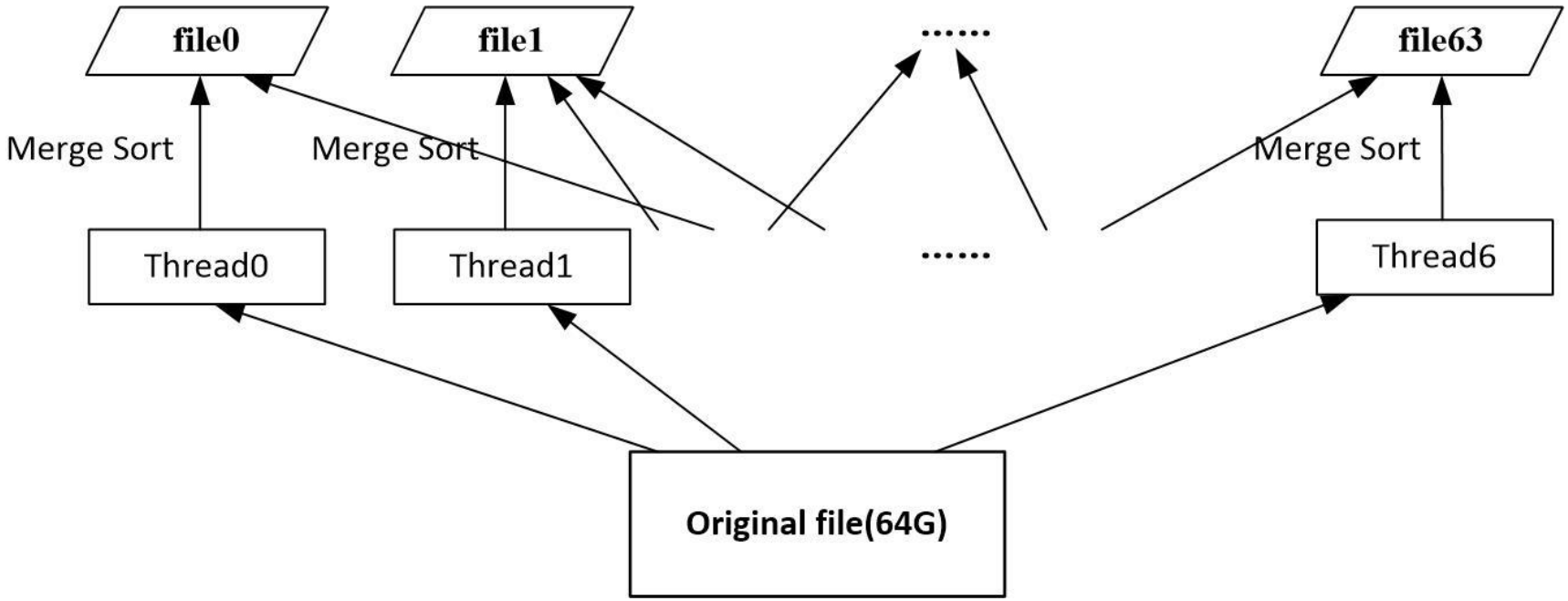
A Multi-thread Sort

Main Idea

- Do sorting on a single machine
  - Avoid data transmission time which might be affected by network
- Use Merge Sort
  - Fast and stable

Multi-thread Merge Sort – Phase 1

Divide all data into 64 parts, merge sort and store them into 64 files.

