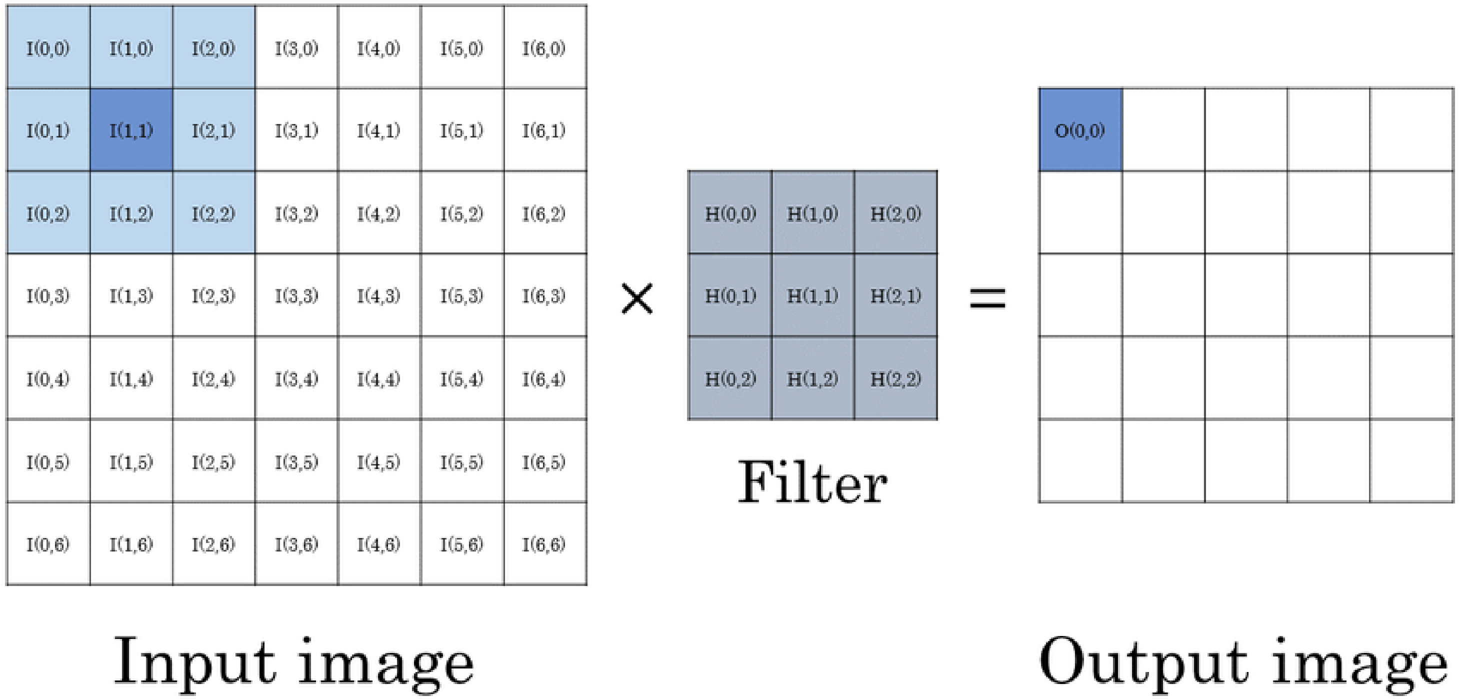
IMAGE CONVOLUTION

CSE633: Parallel Algorithm

Zhi Wen Huang

University at Buffalo The State University of New York

Image Convolution



I(0,0)	I(1,0)	I(2,0)	I(3,0)	I(4,0)	I(5,0)	I(6,0)
I(0,1)	I(1,1)	I(2,1)	I(3,1)	I(4,1)	I(5,1)	I(6,1)
I(0,2)	I(1,2)	I(2,2)	I(3,2)	I(4,2)	I(5,2)	I(6,2)
I(0,3)	I(1,3)	I(2,3)	I(3,3)	I(4,3)	I(5,3)	I(6,3)
I(0,4)	I(1,4)	I(2,4)	I(3,4)	I(4,4)	I(5,4)	I(6,4)
I(0,5)	I(1,5)	I(2,5)	I(3,5)	I(4,5)	I(5,5)	I(6,5)
I(0,6)	I(1,6)	I(2,6)	I(3,6)	I(4,6)	I(5,6)	I(6,6)

Input image

×

H(0,0)	H(1,0)	H(2,0)
H(0,1)	H(1,1)	H(2,1)
H(0,2)	H(1,2)	H(2,2)

Filter

=

O(0,0)				

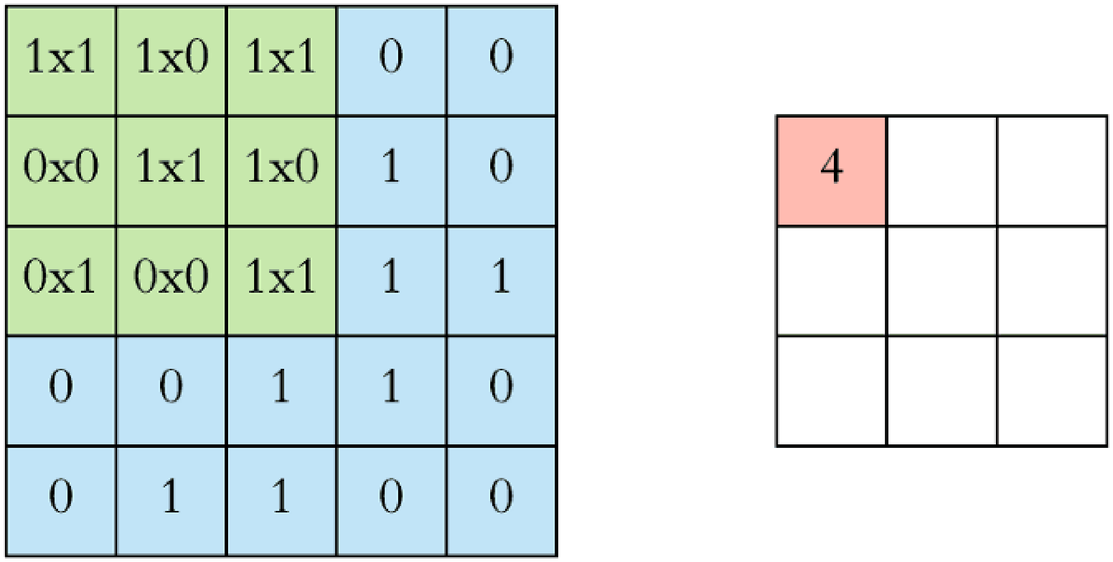
Output image

Sequential Approach

- Matrix Size 2^10 * 2^10
- Kernel Size 3x3
- Use zero-padding images
    - Add extra layers to do convolution.
- Medium filtering – This is use to reduce noisy image.
    - Our method is to find the medium base of the kernel size.

Recap

- Problem: How fast we can reduce the computation for depending on kernel and size.



