Introduction to Parallel & Distributed Programming

Lec 03—Software Parallelism

Subodh Sharma | Jan 09, 2026

Recap

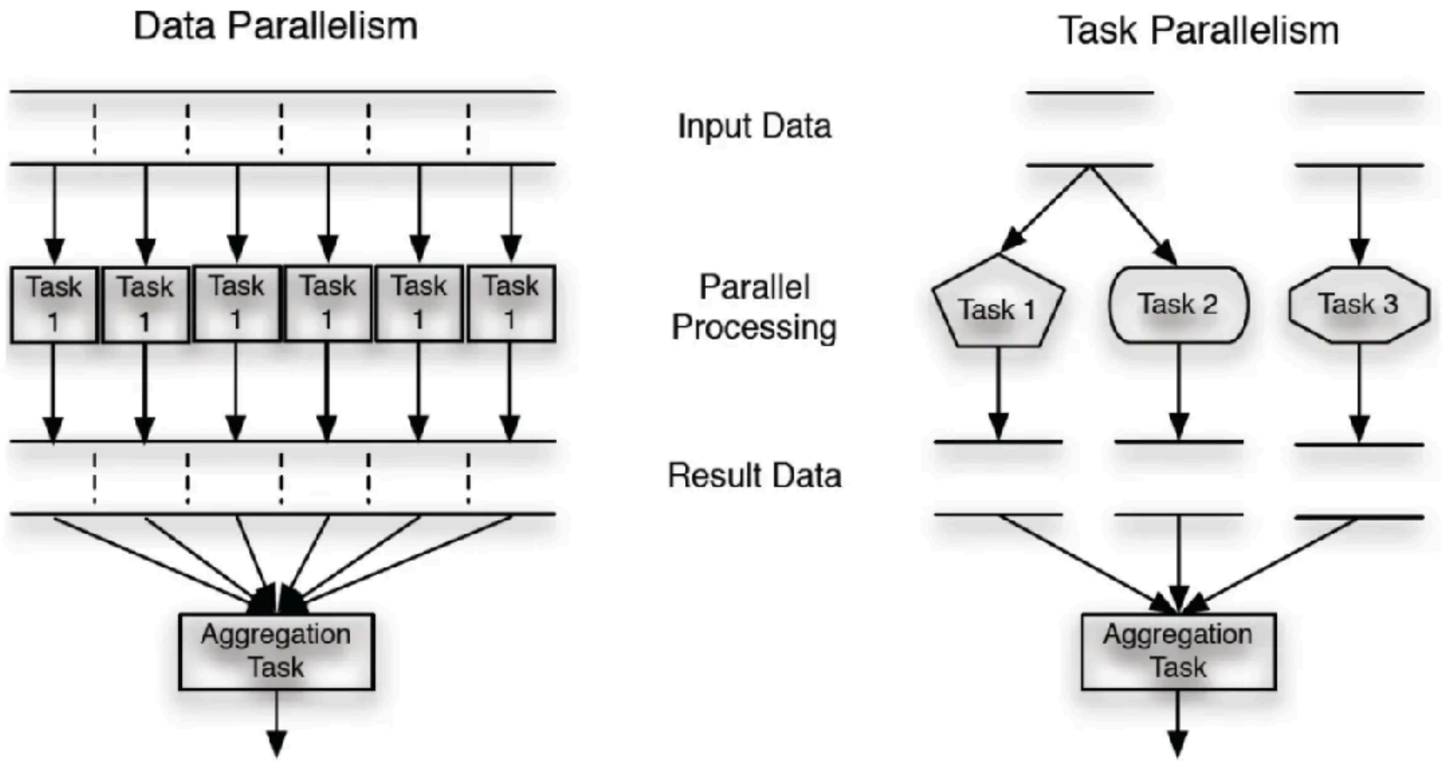
- OS, Architectural Details
- Caches
  - UMA vs NUMA
  - TLB
  - Coherence & Granularity
  - False Sharing

What is a Parallel Program?

- A collection of computations written as a single program but with different labelled regions
  - Each region is viewed as **an instr stream** that can be mapped to a physical core
- **Major models of parallelism**
  - **SIMD:** Same operation on different data (**data parallelism**)
    - **SIMT:** Single operation but multiple threads (GPU-based concurrency)
  - **MIMD**: Multiple operations on different subsets of data (**task parallelism**)
- **Major models of communication (or synchronisation)**
  - Shared Memory, Message Passing

Data vs Task Parallelism

- **Types of Parallelism in software:**



Writing the first Parallel Program

Data Parallelism

- Let the computation be $f(x) = sin\, x + cos\, x + \sqrt{x^2 + 1}$
  - Artificially heavy to showcase the benefits of parallelism
  - How is it data parallel computation?

```
#pragma omp parallel for
    for (int i = 0; i < N; i++) B[i] =f(A[I]);
```

Let us look at the program!

Data Parallelism Example: Serial Version

```
static inline double heavy(double x) {
    for (int k = 0; k < 50; k++) {
        x = sin(x) + cos(x) + sqrt(x*x + 1.0);
    }
    return x;
}
```

Function

```
int main() {

  const int N = 1000 * 1000;   // 1 million

  // 1) Serial

  double t1 = omp_get_wtime();
  for (int i = 0; i < N; i++) B1[i] = heavy(A[i]);
  double t2 = omp_get_wtime();
  double serial = t2 - t1;
```

Data Parallelism Example: Parallel Version

```
static inline double heavy(double x) {
    for (int k = 0; k < 50; k++) {
        x = sin(x) + cos(x) + sqrt(x*x + 1.0);
    }
    return x;
}
```

Function

```
int main() {

  const int N = 1000 * 1000;    // 1 million

   t1 = omp_get_wtime();
#pragma omp parallel for
   for (int i = 0; i < N; i++) B2[i] = heavy(A[i]);
   t2 = omp_get_wtime();
   double parallel = t2 - t1;
```

OMP pragma to parallelise loop iterations