CPU Power Management in Video Transcoding Servers

2014. 3. 20

Minseok Song, Inha University

ACM NOSSDAV' 14

Outline

- Introduction
- System model
- Problem formulation
- Algorithm
- Experimental results
- Conclusions

Introduction –Background

Companies like Netflix, Hulu, Apple, and Amazon helped drive the

Revenue of OTT video is increasing sharply

tablets, are expected to push the market past $20 billion by 2015.

https://www.abiresearch.com/press/over-60-growth-in-worldwide-over-the-top-video-rev

ABI research report 2013

DASH generates a major requirement of transcoding

temporal resolution

Development of DASH

Introduction –Background

Very heterogeneous devices need to be supported

techcrunch.com May 15, 2012

3,997 Models: Android Fragmentation As Seen By The Developers Of OpenSignalMaps

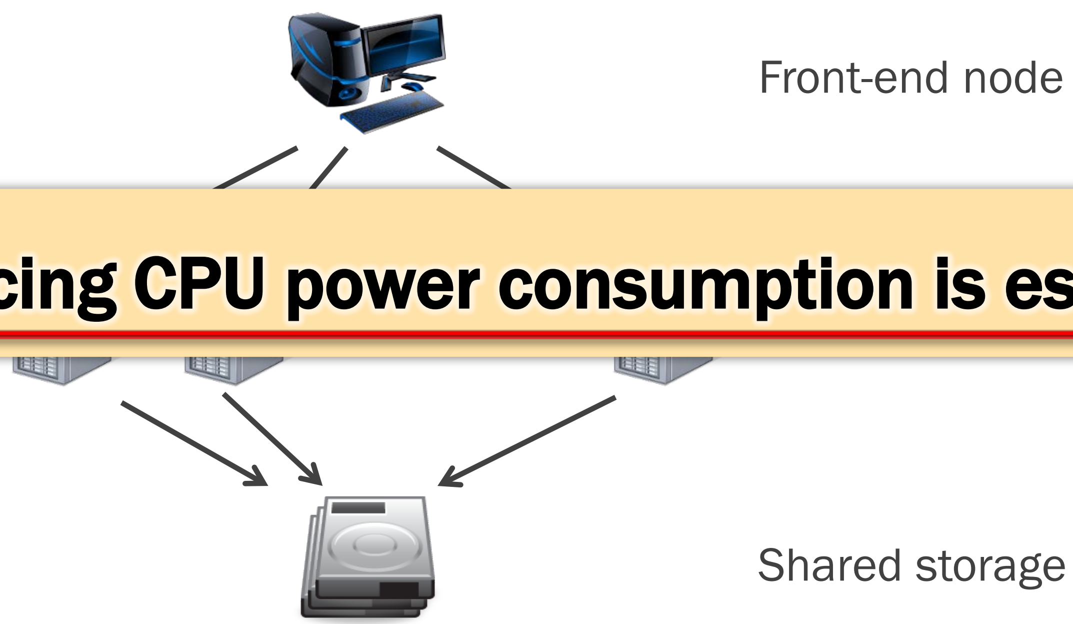
Transcoding is now in great demand

Introduction –Motivation

- Transcoding operations have real-time constraints
  - 1) Sports broadcasting
    - Prompt processing is very essential
  - 2) VoD service
    - Some delays may be allowed
  - 3) Video clips uploaded by users
    - Transcoding requests by high-priority clients need to be processed faster than those requested by low-priority clients
  - 4) Transcoding to the popular video formats needs to be processed first
    - Other formats can be processed later
    - Unpopular videos can be processed later

Introduction –Motivation

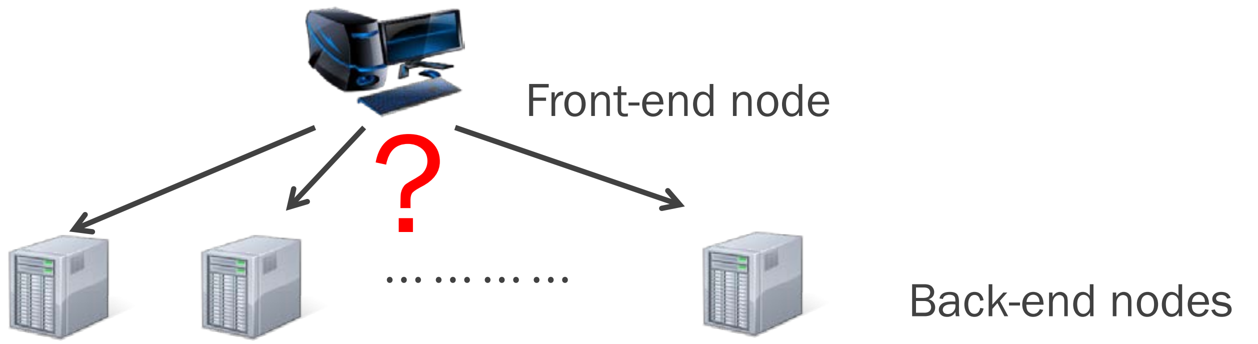
- Transcoding is inherently CPU-intensive
  - Need a lot of machines
    - Result in high power consumption by the CPU
  - Clustered architecture