1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Parallel Approach

- Matrix Size 2^15 x 2^15
- Kernel Size 3x3
- Padding images
  - Each matrixes padded image received from adjacency node if necessarily.
  - Out layer of the matrix are added zero-padded images.
- Medium filtering – This is use to reduce noisy image.
  - The Kernel size is 3x3.
- Template matching – This is to find common in terms of pixels
  - Kernel size is 27x27
  - Uses Sum of Squared Difference algorithm

Things I do..

0

1	**2**	**…**	**N/S**
2	3	…	
…			
N/S			N/S

1

N/S+1	**N/s+2**	**…**	**N**
		…	
…			
N/S			N/S

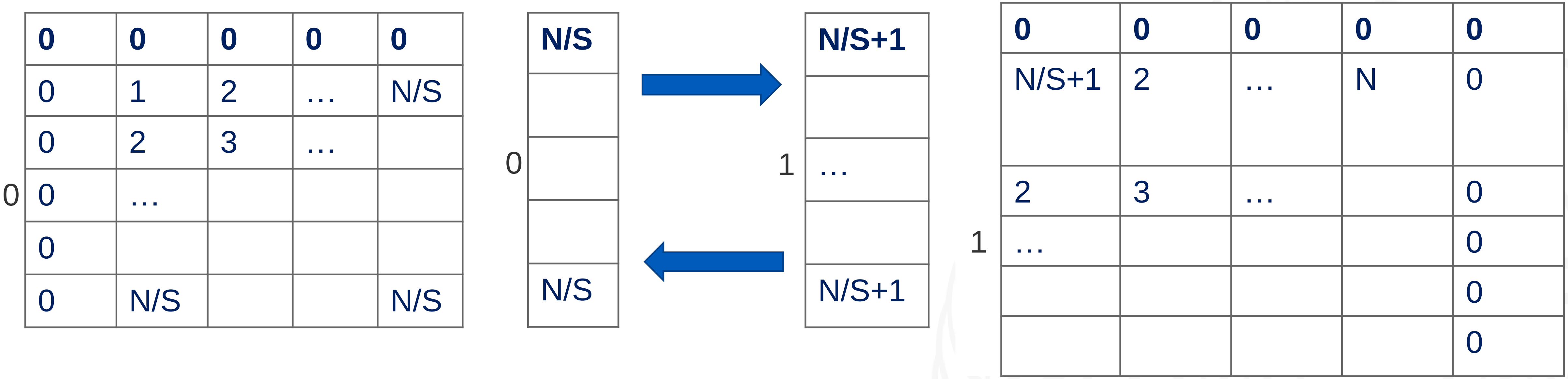
2

N/s+1	**+2**	**…**	**N/S**
+2	3	…	
…			
N			N

3

N/s+1	**+2**	**…**	**N**
+2		…	
…			
N			N

Continued..



	0	**0**	**0**	**0**	**0**
	0	1	2	…	N/S
	0	2	3	…	
0	0	…			
	0				
	0	N/S			N/S

	N/S
0	
	N/S

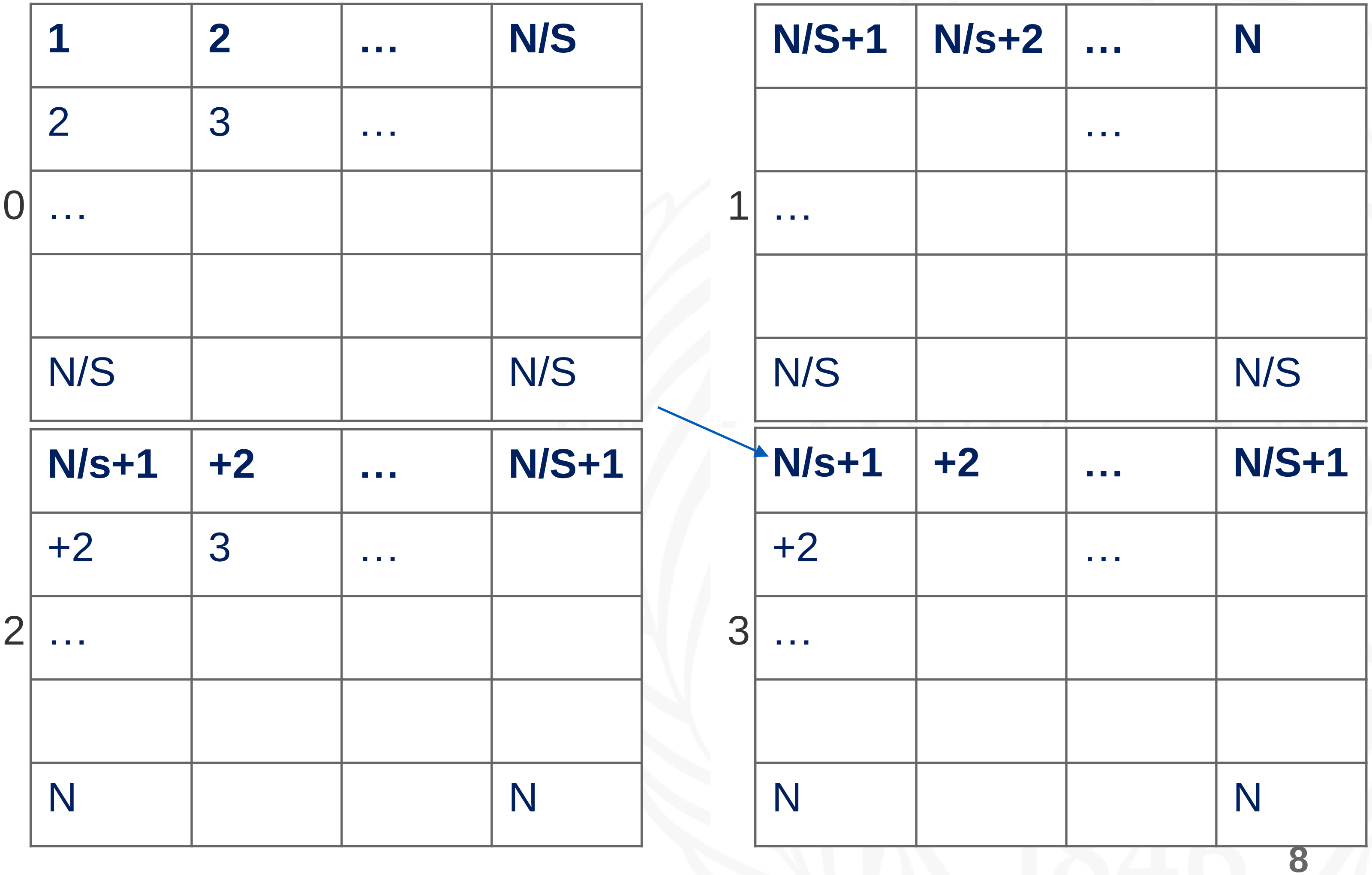
	N/S+1
1	…
	N/S+1

	0	**0**	**0**	**0**	**0**
	N/S+1	2	…	N	0
	2	3	…		0
1	…				0
					0
					0

- When Node0 needs the outside layer from Node1, Node1 send the first column to Node0.
- Node1 waits till Node0 complete the tasks then Node0 will send the completed column to Node1.

