CPU Power Management in Video Transcoding Servers

2014. 3. 20

Minseok Song, Inha University

ACM NOSSDAV’ 14
Companies like Netflix, Hulu, Apple, and Amazon helped drive the over-the-top (OTT) video market past $8 billion in 2012. The three largest markets—North America, Europe, and Asia-Pacific—experienced YoY growth in excess of 50% in 2012. The continued spread of connected CE and increasingly mobile devices, like tablets, are expected to push the market past $20 billion by 2015.


**Revenue of OTT video is increasing sharply**

**DASH generates a major requirement of transcoding**

temporal resolution
Transcoding is now in great demand

Very heterogeneous devices need to be supported

techcrunch.com  May 15, 2012
3,997 Models: Android Fragmentation As Seen By The Developers Of OpenSignalMaps
Introduction

• 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**

![Clustered architecture diagram]

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

- Transcoding requests
- Front-end node
  - Request distribution
  - Frequency allocation
  - Admission control

- Back-end node 1
  - Frequency: 2.4GHz
- Back-end node 2
  - Frequency: 3.2GHz
- Back-end node \( N_{CPU} \)
  - Frequency: 1.2GHz

\( \tau_1 \rightarrow \) backend node 1
\( \tau_{10} \rightarrow \) backend node 2
\( \tau_{15} \rightarrow \) backend node \( N_{CPU} \)

- CPU 1
- CPU 2
- CPU \( N_{CPU} \)

- Shared storage
System model

- CPU model
  - Each CPU $j$ can run at a number of discrete frequency levels
    - $f_j(k)$ is the 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, $\tau_i$, has two parameters $(C_i, D_i)$
    - $C_i$ is the computation time required if the frequency is $f^{\text{base}}$
    - $D_i$ is a relative deadline, which is the time difference between 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 $\tau_i$
  
  $- U_i = \frac{C_i}{D_i}$

- **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.

\[ \sum_{\text{task } i \rightarrow \text{CPU } j} U_i \leq U_j^{\text{bound}}(k) = \frac{f_j(k)}{f_{\text{base}}} \]

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. \[ F_j = \arg \min_{k=1,...,N_j} \sum_{i=1}^{N_{\text{task}}} x_{i,j} U_i \leq U_j^{\text{bound}}(k) \]

\[ x_{i,j} \in \{0,1\} \]

\[ \sum_{j=1}^{N_{\text{cpu}}} x_{i,j} = 1 \]

\( F_j \) : Frequency level of CPU \( j \)

\( x_{i,j} \) : task \( \tau_i \) is assigned to CPU \( j \)

\( P_j(F_j) \) : Power at frequency level \( F_j \)
Algorithm – Basic idea

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

\[ \text{until } \forall i, \sum U_i \leq U_j^{\text{bound}}(k) \]
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. \( \sum_{j=1}^{N_{\text{cpu}}} U_j^{\text{bound}}(F_j) \geq U^{\text{demand}}, \quad F_j = 1, \ldots, f_{N_j}^{\text{freq}} \)

• 3> Greedy algorithm (Initialized to the first frequency level)
  - \( r_j(k) = \frac{P_j(k) - P_j(1)}{U_j^{\text{bound}}(k) - U_j^{\text{bound}}(1)} \)
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 $\sum_{j=1}^{N_{CPU}} U_j (F_j) \geq U^{demand}$
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
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 $l$.

- $F_0 \leftarrow 2$
- $F_1 \leftarrow 2$
- $F_2 \leftarrow 2$
- $F_3 \leftarrow 6$
- $F_N \leftarrow 1$
Algorithm - Flowchart

- Start
- Initialization
  \( \forall \ i, j, \ x_{ij} = 0 \ F_j = 0 \)
- Frequency determination
- Task allocation phase
  L: a list of \( r_j(k), F_j \)
- Updated list of \( L \) and values of \( F_j \)
- Frequency escalation
- \( x_{ij} \), feasible
- feasible
  \( \forall \ j, F_j = N_j^{freq} \)
  TRUE
  Admission fails
  FALSE
  FALSE
  TRUE
- End
Algorithm – Issues

1> Task migration
   – 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> EDF scheduling
   – Linux provides the SCHED_FIFO class, which uses fixed-priority scheduling

3> Admission control
   – 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

Energy relative to non-DVFS

Ondemand

Ondemand-variant

FTA-no

FTA

Light (Low, Thigh, Low)

Heavy

(0.1, 0.0, 0.36) (0.2, 0.5, 0.45) (0.3, 0.6, 0.51) (0.4, 0.7, 0.52) (0.5, 0.8, 0.53) (0.6, 0.9, 0.59) (0.8, 0.9, 0.77)
Experimental results

- Energy relative to non-DVFS

- Number of CPUs

- Ondemand
- Ondemand-variant
- FTA-no
- FTA
Experimental results – Results

All tasks are transcoded before deadlines
Conclusions

10% ~ 16% system-wide energy saving

New DVFS algorithm

High power consumption

Heterogeneous transcoding time requirements

Clustered Transcoding Video Servers