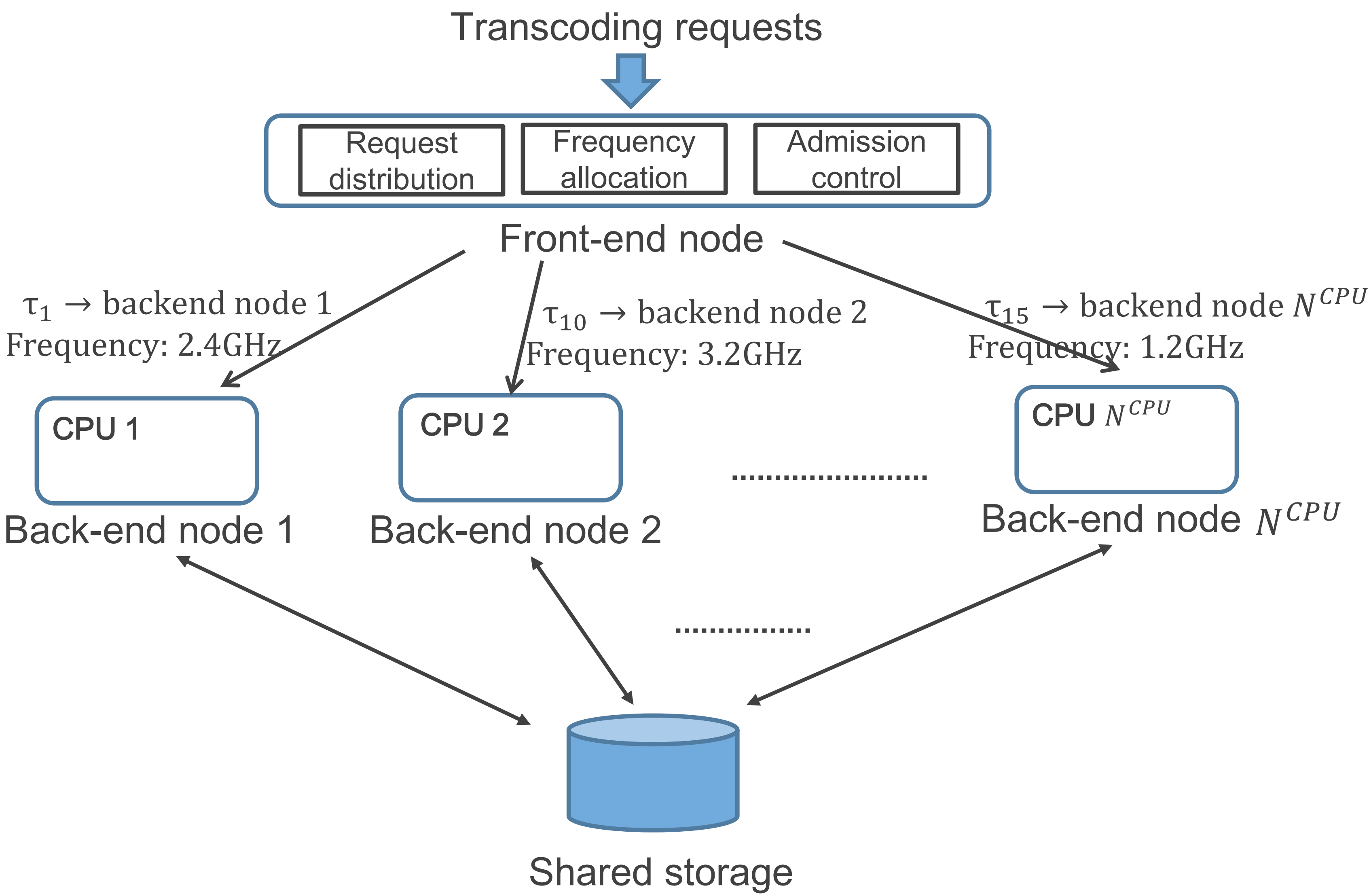
Reducing CPU power consumption is essential

Introduction –Objectives

- This paper reports the first report that
  - Handles 1> real-time constraints of transcoding and 2> power management simultaneously
- How ?
  - 1> Dynamic Voltage and Frequency Scaling (DVFS)
    - Reducing CPU frequency can reduce power consumption but slows down program execution
  - 2> Workload distribution



System model –Architecture



System model –Model

- CPU model
  - Each CPU j can run at a number of discrete frequency levels
    - $f_j(k)$ is thCPU j e frequency at level k for CPU j
    - $f^{\text{base}} = \max_{j=1,\ldots,N^{\text{cpu}}} f_j(N_j^{\text{freq}})$
- Task model
  - Each task, τ_i,has two parameter (C_i, D_i)
    - C_i is the computation time required if the frequency is f^{base}
    - D_i is a relative deadline, which is time difference the absolute deadline and the current time.
  - Actual computation time at frequency
    - $C_i(k) \frac{f^{\text{base}}}{f_j(k)}$