Corners

- Node3 Upper left corner need to wait Node1 and Node2 to send the lower row from Node1 and right column from Node3.
- Why Node3 does not need to get Node0 the corner value?
- Since Node1 and Node2 already received the corner value from Node0, Then when passing the data, we just need to take one of the value either from Node1 or Node2.



0

1	**2**	**…**	**N/S**
2	3	…	
…			
N/S			N/S

1

N/S+1	**N/s+2**	**…**	**N**
		…	
…			
N/S			N/S

2

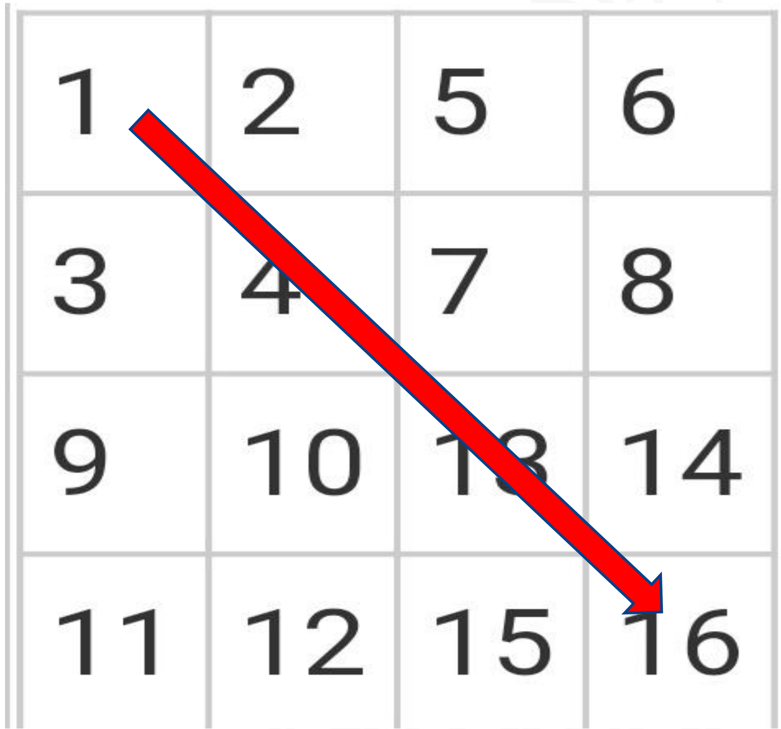
N/s+1	**+2**	**…**	**N/S+1**
+2	3	…	
…			
N			N

3

N/s+1	**+2**	**…**	**N/S+1**
+2		…	
…			
N			N

Parallel Approach cont.

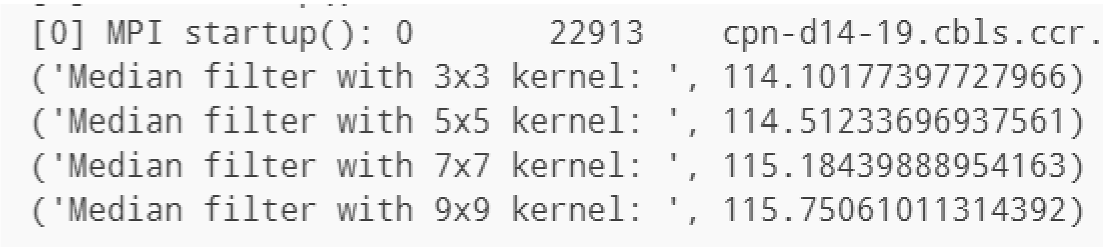
- 1024x1024 Matrix
- 16 Nodes
- Each will have:
    - 256*256 matrix
    - Padded matrix will be 257*257
- 48 Message passing and receiving.



1	2	5	6
3	4	7	8
9	10	13	14
11	12	15	16

Parallel Approach cont.

- Mpi4py – Library for Python to work with MPI
- Python/Anaconda
- What I change with my original idea:
  - Only using 3x3 kernel
  - Changing the image size to max for each node.

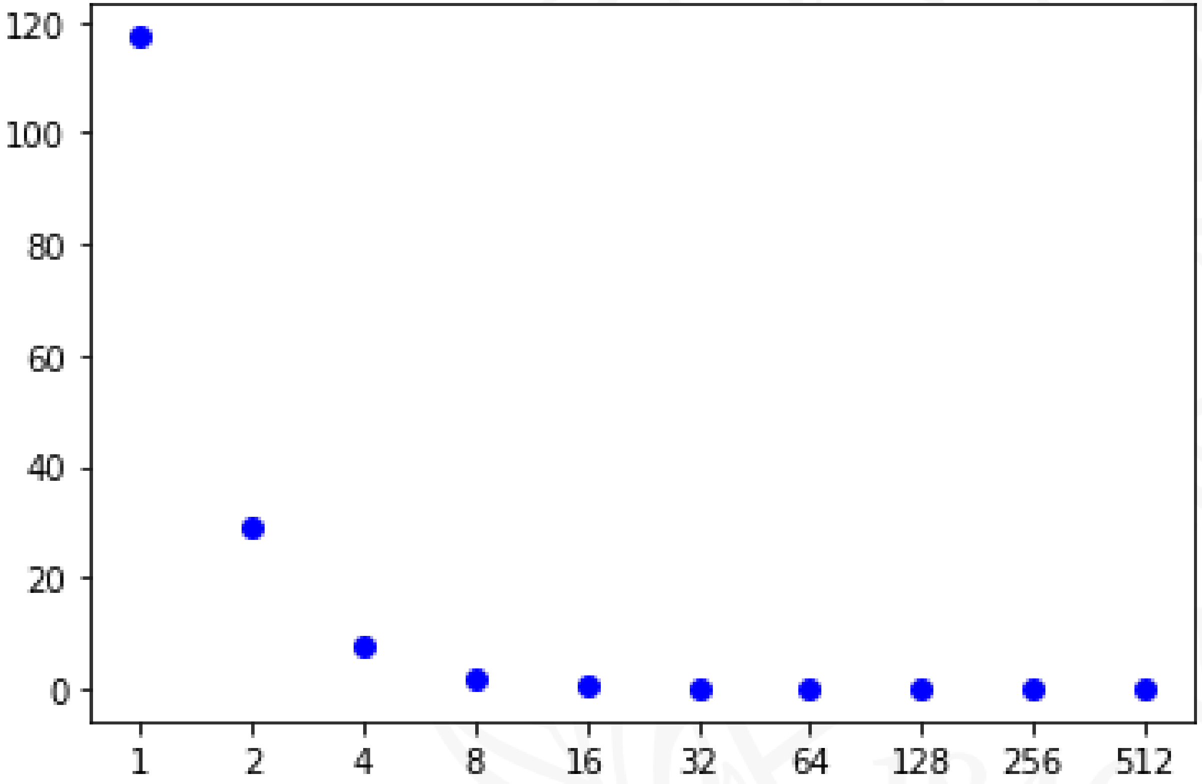


```
[0] MPI startup(): 0       22913    cpn-d14-19.cbls.ccr.
('Median filter with 3x3 kernel: ', 114.10177397727966)
('Median filter with 5x5 kernel: ', 114.51233696937561)
('Median filter with 7x7 kernel: ', 115.18439888954163)
('Median filter with 9x9 kernel: ', 115.75061011314392)
```

