COL380

Introduction to
Parallel & Distributed Programming

2-0-2

Subodh Kumar

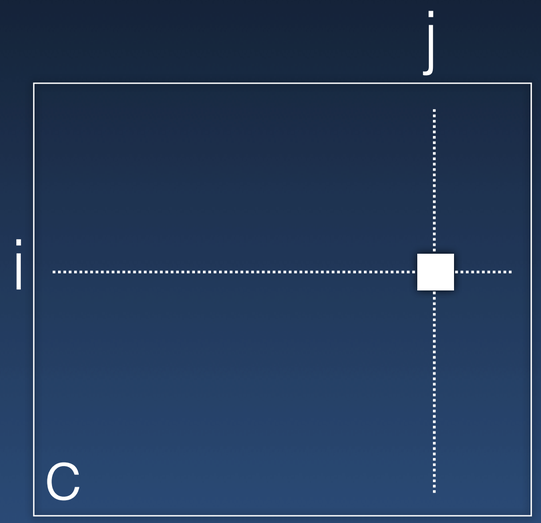
Agenda

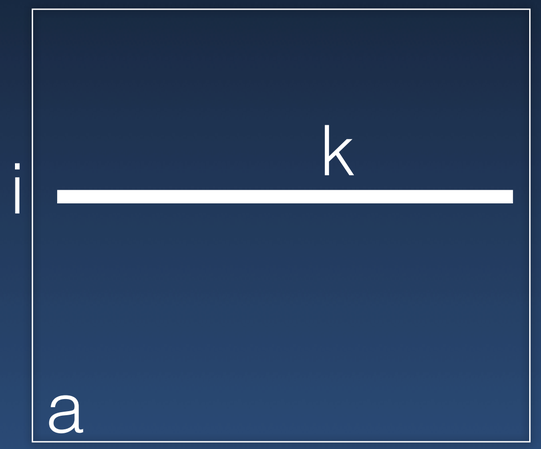
- Cache behavior, false sharing
- Flavors of parallelism
- Memory bottleneck
- Intro to architecture

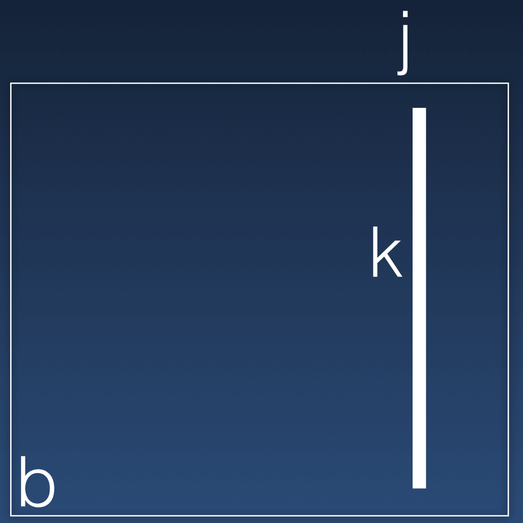
Matrix Multiplication


```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

Matrix Multiplication

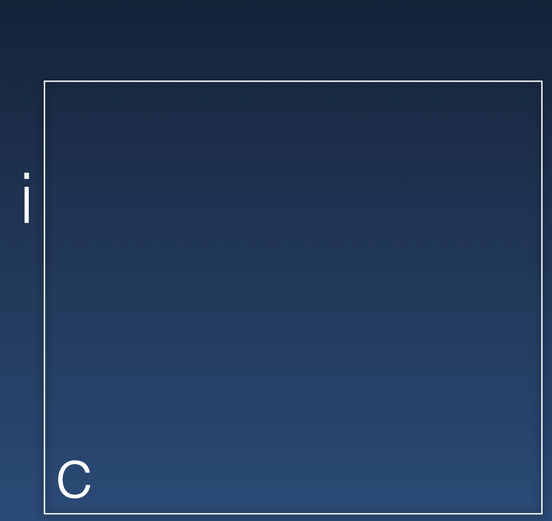


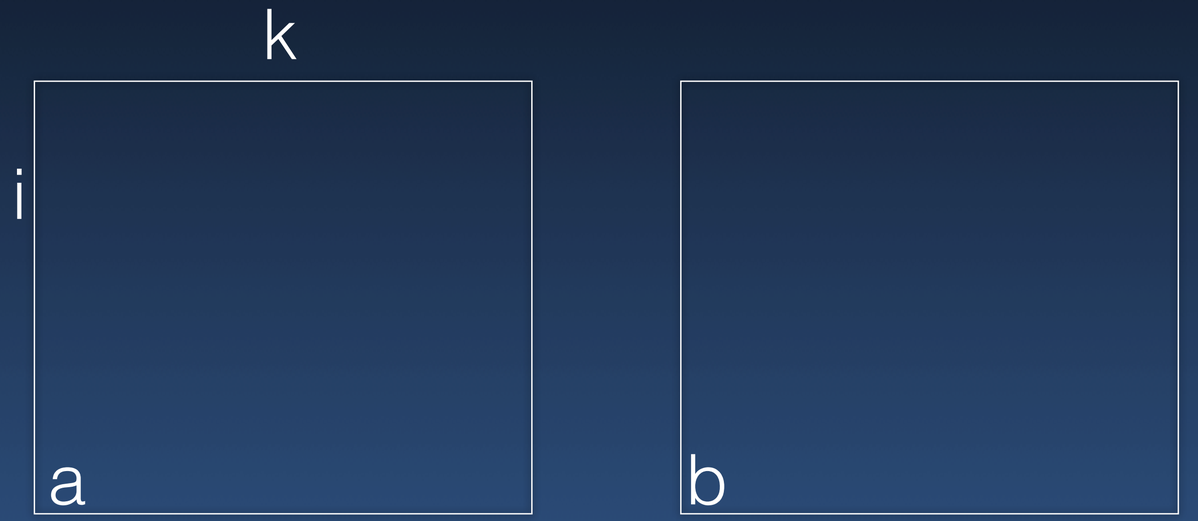