Problem formulation –Some concepts

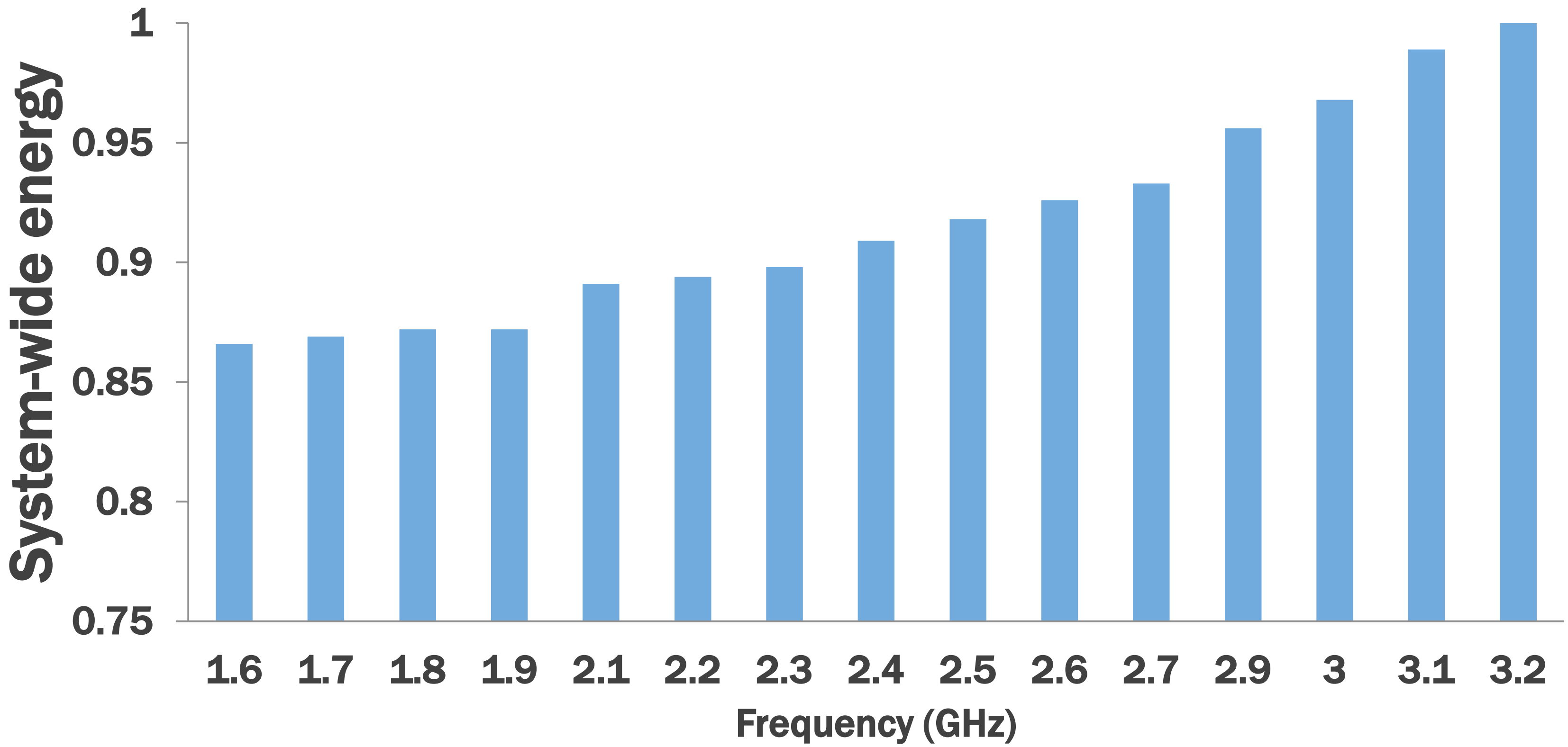
- Utilization factor of task τ_i
  - $U_i = \frac{C_i}{D_i}$
- EDF scheduling
  - Higher priorities are given to tasks with earlier deadlines
- Utilization bound of CPU j at frequency level k
  - $U_j^{\text{bound}}(k) = \frac{f_j(k)}{f^{\text{base}}}$

Problem formulation –Some concepts

- If the sum of utilization factors required by all the tasks on CPU j is less than or equal to $U_j^{\text{bound}}(k)$, then all the tasks can be transcoded before their deadlines
  - $\sum_{task\ i \rightarrow CPU\ j} U_i \leq U_j^{\text{bound}}(k) = \frac{f_j(k)}{f^{\text{base}}}$ ↑

Increasing frequency level increases feasibility bound, allowing more tasks to be transcoded before deadlines

Problem formulation –Tradeoff



Tradeoff between energy and number of tasks transcoded before deadlines !

Problem formulation –Optimization

- Frequency and task allocation problem

Minimize $\sum_{i=1}^{N^{\text{task}}} \sum_{j=1}^{N^{\text{cpu}}} x_{i,j} P_j(F_j)$

s.t. $F_j = \arg\min_{k=1,\ldots,N_j^{\text{freq}}} \sum_{i=1}^{N^{\text{task}}} x_{i,j} U_i \le U_j^{\text{bound}}(\text{k})$

$x_{i,j} \in 0,1$

$\sum_{j=1}^{N^{\text{cpu}}} x_{i,j} = 1$

F_j : Frequency level of CPU j

$x_{i,j}$: task τ_i is assigned to CPU j

$P_j(F_j)$: Power at frequency level F_j

Algorithm – Basic idea

- Three-phase algorithm
  - 1> <u>Frequency determination phase</u>
    - Choose preliminary values of frequencies for each CPU
      - F_j for CPU j
  - 2> <u>Task allocation phase</u>
    - Determines CPU index to which each task τ_i is allocated
      - $x_{i,j}$ for task τ_i
  - 3> <u>Frequency escalation phase</u>
    - If $\sum U_i > U_j^{\text{bound}}(k)$ after the second phase, then frequency levels of some CPUs are escalated

