

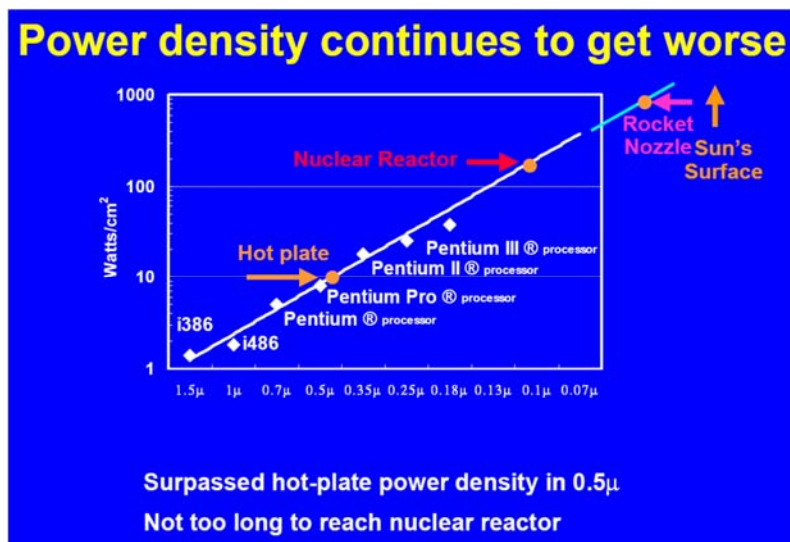
# CS2410: Computer Architecture

## Power and energy

Sangyeun Cho

Computer Science Department  
University of Pittsburgh

## Chips get hotter



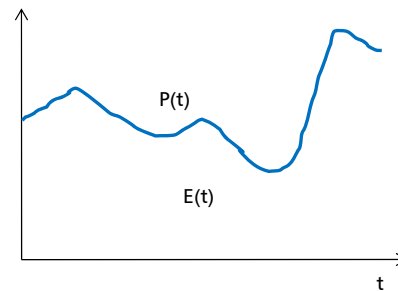
Pollack (Intel)

# Power is important

- Maximum power limit
  - Imposed by reliability requirements, power supply design and environmental considerations
  - <100W usually
  - This limit is not going to be relaxed any time soon
  - When you approach this limit, you may need to reduce power consumption by slowing down processor speeds
- Battery life
  - Needs more aggressive energy saving techniques
- Power is as important as performance today

## Some metrics

- Power
  - W (watt)
  - A time-varying quantity
- Energy
  - J (joule)
  - Wh (watt-hour)
- Power density
  - $W/cm^3$
- Optimizing one metric may not always optimize others
- Energy-delay product (EDP)
  - ED
  - $E^m D^m$




# Power basics

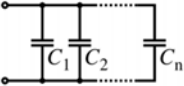
- Charge (Q) is property of fundamental particles
  - Coulomb ( $6.24 \times 10^{18} \times$  charge on an electron)
  - Charge is conserved
- Current (I) is the flow of (positive) charge
  - $I = dQ/dt$
  - Amperes = Coulombs / second
- Voltage (V) is the energy change from moving some charge
  - $V = dE/dQ$
  - Volts = Joules / Coulomb
  - Synonymous with potential
  - Always relative
    - Often to an implicit reference (e.g., ground = 0V)

# Power basics

- Power is energy per time
  - $P = dE/dt = (dE/dQ) \times (dQ/dt) = VI$
  - Watts = Joules per second
- Energy and power are often confused
  - Energy is an amount: joules, calories, kWh
  - Power is a rate: watts, horsepower
- Mechanics and thermodynamics
  - Energy  $\leftrightarrow$  work
  - Energy  $\leftrightarrow$  heat

# Power basics

- Resistor 
  - Circuit element that impedes the flow of charge
  - Ohm ( $\Omega$ )
  - $I = V/R$ ;  $V = IR$

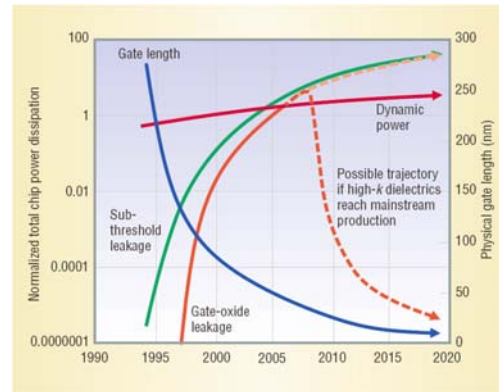
- Capacitor 
  - Capacitors store energy in the electric field created by an accumulation of charge
  - $C = dQ/dV$ ;  $Q = CV$

