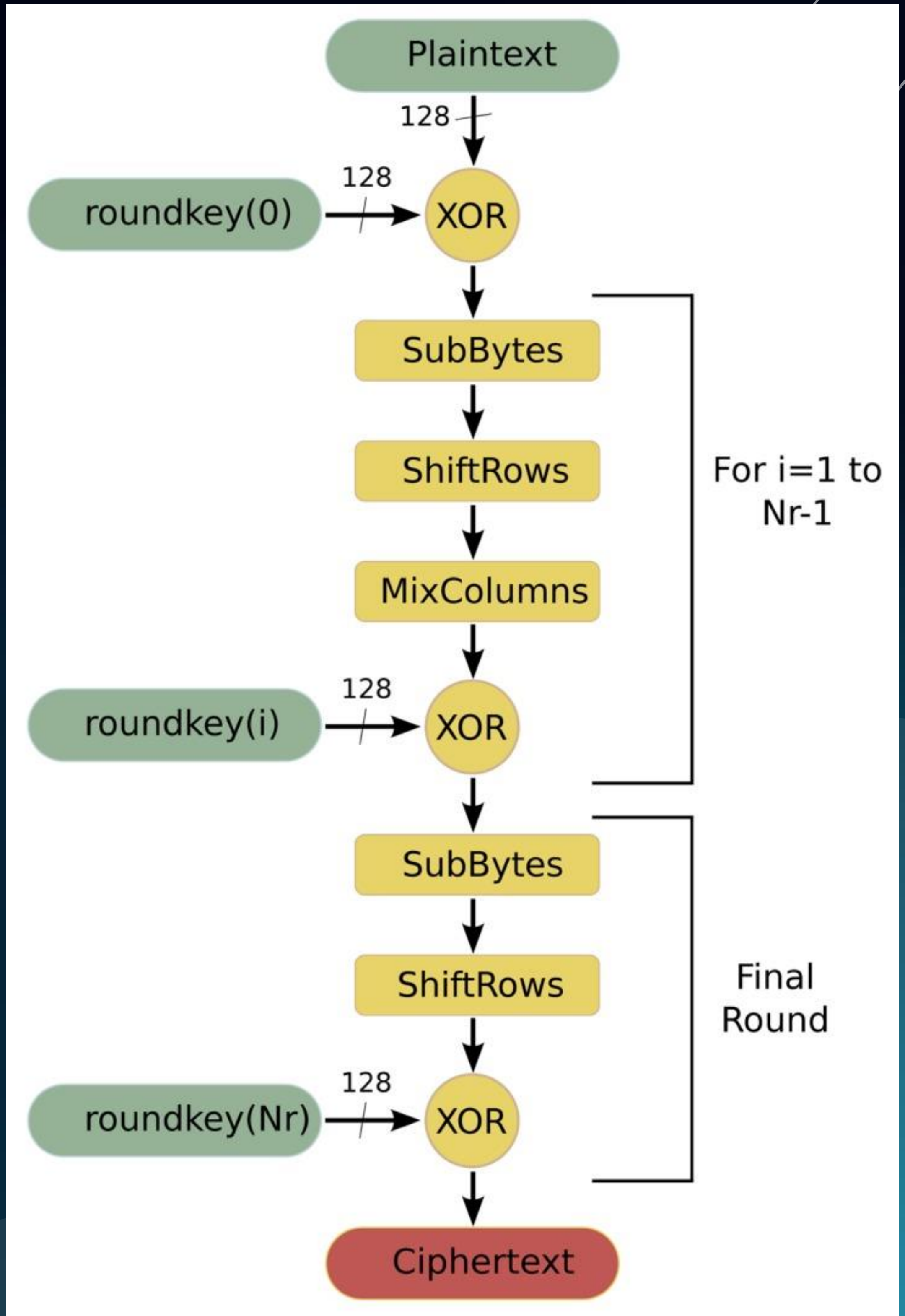
Parallel Implementation of AES Algorithm

