Data Access Complexity: Monotonicity and Proportionality

Chen Ding Yifan Zhu

University of Rochester

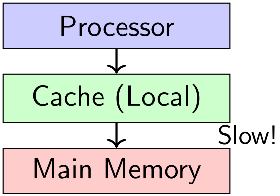
MEMSYS 2025

Outline

The Memory Hierarchy Challenge

Memory Wall Problem:

- Bandwidth constraint is fundamental bottleneck
- Cache manages locality
- Performance depends on:
  - Algorithm
  - Problem size
  - Cache size & policy



Memory Work Complexity

Time Complexity vs Data Access Complexity

- **Time Complexity**: Measures "processor work" (operations)
- **Data Access Complexity**: Measures "memory work" (data transfers)
- Locality = Lower data access complexity

Theoretical Questions

- Does larger problem size always mean worse locality?
- What is the worst-case cache performance we can guarantee?
- But first, why theory?

Shannon's Discovery of Entropy

The Three Axioms

Shannon sought a measure of uncertainty $H(p_1, \ldots, p_n)$ that satisfies:

1. **Continuity**: H is continuous in all p_i
2. **Monotonicity**: H increases with n for equally likely outcomes
3. **Additivity**: H adds for independent events:
   $H(X, Y) = H(X) + H(Y)$ for independent X, Y

The Unique Solution

These axioms lead uniquely to the form:

$$H(p_1, \ldots, p_n) = -K \sum_{i=1}^{n} p_i \log p_i$$

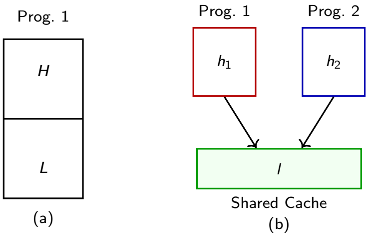
Setting $K = 1$ and using base-2 logarithm gives **bits**.

Brief Digression: Shared Exclusive Cache Hierarchy

Split LRU Stack Abstraction

Models exclusive cache hierarchy with two partitions:

- Upper partition H: Private cache (size h)
- Lower partition L: Shared victim cache (size l)
- Data evicted from H becomes victims stored in L



Victim Cache Requirement (VCR)

VCR Equation: For a single program using an exclusive hierarchy, the miss ratio must equal that of a combined cache:

$$vmr(h, l) = mr(h + l) \quad \text{for all } h, l \geq 0$$

Intuition: VCR ensures consistency with a single-level cache.

Footprint in Victim Cache [Ye et al. TACO 2017]

Victim Footprint (VFP) Definition

$$VFP(h, x) = fp(x_h + x) - h, \quad \text{where } fp(x_h) = h$$

- *fp*: footprint in single-level cache
- *h*: upper level cache size

Theorem (VFP Theorem 3.1)

The VFP defined above is the **only** *solution that satisfies the Victim Cache Requirement.*

Theoretical Significance

- **Uniqueness**: VFP is the only solution satisfying VCR
- **Composability**: VFPs of individual programs can be combined to model shared victim cache

Locality Terminology

Definition (Data Reuse)

Two consecutive accesses to the same data item

Reuse Measures

- **Reuse Interval (RI)**: Time between consecutive accesses
- **Reuse Distance (RD)**: Number of distinct items accessed between reuses

Example

For sequence "abcca":

- Reuse of 'a': RI $= 4$, RD $= 3$
- Reuse of 'c': RI $= 1$, RD $= 1$

Cache Models

LRU Cache

- Fixed size c
- Evicts Least Recently Used
- Miss when RD $> c$
- Most common in hardware

Working-Set Cache

- Variable size
- Data stays for time x
- Miss ratio: $mr(c) = P(ri > x)$
- Theoretical model

Focus

Data movement $=$ **demand caching** (no prefetching).

Locality Monotonicity: The Question

Does a larger problem always have worse locality?

Intuition Says YES

- More data to process
- More memory pressure
- Like time complexity

But...

- Cache behavior is complex
- Depends on access patterns
- Not always monotonic!
- *Example*: n-body simulation

Goal: Formalize conditions under which monotonicity holds

LRU Cache Monotonicity

Definition (Problem Size Ordering)

$\mathcal{P}'$ is larger than $\mathcal{P}$ if:

1. **Sequence Embedding**: Trace of $\mathcal{P}$ embedded in trace of $\mathcal{P}'$
2. **Intercept-Free**: No additional accesses to same data between reuses

Example (Intercept-Free)

$P = (a, b, c, a, d)$ embedded in $P' = (x, a, b, y, c, z, a, d, w)$

- Blue shows embedded accesses
- No intercept between the two 'a' accesses
- ✓ Intercept-free

Intercept Example

Example (With Intercept)

$P = (a, b, c, a, d)$ and $P' = (x, a, b, a, c, a, z, d, w)$

- Blue: embedded accesses from P
- Red: intercept access at position 4
- $\times$ NOT intercept-free
- The red 'a' disrupts the reuse pattern

Theorem (LRU Monotonicity)

For LRU cache of size c: $mc(\mathcal{P}', c) \geq mc(\mathcal{P}, c)$
if $\mathcal{P}$ has intercept-free embedding in $\mathcal{P}'$

Proof Sketch.

Every miss in $\mathcal{P}$ (RD $> c$) remains a miss in $\mathcal{P}'$ (RD' $\geq$ RD) $\square$

Alternative: Monotonic RD Injection

Definition (Monotonic RD Injection)

For each RD of value d in $\mathcal{P}$, there exists an RD in $\mathcal{P}'$ of value $d' \geq d$

Comparison

- Order-independent condition
- Less intuitive than intercept-free embedding
- Maybe difficult to verify

Theorem (LRU Monotonicity via RD)

Monotonic RD injection implies $mc(\mathcal{P}', c) \geq mc(\mathcal{P}, c)$ *for all* $c \geq 0$

Working-Set Cache Monotonicity

Definition (Monotonic RI Injection)

For each RI of value r in $\mathcal{P}$, there exists an RI in $\mathcal{P}'$ of value $r' \geq r$

Theorem (Working-Set Monotonicity)

Let $\mathcal{S}$ be cache hits in $\mathcal{P}$ that remain hits in $\mathcal{P}'$.
Then cache consumption $c(\mathcal{P}', \mathcal{S}) \geq c(\mathcal{P}, \mathcal{S})$

Cache Consumption

- Measured as time-space product
- Tenancy = time from access to next reuse/eviction
- Monotonic RI $\Rightarrow$ Monotonic consumption

Naive Matrix Multiplication

Algorithm:

for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    for $k = 1$ to n **do**
      $C[i,j] +=$
        $A[i,k] * B[k,j]$
    end for
  end for
end for

Table: RI & RD for 3-access (element granularity)

ri	Count	rd	Count
3	$n^3 - n^2$	3	$n^3 - n^2$
$3n$	$n^3 - n^2$	$2n + 1$	$n^3 - n^2$
$3n^2$	$n^3 - n^2$	$n^2 + 2n$	$n^3 - n^2$
∞	$3n^2$	∞	$3n^2$

Observation

Both RI and RD values and counts are monotone in n

Matrix Multiplication Monotonicity

Corollary

Naive matrix multiplication has monotonic locality for:

- *Element granularity caching*
- *Block granularity caching*
- *Both LRU and working-set caches*

Proof Sketch.

- Element granularity: RD/RI values monotonic in n (from table)
- Block granularity: Similar analysis with spatial reuse
- Monotonic injection exists when increasing n to $n+1$
- Therefore, miss count increases with problem size

$\square$

Block Granularity Matrix Multiplication

Table: RI for 4-access matrix multiplication (block size b)

ri	$P(ri)$	Count
1	$\frac{1}{4}$	n^3
3	$\frac{1}{4} - \frac{1}{4bn}$	$n^3 - \frac{n^2}{b}$
4	$\frac{1}{4} - \frac{1}{4b}$	$n^3 - \frac{n^3}{b}$
$4n - 4b + 4$	$\frac{1}{4b} - \frac{1}{4bn}$	$\frac{n^3}{b} - \frac{n^2}{b}$
$4n$	$\frac{1}{4} - \frac{1}{4b}$	$n^3 - \frac{n^3}{b}$
$4n^2 - 4nb + 4n$	$\frac{1}{4b} - \frac{1}{4bn}$	$\frac{n^3}{b} - \frac{n^2}{b}$
∞	$\frac{3}{4bn}$	$\frac{3n^2}{b}$