Results

- Matrix 2^10x2^10 with 3x3 kernel

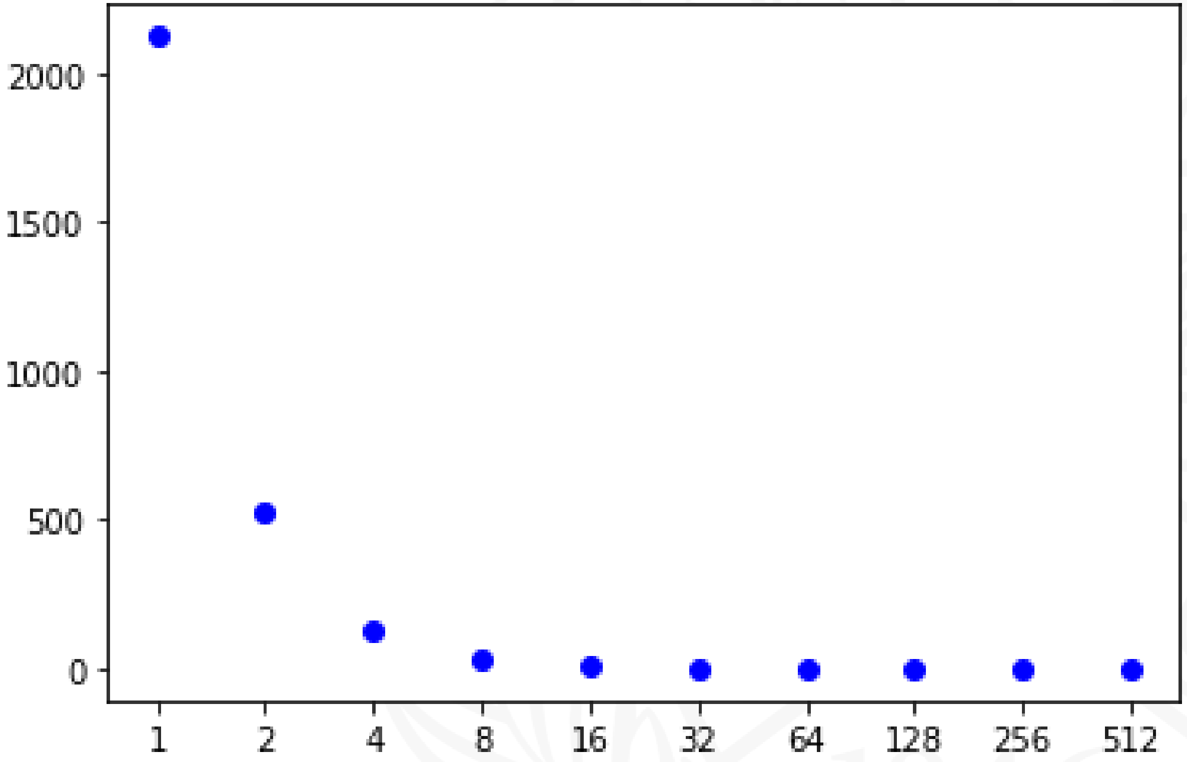
Nodes	Time(s)		Nodes	Time(s)
1	117.34		32	0.10
2	29.21		64	0.022
4	7.44		128	0.007
8	1.85		256	0.024
16	0.41		512	0.030



Results

- Matrix 2^12 x 2^12 kernel 3x3

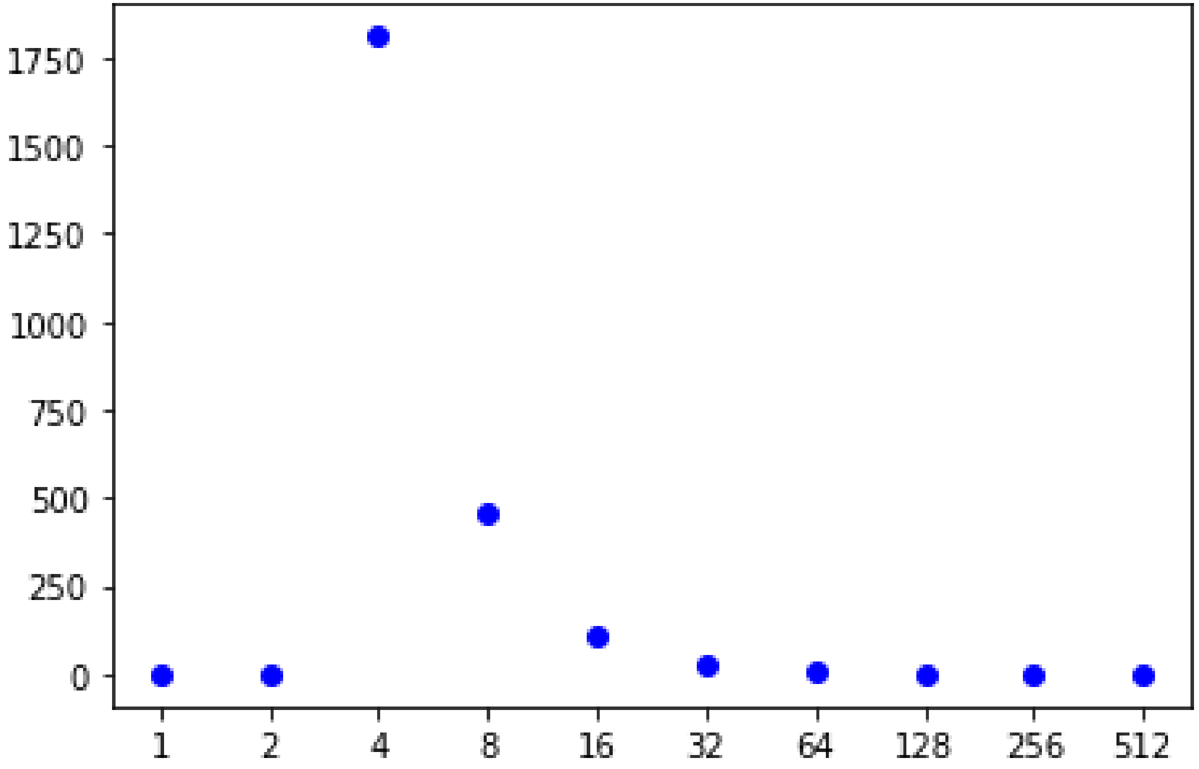
Nodes	Time(s)		Nodes	Time(s)
1	2122.82		32	1.65
2	530.26		64	0.40
4	133.24		128	0.11
8	34.48		256	0.26
16	6.7		512	0.52



