# Power Management of Online Data Intensive Services

Meisner et al

# Characteristics of OLDI

- Large search, online advertising, machine learning
- User queries that must interact with lots of data
- Responsiveness is important (as opposed to Map-Reduce)
- Sub-seconds responsiveness
- Diurnal variations do not lead to energy-proportionality
- Systems rarely completely idle
- Power management is challenging with the latency sensitivity and scale
- Energy proportionality means energy proportional to load

# Objectives

- OLDI energy proportionality with Proc, Mem and Disk
- Production web-search workload at cluster-wide scale
- Fine-grain characterization
- Identify Power-saving opportunities
- Identify challenges in power management
- Develop and validate a perf model that evaluates impact of proc and mem low power modes
- Production google server – 1000 server cluster level
- Idle low power modes versus active low power modes

# Power Down Modes - Example

- **Modes control clock frequency, $V_{DD}$, or both**
  - Active mode: maximum power consumption
    - Full clock frequency at max $V_{DD}$
  - Doze mode: ~10X power reduction from active mode
    - Core clock stopped
  - Nap mode: ~ 50% power reduction from doze mode
    - $V_{DD}$ reduced, PLL & bus snooping stopped
  - Sleep mode: ~10X power reduction from nap mode
    - All clocks stopped, core $V_{DD}$ shut-off

- **Issues and Tradeoffs**
  - Determining appropriate modes and appropriate controls
  - Trading-off power reduction to wake-up time

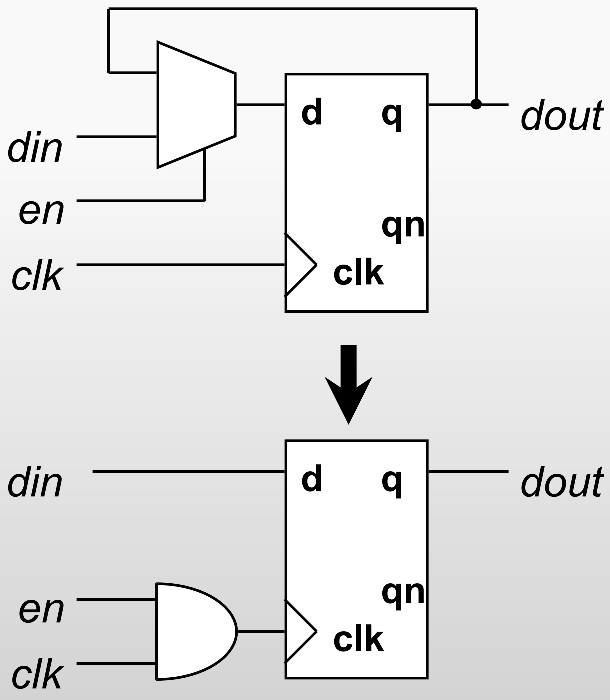
[Ref: S. Gary, D&T' 94]

# Clock Gating, Data Gating

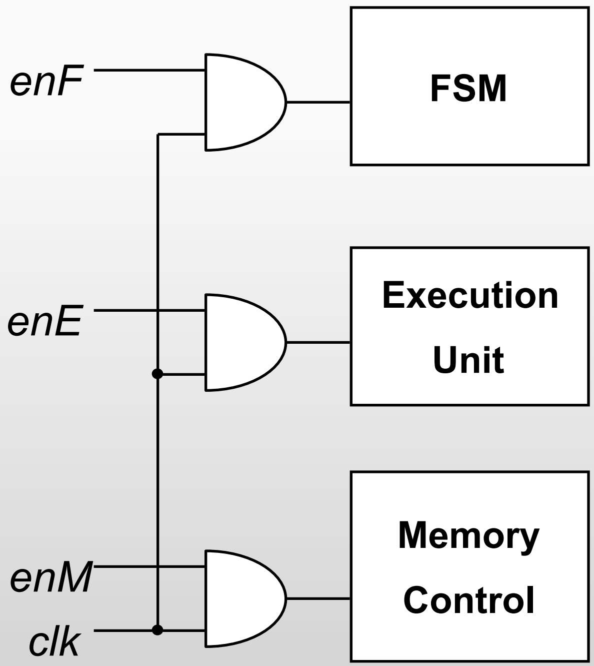
- **Primary objective: minimize $f_{eff}$**
- **Clock gating**
  - Reduces / inhibits unnecessary clocking
    - Registers need not be clocked if data input hasn’t changed
- **Data gating**
  - Prevents nets from toggling when results won’t be used
    - Reduces wasted operations

# Clock Gating

- Power is reduced by two mechanisms
  - Clock net toggles less frequently, reducing $f_{eff}$
  - Registers' internal clock buffering switches less often



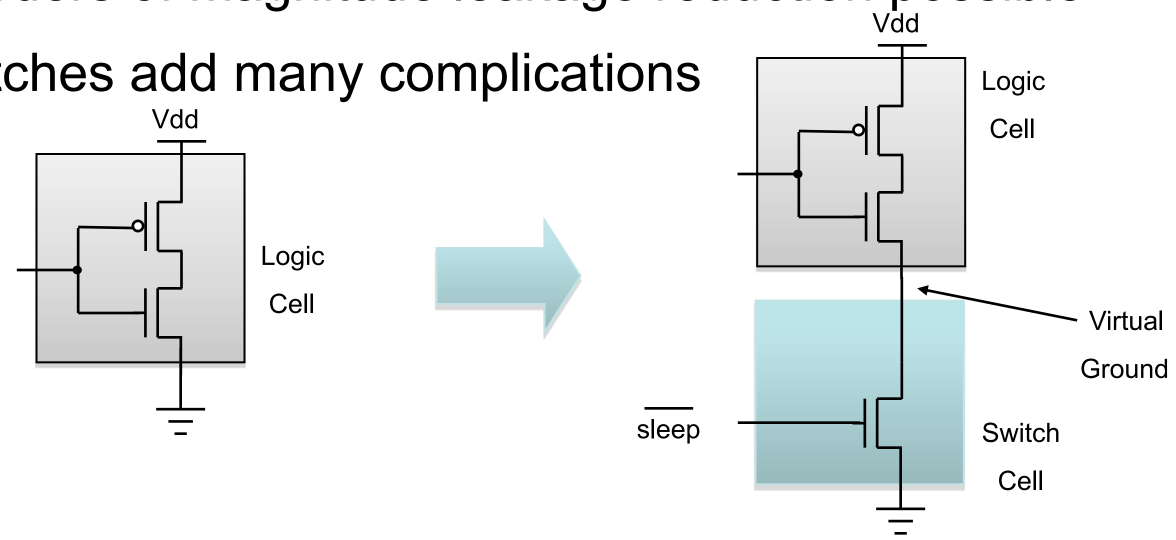

Local Gating




Global Gating

# Power Gating (also called Core Parking)

- **Objective**
  - Reduce leakage currents by inserting a switch transistor (usually high $V_{TH}$) into the logic stack (usually low $V_{TH}$)
    - Switch transistors change the bias points ($V_{SB}$) of the logic transistors
- **Most effective for systems with standby operational modes**
  - 1 to 3 orders of magnitude leakage reduction possible
  - But switches add many complications



## Table 3.8 Example C-states Definition

| C-State | Response Latency(us) |
|---|---|
| $C_0$ | 0 |
| $C_1$ | 10 |
| $C_2$ | 100 |
| $C_3$ | 1000 |
| $C_4$ | 10000 |

ACPI = Advanced Config and Power Interface

- Idle low power states = C-states
- C6 = power gating
- Active low power staes = P-states

