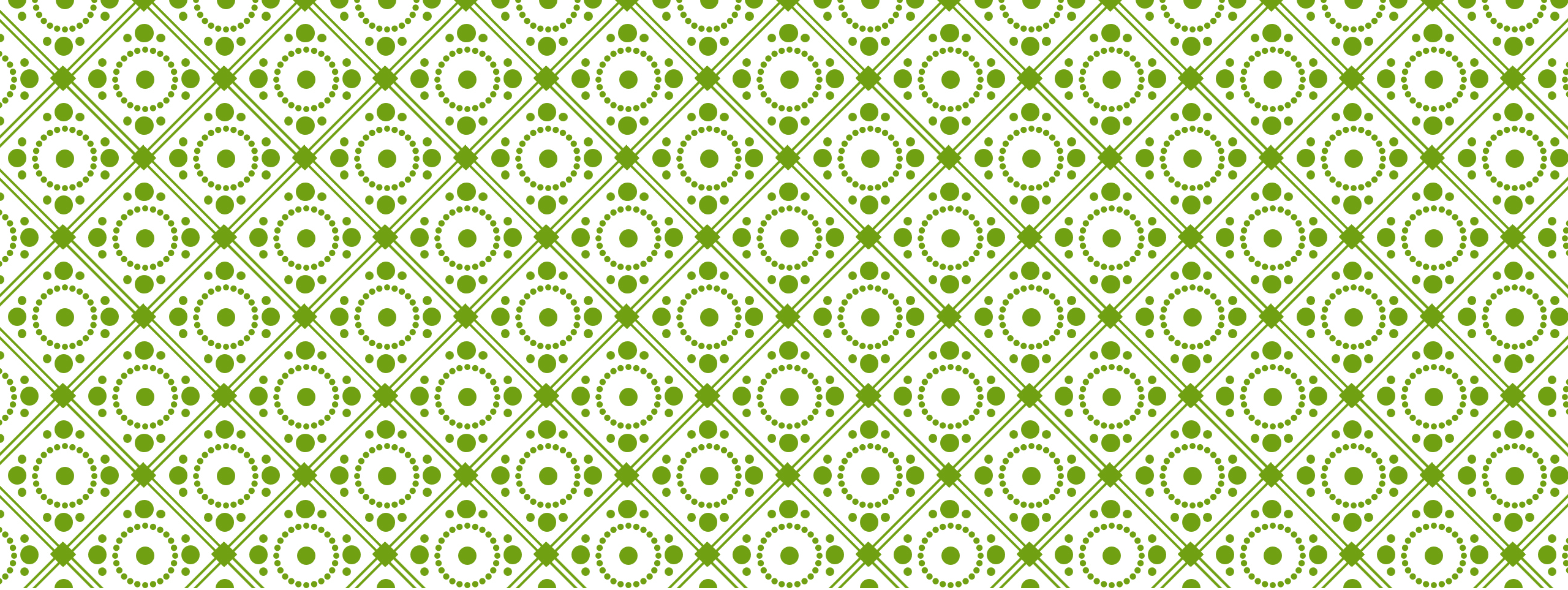


FILTRI BLOOM

RUCI SINDI
7090797
sindi.ruci@stud.unifi.it

COSA SONO I FILTRI BLOOM

I filtri Bloom sono una serie di funzioni di analisi delle informazioni che permettono di determinare se un dato (oppure un insieme di dati) è contenuto all'interno di un dataBase.

CODICE SEQUENZIALE

Il mio codice sequenziale è diviso in varie parti:

- Un file di utilità nel quale ci sono tutte e 8 le funzioni hash da me inventate
- Lettura dei file di inizializzazione e ricerca
- Funzione di inizializzazione dell'array di bit
- Funzione di ricerca delle parole

```
for string in text_array:
    bool_hash_functions(string,array_of_bits,n)

for s in string_to_find:
    is_the_string_present(s,n,array_of_bits)
```