Results

- Matrix 2^15 x 2^15 Kernel 3x3

Nodes	Time(s)		Nodes	Time(s)
1	NAN		32	26.3
2	NAN		64	6.89
4	1810.27		128	1.78
8	454.05		256	0.41
16	105.24		512	0.96



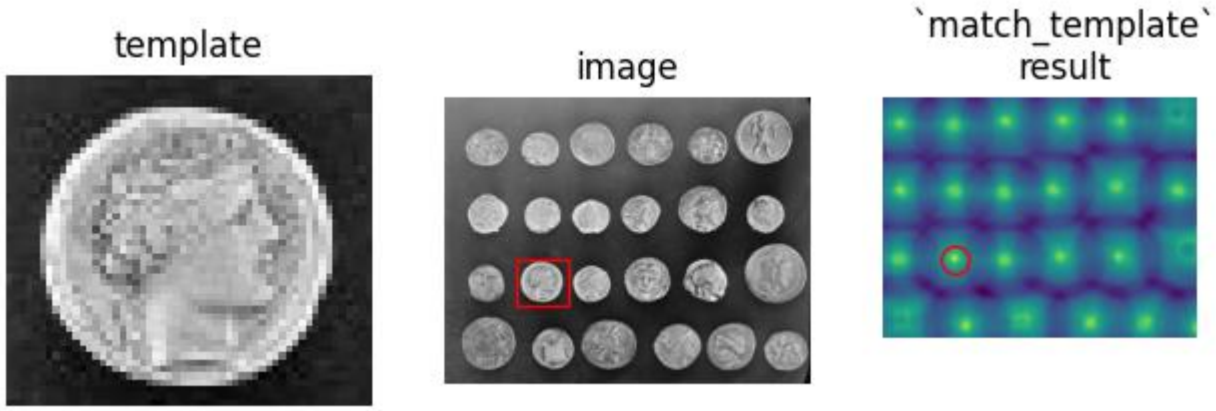
Computations in Numbers

- Trial 1: 2^10 * 2^10 * 3 x 3 = 9437184 computations
- Trial 2: 2^12 * 2^12 * 3 x 3 = 150994944 computations
- Trial 3: 2^15 * 2^15 * 3 x 3 = 9663676416 computations

- However this does not include how long each processor waits.

Template Matching

- In this application, we want to find the correlation between in pixel given the template and the image.
- Treat it as “Where’s Waldo” book where we have he picture of Waldo’s head as the template and image is the background.



Normalized Cross-Correlation

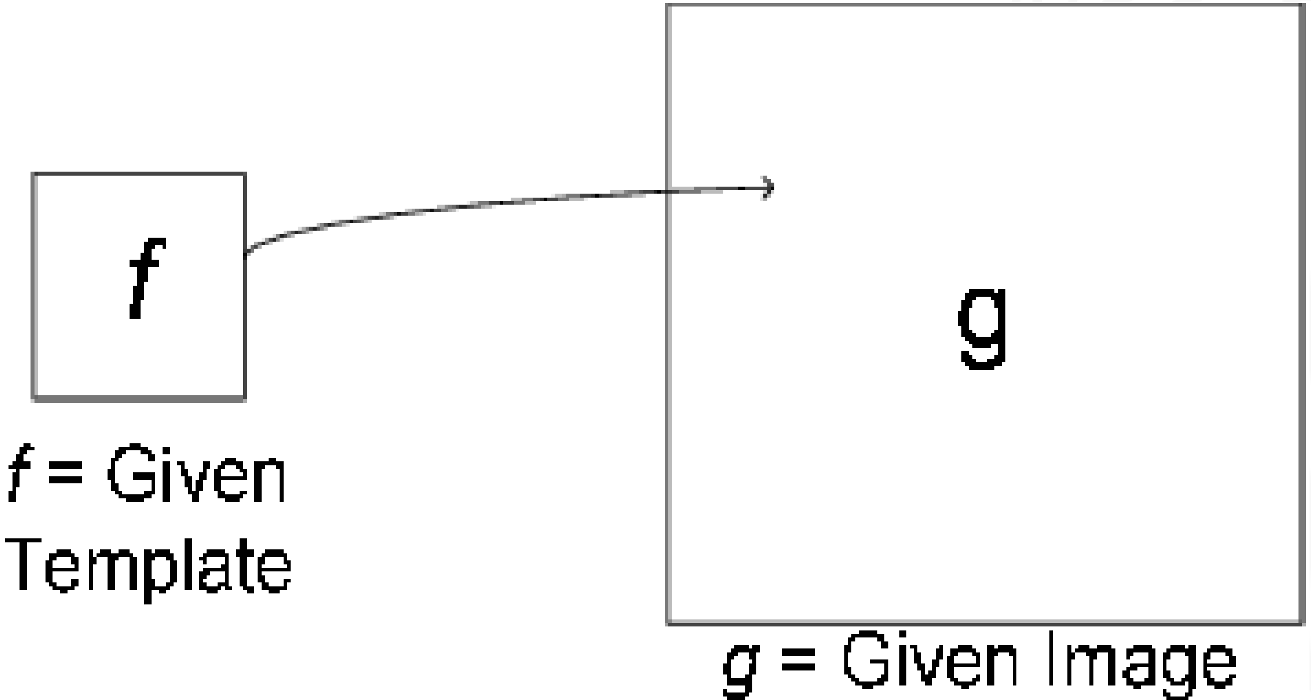
- Normalized Cross-Correlation algorithm is used to detect similarities between the template kernel and the image.

$$\mathrm{NCC} = \frac{\frac{1}{mm'}\sum_{i=p}^{p+m-1}\sum_{i'=p'}^{p'+m'-1}\left(f_{i,i'}-\bar{f}\right) * \cdot \left(g_{i+q-p,i'+q'-p'}-\bar{g}\right)}{\sqrt{\frac{1}{mm'}\sum_{i=p}^{p+m-1}\sum_{i'=p'}^{p'+m'-1}\left|f_{i,i'}-\bar{f}\right|^2}\sqrt{\frac{1}{mm'}\sum_{i=q}^{q+m-1}\sum_{i'=q'}^{q'+m'-1}\left|g_{i,i'}-\bar{g}\right|^2}}$$

$$\equiv \frac{\overline{f^* g}-\left(\bar{f}\right)^* \bar{g}}{\sqrt{\overline{|f|^2}-|\bar{f}|^2}\sqrt{\overline{|g|^2}-|\bar{g}|^2}}$$

