

1 What sequence of computations and communications is performed for each batch?

1. **find all sum** is called. It use node weight to compute the sum and the accumulated sum for each node and its neighbor. These results will be used to determine the rat movement later on. Only the node in the current zone will be iterated on.

2. the movement of rat is calculated. Only the rat in the current batch and in the current zone is calculated. Rats that leave the zone are recorded. The index of rats in the zone is maintained in an array. The index of rats that leave the zone is store in another array, along with their destination node.

3. **exchange rat** is called. Each process sends out the information of rats that leave the zone and also receive the information of rats that come into this zone. The position, rat count, rat seed and the index of rats in the zone are updated.

4. **exchange node state** is called. Each process sends out the node count of its export nodes to its adjacent zone and receive the node count of its import nodes. The rat count is then updated.

5. **compute cur weights** is called. It calculates the node weights for nodes in the current zone, using rat count.

6. **exchange node weight** is called. Each process sends out the node weights of its export nodes to its adjacent zone and receive the node weight of its import nodes. The node weight is then updated.

2 How did you maximize the decoupling of processes to avoid waiting for messages from each other.

1. Use Isend and Irecv for all the point to point communication. Use Gather for the communication that goes form other processes to the root process. And use Bcast when the program need to distribute information from the root process to other process.

2. Simplify the buffer that need to be sent among processes.

For **exchange rat**, rat index, rat destination node and rat seed are all packed into one long array. An offset index for each zone is recorded and used to send out the buffer. Accordingly, on the receive side, an offset index is first fetched by using Probe function to get the size of the message. Then all the message from other zones are pack into one long array.

For **exchange node state** and **exchange node weight**, all the information are packed into a long array using the order from export node list. Then the message is sent. On the receive side, all the message is pack into one array too. Then it uses the order from import node list to unpack the information.

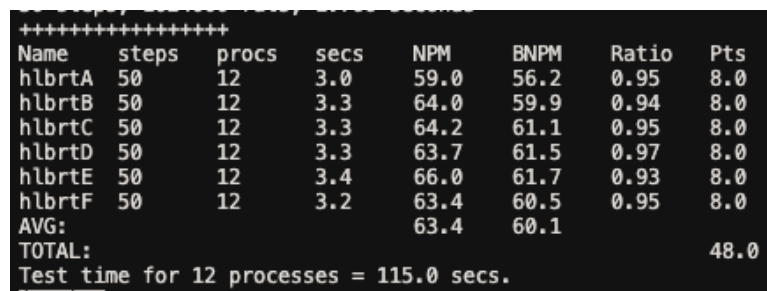
3. simplify the function like find all sum, rat movement and compute all weights to speed up the preparation for the message passing.

For **find all sum** and **compute all weight**, only the nodes in the current zone are calculated using the local node list.

For **rat movement**, an array that contains the index of rat in the current zone is maintained. The array is split by the batch size to store rats in different batch and in each split, the rat index is placed continuously.

3 How successful were you in getting speedup in your program? (This should be backed by experimental measurements.)

Full marks is obtained on Latedays.



```
+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50    12    3.0   59.0  56.2  0.95   8.0
hlbrtB 50    12    3.3   64.0  59.9  0.94   8.0
hlbrtC 50    12    3.3   64.2  61.1  0.95   8.0
hlbrtD 50    12    3.3   63.7  61.5  0.97   8.0
hlbrtE 50    12    3.4   66.0  61.7  0.93   8.0
hlbrtF 50    12    3.2   63.4  60.5  0.95   8.0
AVG:                63.4  60.1
TOTAL:                                     48.0
Test time for 12 processes = 115.0 secs.
```

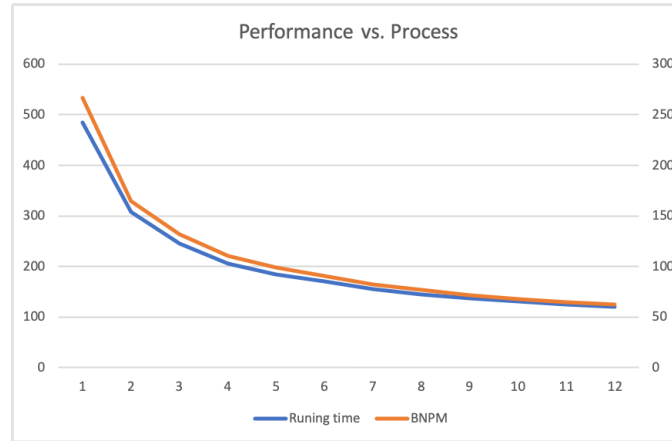
Figur 1: Outcome

4 How does the performance scale as you went from 1 to 12 processes?

The margin of improvement is getting smaller when the number of process grows. The more processors it has, the less work load for each processor and the more communication needed among different processor. Since the algorithm contains lots of point to point communication, the growth of the communication is a second order of the growth of the number of processor. The result shows that the growth of communication work load gradually cancel out the benefit of decreasing local computing work load for each process.