until $\forall i, \sum U_i \leq U_j^{\text{bound}}(k)$

Algorithm – First phase

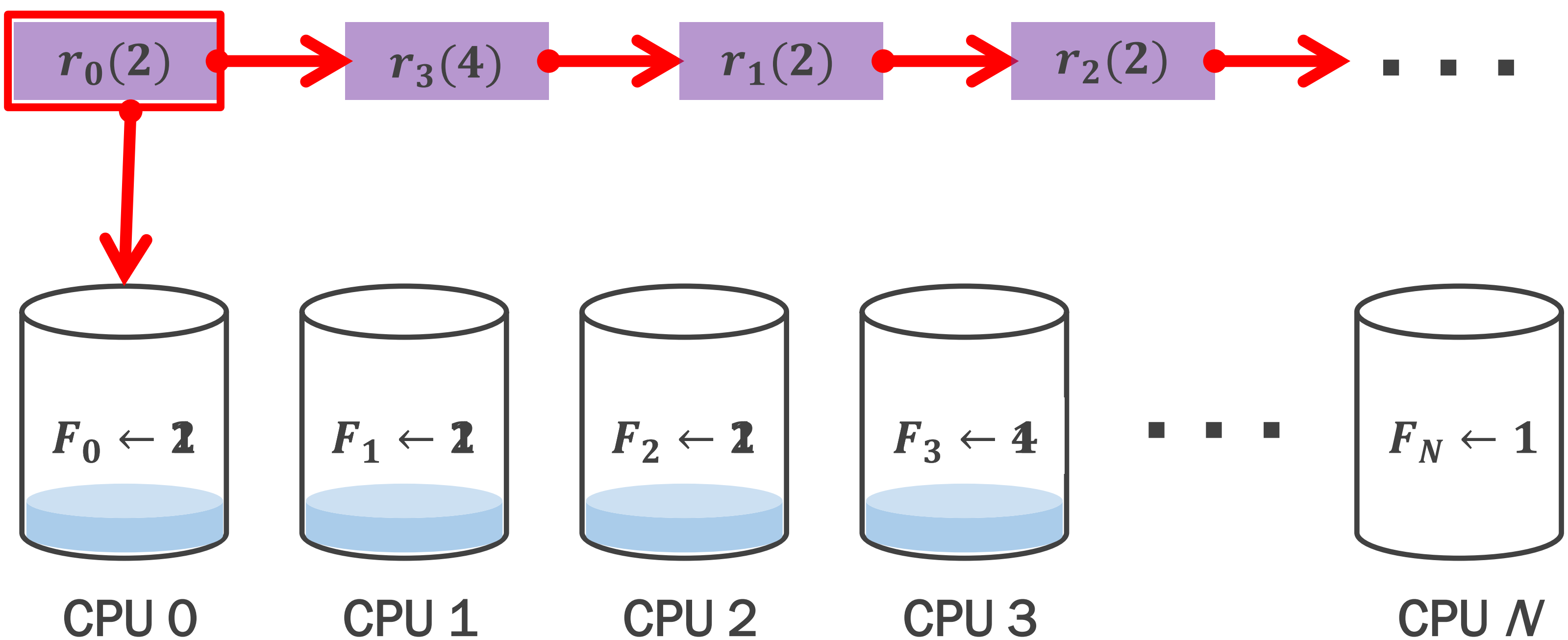
- 1> Minimum demand of utilization factors
  - $U^{\text{demand}} = \sum_{i=1}^{N^{\text{task}}} \frac{C_i}{D_i}$
- 2> Formulation
  - Minimize $\sum_{j=1}^{N^{\text{cpu}}} P_j(F_j)$

    s.t. $\sum_{j=1}^{N^{\text{cpu}}} U_j^{\text{bound}}(F_j) \geq U^{\text{demand}},$

    $F_j = 1, \ldots, f_{N_j^{\text{freq}}}$
- 3> Greedy algorithm (Initialized to the first frequency level)
  - $r_j(k) = \frac{P_j(k) - P_j(1)}{U_j^{\text{bound}}(k) - U_j^{\text{bound}}(1)}$