# Power basics

- Energy stored in a capacitor
  - $E = \int V dQ = \int q/C dQ = Q^2/2C$
  - Since  $Q = CV$ ,  $E = (CV)^2/2C = 0.5CV^2$
- Some energy is lost when charging from a constant voltage, say  $V_{DD}$ 
  - Energy lost is  $0.5CV_{DD}^2$
- Total energy to charge the capacitor =  $CV_{DD}^2$

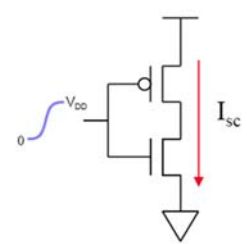
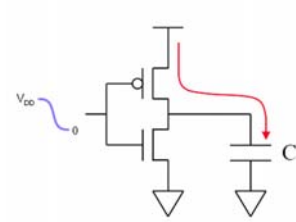
# Power in CMOS circuits

- $P(t) = I_{DD}(t)V_{DD}$
- $P_{total} = P_{dynamic} + P_{static}$
- Dynamic power
  - Charging and discharging of load capacitances
  - “Short-circuit” current while both pMOS and nMOS networks are on
- Static power
  - Sub-threshold conduction through off transistors
  - Leakage current through reverse-biased diodes



# Dynamic power

- Switching power
  - Charging and discharging of load capacitances
  - $P \propto \alpha CV^2f$
- Short-circuit current
  - Current flows from VDD to GND while both nMOS and pMOS transistors are on when input changes
  - Typically less than 10% of switching power

