cases}$$

Full sequential access

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Simple Memory Access Pattern: Cache Misses

$$M_{\llcorner i}((A, o, n, s, b)) =$$

$$\begin{cases} n \cdot \left(\left\lceil \frac{b}{LS_{\llcorner i}} \right\rceil + \frac{(b-1) \bmod LS_{\llcorner i}}{LS_{\llcorner i}} \right), & \text{if } s - b \geq LS_{\llcorner i} \\ |A|_{\llcorner i}, & \text{if } s - b < LS_{\llcorner i} \wedge o = \text{seq} \\ |A|_{\llcorner i} + (n - \min\{|\llcorner i|_{\llcorner i}, |\llcorner i|_s\}) \cdot \left(1 - \min\left\{1, \frac{||\llcorner i||}{||A||}\right\}\right), & \text{if } s - b < LS_{\llcorner i} \wedge o = \text{ran} \end{cases}$$

Random access: compulsory misses + data misses

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Simple Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((A, o, n, s, b)) =$$

$$\begin{cases} n \cdot \left(\left\lceil \frac{b}{LS_{\mathsf{L}i}} \right\rceil + \frac{(b-1) \bmod LS_{\mathsf{L}i}}{LS_{\mathsf{L}i}} \right), & \\ |A|_{\mathsf{L}i}, & \text{if } s - b < LS_{\mathsf{L}i} \wedge o = \mathsf{seq} \\ |A|_{\mathsf{L}i} + (n - \min\{|\mathsf{L}i|_{\mathsf{L}i}, |\mathsf{L}i|_s\}) \cdot \left(1 - \min\left\{1, \frac{||\mathsf{L}i||}{||A||}\right\}\right), & \text{if } s - b < LS_{\mathsf{L}i} \wedge o = \mathsf{ran} \end{cases}$$

Sequential memory access

Random memory access

Beware automatic prefetching!

Sequential access is much cheaper
-5GB/s vs 500MB/s
Workaround: use "amoritzed laten
for sequential accesses

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Repetitive Memory Access Pattern: $rMAP = (S, r, d)$

- repeat $S = (A, o, n, s, b)$ r-times
- for $o =$ seq, we distinguish

  $d =$ uni(−directional) ("round-robin")

  $d =$ bi(−directional) ("back-and-forth")

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Repetitive Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((\mathcal{S}, r, d)) =$$

$$\begin{cases} M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) \leq |\mathsf{L}i|_{\mathsf{L}i} \\ r \cdot M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) > |\mathsf{L}i|_{\mathsf{L}i} \wedge d = \mathsf{uni} \\ M_{\mathsf{L}i}(\mathcal{S}) + (r-1) \cdot (M_{\mathsf{L}i}(\mathcal{S}) - K), & \text{else} \end{cases}$$

$$K = \begin{cases} |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \mathsf{seq} \\ \dfrac{|\mathsf{L}i|_{\mathsf{L}i}}{M_{\mathsf{L}i}(\mathcal{S})} \cdot |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \mathsf{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Repetitive Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((\mathcal{S}, r, d)) =$$

$$\begin{cases} M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) \leq |\mathsf{L}i|_{\mathsf{L}i} \\ r \cdot M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) > |\mathsf{L}i|_{\mathsf{L}i} \wedge d = \text{uni} \\ M_{\mathsf{L}i}(\mathcal{S}) + (r-1) \cdot (M_{\mathsf{L}i}(\mathcal{S}) - K), & \text{else} \end{cases}$$

Perfect reuse

$$K = \begin{cases} |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \text{seq} \\ \dfrac{|\mathsf{L}i|_{\mathsf{L}i}}{M_{\mathsf{L}i}(\mathcal{S})} \cdot |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \text{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Repetitive Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((\mathcal{S}, r, d)) =$$

$$\begin{cases} M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) \leq |\mathsf{L}i|_{\mathsf{L}i} \\ r \cdot M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) > |\mathsf{L}i|_{\mathsf{L}i} \wedge d = \text{uni} \\ M_{\mathsf{L}i}(\mathcal{S}) + (r-1) \cdot (M_{\mathsf{L}i}(\mathcal{S}) - K), & \text{else} \end{cases}$$

No reuse

$$K = \begin{cases} |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \text{seq} \\ \dfrac{|\mathsf{L}i|_{\mathsf{L}i}}{M_{\mathsf{L}i}(\mathcal{S})} \cdot |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \text{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Repetitive Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}((\mathcal{S}, r, d)) =$$

$$\begin{cases} M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) \leq |\mathsf{L}i|_{\mathsf{L}i} \\ r \cdot M_{\mathsf{L}i}(\mathcal{S}), & \text{if } M_{\mathsf{L}i}(\mathcal{S}) > |\mathsf{L}i|_{\mathsf{L}i} \wedge d = \mathsf{uni} \\ M_{\mathsf{L}i}(\mathcal{S}) + (r-1) \cdot (M_{\mathsf{L}i}(\mathcal{S}) - K), & \text{else} \end{cases}$$

