

Understanding parallelism

SpeedUp vs Efficiency

SpeedUp (S_p): relative reduction of execution time whn using P processors with respect sequential.

Efficiency (Eff_p): it is a measure of the fraction of time for which processing element is usefull.

Escalability

- Strong: resources x2 -> scalability x2
- Weaak: resources x2 w. proportional work

Amdahl's law

Par_Fraction

$$\varphi = T_{\text{seq_time_of_par_part}} / T_{\text{seq_exec}}$$

$$S_p = \frac{T_1}{T_p} = \frac{T_1}{(1 - \varphi) \times T_1 + (\varphi \times T_1 / P)}$$

$$S_p = \frac{1}{((1 - \varphi) + \varphi / P)}$$

Note: If P approach to infinit, φ/P approach to 0, then $S_p = 1/(1-\varphi)$.

Ex:

```
seq - 25s
par - 50s
seq - 25s
100s
```

$$\varphi = 100/50 = 0,5 \quad \text{SpeedUp}_{\text{par}} = 50/10 = 5$$

$$\text{SpeedUp} = 100/60 = 1.67$$

Sources of overhead

- task creation
- barrier sync
- task sync
- exclusive access to data
- data sharing
- Idleness
- Computation (extra work to obtain a parallel algorithm)
- Memory (extra memory to obtain a parallel algorithm)
- Contention (competition for the access to shared resources)

$$T_p = (1 - \varphi) \times T_1 + \varphi \times T_1/p + \textit{overhead}$$

How to model data sharing overload?

Example:

Jacobi solver

$$T_{\text{calc}} = (N^2/P) \times t_{\text{body}}$$

$$T_p = T_{\text{calc}} + T_{\text{comm}}$$

$$T_{\text{comm}} = 2(t_s + t_w \times N)$$

Data sharing modeling

Example 4

```
void compute(int n, double *u, double *utmp) {
    int i, j;
    double tmp;
    for (i = 1; i < n-1; i++) {
        for (j = 1; j < n-1; j++) {
            tmp = u[n*(i+1) + j] + u[n*(i-1) + j] + // elements u[i+1][j]
and u[i-1][j]
            u[n*i + (j+1)] + u[n*i + (j-1)] - // elements u[i][j+1] and
u[i][j-1]
            4 * u[n*i + j]; // element u[i][j]
            u[n*i + j] = tmp/4; // element u[i][j]
        }
    }
}
```

Each cpu with n^2/P elements. Tasks compute segments of n/c rows by c columns.

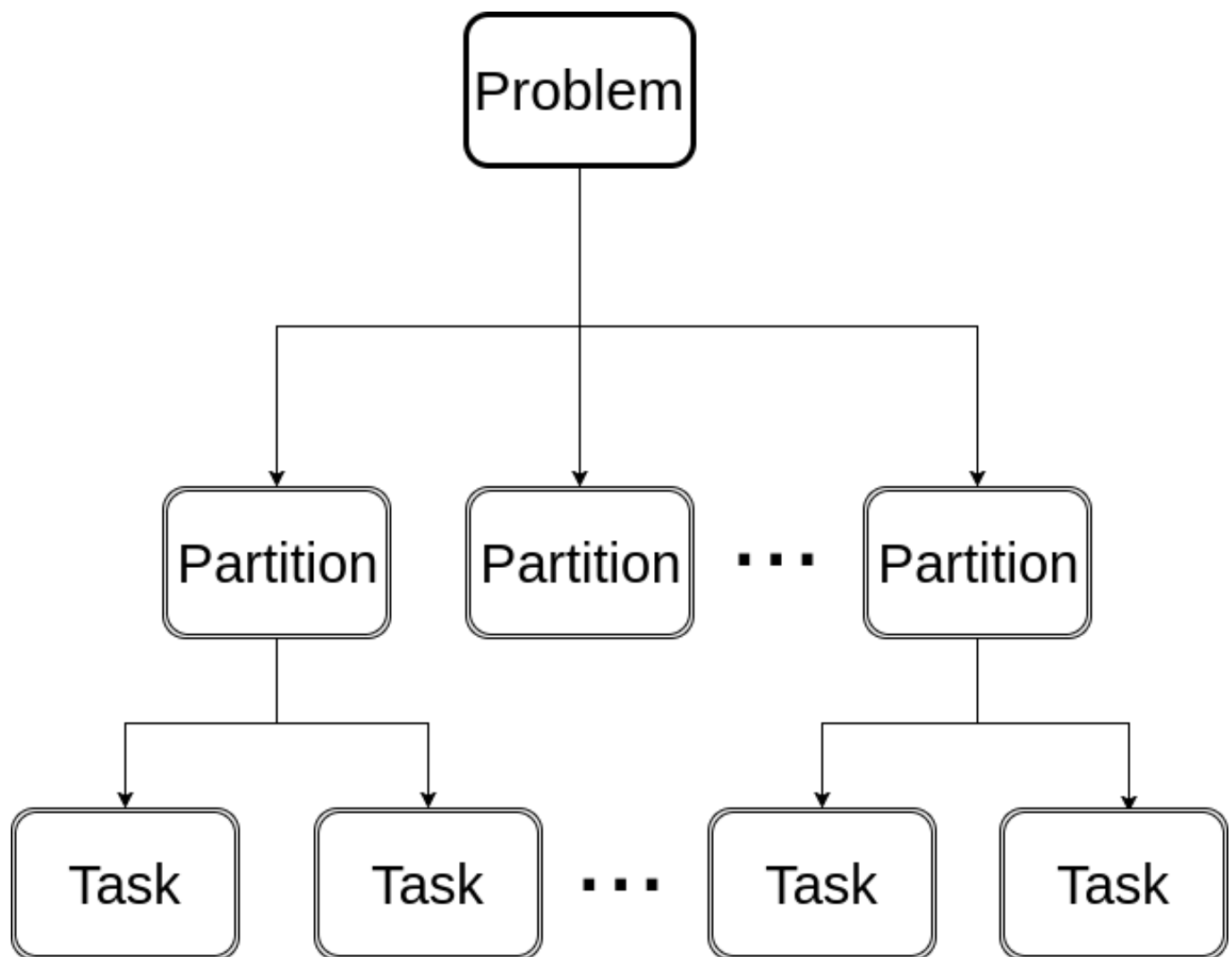
Then, time acquire the following form:

$$T_P = \left(\frac{n}{c} + P - 1\right) \times \left(\frac{n}{P} \times c\right) \times t_{body} +$$

$$(t_s + n \times t_w) + \left(\left(\frac{n}{c} + P - 2\right) \times (t_s + c \times t_w)\right)$$

$$S_P = \frac{T_1}{T_P} = \frac{n^2 \times t_{body}}{T_P}$$

Task decomposition



Types:

- Lineal Task decomposition

Code block or procedure

- Iterative task decomposition

Iterative constructs

- Recursive task decomposition

Recursive procedures

Decomposition Strategies

Leaf strategy

Create one task sequentially for each leaf of task tree.

Less tasks
Less overhead

Tree strategy

Creates one task for each invocation.

More Tasks
More Overhead

Cut-off control

If tree strategy is in use, when a certain number of tasks are created or the granulation is too small or in a certain number of recursive calls; you can change the strategy to Leaf in order to reduce overhead.

Task ordering constraints

Dependences

Constraints in the parallel execution of tasks

- Task ordering constraints

They force the execution of tasks in a required order

- Data sharing constraints

They force the access to data to fulfil certain properties.