Ali Khadangi
Atiye Bonakdar

AES Algorithm

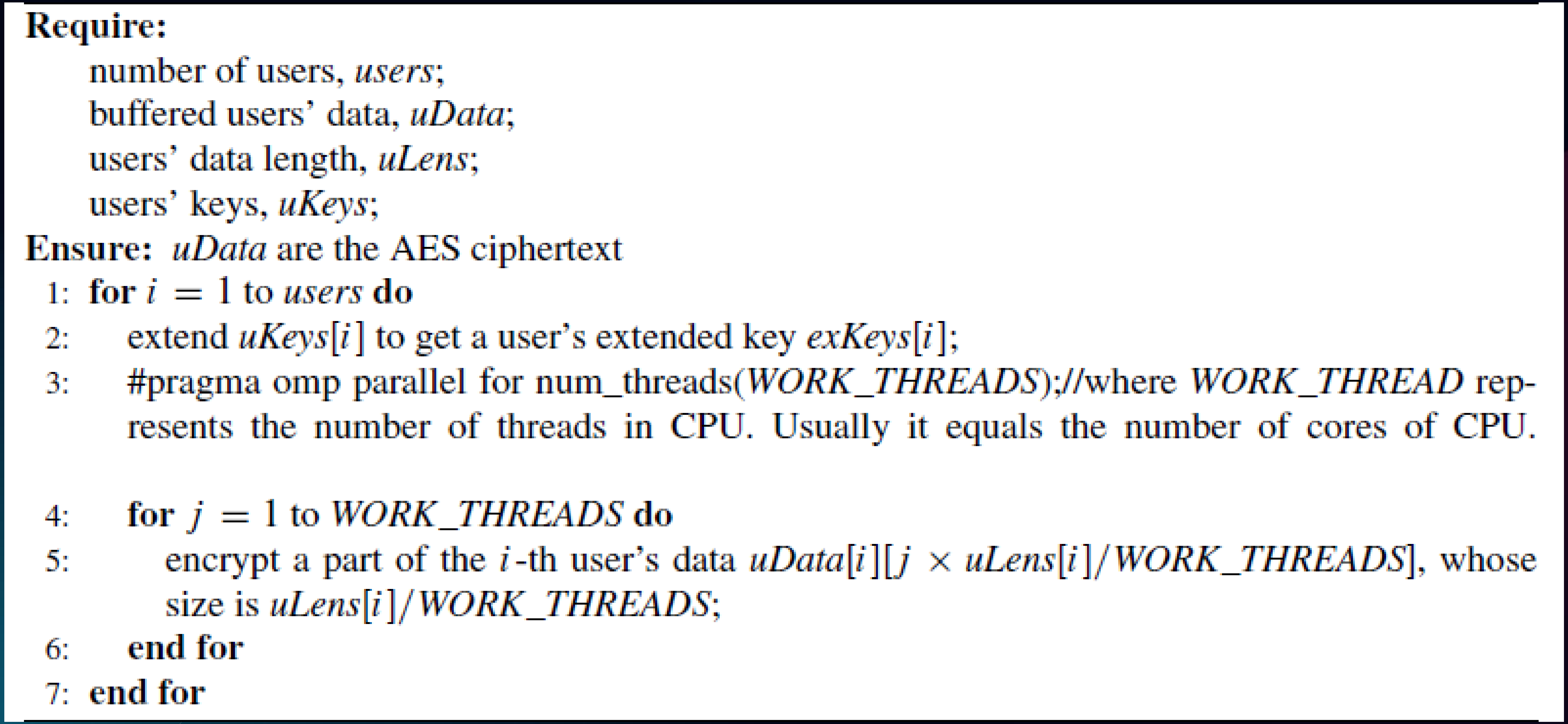
AES-128 encryption algorithm works by taking a 128-bit plaintext block and applying a series of substitution, permutation, and mixing operations using a key of 128 bits. This process involves multiple rounds (typically 10 rounds for AES-128) to transform the plaintext into ciphertext. The key schedule generates round keys used in each round to mix the data. The final output is the encrypted ciphertext.



System Specifications

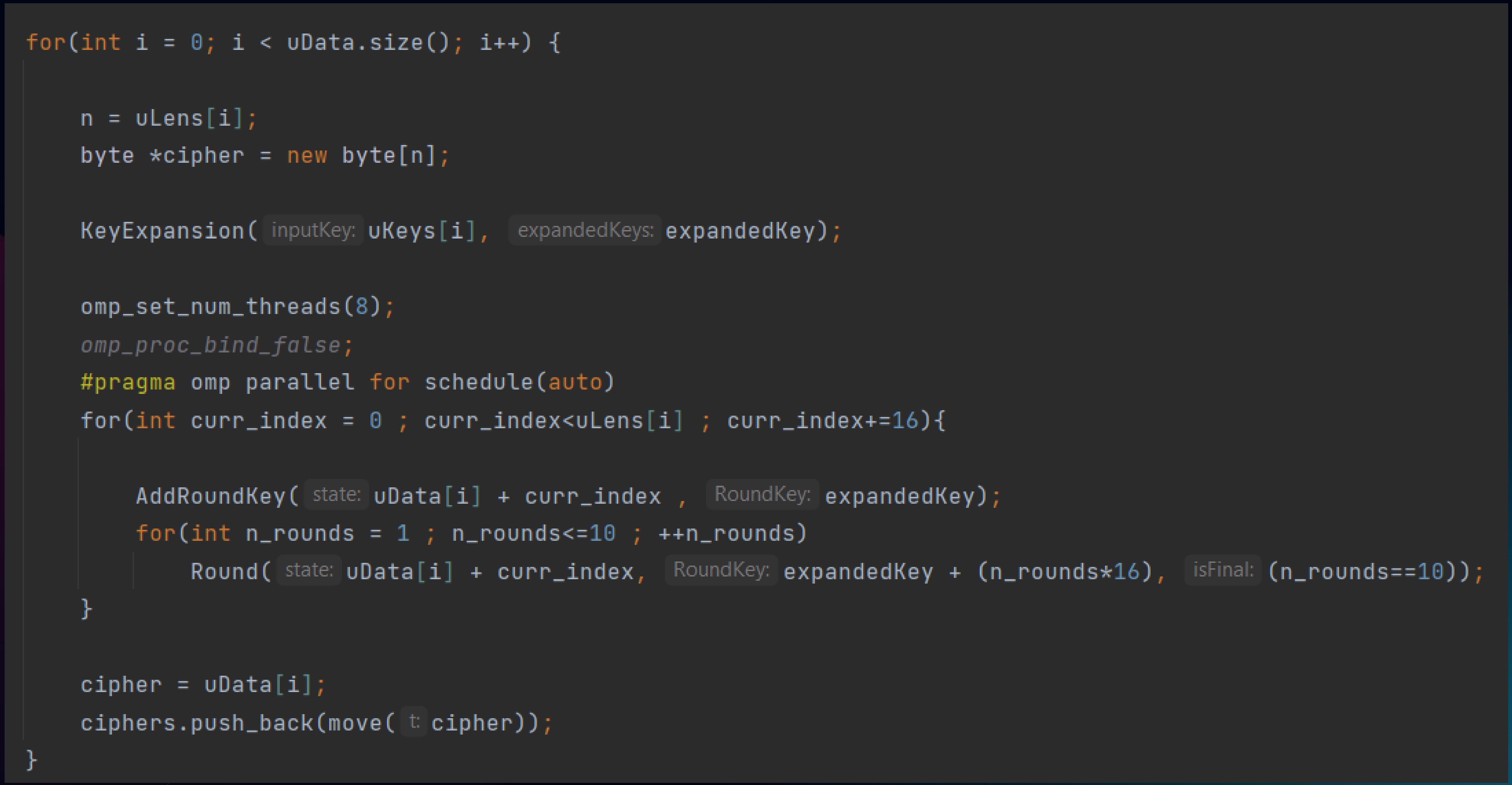
- CPU:
  - Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
  - Cores: 4
  - Logical processors: 8
- Cache:
  - L1 cache: 320 KB
  - L2 cache: 2.0 MB
  - L3 cache: 8.0 MB
- RAM:
  - 16 GB DDR4 2667 MHz

