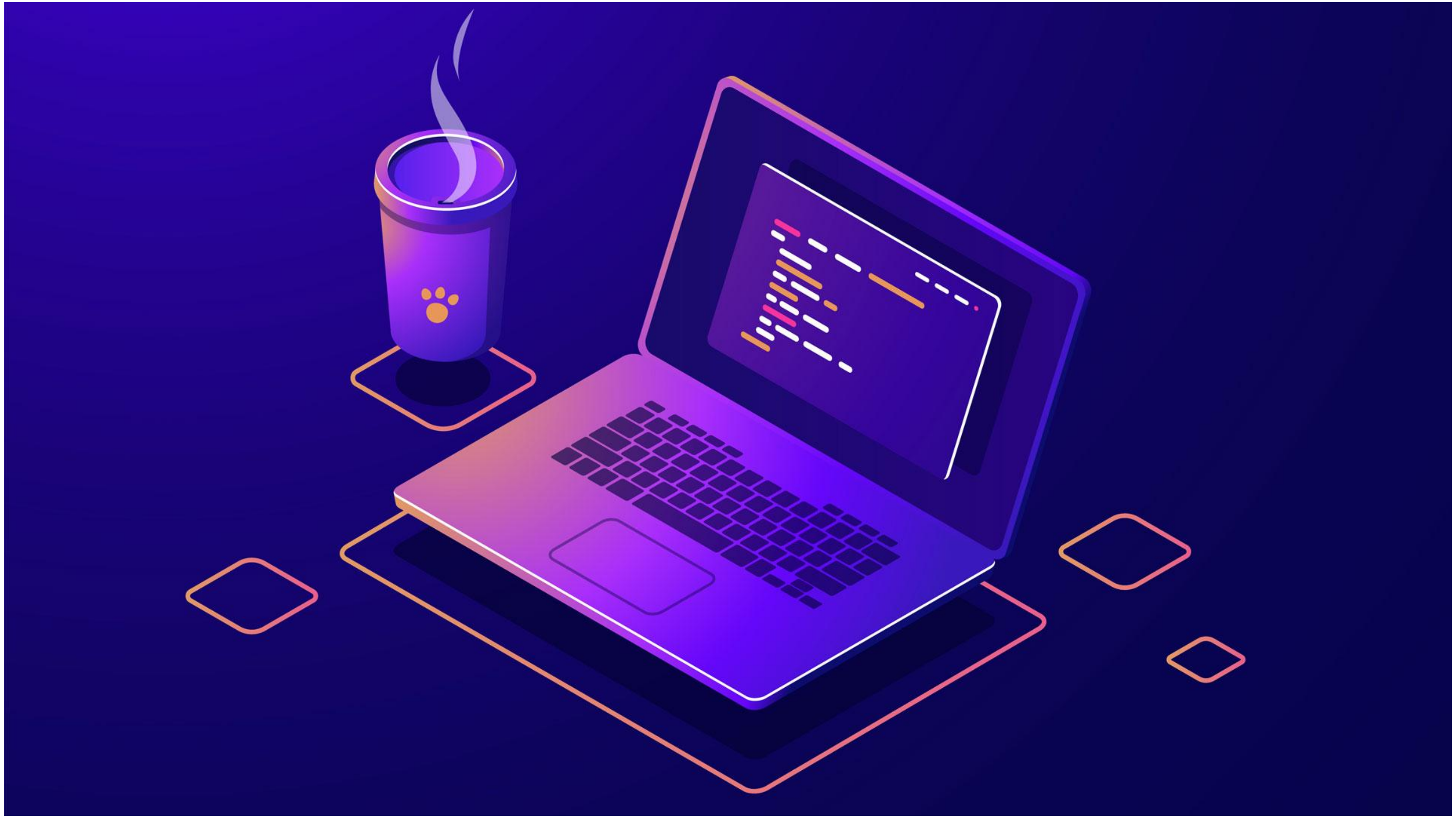
Advanced Granularity



Advanced Granularity

Given a sequential program, the number of tasks that one can generate and the size of the tasks (what is called **granularity**) are related one to the other.