## Table 3.7 Example P-states Definition

| P-State | Frequency (MHz) | VDD (Volts) |
|---|---|---|
| $P_0$ | $F_{Max} \cdot 100\%$ | $V_{Max} \cdot 100\%$ |
| $P_1$ | $F_{Max} \cdot 85\%$ | $V_{Max} \cdot 96\%$ |
| $P_2$ | $F_{Max} \cdot 75\%$ | $V_{Max} \cdot 90\%$ |
| $P_3$ | $F_{Max} \cdot 65\%$ | $V_{Max} \cdot 85\%$ |
| $P_4$ | $F_{Max} \cdot 50\%$ | $V_{Max} \cdot 80\%$ |

# Linux DVFS governors (On-demand, ladder etc)

**Linux On-demand frequency Governor pseudocode**

**(Util = Time_active /(Time_active + Time_idle)**

```
#define up_threshold 0.90
for(each sampling interval){
if(utilization > up_threshold)
freq = max_freq;
else
freq = next_lower_freq;
}
```

# Past observations

- Lightly loaded servers –
- good energy proportionality by idle low power modes
- Technique works well if average utilization low

- Energy proportionality at cluster level possible by
- VM migration and selective power-down of servers

- Servers and OLDI services are different
- OLDI rarely completely idle

# Core-Level Activity Prediction, Bircher&John2011

- Vista reactive p-state algorithm often over or under provisions core frequency.
- Our predictive p-state selection algorithm reduces core power consumption by 5.4% and increases performance by 3.8%
- SYSMARK

| | E-Learning | | Productivity | | Video Creation | | 3D | |
|---|---|---|---|---|---|---|---|---|
| | Predictive (PPPP) | Reactive (Vista) | Predictive (PPPP) | Reactive (Vista) | Predictive (PPPP) | Reactive (Vista) | Predictive (PPPP) | Reactive (Vista) |
| Active Frequency (GHz) | 1.72 | 1.56 | 1.47 | 1.34 | 1.65 | 1.51 | 1.92 | 1.77 |
| Idle Frequency (GHz) | 1.01 | 1.24 | 0.94 | 1.07 | 1.01 | 1.20 | 1.07 | 1.32 |

Lloyd Bircher and Lizy K. John, Core-Level Activity Prediction for Multi-Core Power Management, **IEEE Journal** on Emerging and Selected Topics in Circuits and Systems (JETCAS), September 2011, pp. 218-227.

# Conclusions

- CPU active low-power modes good but not sufficient
- for energy-proportionality
- CPU idle low-power modes good at core level, but not enough
- for shared caches and on-chip memory controllers
- Ample underutilization in memory and so opportunity in memory
- with active low-power modes
- Power-Nap (useful for data-center workloads) ineffective for OLD
- Energy-proportionality for OLDI with acceptable query latency
- only with full-system active low-power modes

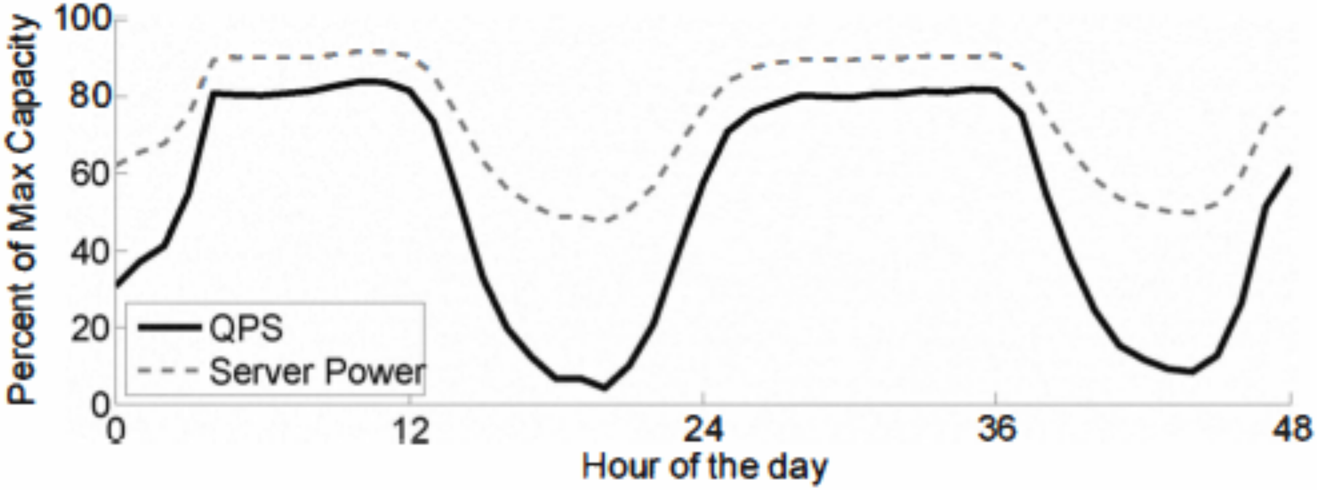



**Figure 1: Example diurnal pattern in queries per second (QPS) for a Web Search cluster:** Non-peak periods provide significant opportunity for energy-proportional servers. For a perfectly energy proportional server, the percentage of peak power consumed and peak QPS would be the same. Server power is estimated for systems with 45% idle power.