Pseudocode

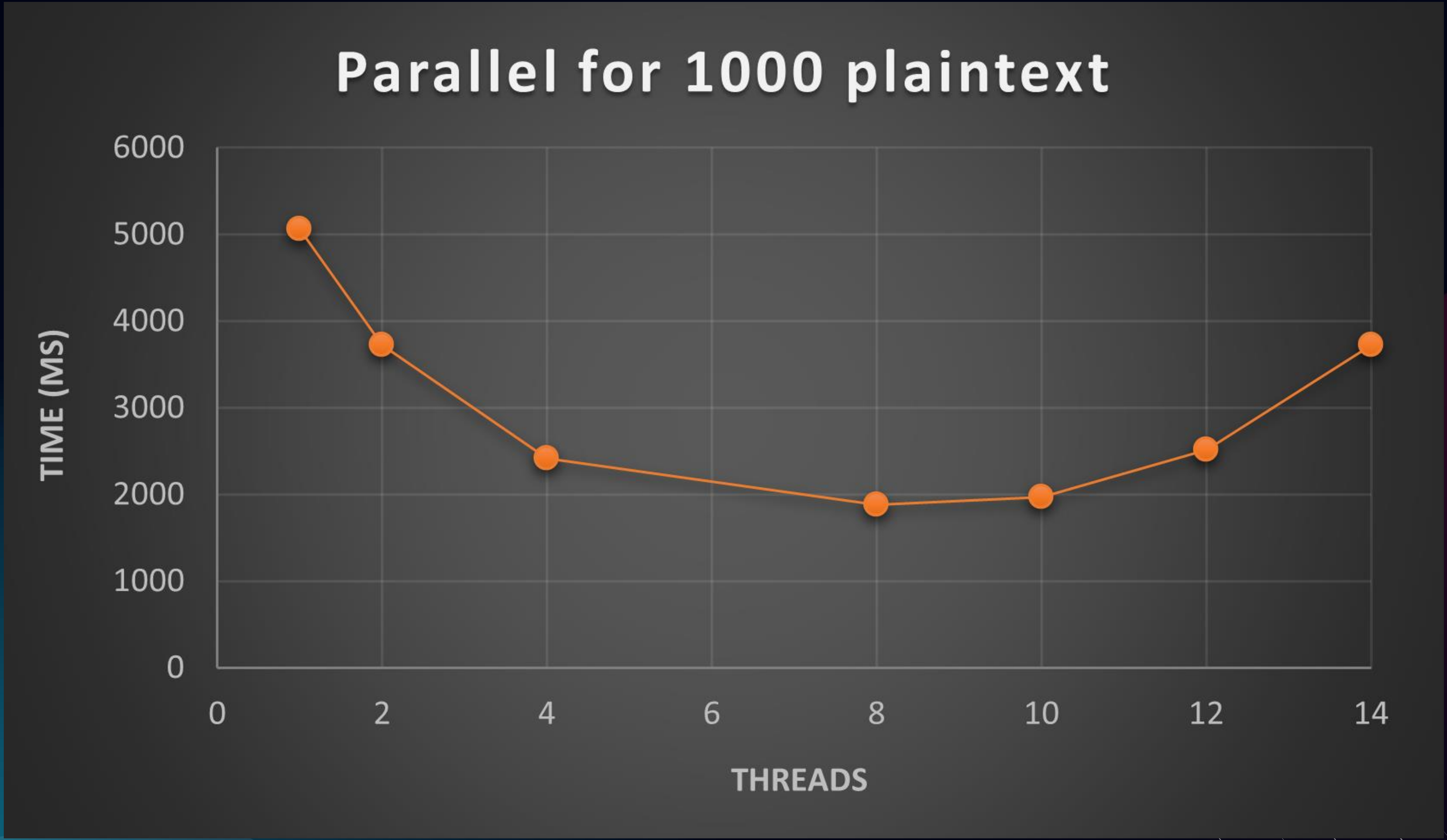


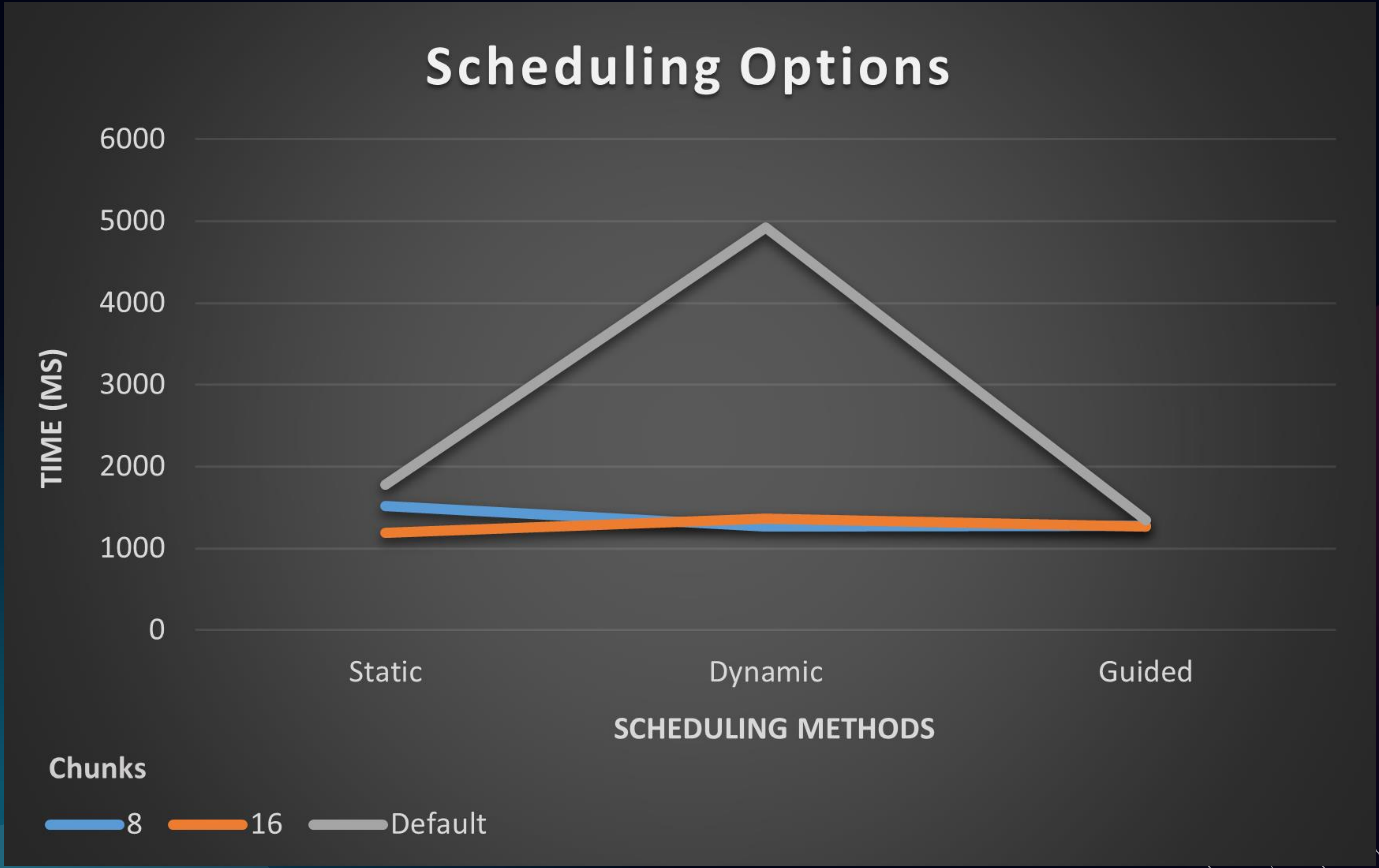
```
Require:
    number of users, $users$;
    buffered users' data, $uData$;
    users' data length, $uLens$;
    users' keys, $uKeys$;
Ensure: $uData$ are the AES ciphertext
for $i = 1$ to $users$ do
    extend $uKeys[i]$ to get a user's extended key $exKeys[i]$;
    #pragma omp parallel for num_threads($WORK\_THREADS$);//where $WORK\_THREAD$ represents the number of threads in CPU. Usually it equals the number of cores of CPU.
    for $j = 1$ to $WORK\_THREADS$ do
        encrypt a part of the $i$-th user's data $uData[i][j \times uLens[i]/WORK\_THREADS]$, whose size is $uLens[i]/WORK\_THREADS$;
    end for
end for
```

