Energy Management in Real-time Multi-tier Servers

Tibor Horvath (PhD expected Jan’07)
Kevin Skadron
Univ. of Virginia

Tarek Abdelzaher
University of Illinois
• Originally funded by NSF Parallel and Distributed Operating Systems program
  • Also partial funding from ARO re: graceful response to load spikes
• Now funded by AES track
Multi-tier Servers

- Requests are processed by a server pipeline
  - E.g. HTTP Front-end, Application Server, Database Server
- Functionally distributed
- Can be significantly imbalanced
  - Each request has different resource needs on each stage
Motivation

Sources: IDC data for installed base, shipments, and most popular models, and manufacturer data on power use for individual server models. Total expenditures assume US industrial electricity prices (2006 dollars)

1.2% of 2005 U.S. electricity sales, $2.7B/year

0.8% of estimated 2005 world electricity sales, $7.2B/year

 Cooling and auxiliary equipment

 High-end servers

 Mid-range servers

 Volume servers

 World


 Does not include many servers, e.g. Google data centers!
Typical Workloads

- Peak load much higher than average
  - Capacity is planned to satisfy worst-case load
- Light load during long periods of time
  - The server sits idle
  - Idle operation wastes energy
- Great potential for energy savings
- First focus: DVS

Source: Bohrer et al., *The Case For Power Management In Web Servers* (IBM Research)
Constraints

- Soft real-time performance
  - Power management must not impair user experience significantly
  - User experience $\rightarrow$ only end-to-end delay guarantees are relevant
  - DVS settings across the pipeline must be coordinated to meet deadlines while minimizing power consumption
- Commodity server software
  - Linux, Apache, JBoss, MySQL
- Dynamic workload with target latencies
  - TPC-W benchmark
Algorithms

- **Simple DVS**
  - Good approximation for homogeneous systems
  - Feedback controller with simple rules:
    - If total latency > target → speed up stage with maximal CPU utilization
    - If total latency < target → slow down stage with minimal CPU utilization

- **Weighted DVS**
  - Based on analytical optimality condition
    - With knowledge of workload and machine power characteristics
  - Feedback controller adjusts CPU speeds to stay close to the optimality condition
    - Dead zone feedback control
    - Thresholds determined by max tolerable deadline miss ratio (e.g., 5%), conditional probability analysis
Optimality Condition

- Workload-dependent delay function:
  \[ D_{i}^{\text{CPU}} = \frac{T_i}{1 - U_i} \]

- Hardware-dependent power function:
  \[ P_i = A_i f_i^n + B_i \]

- End-to-end latency constraint:
  \[ \sum_{i=1}^{N} D_{i}^{\text{CPU}} + D_{i}^{\text{block}} \leq L \]

- Solution:
  \[ W_1 H(U_1) = W_2 H(U_2) = \ldots = W_N H(U_N) \]
  \[ W_i : \text{weight calculated from workload and power fns} \]
  \[ H(U_i) = \frac{(1 - U_i)^2}{U_i^{n+1}} \]

- Basic idea: weighted utilizations should be equalized across tiers
Testbed of 3 AthlonXP laptops with multiple DVS levels
Results

- Target performance achieved
  - End-to-end deadline miss rate within 3% of baseline (max tolerable set at 5%)
  - Throughput was almost unaffected

- Up to 30% power savings are achieved
  - Weighted DVS was superior
  - Simple DVS was a good approximation

To appear in *IEEE Trans. Computers, 4/07*
Service Prioritization

• Different clients – different performance requirements
  • For example, interactive vs. background tasks; paying vs. free customers

• Deadlines of lower-priority requests can be relaxed

• Additional energy savings can be realized
  • Servers need priority request scheduling
  • DVS algorithm needs to recognize the different classes

• Questions:
  • How to implement this with the least effort?
  • How much energy can be saved?
  • How much is the performance penalty?
Multi-tier Server Prioritization

• Ideal design is expensive to implement:
  • Server applications do not typically support priority scheduling; many are closed source
  • Widely used server OSs do not support priorities for all resources
  • Communication protocols between tiers do not propagate priority information

• Simple, inexpensive design:
  • Run multiple server application instances, prioritized at the process level; no application or OS modification
  • Requires real-time process priorities in OS
  • Effectively creates separate queues and communication channels for each class of service
  • Has limitations: e.g. databases, I/O-bound workloads
    – Solution: minimize queuing in such tiers
Prioritized System Results

- Load is evenly divided into 3 priority classes
- Comparison
  - Baseline: no DVS
  - NP-DVS: Non-priority aware DVS
  - P-DVS: Priority-aware DVS

- Additional energy savings of up to 15%
- Less than 3% increase in average deadline miss rate

Testbed of 8 Athlon64 desktops:
1 front end, 2 Apache, 4 JBoss, 1 MySQL
Current Work

• Power management for large datacenters
  • Sleep modes can be used (in addition to DVS)
    – Comprehensive power management policy
    – Find the optimal balance of the different power states available
  • Dynamic assignment of machines to tiers
    – Helpful if the bottleneck tier shifts over time
  • New optimization problem
    – New optimality conditions for:
      • number of machines in each tier
      • CPU frequencies for each tier
    – More complex feedback controller needed
• Sensor-actuator based control framework
Future Work

• AES project:
  • Implications of multicore processors
  • Supporting virtualized environments
    – How will multi-tier apps be consolidated?
    – How to ensure end-to-end delays?
    – Dealing with sessions
  • Accounting for thermal load
Future Work

• NGS-related work (NSF SEI/IIS, Intel, NVIDIA)
  • Hardware support to simplify parallel programming
    – Key problem: legacy codes and legacy brains
  • Can already support dozens of threads/core, hundreds of PEs/chip
    – Let programmer use these for performance or simplified programming model
  • Must all be subject to power and thermal constraints
  • Major complicating factor: heterogeneous architecture
    – Accelerators
    – Parameter variations and hard faults
Bullet for Later Discussion

- How to make decentralized but globally optimal decisions while preserving real-time characteristics
Power Management Methods

• **Sleep Modes**
  • Turn off unnecessary machines in a cluster
    – Wakeup solution required
  • Consolidate remaining work on alive machines
    – Not possible with some workloads (e.g. large state)
  • Saves most power
  • High impact on:
    – software design (must work in a dynamic cluster)
    – performance (sleeping nodes perform no work)

• **Dynamic Voltage Scaling (DVS)**
  • Slow down the CPUs of machines
  • Saves significant power
  • Low impact (all cluster nodes still work)
  • Can take advantage of I/O bottlenecks
    – CPU slowdown has very little effect on I/O delay