BitWatts

Process-level Power Estimation for VM-based Systems

Source

This presentation describes following article:

Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loic Huertas, Romain Rouvoy, Anita Sobe: “Process-level Power Estimation in VM-based Systems”

Problem definition

- Fine-grained, per-process power usage estimation
- Virtualisation support
  - Process might run in a virtual machine
  - The virtual machine might run in another virtual machine
  - Multiple processes might run in the same virtual machine
- Multi-core and multi-processor
- Distributed
  - Aggregate data from multiple physical hosts
- Application-agnostic

Process definition - continued

- No hardware investments
- Low power- and performance overhead

Motivation

- Power consumption-aware pricing models
- Environmental issues
  - Large data centers contribute 2% of global greenhouse gas and consume as much power as hundreds of thousands of households
- Energy-based task scheduling and workload consolidation
- Identify applications that consume most power and create energy-efficient software

Existing solutions

- Hardware solutions
  - Expensive to deploy
  - Coarse granularity (system-wide or per-device)
- Software solutions
  - Sampling the activity of applications and measure the consumption of entire system using hardware sensors
  - No support for virtualisation: power consumption can be reported per entire virtual machine only

Challenges of virtualization support

- No access to physical CPUs when running in virtual environment
- One can only observe the virtual CPU emulated by the vm’s hypervisor
- The underlying physical resources can change dynamically

Introducing BitWatts

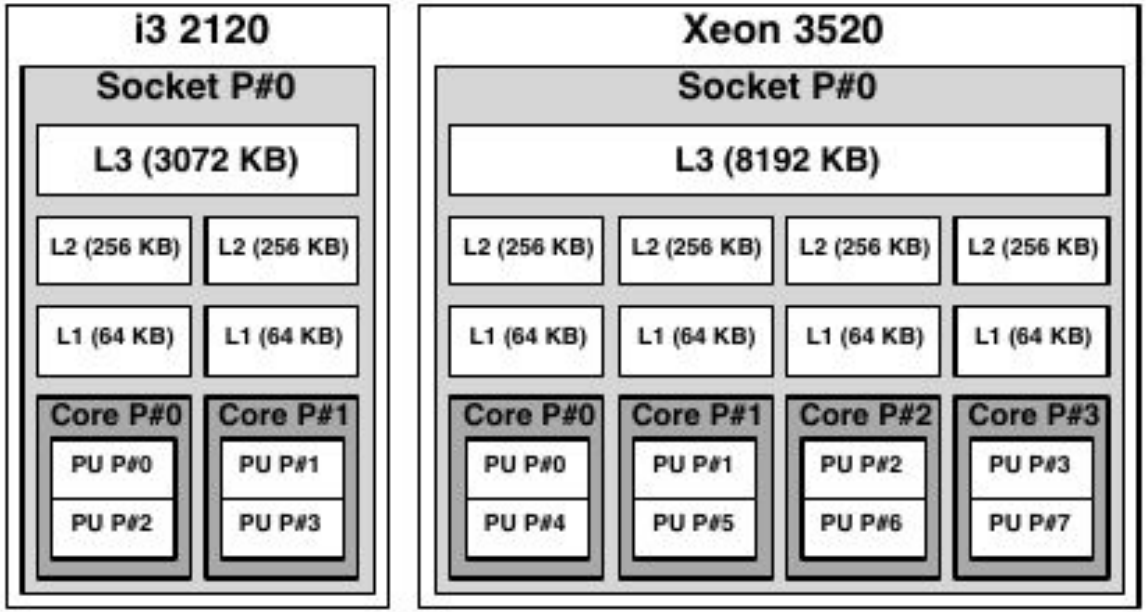
- Modular framework to estimate power consumption of software processes running in virtualized environments
- Accounts for power extensions of modern CPUs
  - Hyper-threading
  - Multiple cores
  - Dynamic voltage and frequency scaling
  - Dynamic underclocking and overclocking
- Operates in distributed settings using publish/subscribe middleware

Existing CPU power models

- Running Average Power Limit (RAPL)
  - Available since Intel Sandy Bridge
  - Provides power consumption monitoring per CPU-package
  - Close to hardware-based monitoring
- Performance counters
  - Provided by the CPU, exposed in libpfm4 library
  - Common approach: use regression model on performance counters to determine power consumption
  - Important examples:
    - Instructions per cycle (IPC)
    - Reference cycles: number of cycles count at a reference frequency
    - Unhalted cycles: number of cycles executed

Internal complexity of the CPU

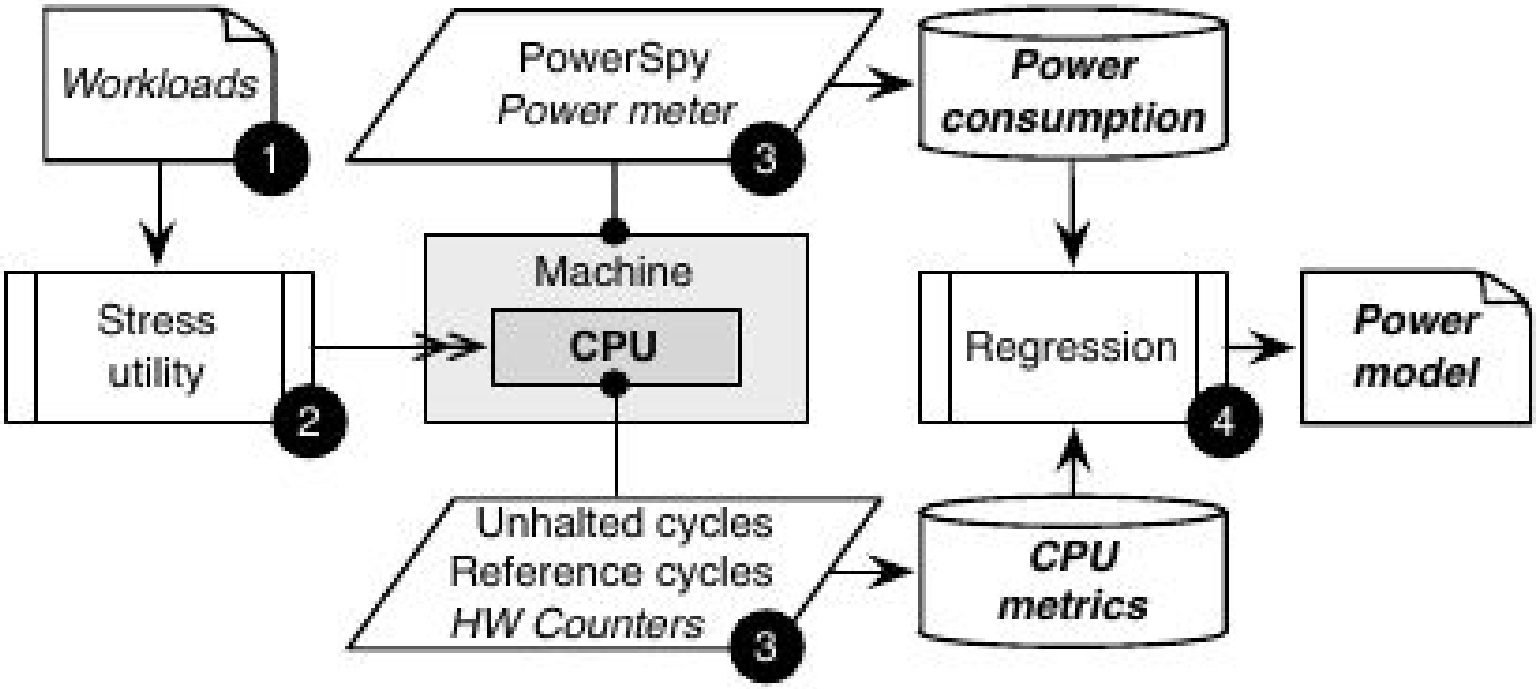
Can be obtained using hwloc software package



Power model learning

- Stress different components such as CPU, memory and disk
- Stress single core to obtain maximum frequency (TurboBoost) and observe effects of Hyper Threading
- Dynamically change the CPU load to observe the effects of frequency scaling (SpeedStep)
- Stress increasing number of cores
- Collect data using hardware performance counters
    - The counters used should be available on large number of processors and impose little overhead
- Use a power meter to obtain the consumption of entire machine
- Use polynomial regression to infer model parameters

Power model learning, c-ed



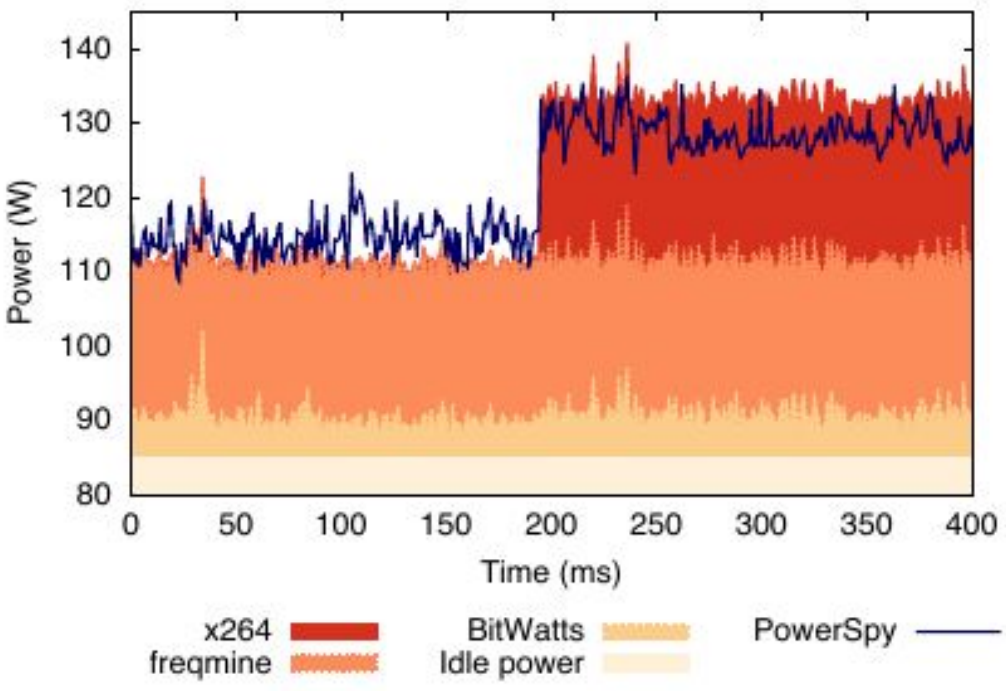
Power model learning, c-ed

$$P_{host}(f) = P_{idle}(f) + \sum_{pid \in PIDs} P_{cpu}(f, uc^1_{pid} \ldots uc^N_{pid})$$

$$P_{cpu}(f, uc^1_{pid} \ldots uc^N_{pid}) = \sum_{n=1}^{N} P_f(uc^n_{pid}).$$

$$P_{2.90}(uc_{pid}) = \frac{8.64 \cdot uc_{pid}}{10^9} - \frac{6.10 \cdot uc^2_{pid}}{10^{18}}$$

Estimated power consumption of concurrent processes



Virtual CPU power model

- Simpler architecture
- Physical cores are mapped as logical processors
- Typically no support for frequency scaling or dynamic voltage

$$P_{vm}(app) = P_{cpu}(f, uc_{vm}^{1} \ldots uc_{vm}^{N}) \cdot \frac{U_{vm}(app)}{U_{vm}(total)}$$

Host-guest and inter-VM communication channels

- VirtioSerial
  - Virtio-pci device on the host machine
  - Virtio serial port on the guest machine
- Distributed setup
  - Publish/subscribe mechanism using ZeroMQ

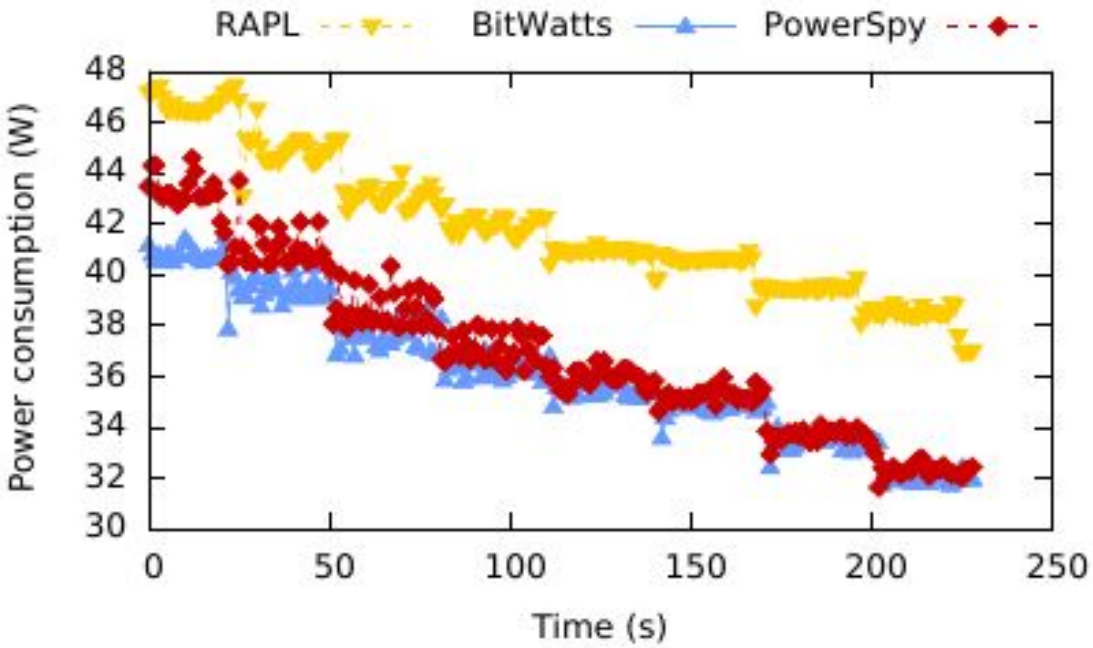
BitWatts components

- Sensor
  - Connects to software-defined power meters to collect raw measurements of system activity
  - Examples: power meters, embedded sensors, kernel statistics
- Formula
  - Receives raw data from sensors
  - Computes power estimation using specified power model
- Aggregator
  - For example per PID or per timestamp
- Reporter
  - Prints consumption estimate produced by the aggregator into a suitable format

BitWatts components, c-ed

- Describe the components pipeline that transfers data from host to guest machine
  - Raw measurements are captured by a sensor running on host machine, transformed using formula and aggregator and send to virtio-pci device by a reporter
  - Virtio serial port is exposed as a sensor on the guest machine

Evaluation of BitWatts



Evaluation of BitWatts

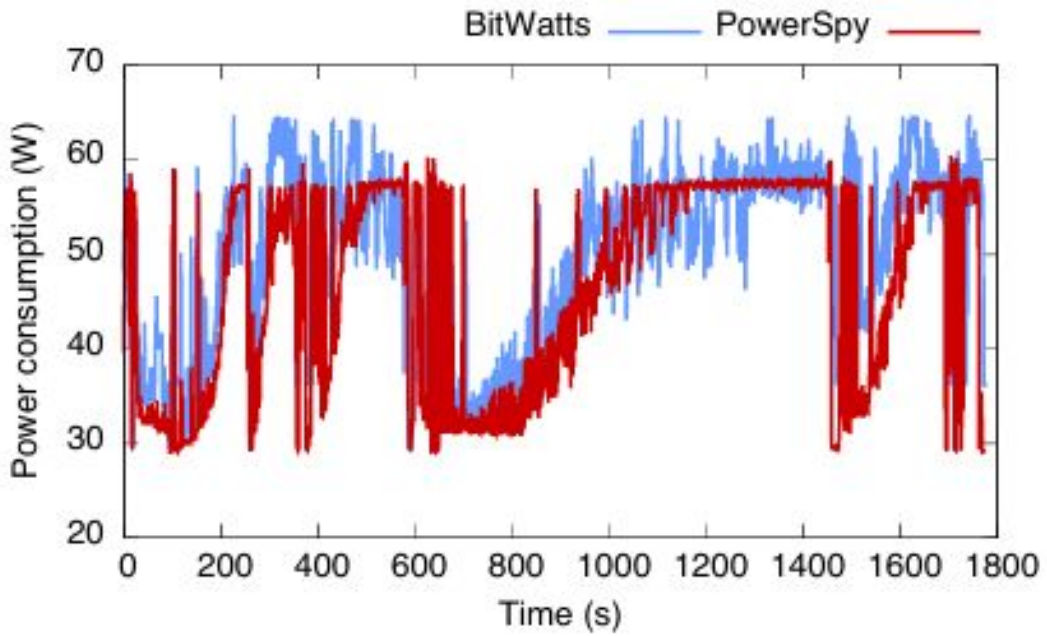


Figure 15. Power consumption during the execution of SPECjbb on the *i*3 with 2 threads.

Evaluation of BitWatts

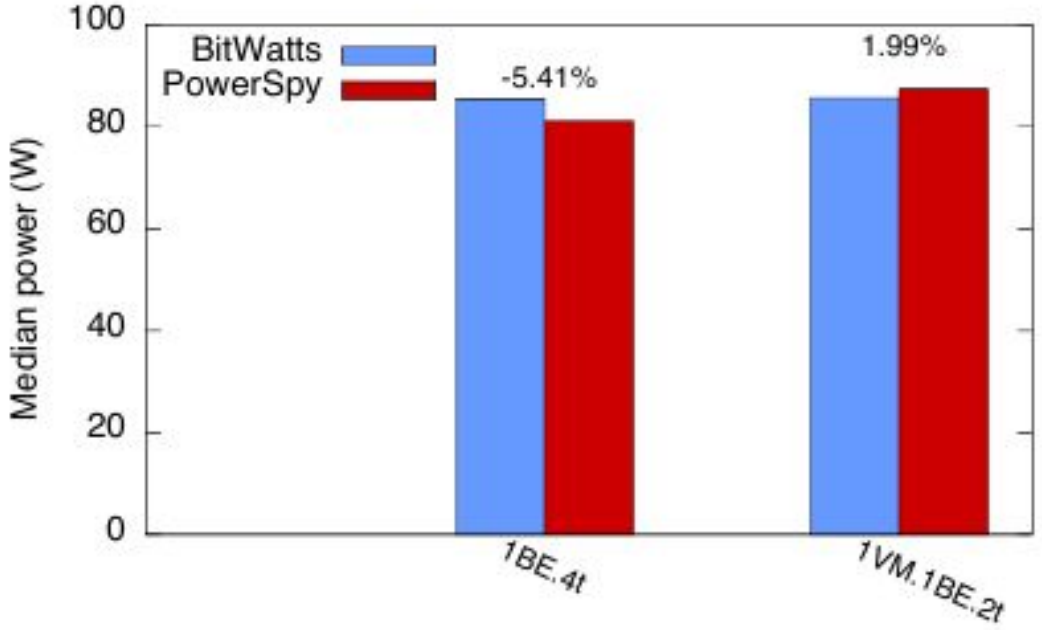


Figure 17. Median power consumption for SPECjbb on *i*3 for a distributed setup, virtualized and non-virtualized.

Estimation of BitWatts

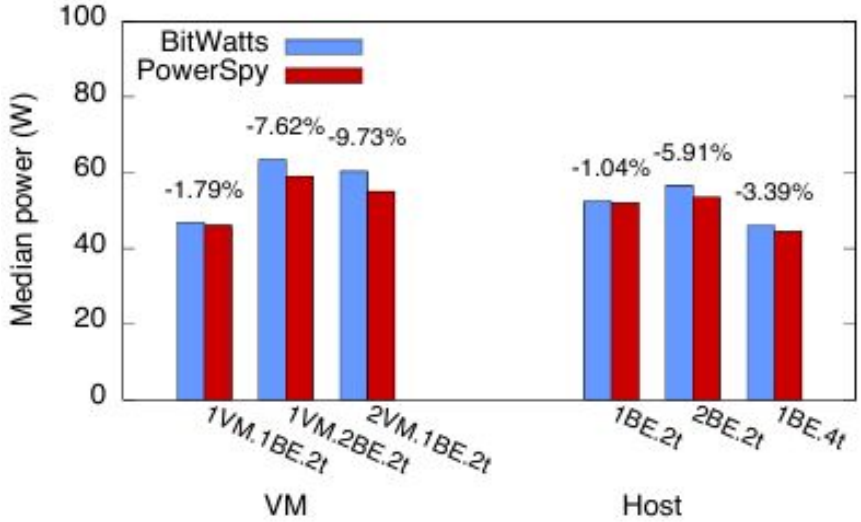


Figure 16. Median power consumption for SPECjbb on $i3$ with different resources assigned to a single or multiple VMs on one host.