"LRU" reuse

$$K = \begin{cases} |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \mathsf{seq} \\ \dfrac{|\mathsf{L}i|_{\mathsf{L}i}}{M_{\mathsf{L}i}(\mathcal{S})} \cdot |\mathsf{L}i|_{\mathsf{L}i}, & \text{if } o = \mathsf{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Sequential Execution

- $\mathcal{P}_1$, $\mathcal{P}_2$ memory access pattern

$$M_{\mathsf{L}i}(\mathcal{P}_1 ; \mathcal{P}_2) = M_{\mathsf{L}i}(\mathcal{P}_1) + M_{\mathsf{L}i}(\mathcal{P}_2)$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Concurrent Execution

- $\mathcal{Q}_1$, $\mathcal{Q}_2$ memory access pattern that do not revisit/re-use cache lines

- $\mathcal{R}_1$, $\mathcal{R}_2$ memory access pattern that do revisit/re-use cache lines

$$M_{\mathsf{L}i}(\mathcal{Q}_1 || \mathcal{Q}_2) = M_{\mathsf{L}i}(\mathcal{Q}_1) + M_{\mathsf{L}i}(\mathcal{Q}_2)$$

$$M_{\mathsf{L}i}(\mathcal{Q}_1 || \mathcal{R}_1) = M_{\mathsf{L}i}(\mathcal{Q}_1) + M_{\mathsf{L}i}(\mathcal{R}_1) = M_{\mathsf{L}i}(\mathcal{R}_1 || \mathcal{Q}_1)$$

$$M_{\mathsf{L}i}(\mathcal{R}_1 || \mathcal{R}_2) = M_{\mathsf{L}i/2}(\mathcal{R}_1) + M_{\mathsf{L}i/2}(\mathcal{R}_2)$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Concurrent Execution

- $\mathcal{Q}_1$, $\mathcal{Q}_2$ memory access pattern that do not revisit/re-use cache lines

- $\mathcal{R}_1$, $\mathcal{R}_2$ memory access pattern that do revisit/re-use cache lines

$$M_{\mathsf{L}i}(\mathcal{Q}_1 || \mathcal{Q}_2) = M_{\mathsf{L}i}(\mathcal{Q}_1) + M_{\mathsf{L}i}(\mathcal{Q}_2)$$

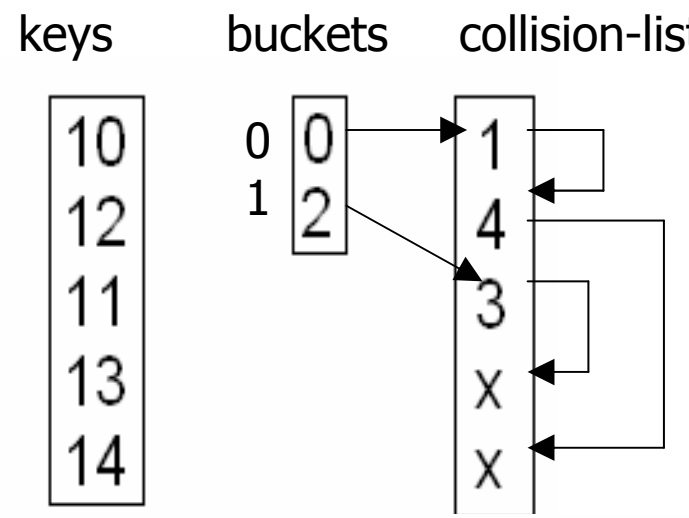
$$M_{\mathsf{L}i}(\mathcal{Q}_1 || \mathcal{R}_1) = M_{\mathsf{L}i}(\mathcal{Q}_1) + M_{\mathsf{L}i}(\mathcal{R}_1) = M_{\mathsf{L}i}(\mathcal{R}_1 || \mathcal{Q}_1)$$

$$M_{\mathsf{L}i}(\mathcal{R}_1 || \mathcal{R}_2) = M_{\mathsf{L}i/2}(\mathcal{R}_1) + M_{\mathsf{L}i/2}(\mathcal{R}_2)$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Hash Join $B := L \bowtie R$

$$\begin{aligned}\texttt{Build} = (\ & (R \quad\quad\quad\quad, \texttt{seq}, R.\texttt{BATcount}, R.\texttt{BUNsize}, R.\texttt{BUNsize}) \\ \| & (R \rightarrow \texttt{link}, \texttt{seq}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ \| & (R \rightarrow \texttt{mask}, \texttt{ran}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad))\end{aligned}$$



