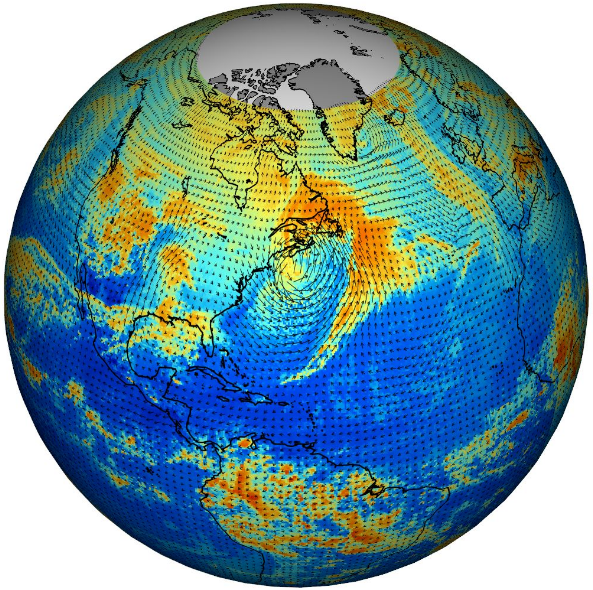
Introduction to Parallel Programming

By Natalia Frumkin & Vijay Thakkar

Para… what?

Climate Simulation Model



- Modeling the exchange of matter + energy over time
  - Wind and water currents
  - Atmospheric pressure
  - Weather tracking
- How is it done?
  - Discretize globe into fine grid
  - Equations characterize energy movement
  - *Solving equations repeatedly*

→ Highly parallel task

Problem: "Too Much Computation"

Running a simulation in series from start to finish would take forever!

Notice: These individual elements can be done concurrently! The operation is the same!

Idea: Parallelize operations for all elements

Big Idea: Parallelization

1. **Decompose this problem into smaller parts**

Discretize the Grid

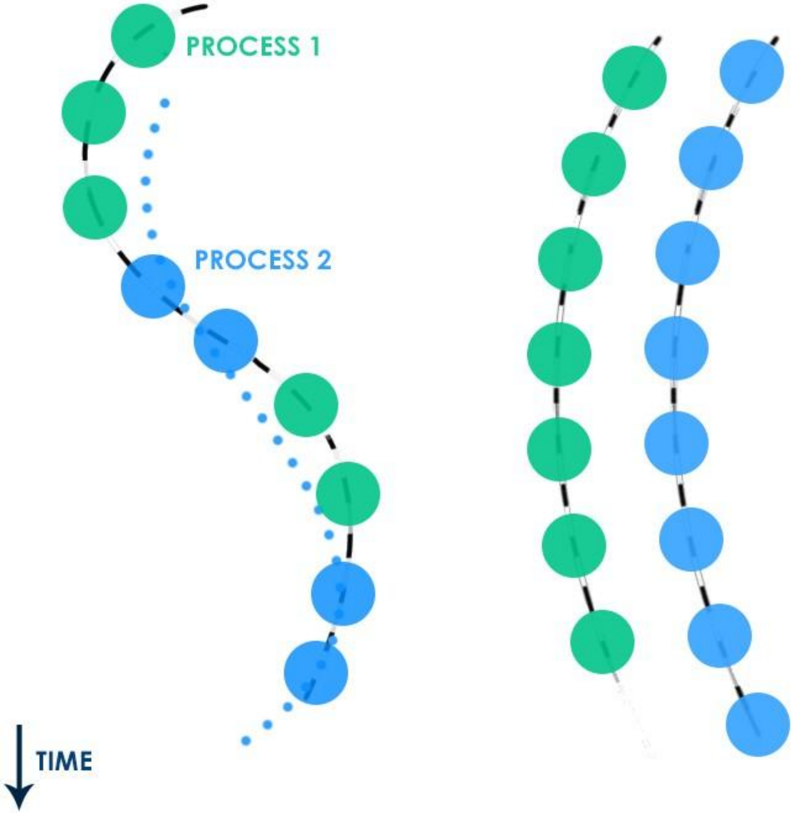
2. Do those on multiple concurrent processing streams