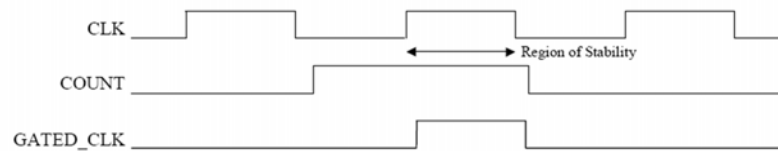


# Reducing dynamic power

- $P \propto \alpha CV^2f$
- Reduce  $\alpha C$
- Reduce  $V$
- Reduce  $f$

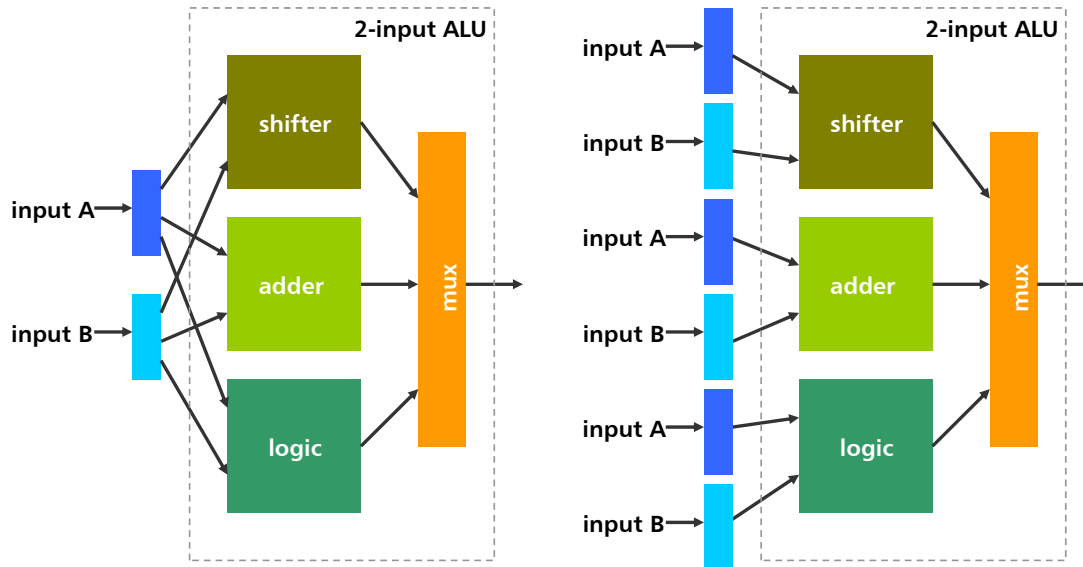
## Reducing $\alpha C$

- Guiding principle
  - Stop "unnecessary" activities
  - E.g., clock gating



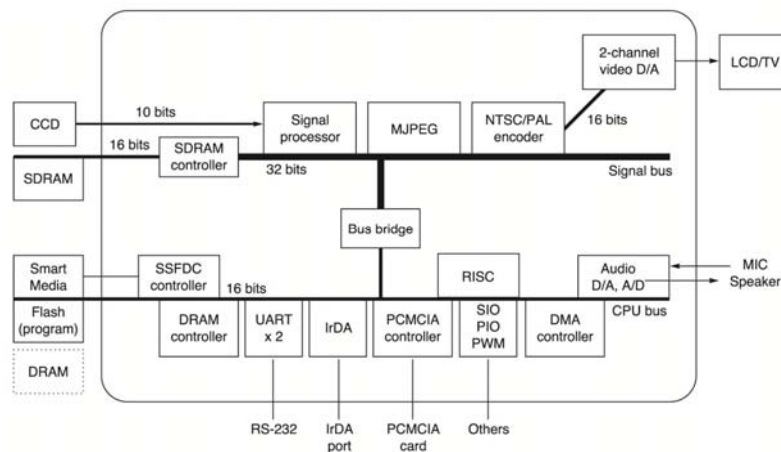
- Okay to increase  $C$  if you can decrease  $\alpha$  more
  - E.g., separate input registers for adder, logic unit, shifter, ...

# ALU example



# "System"-level clock gating

- Turn off functional units that are not used
  - E.g., MP3 decoding logic in a cell phone while you talk to the other party



# Reducing V

- V has quadratic impact on power
  - V has (linear) impact on performance (f)
- New technologies allow lower V
  - 1.8V @180nm
  - 1.2V @130nm
  - 1.0V @90nm
- DVS (Dynamic Voltage and Frequency Scaling)
  - Crusoe example
  - Intel SpeedStep

# Reducing f

- Guiding principle
  - Don't be "too" fast
- Lowering f has linear impact on P
- With a lower f, V can be made lower also! (DVS)
  - Care must be taken not to cause circuit failure due to "too" low V for a given f
  - Processor vendor provides a table with safe V-f pairs

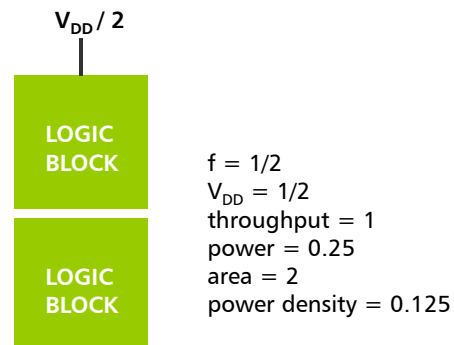
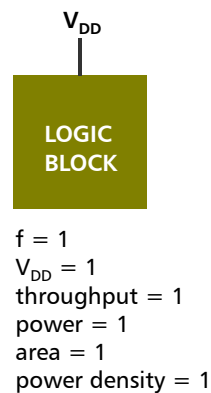


# Reducing f

- Steps to reduce f
  - First reduce f
  - Then reduce V
- Steps to increase f
  - First increase V
  - Then increase f

# Reducing f via parallel processing

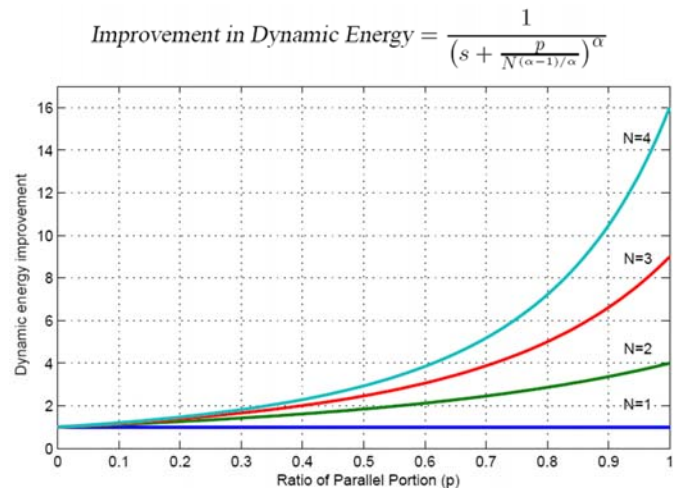
- Based on the “same throughput” assumption



(Borkar. Micro 2006.)

