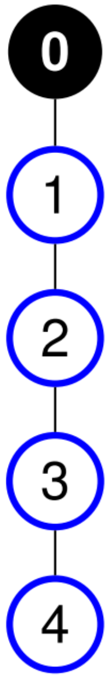
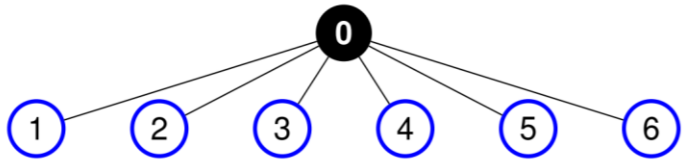
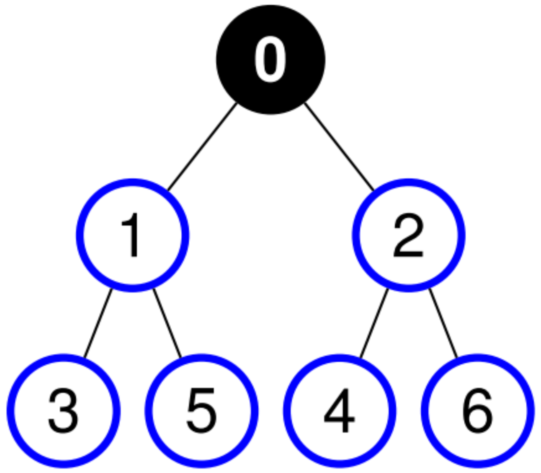
Exercise 2

IMPLEMENTATION OF A BROADCAST ALGORITHM IN A DISTRIBUTED MEMORY


(b) Chain tree


(a) Flat tree


(c) Binary tree

Chosen algorithms

Chain broadcast

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

void chain_broadcast(int *data, int my_rank, int num_procs, int root_rank, int
    num_elements) {
    MPI_Status status;
    int parent_rank = my_rank -  1;
    int child_rank = my_rank +  1;


    if (my_rank == root_rank) {
        MPI_Send(data, num_elements, MPI_INT, child_rank,  0, MPI_COMM_WORLD);
    } else {
        MPI_Recv(data, num_elements, MPI_INT, parent_rank,  0, MPI_COMM_WORLD, &
    status);
        if (child_rank < num_procs) {
            MPI_Send(data, num_elements, MPI_INT, child_rank,  0, MPI_COMM_WORLD)
    ;
        }
    }
}
```

Flat broadcast

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

void flat_tree_broadcast(int *data, int my_rank, int num_procs, int root_rank,
    int  num_elements) {
    MPI_Status status;

    if (my_rank == root_rank) {
            for (int i=1; i<num_procs;i++){
      MPI_Send(data,  num_elements , MPI_INT, i,  0, MPI_COMM_WORLD);
    }}
    else {
        MPI_Recv(data,   num_elements , MPI_INT, root_rank,  0, MPI_COMM_WORLD, &
    status);
    }
}
```

Binary tree

```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

void binary_tree_broadcast(int *data, int my_rank, int num_procs, int root_rank,
    int num_elements) {
    MPI_Status status;
    int parent_rank = (my_rank - 1) / 2;
    int left_child_rank = 2 * my_rank + 1;
    int right_child_rank = 2 * my_rank + 2;

    if (my_rank == root_rank) {
        if (left_child_rank < num_procs)
            MPI_Send(data, num_elements, MPI_INT, left_child_rank, 0,
    MPI_COMM_WORLD);
        if (right_child_rank < num_procs)
            MPI_Send(data, num_elements, MPI_INT, right_child_rank, 0,
    MPI_COMM_WORLD);
    } else {
        MPI_Recv(data, num_elements, MPI_INT, parent_rank, 0, MPI_COMM_WORLD, &
    status);
        if (left_child_rank < num_procs)
            MPI_Send(data, num_elements, MPI_INT, left_child_rank, 0,
    MPI_COMM_WORLD);
        if (right_child_rank < num_procs)
            MPI_Send(data, num_elements, MPI_INT, right_child_rank, 0,
    MPI_COMM_WORLD);
    }
}
```