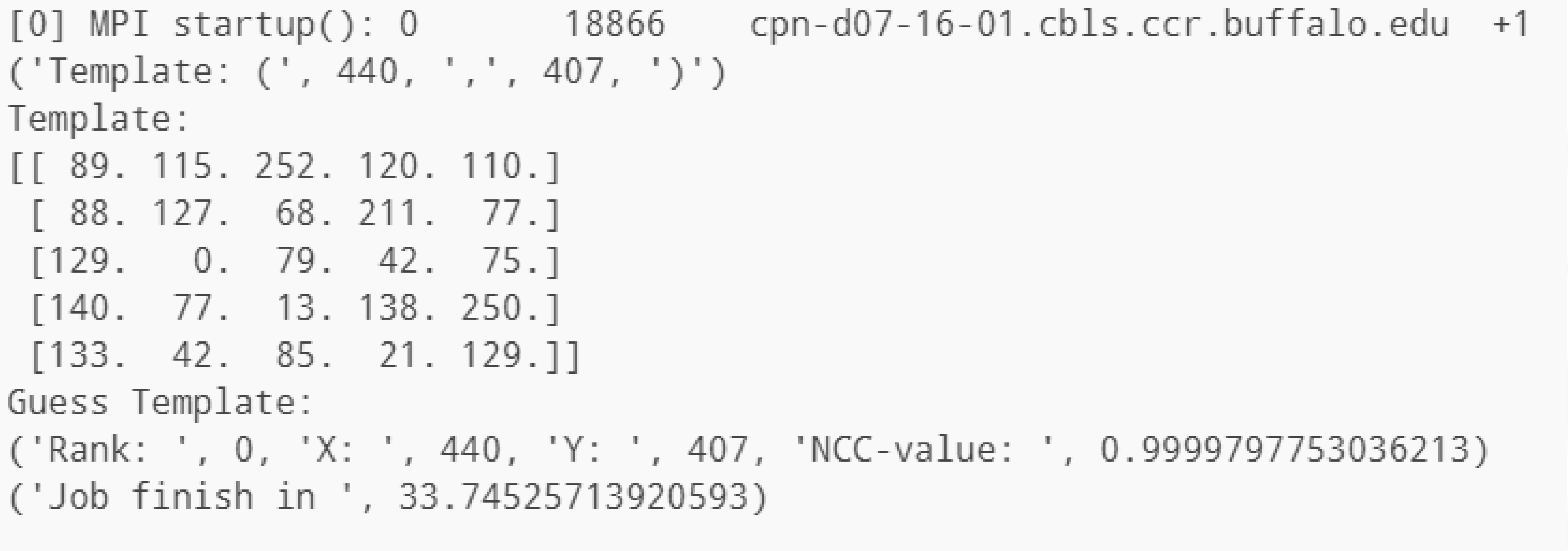
Sequential Approach

- Given a template 5x5 and an Image
- Calculate the NCC value for each image patch and return the max NCC value.
- Image patch size is the same as the template size.



Results

- Template is located at X = 55 , Y = 455
- Our guess is located at X = 55, Y = 455
- Ncc-value = 0.99 which is near 1 means that good correlation.

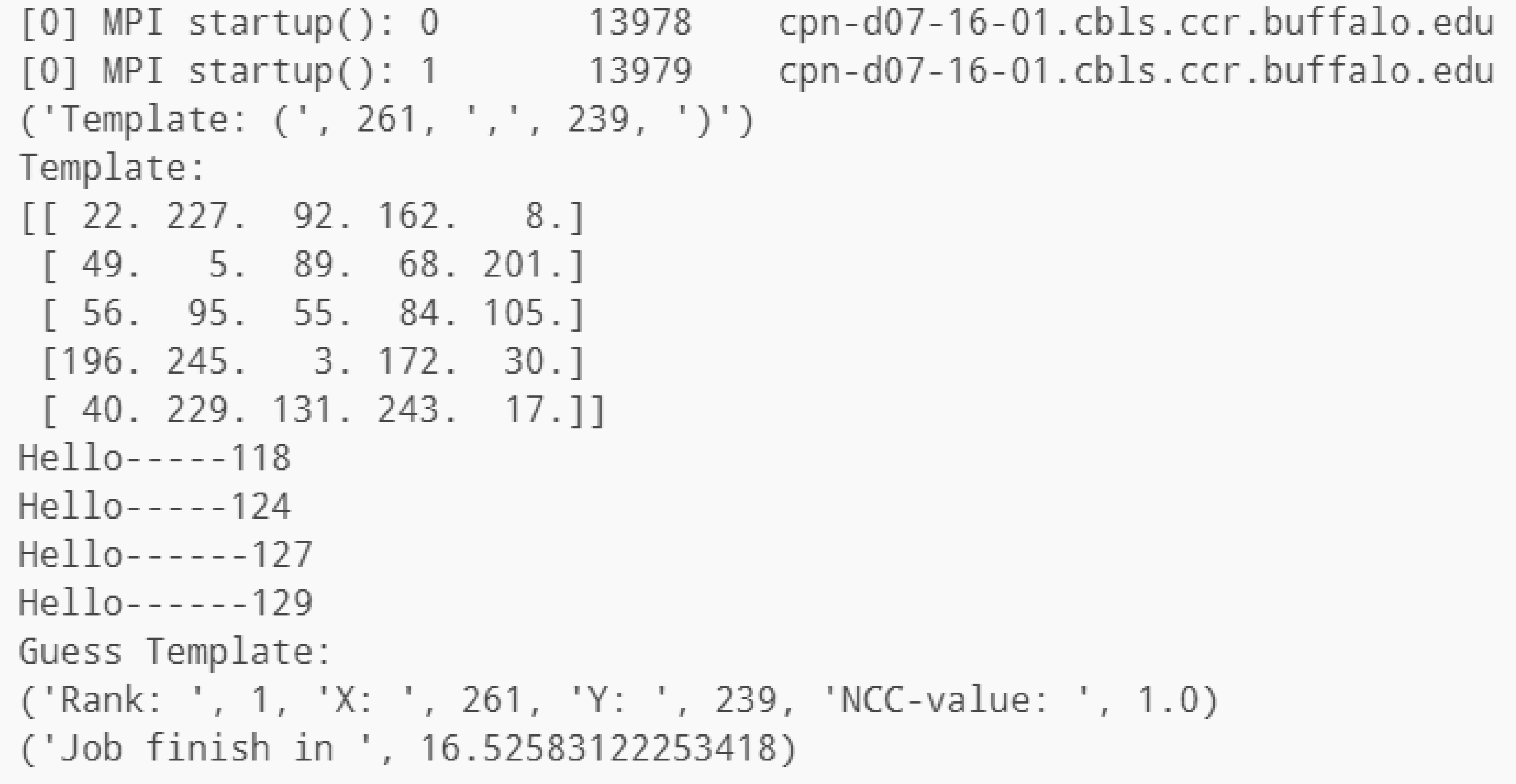


```
[0] MPI startup(): 0       18866    cpn-d07-16-01.cbls.ccr.buffalo.edu  +1
('Template: (', 440, ',', 407, ')')
Template:
[[ 89. 115. 252. 120. 110.]
 [ 88. 127.  68. 211.  77.]
 [129.   0.  79.  42.  75.]
 [140.  77.  13. 138. 250.]
 [133.  42.  85.  21. 129.]]
Guess Template:
('Rank: ', 0, 'X: ', 440, 'Y: ', 407, 'NCC-value: ', 0.9999797753036213)
('Job finish in ', 33.74525713920593)
```