"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Hash Join $B := L \bowtie R$

$$\begin{aligned}\texttt{Build} = (\ & (R \quad\quad\quad\quad, \textsf{seq}, R.\texttt{BATcount}, R.\texttt{BUNsize}, R.\texttt{BUNsize}) \\ & \|(R \rightarrow \texttt{link}, \textsf{seq}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \rightarrow \texttt{mask}, \textsf{ran}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad))\end{aligned}$$

$$\begin{aligned}\texttt{Lookup} = (\ & (R \rightarrow \texttt{mask}, \textsf{ran}, L.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \rightarrow \texttt{link}, \textsf{ran}, B.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \quad\quad\quad\quad, \textsf{ran}, B.\texttt{BATcount}, R.\texttt{BUNsize}, R.\texttt{BUNsize}))\end{aligned}$$

$$\begin{aligned}\texttt{HashJoin} = (\texttt{Build}; (\ & (L \quad\quad\quad\quad, \textsf{seq}, L.\texttt{BATcount}, L.\texttt{BUNsize}, L.\texttt{BUNsize}) \\ & \| \texttt{Lookup} \\ & \|(B \quad\quad\quad\quad, \textsf{seq}, B.\texttt{BATcount}, B.\texttt{BUNsize}, B.\texttt{BUNsize})))\end{aligned}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Hash Join $B := L \bowtie R$

Writes are non-blocking ⇒ Latency = 0

$$\begin{aligned}\texttt{Build} = (\ & (R \quad\quad\quad\quad, \textsf{seq}, R.\texttt{BATcount}, R.\texttt{BUNsize}, R.\texttt{BUNsize}) \\ & \|(R \rightarrow \texttt{link}, \textsf{seq}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \rightarrow \texttt{mask}, \textsf{ran}, R.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad))\end{aligned}$$

$$\begin{aligned}\texttt{Lookup} = (\ & (R \rightarrow \texttt{mask}, \textsf{ran}, L.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \rightarrow \texttt{link}, \textsf{ran}, B.\texttt{BATcount}, 4 \quad\quad, 4 \quad\quad) \\ & \|(R \quad\quad\quad\quad, \textsf{ran}, B.\texttt{BATcount}, R.\texttt{BUNsize}, R.\texttt{BUNsize}))\end{aligned}$$

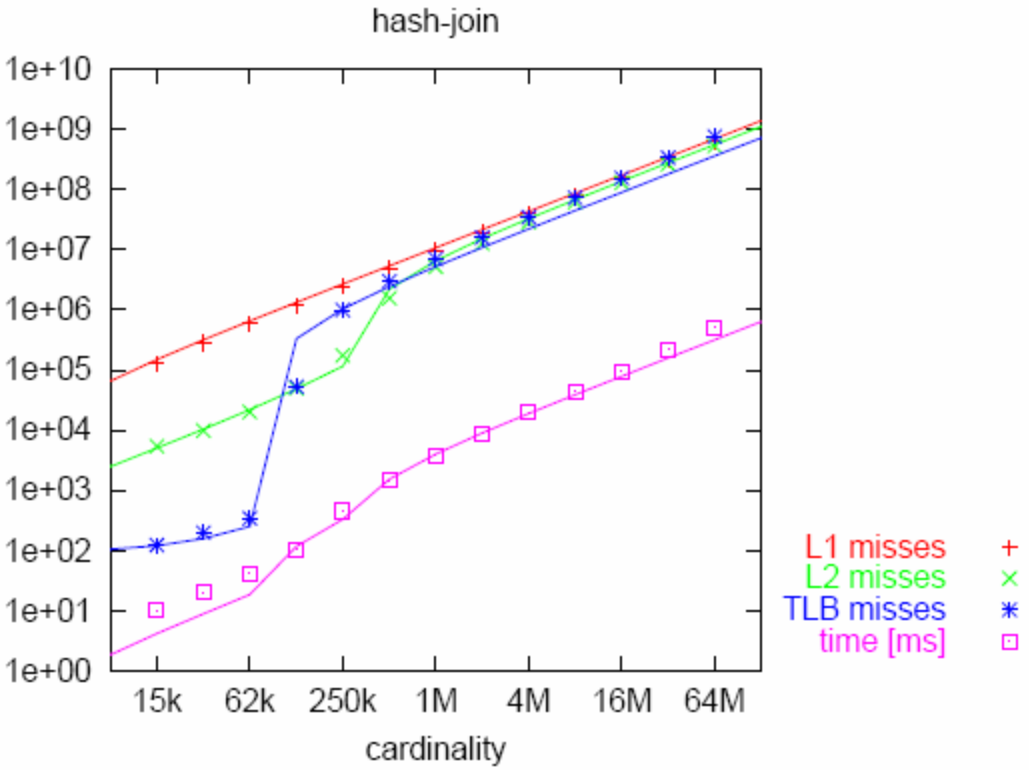
$$\begin{aligned}\texttt{HashJoin} = (\texttt{Build}; (\ & (L \quad\quad\quad\quad, \textsf{seq}, L.\texttt{BATcount}, L.\texttt{BUNsize}, L.\texttt{BUNsize}) \\ & \| \texttt{Lookup} \\ & \|(B \quad\quad\quad\quad, \textsf{seq}, B.\texttt{BATcount}, B.\texttt{BUNsize}, B.\texttt{BUNsize})))\end{aligned}$$