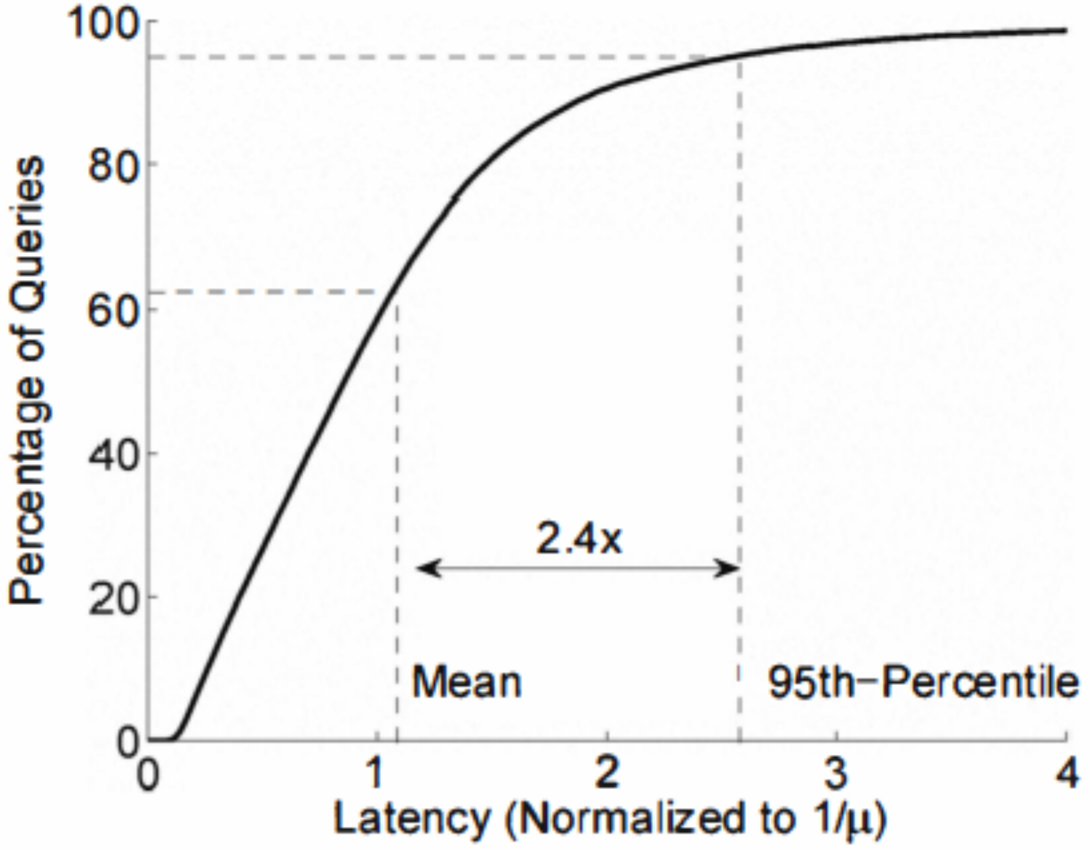



**Figure 2: Example leaf node query latency distribution at 65% of peak QPS.**

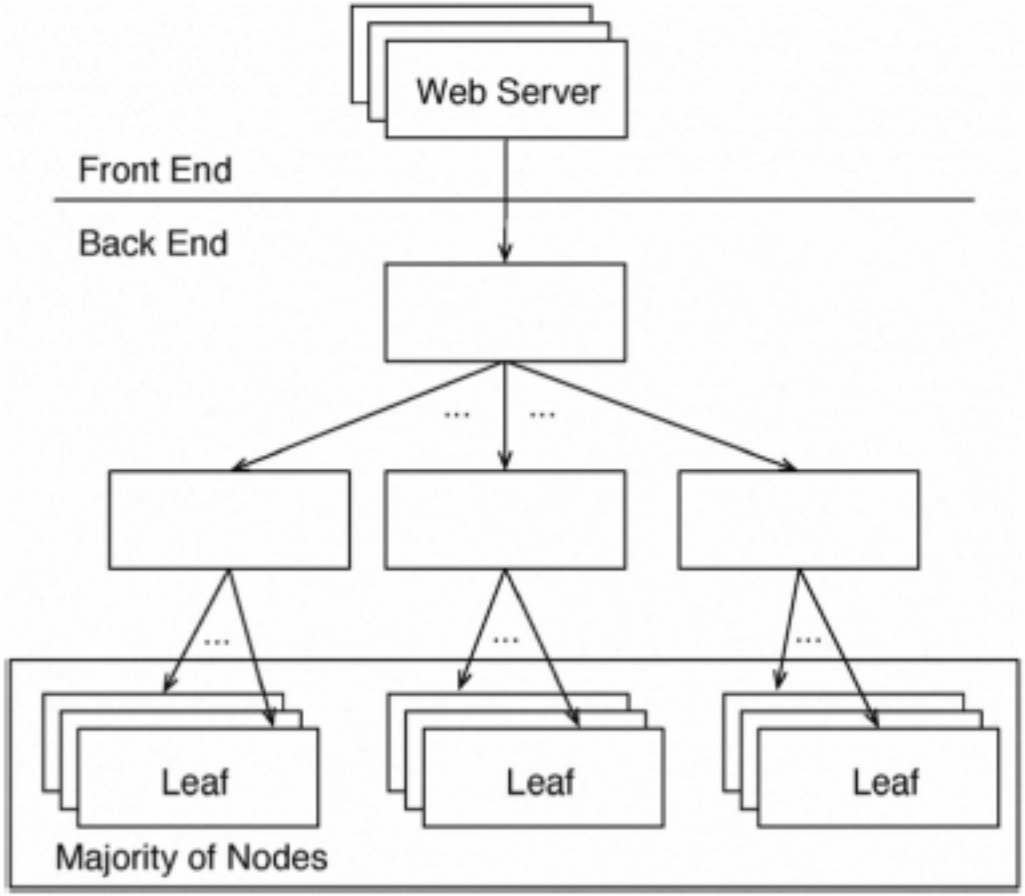



**Figure 3: Web Search cluster topology.**

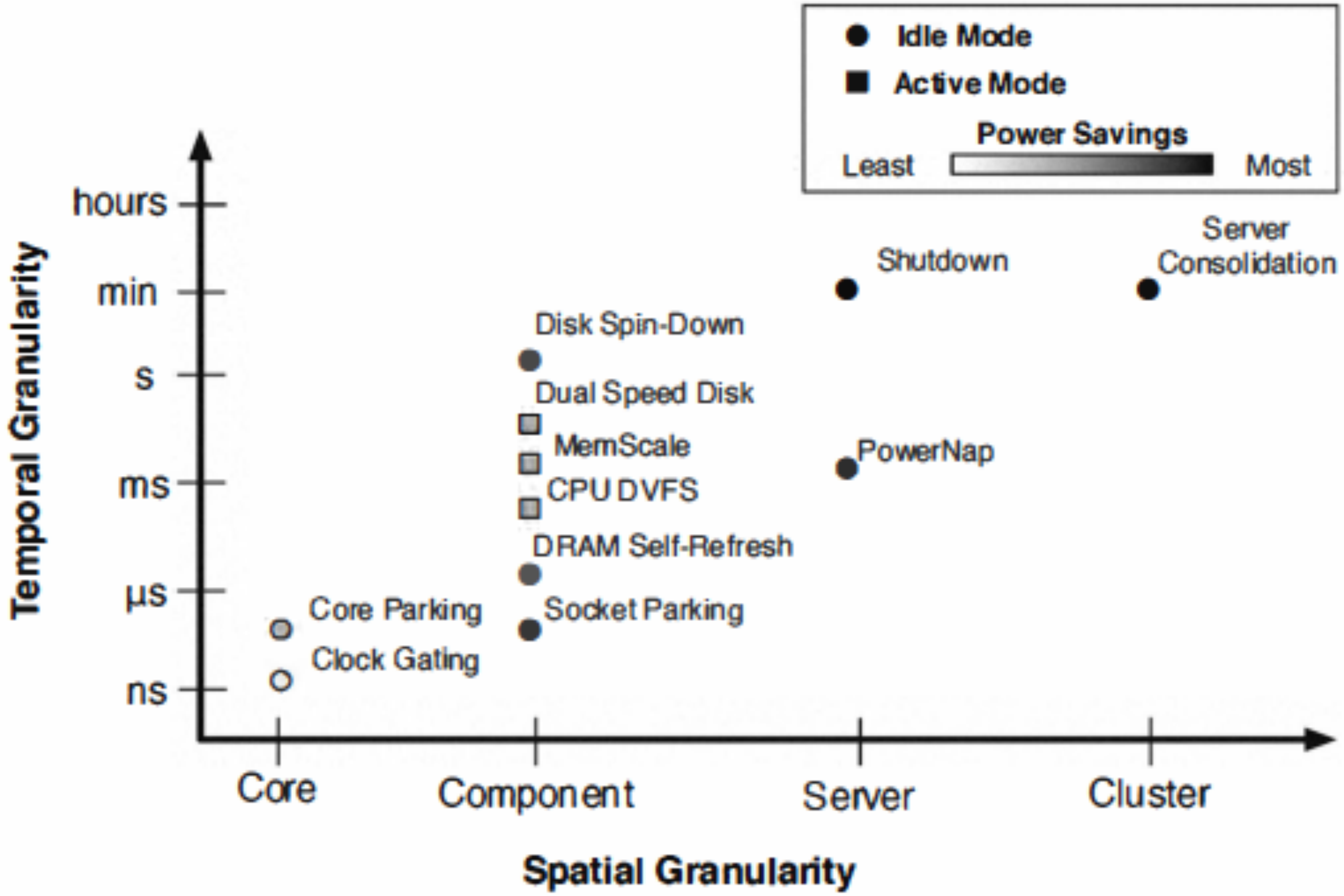



**Figure 4: Low-power mode taxonomy.** Modes with the greatest power savings must be applied at coarse granularity. Modes that apply at fine granularity generally yield less savings.

