UNIVERSITÀ DEGLI STUDI FIRENZE

Parallel implementation of Edit Distance in Java and OpenMP

Bianucci Stefano

Firenze 05/09/2019

Project Goal

- **Implementing a sequential version of Edit Distance algorithm in Java**
- **Implementing a sequential version of Edit Distance algorithm in Cpp**

- **Implementing a parallel version using Java Threads**
- **Implementing a parallel version using OpenMP**

- **Compare the performance of the two versions on different dataset**

Edit Distance algorithm

The Edit Distance is a measure of dissimilarity between two strings. Formally, the edit distance between strings S and T is the minimum number of operations required to transform S to T.

The valid operations are:

- Insertion: ex: “ab” ⇒ “axb”
- Deletion: ex: “abc” ⇒ “ac”
- Substitution: ex: “abc” ⇒ “xbc”

Sequential implementation Java and Cpp classes

String "T"

String "S"	""	P	A	R	T
""	0	1	2	3	4
S	1	1	2	3	4
P	2	1	2	3	4
A	3	2	1	2	3
R	4	3	2	1	2
T	5	4	3	2	1
A	6	5	4	3	2
N	7	6	5	4	3

```java
public void calculate() {

    int i, j;
    for (i = 0; i < this.firstStringLength + 1; i++) {
        for (j = 0; j < this.secondStringLength + 1; j++) {
            if (i == 0 || j == 0) {
                this.editMatrix[i][j] = Math.max(i, j);
            } else {
                this.editMatrix[i][j] = min3(
                        this.editMatrix[i - 1][j] + 1,
                        this.editMatrix[i][j - 1] + 1,
                        this.editMatrix[i - 1][j - 1] + (this.firstString.charAt(i - 1) != this.secondString.charAt(j - 1) ? 1 : 0));
            }
        }
    }

    System.out.println();
    System.out.println("La distanza è " + this.editMatrix[this.firstStringLength][this.secondStringLength]);
}
```

```cpp
int calculate() {

    int i, j;
    for (i = 0; i < firstStringLength + 1; i++) {
        for (j = 0; j < secondStringLength + 1; j++) {
            if (i == 0 || j == 0) {
                editMatrix[i][j] = std::max(i, j);
            } else {
                editMatrix[i][j] = min3(
                        editMatrix[i - 1][j] + 1,
                        editMatrix[i][j - 1] + 1,
                        editMatrix[i - 1][j - 1] + (firstString.at(i - 1) != secondString.at(j - 1) ? 1 : 0));
            }
        }
    }

    cout << " ";
    cout << "La distanza è " << editMatrix[firstStringLength][secondStringLength];
    return 0;
}
```

Parallel implementation Java and OpenMP classes

String "T"

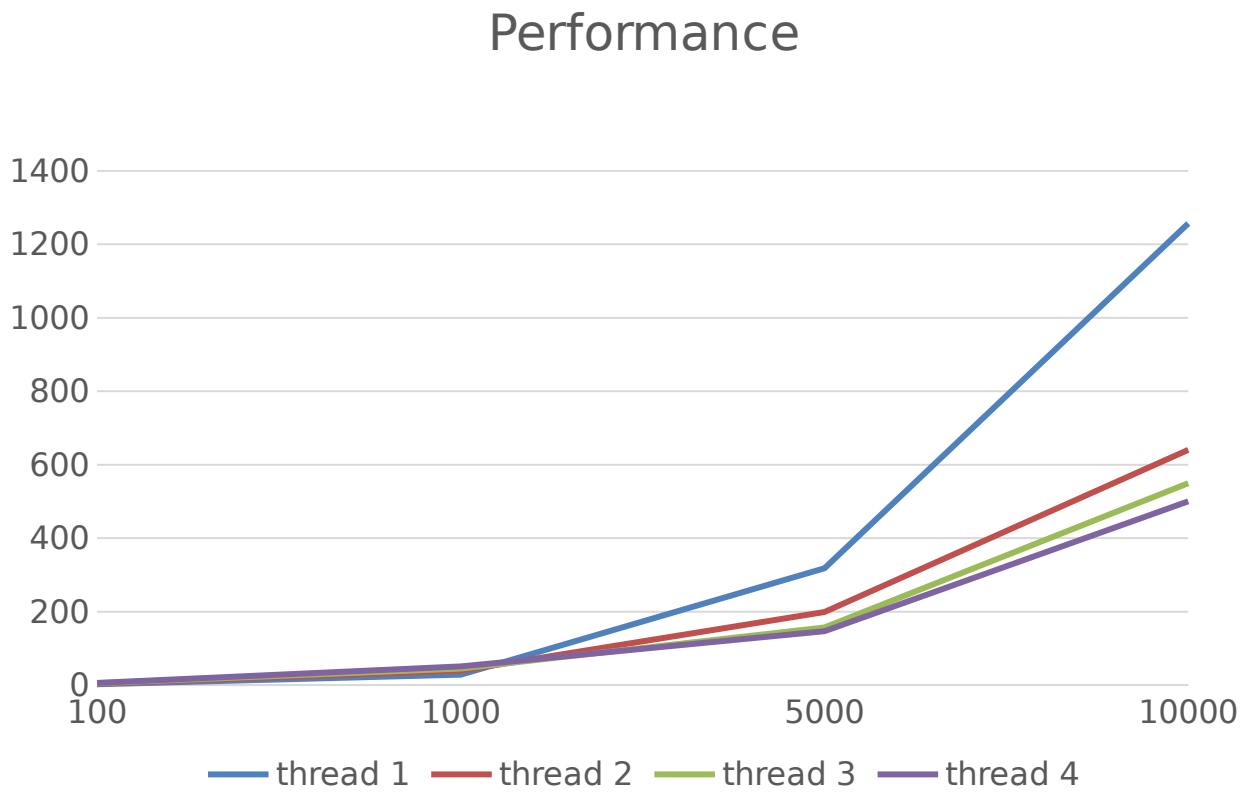
String "S"

	""	P	A	R	T
""	0	1	2	3	4
S	1	1	2	3	4
P	2	1	2	3	4
A	3	2	1	2	3
R	4	3	2	1	2
T	5	4	3	2	1
A	6	5	4	3	2
N	7	6	5	4	3

```java
switch (this.nT) {
    case 2:

        EditThread oneTwo = new EditThread(editMatrix, 1, this.chunckHeight, 1, this.chunckWidth, firstString, secondString);
        oneTwo.start();
        while (oneTwo.isAlive()) {
        }

        EditThread twoTwo = new EditThread(editMatrix, this.chunckHeight + 1, this.firstStringLength, 1, this.chunckWidth, firstString,
                secondString);
        EditThread threeTwo = new EditThread(editMatrix, 1, this.chunckHeight, this.chunckWidth + 1, this.secondStringLength, firstString,
                secondString);
        twoTwo.start();
        threeTwo.start();
        while (oneTwo.isAlive() || twoTwo.isAlive()) {
        }

        EditThread fourTwo = new EditThread(editMatrix, this.chunckHeight + 1, this.firstStringLength, this.chunckWidth + 1,
                this.secondStringLength, firstString, secondString);
        fourTwo.start();
        while (fourTwo.isAlive()) {
        }

        break;
```

```openmp
EditDistance ed("gagcccgtttcggatattcgcctttcggccaaaaatatggaatttagatagtccttg
clock_t tStart = clock();
ed.calculate(0, 750, 0, 750);
#pragma omp parallel sections
{
    #pragma omp section
    ed.calculate(751, 1500, 0, 750);
    #pragma omp section
    ed.calculate(0, 750, 751, 1500);
}
ed.calculate(751, 1500, 751, 1500);
cout << "\n";
printf("Time taken: %.5fs\n", (double)(clock() - tStart)/CLOCKS_PER_SEC);
return 0;
```

Performance comparison



Conclusion

- Performance worsening on a few data due to process management costs
- Linear performance improvement on many data