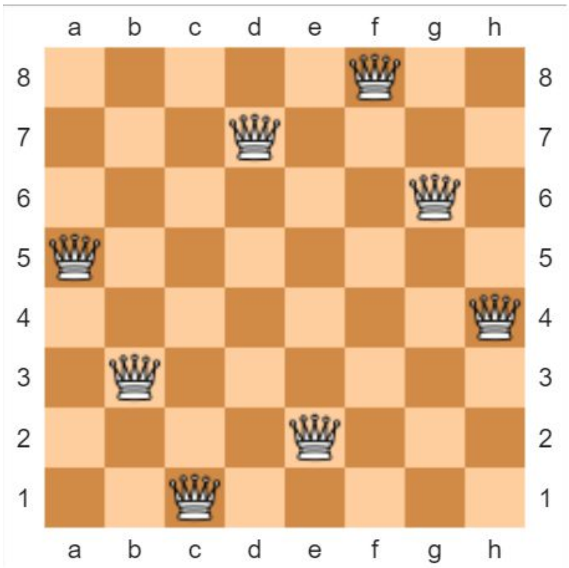
CS-301
N-QUEEN

YATIN PATEL 201601454
JAYKISHAN PARMAR 201601458

1. Introduction

In N-queen problem we have to place n queens in NXN chess board such that no queens attack each other. In other words we have to place n queens in NXN board such that no two queens share its column,row or diagonal. We can place a queen in chess board with given condition with many configuration. One of them for 8X8 board is shown below.



Now we want how many total configuration can make such that no queen attack each other. For Example if N=4 then total number of configuration is 2 given below.

	Q1		
			Q2
Q3			
		Q4	

Solution 1

		Q1	
Q2			
			Q3
	Q4		

Solution 2

2. Hardware Details

CPU(s):	:6
Thread per core	:1
L1 cache	:32K
L2 cache	:256K
L3 cache	:20480K

6. Algorithm Technique

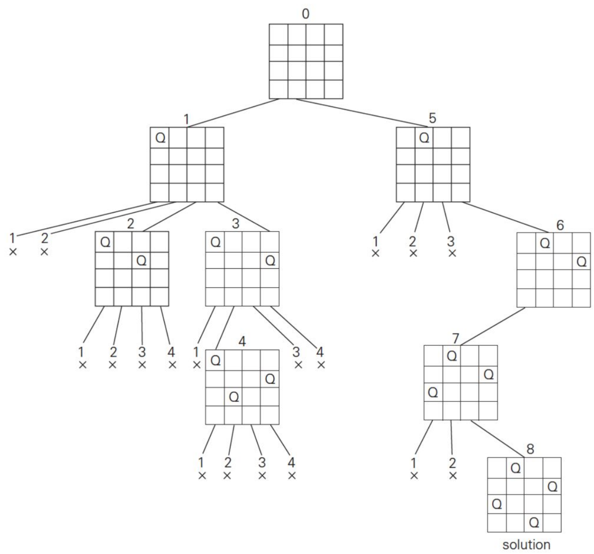
Naive Approach

It is a brute force solution in which you have to check all the Combination of queens and check the given configuration whether it is correct or not. This is very time taking approach. Because in this approach we have to check N^2C_N configuration.

Backtracking

We are putting the queen in row one by one. First we put the queen in first row.

And then go to the second row. And for second row we will check for each column of



Backtracking Approach

the second row and check for already placed queen. If in the current square none of the queen is attacking then we put the queen in this particular square and call recurrence function for the next row. If we didn't find any column for the particular row where we can put the queen then we simply return the 0.