“Generic Database Cost Models for Hierarchical Memory Systems”, Manegold, Boncz, Kersten, VLDB’02

algorithm	pattern description		name
$W \leftarrow select(U)$	$\mathsf{s\_tra}(U) \odot \mathsf{s\_tra}(W)$		
$W \leftarrow nested\_loop\_join(U,V)$	$\mathsf{s\_tra}(U) \odot \mathsf{rs\_tra}(U.n, \mathtt{uni}, V) \odot \mathsf{s\_tra}(W)$	$=:$	$\mathsf{nl\_join}(U,V,W)$
$H \leftarrow hash\_build(V)$	$\mathsf{s\_tra}(V) \odot \mathsf{r\_tra}(H)$	$=:$	$\mathsf{build\_hash}(V,H)$
$W \leftarrow hash\_probe(U,H)$	$\mathsf{s\_tra}(U) \odot \mathsf{r\_acc}(U.n, H) \odot \mathsf{s\_tra}(W)$	$=:$	$\mathsf{probe\_hash}(U,H,W)$
$W \leftarrow hash\_join(U,V)$	$\mathsf{build\_hash}(V,H) \oplus \mathsf{probe\_hash}(U,H,W)$	$=:$	$\mathsf{h\_join}(U,V,W)$
$\{U_j\}\|_{j=1}^{m} \leftarrow cluster(U,m)$	$\mathsf{s\_tra}(U) \odot \mathsf{nest}(\{U_j\}\|_{j=1}^{m}, m, \mathsf{s\_tra}(U_j), \mathtt{ran})$	$=:$	$\mathsf{part}(U,m,\{U_j\}\|_{j=1}^{m})$
$W \leftarrow part\_nl\_join(U,V,m)$	$\mathsf{part}(U,m,\{U_j\}\|_{j=1}^{m}) \oplus \mathsf{part}(V,m,\{V_j\}\|_{j=1}^{m})$ $\oplus \mathsf{nl\_join}(U_1,V_1,W_1) \oplus \ldots \oplus \mathsf{nl\_join}(U_m,V_m,W_m)$		
$W \leftarrow part\_h\_join(U,V,m)$	$\mathsf{part}(U,m,\{U_j\}\|_{j=1}^{m}) \oplus \mathsf{part}(V,m,\{V_j\}\|_{j=1}^{m})$ $\oplus \mathsf{h\_join}(U_1,V_1,W_1) \oplus \ldots \oplus \mathsf{h\_join}(U_m,V_m,W_m)$		

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

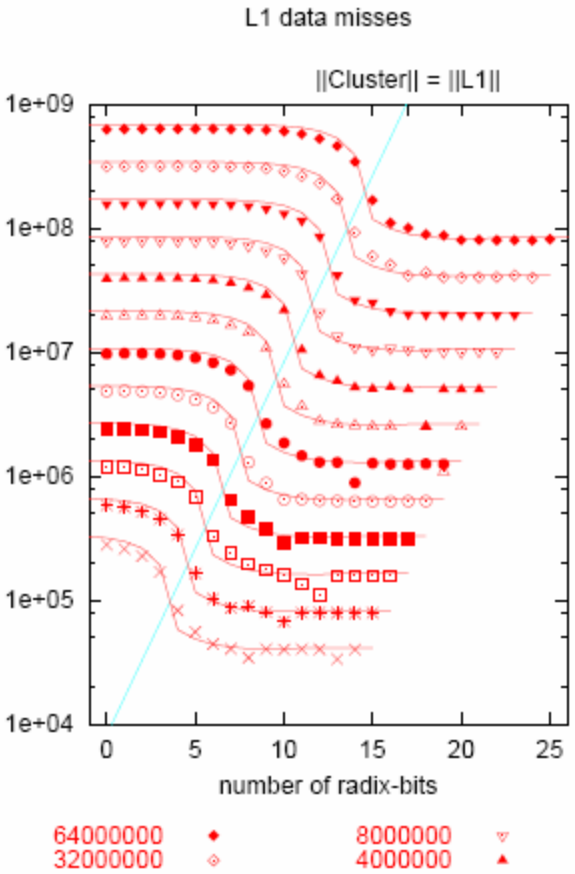
Compound Memory Access Pattern: Hash Join