```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

Matrix Multiplication



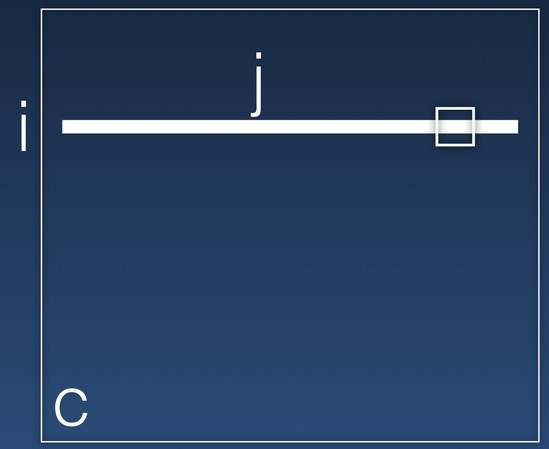


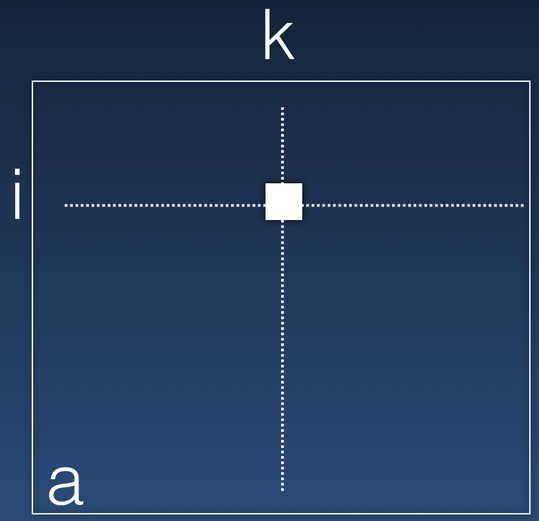

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

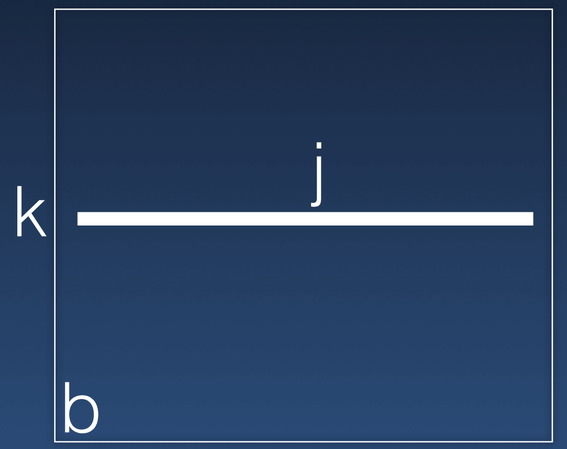
```
for (int i = 0; i < N; i++)
    for (int k = 0; k < N; k++)
        for (int j = 0; j < N; j++)
            c[i][j] += a[i][k]*b[k][j];
```

Subodh Kumar

Matrix Multiplication





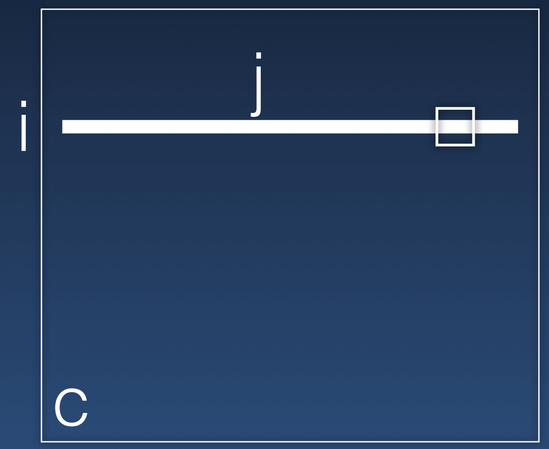


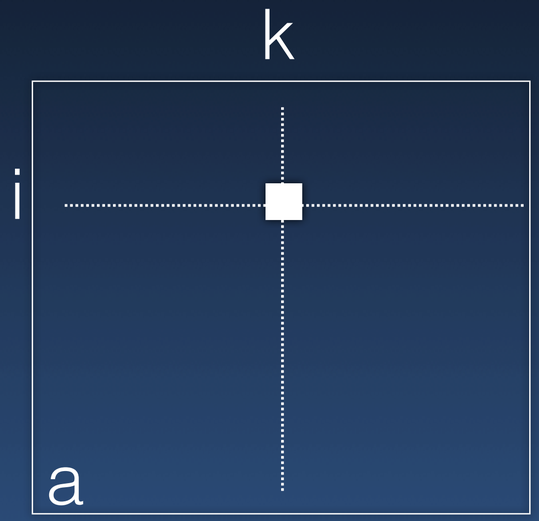

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

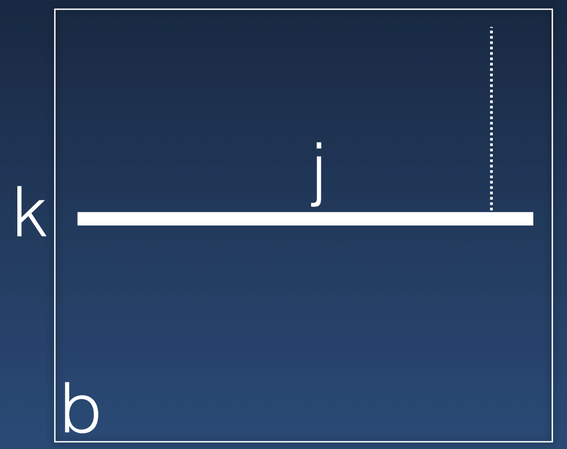
```
for (int i = 0; i < N; i++)
    for (int k = 0; k < N; k++)
        for (int j = 0; j < N; j++)
            c[i][j] += a[i][k]*b[k][j];
```