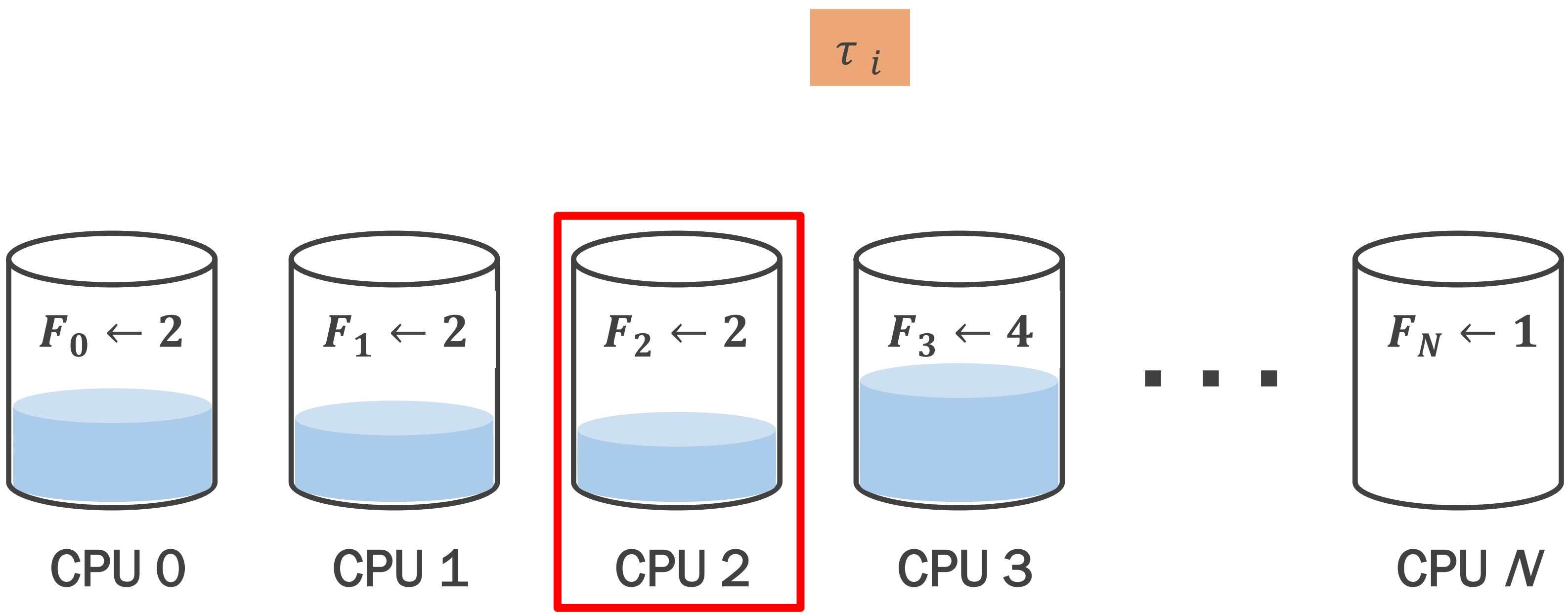
Algorithm – First phase

1. All the variables F_j are set to 1
2. Find the pair of CPU index c and frequency level l that has the lowest value of $r_c(l)$
3. If the frequency level l is higher than F_c, the value of F_c is increased to l
4. Repeat until $\sum_{j=1}^{NCPU} U_j^{bound}(F_j) \geq U^{demand}$



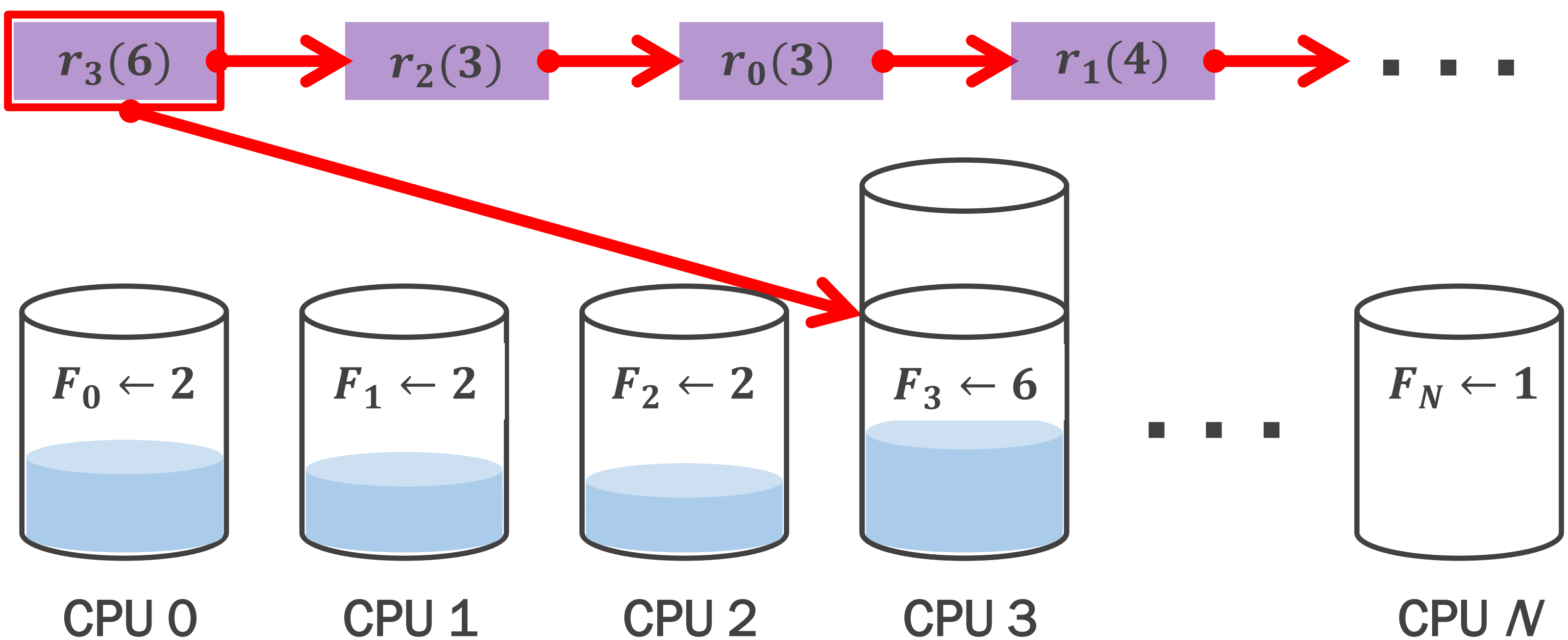
Algorithm – Second phase

1. Maintains the array, A: $\{i \mid \forall x_{i,j} = 0\}$
2. Find the index h that has the highest value of the utilization factor for CPU h and assign task i to CPU h (set the value of $x_{i,h}$ to 1)
3. Repeat until $A = \emptyset$ ($\forall x_{i,j} = 1$)
4. If all the tasks can't be assigned to a CPU without violating the constraint, jump to third phase to satisfy the constraint

