First assignment

Cholesky factorization

Benatti Sebastiano, Carpi Samuele,
Pasquali Mattia

Cholesky decomposition factors a symmetric, positive-definite matrix **A** into

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

where **L is a lower triangular matrix and L^T is the transpose of L.** It provides a numerically efficient way to solve linear systems and is widely used in optimization, statistics, and numerical analysis.

Diagonal elements

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

Off-diagonal elements

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$$

```
int i, j, k;
DATA_TYPE x;
for (i = 0; i < _PB_N; ++i)
{
  x = A[i][i];
  for (j = 0; j <= i - 1; ++j){
    x -= A[i][j] * A[i][j];
  }
  p[i] = 1.0 / sqrt(x);

  for (j = i + 1; j < _PB_N; ++j)
  {
    x = A[i][j];
    for (k = 0; k <= i - 1; ++k)
      x -= A[j][k] * A[i][k];
    A[j][i] = x * p[i];
  }
}
```

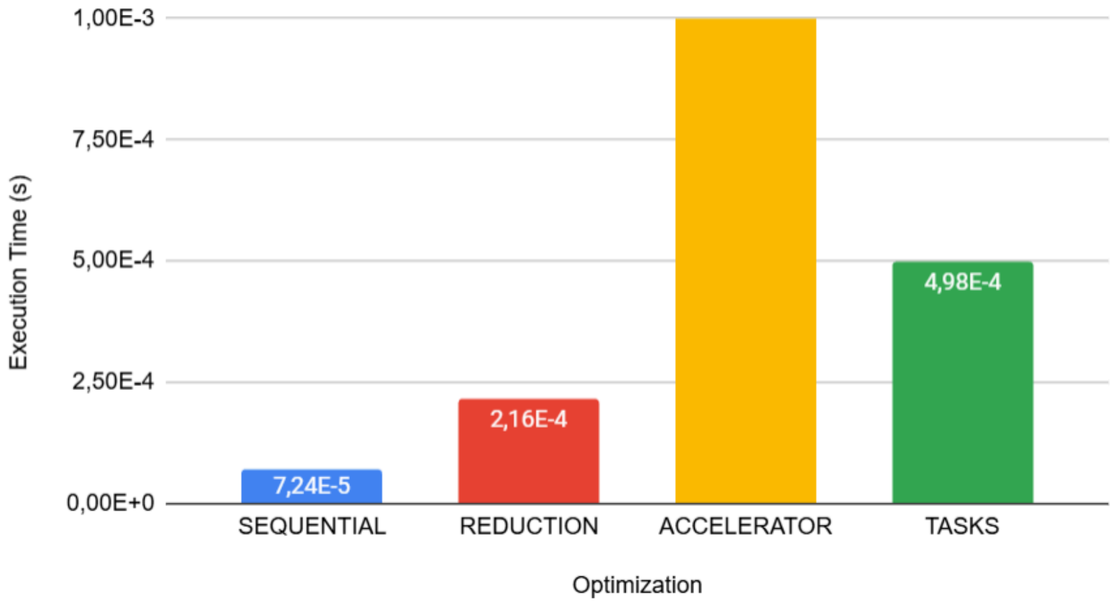
PARALLELIZATION APPROACHES

We tried to use 3 different optimization approaches:

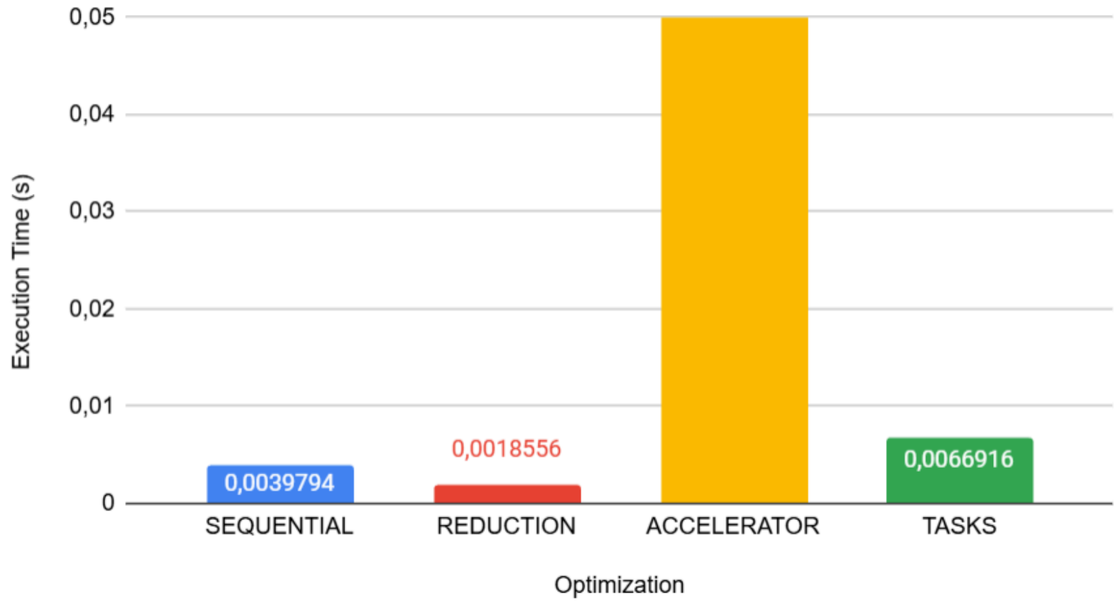
1. Data Parallelism (Reduction)
2. Accelerator Offloading
3. Task Parallelism

EXPERIMENTAL RESULTS

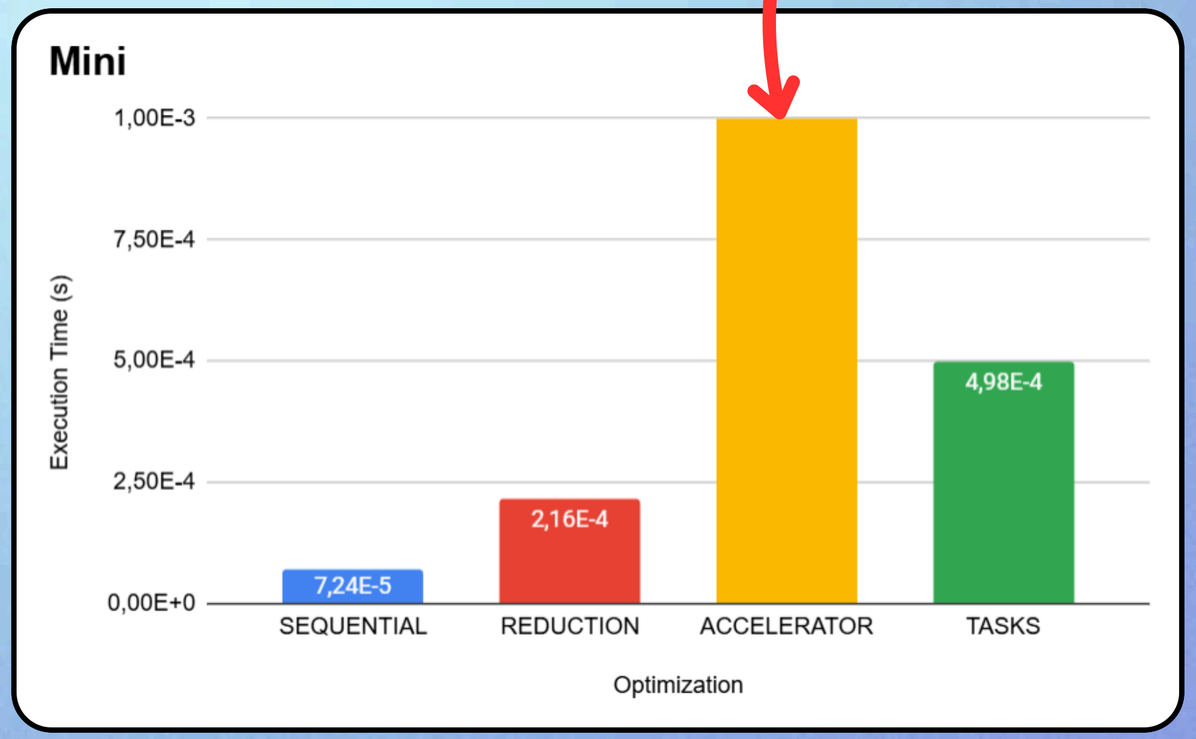
Mini

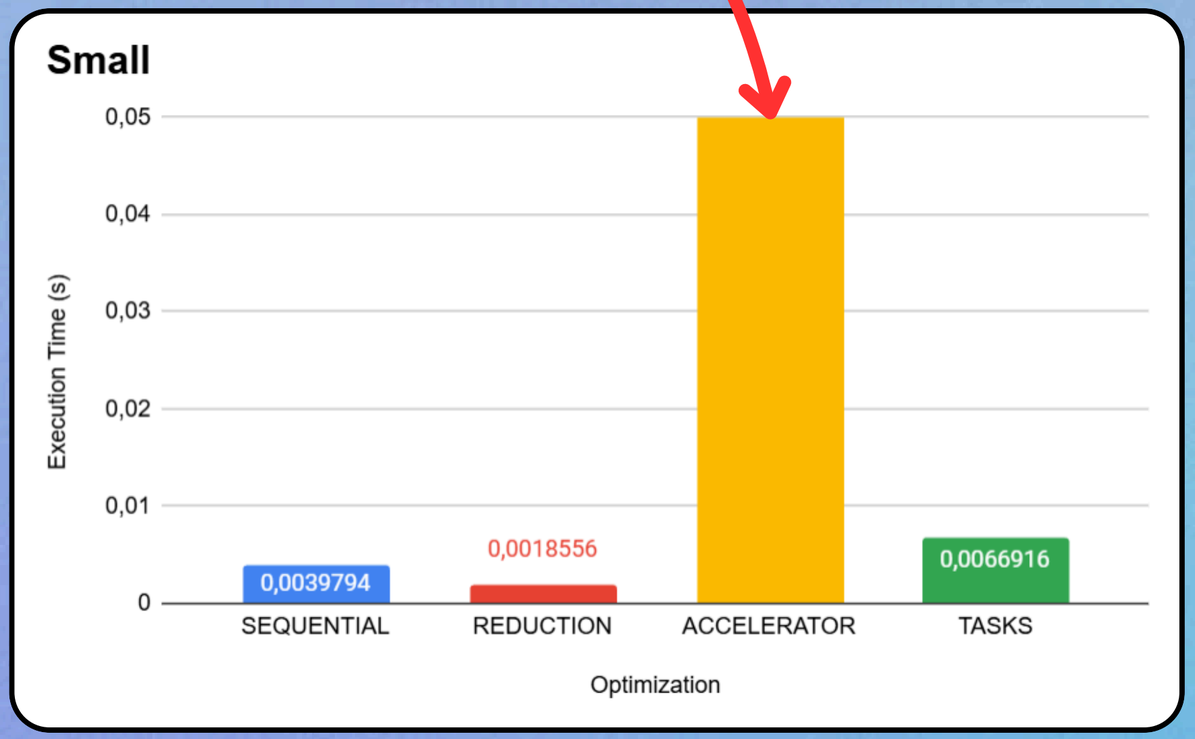

1,00E-3
7,50E-4
5,00E-4
2,50E-4
0,00E+0
Execution Time (s)
7,24E-5
2,16E-4
4,98E-4
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization

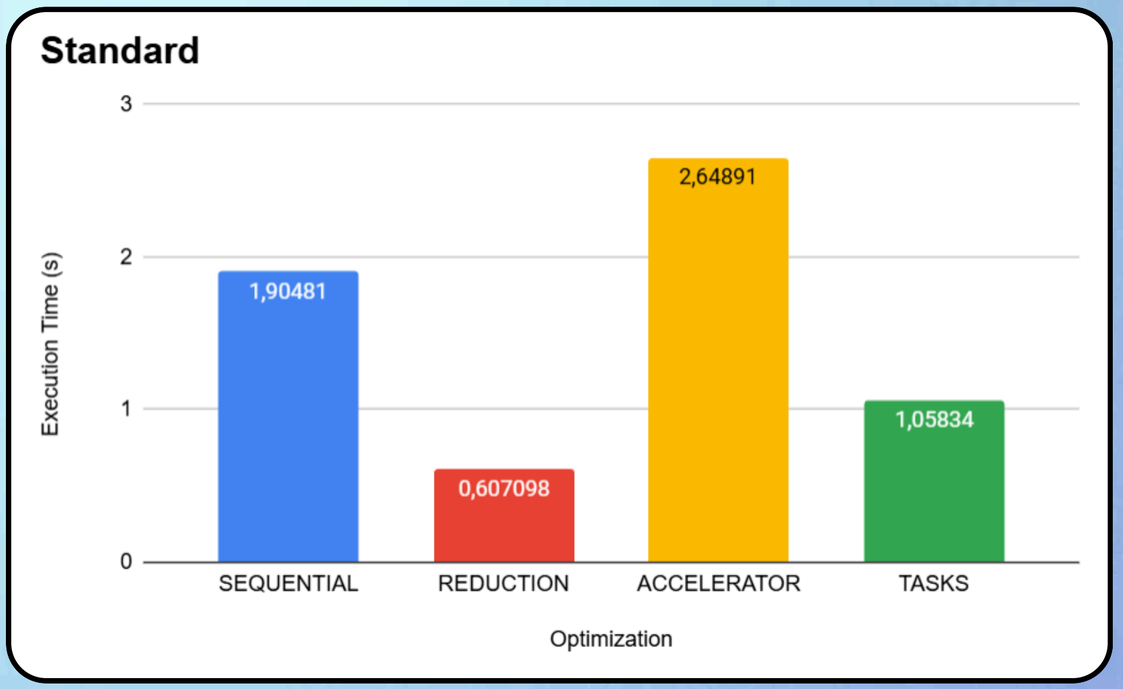
Small

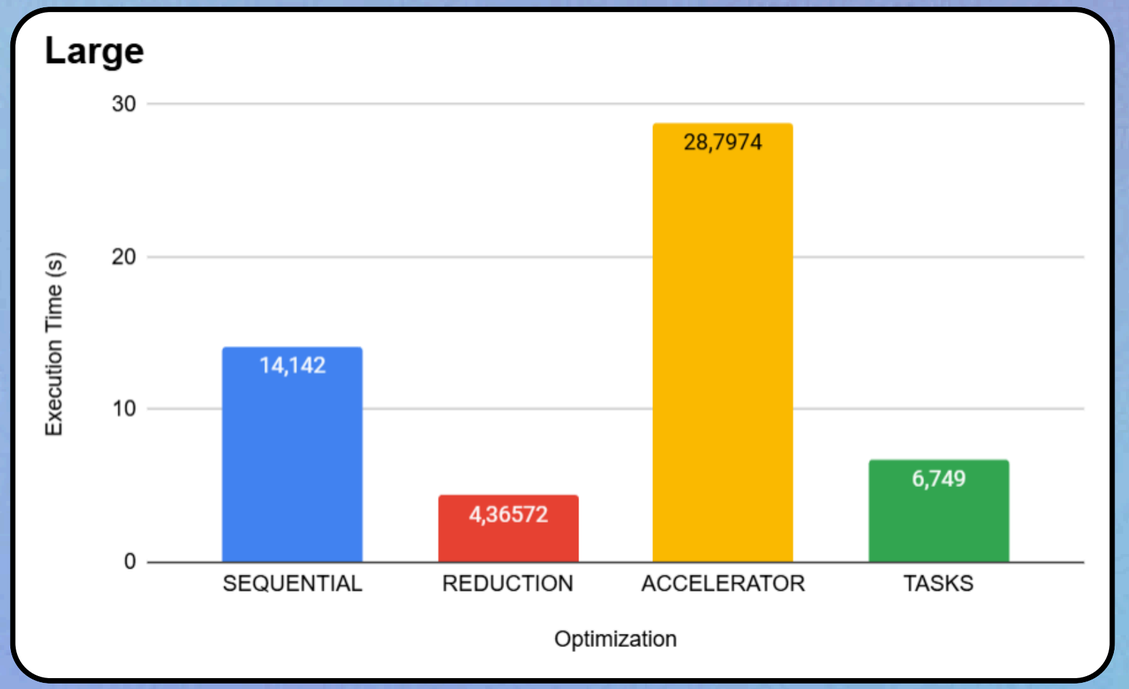

0,05
0,04
0,03
0,02
0,01
0
Execution Time (s)
0,0039794
0,0018556
0,0066916
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization

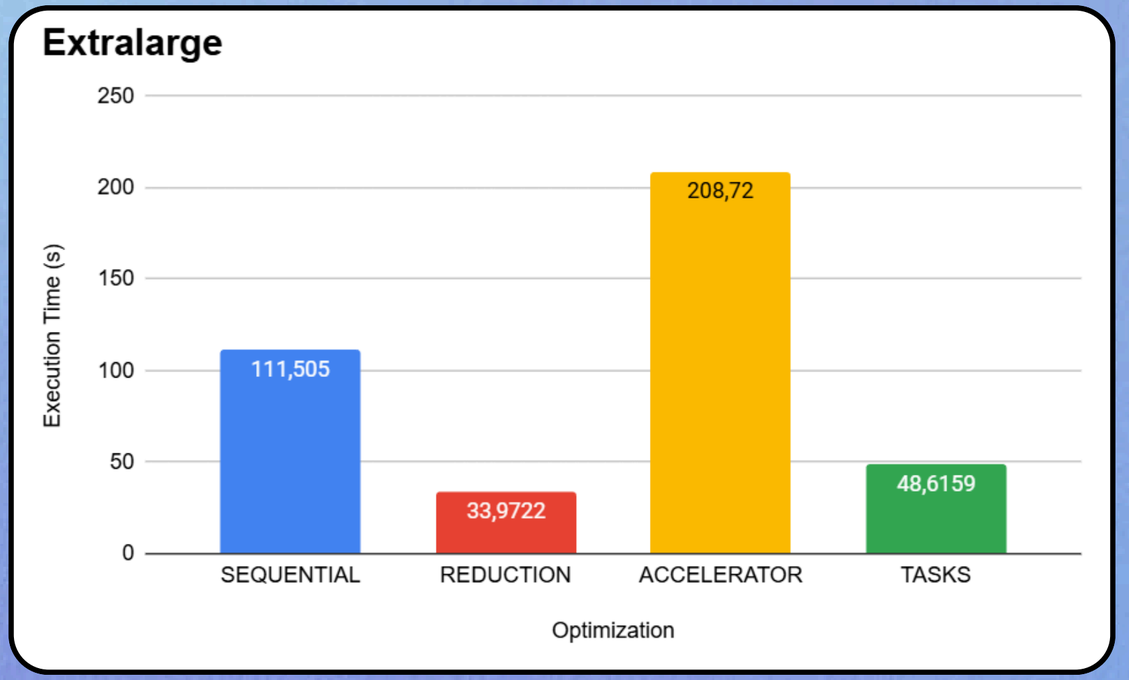
OUT OF SCALE !!! 😱


Mini
1,00E-3
7,50E-4
5,00E-4
2,50E-4
0,00E+0
Execution Time (s)
7,24E-5
2,16E-4
4,98E-4
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization

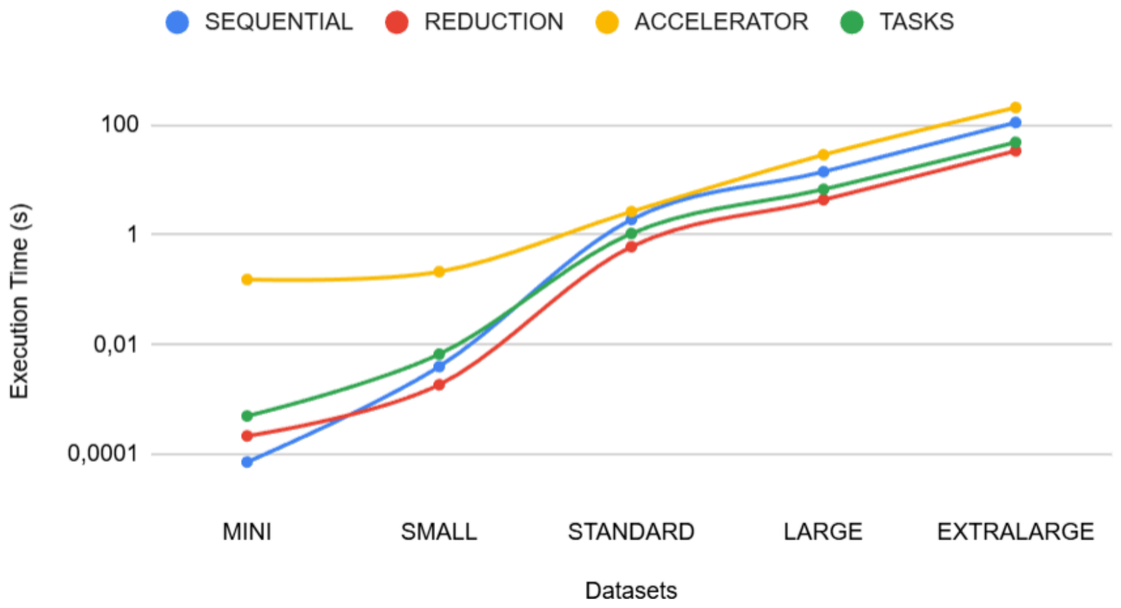

Small
0,05
0,04
0,03
0,02
0,01
0
Execution Time (s)
0,0039794
0,0018556
0,0066916
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization


Standard
Execution Time (s)
3
2
1
0
1,90481
0,607098
2,64891
1,05834
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization

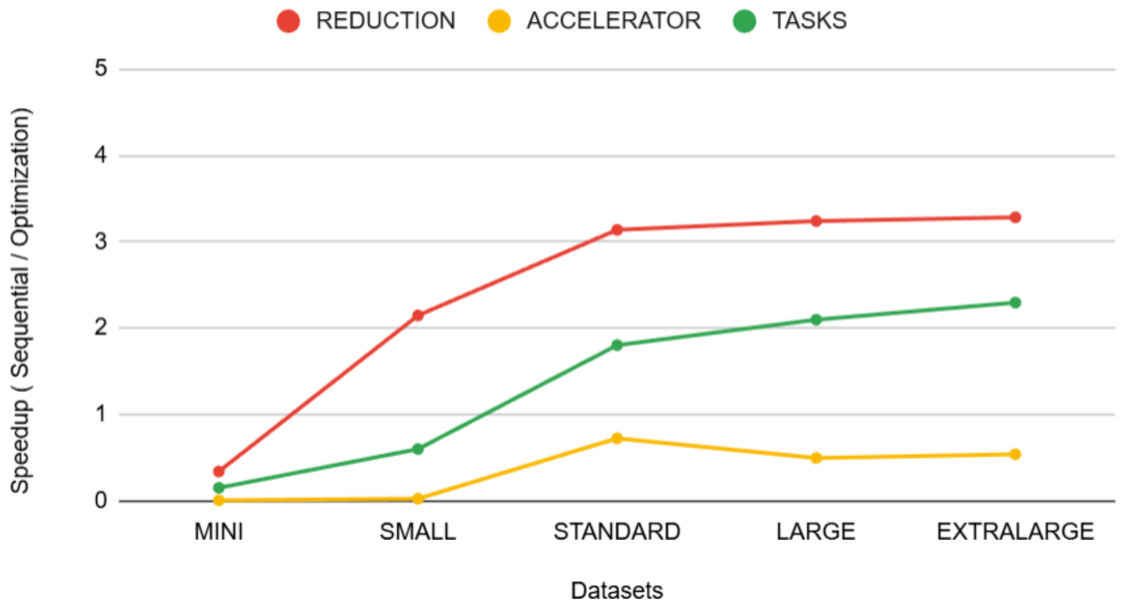

Large
Execution Time (s)
30
20
10
0
14,142
4,36572
28,7974
6,749
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization


Extralarge
Execution Time (s)
250
200
150
100
50
0
111,505
33,9722
208,72
48,6159
SEQUENTIAL
REDUCTION
ACCELERATOR
TASKS
Optimization

Execution Times



Speedup



SOLUTION
The code

```
static void kernel_cholesky(int n,DATA_TYPE POLYBENCH_1D(p, N, n),DATA_TYPE POLYBENCH_2D(A, N, N, n, n))
{
  int i, j, k;
  DATA_TYPE x;
  for (i = 0; i < _PB_N; ++i)
  {
    x = A[i][i];
    #pragma omp parallel for reduction(-:x) schedule(static)
    for (j = 0; j <= i - 1; ++j){
      x -= A[i][j] * A[i][j];
    }
    p[i] = 1.0 / sqrt(x);
    #pragma omp parallel for private(k,x) schedule(static)
    for (j = i + 1; j < _PB_N; ++j)
    {
      x = A[i][j];
      for (k = 0; k <= i - 1; ++k)
        x -= A[j][k] * A[i][k];
      A[j][i] = x * p[i];
    }
  }
}
```

CONCLUSIONS

- The **Task** solution is, more or less, **similar** to the Sequential program (too much **data dependencies** between “for” cycles)
- The worst solution is the offload with the **Accelerator** (too much overhead exchanging data between memories and a **less calculation rather than data exchange**)
- The fastest solution is the **Reduction**