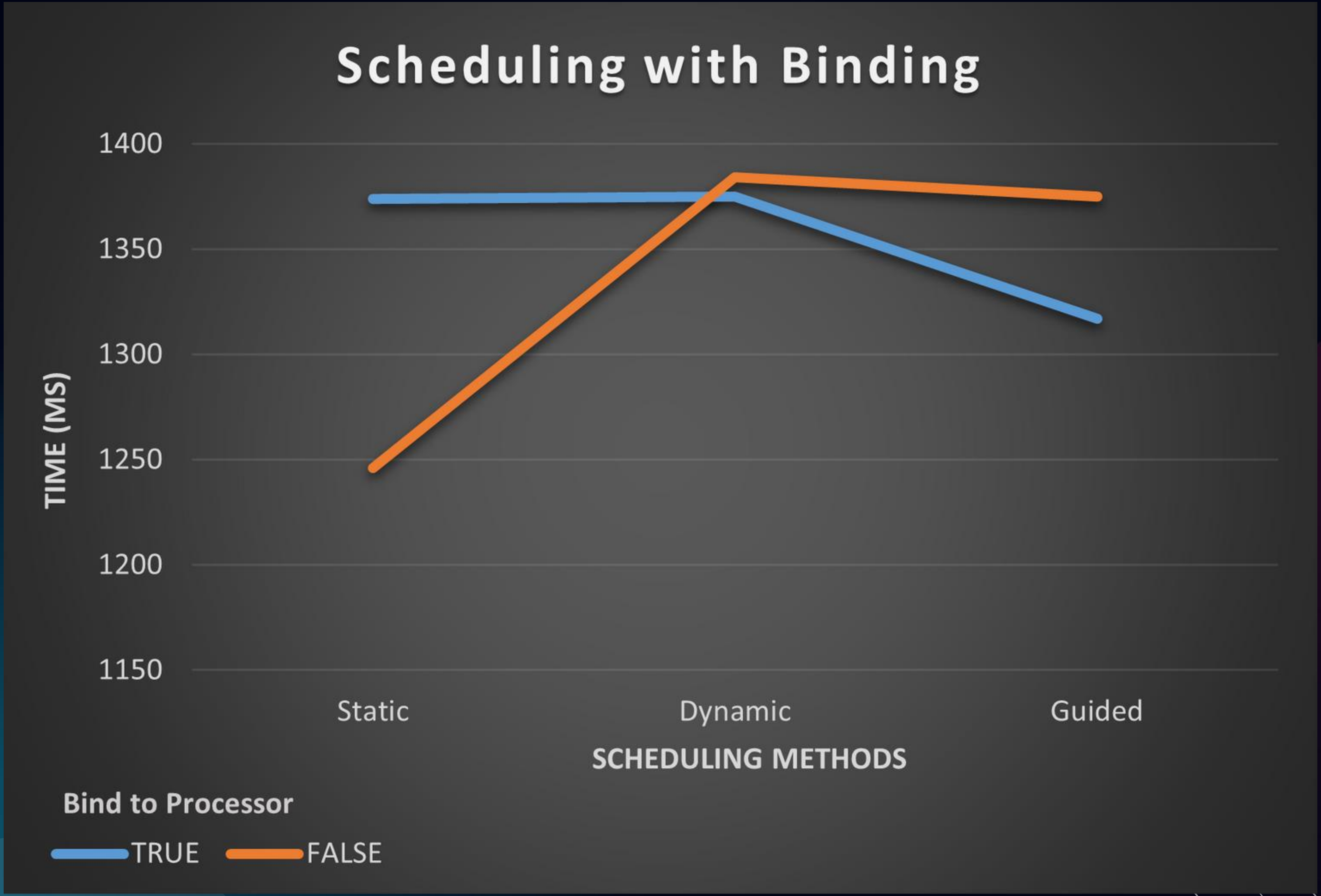
Parallel Loop



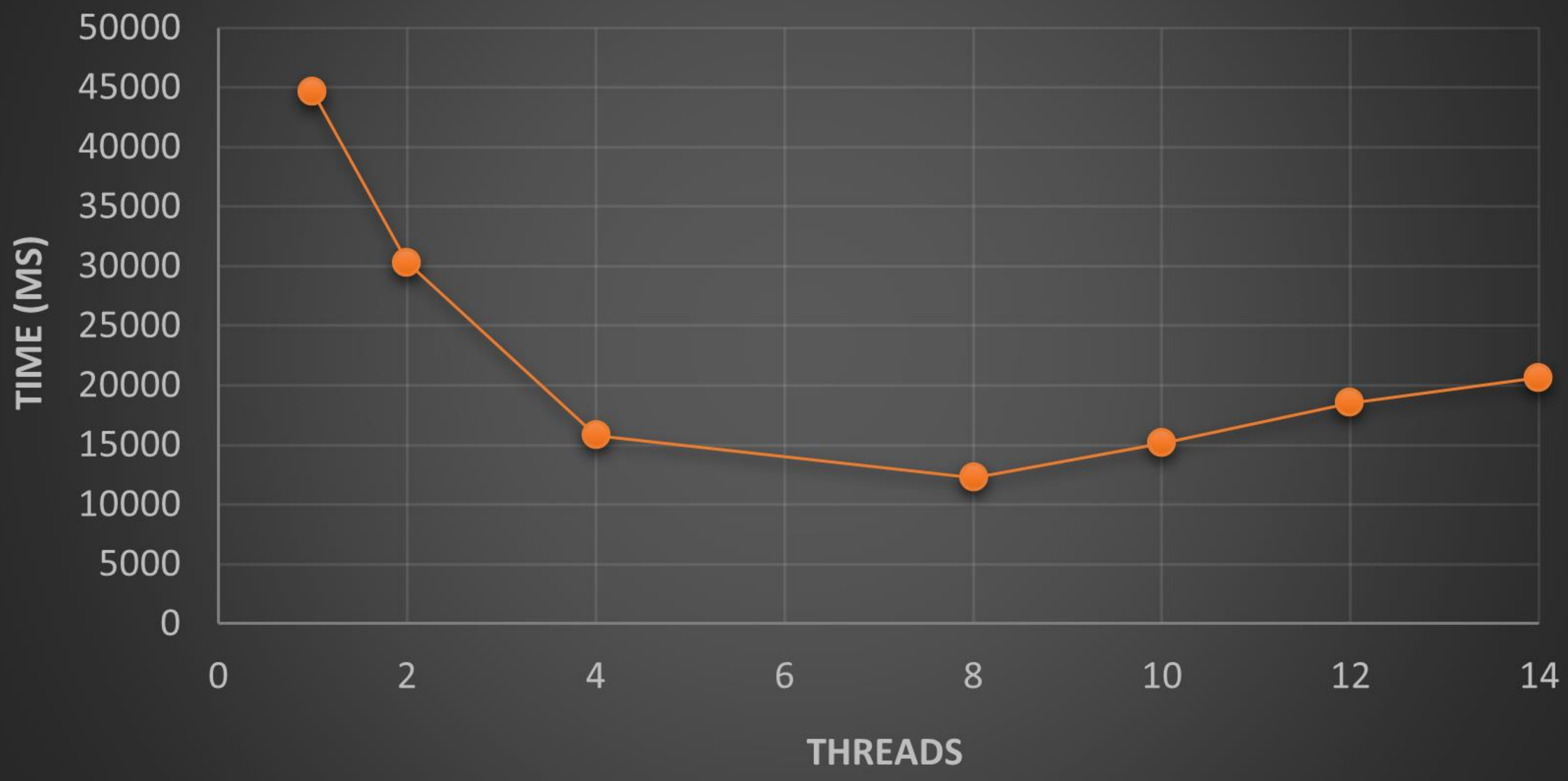
```
for(int i = 0; i < uData.size(); i++) {

    n = uLens[i];
    byte *cipher = new byte[n];

    KeyExpansion( inputKey: uKeys[i],  expandedKeys: expandedKey);

    omp_set_num_threads(8);
    omp_proc_bind_false;
    #pragma omp parallel for schedule(auto)
    for(int curr_index = 0 ; curr_index<uLens[i] ; curr_index+=16){

        AddRoundKey( state: uData[i] + curr_index ,  RoundKey: expandedKey);
        for(int n_rounds = 1 ; n_rounds<=10 ; ++n_rounds)
            Round( state: uData[i] + curr_index,  RoundKey: expandedKey + (n_rounds*16),  isFinal: (n_rounds==10));
    }

    cipher = uData[i];
    ciphers.push_back(move( t: cipher));
}
```