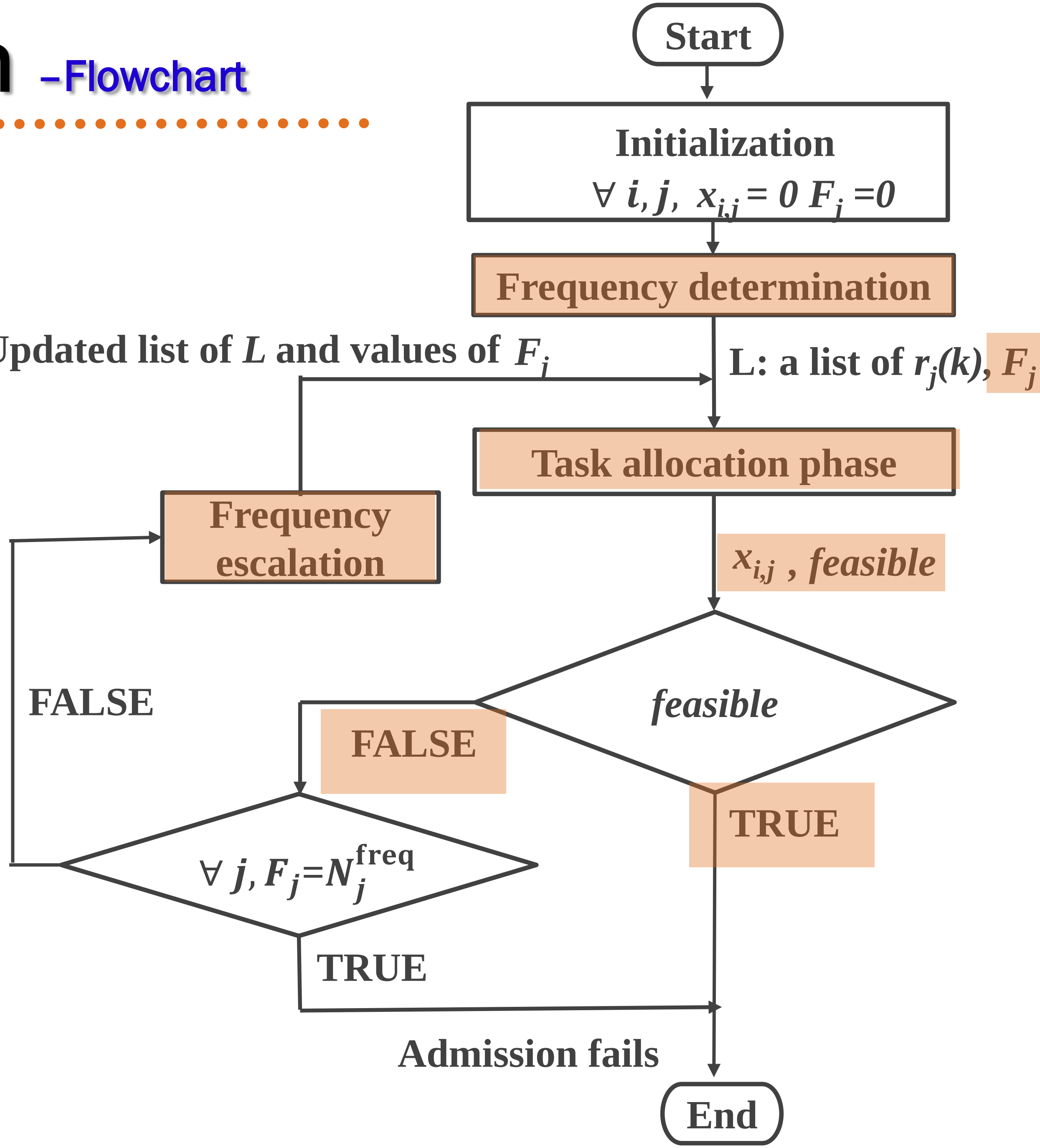


Algorithm – Third phase

1. Find the pair of CPU index c and frequency level l that has the lowest value of $r_c(l)$
2. If the frequency level l is higher than F_c, the value of F_c is increased to l