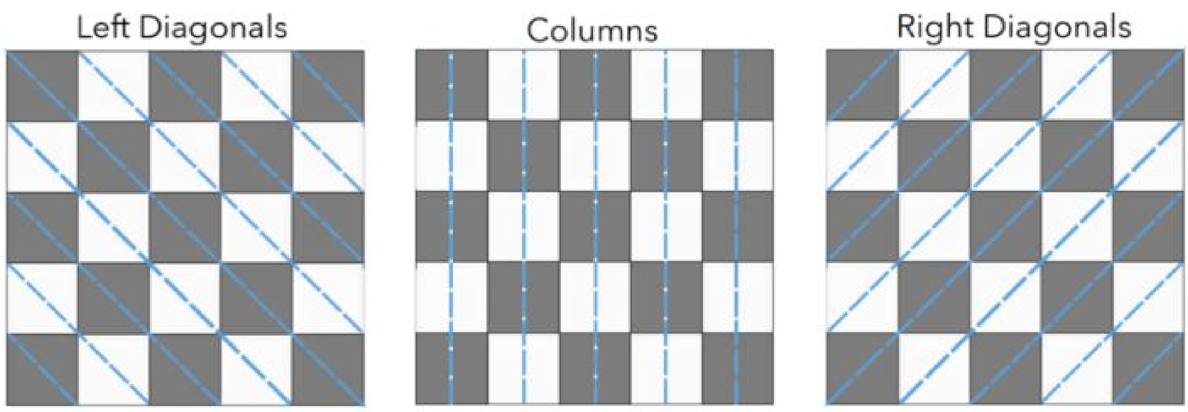
Bit Masking Approach

In Backtracking approach we have to check for each row and column such that no previous queen attack current square. To overcome this problem we create the solution of n-queen problem using bit masking.

What is bit masking?

Bit masking is method in which we hide some data in binary numbers. Suppose we have an array of 5 length A = {1,2,3,4,5}. Now we take the subset of the array {1,3,5}. To represent this subset we just store one number with 1st,3rd and 5th bit on(10101). Now if we have array of 20 numbers we have to just store one binary number with length of 20.

Bit masking in N-queen problem



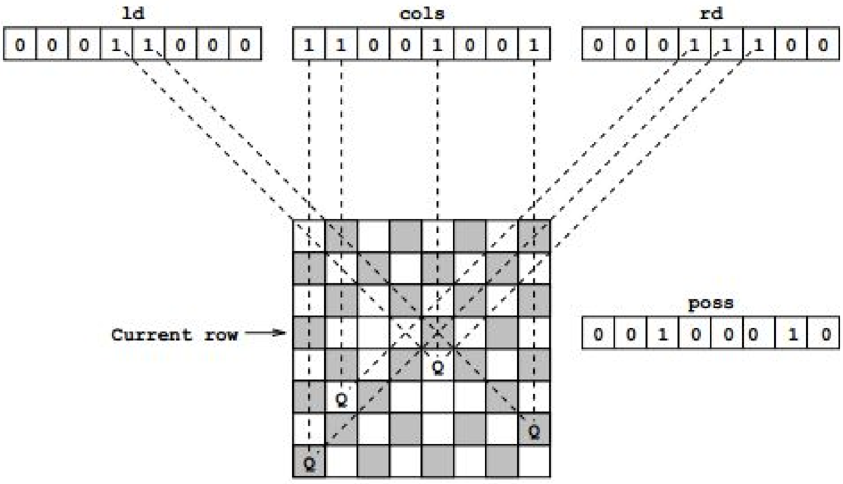
So for bit masking in n-queen problem we have to keep track of three array Left Diagonals,Column and Right Diagonals. Here 1 denotes that queen is attacking on this square. And 0 denotes the queen is not attacking for particular square.

Now form three array how we can conclude that queen is attacking or not on a particular square in particular row. For this we make this equation

Bitmap = (mask & ~(left | right | column))

(initially left=right=column=0 and mask = 2^N - 1)

Here bitmap is binary number with size of n length. If we have at ith row then we can put the queen in jth column if jth bit of bitmap is on.



If we put the queen in jth column then we have to set bit of the left,right and column. So for that

Temp = 1 << j;

Column = Column | Temp

Right = (Right | Temp) >> 1

Left = (Left | Temp) << 1

And call recurrence function for next row

Sum = Sum + recur (row + 1,Column,Right,Left);

<u>Solution of N-queen</u>

```
4  - 2
5  - 10
6  - 4
7  - 40
8  - 92
9  - 352
10 - 724
11 - 2680
12 - 14200
13 - 73712
14 - 365596
15 - 2279184
16 - 14772512
17 - 95815104
18 - 666090624
19 - 4968057848
20 - 39029188884
21 - 314666222712
22 - 2691008701644
23 - 24233937684440
24 - 227514171973736
25 - 2207893435808352
26 - 22317699616364044
27 - 234907967154122528
28 - NOT FOUND YET
```

