Cache and Locality of Reference

Debapriya Roy

Department of Computer Science

October 30, 2025

Motivation

- Modern processors are much faster than main memory.
- To bridge this speed gap, a **cache memory** is used.
- Cache works efficiently due to a key property of program behavior called the **Locality of Reference**.
- Understanding locality helps in designing faster systems and efficient programs.

Definition: Locality of Reference

Definition

Locality of reference refers to the tendency of a program to access a relatively small portion of its address space during any short period of time.

- Memory accesses are not random.
- Access patterns exhibit **repetition** and **clustering**.
- Two major forms:
  - **Temporal Locality**
  - **Spatial Locality**

Temporal Locality

Concept

Temporal locality means that if a memory location is referenced, it is likely to be referenced again in the near future.

- Example: Loop counters, frequently used variables, instruction sequences.
- Caches keep recently accessed data for reuse.

Example

```
for (i = 0; i < 1000; i++)          sum += arr[i];
```

- The variable `i` and `sum` are reused repeatedly — showing temporal locality.

Spatial Locality

Concept

Spatial locality means that if a memory location is accessed, nearby memory locations are likely to be accessed soon.

- Occurs in sequential access of arrays, loops, and instruction fetches.
- Caches fetch **blocks of data (cache lines)** rather than individual bytes.

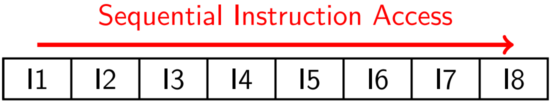
Example

Accessing an array sequentially: `for (i = 0; i < 100; i++) sum += A[i];`

- Once a block containing `A[i]` is fetched, elements `A[i+1]`, `A[i+2]` etc. are also likely available in cache.

Sequential Locality

- A special case of spatial locality.
- Common during instruction fetch in loops or functions.
- Example: Instructions stored consecutively in memory.



Sequential Instruction Access

I1	I2	I3	I4	I5	I6	I7	I8

Cache and Locality Relationship

- Cache memory stores frequently or recently accessed data.
- Exploits both:
    - **Temporal locality:** Reuse of data/instructions.
    - **Spatial locality:** Fetching nearby memory locations together.
- Without locality, caching would be ineffective.

Example

Instruction and data caches in CPUs are designed with block-based fetching to exploit spatial locality.

Hardware Mechanisms for Locality

- **Cache Lines:** Groups of contiguous bytes fetched together.
- **Prefetching:** Anticipating sequential access.
- **Replacement Policies:** Retain recently used blocks (e.g., LRU).

Observation

Efficient use of locality can reduce cache misses and improve CPU performance.

Summary Table

Type	Description	Cache Strategy
Temporal Locality	Reuse of recently accessed data or instructions.	Keep recently used data in cache.
Spatial Locality	Access of nearby data or instructions in memory.	Fetch cache lines or blocks containing adjacent addresses.
Sequential Locality	Sequential access of data or instructions, e.g., loops.	Prefetch next block sequentially.

Key Takeaways

- Locality of reference is fundamental to cache efficiency.
- Programs with high locality perform better on modern architectures.
- Compilers and programmers can optimize code to improve locality.
- Examples:
  - Loop tiling in matrices (improves spatial locality)
  - Reusing variables within loops (improves temporal locality)

Thank You! Questions?