Algorithm –Flowchart



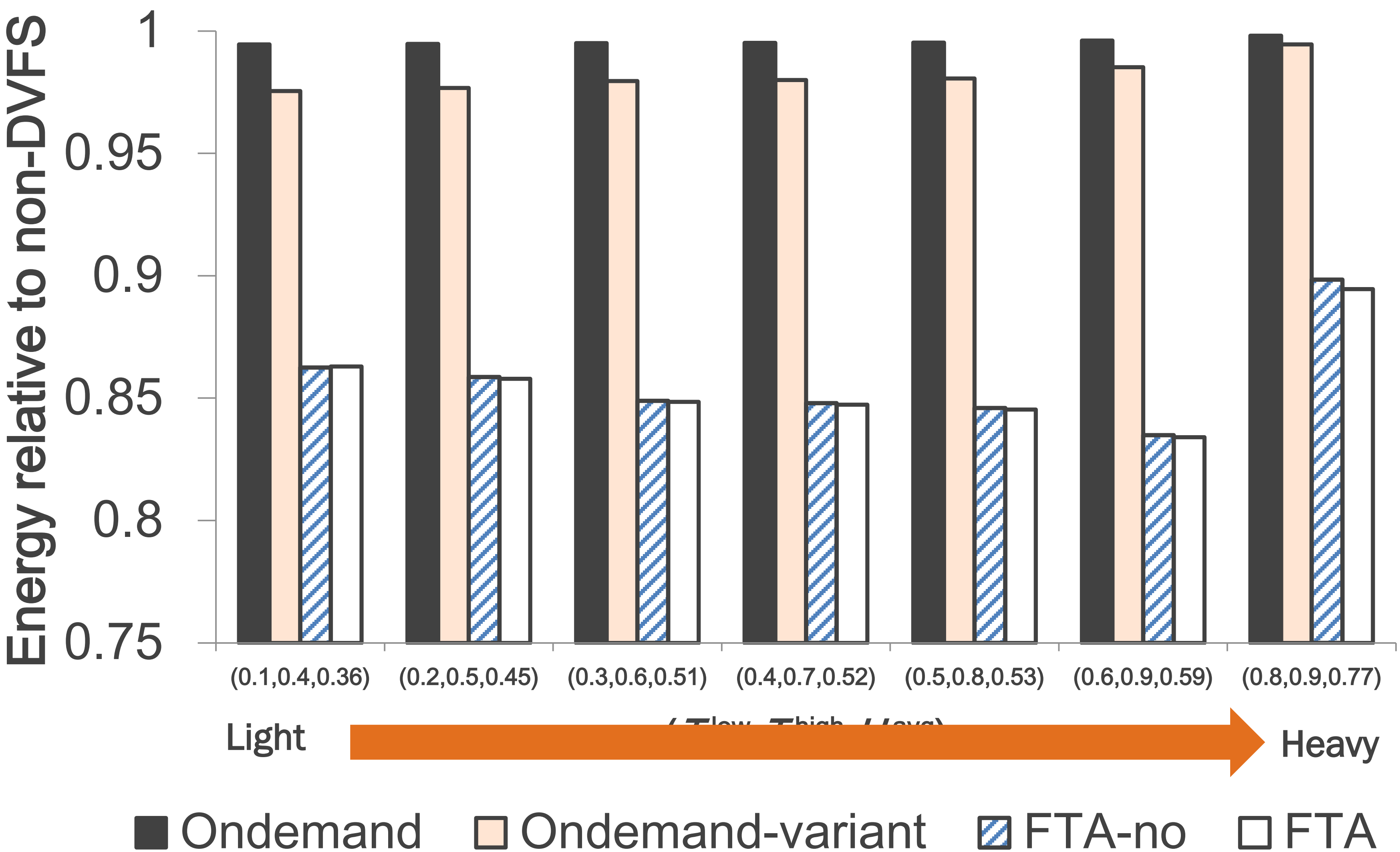
Algorithm – Issues

- 1> <u>Task migration</u>
  - It is impossible to change the values of $x_{i,j}$ for all existing tasks
    - All the values of $x_{i,j}$ of existing clients are maintained
    - 1) $x_{i,j}$ of new clients and 2) F_j values can be determined
- 2> <u>EDF scheduling</u>
  - Linux provides the SCHED_FIFO class, which uses fixed-priority scheduling
- 3> <u>Admission control</u>
  - If the algorithm is infeasible even though the highest frequency is chosen for every CPU, then admission fails