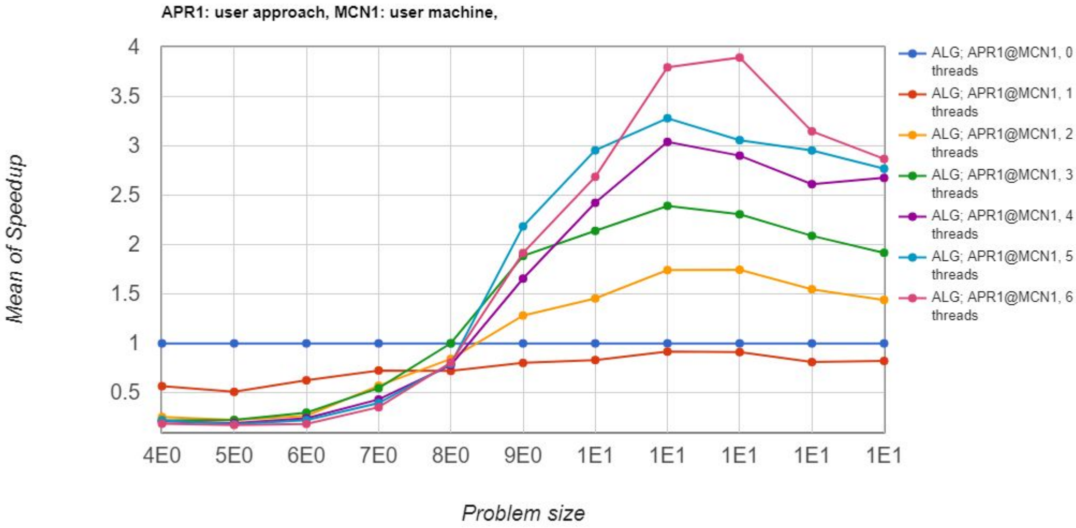
Scope of parallelization

Here, in each backtrack iteration is only dependant on its previous queens configurations means in row 2 iteration, it only needed that where is queen is placed in row 1 and which column. So, here we are trying by different methods to divide whole backtracking search into p processors such that work load among p processors equally divided so that more speedup will get.

Parallelization strategies, and Results-quantitative analysis-discussion with graphs and supporting justifications

<u>Parallelizing Backtracking (Method 1)</u>

In this method we generate new task for each thread and backtrack it.
Here, for every backtrack task is generated and added to queue and from that each thread takes them and executes. Here, exact timing of executing tasks in queue is up to task scheduler. Here, task switching cost is more.

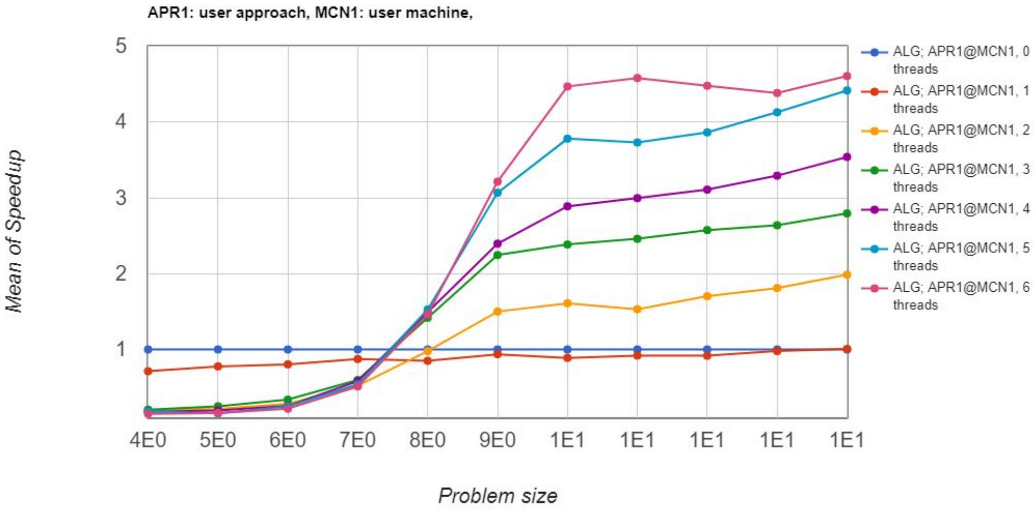


Parallelizing Backtracking (Method 2)

In this method we divide work division as each thread (0 to p) will generate solutions in which first queen which is at first row will be at ith column same as thread id.