FUNZIONI HASH

```python
def h0(string,n):
    j = 0
    for i in string:
        j = j + ord(i)
    j = j * 2668478416865146565
    return j % n

def h1(string,n):
    j = 0
    for i in string:
        j = j + hash(i)
    j = j * 266847841684515+6468546985484
    return j % n

def h2(string,n):
    j = 1
    for i in string:
        j = j * ord(i)
    j = j * 2668478651
    return j % n
```

```python
def h3(string,n):
    j = 1
    for i in string:
        j = j ** 3
    j = j * 266847856
    return j % n

def h4(string,n):
    j = 0
    for i in string:
        j = j + ord(i)
        j = j * 1651651
    j = int(j / 52)
    return j % n
```

```python
def h5(string,n):
    arrray_local = [0] * len(string)
    i = 0
    for j in string:
        arrray_local[i] = ord(j)
        i = i + 1
    matrix = np.vander(arrray_local, n)
    sumElements = 0
    for j in range(0, len(matrix)):
        for i in range(0, len(matrix[0])):
            sumElements = sumElements + (matrix[j][i]) % 20
    return sumElements % n

def h6(string,n):
    j = 0
    for i in string:
        j = j + hash(i)
        j = j - 2531859
    j = int(j / 69)
    return j % n
```

```python
def h7(string,n):
    j = 0
    for i in string:
        j = j + ord(i)
        j = j - 6464
    j = int(j / 20)
    return j % n
```

FUNZIONI DI INIZIALIZZAZIONE E DI RICERCA

```python
def bool_hash_functions(string, array_of_bits,n):
    i = h0(string,n)
    array_of_bits[i] = 1
    i = h1(string,n)
    array_of_bits[i] = 1
    i = h2(string,n)
    array_of_bits[i] = 1
    i = h3(string,n)
    array_of_bits[i] = 1
    i = h4(string,n)
    array_of_bits[i] = 1
    i = h5(string,n)
    array_of_bits[i] = 1
    i = h6(string,n)
    array_of_bits[i] = 1
    i = h7(string,n)
    array_of_bits[i] = 1
```

```python
def is_the_string_present(string_to_find,n,array_of_bits):
    if array_of_bits[h0(string_to_find, n)] == 0:
        pass
    if array_of_bits[h1(string_to_find, n)] == 0:
        pass
    if array_of_bits[h2(string_to_find, n)] == 0:
        pass

    if array_of_bits[h3(string_to_find, n)] == 0:
        pass

    if array_of_bits[h4(string_to_find, n)] == 0:
        pass

    if array_of_bits[h5(string_to_find, n)] == 0:
        pass

    if array_of_bits[h6(string_to_find, n)] == 0:
        pass

    if array_of_bits[h7(string_to_find, n)] == 0:
        pass

    else:
        print('the string: ' + string_to_find +  ' is in the array')
```

FUNZIONI DI INIZIALIZZAZIONE E DI RICERCA

Entrambe le funzioni vengono iterate su ogni parola presente nel file. Nel caso della funzione di ricerca si itera su ogni parola del file delle parole da cercare, per la funzione di inizializzazione si itera sul file utilizzato per «riempire» l'array

TEMPI DI ESECUZIONE

I tempi di esecuzione del codice sequenziale sono i seguenti:

- Inizializzazione sequenziale 8 secondi
- Ricerca sequenziale 3,45 secondi

```
the string: che  is in the array
the string: scrittura  is in the array
the string: survey is in the array
the string: geez is in the array
the string: dishonour is in the array
the string: appliance is in the array
the string: powerless is in the array
the string: level is in the array
the string: rightfully is in the array
the string: che  is in the array
the string: all is in the array
the string: all is in the array
the string: un is in the array
the string: che  is in the array
the string: pagine is in the array
the string: caratteri is in the array
3.4530439376831055
```

CODICE PARALLELO

Il codice parallelo presenta i seguenti punti fondamentali:

- Lettura dei file di inizializzazione e ricerca
- Funzioni hash
- Inizializzazione dell'array in parallelo
- Ricerca delle parole in parallelo

