

DDVS: Distributed Dynamic Voltage Scaling

Timothy Durnan
 Dept. of Computer Science and Engineering
 University of Notre Dame
 Notre Dame, IN 46556
 tdurnan@cse.nd.edu

Christian Poellabauer
 Dept. of Computer Science and Engineering
 University of Notre Dame
 Notre Dame, IN 46556
 cpoellab@cse.nd.edu

Motivation and Problem Statement. Dynamic voltage scaling (DVS [1]) is a popular technique in energy-aware systems: when a CPU is under-utilized, reduce its speed and voltage, thereby trading off increased execution times with reduced energy costs, while meeting real-time tasks' deadline constraints. However, DVS and other energy-saving techniques are 'selfish' in nature, i.e., they are only concerned with reducing their local energy consumptions, disregarding the consequences their use may have on other devices. Consider two devices A and B , where A captures and compresses video images, sends them to B , where they will be decompressed, processed, and displayed. The latest point in time an image has to be received, processed, and displayed by B denotes an *end-to-end deadline (E2E)* T_d . Figure 1 visualizes the problem: the shaded areas show periods of CPU activity (image processing), with the height of the area indicating the power costs, the arrows denote communications between devices, and the vertical line shows T_d for a given image. When both A and B use DVS, their processing times

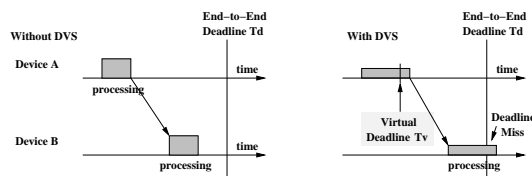


Figure 1: E2E latencies with/without DVS.

increase, introducing delays that may cause the deadline to be missed. The consequence is that A and B need to negotiate the *allowable* slow-down (and delay) each device can introduce. This negotiation is driven by the current battery charge levels and the overall goal of the system, e.g., if B is essential to the operation of the distributed system, but at the same time the more energy-constrained device, it should be allowed to fully utilize DVS, while A can utilize DVS only to an extent that does not cause deadlines misses.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SOSP'05, October 23–26, 2005, Brighton, United Kingdom.
 Copyright 2005 ACM 1-59593-079-5/05/0010 ...\$5.00.

Distributed Dynamic Voltage Scaling (DDVS). In a video conference, it is often preferable to be able to continue for another 30 minutes with *all* participants than to continue for another 60 minutes, but only half the audience (assuming the other half has depleted batteries). As an example for collaborative resource management we introduce *DDVS – Distributed Dynamic Voltage Scaling* – an end-to-end or distributed DVS algorithm that allows devices to negotiate their use of DVS in form of a feedback control loop. Here, a controller (e.g., PI or PID) computes a *virtual deadline* T_v , which is imposed on task execution at device A depending on input variables (e.g., task parameters), battery charge levels of A and B , and the *slack* d_s , which is defined as $d_s = T_d - T_f$, where T_f is the *finish time* of the image processing task at device B (see Figure 2). Based on the monitored slack and charge levels, T_v is dynamically adjusted to ensure the desired balance between maximum energy savings for both devices and meeting the end-to-end deadlines. Positive slack indicates that there is room for more savings and T_v is increased (allowing A to exploit DVS more), negative slack indicates a missed deadline and T_v is decreased. In this work, we assume periodic tasks (e.g., multimedia applications) and both devices' CPU schedulers use EDF, with $D \leq P$ (D =deadline, P =period). On each device, a DVS algorithm computes appropriate speed/voltage levels based on the current utilization of the device (this algorithm is an extension to our prior work on DVS [2]).

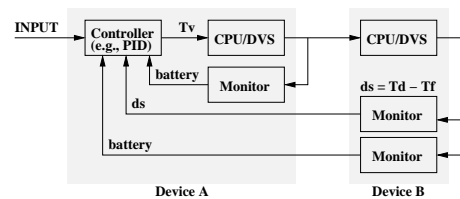


Figure 2: Feedback-based DDVS.

References

- [1] P. Pillai and K. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *Proc. of the 18th ACM Symposium on Operating Systems Principles*, 2001.
- [2] C. Poellabauer, L. Singleton, and K. Schwan. Feedback-Based Dynamic Frequency Scaling for Memory-Bound Real-Time Applications. In *Proc. of the 11th Real-Time and Embedded Technology and Applications Symposium*, March 2005.

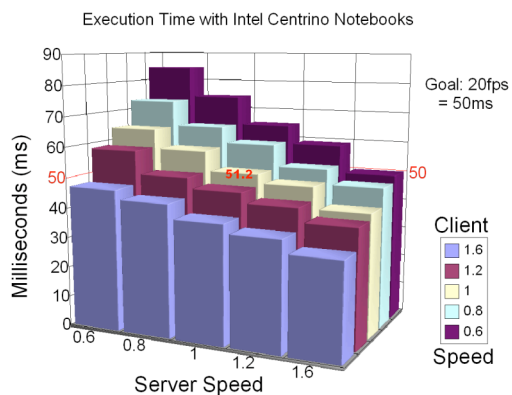
DDVS: Distributed Dynamic Voltage & Frequency Scaling

Timothy J Durnan
& Christian Poellabauer, PhD
Computer Science & Engineering
University of Notre Dame
Notre Dame, IN USA

DDVS – Problem Statement

- Dynamic Voltage Scaling provides energy benefits for mobile computing
- Side-effect – causes execution time of applications to increase in duration
- Unfortunate consequences in a distributed environment
- Compounded problem if it is a real-time system

DDVS – Example Scenario



DDVS – Current Technology

- Client is the only one to see the end-to-end deadline misses
- Client, therefore, is the only one with the capability to perform any changes in the system
- Easily can be shown to have poor performance
 - Example: Server has 95% battery remaining while client has only 15% battery remaining

DDVS – Implementation

- Utilize a feedback loop to exchange battery and deadline information between server and client
 - If the client's battery has more energy remaining, no changes are needed – traditional V/S
 - Else, create a floating *virtual deadline for the application on the server* in order to limit that machine's ability to use DVS
 - removes strain from the client
 - allows for better overall system performance

DDVS – Related Works

- P. Pillai and K. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *Proc. of the 18th ACM Symposium on Operating Systems Principles*, 2001.
- C. Poellabauer, L. Singleton, and K. Schwan. Feedback-Based Dynamic Frequency Scaling for Memory-Bound Real-Time Applications. In *Proc. of the 11th Real-Time and Embedded Technology and Applications Symposium*, March 2005.
- S. Baruah, R. Howell, and L. Rosier. Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor. *Real Time Systems 2*, pp 301-324, 1990.