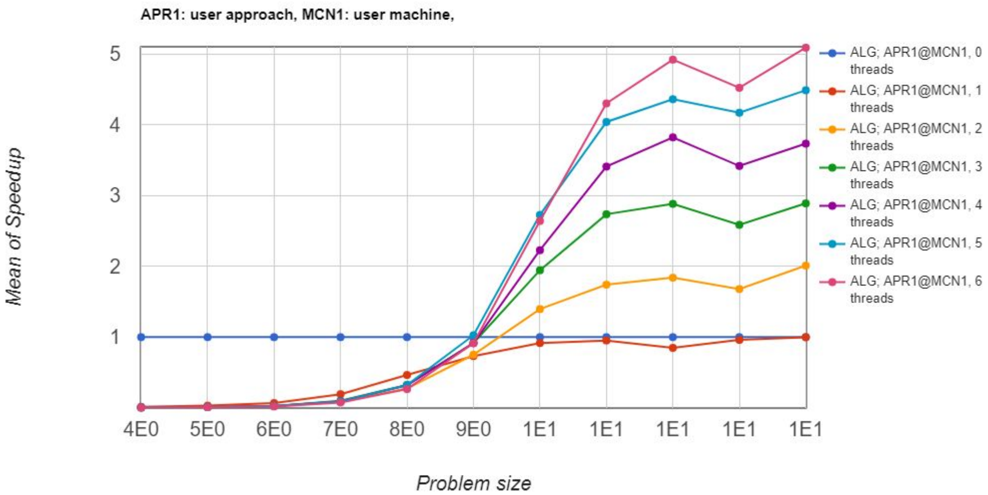
Here, is speedup is better but not that much beacuse of the asymmetry in the number of the solution that start at first quuen at ith column. For n = 14 distribution are as follow.

```
436228 569531 736363 892999 1050762 1160280 1249262 1290831 1290831 1249262 1160280 1050762 892999 736363 569531 436228
Total soluitons: 14772512
```



Parallelizing Backtracking (Method 3)

This method has solution of previous two's methods problems. In this problem we create new child for certain level and deeper than this level search linearly. So that we can get more efficient algorithm.



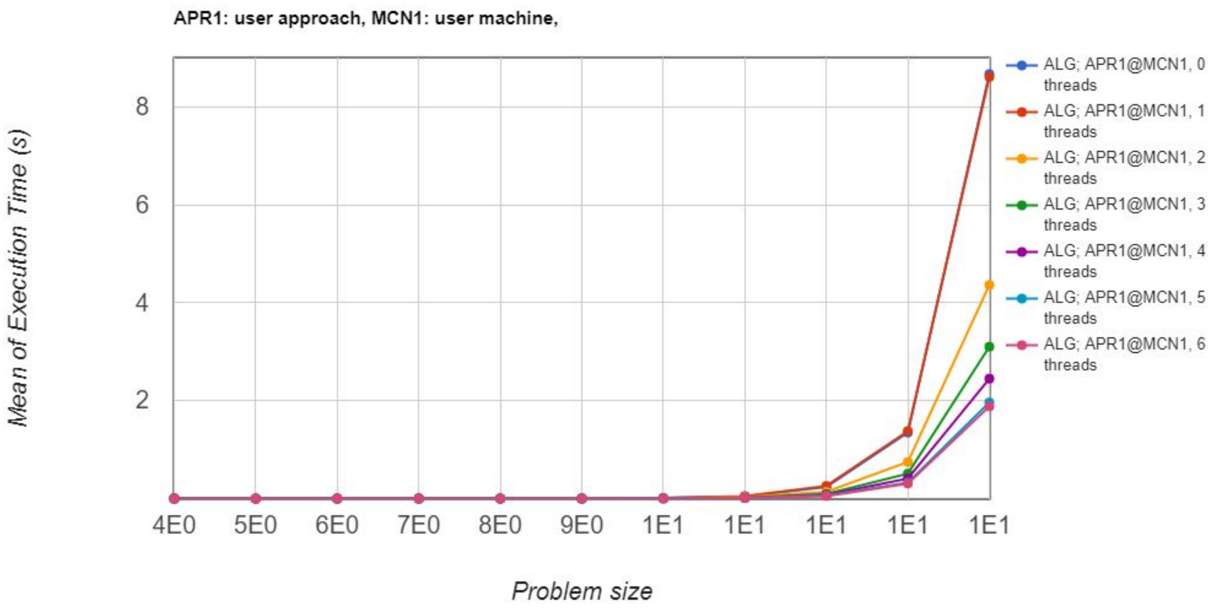
Parallelizing Bitmasking(Method 4)

We can parallelize bit masking backtracking using reduction.
Snippet of the algorithm is shown below