Subodh Kumar

Matrix Multiplication



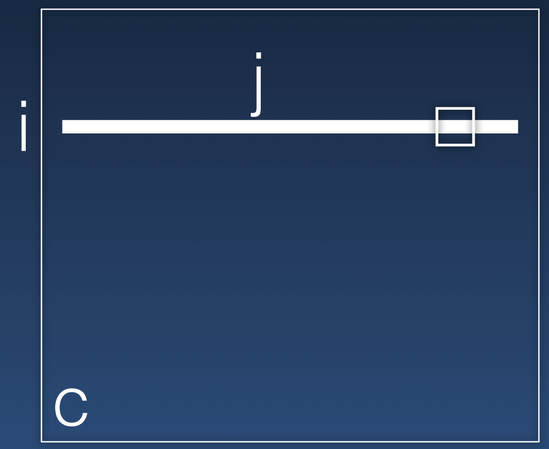


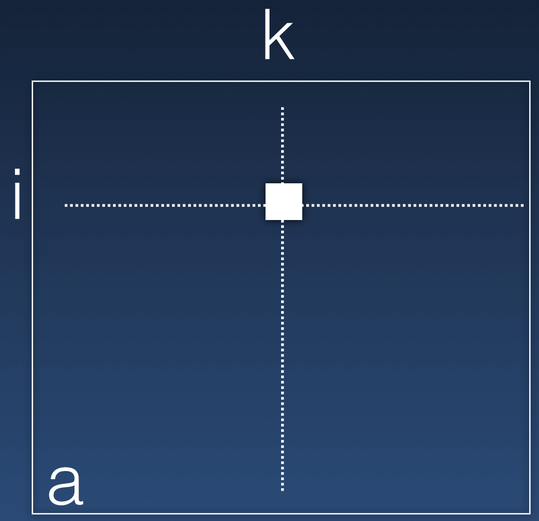


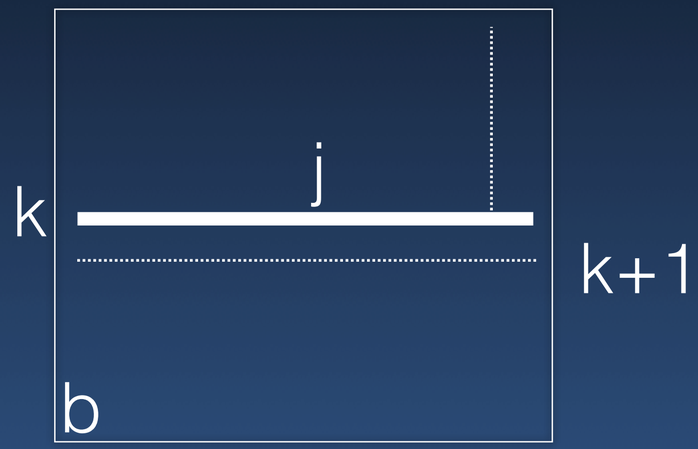

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

```
for (int i = 0; i < N; i++)
    for (int k = 0; k < N; k++)
        for (int j = 0; j < N; j++)
            c[i][j] += a[i][k]*b[k][j];
```

Matrix Multiplication








```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k]*b[k][j];
```

```
for (int i = 0; i < N; i++)
    for (int k = 0; k < N; k++)
        for (int j = 0; j < N; j++)
            c[i][j] += a[i][k]*b[k][j];
```

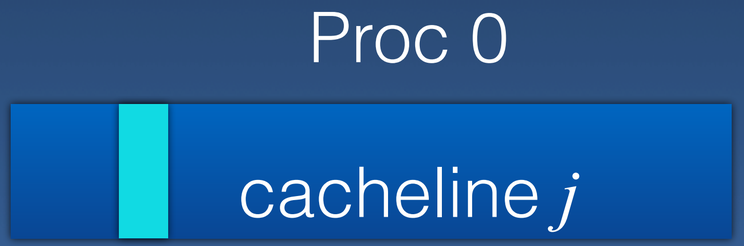
Subodh Kumar

Multiple Caches

- Coherent caches
  - ➡ Writes in on cache are updated in all cache copies
- Caches operate on line granularity
  - ➡ To write one item, may need to update the other items first

Multiple Caches

- Coherent caches
  - ➡ Writes in on cache are updated in all cache copies
- Caches operate on line granularity
  - ➡ To write one item, may need to update the other items first