Strong scaling

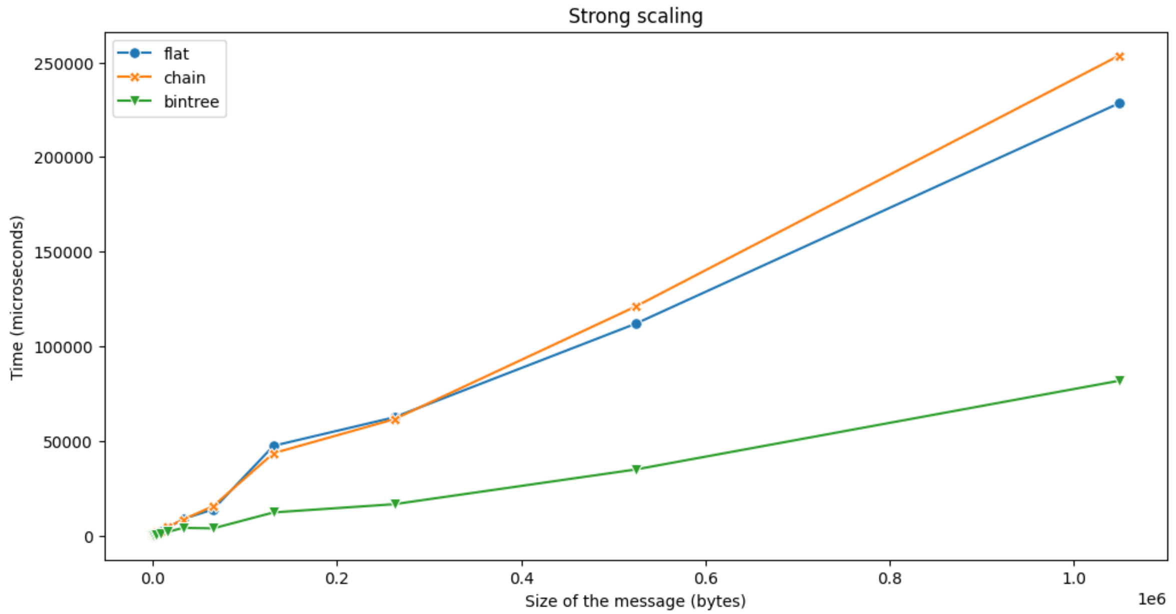


Figure 1: Strong scaling on EPYC node

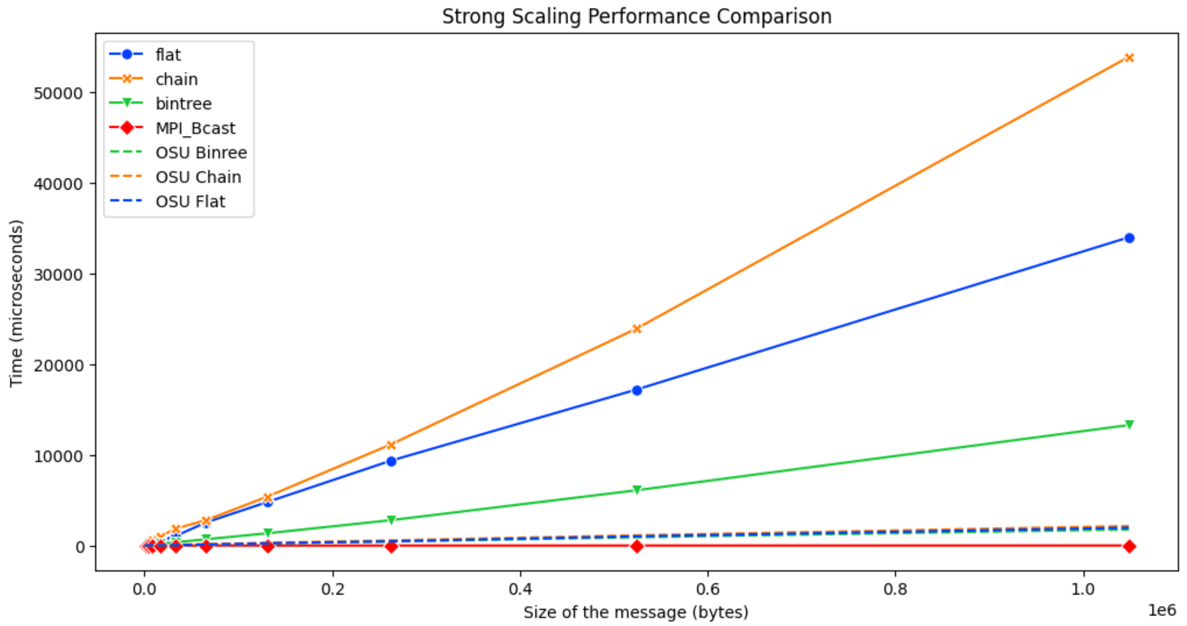


Figure 2: Comparison with MPI

