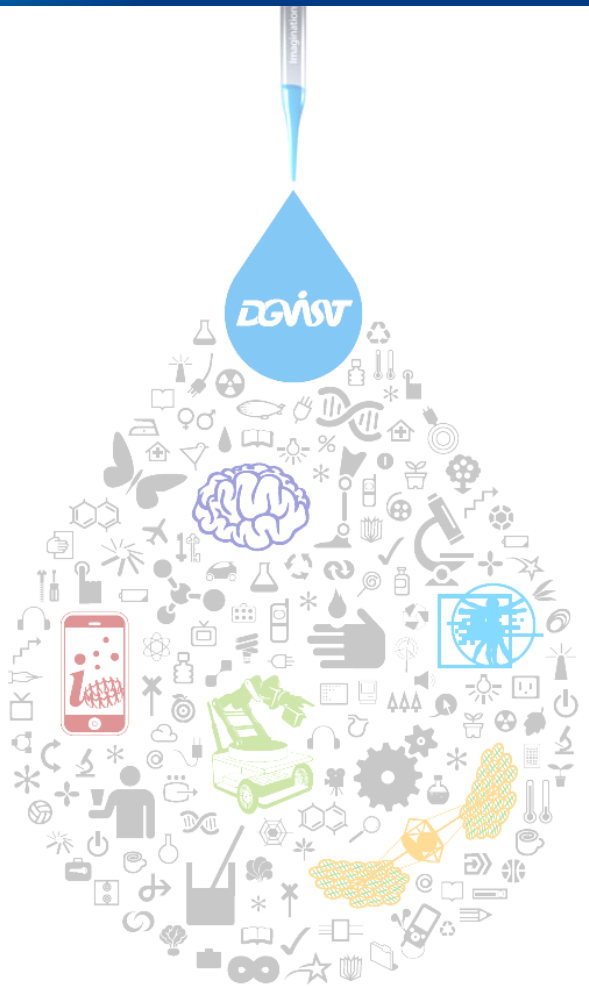
InfoLab

IC621: Distributed and Parallel Computing

Lab 03: MPI (sort)



Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Parallel Odd-Even Transposition Sort

- **Implement parallel odd-even transposition sort by using either**
  - a pair of `MPI_Send` and `MPI_Recv`, or
  - `MPI_Sendrecv`

- **Perform experiments**
  - data : random numbers of the integer type
    - four data sets : 2M, 4M, 8M, 16M
  - # processes : {1, 2, 4, 8}

- **Calculate speedup and efficiency and decide your program is scalable or not**

Submission

- **Source code**

- **Report**
  - elapsed time table
  - speed up table
  - efficiency table
  - your conclusion about the scalability of your program

	# of keys (in millions)			
# processes	2M	4M	8M	16M
1				
2				
4				
8				

- **Submit your codes by 1:00pm, 10/10(Tue) to TAs (bm010515@dgist.ac.kr, chan150@dgist.ac.kr)**

Serial version

```
1  void Odd_even_sort(
2         int  a[]  /* in/out */,
3         int  n    /* in     */) {
4     int phase, i, temp;
5
6     for (phase = 0; phase < n; phase++)
7        if (phase % 2 == 0) { /* Even phase */
8           for (i = 1; i < n; i += 2)
9              if (a[i-1] > a[i]) {
10                temp = a[i];
11                a[i] = a[i-1];
12                a[i-1] = temp;
13             }
14       } else { /* Odd phase */
15          for (i = 1; i < n-1; i += 2)
16             if (a[i] > a[i+1]) {
17                temp = a[i];
18                a[i] = a[i+1];
19                a[i+1] = temp;
20             }
21       }
22 }  /* Odd_even_sort */
```

Algorithm outline

```
Sort local keys;
for (phase = 0; phase < comm_sz; phase++) {
   partner = Compute_partner(phase, my_rank);
   if (I'm not idle) {
      Send my keys to partner;
      Receive keys from partner;
      if (my_rank < partner)
         Keep smaller keys;
      else
         Keep larger keys;
   }
}
```