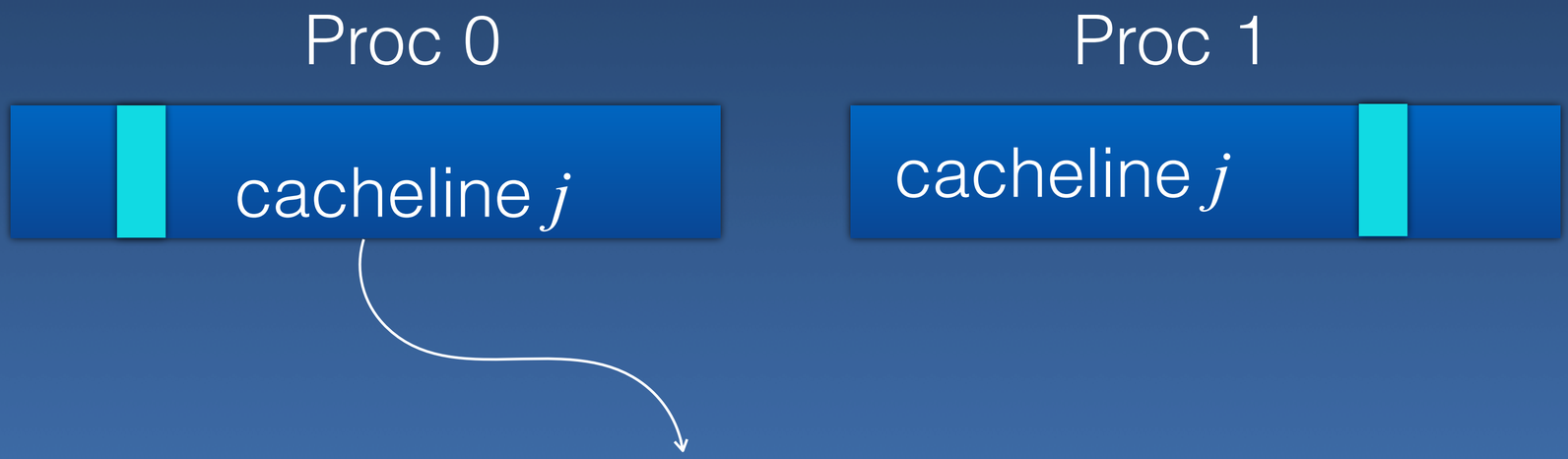


Multiple Caches

- Coherent caches
  - ➡ Writes in on cache are updated in all cache copies
- Caches operate on line granularity
  - ➡ To write one item, may need to update the other items first