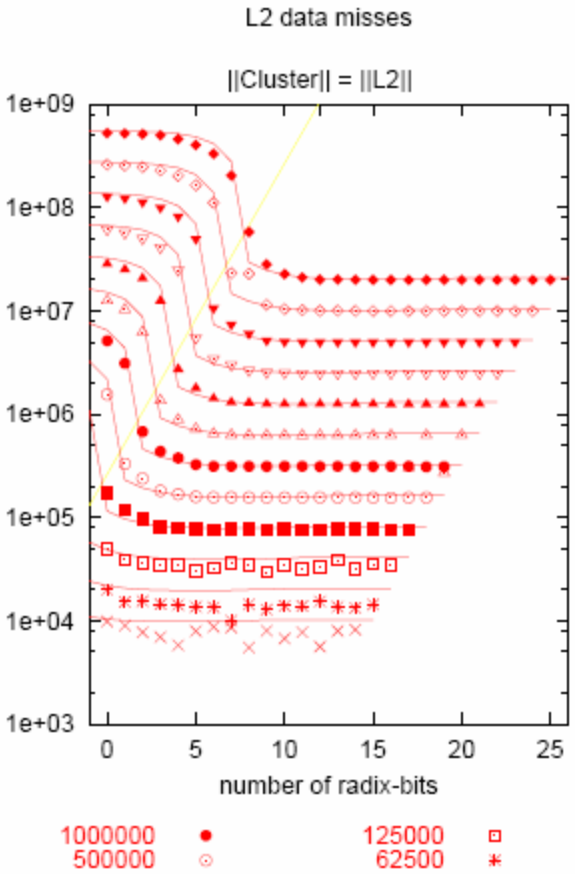


MonetDB

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

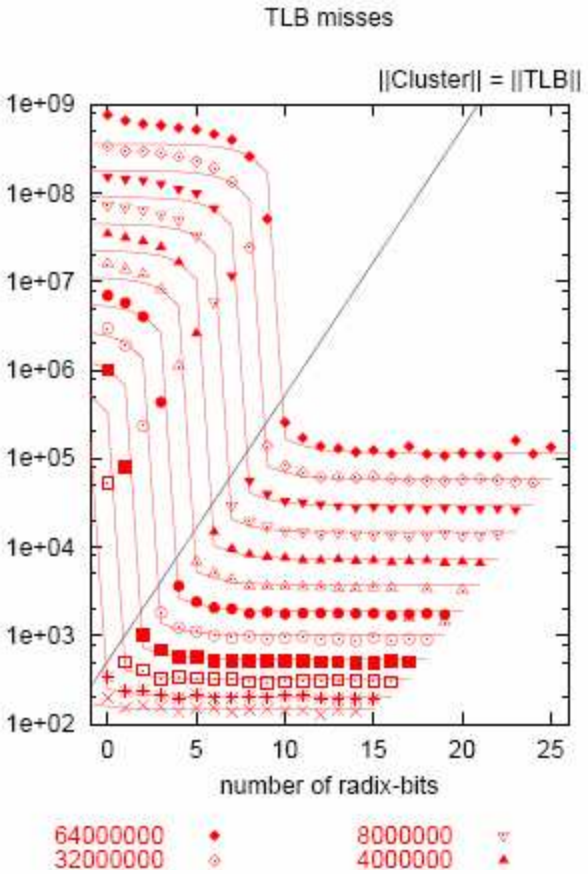
Compound Memory Access Pattern: Partitioned Hash Join