FUNZIONI DI INIZIALIZZAZIONE E RICERCA IN PARALLELO

```
Parallel(n_jobs=cores)(delayed(is_the_string_present)(string, array_of_bits) for string in string_to_find)
```

```
array_0 = Parallel(n_jobs=cores)(delayed(h0) (string) for string in text_array)
array_1 = Parallel(n_jobs=cores)(delayed(h1) (string) for string in text_array)
array_2 = Parallel(n_jobs=cores)(delayed(h2) (string) for string in text_array)
array_3 = Parallel(n_jobs=cores)(delayed(h3) (string) for string in text_array)
array_4 = Parallel(n_jobs=cores)(delayed(h4) (string) for string in text_array)
array_5 = Parallel(n_jobs=cores)(delayed(h5) (string) for string in text_array)
array_6 = Parallel(n_jobs=cores)(delayed(h6) (string) for string in text_array)
array_7 = Parallel(n_jobs=cores)(delayed(h7) (string) for string in text_array)

for i in range(len(array_0)):
    array_of_bits[array_0[i]] = 1
    array_of_bits[array_1[i]] = 1
    array_of_bits[array_2[i]] = 1
    array_of_bits[array_3[i]] = 1
    array_of_bits[array_4[i]] = 1
    array_of_bits[array_5[i]] = 1
    array_of_bits[array_6[i]] = 1
    array_of_bits[array_7[i]] = 1
```

La funzione is_the_string_present() è esattamente la stessa vista nelle slide sul codice sequenziale

TEMPI DI ESECUZIONE

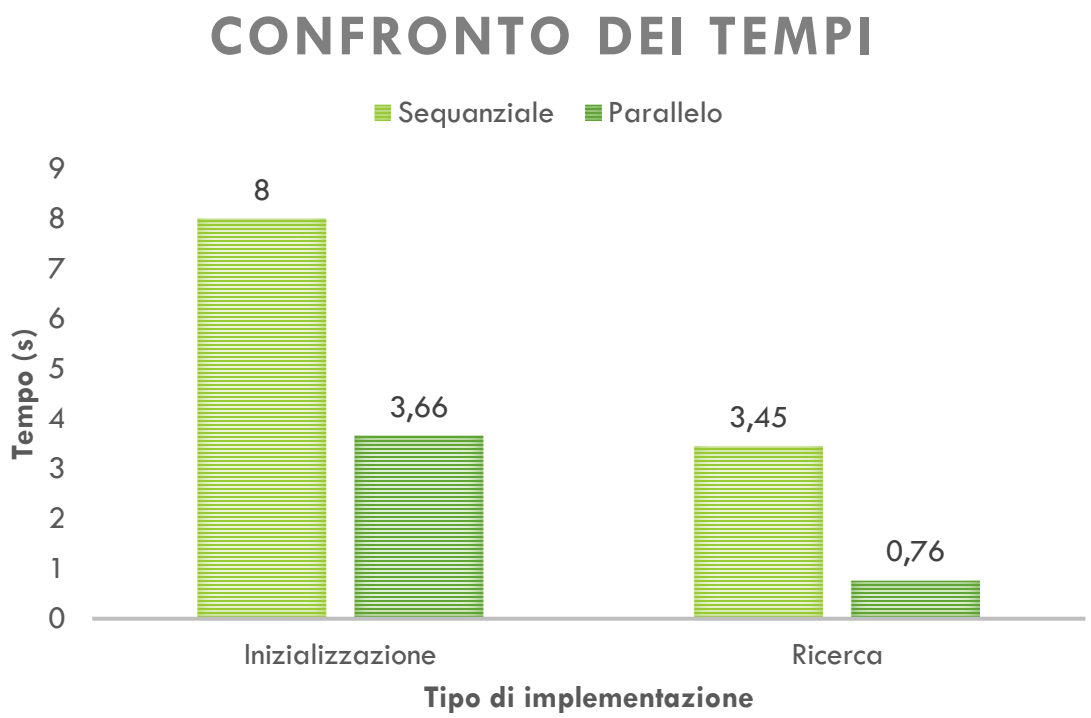
I tempi di esecuzione del codice parallelo sono i seguenti:

- Il tempo necessario a inizializzare l'array è 3,66 secondi
- Il tempo necessario per la ricerca è 0,76 secondi

