Dolla Dolla Bills

Cache-Friendly Rust

Zach Mitchell

Note

This is a summary of another talk:

- Rust Cologne, March 2018: Florian Zeitz - Caches and You

Which is largely a summary of another talk:

- code::dive conference 2014 - Scott Meyers: CPU Caches and Why You Care

tl;dr

- Store data contiguously
- Access data linearly
- Measure everything
- Use Vec
- Don't use linked lists

CPU Caches

L1	L2	L3	RAM
Core-local	Core-local	Shared	
32 KiB	256 KiB	8 MB	> 1 GB
1 ns	3 ns	10 ns	60 ns

Cache Lines

- Read/Writes to RAM are in units of cache lines
- 64 bytes on most systems
- Nearby cache lines are prefetched when linear reads are detected

Cache Coherence

- The data in L1/L2 cache is the same for each core
- Data is mirrored in higher cache levels
- Writing to a cache line invalidates it for other cores

Example: Matrix Multiplication

- **Row-major**: the default, rows of an array are stored contiguously
- **Column-major**: columns of an array are stored contiguously

Example: Matrix Multiplication

Say you have two matrices, A and B.

$$C = A \times B$$

```
for row in A:
    for column in B:
        sum = 0
        for k in 0..N:
            sum += A[row, k] * B[k, column]
        C[row, column] = sum
```

Example: Matrix Multiplication

```
for row in A:
    for column in B:
        sum = 0
        for k in 0..N:
            sum += A[row, k] * B[k, column]
        C[row, column] = sum
```

- Proceeds linearly through a single column of A
- Makes row-sized jumps through B

Solution: store B as column major

Example: Matrix Multiplication

What were columns of B are now rows:

```
for row_a in A:
    for row_b in NewB:
        sum = 0
        for k in 0..N:
            sum += A[row_a, k] * NewB[row_b, k]
        C[row_a, row_b] = sum
```

- This proceeds linearly through both A and B
- 9x faster on the speaker's laptop

Example: Matrix Multiplication

Bottom line:

- How you store your data matters
- CPUs are good at linear reads
- Avoid skipping large chunks of data

Objects

How not to do it:

- Say struct `Foo` is larger than a cache line
- `Foo` has a field `bar: u8`
- Fill an array with `Foo`s
- You iterate, checking each `Foo.bar`

The `bar`s are a small fraction of that array

Objects

Try this instead:

- Store an external array of `bars`
- Try a struct of arrays rather than an array of structs
  - Apparently common in HPC and physics

Code Size

- Smaller code fits better in cache, performs better
- Sometimes it's better not to inline functions
  - Use `#[inline(never)]` for this
- A cached function may be faster than inlining in multiple places

Code Size

- Generic functions duplicate code due to monomorphization
  - Try Trait objects instead
- Measure, measure, measure!

"False Sharing"

- Writing to a cache line from one core invalidates it for other cores
- This causes a cache miss on the other cores

Data Structures: Vec

- One heap allocation
- One contiguous chunk of memory
- As cache-friendly as it gets when read linearly
- Beats other data structures at their own game due to cache efficiency

Data Structures: HashMap

- Rust's implementation is very cache efficient
- Values with the same hash are stored contiguously

Data Structures: BTreeMap

- Binary tree with multiple elements per node
- Better cache efficiency than normal binary tree

Data Structures: Linked List

- One heap allocation per node
- Every element is a cache miss, iteration is expensive
- Have to walk the list to find an element
- Rust doesn't even let you insert into the middle of the list

LAME

Data Structures: VecDeque

- For when you want a linked list but have self-respect
- One heap allocation
- Basically a Vec with fast `push_front` and `push_back`

END

Resources

- Rust Cologne, March 2018: Florian Zeitz - Caches and You
- A Survey of CPU Caches | Lukas Waymann
- What Every Programmer Should Know About Memory

Resources

- How L1 and L2 CPU Caches Work, and Why They're an Essential Part of Modern Chips
- code::dive conference 2014 - Scott Meyers: Cpu Caches and Why You Care
  - Handouts

Contact

- Zach Mitchell
- tinkering.xyz
- Email: zmitchell@fastmail.com
- LinkedIn