云计算技术与工程

数据中心能耗与节能技术

徐敏贤

副研究员
云计算研究中心
中国科学院深圳先进技术研究院
http://www.minxianxu.info/vcc

自古逢秋悲寂寥，我言秋日胜春朝。晴空一鹤排云上，便引诗情到碧霄。
——（唐）刘禹锡

内容回顾

- Q1: 数据中心资源分配包括私有（private）和共享（shared）分配，两者各有什么优缺点?

- Private Assignment
  - Simplicity
  - Performance isolation (?)
  - Security isolation (?)
  - Allows for Hardware specialization
  - Failures & maintenance
  - Low utilization
- Shared Assignment
  - flexibility → high utilization
  - helps with failures, maintenance, and provisioning

内容回顾

- Q2: 集群管理器的设计（cluster manager）的主要作用是什么？设计（design）时有哪几种典型的架构（architecture）？

- The manager schedules apps on shared resources
  - Manager allocates and assigns specific resources
  - Considering performance, utilization, fault tolerance, priorities, …
- Architecture:
  - **Centralized:**
    - A single manager responsible for the entire cluster
  - **Two-level:**
    - One central master and multiple framework schedulers
  - **Distributed:**
    - Multiple concurrently operating sched. agents (partitioned or shared state)

内容回顾

- Q3: 导致资源超卖（overprovisioning）和低载（underutilization）的主要原因有哪些?

- Overprovisioning
  - Service provider provides more resources than needed to ensure quality of service
- Underutilization
  - When software changes, OS changes, platform changes…
  - Requests variances
  - Interference
  - …

内容回顾

- Q4: 自动扩展方法的核心思想是什么？设计自动扩展方法时有哪些权衡（trade-offs）因素？

- Autoscaling
  - Monitor app performance or server load
  - Add or remove to meet performance goal
  - Feedback-based control loop
- Tradeoffs
  - Monitor performance or resource utilization?
  - Key parameters for controller design?
  - Autoscale within (scale-up) or across servers (scale-out)?
  - Reactive or predictive control?

目录 Contents

注：本章部分内容选自《The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines》一书第二版

Why Are Power & Energy Important?

- Battery life for mobile devices
- Reliability at high temperatures
- Power density (cooling)
  - Limits compaction & integration
- Cost
  - Energy cost
  - Cost of power delivery, cooling system, packaging
- Environmental issues
  - IT responsible for 0.53 billion tons of CO2 in 2020

Reminder: Metrics

- Energy (Joules) = Power (Watts) * Time (sec)
  - Power is limited by infrastructure (e.g., power supply)
  - Energy: what the utilities charge for or battery can store
- Power density = power/area
  - The major metrics for the cooling system
- Combined metrics
  - How to tradeoff performance for power savings
  - TPS/W, energy x delay (EDP), energy x delay^2 (EDP^2), …

Reminder : Power Consumption in ICs

$$P = C*Vdd^2*F_{0\rightarrow1} + Tsc*Vdd*Ipeak*F_{0\rightarrow1} + Vdd*I_{leakage}$$

- Dynamic or active power consumption
  - Charging and discharging capacitors
  - Depends on switching activity
- Leakage current or static power consumption
  - Leaking diodes and transistors
  - Gets worse with smaller devices and lower Vdd
  - Gets worse with higher temperatures

A Sample of Power Optimizations

$$P = C*Vdd^2*F_{0\to1} + Tsc*Vdd*Ipeak*F_{0\to1} + Vdd*I_{leakage}$$

Average power, peak power, power density, energy-delay, …

CIRCUITS

- Voltage scaling/islands
- Clock gating/routing

Clock-tree distribution, half-swing clocks

- Redesigned latches/flip-flops

pin-ordering, gate restructuring, topology restructuring, balanced delay paths, optimized bit transactions

- Redesigned memory cells

Low-power SRAM cells, reduced bit-line swing, multi-Vt, bit line/word line isolation/segmentation

- Other optimizations

Transistor resizing, GALS, low-power logic

ARCHITECTURE

- Voltage/freq scaling (DVFS,RAPL)
- Gating

Pipeline, clock, functional units, branch prediction, data path

- Split instr windows
- SMT thread throttling

- Bank partitioning
- Cache redesign

Sequential, MRU, hash-rehash, column-associative, filter cache, sub-banking, divided word line, block buffers, multi-divided module, scratch

- Low-power states
- DRAM refresh-control

COMPILER, OS, APP

- Switching control

Register relabeling, operand swapping, instruction scheduling

- Memory access reduce

Locality optimizations, register allocation

- Power-mode-control

- CPU/resource schedule
- Memory/disk control

Disk spinning, page allocation, memory mapping, memory bank control

- Networking

Power-aware routing, proximity-based routing, balancing hop count, …

- Distributed computing

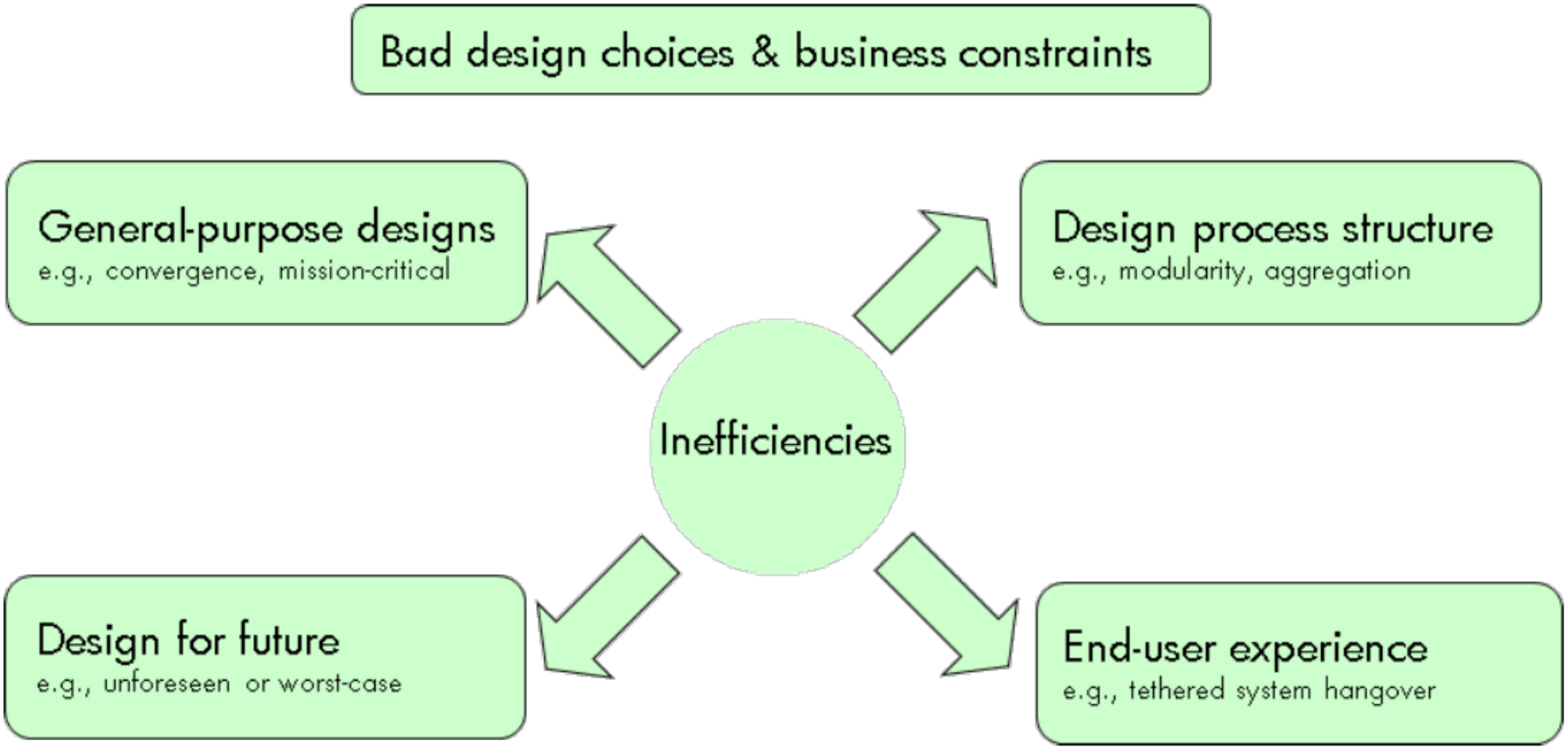
Mobile agents placement, network-driven computation

- Fidelity control
- Power API

Reducing Power/Energy

- An interdisciplinary issue
  - Circuits, architecture, software, systems
- Our focus
  - A framework to reason about power management
  - A few selected examples for
    - Server components
    - Servers
    - Datacenters

Sources of Inefficiency



Optimization Approaches

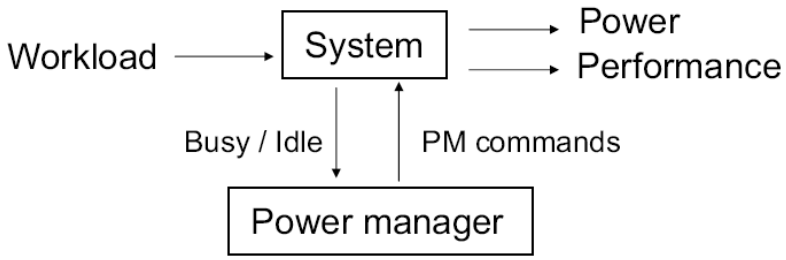
- Energy-efficient technologies
  - E.g., replace disk with flash, replace copper with optics, …
- Match power to work
  - E.g., turn-off/dial-down unused components, …
- Match work to power
  - E.g., asymmetric multicores, temp-aware scheduling, …
- Piggy-back energy events
  - E.g., shared caches, coalesced request streams, …
- Special purpose solutions
  - E.g., GPUs, ASICs, …

Optimization Approaches

- Cross layers for efficiency
  - E.g., coordinate management across rack/cluster, …
- Tradeoff some other metric
  - E.g., fidelity-aware energy management
- Tradeoff the uncommon-case
  - E.g., popular query optimization, power-supply efficiency, …
- Spend somebody's power
  - E.g., remote server offload for mobile power
- Spend power to save power
  - E.g., periodic cleanup to save energy, …

Typical Power Management

- Components have multiple power modes/states
  - Active: different levels of performance/power consumption
  - Idle: different power consumption/wake-up time
- Select power states to match constraints
  - Exploit fluctuations in use (requirements/idle times)
  - Done by the hardware, OS, compiler, and/or the user
  - Tradeoffs: power saving Vs. QoS Vs. speed of resuming

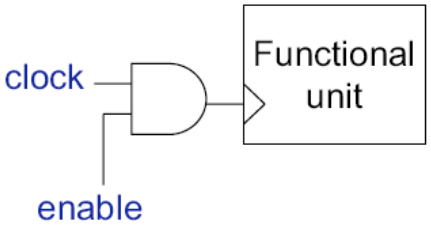


Advanced Configuration and Power Interface (ACPI)

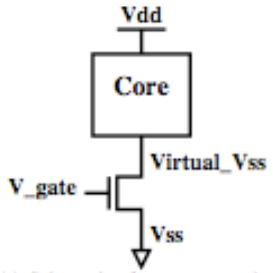
- A standard for power management of systems
  - Describes power states for system, devices, cores, …
  - Interface for software to query and manage power states
- Global system states
  - G0 working, G1 sleeping, G2 wakeup on LAN, G3 hard off
- Processor states
  - C0 is fully on
    - With P states related to DVFS stages (see following slides)
  - C1 to C3 are various idle modes
    - Clock may be stopped but state is maintained
  - C4 and beyond are various power off states
    - First the caches, then cores, and finally the whole chip

Idle Power States in Processors

- Clock gating for idle units
  - Clock is a major power contributor
  - Done automatically in most designs
  - Near instantaneous on/off behavior

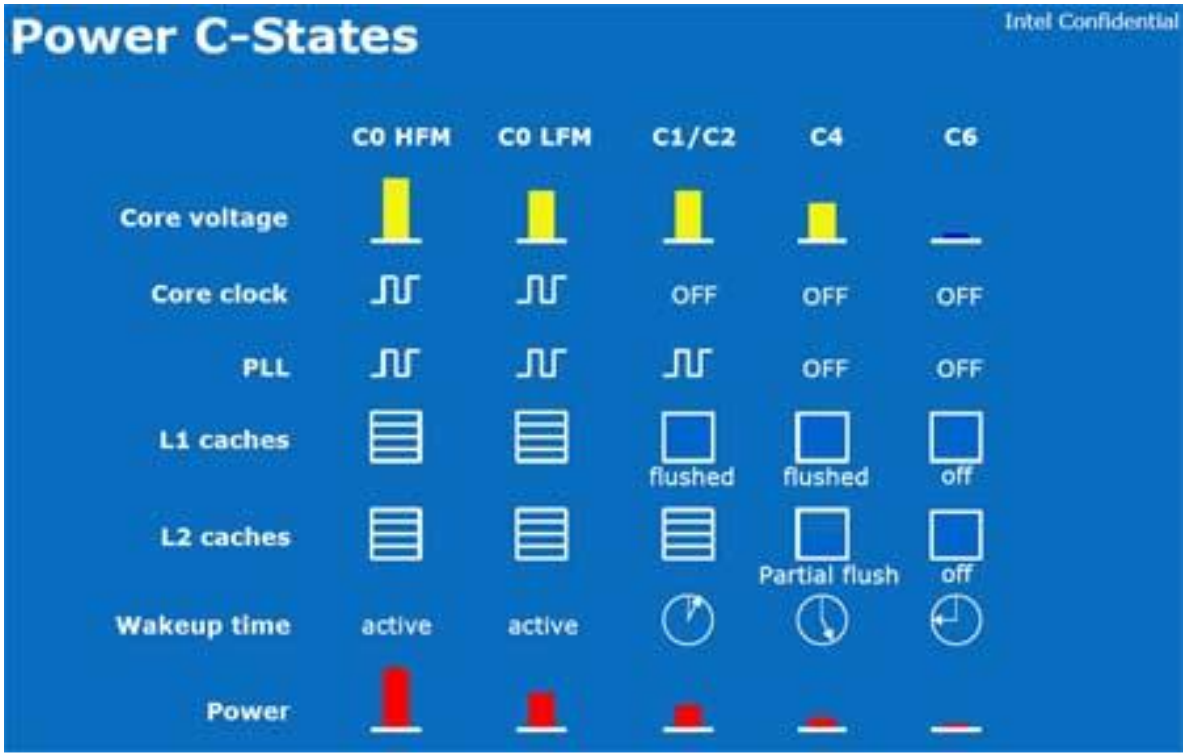


- Power gating (C4 or beyond)
  - Turn off power to unused cores/caches
  - High latency for on/off
    - Saving SW state, flushing dirty cache lines, turning off clock tree
    - Carefully done to avoid voltage spikes or memory bottlenecks
  - Area & power consumption of gate
  - Opportunity: use thermal headroom for other cores