Separate system model into local and global phenomena

3. Run these concurrent streams in parallel

Spread Compute across multiple processors

Concurrent vs. Parallel



Interrupts

Simultaneous

Consider Single Precision ax + y:

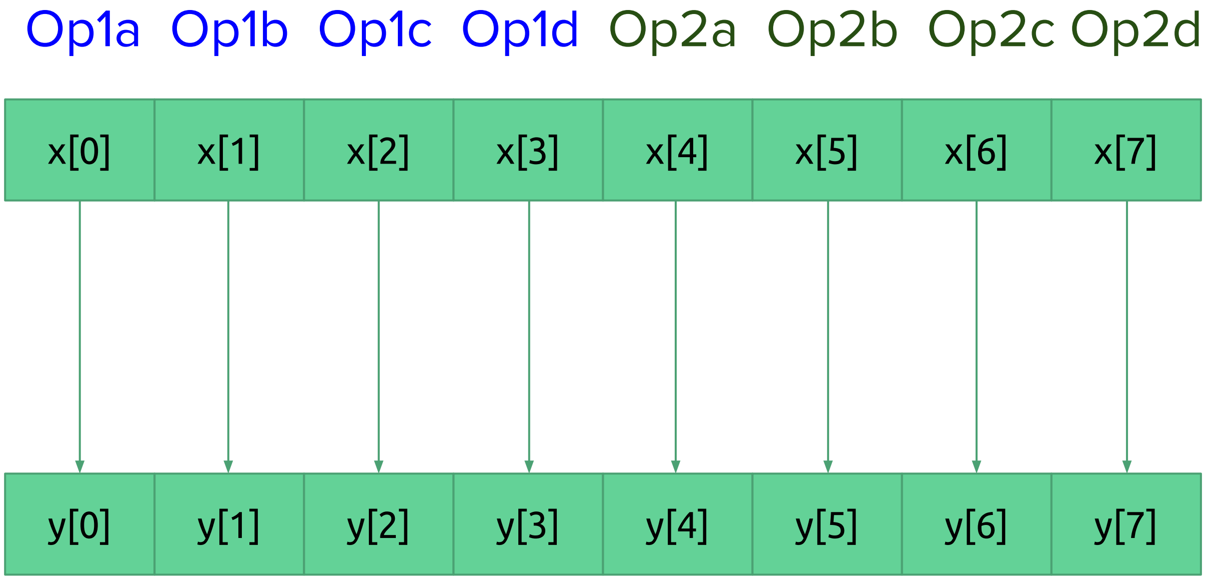
```
void saxpy(int n, float a, float * restrict x, float * restrict y)
{
  for (int i = 0; i < n; ++i)
      y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy(1<<20, 2.0, x, y);
```

Serially...

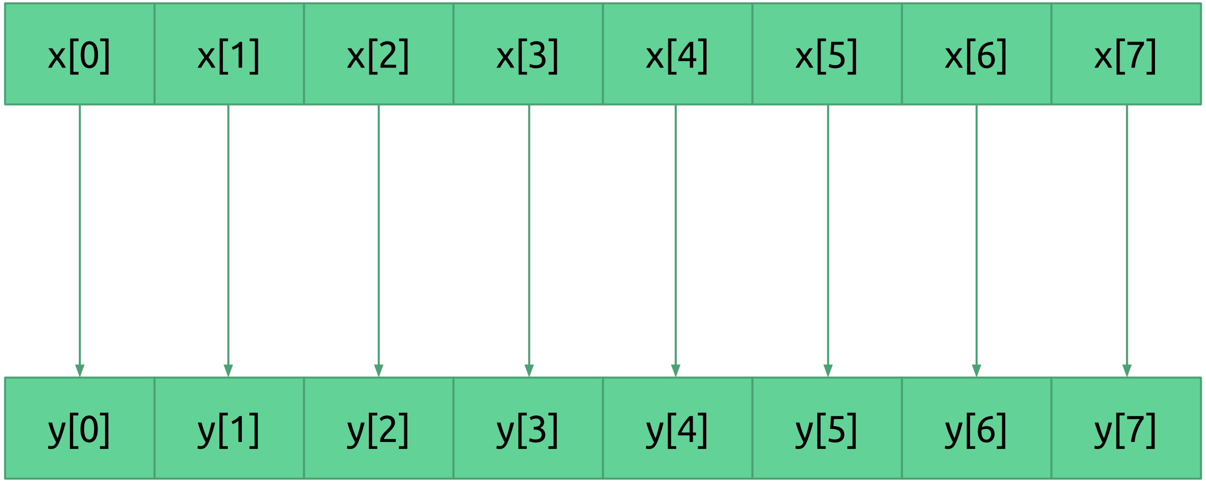

Op1 Op2 Op3 Op4 Op5 Op6 Op7 Op8
x[0] x[1] x[2] x[3] x[4] x[5] x[6] x[7]
y[0] y[1] y[2] y[3] y[4] y[5] y[6] y[7]

Concurrently...

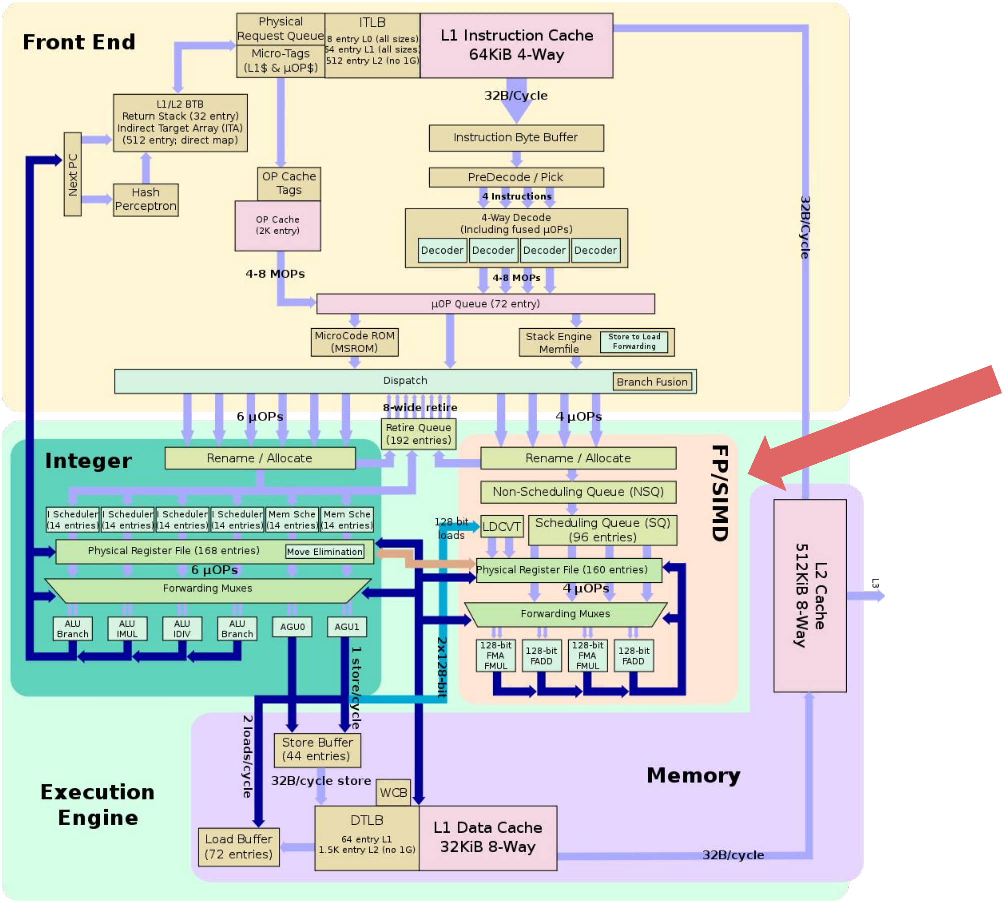


In Parallel...