Weak scaling

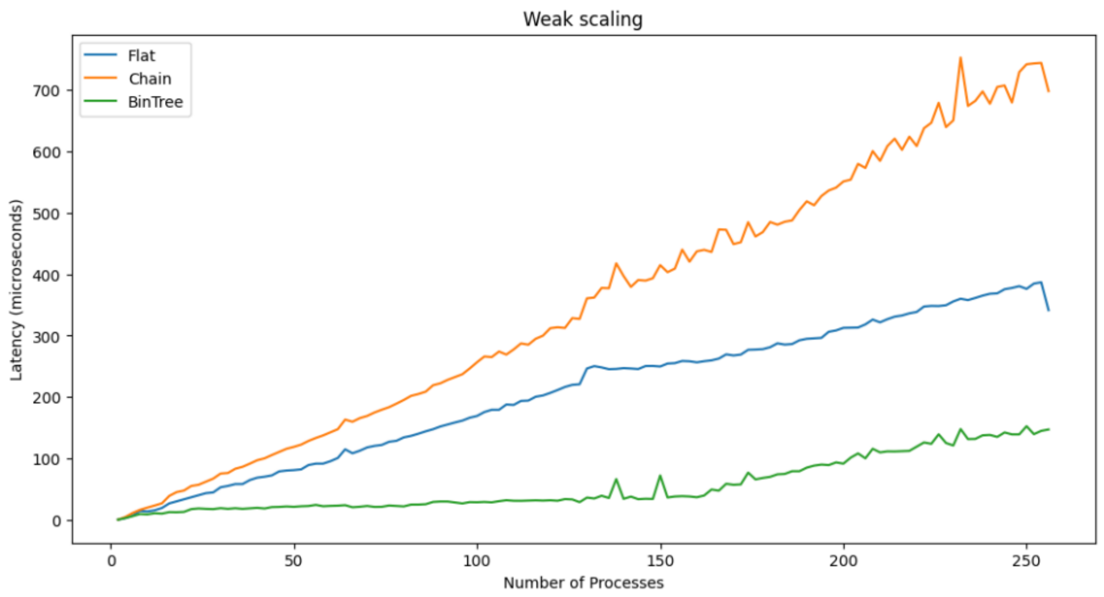


Figure 3: Weak scaling epyc

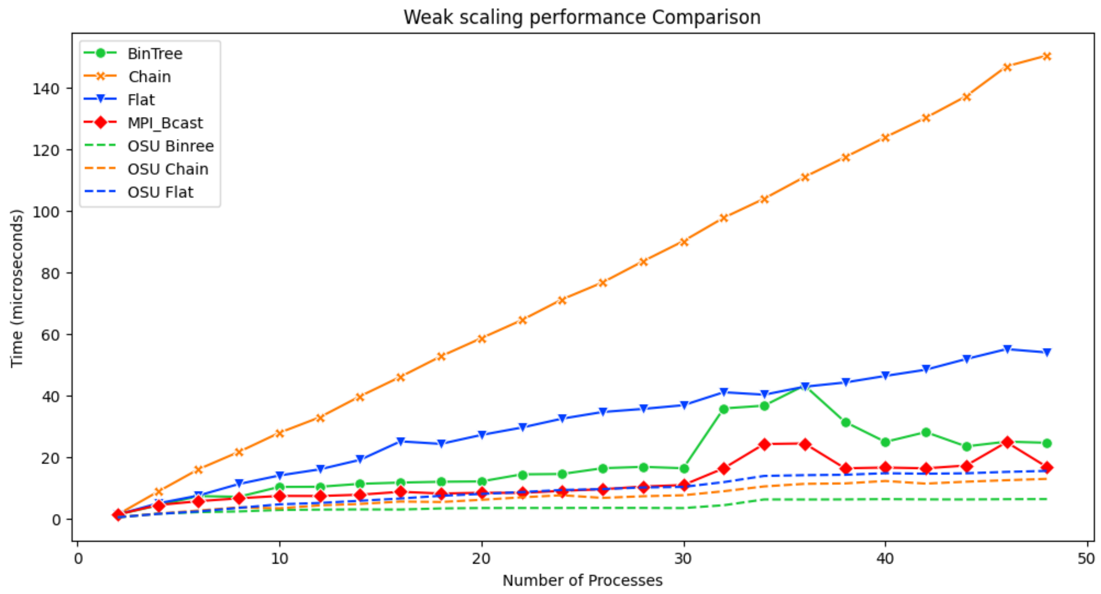


Figure 5: Comparison with MPI