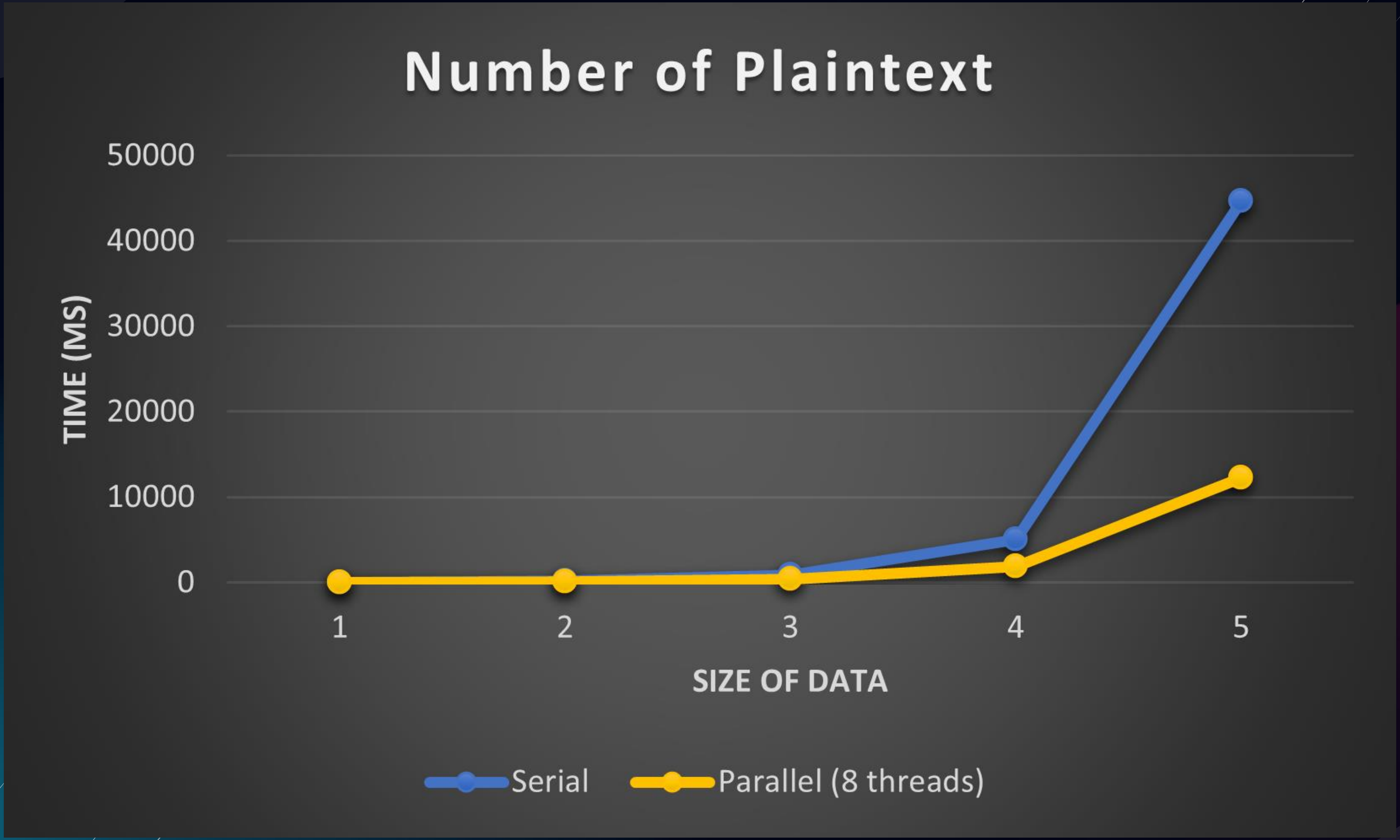

Parallel for 1000 plaintext
6000
5000
4000
3000
2000
1000
0
TIME (MS)
0
2
4
6
8
10
12
14
THREADS