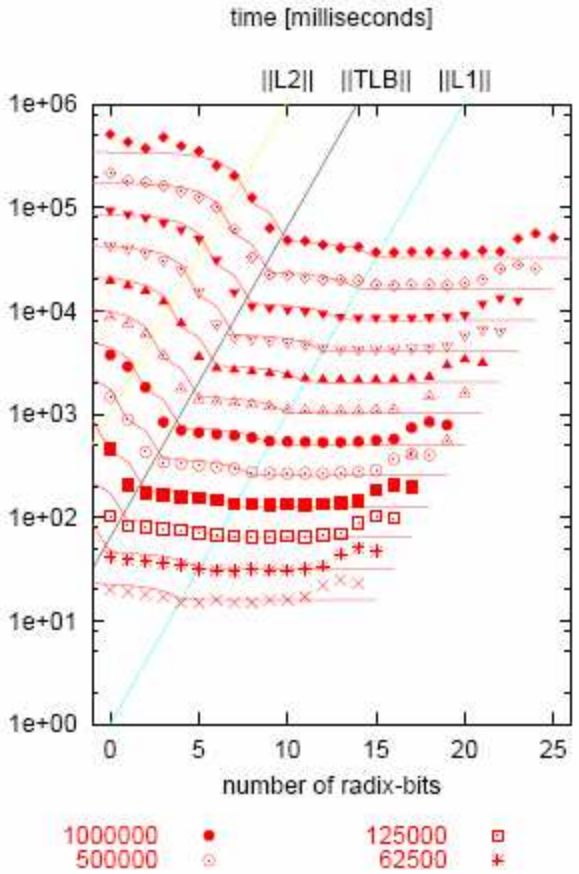




"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

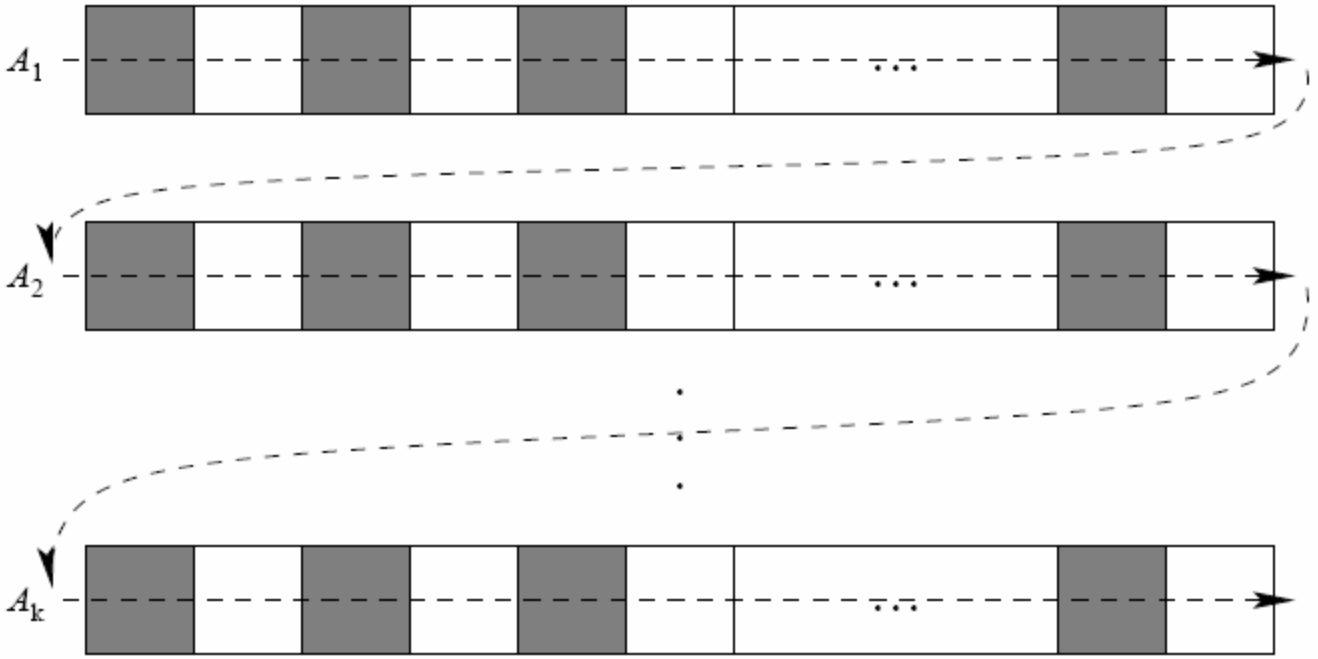
Compound Memory Access Pattern: Partitioned Hash Join





"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Nested Memory Access Pattern: $nMAP = (<S_1, \ldots, S_k>, Y, O, D)$



$Y = \text{con(catenated)}, \quad O = \text{seq}, \quad D = \text{uni}$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Nested Memory Access Pattern: Cache Misses

$$M_{\mathsf{L}i}\left((<\mathcal{S}_1,\ldots,\mathcal{S}_k>,Y,O,D)\right) =$$

$$\begin{cases} k \cdot M_{\mathsf{L}i}\left(\mathcal{S}_j\right), & \text{if } Y = \mathsf{con} \wedge s-b < LS_{\mathsf{L}i} \\ M_{\mathsf{L}i}\left(\mathcal{S}(O)\right), & \text{if } Y = \mathsf{int} \wedge o = \mathsf{ran} \wedge n = 1 \\ M_{\mathsf{L}i}\left(\mathcal{S}(o)\right) + X & \text{if } Y = \mathsf{int} \wedge o = \mathsf{seq} \wedge \\ & \quad s-b < LS_{\mathsf{L}i} \wedge k \cdot \left\lceil \frac{b}{LS_{\mathsf{L}i}} \right\rceil > |\mathsf{L}i|_{\mathsf{L}i} \\ M_{\mathsf{L}i}\left(\mathcal{S}(o)\right), & \text{else} \end{cases}$$

$$\mathcal{S}(x) = \left(\bigcup_{j=1}^{k} A_j, x, k \cdot n, s, b\right)$$

$$X = (n-1)\cdot(k-h)\ , \qquad h = \begin{cases} 0, & \text{if } O = \mathsf{seq} \wedge D = \mathsf{uni} \\ |\mathsf{TLB}|, & \text{if } O = \mathsf{seq} \wedge D = \mathsf{bi} \\ \frac{|\mathsf{TLB}|}{k} \cdot |\mathsf{TLB}|, & \text{if } O = \mathsf{ran} \end{cases}$$

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Radix Cluster

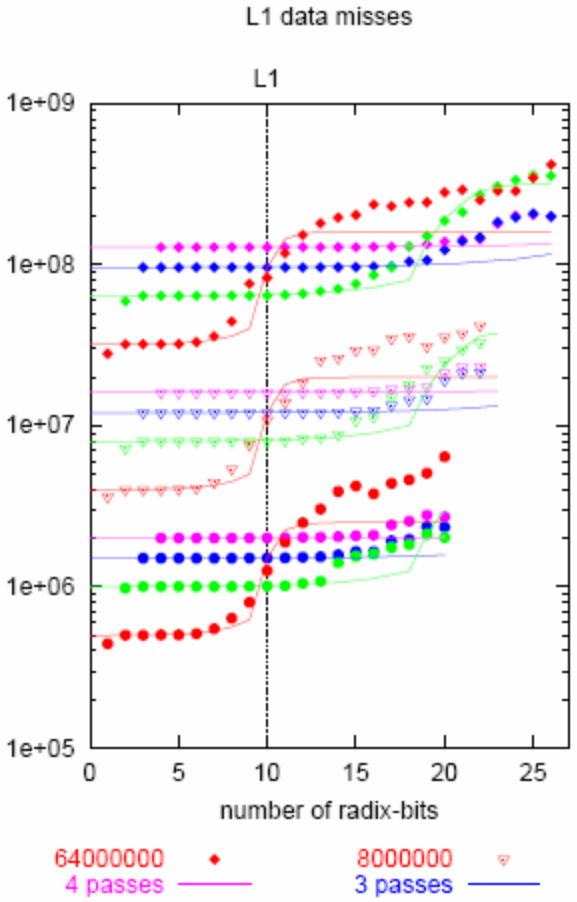
$$\mathcal{S}_j = (S_j, \mathsf{seq}, S_j.\mathtt{BATcount}, S_j.\mathtt{BUNsize}, S_j.\mathtt{BUNsize})$$

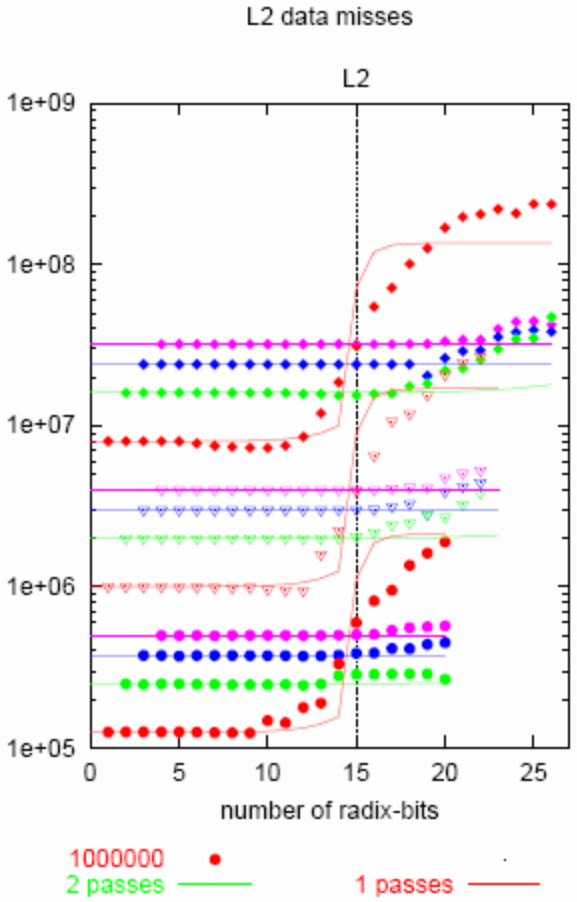
$$\mathtt{RadixCluster} = (\ (R, \mathsf{seq}, R.\mathtt{BATcount}, R.\mathtt{BUNsize}, R.\mathtt{BUNsize}) \\ \| (< \mathcal{S}_1, \ldots, \mathcal{S}_k >, \mathsf{int}, \mathsf{ran}) \ )$$