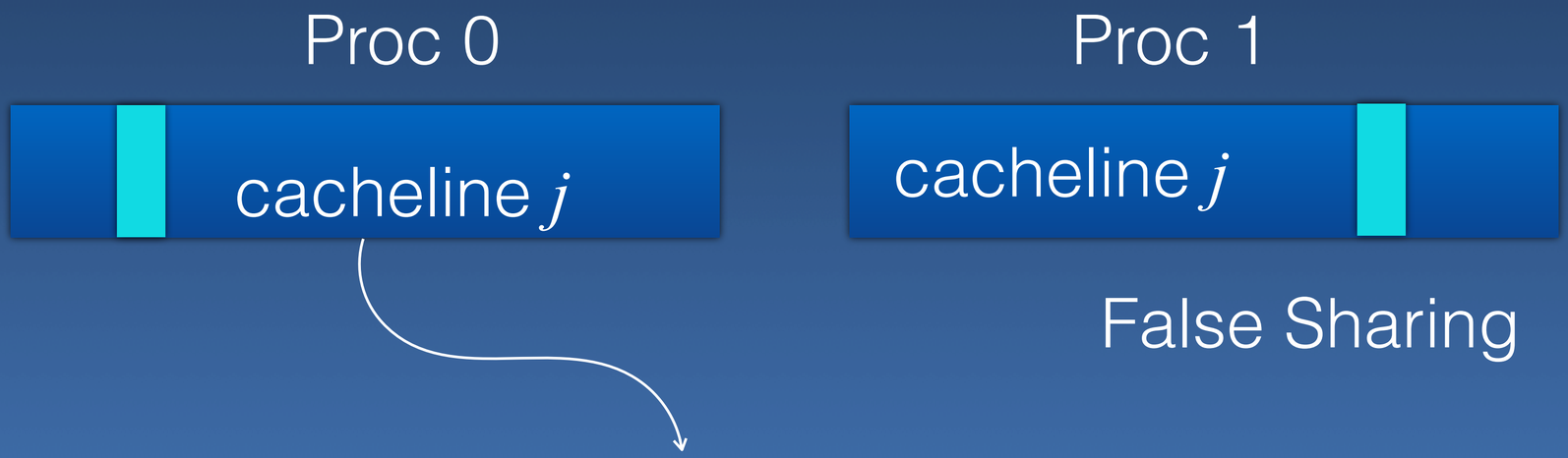
Multiple Caches

- Coherent caches
  - ➡ Writes in on cache are updated in all cache copies
- Caches operate on line granularity
  - ➡ To write one item, may need to update the other items first



Multiple Caches

- Coherent caches
  - ➡ Writes in on cache are updated in all cache copies
- Caches operate on line granularity
  - ➡ To write one item, may need to update the other items first



Parallelize Version 0

- Compiler flag
  - gcc -O2 -ftree-vectorize ### -mavx512f -fopt-info-vec-optimized
  - Or, gcc -O3
- Embarrassingly Parallel
  - Subdivide work in p equal parts, given p processors
  - forall i in [0,p)
    - Do Workpart(i)

Parallelize Version 0

- Compiler flag
  - gcc -O2 -ftree-vectorize ### -mavx512f -fopt-info-vec-optimized
  - Or, gcc -O3
- Embarrassingly Parallel
  - Subdivide work in p equal parts, given p proce
  - forall i in [0,p)
    - Do Workpart(i)

```
make -j

#!/bin/bash
for ((i=0; i< $count; i++));
do