Parallel Approach

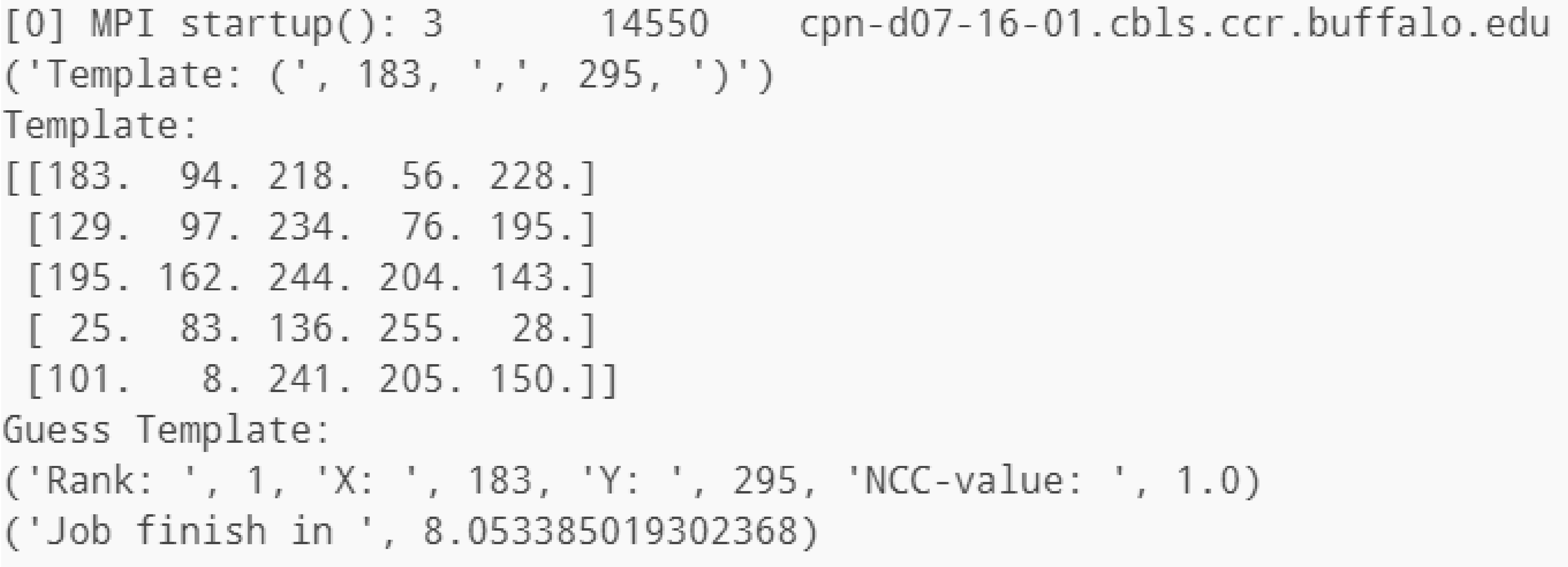
- Each processor gets part of the input image.
- Each processor will have the same template to work on.
- Individually, compute the NCC algorithm to find the max correlation on the given image
- Detects borders and corner pixels and receive data from neighbor nodes to complete the algorithm
- Return the highest NCC value and check if the return axis matches with the template.

Results: 2 Nodes



```
[0] MPI startup(): 0       13978    cpn-d07-16-01.cbls.ccr.buffalo.edu
[0] MPI startup(): 1       13979    cpn-d07-16-01.cbls.ccr.buffalo.edu
('Template: (', 261, ',', 239, ')')
Template:
[[ 22. 227.  92. 162.   8.]
 [ 49.   5.  89.  68. 201.]
 [ 56.  95.  55.  84. 105.]
 [196. 245.   3. 172.  30.]
 [ 40. 229. 131. 243.  17.]]
Hello-----118
Hello-----124
Hello------127
Hello------129
Guess Template:
('Rank: ', 1, 'X: ', 261, 'Y: ', 239, 'NCC-value: ', 1.0)
('Job finish in ', 16.52583122253418)
```

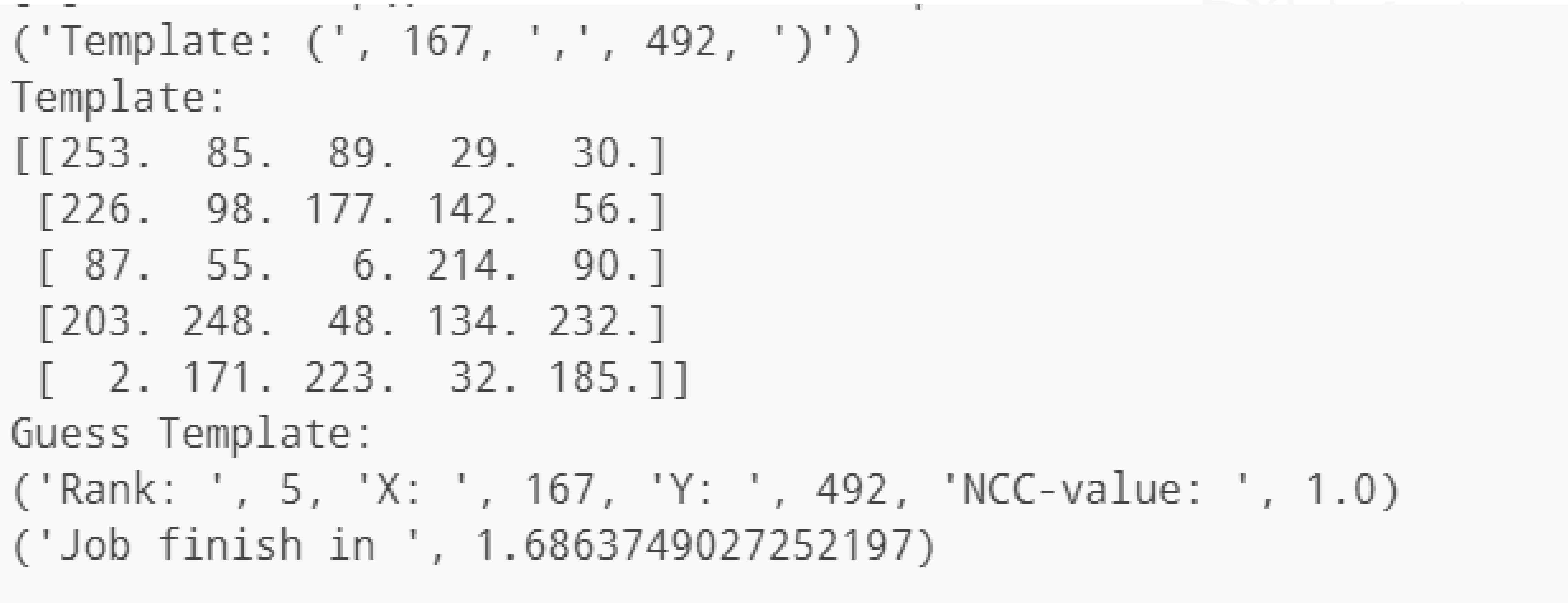
Results: 4 Nodes



```
[0] MPI startup(): 3       14550    cpn-d07-16-01.cbls.ccr.buffalo.edu
('Template: (', 183, ',', 295, ')')
Template:
[[183.  94. 218.  56. 228.]
 [129.  97. 234.  76. 195.]
 [195. 162. 244. 204. 143.]
 [ 25.  83. 136. 255.  28.]
 [101.   8. 241. 205. 150.]]
Guess Template:
('Rank: ', 1, 'X: ', 183, 'Y: ', 295, 'NCC-value: ', 1.0)
('Job finish in ', 8.053385019302368)
```