Scheduling Options
6000
5000
4000
3000
2000
1000
0
TIME (MS)
Static
Dynamic
Guided
SCHEDULING METHODS
Chunks
8
16
Default

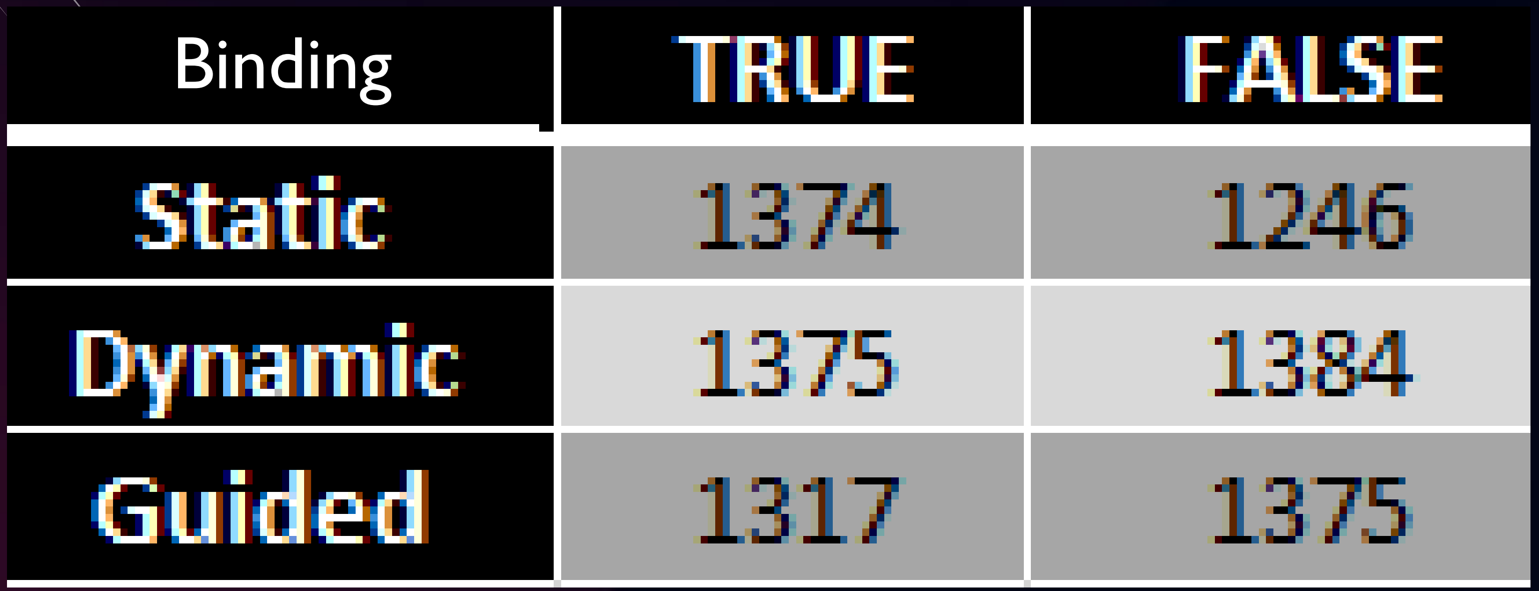

Scheduling with Binding
1400
1350
1300
1250
1200
1150
TIME (MS)
Static
Dynamic
Guided
SCHEDULING METHODS
Bind to Processor
TRUE
FALSE