NOTA SUI CODICI

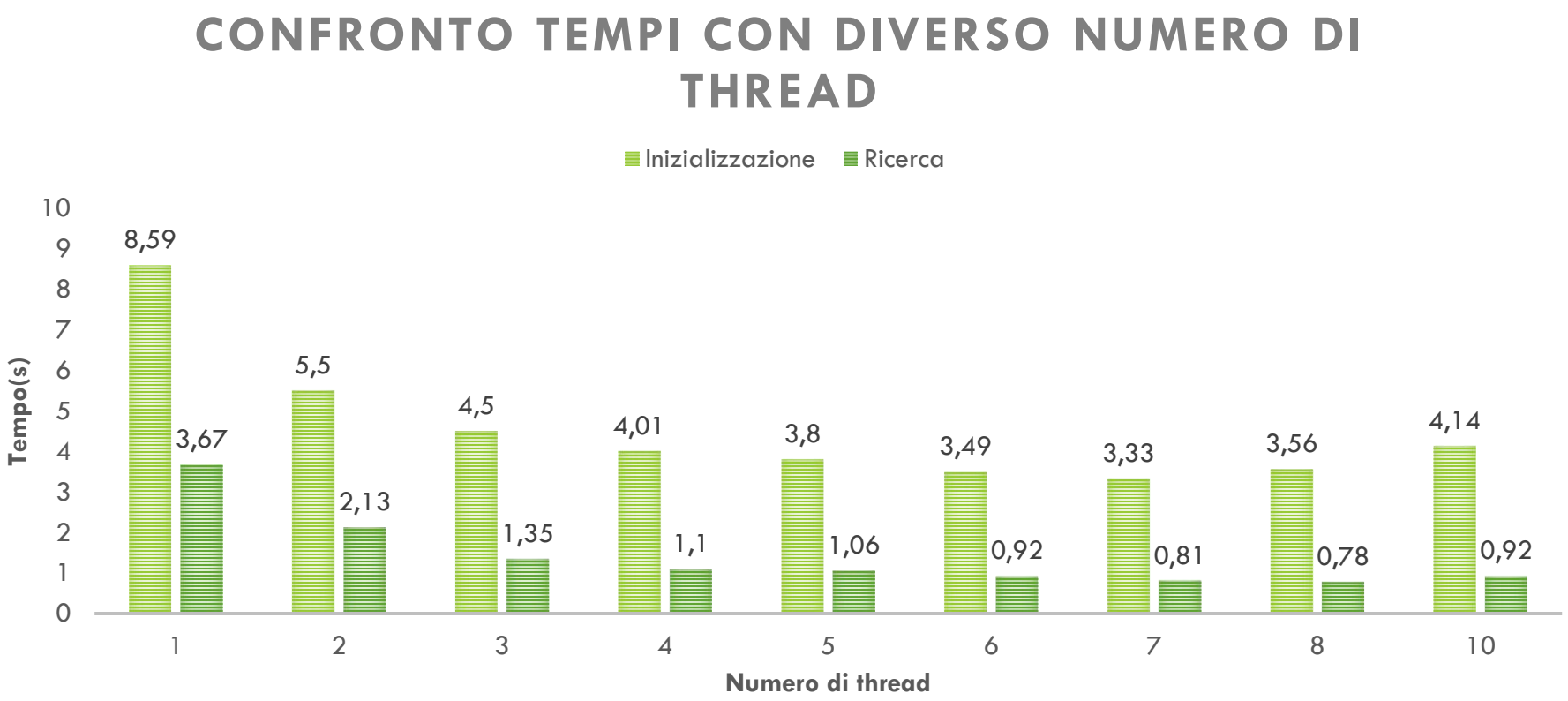
I codici hanno una percentuale di falsi positivi pari al 7% circa

CONFRONTO TRA LE DUE IMPLEMENTAZIONI



Con il grafico è ancora più evidente il notevole vantaggio della parallelizzazione

CONFRONTO TRA THREAD



Meglio il codice sequenziale che il parallelizzato con un solo thread