Lecture 8
Energy Management for Hypervisor-Based Virtual Machines

Jan Stoess, Christian Lang, Frank Bellosa

Operating Systems Practical

November 26, 2014
Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
Power Management

- requires the OS to have full knowledge and full control of underlying hardware
- account power management and allocate resources
- constraints on energy usage
- increasing power density and dissipation of modern servers
- generally use OSes with monolithic kernel
Virtual Machines

- server consolidation, transparent migration, secure computing
- distributed, multi-layered software stack
- hypervisor, VM, guest OS
- hypervisor and host driver modules have full control over hardware
Virtual Machines

Applications

Guest OS

Service

Hypervisor

Applications

Guest OS

Service

Hypervisor

vCPU  vNIC  vDISK

vCPU  vNIC  vDISK

vCPU

vCPU

vCPU

vCPU

NIC

DISK
- device control and accounting information are distributed across the whole system
- centralized energy management is unfeasible
- the host doesn’t possess knowledge of the energy consumption of individual applications
- minimal hypervisor → direct control over a small set of devices (oblivious to others)
- guest OSes know the application but not the physical hardware
- recursive power consumption (caused by the virtualized layer itself)
Issues

- recursive power consumption may be high
- non-partitionability (temperature)
- current virtualization solutions disregard most energy-related aspects of the hardware platform
managing energy in distributed, multi-layered OS environments

contributions

- model for partitioning and distributing energy effects
  - energy is the base abstraction
  - physical effects of power consumption in a distributable way
  - may be partitioned

- distributed energy accounting approach
  - direct and side-effectual energy consumption

- expose resource allocation mechanisms
  - enables remote regulation of energy consumption
Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
Goals

- unified energy model
- approach to control energy consumption across all boundaries
- flexibility in supporting diverse energy management paradigms
- temperature constraints
- power limits
- per-user base power consumption
- flexible and extensible to suit a diversity of goals and algorithms
- energy – base abstraction: quantifies power consumption in a partitionable way
- energy constraints may be partitioned from global notions to local, component-wise ones
OS = set of components (control a hardware device, export a service library, provide a software resource)

- separate policy from mechanism
- energy management – feedback loop
  - determine power consumption, account to the originating activities (*mechanism* – *resource driver*)
  - analyze accounting, make decisions (*policy* – *energy manager module*)
  - respond with allocation and deallocation of energy consuming resources (*mechanism*)
  - goal is to align the energy consumption to constraints

- multiple energy managers may exist
- distributed energy accounting & dynamic, exposed resource allocation
Distributed Energy Accounting

- each resource driver must be capable of determining/estimating energy consumption
- driver propagates information to the manager
- incorporate recursive energy consumption
  - may be heavy: disk driver that encrypts or decrypts requests
  - the disk driver is responsible with computing its own CPU energy
  - periodically query the CPU resource driver for energy consumption information
resource drivers expose its allocation mechanism to the manager subsystem

manager leverages consumption to match constraints

hardware and software allocation mechanisms

hardware – power saving features

software – control rate of served requests (that directly influences energy consumption)
Outline

Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
two-level every management framework (host-level, guest-level)

hypervisor-based VM

energy consumers: CPU and disk (so far)

IA-32 microprocessor

L4 microkernel – privileged hypervisor
  - core abstractions: virtual CPUs, communication, address spaces
    - I/O devices are managed at user level; L4 exposes interrupts

paravirtualized Linux
  - VMM based on L4 abstractions

a dedicated driver for each device (exports a virtual interface to client VLs)
  - a guest OS instance itself
host-level energy manager

- periodically obtains the per-VM CPU and disk energy consumption

- (optional) energy aware OS implements resource container abstraction
  - fine-grained energy management for Linux-compatible applications
Energy Model

- access consumption & idle consumption
- reduce access consumption: control device allocation (client request rate)
- reduce idle consumption: sleep mode, low-power state, frequency scaling etc.
CPU Energy Model

- event sampling
- performance counters defined by IA-32
- add a weight to each counter: contribution to the processor energy (calibration)
- sum up number of events multiplied with their weight
Disk Energy Model

- time-based approach
- attribute energy consumption to device states (active and idle)
- active when transferring data: how long does it take?
  \[ \text{time} = \frac{\text{size}}{\text{disk transfer rate}} \]
- ignore seek time; it remains sufficiently accurate
- access consumption is charged directly to each request
- idle consumption is allotted to all client VMs
- allocation mechanisms are exposed
  - CPU throttling
  - disk request shaping
CPU Energy Accounting

- trace performance counter events in the hypervisor
- sent to user-space energy manager
- uses a memory mapped buffer
- separate mechanism from policy
- energy manager is invoked every 20 ms
- takes into account weights
- splits computation among idle cost and access cost
CPU Energy Accounting (2)
Disk Energy Accounting

- disk driver implemented as a Linux driver code in a VM
- translation module translates requests from other VMs to and from the driver (disk requests → Linux block I/O requests)
- accounting completely implemented in the translation module
- dedicated procedure invoked every 50 ms for idle time calculation
Recursive Energy Accounting