MonetDB

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Radix Cluster

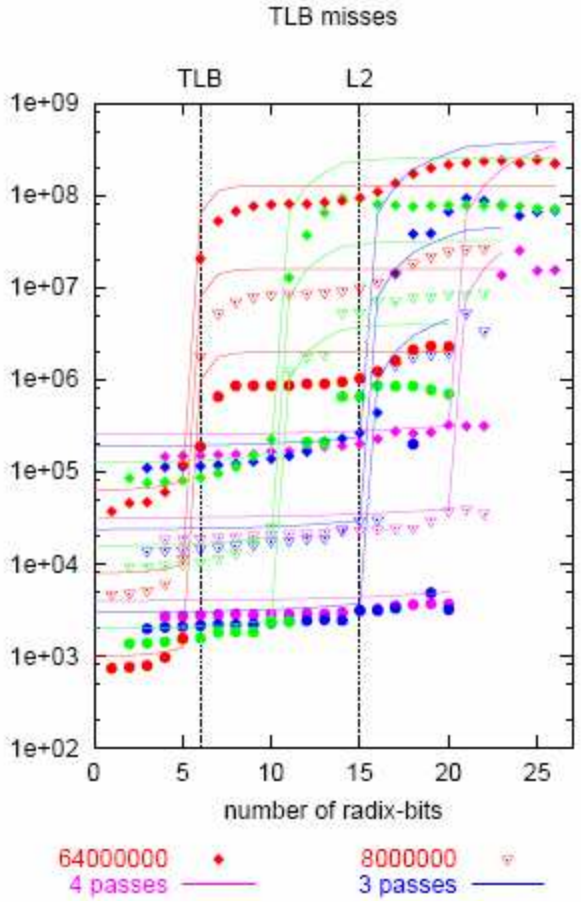


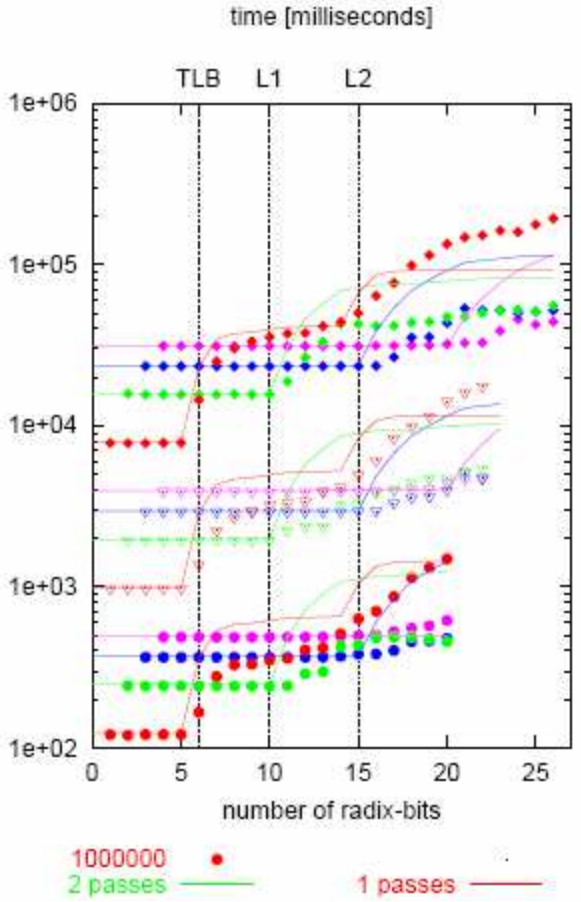


MonetDB

"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Compound Memory Access Pattern: Radix Cluster





"Generic Database Cost Models for Hierarchical Memory Systems", Manegold, Boncz, Kersten, VLDB'02

Conclusions

- generic, architecture-independent main-memory cost-modeling is possible (at least in Monet)
- cache impact is crucial for intra-operator costs
- use operator cost functions to always pick *the best* algorithm
  $\Rightarrow$ optimization on higher levels can use simpler cost models