Example C State Implementation

- Gets more complex with multi-core and power-gating

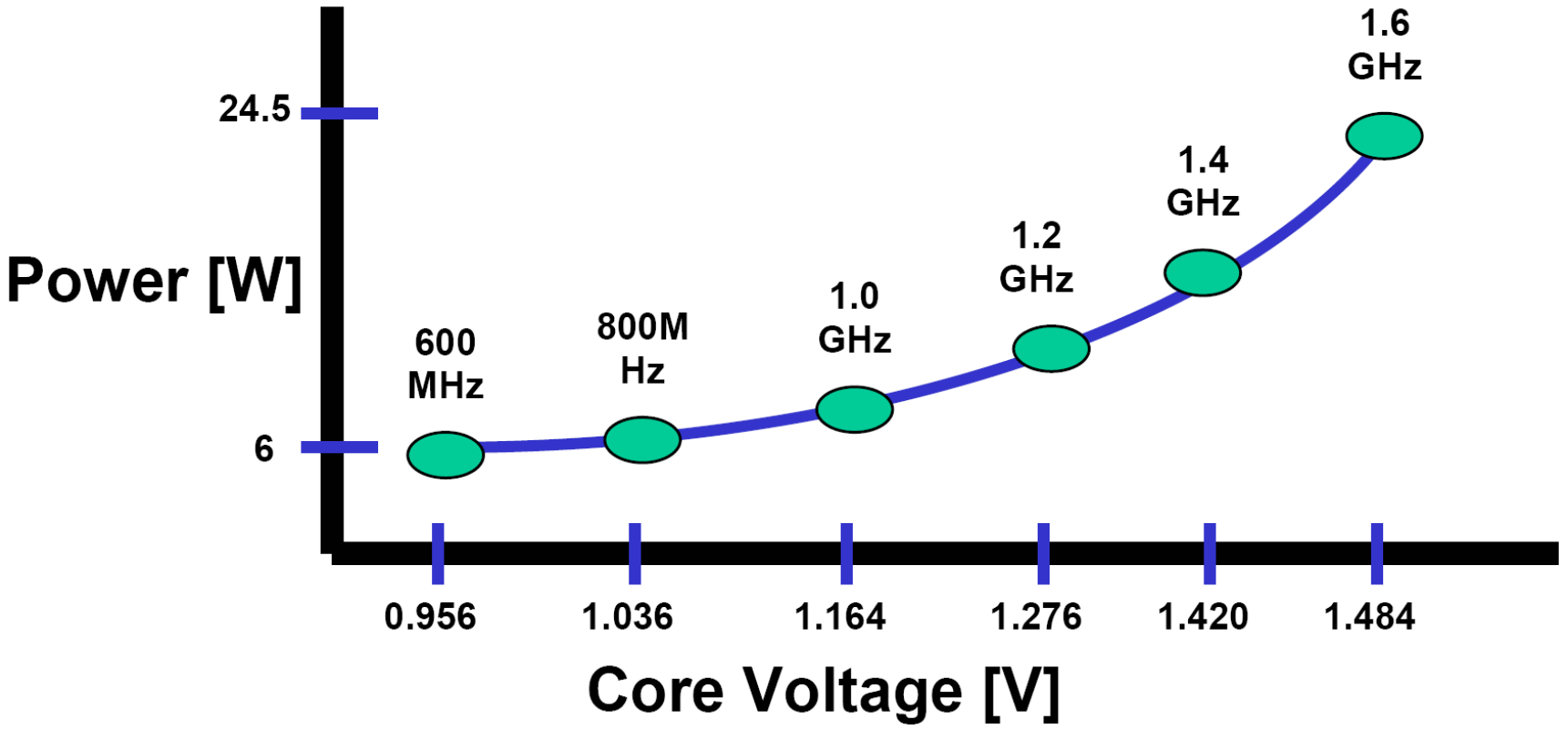


DVFS: Dynamic Voltage/Frequency Scaling

- Set frequency to the lowest needed
  - Execution time = IC * CPI / F
- Scale back Vdd to lowest for that frequency
  - Lower voltage => slower transistors
  - Power = C * Vdd2 * F
- Provides P states for power management
  - Heavy load: frequency, voltage, power high
  - Light load: frequency, voltage, power low
  - Trade-off: power savings vs overhead of scaling
  - Effectiveness limited by voltage range

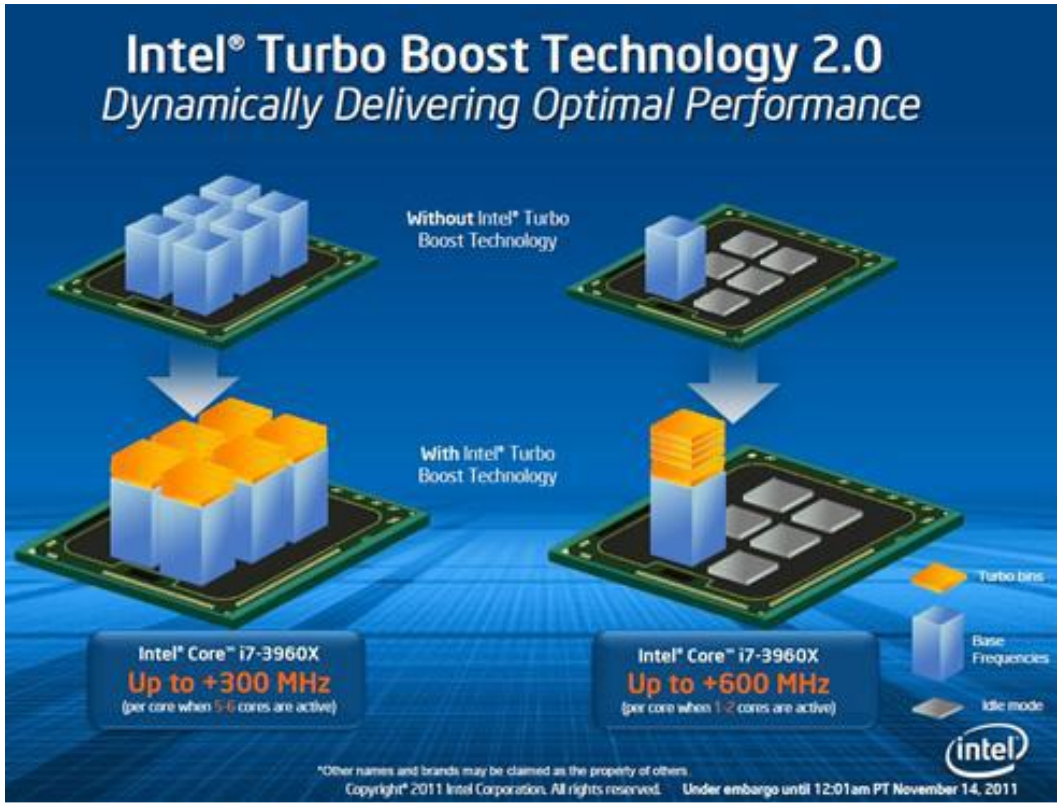
Example DVFS Implementation

- Transitions typically take a few usec



Turbo Mode

- Use power budget of idle cores to boost single thread perf



Power/Thermal Budgeting

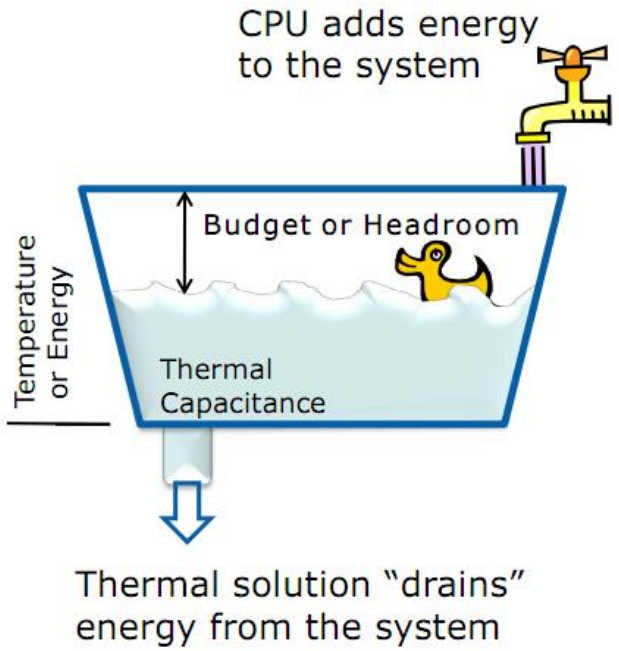
Intel® Turbo Boost Technology 2.0 Power Budget Concept

Bathtub Model

- Flow from the faucet (CPU power) …
- can exceed drain flow (thermal solution capability) …
- if the bathtub (energy or temperature)…
- is not full (at limits)…
- for a short time (depending on capacity, level and flow)

Implications

- CPU can safely operate >TDP for short periods without throttling..
- More power headroom for turbo when system has been at low power for a while (tub level is low) or ..
- Less power headroom for turbo if system has been running at high power without a rest (tub level is high)



Power Budget Management allows bursts of power for maximum performance when headroom exists

IDF2011 INTEL DEVELOPER FORUM

DDR3 Power Management

- Example: 5 states in DDR3
- Tradeoff: power savings vs resync penalty

Power State	Operating Mode	Resync -time	% Active power
Active	All modules ready	0 cycles	100%
Standby	Column multiplexers disabled	2 cycles	60%
Napping	Row decoders turned off	30 cycles	10%
Power Down	Clock sync to Controller interface turned off	9000 cycles	1%
Disabled	No refresh; data lost	Reboot	0%

Food for Thought

- Does the interleaving scheme affect DRAM power management?

- Would DVFS work for DRAM?

Broad View on DRAM Power

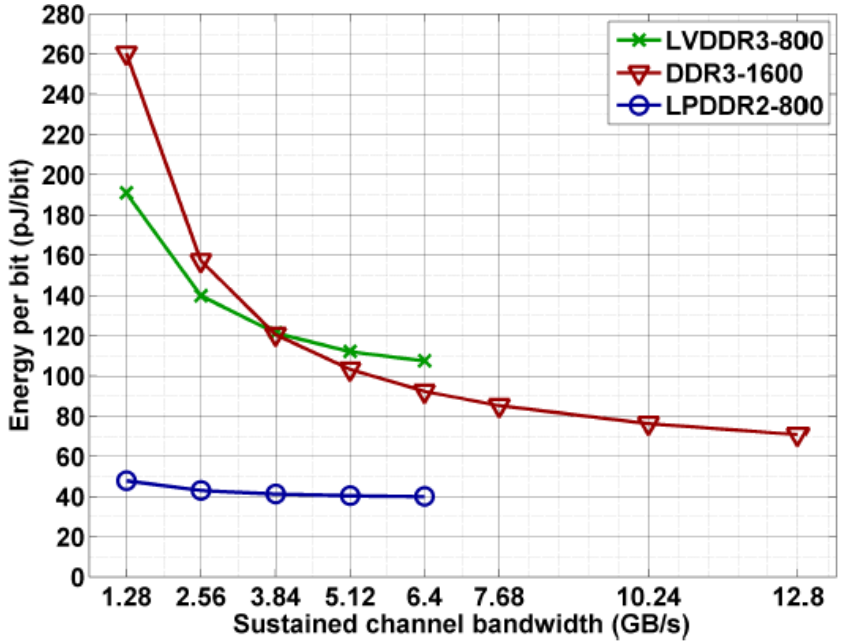
- Problem: many chips involved with DRAM access
  - i.e., x4 chips on 64-bit bus means 16 chips per access!
- Problem: emphasis on high capacity and bandwidth
  - Fast clock means high voltage
    - DDR3 is 1.5V @ 800MHz
  - Multiple ranks means high-current drivers and termination needed, very high energy per bit

Mobile DRAM: LPDDR2

- Similar internal structure
  - Same manufacturing technology, latency, array structure
- Low voltage, low frequency: 1.2V @ 400MHz
  - P = C * f * V*V
- x16 or x32 interface! (often, <64 bit channel)
- No termination, no local clock regeneration
  - 1 rank maximum, so hard to get high capacity
  - Fast exit from power-down mode (10s of ns)

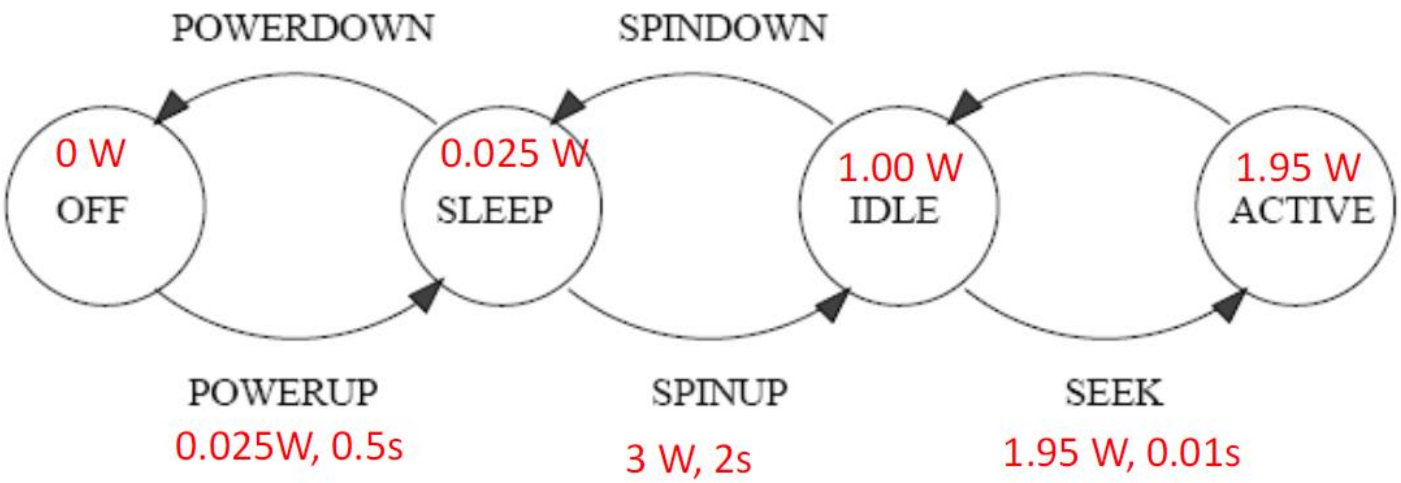
DDR3 vs LPDDR2

- 1/2 the pin bandwidth
- 1/5 the energy consumption
- Better proportionality
  - why is this important?



Disk Drive Power Modes

- Common optimization
  - Stop spinning disk when it is unused for a certain period of time



State diagram from Li, 1993.

Per-server Power Management

- Monitor and manage each server in a data-center
  - Q: how do you monitor power consumption?
- Various policies may be desirable
  - Power capping: limit power consumption to available budget
    - Why? Optimizations?
  - Conservation policy: switch to low power modes in the evenings
    - Why does this make sense?
  - Emergency issues: drop some server in lower power state on emergencies
- Other?

Cluster-level Power Management

- Power-aware load distribution to a server cluster
  - Try to create idle resources to send to low-power/off
  - Try to stay within “good” operating point of components
    - Coarser-grain power capping or temperature control
  - Interactions DVFS and inter-server load balance
  - Watch out for heterogeneity
  - Interactions with performance and more broadly SLAs
- Lots of policies
  - Predictions, economy-based, batching
  - What is the tradeoff?

Energy Proportionality

- Systems often underutilized
  - Diurnal traffic patterns, spikes, design for future growth, imbalance
- Need to have energy scale with work done
  - "Energy scale-down" or "do nothing well"
  - Unfortunately most computers are not proportional
    - Why?

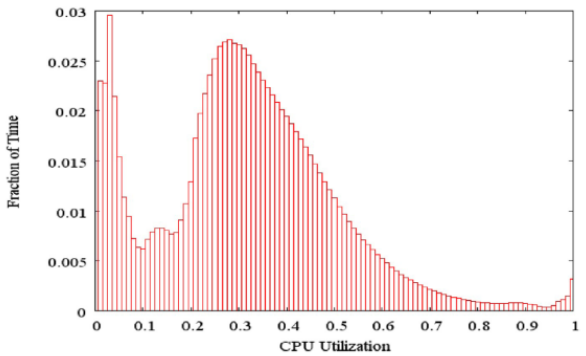


FIGURE 5.5: Activity profile of a sample of 5,000 Google servers over a period of 6 months.

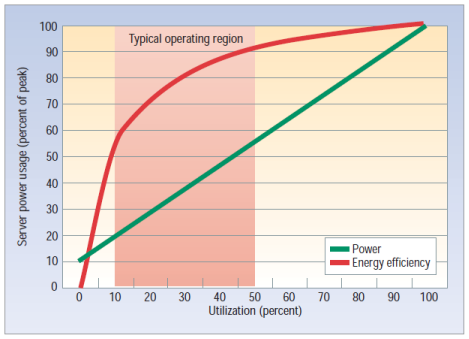


Figure 4. Power usage and energy efficiency in a more energy-proportional server. This server has a power efficiency of more than 80 percent of its peak value for utilizations of 30 percent and above, with efficiency remaining above 50 percent for utilization levels as low as 10 percent.

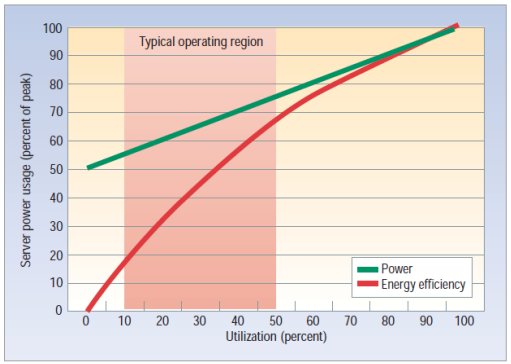


Figure 2. Server power usage and energy efficiency at varying utilization levels, from idle to peak performance. Even an energy-efficient server still consumes about half its full power when doing virtually no work.

In case you are wondering: Humans are More Energy Proportional

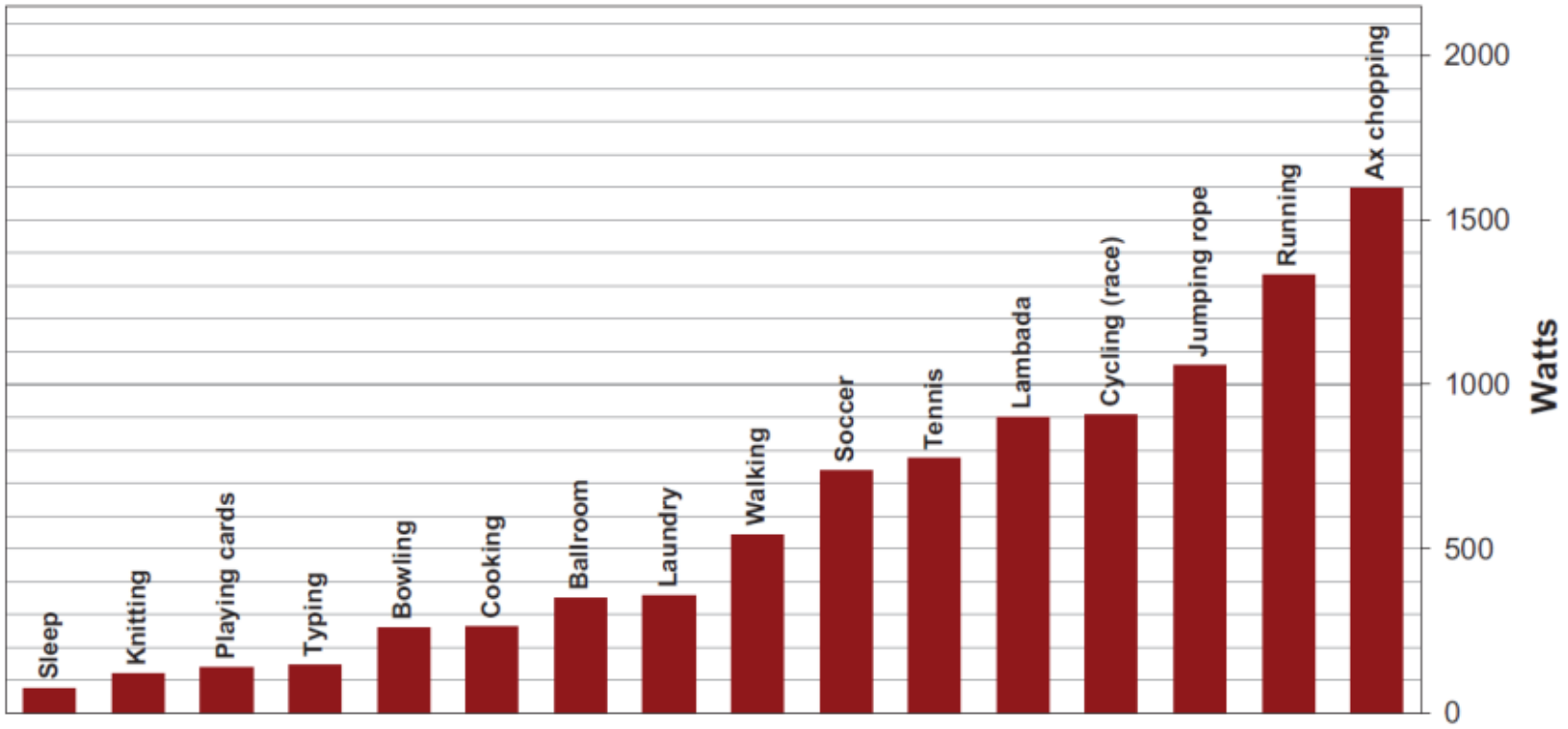


FIGURE 5.7: Human energy usage vs. activity levels (adult male) [52].

Optimizations for Energy Proportionality

- Component-based approach
  - Improve proportionality of *all* components
    - Processors: use DVFS or power-gating
    - Missing similar capabilities for memory/disks
  - Improve efficiency range of power distribution
- System-based approach
  - Scale-down: consolidate work & power-down idle servers
    - Issue: latency QoS robustness
    - Difficult if servers are part of the global file system
  - Monitor app performance & manage DVFS/core power
    - Global & fine-grain control loop for power states

Pegasus: Power Management @Scale

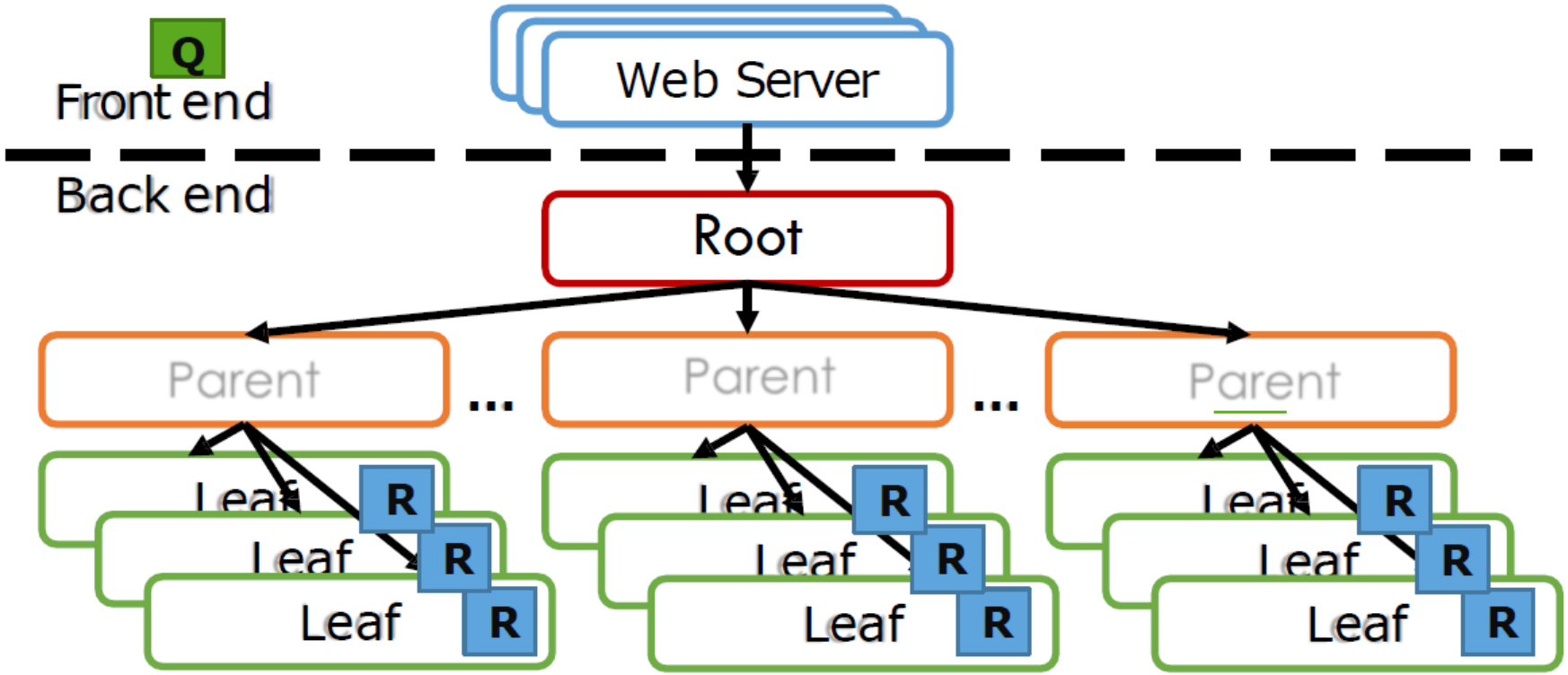
- Energy waste is caused by overachieving performance
  - Lower latency than what is necessary
- **Solution**: Match power to work (Service Level Objectives – SLOs)
  - End-to-end latency monitoring
  - Fine-grain power saving mechanism (RAPL)
- Controller based on feedback control
  - 20-30% power savings in Google production search without SLO violations

OLDI Workloads

- On-Line Data Intensive workloads: user-facing interactive, latency-critical
  - SLOs for tail latency
  - High fan-out with distributed state

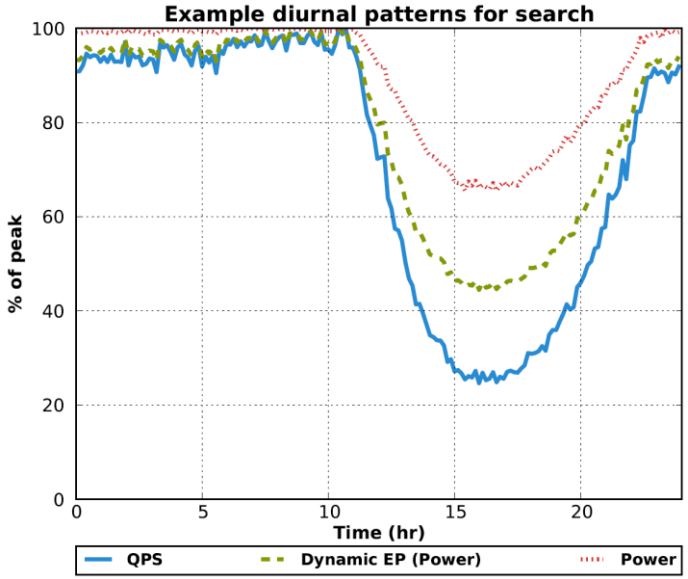
- **Focus on Google websearch**
  - Query service portion of production Google search

Websearch


Q
Front end
Back end
Web Server
Root
Parent
Parent
Parent
Leaf
Leaf
Leaf
R

The need for energy-proportionality

- Diurnal variation in cluster load and power across a 24 hour period
- Cluster not fully utilized half the time
- Gap between measured power and EP curves represent potential savings



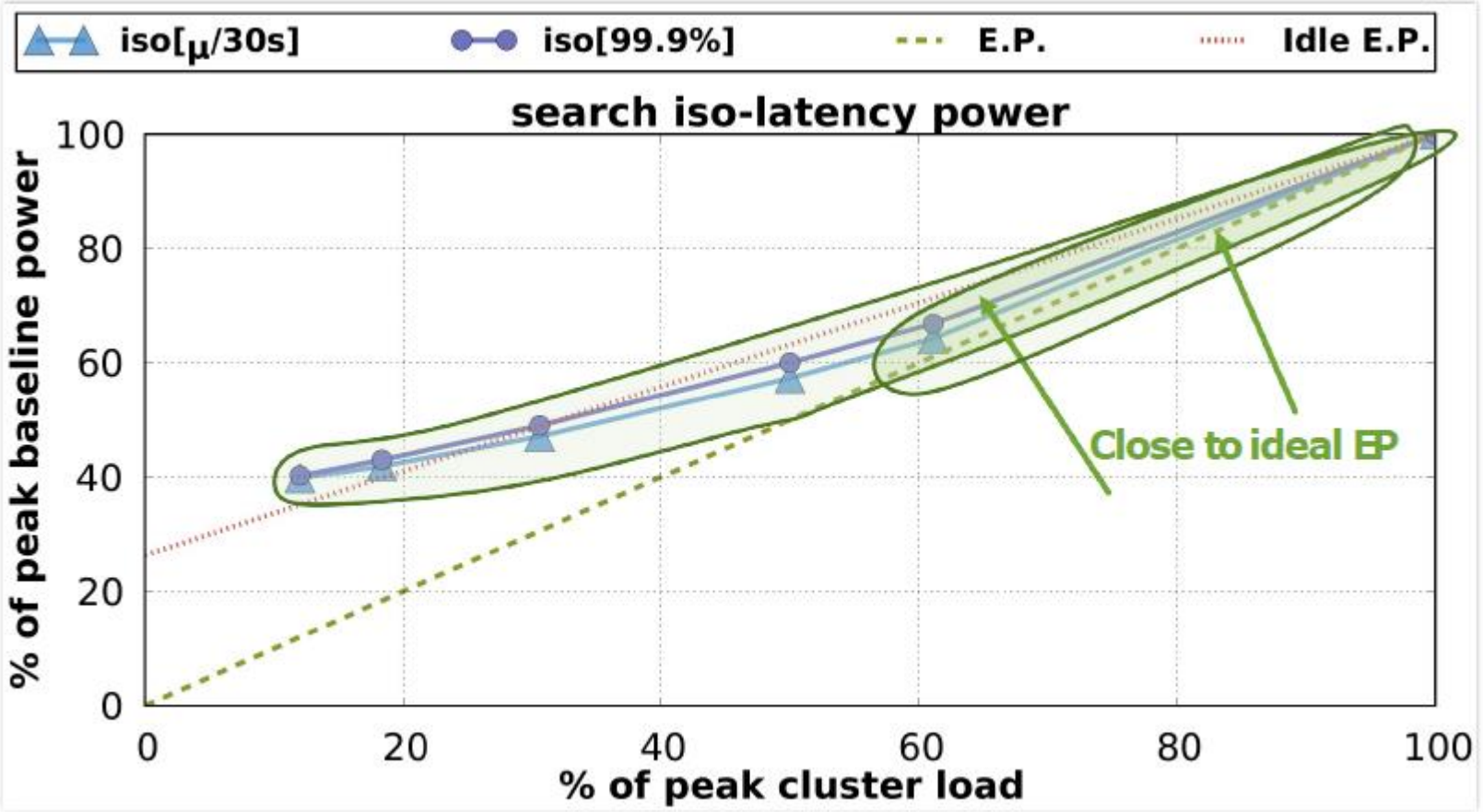
Previous Cluster-level Power Mgmt

- Consolidate load on fewer servers during low utilization
  - **Issue**: state of OLDI apps cannot fit on fewer servers
- Use very low power idle modes
  - **Issue**: OLDI request rate is always to high, e.g., >1k RPS
- Batch requests to form long enough idle periods
  - **Issue**: OLDI apps cannot tolerate msec completion times and batching delays

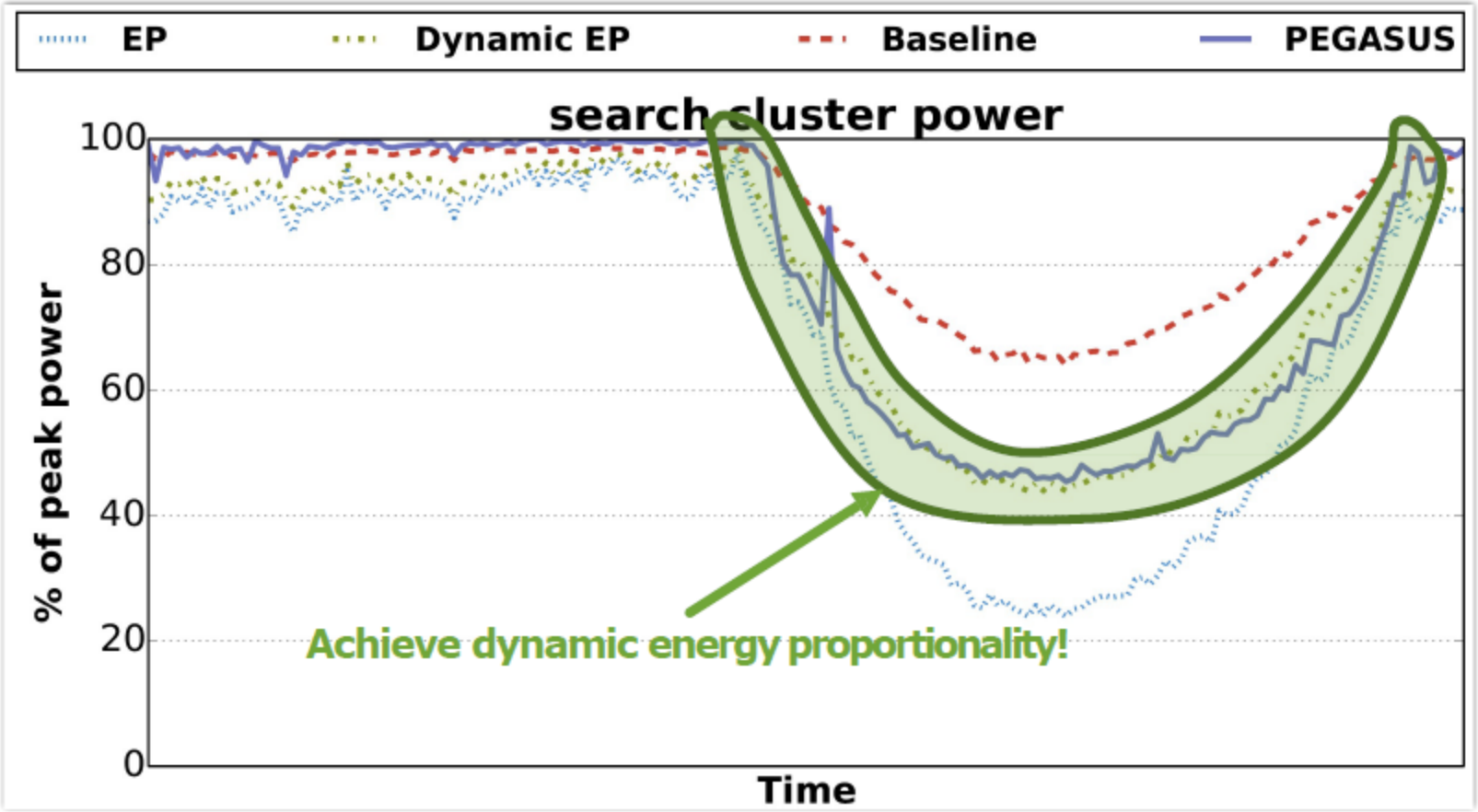
Key Assumption

- Beating the end-to-end SLO is no better than meeting it
  - I.e., lower latency than the SLO does not help
  - The end-user only cares if the search query takes a long time to load
- Iso-latency policy: use RAPL to lower power s.t. you *just* meet the latency SLO
  - RAPL: Running Average Power Limit
  - Fine-grained: power limit increments as small as 0.125W
  - Fast: <1msec delay to take effect

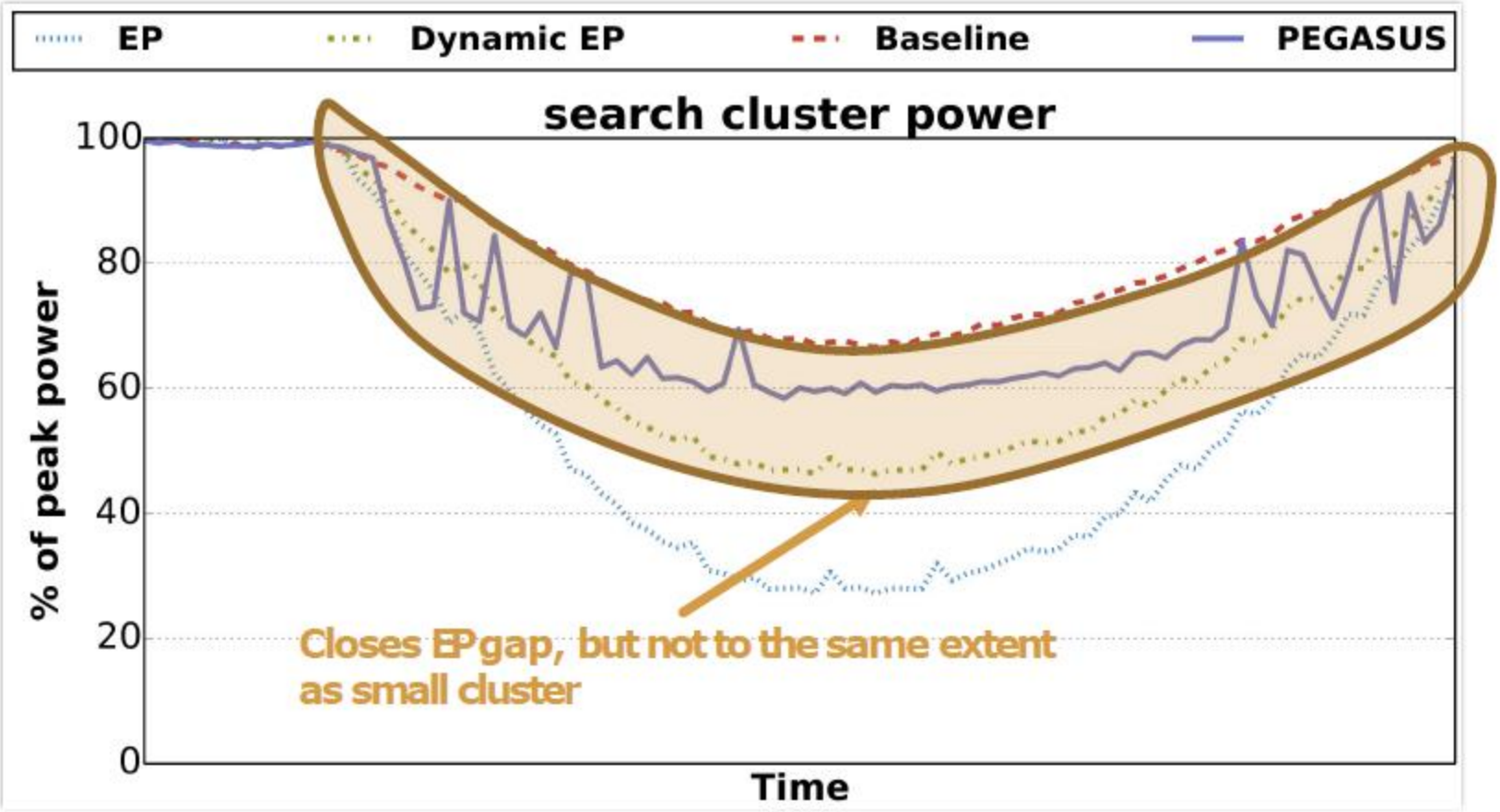
Iso-latency Potential Power Savings



Pegasus Improving Energy Proportionality

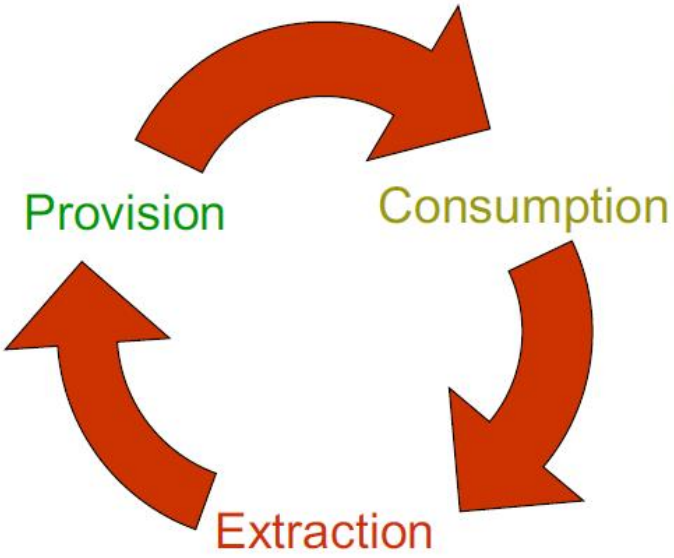


Pegasus @ Scale

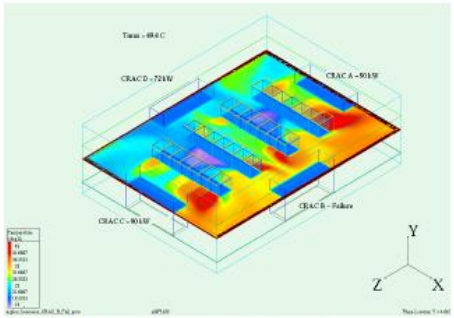


DC: Power, Energy, & Cooling

(Feeds, PDUs, UPS, etc.)



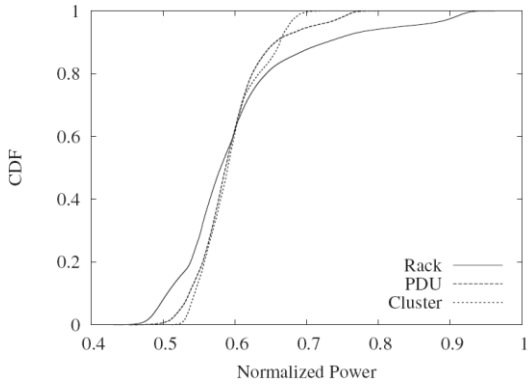
(Compute nodes)



(CRAC units, floor tiles, etc.)

Energy Provisioning

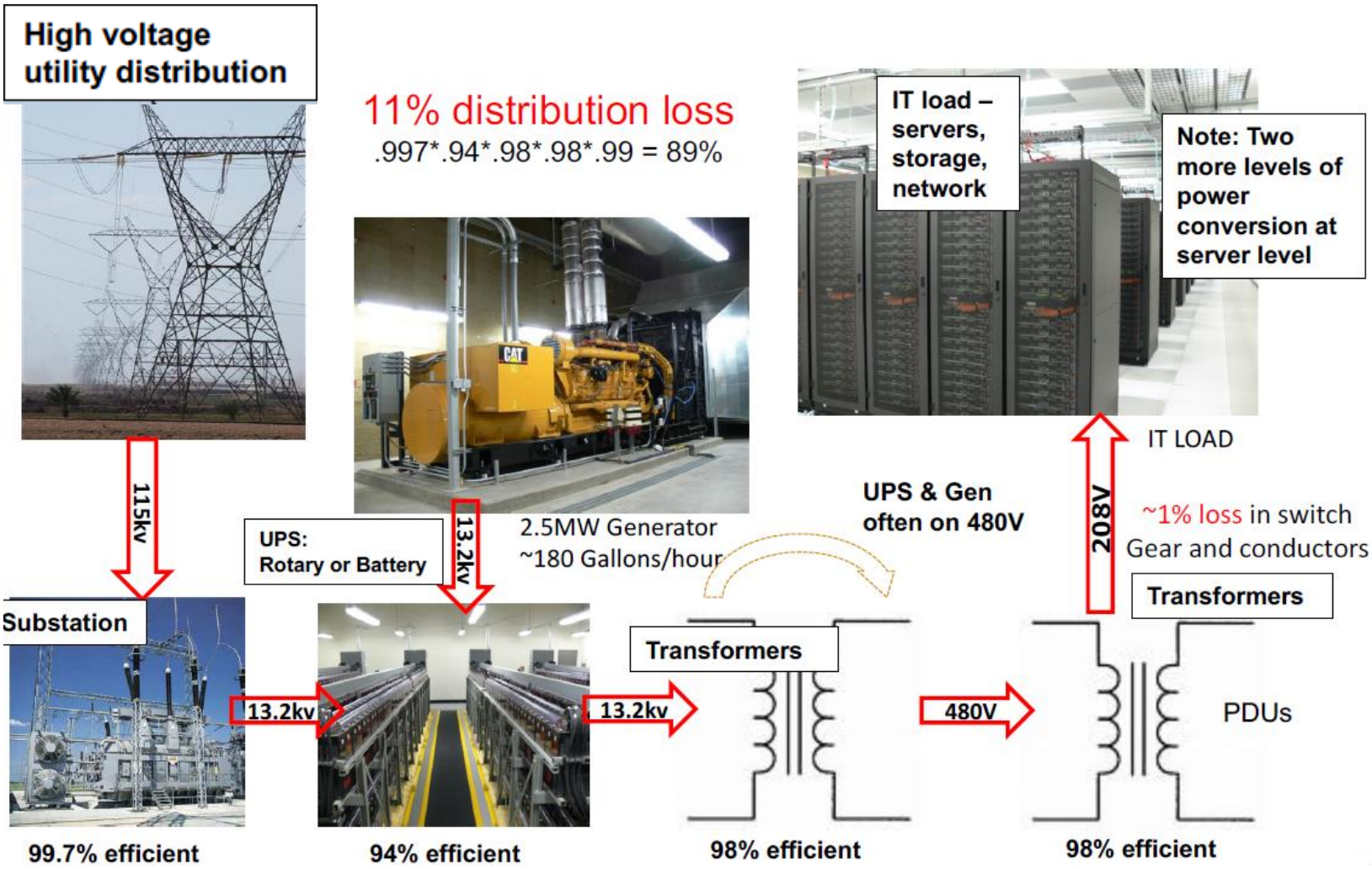
- How much power does the DC need?
  - ~$15/Watt for building costs
  - Limitations of nearby generation facilities



- Conservative: max server power x # of servers
  - But for most apps, max power never reached

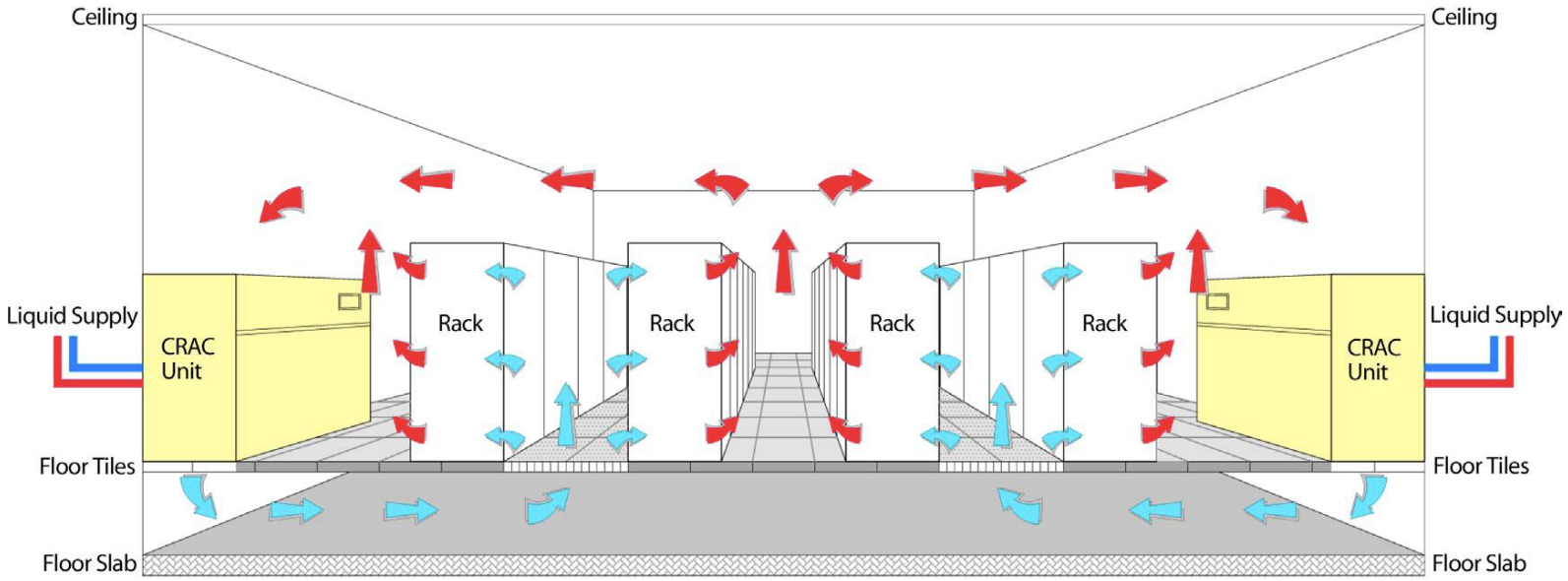
- Alternative: oversubscribe facility power
  - Using typical/max power consumption for apps
  - Requires careful management of unexpected peaks
  - Power capping

Power Distribution



Cooling: Cold/Hot Aisles

- CRAC = computer room air conditioning
  - Cold airs goes through servers and exits in hot aisle
  - Cold aisles ~18-22C, hot aisles ~35C
  - CRAC units consume significant amount of energy!



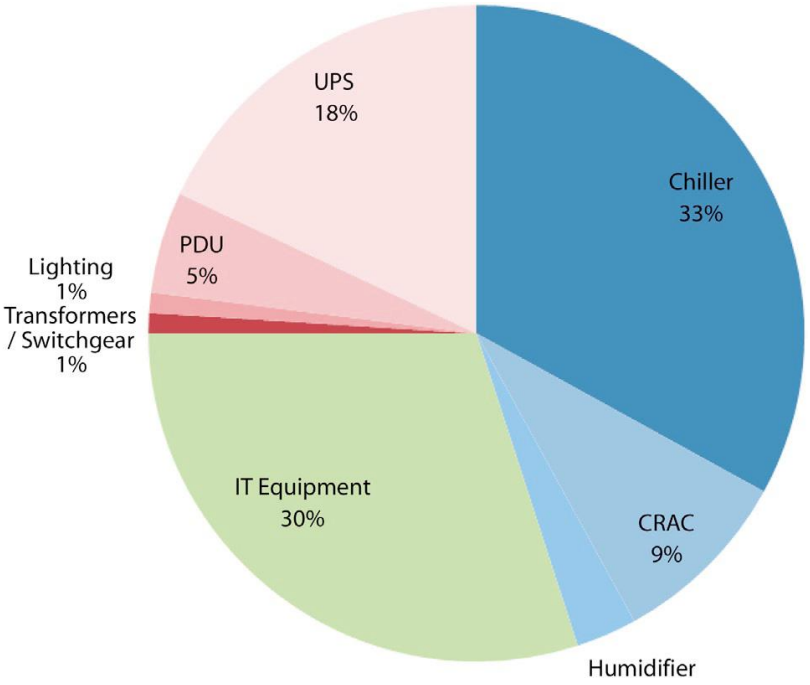
DC Energy Efficiency

$$\text{Efficiency} = \frac{\text{Computation}}{\text{Total Energy}} = \left(\frac{1}{\text{PUE}}\right) \times \left(\frac{1}{\text{SPUE}}\right) \times \left(\frac{\text{Computation}}{\text{Total Energy to Electronic Components}}\right)$$

- PUE = power usage effectiveness
  - Building power/power of IT (servers, switches etc)
  - Some DCs as bad as PUE = 3
  - Current state of the art PUE = ~1.2
- SPUE = server power usage effectiveness
  - Server power/power for CPUs, DRAM, disk, etc
  - Most servers have SPUE = 1.6
  - State of the art SPUE = 1.2
- If PUE=SPUE=1.2 => 30% of energy is "wasted"

Energy Use in a DC

- Cooling infrastructure is a major contributor
  - Picture from a PUE=3 data center
  - Current datacenters: PUE: 1.1 to 2



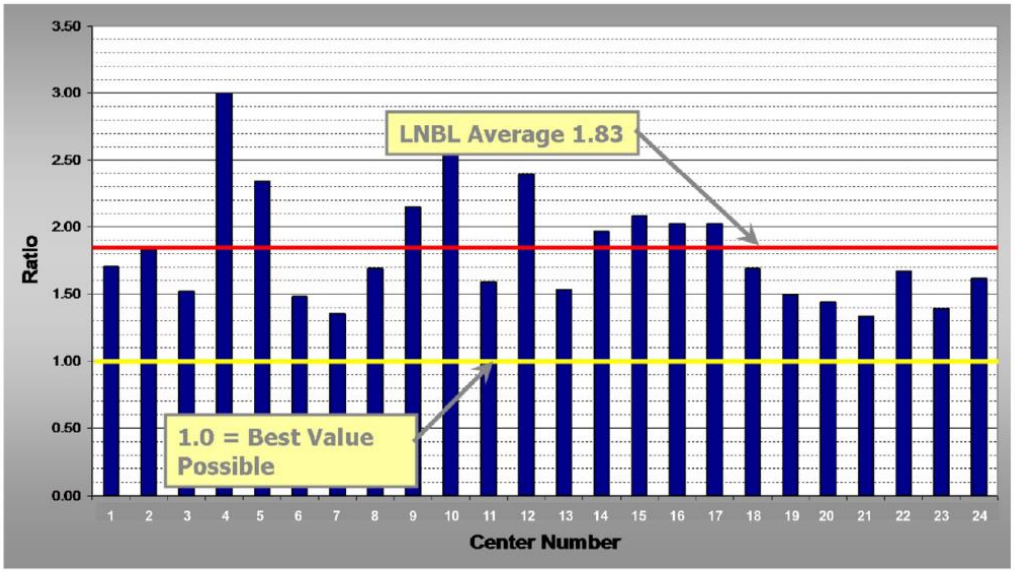


FIGURE 5.1: LBNL survey of the power usage efficiency of 24 datacenters, 2007 (Greenberg et al.)

PUE Optimizations: Power Distribution

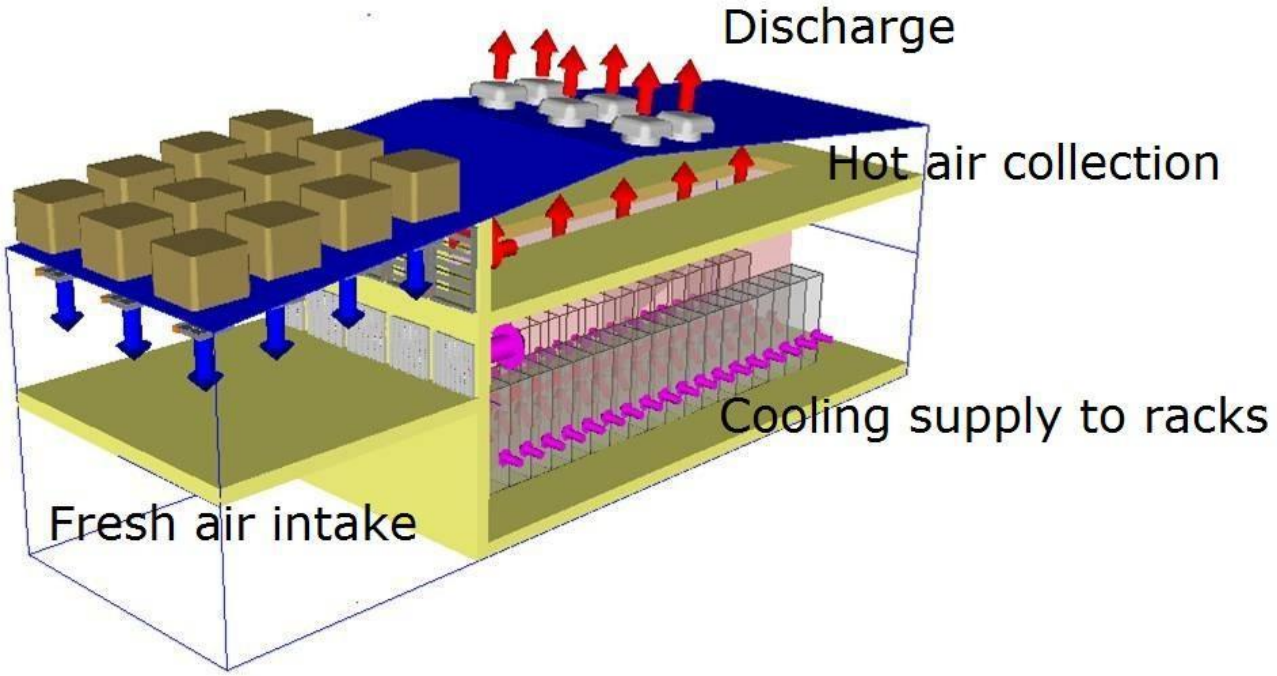
- Reduce number/increase efficiency of conversions
- High V close to load
- DC distribution within rack
- Per-rack or per-server 12V battery
  - No UPS
  - Can also use “UPS” capability to deal with power spikes
    - But watch out for impact on battery lifetime

PUE Optimizations: Cooling

- Higher DC temperatures
  - E.g., 27C cold aisles
- Better airflow handling
  - Separate hot/cold aisles (e.g., using vinyl curtains)
  - Shorter airflow paths
- Air-side economization
  - Open the window 😊
- Water-side economization
  - Use cold water to avoid running A/C
- Waste-heat energy reclamation

Example: Airflow with Air-economization

- Open-loop system with mostly free cooling
  - Need to filter and push air around though



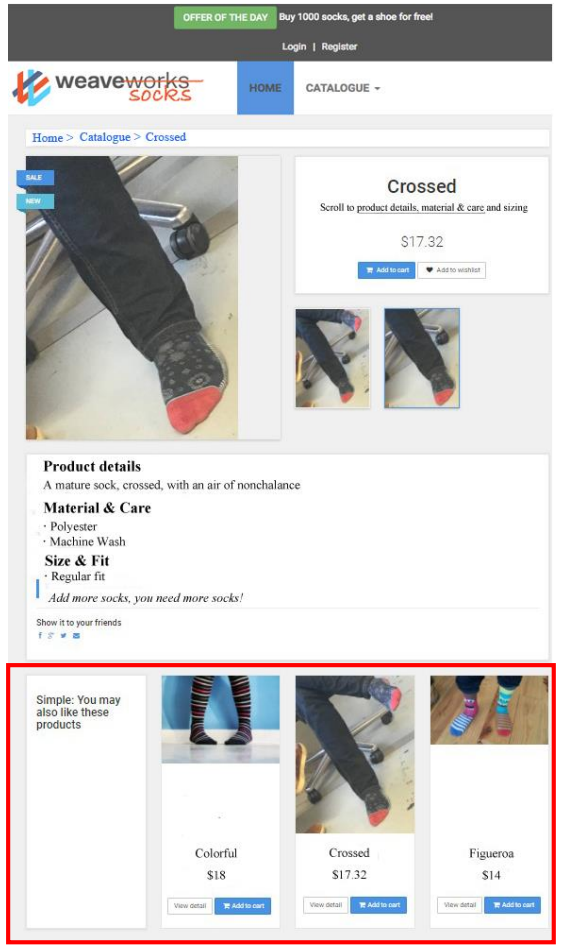
Brownout

- Brownout is a self-adaptive paradigm that enables or disables optional parts in the system concerning how to handle unpredictable workloads
  - Inspired from voltage shutdown to cope with emergency cases
  - In cloud scenario, it is used for load balance, admission control

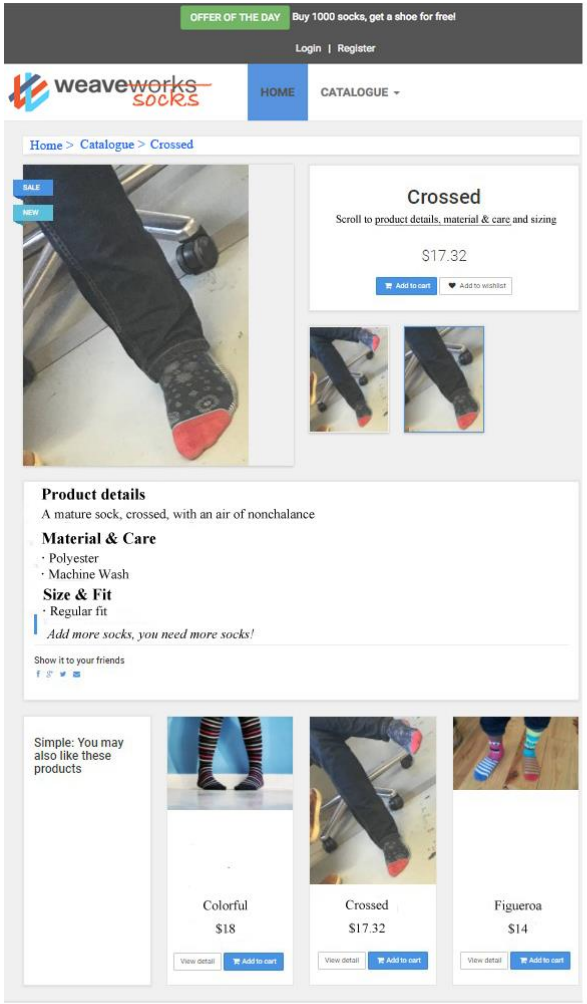
Motivational Example

- Scenario
  - Sina Weibo
- Motivation
  - Brownout can also be applied to reduce data center energy consumption
    - Apply with application components or microservices
    - Set optional components as disabled （recommendation engine）, but discount may be offered

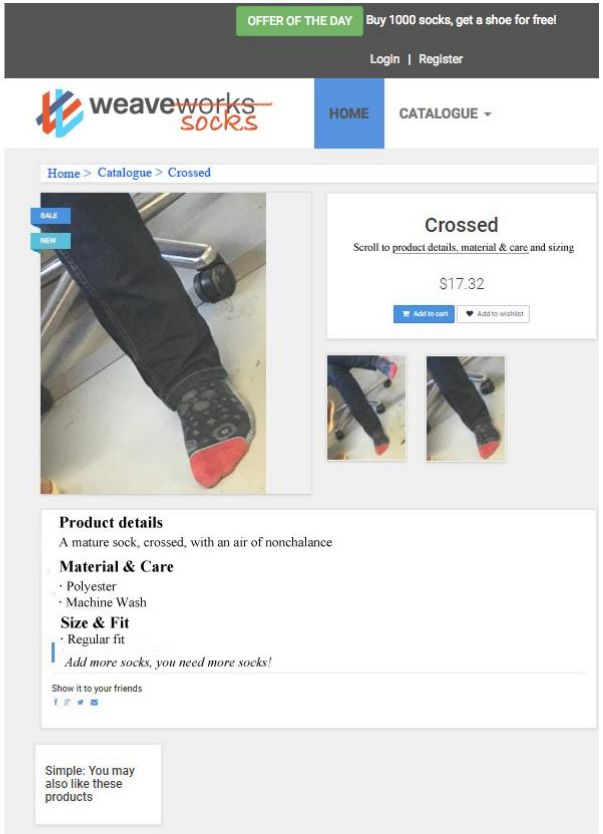
Recommendation Engine →



Brownout Impacts on Applications



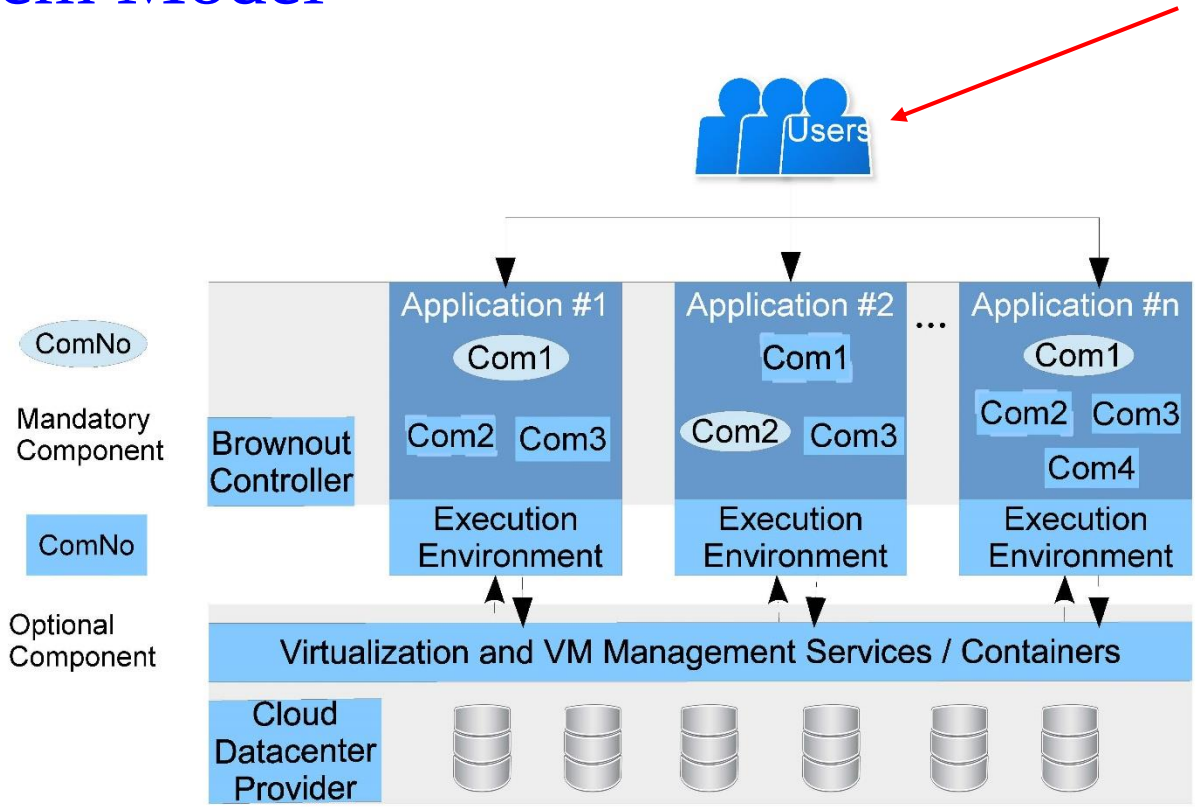
(a) Brownout is not triggered



(b) Brownout is triggered

System Model and Problem Definition

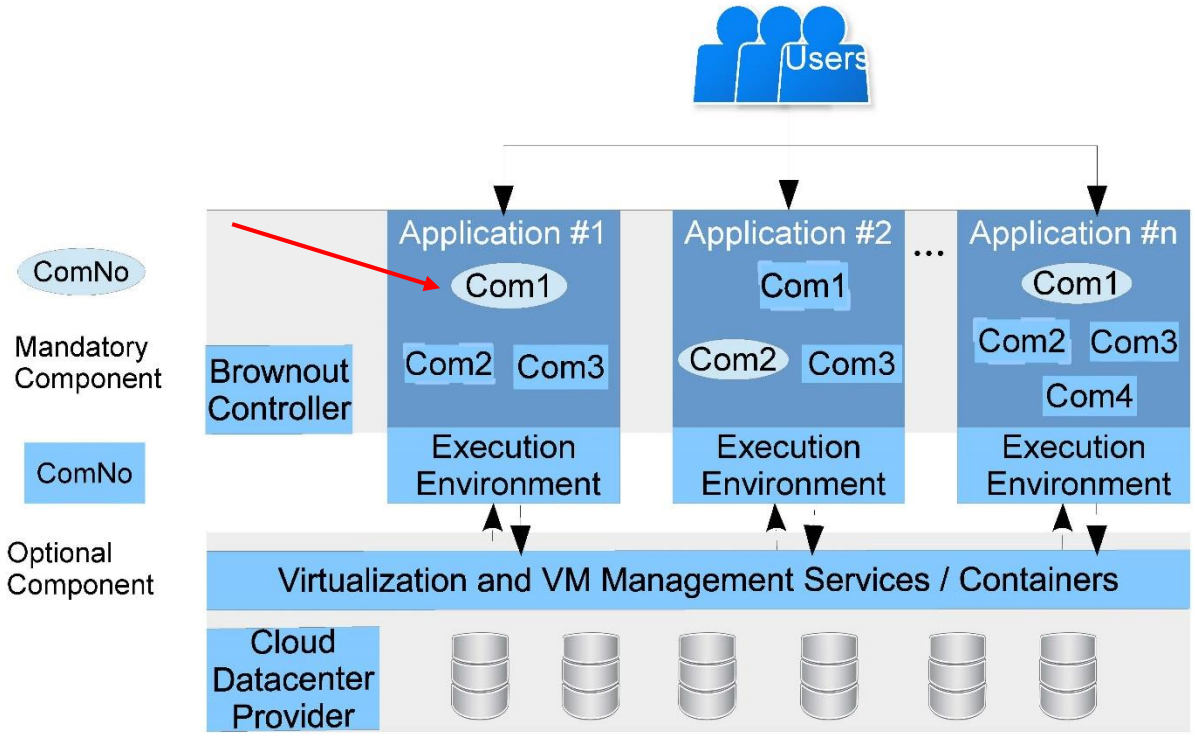
- System Model



Optional components can be deactivated according to system status

System Model and Problem Definition

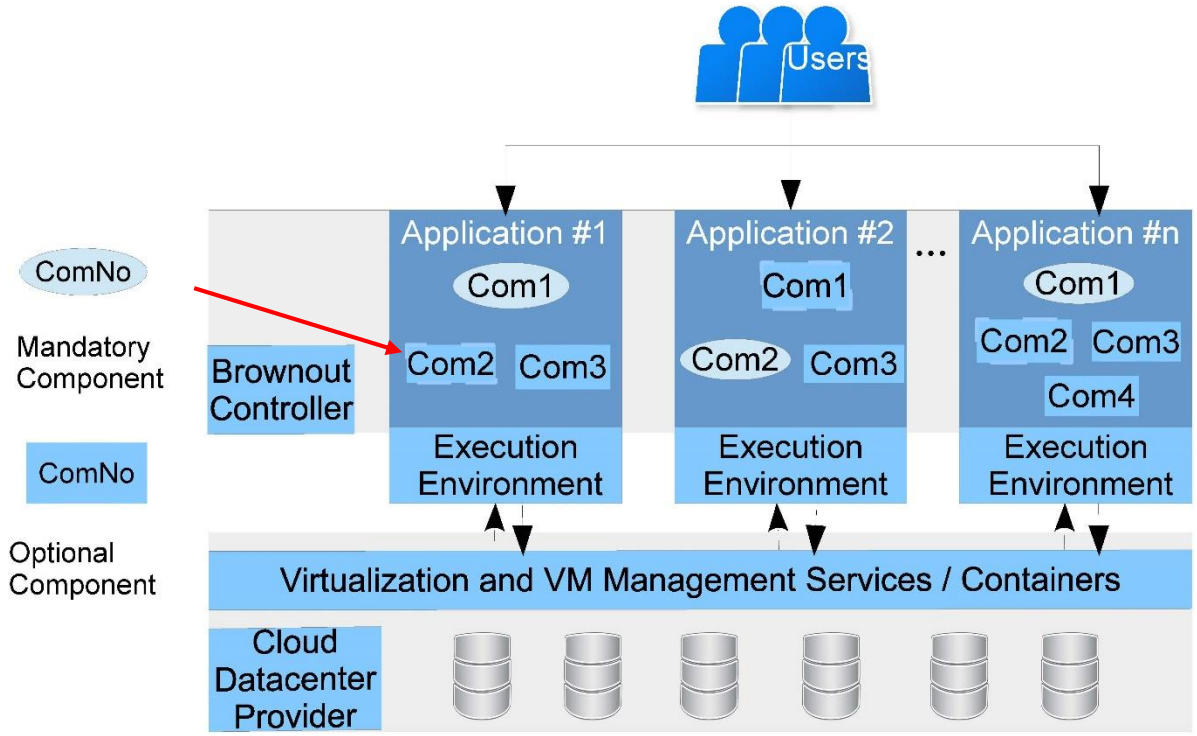
- System Model



Optional components can be deactivated according to system status

System Model and Problem Definition

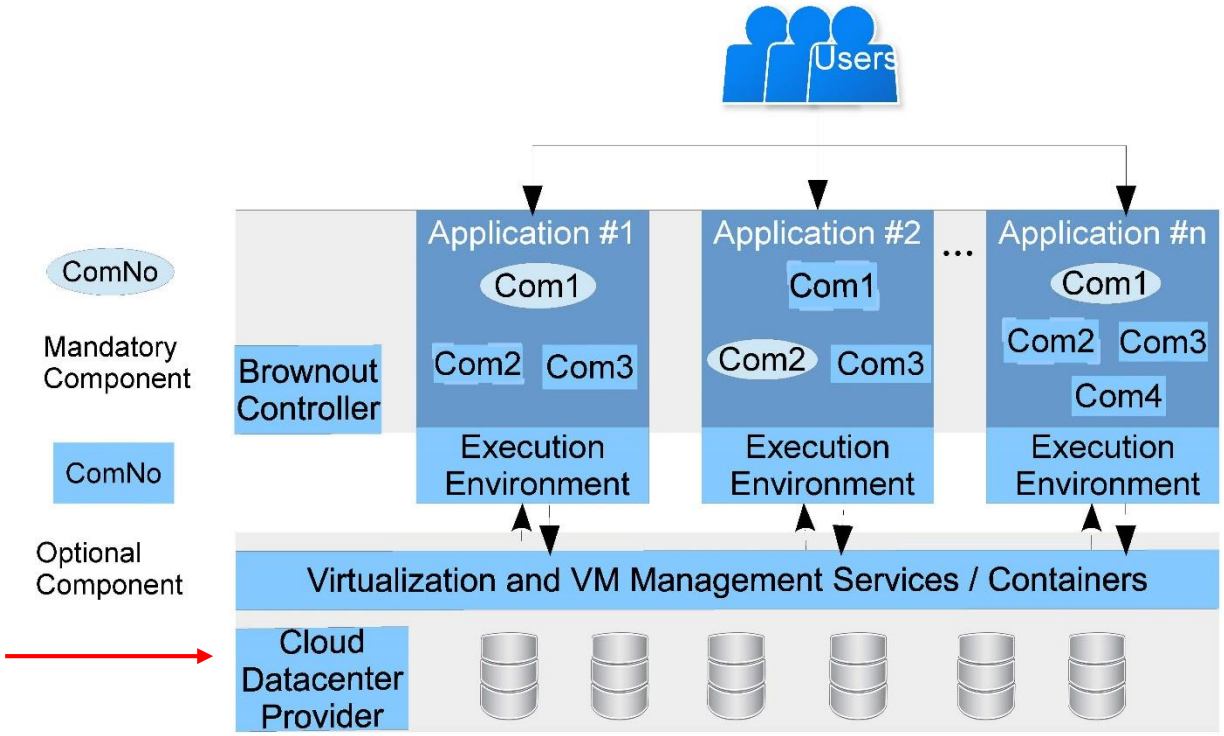
- System Model



Optional components can be deactivated according to system status

System Model and Problem Definition

- System Model

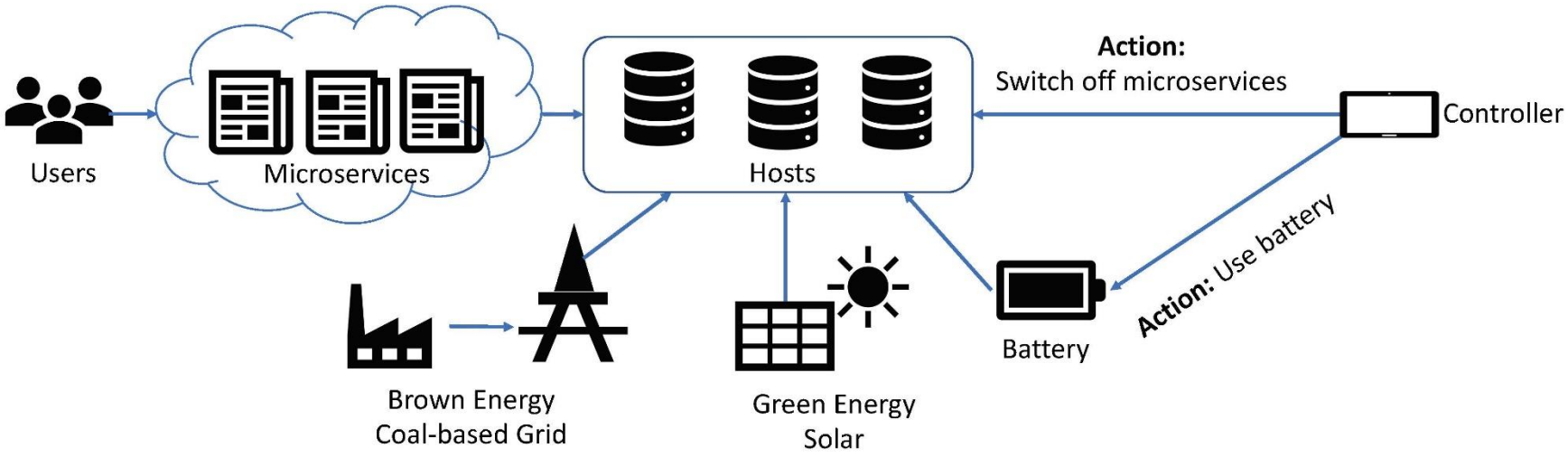


Q: Any challenges in implementing this model?

Q: Any trade-off in utilizing this model?

Schematic View (ICSOC '19)

- Schematic View of System

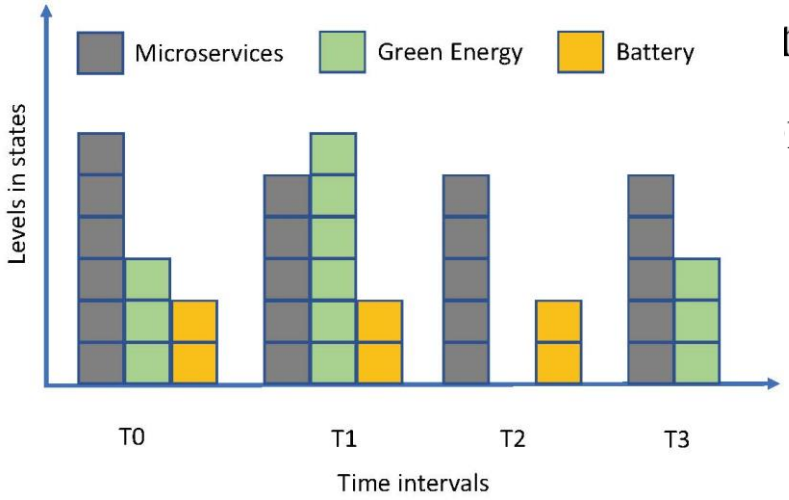


Question:

- When and in what capacity should batteries be discharged to maximize the overall executed number of microservices?

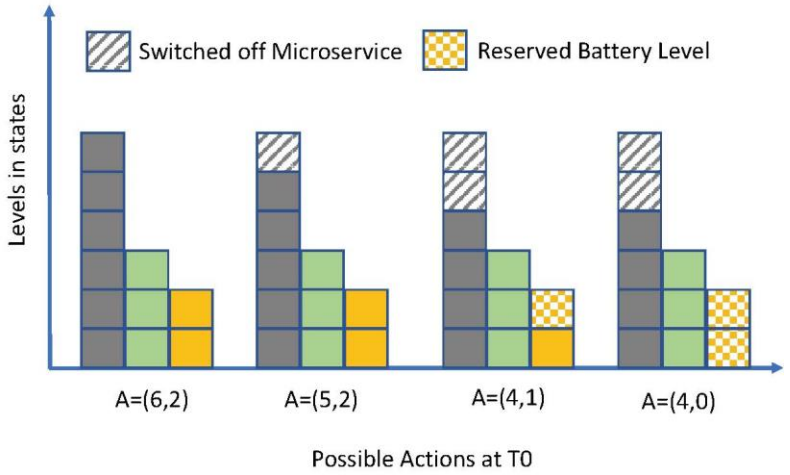
States and Actions

- Example: Possible States and Actions
  - States: $S(t) \triangleq [W(t), G(t), B(t)] \in \mathbb{S}$
    - Demanded microservices: $W(t)$
    - Available renewable energy: $G(t)$
    - Level of battery: $B(t)$
  - Actions: $A(a, b)$



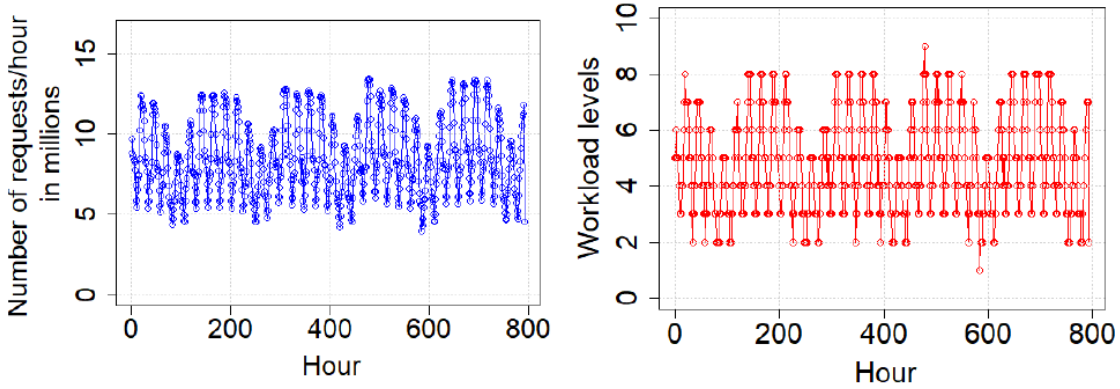
mic

nsu

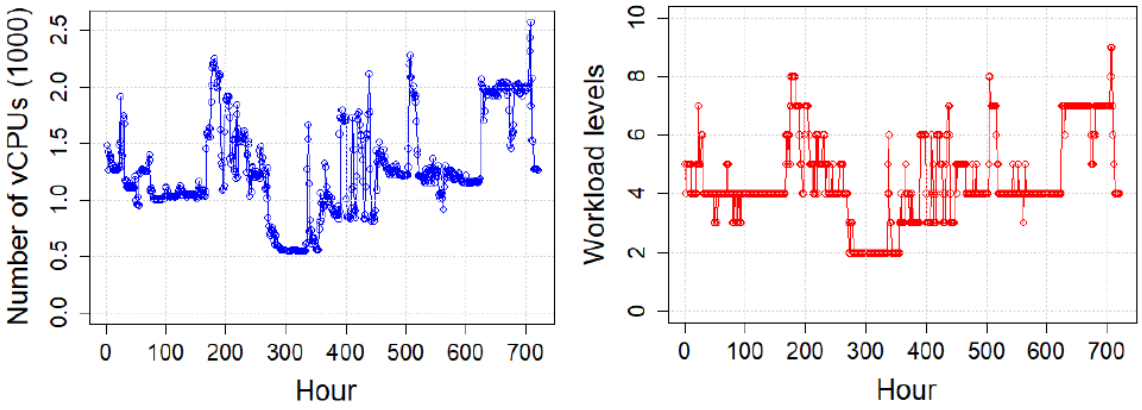


Workload Traces

- Wikipedia (original and converted workloads)



- Nectar (original and converted workloads)



Dividing the workloads into a set of levels

Solar Power Levels

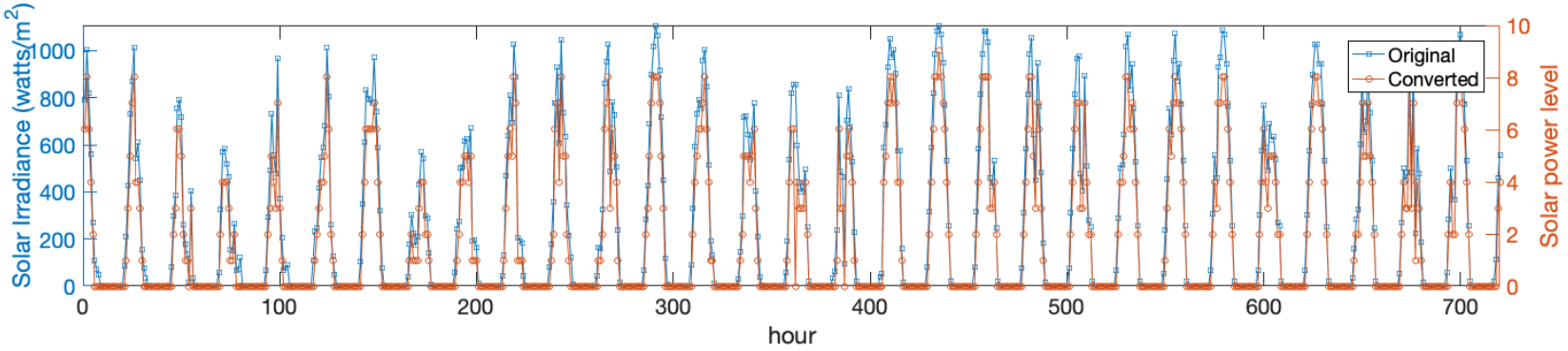
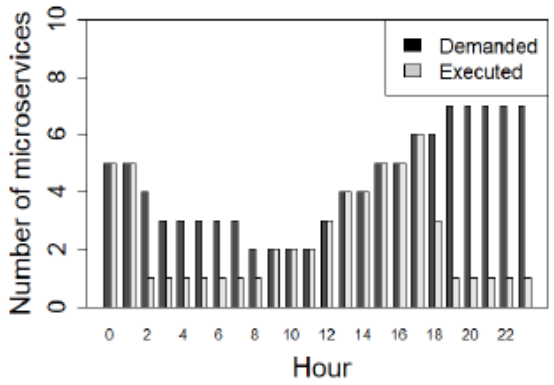


Fig: Historical and Converted solar Irradiance

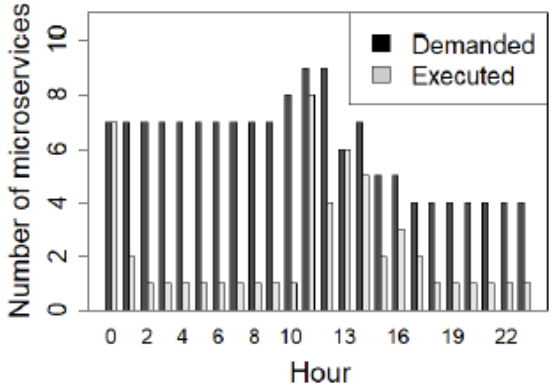
Data from Australia Government Bureau of Meteorology
Location: Clayton Campus, Monash University, Australia
Also dividing into a set of levels, e.g. 10 levels

Performance Comparison-Actions

- Actions corresponding to demanded workloads

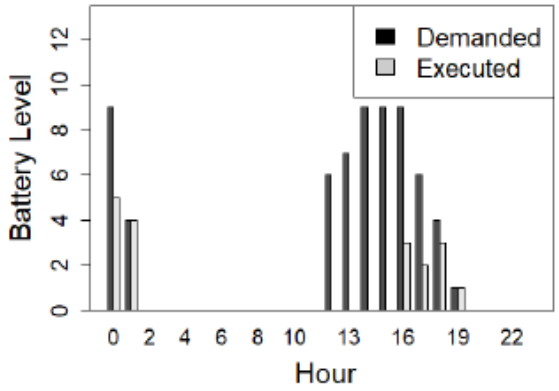


Wikipedia

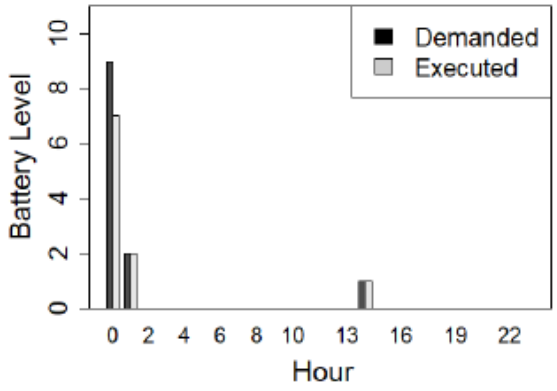


Nectar

- Battery actions corresponding to available battery

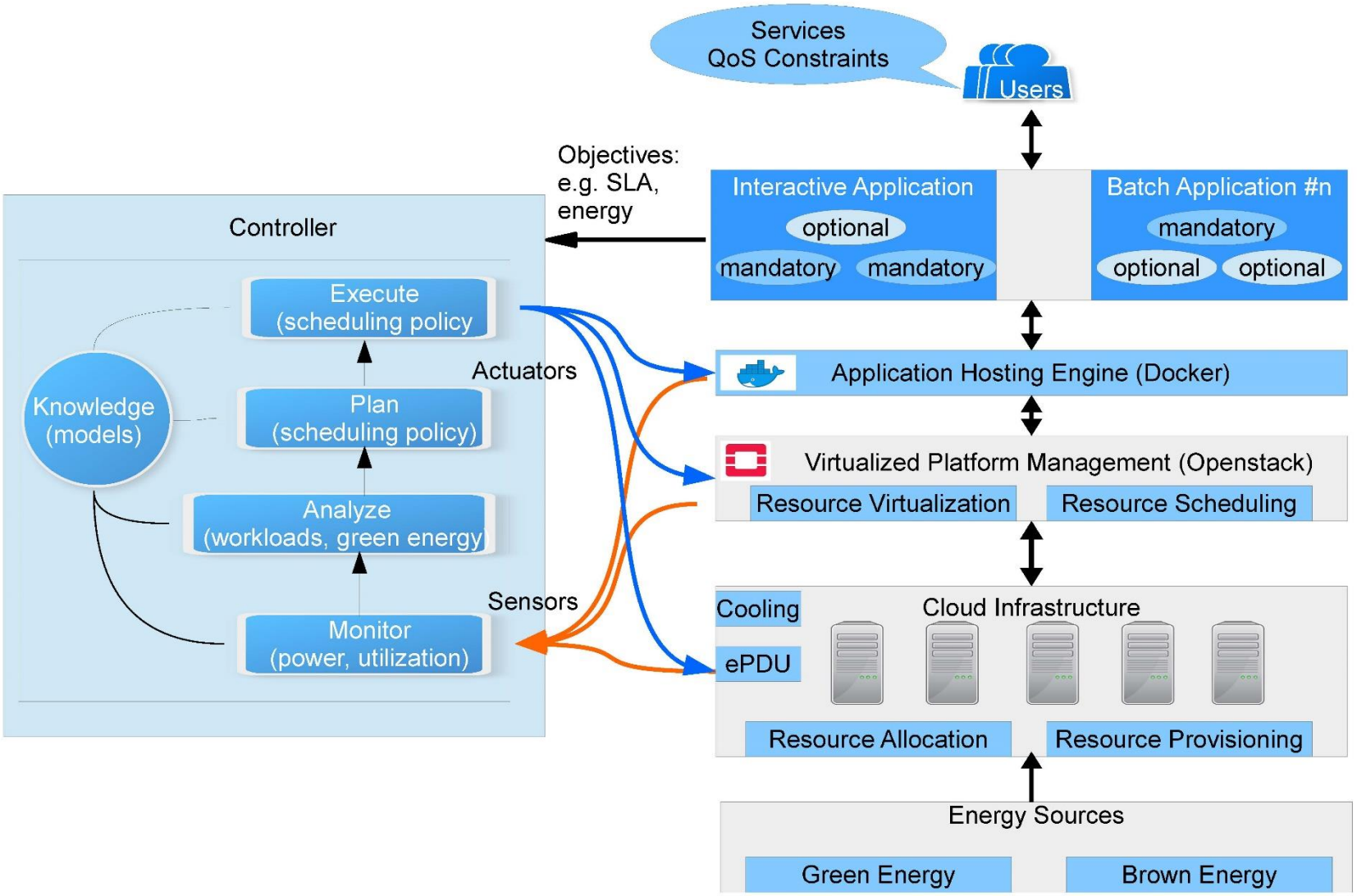


Wikipedia



Nectar

System Model (TSUSC' 2021)



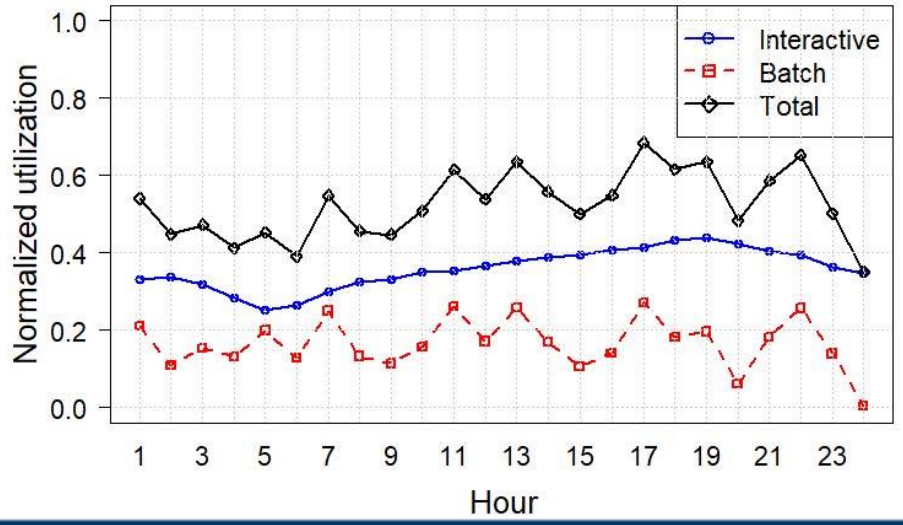
Problem Modelling

- Server Power consumption
  - Server Power: $$P_i^s = \begin{cases} P_i^{idle} + \theta_t \sum_{j=1}^{w_i} U_{i,j}^{vm} \times P_i^{dynamic} & , w_i > 0 \\ 0 & , w_i = 0 \end{cases}$$
  - VM Utilization: $$U_{i,j}^{vm} = \sum_{k=1}^{A_j} U_{j,k}^{ms}$$
- Cooling power consumption
  - Temperature related: $$CoP(T_{sup}) = 0.0068T_{sup}^2 + 0.0008T_{sup} + 0.458$$
- Workloads Model
  - Two types of workloads: $$d(t) = \sum_m a_m(t) + \sum_n b_n(t)$$
- Optimization objectives

$$min \sum_t (max(P_t - R_t, 0))$$
$$s.t. 0 \le d(t) \le D, \ \forall t$$
$$0 \le \theta_t \le 1, \ \forall t$$
$$sla \le \alpha, \forall t$$
$$avg_b(t) \le \beta, \forall t$$

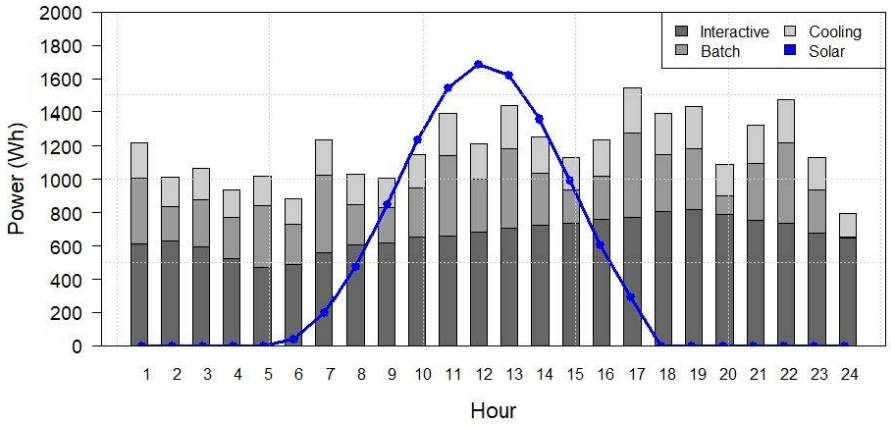
Workload Distribution

- Interactive
  - Wikipedia trace
  - It can consume up to 43% cluster utilization
- Batch
  - Facebook trace
  - It can consume up to 27% cluster utilization

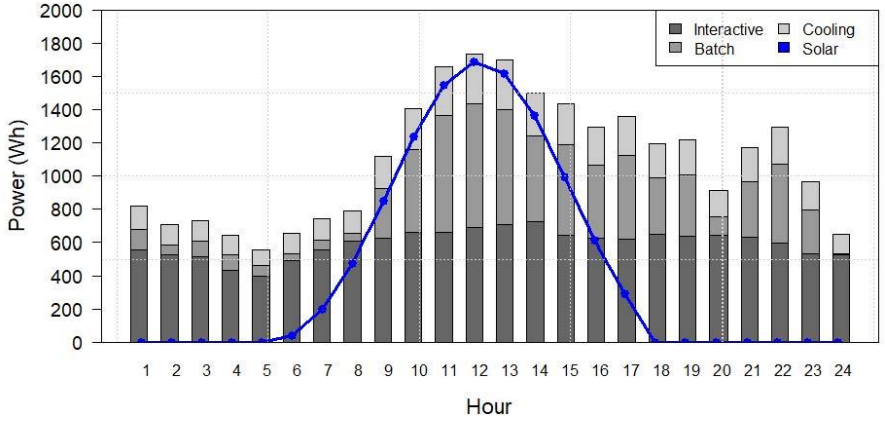


Energy Consumption Comparison

- HS
  - VM consolidation + host scaling
- GSA
  - VM consolidation + host scaling + green-aware scheduling



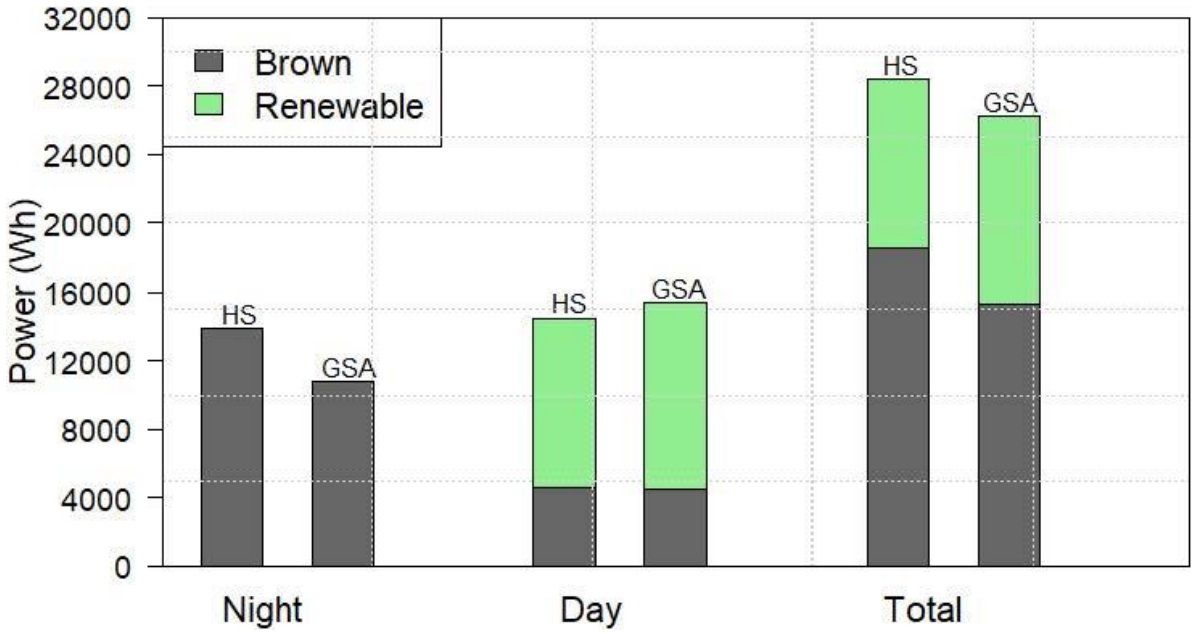
HS approach



GSA approach

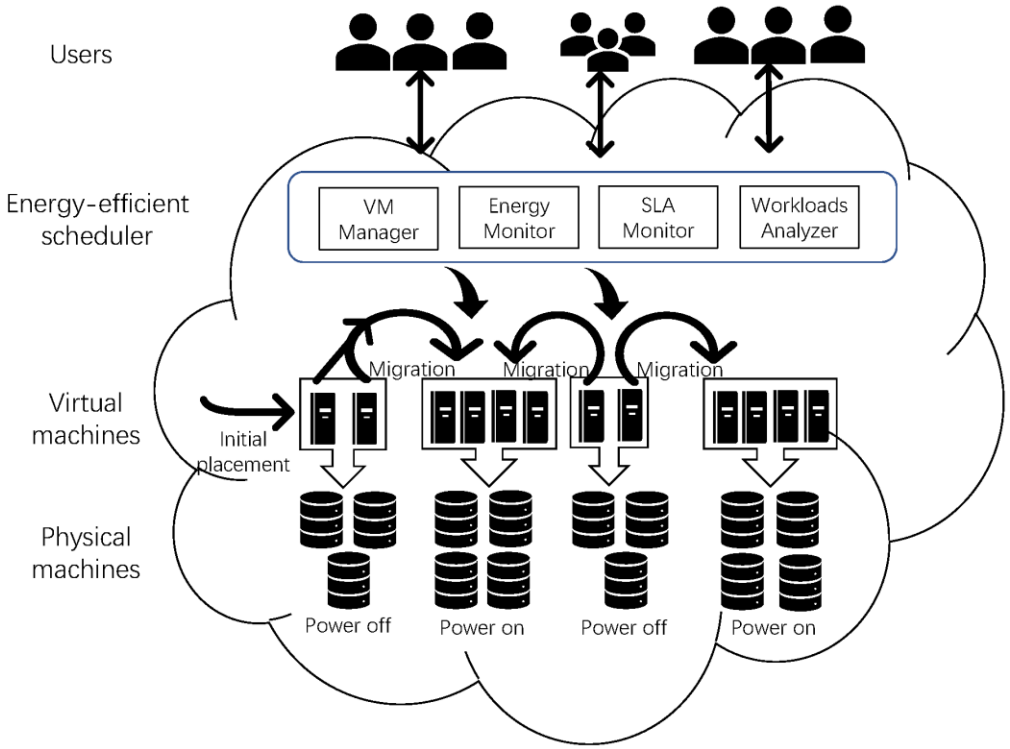
Power Usage Comparison

- Brown energy usage is reduced 17%
- Renewable energy usage is improved 13%



Comparison of Energy Efficient Algorithms

- Comparison of Energy Efficient Algorithms based on VM Consolidation (CCGIRD'2020)



Motivations

- The need and demand for understanding existing VM-based energy efficient algorithms for cloud data centers.
- The proposed algorithms were evaluated in different scenarios and configurations, thus their advantage and disadvantage are not carefully examined.
- The requirement for selecting the best suitable algorithm based on different priorities.

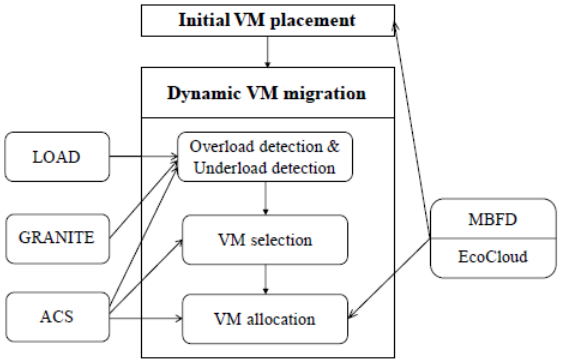
Selection Criteria

- **Persuasive:** the algorithms were published in prominent journals or conferences, and the algorithms can be representative of a category of algorithms.
- **Reproducible:** the algorithms were implemented in CloudSim or can be easily evaluated in CloudSim.
- **Comparable:** the algorithms should have been evaluated with the same baseline.

Approach	Authors	Publisher
MBFD	Beloglazov et al.	FGCS 2012
EcoCloud	Mastroianni et al.	TCC 2013
GRANITE	Li et al.	TPDS 2018
LOAD	Ranjbari et al.	JPDC 2018
ACS	Farahnakian et al.	TSC 2015

Compared Algorithms

- MBFD:
  - bin-packing problem,
  - baseline for many VM consolidation based energy efficient algorithm
- EcoCloud:
  - probabilistic approach with low complexity
  - reduce energy while minimizing VM migrations
- GRANITE:
  - representative holistic VM scheduling
- LOAD:
  - typical algorithm using leaning automata algorithm
- ACS:
  - online meta-heuristic optimization algorithm
  - based on ant colony optimization



Cross-sectional Comparison

TABLE I: High-level Comparison of Investigated Algorithms

Algorithm	Application Type	Operational Environment	Objective Function	Power Model	Power Component	Scheduling Mechanism	Workloads	Merits	Demerits
MBFD [18]	Dynamic workloads (web service)	Distributed and Heterogeneous	To optimize energy consumption	Linear	CPU and memory	Dynamic Consolidation (Proactive)	Heterogeneous workloads	Reduced energy consumption and SLA violation rate	There is a need of holistic resource management
EcoCloud [19]	Dynamic workloads	Distributed and homogeneous	To improve consolidation of VMs	Linear	Memory and CPU	Bounouli-based scheduling (Proactive)	Planetlab workloads traces	Evaluated the effect of power consumption on SLA violations	To investigate the impact of number of VM migrations on SLA violations
GRANITE [20]	Heterogeneous	Distributed	To investigate the temperature distribution of airflow and server CPU	Linear	Cooling, storage, memory, CPU and network	2D-Computational Fluid Dynamics (CFD) modelling-based scheduling (Proactive)	Google datacenter trace logs	Minimizing total datacenter energy consumption (cooling and computing)	To improve accuracy, 2-dimensional CFD model can be enhanced to 3-dimensional CFD model
LOAD [21]	Dynamic workloads	Homogeneous	To optimize energy consumption, number of VM migrations and SLA violations	Linear	CPU	Learning automata based scheduling (proactive)	CPU utilization	Improved CPU utilization and reduced SLA violations and energy consumption	Under-utilization of resource is not considered
ACS [22]	CPU and memory-intensive workloads	Distributed and Heterogeneous	To investigate the relationship between energy consumption, VM migrations and QoS	Linear	CPU and, Memory	Ant colony optimization and (reactive)	Heterogeneous workloads	Reduced energy and VM migrations	The impact of VM migration on network bandwidth can be investigated to reduce power consumption further

Experimental Settings

- Metrics
  - Total energy consumption
  - Number of active servers
  - SLA violation percentage
  - VM migration time
  - Average SLA violation
- Workloads
  - Synthetic workloads
  - PlanetLab
- Other configurations are set as the ones predefined in paper

TABLE IV: Host / VM Types and Capacity

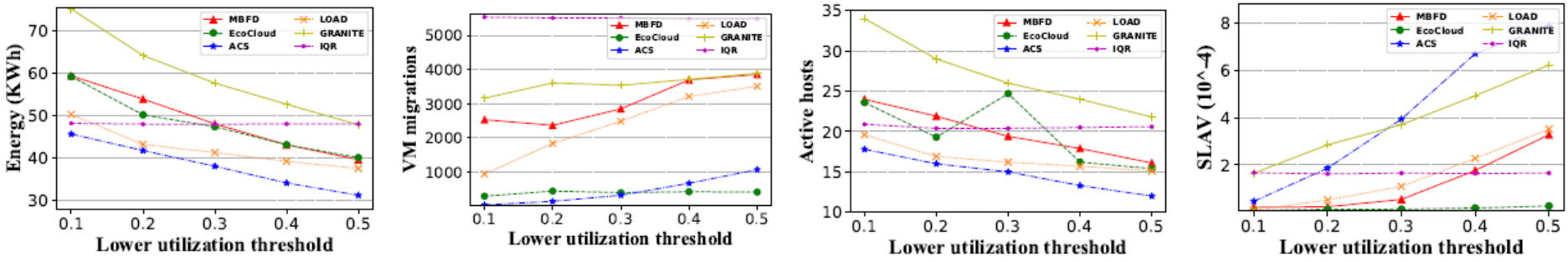
Name	CPU (MIPS)	Cores	Memory	Bandwidth	Storage
Host Type 1	1.86 GHz	2	4 GB	1 Gbit/s	1 TB
Host Type 2	2.66 GHz	2	4 GB	1 Gbit/s	1 TB
VM Type 1	2.5 GHz	1	870 MB	100 Mbit/s	1 GB
VM Type 2	2.0 GHz	1	1740 MB	100 Mbit/s	1 GB
VM Type 3	1.0 GHz	1	1740 MB	100 Mbit/s	1 GB
VM Type 4	0.5 GHz	1	613 MB	100 Mbit/s	1 GB

TABLE III: Energy consumption with different CPU utilization in Watts

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

Results with Synthetic workloads

- The lower utilization threshold is 0.1~0.5 with increment as 0.1, the higher utilization threshold is 0.4 higher than the lower threshold
- ACS performs the best on energy consumption by reducing 21.1% power compared with MBFD
- EcoCloud requires less than 600 migration times, which are much fewer than other algorithms
- For SLA violation comparison, these algorithms perform better when the lower threshold is set to be 0.1



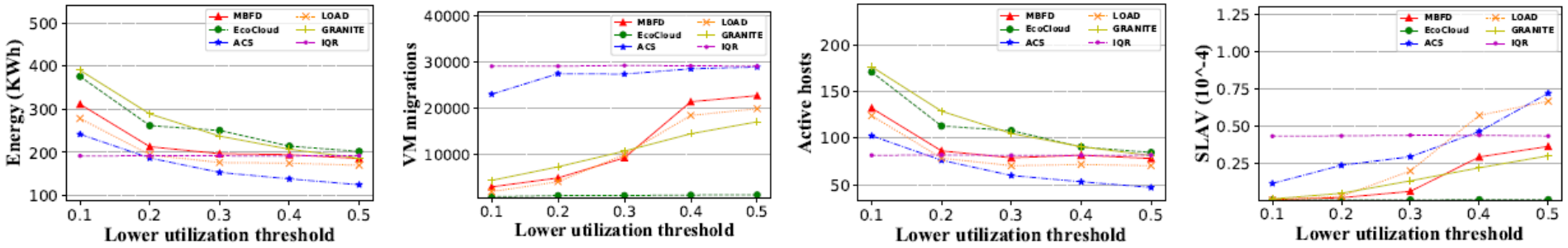
(a) Energy consumption (b) VM migrations (c) Number of active hosts (d) SLAV

Fig. 3: Performance comparison of algorithms under Synthetic workloads (ratio of hosts and VMs number is 50:50)

79

Results with PlanetLab workloads

- The lower CPU utilization threshold varies from **0.1 to 0.5**
- the interval between the lower threshold and the higher threshold is fixed at 0.4
- The number of host is set as **800**
- LOAD achieves improvement in the number of migrations and reduces **12.4%** migrations compared with MBFD.
- LOAD and ACS perform worse on SLA violations compared with MBFD and EcoCloud.



(a) Energy consumption (b) VM migrations (c) Number of active hosts (d) SLAV

Fig. 6: Performance comparison of algorithms under PlanetLab workloads