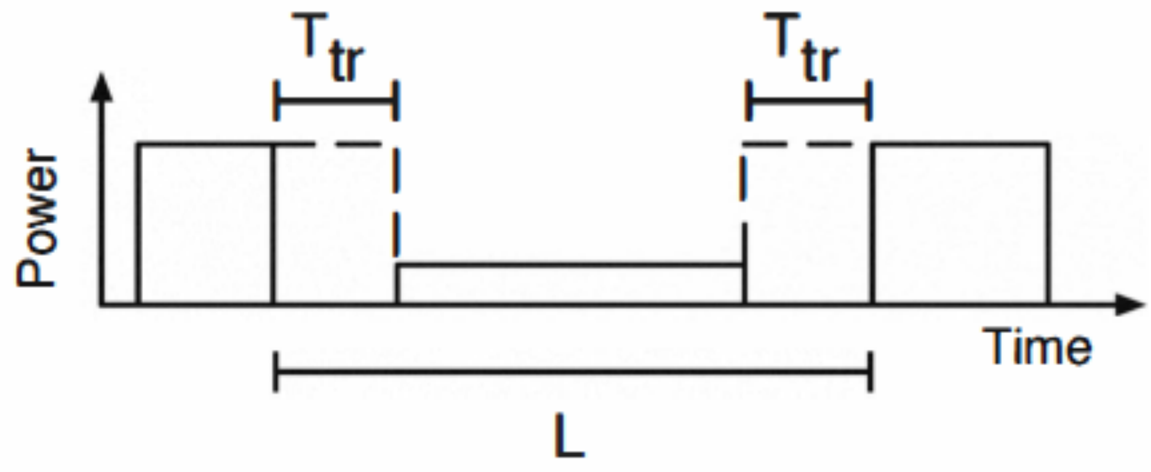
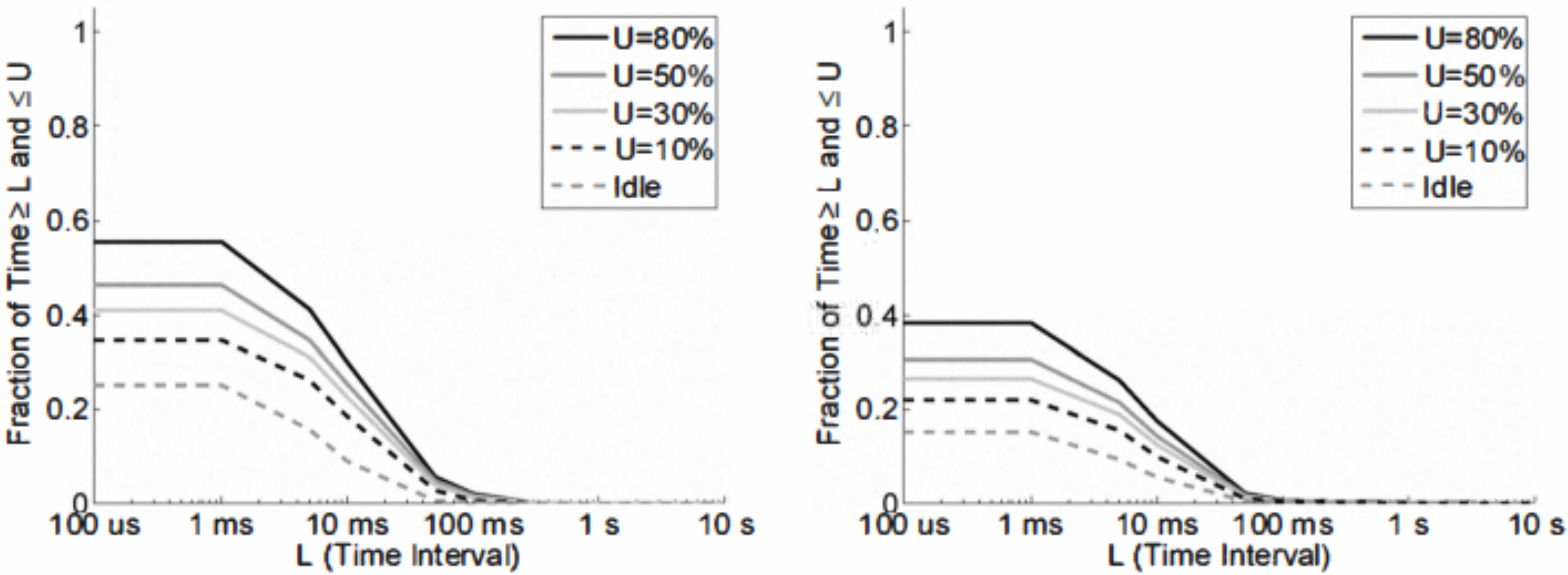



**Figure 5: Idealized low-power mode.** $L$ is the length of the idle period and $T_{tr}$ is the time required to transition in and out of the low-power state.

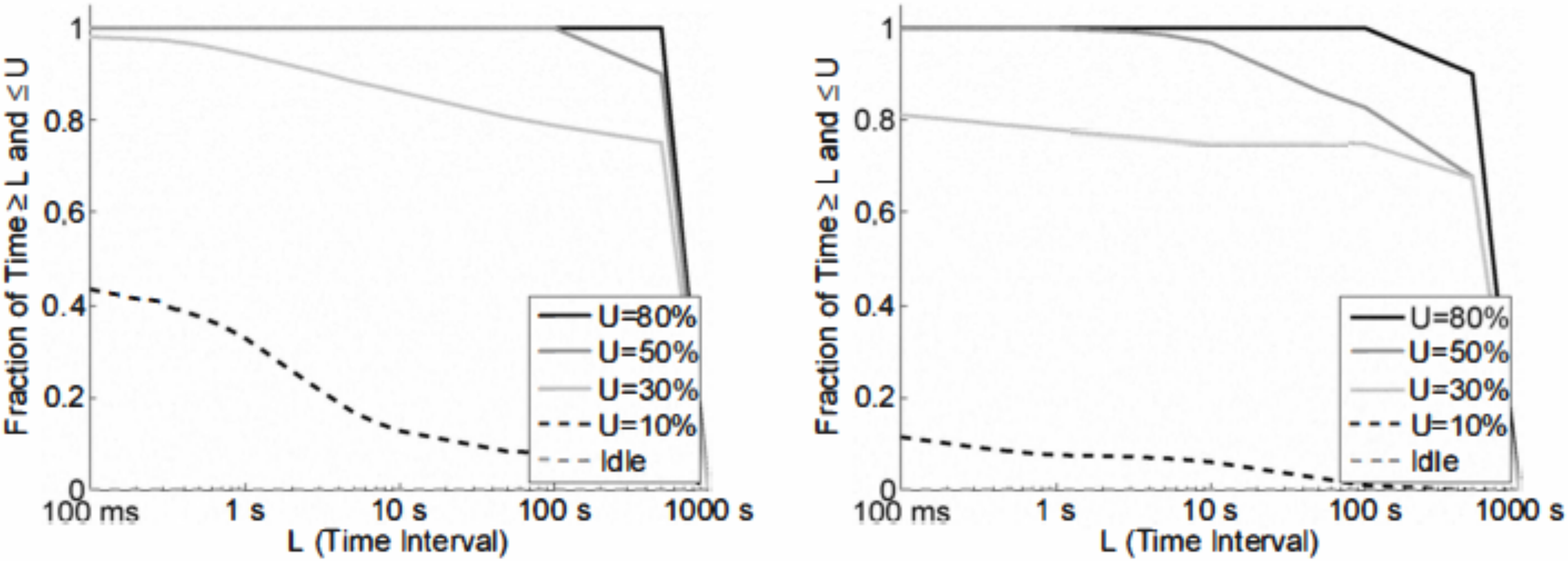
# Activity Graph – fraction of time a component spends at or below Given Utilization (U) for a time L or greater