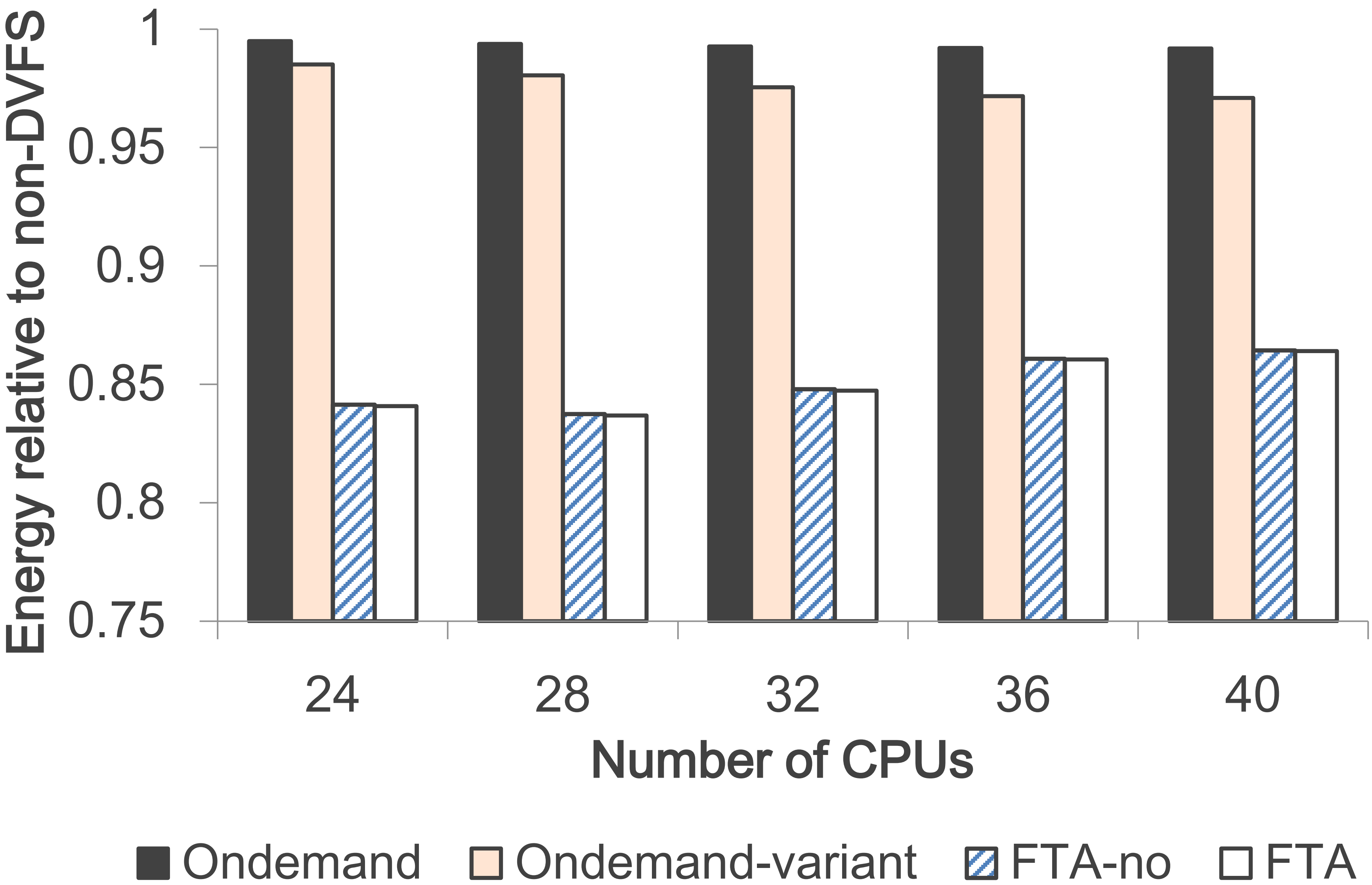
Experimental results – Simulation setup

- Simulation environments
  - Measured energy values
    - System-wide energy for 4 PCs
  - Transcoding time
    - Randomly selected between 30s and 300s
  - Utilization factor
    - Randomly chosen between 0.02 and 0.12
- Comparison
  - 1>Non-DVFS
    - Only highest frequency is chosen
  - 2> Ondemand
    - CPU utilization over recent 30 seconds exceed 80%, then the maximum frequency level is chosen
    - CPU utilization is below 0.4, then the frequency level is reduced by one
  - 3> Ondemand-variant

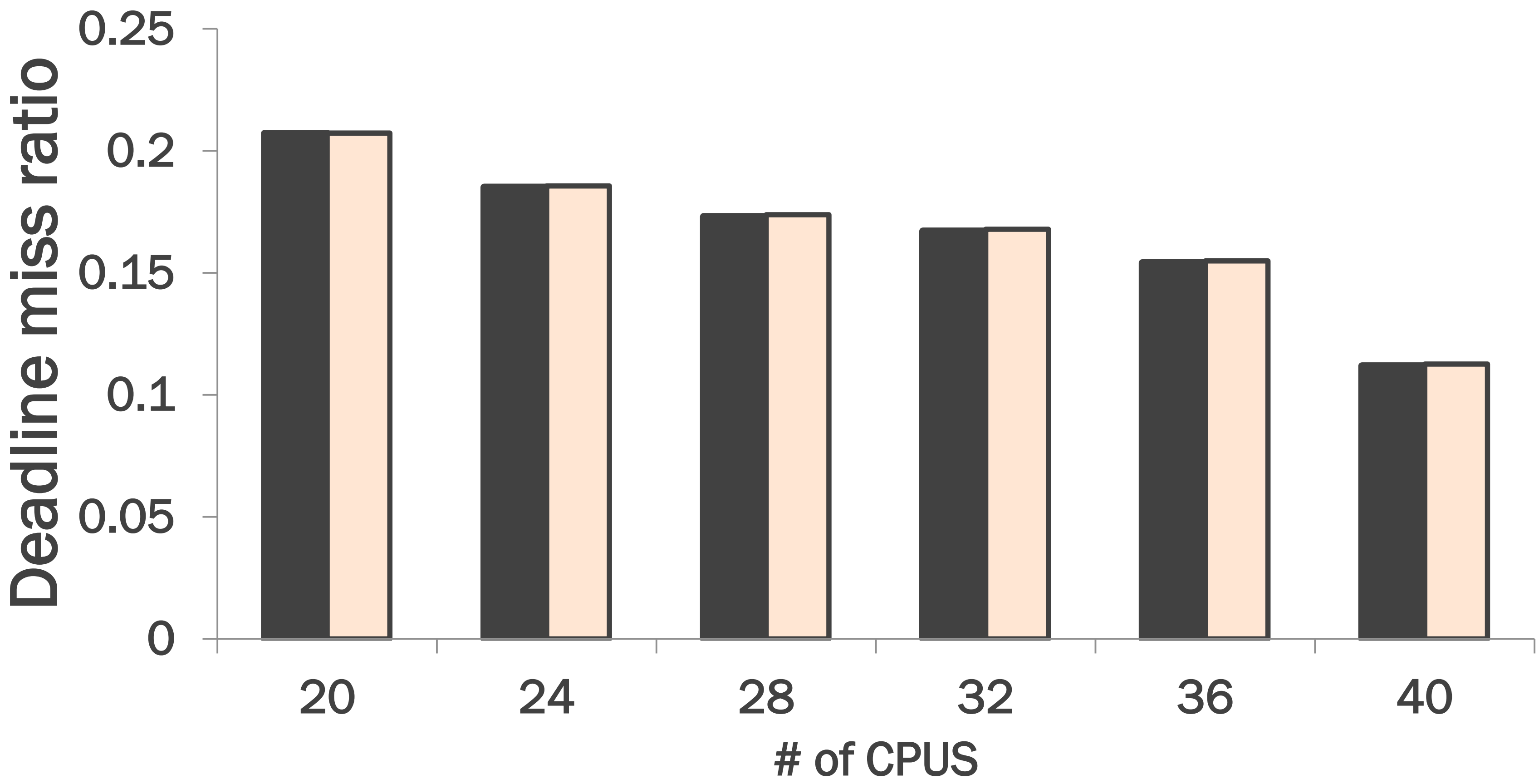
Experimental results – Results



Experimental results – Results



Experimental results – Results



All tasks are transcoded before deadlines

Conclusions