Send and Recv (I)

```
if (my_rank % 2 == 0) {
   MPI_Send(msg, size, MPI_INT, (my_rank+1) % comm_sz, 0, comm);
   MPI_Recv(new_msg, size, MPI_INT, (my_rank+comm_sz-1) % comm_sz,
         0, comm, MPI_STATUS_IGNORE).
} else {
   MPI_Recv(new_msg, size, MPI_INT, (my_rank+comm_sz-1) % comm_sz,
         0, comm, MPI_STATUS_IGNORE).
   MPI_Send(msg, size, MPI_INT, (my_rank+1) % comm_sz, 0, comm);
}
```

Send and Recv (II)

```
if (phase % 2 == 0)          /* Even phase */
   if (my_rank % 2 != 0)       /* Odd rank */
      partner = my_rank - 1;
   else                        /* Even rank */
      partner = my_rank + 1;
else                         /* Odd phase */
   if (my_rank % 2 != 0)       /* Odd rank */
      partner = my_rank + 1;
   else                        /* Even rank */
      partner = my_rank - 1;
if (partner == -1 || partner == comm_sz)
   partner = MPI_PROC_NULL;
```

```
MPI_Sendrecv(my_keys, n/comm_sz, MPI_INT, partner, 0,
      recv_keys, n/comm_sz, MPI_INT, partner, 0, comm,
      MPI_Status_ignore);
```

Keeping smaller keys

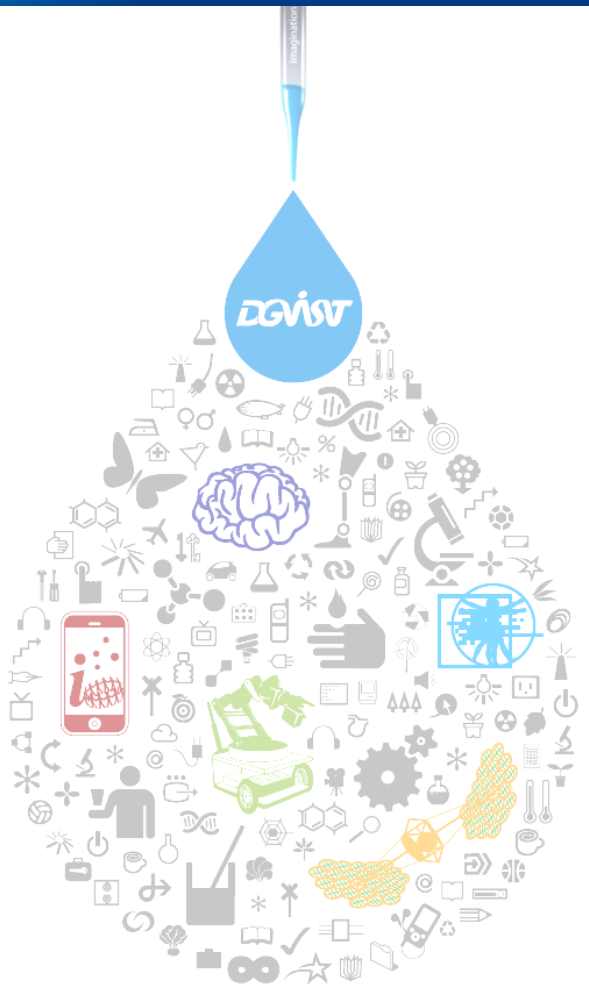
```
void Merge_low(
        int  my_keys[],      /* in/out    */
        int  recv_keys[],    /* in        */
        int  temp_keys[],    /* scratch   */
        int  local_n         /* = n/p, in */) {
   int m_i, r_i, t_i;

   m_i = r_i = t_i = 0;
   while (t_i < local_n) {
      if (my_keys[m_i] <= recv_keys[r_i]) {
         temp_keys[t_i] = my_keys[m_i];
         t_i++; m_i++;
      } else {
         temp_keys[t_i] = recv_keys[r_i];
         t_i++; r_i++;
      }
   }

   for (m_i = 0; m_i < local_n; m_i++)
      my_keys[m_i] = temp_keys[m_i];
}  /* Merge_low */
```

InfoLab

Thank you!



DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology