Lecture 9

Energy Management for Hypervisor-Based Virtual Machines

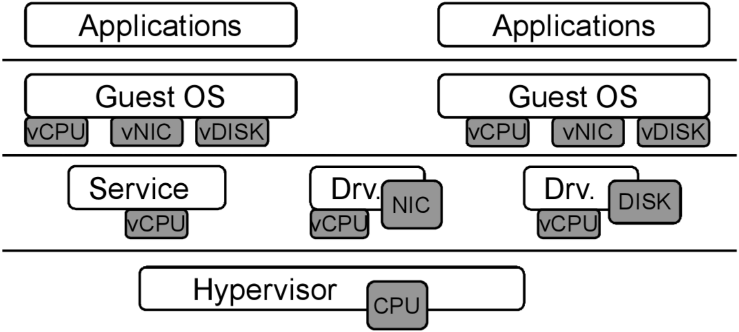
Jan Stoess, Christian Lang, Frank Bellosa

Operating Systems Practical

27 November, 2013

- 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

- 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


Applications
Applications
Guest OS
vCPU
vNIC
vDISK
Guest OS
vCPU
vNIC
vDISK
Service
vCPU
Drv.
NIC
vCPU
Drv.
DISK
vCPU
Hypervisor
CPU

- 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 hypevisor $\rightarrow$ 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)

- recursive power consumption may be high
- non-partitionability (temperature)
- current virtualization solutions disregard most energy-related aspects of the hardware platform

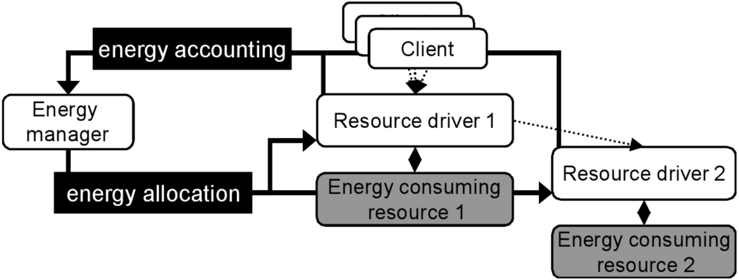
PM Framework

- 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

- 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 contraints may be partitioned from global notions to local, component-wise ones

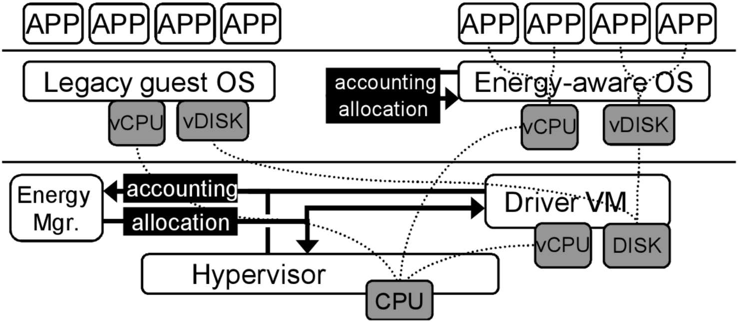
- OS = set of components (control a hardware device, export a service library, provide a software resource)
- separate policy from mechanism
- energy management – feedback loop
  - determine power consumption, account to the originating activities (*mechanism* – *resource driver*)
  - analyze accounting, 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


energy accounting
Client
Energy manager
Resource driver 1
Resource driver 2
energy allocation
Energy consuming resource 1
Energy consuming resource 2

- 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)

- 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


APP APP APP APP
APP APP APP APP
Legacy guest OS
vCPU
vDISK
accounting
allocation
Energy-aware OS
vCPU
vDISK
Energy Mgr.
accounting
allocation
Driver VM
vCPU
DISK
Hypervisor
CPU

- 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

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

- 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

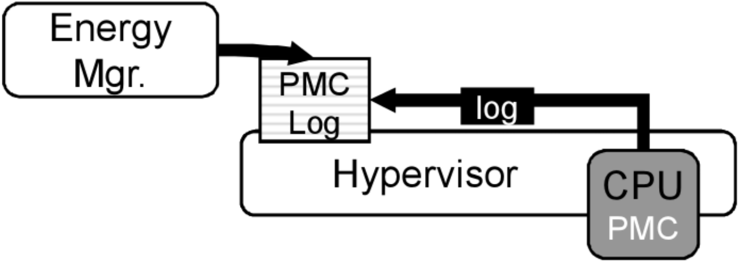
- time-based approach
- attribute energy consumption to device states (active and idle)
- active when transferring data: how long does it take?
- time = size / disk transfer rate
- ignore seek time; it remains sufficiently accurate

Outline

- access consumption is charged directly to each request
- idle consumption is alloted to all client VMs
- allocation mechanisms are exposed
  - CPU throttling
  - disk request shaping

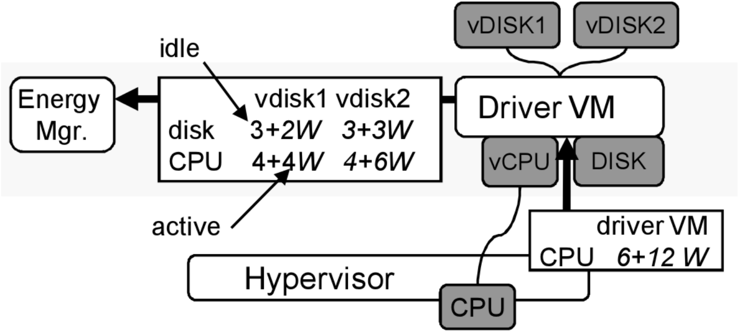
- trace performance counter events in the hypervisor
- sent to user-space energy manager
- uses a memory mapped buffer
- separate mechanism from policy
- energy manager is invoked every 20 ms
- takes into account weights
- splits computation among idle cost and access cost

CPU Energy Accounting (2)



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

- accounting must time into account energy spent recursively in the virtualization layer
- each driver determines the energy spent and passes it to the client
- currently only required for the virtual disk driver in the driver VM
- instrumented the translation module to determine active and idle CPU energy per client VM


vDISK1
vDISK2
idle
Energy Mgr.
vdisk1 vdisk2
disk 3+2W 3+3W
CPU 4+4W 4+6W
Driver VM
vCPU
DISK
active
driver VM
CPU 6+12 W
Hypervisor
CPU

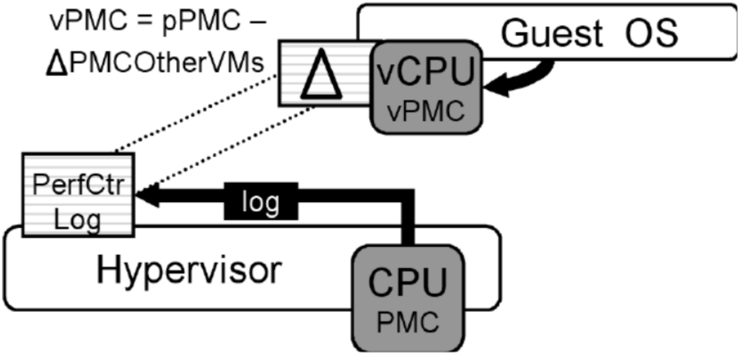
- hypervisor provides throttling the CPU
- the energy manager alters CPU shares tor virtual processors
- add *idle virtual processor* – guarantees to spend time in halt

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

- 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

- application specific energy management
- accounting and control for virtual devices

- virtual performance counter
- factor out events of other VMs
  - obtain hardware counters and subtract advances on performance counters

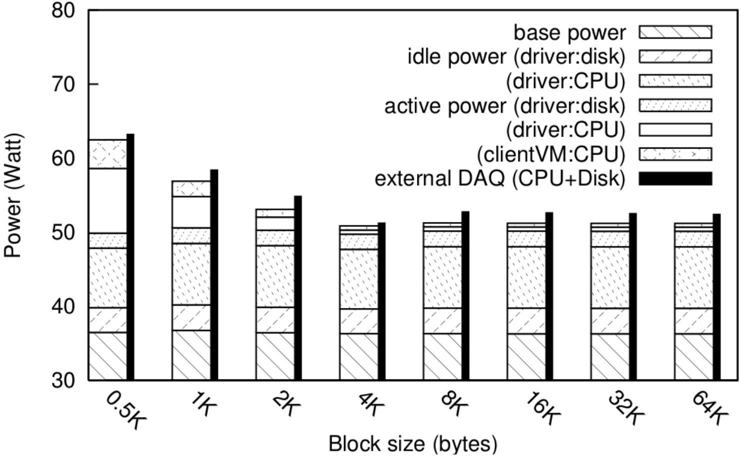

vPMC = pPMC –
ΔPMCOtherVMs
Guest OS
Δ
vCPU
vPMC
PerfCtr
Log
log
Hypervisor
CPU
PMC

- 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

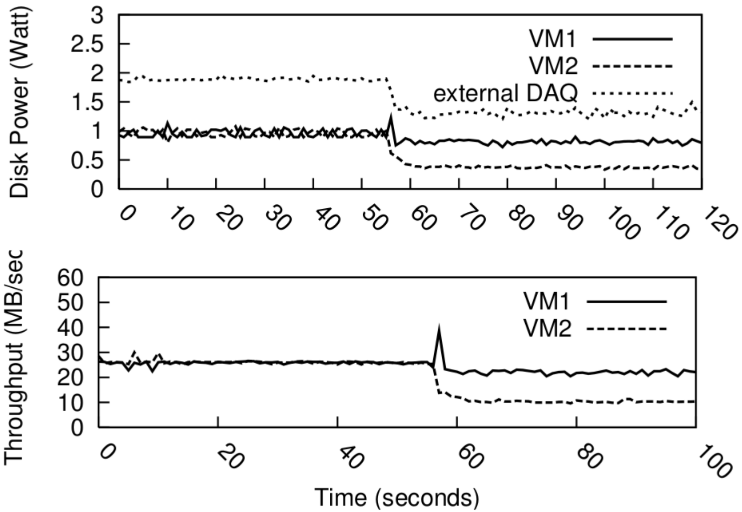
- Pentium D 830 $\rightarrow$ 2 cores at 3GHz (only one core enabled)
  - 42W when idle
  - 100W under full load
- Maxtor DiamondMax Plus 9 IDE, 160GB
  - active power: 5.6W
  - idle power: 3.6W

- syntetic 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 $\rightarrow$ 32KB

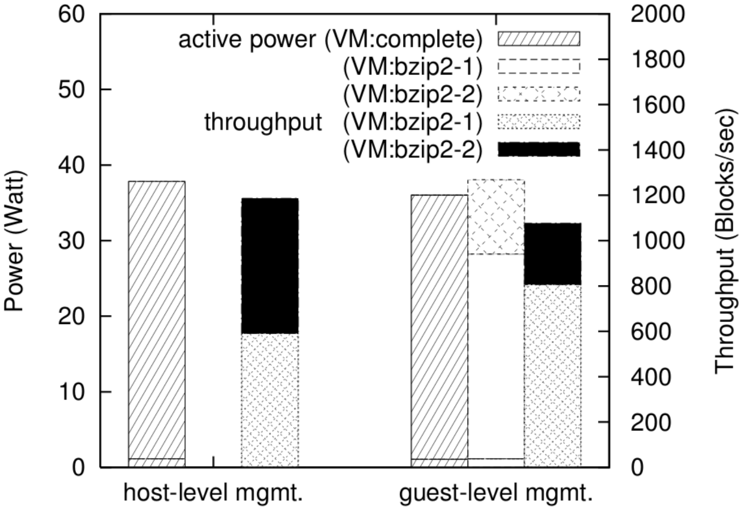

80
70
60
50
40
30
Power (Watt)
base power
idle power (driver:disk)
(driver:CPU)
active power (driver:disk)
(driver:CPU)
(clientVM:CPU)
external DAQ (CPU+Disk)
0.5K
1K
2K
4K
8K
16K
32K
64K
Block size (bytes)

- 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

- 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


Disk Power (Watt)
3
2.5
2
1.5
1
0.5
0
VM1
VM2
external DAQ
0 10 20 30 40 50 60 70 80 90 100 110 120
Throughput (MB/sec
60
50
40
30
20
10
0
VM1
VM2
0 20 40 60 80 100
Time (seconds)

- two instances of compute-intensive bzip2 in energy-unaware guest OS
  - unconstrained: more than 50W CPU power
  - guest is alloted 40W
  - enforced by the host-level subsystem
- two instances of bzip2 in energy-aware guest
  - budged 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


active power (VM:complete)
(VM:bzip2-1)
(VM:bzip2-2)
throughput (VM:bzip2-1)
(VM:bzip2-2)
Power (Watt)
Throughput (Blocks/sec)
0
10
20
30
40
50
60
0
200
400
600
800
1000
1200
1400
1600
1800
2000
host-level mgmt.
guest-level mgmt.

- 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

- 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

- 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