5 How important is having an even load balance vs. minimizing communication for the different benchmarks?

For benchmark A and C, the standard deviation of their region node count is below 120. So it is easier for them to have a balanced node count distribution when using the interval



separations method. It is more important to alleviate the communication among zones by making the zone continuous.

For benchmark D,E,F, their standard deviation is above 300. The various of region side make it hard to have a balanced node count for each zone if we use the interval separation method. Therefore, I used the greedy method, because it is more important to have a balanced node count and a balanced load for each process than alleviating the communication now.

For benchmark B, it has a deviation of 123 for the region node count. It is in the middle range where both the load balance and the communication need to be considered. If we want to alleviate the communication, we should make continuous region into one zone. If we want a balanced load, we should allocate the region according to the node count of each zone greedily. So the greedy algorithm is modified. If the assigning the region greedily can improve the load balance a lot, the algorithm will assign it, otherwise, it is assign to its previous zone for a better communication. The criterion here is whether whether assigning greedily make the assigned zone have the same or greater amount of calculation than the previous zone.

6 Were there any techniques that you tried but found ineffective?

1. Sending out rat index, rat count, rat seed independently is slow.
2. Storing rat count, node weight in different buffer for each zone is memory-consuming.
3. Using Isend/Irecv is much slower than using Bcast and Gather.
4. Iterating on rat by the index and then determining whether the rat is in the current zone is slow. It is better to store the index of the rat in the zone explicitly. Since the rat index is iterated in batch, index should be separated into different parts and each part represents rats in the same batch. Then within the batch, the order of index doesn't matter.