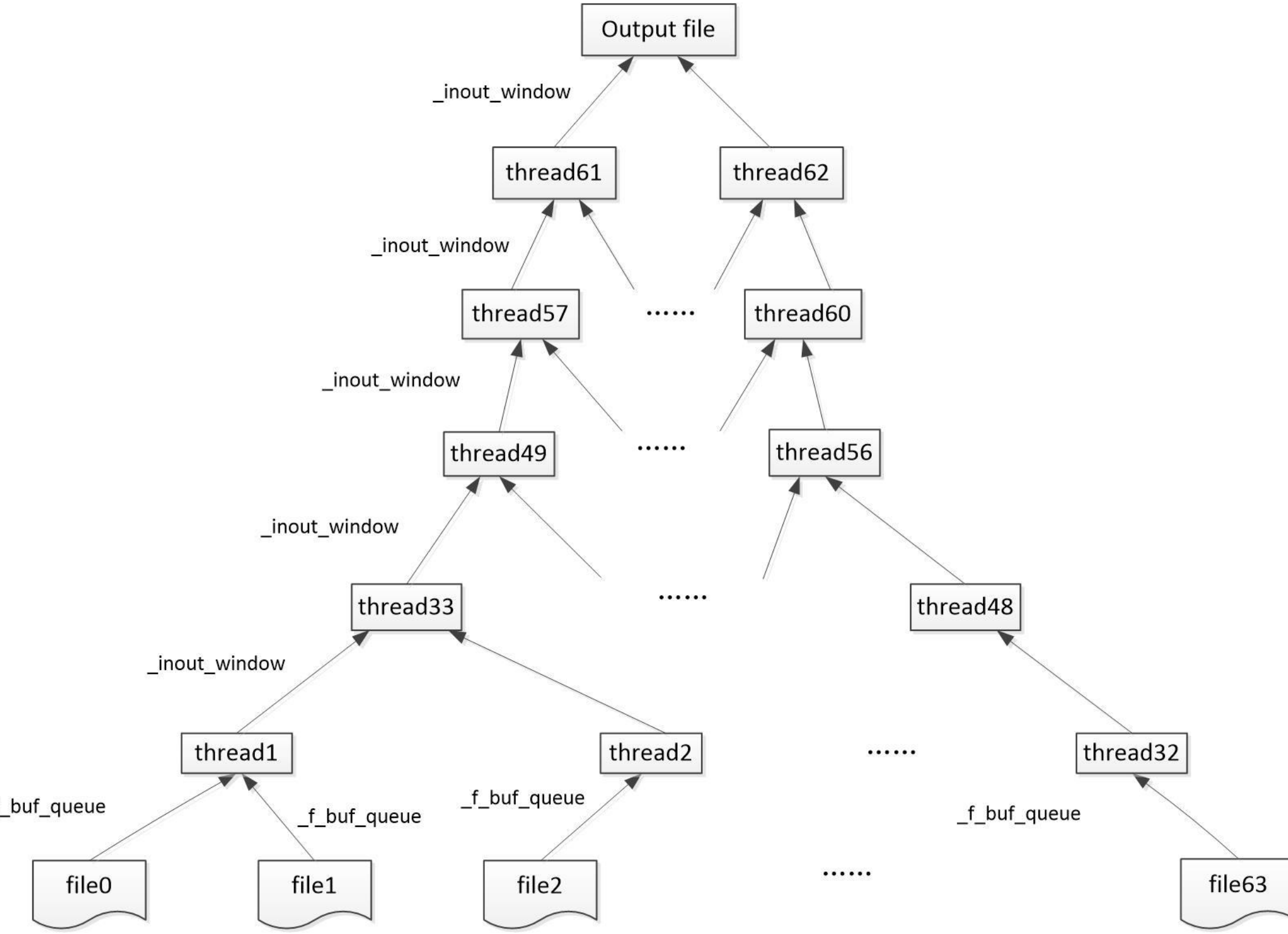


Multi-thread Merge Sort – Phase 2

Create 32+16+8+4+2=62 threads and merge data

The 62 threads are logically divided into 5 layers:

- Threads of bottom layer(layer 1) read from all sorted files in phase 1, and output to window;
- Threads of middle layers(layers 2-4) read from lower-layers, merge and output to windows;
- Threads of top layer(layer 5) write every 10 number to final output file.


Output file
_inout_window
thread61
thread62
_inout_window
thread57
.....
thread60
_inout_window
thread49
.....
thread56
_inout_window
thread33
.....
thread48
_inout_window
thread1
thread2
.....
thread32
_buf_queue
_f_buf_queue
_f_buf_queue
_f_buf_queue
file0
file1
file2
.....
file63

Data Structure Design

```c
#define sortqueue_len  2500000  //~20MB, ~2.5M numbers

typedef struct _inout_window {
    sort_window * p_in_window1; // read-in window1
    sort_window * p_in_window2; // read-in window2
    sort_window * p_out_window; // output window
    int in_fd1; // file descriptor of input file1 (just use for layer-1 threads)
    int in_fd2; // file descriptor of input file2 (just use for layer-1 threads)
    int out_fd; // file descriptor of final output file (just use for layer-5 threads)
} inout_window;

typedef struct _sort_window {
    pthread_mutex_t mutex; // mutex lock
    pthread_cond_t cond_r; // condition variable for allowing read
    pthread_cond_t cond_w; // condition variable for allowing write
    unsigned long long s_queue[sortqueue_len+1]; //circular queue
    int head; // pointer to the begin of the circular queue
    int tail; // pointer to the end of the circular queue
} sort_window;
```

Running Result

Test0: 28min27sec

```
388000000 / 400000000 numbers completed.
392000000 / 400000000 numbers completed.
396000000 / 400000000 numbers completed.
400000000 / 400000000 numbers completed.
Time used so far is 0 h 42 m 57 s.

real    42m56.727s
user    20m46.853s
sys     0m53.756s
```

Test1: 29min13s

```
392000000 / 400000000 numbers completed.
396000000 / 400000000 numbers completed.
400000000 / 400000000 numbers completed.
Time used so far is 0 h 29 m 13 s.

real    29m13.221s
user    7m39.167s
sys     0m53.805s
```

Thank you!