Parallel for 10000 plaintext

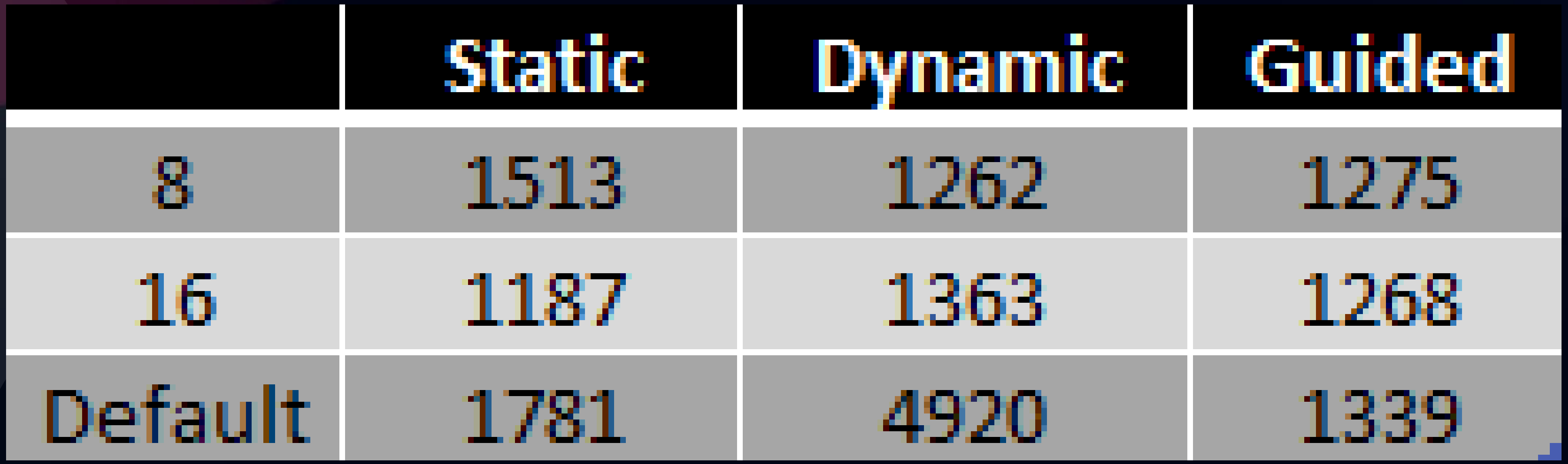



Number of Plaintext
50000
40000
30000
20000
10000
0
TIME (MS)
1
2
3
4
5
SIZE OF DATA
Serial
Parallel (8 threads)

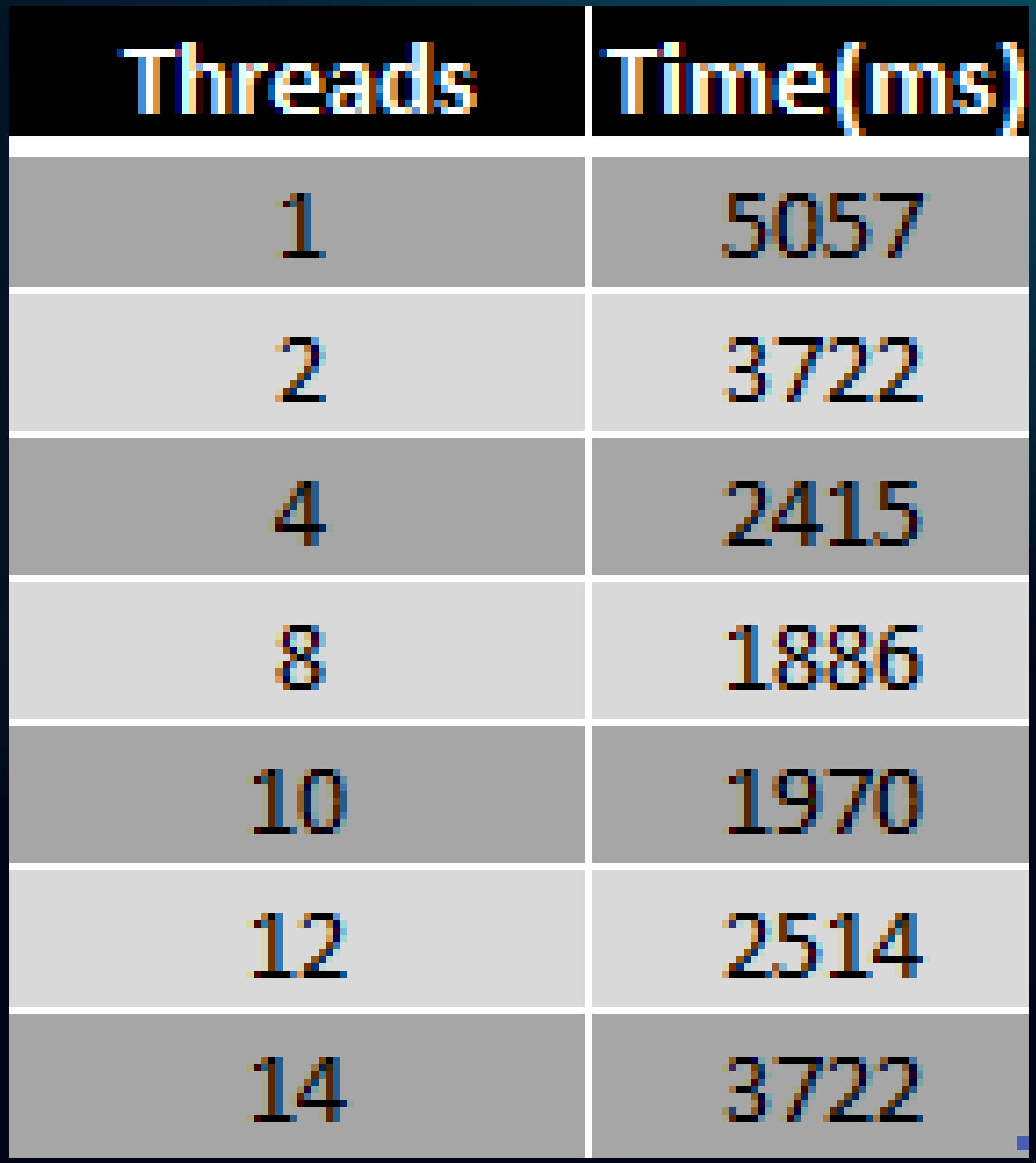
Result tables



Binding	TRUE	FALSE
Static	1374	1246
Dynamic	1375	1384
Guided	1317	1375



	Static	Dynamic	Guided
8	1513	1262	1275
16	1187	1363	1268
Default	1781	4920	1339



Threads	Time(ms)
1	5057
2	3722
4	2415
8	1886
10	1970
12	2514
14	3722

	Serial	Parallel (8 threads)	SpeedUp
6 (320 KB)	33	10	3.3
40 (2 MB)	228	114	2
160 (8 MB)	905	375	2.41
1000	5057	1886	2.68
10000	44630	12265	3.64

Thanks