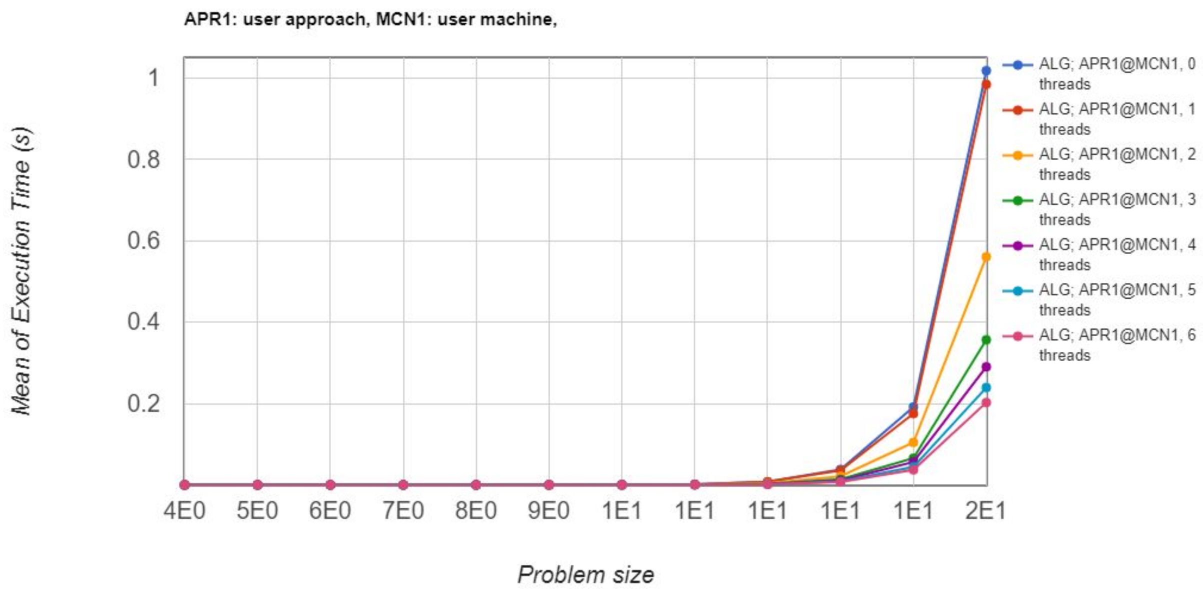
```
#pragma omp parallel for schedule (dynamic:chunk) reduction(+:count)
        for (i=0;i<n;i++)
                if((bitmap>>i) & 1)
                        bit=1<<i;
                        Count += backtrack(row + 1,column|bit,(left|bit)<<1,(right|bit)>>1);
```

Bit masking is much faster than normal backtracking because in this algorithm we don't have to check whether we can put the queen in this column or not. It is store in bit variable.