(b) 50% QPS

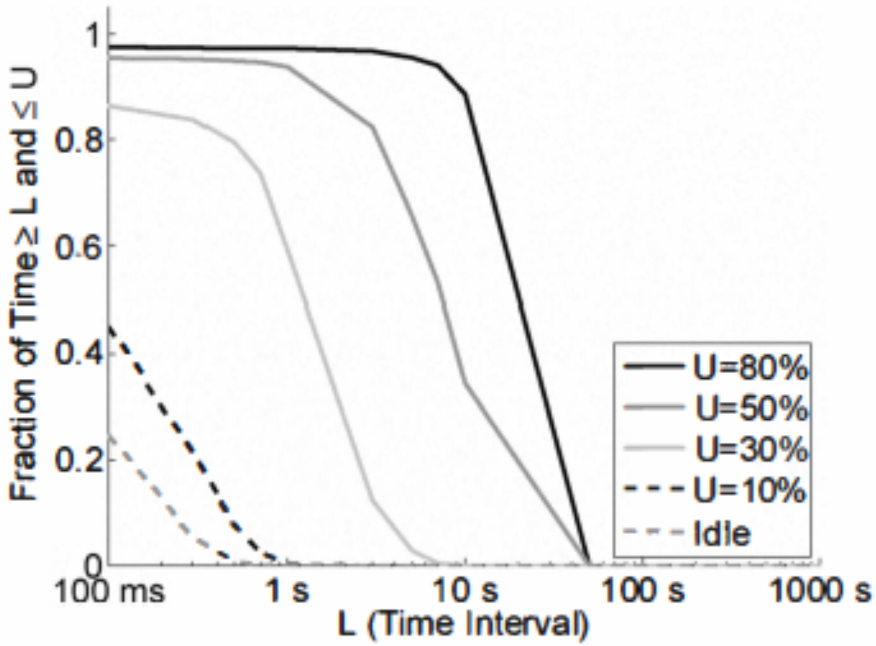
(c) 75% QPS

hs. Opportunities for CPU power savings exist only below 100 ms regardless of load.




(b) 50% QPS (c) 75% QPS

**graphs.** Memory bandwidth is undersubscribed, but the sub-system is never idle.




(b) 50% QPS (c) 75% QPS

hs. Periods of up to tens of seconds with moderate utilization are common for disks.

Table 1: Low-power mode characteristics.

| Power Mode | $T_{tr}$ | $u_{\text{threshold}}$ | $\frac{\Delta P_{\text{Mode}}}{P_{\text{Nominal}}}$ | Ref. |
|---|---|---|---|---|
| C1E → ACPI C3 | 10 $\mu$s | Idle | 2% | [2] |
| C1E → ACPI C6 | 100 $\mu$s | Idle | 44% | [2] |
| Ideal CPU $V_{dd}$ Scaling | 10 $\mu$s | 50% | 88% | [2] |
| Ideal Mem. $V_{dd}$ Scaling | 10 $\mu$s | 50% | 88% | [17] |
| Dual-Speed Disk | 1 sec | 50% | 59% | [22] |
| Disk Spin-Down | 10 sec | Idle | 77% | [7] |

# Leaf Node Performance Model

- L_query = L_service + L_wait
- L_service
- Modeling L_wait
- G/G/k queue
- Arbitrary inter-arrival and service time distributions
- Average throughput $\lambda$
- Average service rate $\mu$
- K servers
- Average load $\rho = \frac{\lambda}{\mu \cdot k}$

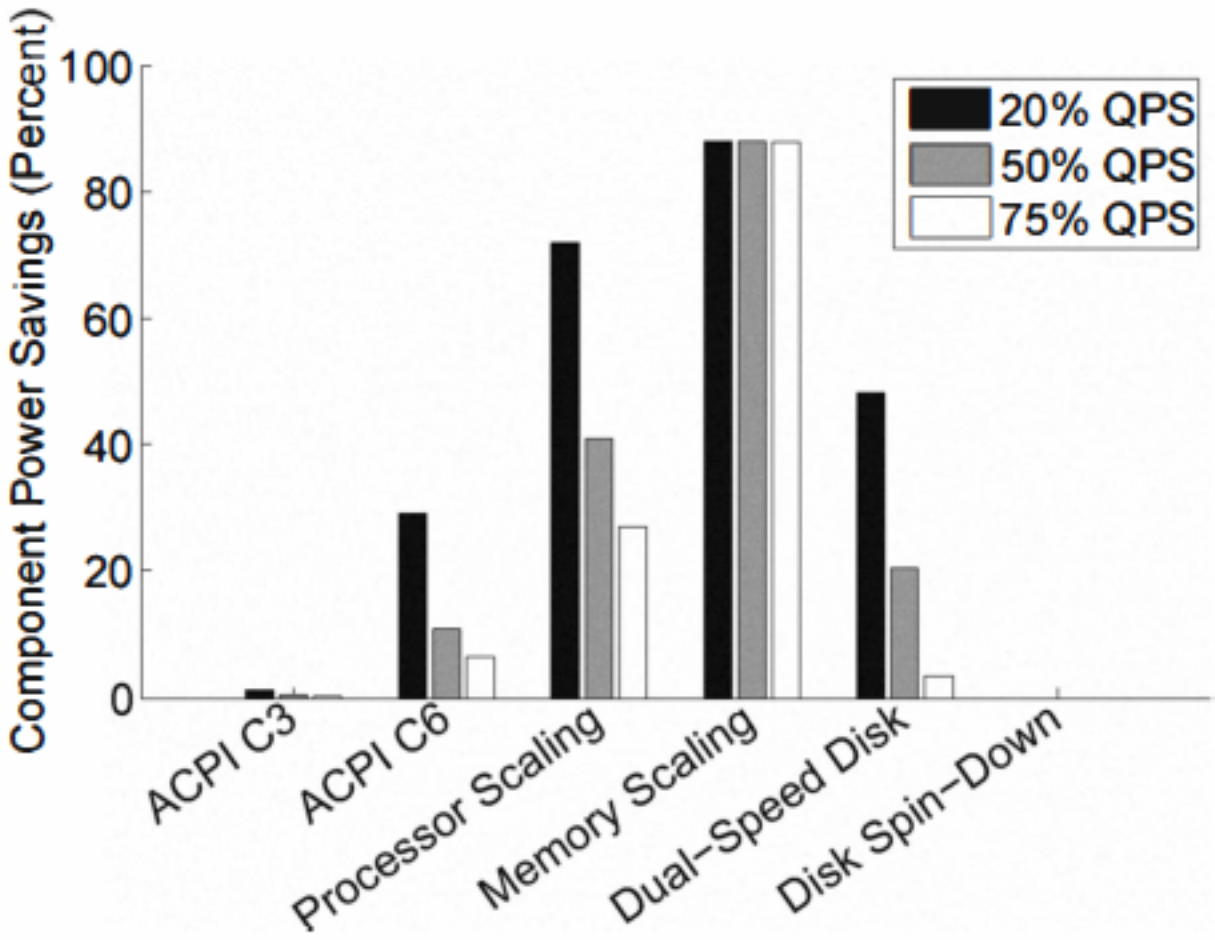
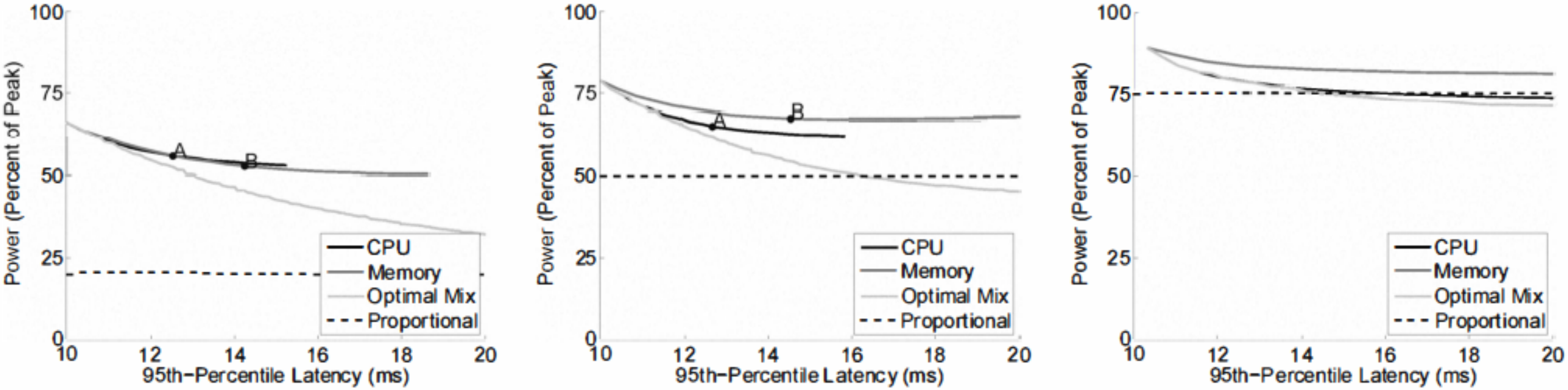
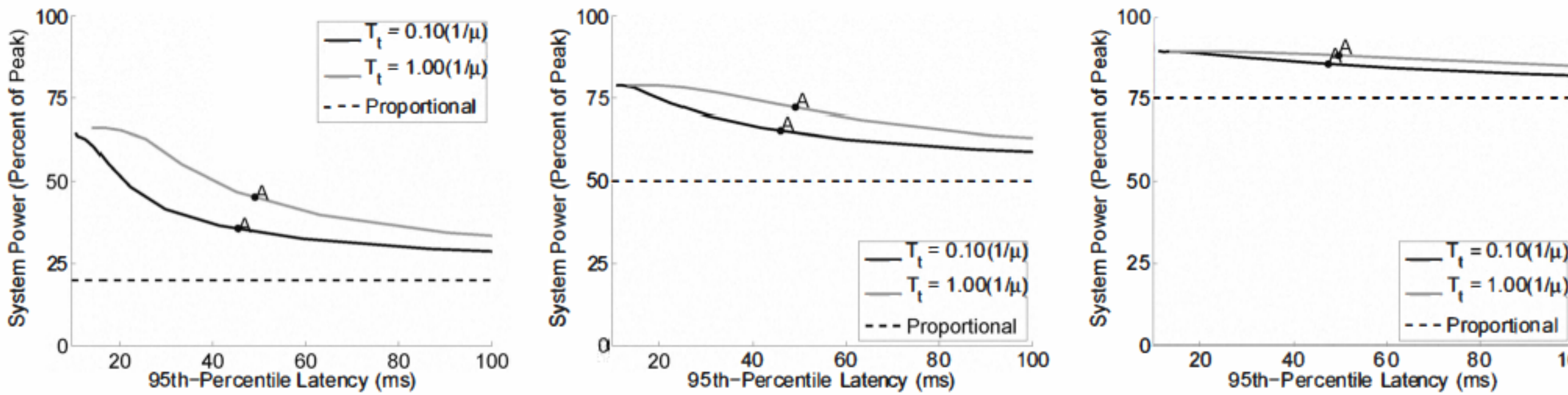



**Figure 9: Power savings potential for available low-power modes.**




(a) 20% QPS. (b) 50% QPS. (c) 75% QPS.

**Figure 15: System power vs. latency trade-off for processor and memory scaling ($P \propto f^{2.4}$):** The point "A" represents $S_{\mathrm{CPU}} = 1.5$ and "B" represents $S_{\mathrm{Mem}} = 1.5$. “A” and “B” do not appear in graph (c) because the latency exceeds 20 ms.




(a) 20% QPS. (b) 50% QPS. (c) 75% QPS.

**Figure 17: System power vs. latency trade-off for query batching with PowerNap:** “A” represents a batching policy that holds jobs for periods equal to 10x the average query service time.

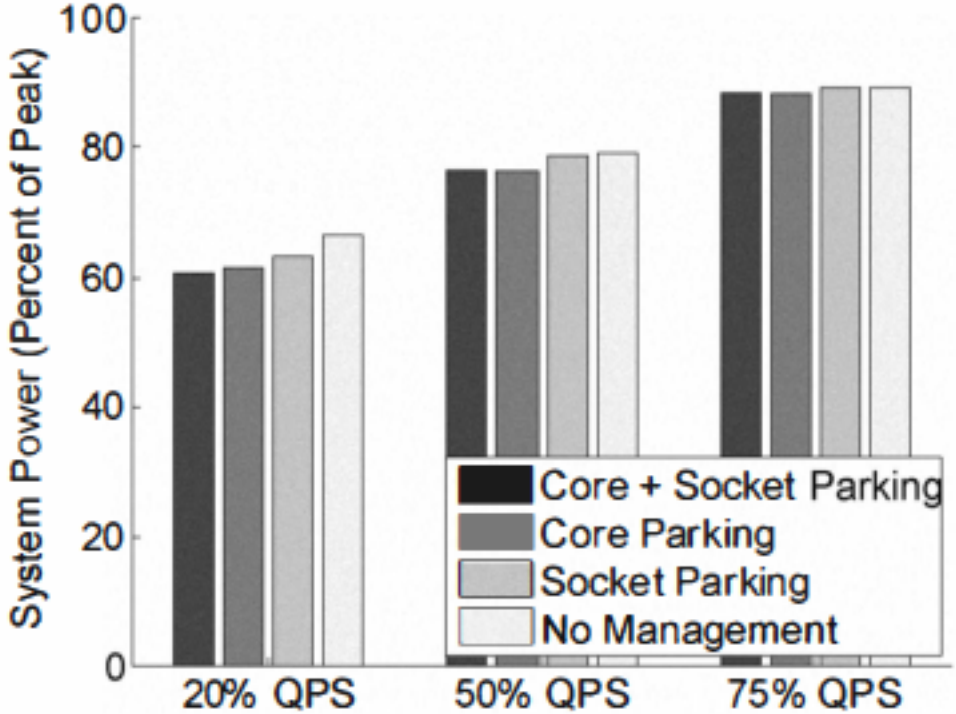



**Figure 16: System power savings for CPU idle low-power modes:** Core parking only yields marginal gains over C1E. Power savings from socket parking is limited by lack of per-socket idleness.

**Table 3: Processor idle low-power modes.**

| | Power (% of Peak) | | | |
|---|---|---|---|---|
| | Active | Idle (HLT) | Parking | |
| | | | Core | Socket |
| Per-Core (x4) | 20% | 4% | 0% | 0% |
| Uncore | 20% | 20% | 20% | 0% |

Scaling – uses active low power modes

Core – uses idle low power modes

PowerNap – System-idle low power modes

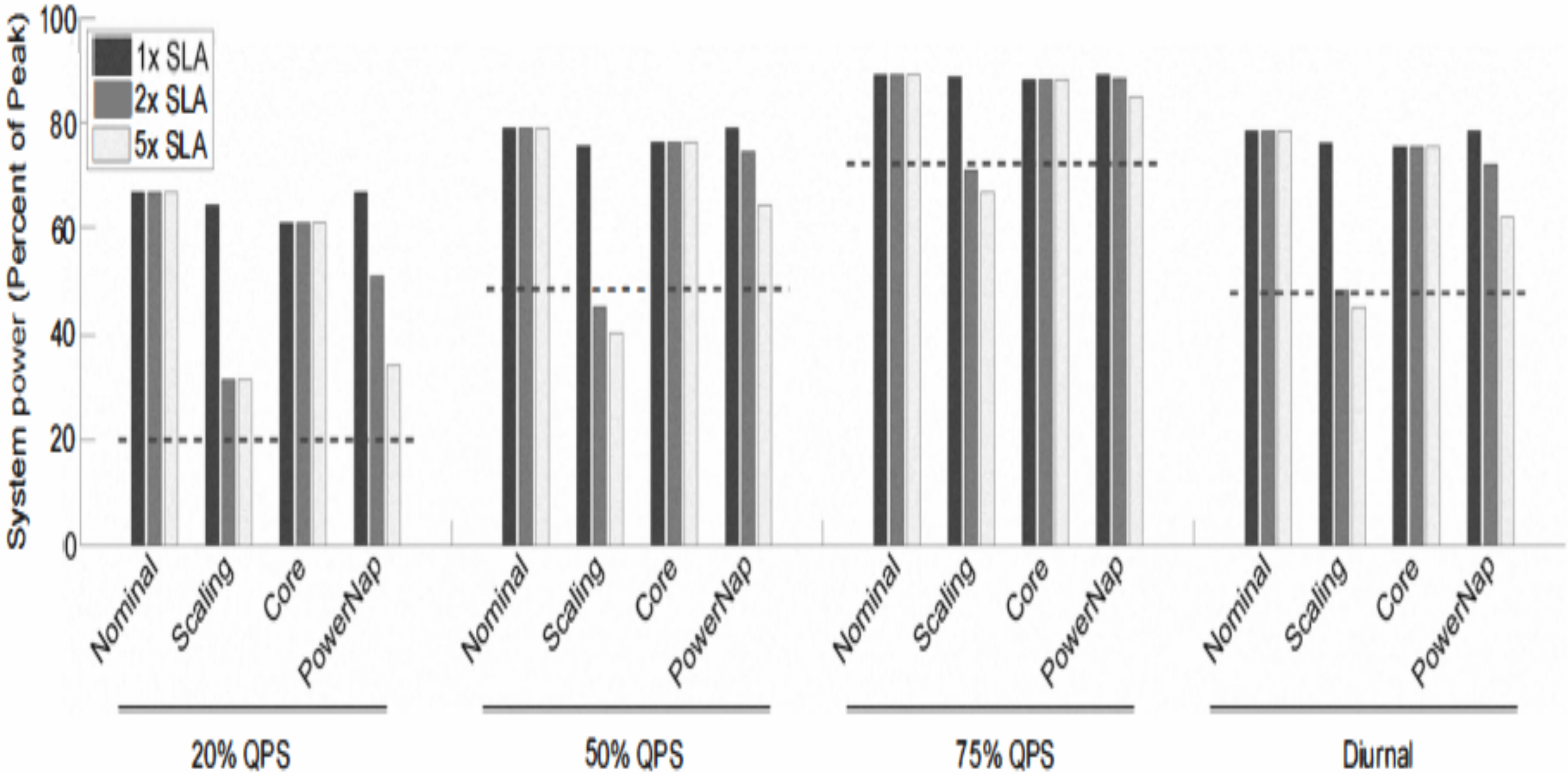



**Figure 18: Power consumption at each QPS level for a fixed 95th-percentile increase:** The dotted line at each QPS level represents the power consumption of an energy proportional system. "Diurnal" represents the time-weighted daily average from Figure 1. An energy-proportional server would use 49% of its peak power on average over the day.

# Question

- Authors say OLDI needs a new kind of power management utilizing active low power modes. Why do they say new? DVFS already uses those, right?
- They say DVFS will be less significant in future, why?

- Why do they argue for full-system active low power modes?
- What is the significance of the tail in the distribution?
- $95^{th}$ percentile is 2.4X mean

# QUEUING THEORY BASICS

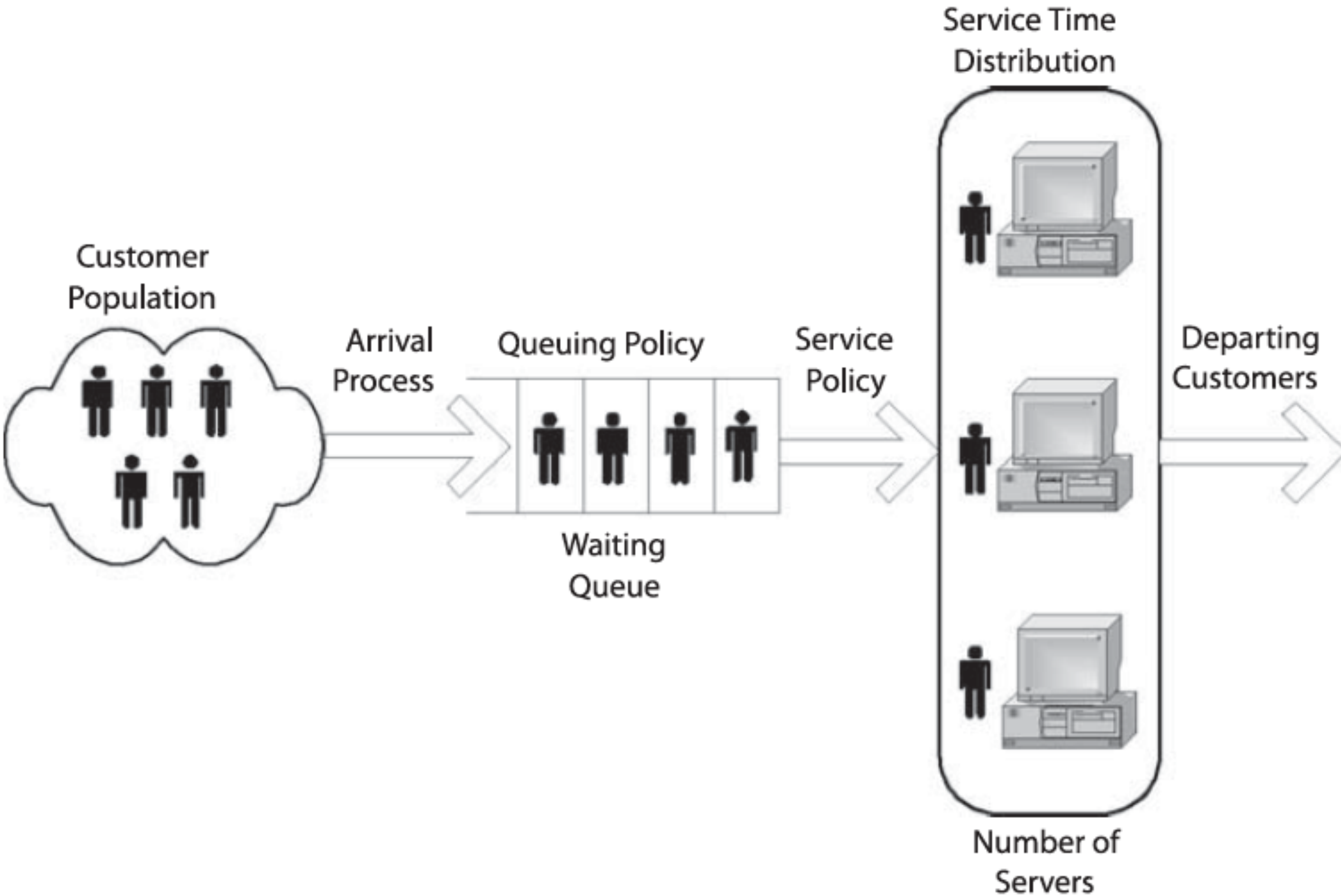



*Figure 10.1* A Generic queueing system.