All values and counts are monotone in $n \Rightarrow$ Monotonic RI injection

Worst-Case Locality

Question

What is the **worst** locality we can expect under optimal caching?

Theorem (Random Access Miss Ratio)

When accessing n blocks randomly with any stack algorithm:

$$mr(c) = \begin{cases} \frac{n-c}{n} & 0 \leq c \leq n \\ 0 & c > n \end{cases}$$

- Stack algorithms: LRU, MRU, LFU, Random, OPT
- All have the same miss ratio for random access
- Linear decrease with cache size

Cyclic Access Patterns

Cyclic Pattern: $a, b, c, d, a, b, c, d, \ldots$

Trace	a	b	c	d	a	b	c	d	a	b	c	d
OPT Stack 1	a	b	c	d	a	b	c	d	a	b	c	d
2		a	a	a	d	d	d	c	c	c	b	b
3			b	b	b	a	a	a	d	d	d	c
4				c	c	c	b	b	b	a	a	a
OPT Dist	∞	∞	∞	∞	2	3	4	2	3	4	2	3
c=2 Hits					H			H			H	

OPT stack for cyclic accesses ($n = 4$)

Theorem (Cyclic Access Locality)

Optimal locality of cyclic accesses is asymptotically the same as random access

Proof Sketch.

OPT distance uniformly distributed in $[2, n]$

$$mr(c) = \frac{n-c}{n-1} \approx \frac{n-c}{n}$$

$\square$

Worst-Case Proportionality

Theorem (Main Result: Proportionality)

For any workload accessing n data items, with optimal caching:

$$mr(c) \leq \begin{cases} \frac{n-c}{n-1} & 0 \leq c \leq n \\ 0 & c > n \end{cases}$$

Proof Idea.

- Sort items by access frequency
- Cache top c most frequently accessed blocks
- Each appears at least $\frac{kc}{n}$ times (in k total accesses)
- Guarantees: $mr(c) \leq \frac{n-c}{n} \leq \frac{n-c}{n-1}$

$\square$

Proportionality: Implications

Universal Guarantee

- Holds for **all programs and inputs**
- Requires only optimal cache management
- No program optimization needed
- Bound is **reachable**

What it means:

- Doubling cache size
- $\Rightarrow$ At least halves miss ratio

Contributions

1. Monotonicity

- Formalized conditions for locality monotonicity
- **LRU**: Intercept-free embedding or Monotonic RD injection
- **Working-set**: Monotonic RI injection
- Proved for naive matrix multiplication (element & block granularity)

2. Proportionality

- Worst-case miss ratio is $\leq \frac{n-c}{n-1}$ for optimal caching
- Universal property (all programs)
- Linear improvement with cache size is guaranteed
- Bound is tight (achieved by cyclic/random access)

Practical Implications

For Algorithm Designers

- Check monotonicity conditions for your algorithm
- Monotonic programs: cannot improve locality by increasing problem size

For Cache Implementers

- Target at least linear performance scaling
- If observed performance is sub-linear, optimization possible
- Proportionality is achievable with frequency-based replacement

Key Message

These are **complexity theorems** — properties that hold across all inputs

Future Directions

- **Monotonicity:**
  - Characterize larger class of monotonic programs
  - Automated proofs of monotonicity
  - Extension to parallel programs
- **Proportionality:**
  - Practical algorithms achieving proportionality
  - Multi-level shared cache

Thank You!

Questions?

Data Access Complexity:
Monotonicity and Proportionality

Chen Ding Yifan Zhu
University of Rochester

cding@cs.rochester.edu
yifanzhu@rochester.edu