      grep $pattern file$i &
done
```

Can files be read in parallel?

Hello OpenMP

```
#include <omp.h>

#pragma omp parallel // num_threads(8)
{
    tid = omp_get_thread_num();
    dowork(input, tid);
}
```

```
> g++ file.c -fopenmp
```

Sequential

OP operands
OP operands
OP operands
OP operands
OP operands
OP operands
OP operands
OP operands

Parallel

OP operands	OP operands
OP operands	OP operands
OP operands	OP operands
OP operands	OP operands
OP operands	OP operands
OP operands	OP operands
OP operands	OP operands
OP operands	OP operands

Threads of Execution: Instructions executed in order

Subodh Kumar

Sequential	Parallel	
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands
OP operands	OP operands	OP operands

Threads of Execution: Instructions executed in order

Parallel & Distributed

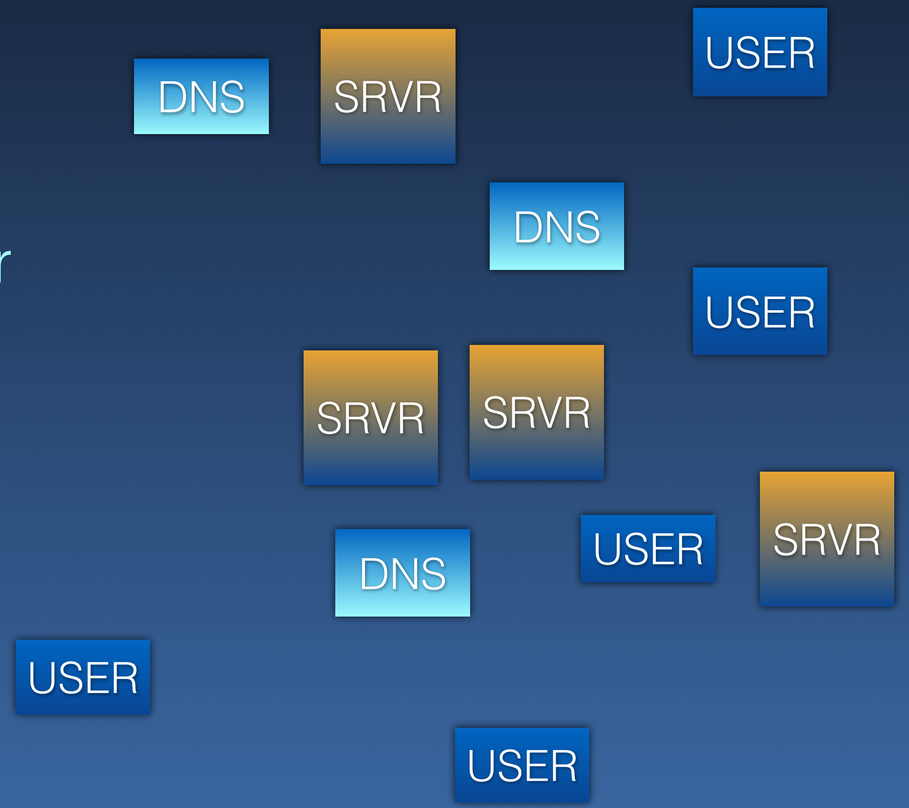
- Parallel:
  - ➡ Focus on doing many things at the same time
- Distributed:
  - ➡ How the multiple things interact with each other