7 All results

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  1  11.4  223.2  232.2  1.04  8.0
hlbrtB 50  1  13.4  262.6  271.3  1.03  8.0
hlbrtC 50  1  13.8  270.2  279.0  1.03  8.0
hlbrtD 50  1  13.4  262.5  271.2  1.03  8.0
hlbrtE 50  1  13.7  266.9  275.5  1.03  8.0
hlbrtF 50  1  13.4  260.9  269.7  1.03  8.0
AVG:      257.7  266.5
TOTAL:      48.0
Test time for 1 processes = 485.3 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  3  6.0  118.0  118.9  1.01  8.0
hlbrtB 50  3  6.9  135.1  132.4  0.98  8.0
hlbrtC 50  3  7.0  136.9  135.0  0.99  8.0
hlbrtD 50  3  6.9  133.9  132.5  0.99  8.0
hlbrtE 50  3  7.0  135.8  137.1  1.01  8.0
hlbrtF 50  3  6.9  134.5  133.9  1.00  8.0
AVG:      132.4  131.6
TOTAL:      48.0
Test time for 3 processes = 244.9 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  5  4.6  90.2  89.2  0.99  8.0
hlbrtB 50  5  5.1  100.4  98.9  0.99  8.0
hlbrtC 50  5  5.3  102.9  101.0  0.98  8.0
hlbrtD 50  5  5.2  100.9  99.9  0.99  8.0
hlbrtE 50  5  5.2  101.8  102.5  1.01  8.0
hlbrtF 50  5  5.2  100.9  104.0  1.03  8.0
AVG:      99.5  99.2
TOTAL:      48.0
Test time for 5 processes = 184.7 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  7  3.9  77.1  75.1  0.97  8.0
hlbrtB 50  7  4.4  86.4  82.9  0.96  8.0
hlbrtC 50  7  4.4  86.6  85.0  0.98  8.0
hlbrtD 50  7  4.5  87.7  83.3  0.95  8.0
hlbrtE 50  7  4.6  89.0  83.6  0.94  8.0
hlbrtF 50  7  4.3  84.4  83.0  0.98  8.0
AVG:      85.2  82.1
TOTAL:      48.0
Test time for 7 processes = 155.8 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  9  3.5  67.9  65.3  0.96  8.0
hlbrtB 50  9  3.9  76.2  71.9  0.94  8.0
hlbrtC 50  9  3.9  76.5  72.7  0.95  8.0
hlbrtD 50  9  3.8  74.5  73.1  0.98  8.0
hlbrtE 50  9  3.9  75.7  74.4  0.98  8.0
hlbrtF 50  9  4.0  77.2  72.5  0.94  8.0
AVG:      74.7  71.6
TOTAL:      48.0
Test time for 9 processes = 136.6 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  11 3.2  63.0  60.6  0.96  8.0
hlbrtB 50  11 3.5  68.8  65.8  0.96  8.0
hlbrtC 50  11 3.6  70.6  66.4  0.94  8.0
hlbrtD 50  11 3.5  68.6  65.3  0.95  8.0
hlbrtE 50  11 3.6  70.3  65.9  0.94  8.0
hlbrtF 50  11 3.5  68.1  65.8  0.97  8.0
AVG:      68.2  65.0
TOTAL:      48.0
Test time for 11 processes = 124.1 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  2  7.6  148.7  147.3  0.99  8.0
hlbrtB 50  2  8.6  167.9  167.1  1.00  8.0
hlbrtC 50  2  8.7  169.0  168.4  1.00  8.0
hlbrtD 50  2  8.8  172.2  169.7  0.99  8.0
hlbrtE 50  2  8.7  170.8  168.6  0.99  8.0
hlbrtF 50  2  8.6  167.9  166.6  0.99  8.0
AVG:      166.1  164.6
TOTAL:      48.0
Test time for 2 processes = 307.3 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  4  5.1  99.2  98.6  0.99  8.0
hlbrtB 50  4  5.8  113.6  111.4  0.98  8.0
hlbrtC 50  4  5.9  115.0  113.1  0.98  8.0
hlbrtD 50  4  5.9  115.2  112.4  0.98  8.0
hlbrtE 50  4  5.9  114.3  112.0  0.98  8.0
hlbrtF 50  4  5.8  113.1  113.1  1.00  8.0
AVG:      111.7  110.1
TOTAL:      48.0
Test time for 4 processes = 205.9 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  6  4.3  83.1  81.9  0.99  8.0
hlbrtB 50  6  4.8  94.7  91.9  0.97  8.0
hlbrtC 50  6  4.8  94.3  92.1  0.98  8.0
hlbrtD 50  6  4.9  95.6  92.4  0.97  8.0
hlbrtE 50  6  4.9  96.3  93.3  0.97  8.0
hlbrtF 50  6  4.7  92.3  93.9  1.02  8.0
AVG:      92.7  90.9
TOTAL:      48.0
Test time for 6 processes = 171.1 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  8  3.7  71.5  70.2  0.98  8.0
hlbrtB 50  8  4.1  79.5  76.0  0.96  8.0
hlbrtC 50  8  4.1  80.2  77.8  0.97  8.0
hlbrtD 50  8  4.1  80.2  78.9  0.98  8.0
hlbrtE 50  8  4.1  80.5  78.2  0.97  8.0
hlbrtF 50  8  4.1  80.0  77.9  0.97  8.0
AVG:      78.6  76.5
TOTAL:      48.0
Test time for 8 processes = 144.4 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  10 3.4  66.3  63.7  0.96  8.0
hlbrtB 50  10 3.7  71.5  67.6  0.95  8.0
hlbrtC 50  10 3.8  73.4  69.5  0.95  8.0
hlbrtD 50  10 3.7  71.3  68.9  0.97  8.0
hlbrtE 50  10 3.7  72.4  69.6  0.96  8.0
hlbrtF 50  10 3.7  72.1  68.5  0.95  8.0
AVG:      71.2  68.0
TOTAL:      48.0
Test time for 10 processes = 130.1 secs.

```

```

+++++
Name  steps  procs  secs  NPM  BNPM  Ratio  Pts
hlbrtA 50  12 3.2  62.1  58.1  0.94  8.0
hlbrtB 50  12 3.4  66.4  62.3  0.94  8.0
hlbrtC 50  12 3.4  67.0  63.0  0.94  8.0
hlbrtD 50  12 3.4  66.6  64.0  0.96  8.0
hlbrtE 50  12 3.5  68.8  64.1  0.93  8.0
hlbrtF 50  12 3.3  65.4  63.5  0.97  8.0
AVG:      66.1  62.5
TOTAL:      48.0
Test time for 12 processes = 120.4 secs.

```