Each operation assigned to a different execution unit



Hardware Approach: Vectorization



Hardware Approach: Vectorization

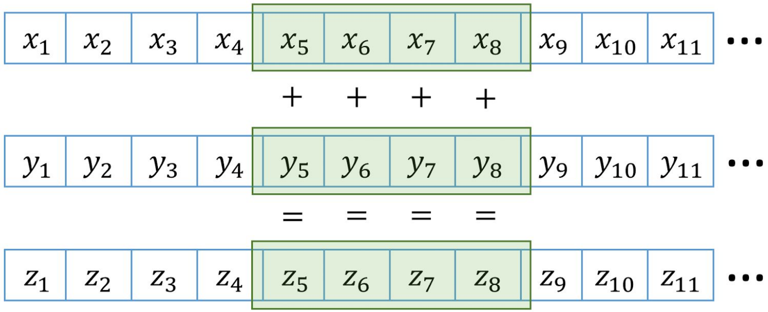
Fine Grained Parallelism:

Do the same math operation on multiple elements in the same instruction

SIMD:

Single Instruction - Multiple Data

aka. Vectorization



Case Study: AVX Extensions

```c
void saxpy_avx256(size_t n, float a, float *x, float *y, float *z) {
    __m256 a_vec = _mm256_set1_ps(a);

    for (size_t i = 0; i < n; i += sizeof(__m256) / sizeof(float)) {
        __m256 x_vec = _mm256_load_ps(&x[i]);
        __m256 y_vec = _mm256_load_ps(&y[i]);
        __m256 res_vec = _mm256_add_ps(_mm256_mul_ps(a_vec, x_vec), y_vec);
        _mm256_store_ps(&z[i], res_vec);
    }
}
```

Case study: OpenMP `#pragma omp simd`

```c

void saxpy_omp(size_t n, float a, float *__restrict__ x, float *__restrict__ y,
               float *__restrict__ z) {
#pragma omp simd aligned(x, y, z : 64)
  for (size_t i = 0; i < n; i++) {
    z[i] = (a * x[i]) + y[i];
  }
}
```

Case study: GPU parallel code

```
__global__ void saxpy_cuda(size_t n, float a, float *x, float *y, float *z) {
  int idx = (blockIdx.x * blockDim.x) + threadIdx.x;
  if (idx < n)
    z[idx] = (a * x[idx]) + y[idx];
}
```

```
  cudaMalloc(&d_x, n * sizeof(float));
  cudaMalloc(&d_y, n * sizeof(float));
  cudaMalloc(&d_z, n * sizeof(float));
  cudaMemcpy(d_x, x, n * sizeof(float), cudaMemcpyHostToDevice);
  cudaMemcpy(d_y, y, n * sizeof(float), cudaMemcpyHostToDevice);
  cudaMemcpy(d_z, z, n * sizeof(float), cudaMemcpyHostToDevice);
  cudaDeviceSynchronize();

  auto start = high_resolution_clock::now();
  for (auto i = 0; i < itrs; i++) {
    saxpy_cuda<<<(n + 255) / 256, 256>>>(n, a, d_x, d_y, d_z);
    cudaDeviceSynchronize();
  }
  auto end = high_resolution_clock::now();
```

Reading List

- https://colfaxresearch.com/skl-avx512
- http://svmoore.pbworks.com/w/file/fetch/70583970/VectorOps.pdf
- https://software.intel.com/sites/landingpage/IntrinsicsGuide/
- https://www.youtube.com/playlist?list=PLAwxTw4SYaPnFKojVQrmyOGFCqHTxfdv2
- https://medium.com/@smallfishbigsea/basic-concepts-in-gpu-computing-3388710e9239
- https://devblogs.nvidia.com/easy-introduction-cuda-c-and-c/
- https://devblogs.nvidia.com/six-ways-saxpy/