Parallel & Distributed

- Parallel:
  - ➡ Focus on doing many things at the same time
- Distributed:
  - ➡ How the multiple things interact with each other
- Concurrency
  - ➡ Unordered

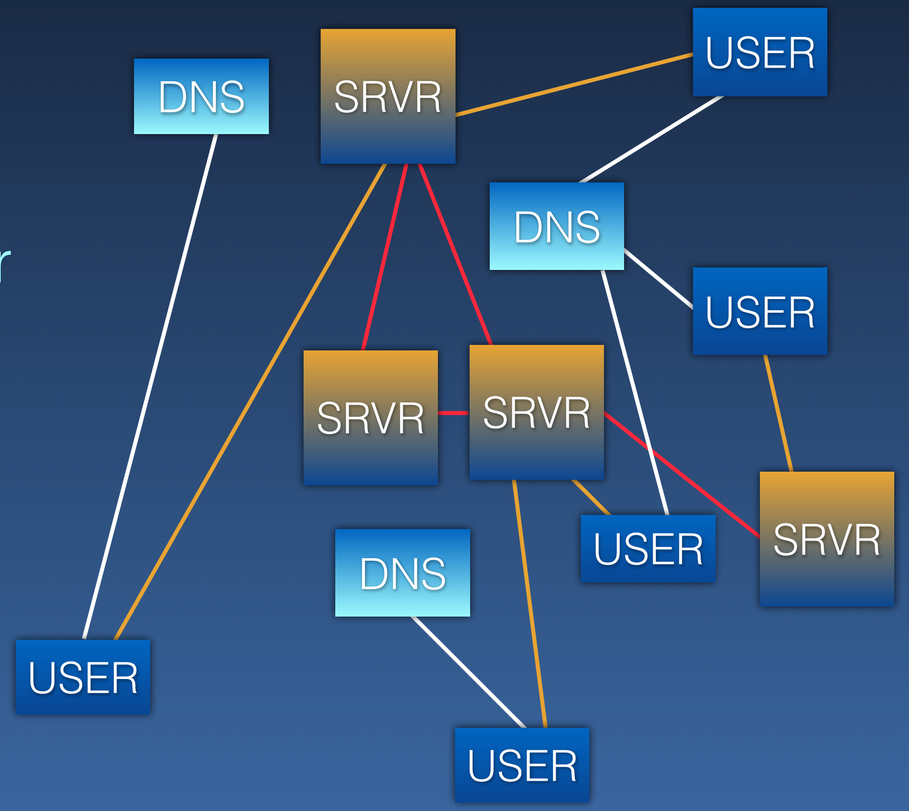
Parallel & Distributed

- Parallel:
  - ➡ Focus on doing many things at the same time
- Distributed:
  - ➡ How the multiple things interact with each other
- Concurrency
  - ➡ Unordered



Parallel & Distributed

- Parallel:
  - ➡ Focus on doing many things at the same time
- Distributed:
  - ➡ How the multiple things interact with each other
- Concurrency
  - ➡ Unordered



Review

- Cache behavior, false sharing
- Flavors of parallelism
- Memory bottleneck
- Intro to architecture