# KENDALL's NOTATION

## 10.2.2 *A generic queueing system*

A generic queueing system is represented by a six-tuple notation, given by A/S/m/B/N/SD, where the first term stands for the arrival process, the second term represents the service time distribution, the third term denotes number of servers, the fourth term represents the buffer or queue size, the fifth term represents the population size, and the last term represents the service discipline [4]. A general queueing system depicting the six terms is shown in Figure 10.1.

the arrival and service time distributions, the commonly used distributions are exponential or memoryless (M), deterministic (D), and general (G). Other distributions such as Erlang and hyperexponential have been used to capture the service time variation of computer systems [5].

*M/M/1*: This is the simplest queueing system to analyze. The arrival and service times are exponentially distributed (Poisson processes), and the system consists of only one server. This queueing system can be applied to a wide variety of problems because any system with a very large number of independent customers can be approximated as a Poisson process. However, exponential service time distribution is not realistic for many applications and, thus, is only a crude approximation.

*M/D/n*: The arrival process is a Poisson process and the service time distribution is deterministic. The system has *n* servers (e.g., a ticket booking counter with *n* cashiers), and the service time is the same for all customers.

*G/G/n*: This is the most general queueing system, where the arrival and service time distributions are both arbitrary. The system has *n* servers. This is the most complex system, for which no analytical solution is known.

Let us assume that $A$ is the number of arrivals during time $T$ to the queueing system, depicted in Figure 10.1, $C$ is the number of completion during this observation period, and $B$ is the system busy time. Using these measured quantities, we can define the following simple relations:

- Arrival rate

$$\lambda = \frac{A}{T}$$

- Throughput

$$X = \frac{C}{T}$$

- Utilization

$$U = \frac{B}{T}$$

- Mean service time

$$S = \frac{B}{C}$$

### 10.2.3.1 *Utilization law*

- M

$$U = \frac{B}{T} = \frac{C}{T} \times \frac{B}{C} = XS$$

### *10.2.3.3 Little's law*

Little's theorem states that the average number of customers ($N$) can be determined as

$$N = \lambda R, \tag{1}$$

$\lambda$

where is the average customer arrival rate and $R$ is the average service time of a customer. The proof of this theorem can be found in any standard textbook on queueing theory [1]. Here, we will focus on an intuitive under-

# Queueing models are usually solved using Markov Chain (MC) models.

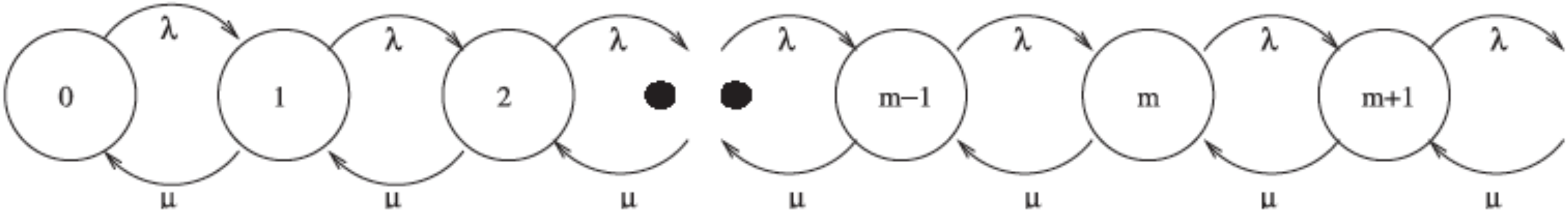



*Figure 10.2* State transition diagram of the *M/M/1* system.

First we define, the traffic intensity (sometimes called occupancy) as $\rho = (\lambda/\mu)$. For a stable system, the average service rate should always be higher than the average arrival rate. (Otherwise the queues would grow indefinitely).

Now, using the state probabilities, the mean number of customers in the system ($N$) becomes

$$E[N] = \sum_{i=1}^{\infty} np_n = \frac{\rho}{1-\rho} \quad (1)$$

Now, using the state probabilities, the mean number of customers in the system ($N$) becomes

$$E[N] = \sum_{i=1}^{\infty} np_n = \frac{\rho}{1-\rho} (1)$$

The average response time including service time, is computed using Little's Law as $N = R$ or

$$E[R] = \frac{1}{\mu - \lambda}$$

# My old notes for M/M/1 queue

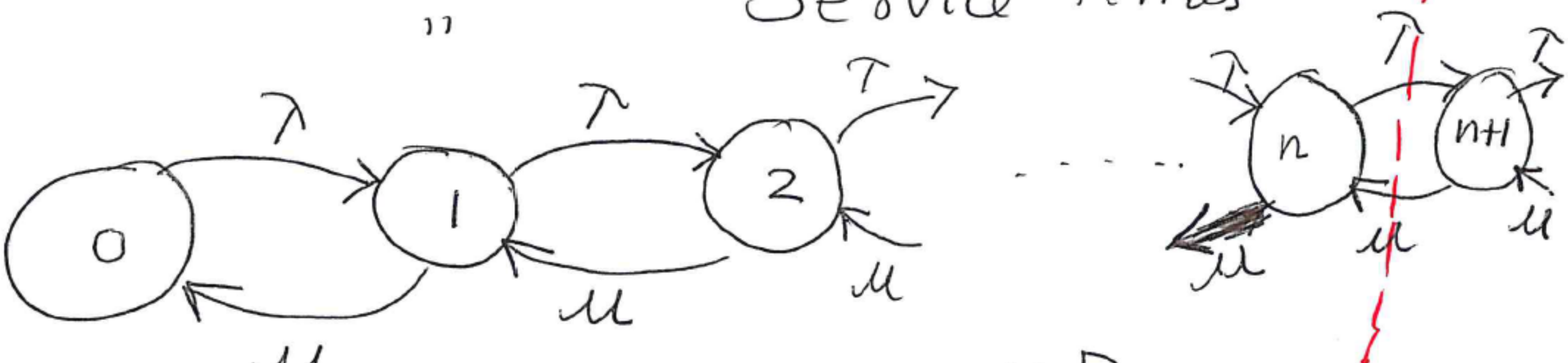
## M/M/1 Queue

(5)

Single Queue Single Server

Exponential Arrival

" Service times




For steady state $\lambda \cdot P_{n-1} = \mu P_n$

$$\therefore P_n = \frac{\lambda}{\mu} \cdot P_{n-1}$$

But $\frac{\lambda}{\mu} = \rho$

But $\frac{\lambda}{\mu} = \rho$

$\therefore P_1 = \rho P_0 ; \quad P_2 = \rho P_1 = \rho^2 P_0 ; \quad P_3 = \rho^3 P_0$

But all probabilitees add to 1

$P_0 + \rho P_0 + \rho^2 P_0 + \rho^3 P_0 \ldots = 1$

$$P_0 = \frac{1}{1+\rho+\rho^2+\rho^3+\ldots} = \sum_{n=0}^{\infty} \frac{1}{\rho^n} = 1-\rho$$

With $\rho < 1$, this sum

$$\therefore \quad P_n = (1-\rho)\,\rho^n$$

1) Average # of jobs in system be E[n]

$$E[n] = \sum_{n=0}^{\infty} n \cdot P_n = \sum_{n=0}^{\infty} n \cdot (1-\rho)\rho^n = \frac{\rho}{1-\rho}$$

2) $Var[n] = \frac{\rho}{(1-\rho)^2}$

6

4) queue length = $\frac{\rho^2}{1-\rho}$

5) wait time = $\frac{\rho}{\mu(1-\rho)}$

6) Prob of finding K or more jobs in the system = $P_r(K \geq K)$

$$= \sum_{n=K}^{\infty} P_n$$

$$= \rho^K$$

7) Utilization = $\rho$

$$1 - P_0 = 1 - (1-\rho) = \rho$$