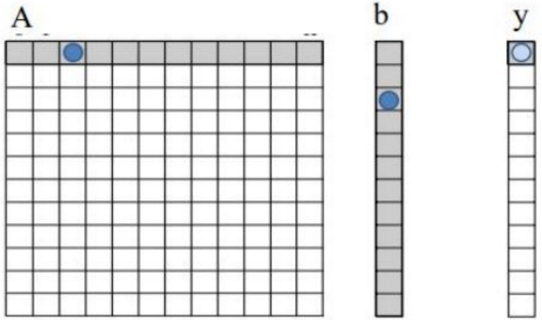
- Fine-grained tasks vs. coarse-grained tasks
- The parallelism increases as the decomposition becomes finer in granularity (small tasks) and vice versa

Example 1: matrix-vector product

Fine-grained Decomposition

Example: matrix-vector product (n by n matrix):

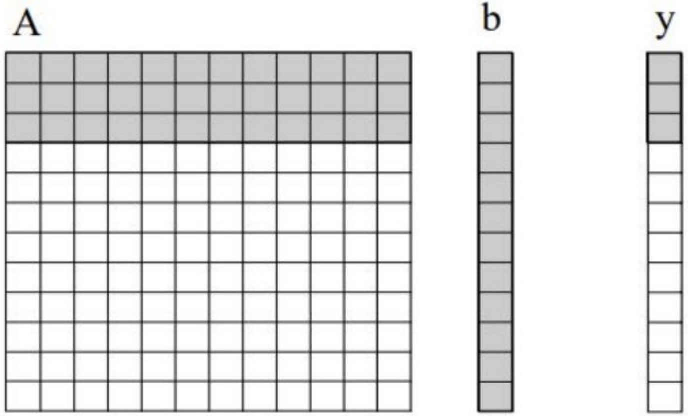
- A task could be each individual $\times$ and $+$ in the dot product that contributes to the computation of an element of y $(y[i] = y[i] + A[i][j] * b[j])$



- A task could also be each complete dot product to compute an element of y $\left(y[i] = y[i] + \sum_{j=1}^{j=n}(A[i][j] * b[j])\right)$

Coarse-grained Decomposition

- A task could be in charge of computing a number of consecutive elements of y (e.g. three elements)



- A task could be in charge of computing the whole vector y

Lucas Bazilio - Udemy

So…

- It would appear that the parallel time can be made arbitrarily small by making the decomposition finer in granularity but…
  - Inherent bound on how fine the granularity of a computation can be
    - *e.g. matrix-vector multiply:* (n^2) *concurrent tasks.*
  - Tradeoff between the granularity of a decomposition and associated overheads (sources of overhead: creation of tasks, task synchronization, exchange of data between tasks, ...)
  - The granularity may determine performance bounds

Example 2: stencil computation using Jacobi solver

Stencil algorithm that computes each element of matrix `utmp` using 4 neighbor elements of matrix `u`, both matrices with $n \times n$ elements

```
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]
            utmp[n*i + j] = tmp / 4;                // element utmp[i][j]
        }
    }
}
```

Example 2: stencil computation using Jacobi solver

What tasks can be? Assume: 1) the innermost loop body takes t_{body} time units; and 2) n is very large, so that $n - 2 \simeq n$

Task is ... (granularity)	Num. tasks	Task cost	T_1	T_∞	Parallelism
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	$n^2 \cdot t_{body}$	1
Each iteration of i loop	n	$n \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot t_{body}$	n
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{body}$	t_{body}	n^2
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$n^2 \cdot t_{body}$	$n \cdot r \cdot t_{body}$	$n \div r$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$n^2 \cdot t_{body}$	$c \cdot t_{body}$	$n^2 \div c$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$n^2 \cdot t_{body}$	$r \cdot c \cdot t_{body}$	$n^2 \div (r \cdot c)$

Finer grain task decomposition $\rightarrow$ higher parallelism, but ...

Example 2: stencil computation using Jacobi solver

... what if each task creation takes t_{create}?

Task is ... (granularity)	Num. tasks	Task cost	Task creation ovh
All iterations of i and j loops	1	$n^2 \cdot t_{body}$	t_{create}
Each iteration of i loop	n	$n \cdot t_{body}$	$n \cdot t_{create}$
Each iteration of j loop	n^2	t_{body}	$n^2 \cdot t_{create}$
r consecutive iterations of i loop	$n \div r$	$n \cdot r \cdot t_{body}$	$(n \div r) \cdot t_{create}$
c consecutive iterations of j loop	$n^2 \div c$	$c \cdot t_{body}$	$(n^2 \div c) \cdot t_{create}$
A block of r x c iterations of i and j, respectively	$n^2 \div (r \cdot c)$	$r \cdot c \cdot t_{body}$	$(n^2 \div (r \cdot c)) \cdot t_{create}$

Trade-off between task granularity and task creation overhead

Instructor Social Media

Youtube: Lucas Science

Instagram: lucaasbazilio

Twitter: lucasebazilio