1846

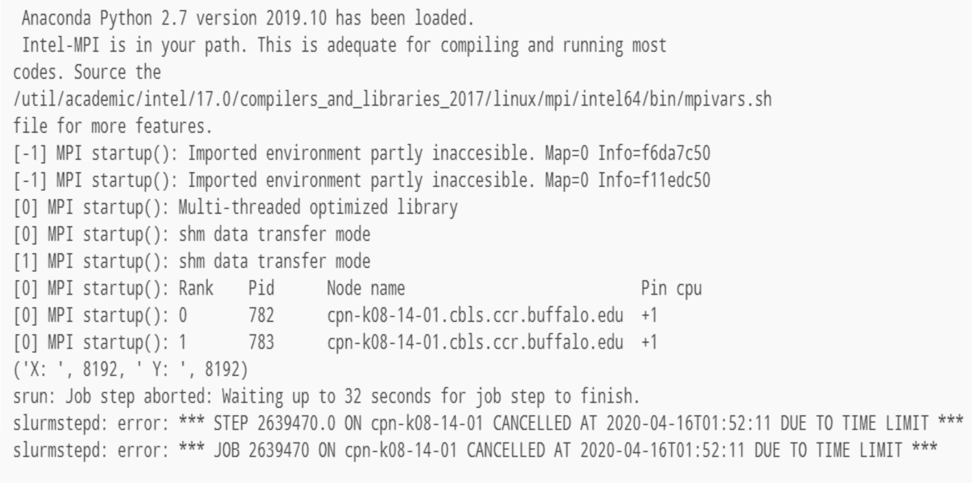
Results: 16 Nodes



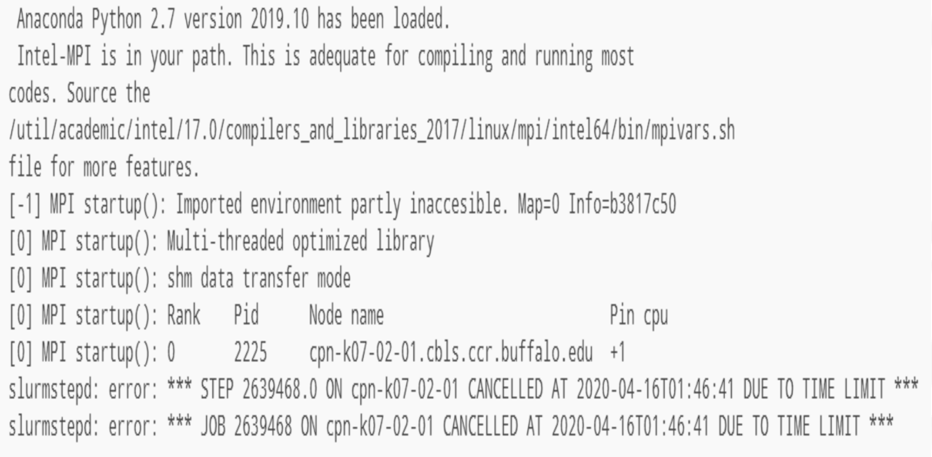
```
('Template: (', 167, ',', 492, ')')
Template:
[[253.  85.  89.  29.  30.]
 [226.  98. 177. 142.  56.]
 [ 87.  55.   6. 214.  90.]
 [203. 248.  48. 134. 232.]
 [  2. 171. 223.  32. 185.]]
Guess Template:
('Rank: ', 5, 'X: ', 167, 'Y: ', 492, 'NCC-value: ', 1.0)
('Job finish in ', 1.6863749027252197)
```

Challenges

- Input data. When memory is not sufficient, creates many bugs and error.
- Running a large file, sometimes the scheduler doesn’t stop, and it will continued running on the time you set on.
- Coding with MPI library, very hard to keep track each nodes and their data. Ex. When to stop executing each node, so that they receive all the data they needed.
- Currently working on template matching, where I have a 1024x1024 matrix size, and a 27x27 template. Got stuck when, the template is on more than 4 nodes.



```
 Anaconda Python 2.7 version 2019.10 has been loaded.
 Intel-MPI is in your path. This is adequate for compiling and running most
codes. Source the
/util/academic/intel/17.0/compilers_and_libraries_2017/linux/mpi/intel64/bin/mpivars.sh
file for more features.
[-1] MPI startup(): Imported environment partly inaccesible. Map=0 Info=f6da7c50
[-1] MPI startup(): Imported environment partly inaccesible. Map=0 Info=f11edc50
[0] MPI startup(): Multi-threaded optimized library
[0] MPI startup(): shm data transfer mode
[1] MPI startup(): shm data transfer mode
[0] MPI startup(): Rank    Pid      Node name                          Pin cpu
[0] MPI startup(): 0       782      cpn-k08-14-01.cbls.ccr.buffalo.edu  +1
[0] MPI startup(): 1       783      cpn-k08-14-01.cbls.ccr.buffalo.edu  +1
('X: ', 8192, ' Y: ', 8192)
srun: Job step aborted: Waiting up to 32 seconds for job step to finish.
slurmstepd: error: *** STEP 2639470.0 ON cpn-k08-14-01 CANCELLED AT 2020-04-16T01:52:11 DUE TO TIME LIMIT ***
slurmstepd: error: *** JOB 2639470 ON cpn-k08-14-01 CANCELLED AT 2020-04-16T01:52:11 DUE TO TIME LIMIT ***
```



```
 Anaconda Python 2.7 version 2019.10 has been loaded.
 Intel-MPI is in your path. This is adequate for compiling and running most
codes. Source the
/util/academic/intel/17.0/compilers_and_libraries_2017/linux/mpi/intel64/bin/mpivars.sh
file for more features.
[-1] MPI startup(): Imported environment partly inaccesible. Map=0 Info=b3817c50
[0] MPI startup(): Multi-threaded optimized library
[0] MPI startup(): shm data transfer mode
[0] MPI startup(): Rank    Pid      Node name                          Pin cpu
[0] MPI startup(): 0       2225     cpn-k07-02-01.cbls.ccr.buffalo.edu  +1
slurmstepd: error: *** STEP 2639468.0 ON cpn-k07-02-01 CANCELLED AT 2020-04-16T01:46:41 DUE TO TIME LIMIT ***
slurmstepd: error: *** JOB 2639468 ON cpn-k07-02-01 CANCELLED AT 2020-04-16T01:46:41 DUE TO TIME LIMIT ***
```

References

- UB CCR Support Home page.
- https://mpi4py.readthedocs.io/en/stable/tutorial.html
- https://pypi.org/project/mpi4py/

Thank you!