会计必须考虑到在虚拟化层中递归地花费的能量。

每个驱动器确定所花费的能量，并将其传递给客户端。

目前仅适用于驱动 VM 中的虚拟磁盘驱动器。

我们已经对翻译模块进行了标记，以确定每个客户端 VM 的活跃和空闲 CPU 能量。
Recursive Energy Accounting (2)
CPU Resource Allocation

- hypervisor provides throttling the CPU
- the energy manager alters CPU shares for virtual processors
- add *idle virtual processor* – guarantees to spend time in halt
Disk Request Shaping

- throttle CPU requests
- process a client VM’s disk requests to a specific budget; delay pending requests
Energy Manager

- relies on accounting and allocation mechanisms described earlier
- initialization
  - define upper power limits for each VM and each device
- feedback loop
  - invoked periodically: 100 ms for CPU, 200 ms for HDD
  - predict future consumptions based on current consumptions
  - compare energy consumption with power limit; if no match, regulate device consumption
Outline

Introduction
Distributed Energy Management
Prototype
Host-Level Energy Management
Virtualized Energy
Experiments and Results
Epilogue
Keywords
Questions
Virtualized Energy

- application specific energy management
- accounting and control for virtual devices
Virtual CPU Energy Accounting

- virtual performance counter
- factor out events of other VMs
  - obtain hardware counters and subtract advances on performance counters
Virtual CPU Energy Accounting (2)

vPMC = pPMC - δPMC_{OtherVMs}

Guest OS

vCPU

vPMC

PerfCtr

Log

log

Hypervisor

CPU PMC
cannot use the same model as the CPU
para-virtual device extension
expose disk energy meter as an extension to the virtual disk device
energy-aware guest OSes customize the device driver appropriately
resource container concept
performs scheduling based on energy criteria
each application is assigned to a resource container (accounts energy)
energy is charged to active container
if a container exhausts the energy budget of the current period, it is preempted until a refresh occurs in the next period
Outline

Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
Setup

- **Pentium D 830 → 2 cores at 3GHz (only one core enabled)**
  - 42W when idle
  - 100W under full load
- **Maxtor DiamondMax Plus 9 IDE, 160GB**
  - active power: 5.6W
  - idle power: 3.6W
synthetic stress test within a Linux Guest OS

- runs on virtual hard drive (multiplexed by the disk driver VM)
- generates heavy disk load
- *raw mode*: bypasses OS caching
- block size: 0.5KB → 32KB
► base CPU power consumption – 36W
► idle HDD power (only the client VM) – 3.5W
► active HDD power (only the client VM) – 2W
► idle CPU power – 8W
► active CPU power in the driver VM – 9W to 1W, depending on the block size
► active CPU power in the client VM – varies with block size but at a lower level
Enforcing Power Constraints

- two clients that simultaneously require disk service from the driver
- operate on distinct hard disk partition (same disk driver VM)
- active driver power limit for client VM 1: 1W
- active driver power limit for client VM 2: 0.5W
- limit is set after 45 seconds
Guest-Level Energy Allocation

- two instances of compute-intensive bzip2 in energy-unaware guest OS
  - unconstrained: more than 50W CPU power
  - guest is allotted 40W
  - enforced by the host-level subsystem
- two instances of bzip2 in energy-aware guest
  - budget is distributed among two bzip2 instances
  - 10W for the first, 30W for the second
- host-level controls enforces budgets independent of the guest’s capabilities; treats bzip2 instances proportionally
- guest-level management allows user priorities and preferences
Guest-Level Energy Allocation (2)

The graph compares the power consumption and throughput of host-level and guest-level management in VMs.

**Active Power**: The power consumption for VMs with different compression methods (VM:bzip2-1 and VM:bzip2-2).

**Throughput**: The throughput measured in blocks per second for the same VMs.

**Comparison**:
- **Host-level mgmt.**: The bars show the active power and throughput for host-level management.
- **Guest-level mgmt.**: The bars show the active power and throughput for guest-level management.

The graph illustrates how different management methods impact power and performance, highlighting the trade-offs in energy allocation and efficiency.
Outline

Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
novel framework for managing energy in multi-layers OS environments
unified energy model
energy-aware accounting and allocation
recursive energy consumption
host-level subsystem, energy-aware guest OS
Further Work

- support infrastructure to develop and evaluate PM strategies for VMs
- devices with multiple power states
- processors with support for hardware-assisted virtualization
- mulit-core architectures
Introduction

Distributed Energy Management

Prototype

Host-Level Energy Management

Virtualized Energy

Experiments and Results

Epilogue

Keywords

Questions
Keywords

- power management
- virtual machine
- hypervisor
- energy
- performance
- accounting
- allocation
- driver
- CPU
- disk
- resource container
- throttling
- L4
- L4Linux
- PMC
- performance counters
- disk requests
Jan Stoess, Christian Lang, Frank Bellosa – Energy Management for Hypervisor-Based Virtual Machines
Outline

Introduction
Distributed Energy Management
Prototype
Host-Level Energy Management
Virtualized Energy
Experiments and Results
Epilogue
Keywords
Questions