<u>Comparison of execution time of backtracking and bit masking</u>



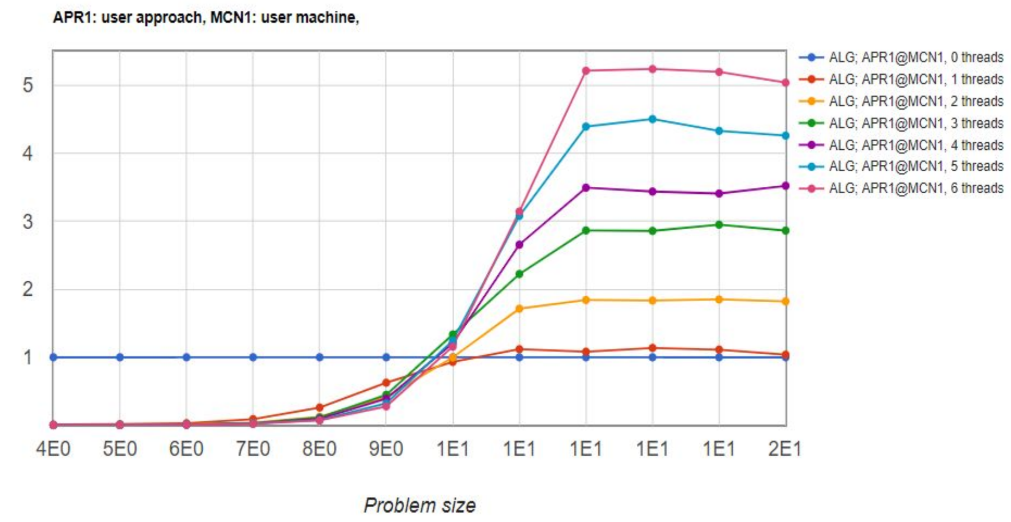
Backtracking



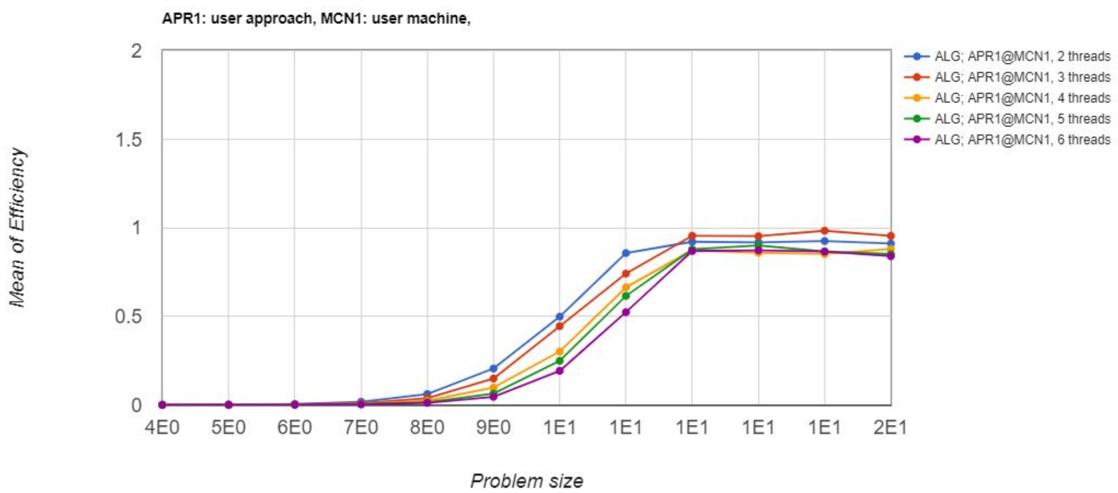
Bit masking

From figure we can see that execution of time for n=14 and backtracking method is 8 second and for bit masking method it takes 1 second. From observation we can conclude that time difference is increasing when N increasing.

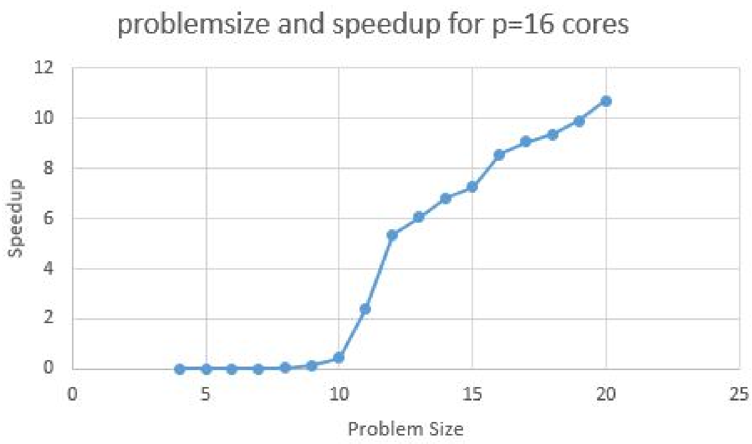
<u>Speedup curve and efficiency curve for bit masking method</u>



Speed Up curve for Bit Masking method.



Efficiency Curve for Bit masking method.



Karp-Flatt metric related analysis

In Karp Flatt analysis, overhead time is taken into consideration for calculating the speed up.

$$e = \frac{(Total\ Amount\ of\ idel\ time + overhaed\ time)}{(p-1)T(n,1)}$$

$$e = \frac{(\frac{1}{\psi} - \frac{1}{p})}{(1 - \frac{1}{p})}$$

No. of processors	2	3	4	5	6
Speedup	**1.842**	**2.864**	**3.492**	**4.39**	**5.211**
e(Estimation of serial fraction)	**0.0857**	**0.0237**	**0.0484**	**0.0347**	**0.0302**

Future Scope of improvement

Here, we have just considered flip symmetry so there are many repetitions that we are computing while computing total answers. In future work we can consider 90, 180, 270 degree symmetries and can compute answer in lesser time but this will not increase the total overall speedup. Second thing that we can impover is that we can change shared memory parallelism to distributed memory parallelism.

Conclude

Here, from above all methods, best method to find N-Queen Number is backtracking using bitmasking. Here, from above techniques we can see that as we increase input number the speedup for 16 thread is increasing.

References

1.) N queen GeeksForGeeks
2.) Wikipedia
3.) N queens using bitmasking article
4.) N queen backtrack + bitmask article
5.) parallel programming with intel parallel studio xe (Chapter-10).