Pseudo Code:

1. *St* <- transmit speed
2. *m* <- sum of total cores
3. *n* <- sum of total blocks
4. *orderedJobs* <- a sorted indices of jobs in descending order of Max Block Size
5. *cores* ("*Set*" in code) <- a priority queue of cores in ascending order of finish time
6. for *job* *i* in *orderedJobs* do:
   1. *orderedBlocks* <- a sorted indices of blocks of *job i* in descending order of corresponding Block Size
   2. *Sc* <- compute speed
   3. *minFinishTime <- 0*
   4. *startTime <- 0*
   5. for *j* from *1* to *min(n, m, MaxJobCores*(e.g.10) *)* do:
      1. *assignedCores* <- an empty priority queue of cores assigned to *job i*
      2. for *block k* in *orderedBlocks* do:
         1. if *cores* aren't not fully assigned:
            1. for *core l* in *cores* with minimal *finishTime* do:

if the *host h* of *core l* stores the data of *block k* do:

insert *core l* to *assignedCores* as *core c*：

*c.processingTime* = *size of block k / Sc* + *size of block k / St*

*c.startTime* = *l.finishTime*

* + - * 1. if *block k* is not assigned with any cores do:

insert the *cores[0]* to *assignedCores* as *core c*:

*c.processingTime* = *size of block k / Sc*

*c.startTime* = *l.finishTime*

* + - 1. else (e.g. blocks are already assigned to *j* cores) do:
         1. for *core l* in *assignedCores*:

*l.finishTime* > *assignedCores[0].processingTime* + *size of block k* / *St* do:

break

else if the *host h* of *core l* stores the data of *block k* do:

assign *block k* to core *l*

*l.processingTime* += *size of block k* / *Sc*

* + - * 1. if *block k* is not assigned with any core do:

assign *assignedCores[0]* to *block k*

*l.processingTime* += *size of block k / St* + *size of block k / Sc*

* + - 1. *maxStartTime* <- max of *startTime* of cores in *assignedCores*;
      2. *maxProcessingTime* <- max of *processingTime* of cores in *assignedCores*;
      3. if *j* = *1* or (*j > 1* and *maxStartTime* + *maxProcessingTime* < *minFinishTime*) do
         1. *prevAssignedCores* <- *assignedCores*
         2. *startTime <-maxStartTime*
         3. *minFinishTime <- maxStartTime* + *maxProcessingTime*

# assign the cores to blocks according to *prevAssignedCores*

* 1. for core *l* in *cores* do:
     1. if *core* *l* in *prevAssignedCores do:*
        1. *l.finishTime* = *minFinishTime*

Analysis

1. Algorithm Description:
   1. Greedy Approach
   2. Idea: To minimize Makespan after assigning each job
      1. Consider to split the job into n parts (n from 1 to 10)
2. Notation:
   1. *St* <- transmit speed
   2. *Sc* <- compute speed
   3. *m* <- sum of total cores
   4. *n* <- sum of total blocks
   5. *cores* ("*Set*" in code) <- a set of cores in ascending order of finish time
   6. *MaxJobCores <-* the maximal number of cores that could be assigned to each job
   7. *host <-* Host
   8. *core <-* Core of Hosts
   9. *block <-* Array of block size of each job
3. Time Complexity: O(n \* m \* log(m) \* k \* log(k) )
   1. n is the total count of blocks
   2. m is the total count of cores
   3. k is the maximal number of cores that could be assigned to each job (i.e. 10 in our case)
4. Space Complexity: O(m + n)