# Energy saving via parallel processing

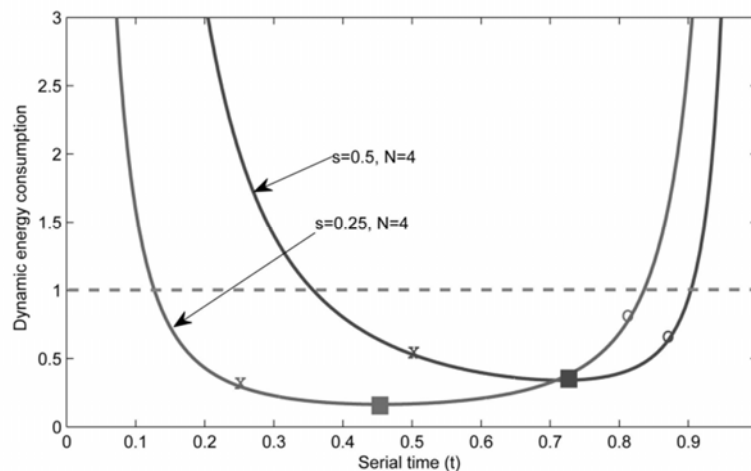
- Based on the "same throughput" assumption



(Cho and Melhem. Computer Architecture Letters, 2007.)

# Energy saving via parallel processing

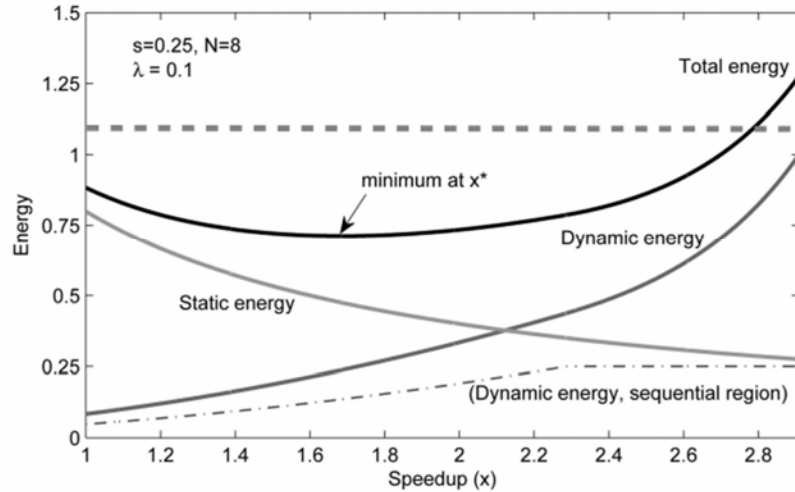
- Effect of processor speeds on energy?



(Cho and Melhem. Computer Architecture Letters, 2007.)

# Energy saving via parallel processing

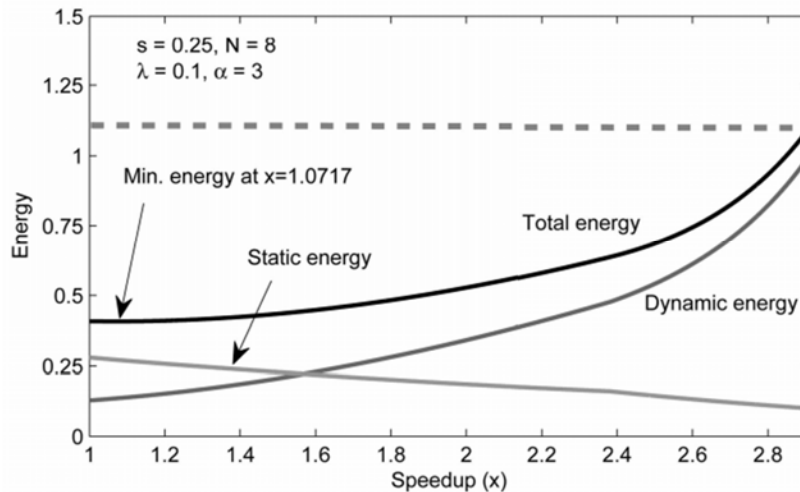
- Target speed and energy savings?



(Cho and Melhem. Computer Architecture Letters, 2007.)

# Energy saving via parallel processing

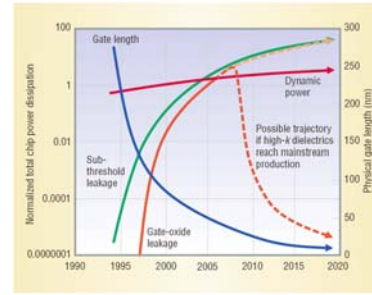
- Benefit of per-core "turn-off" capability



(Cho and Melhem. Computer Architecture Letters, 2007.)

# Reducing static power

- Static power due to
  - Sub-threshold current
  - Gate leakage
  - Reverse-biased junction leakage
- Inherent to CMOS transistors
- Static power is increasing faster than dynamic power in advanced technologies
- There is no "pure" architectural solution here
- Following slides are based on Morad and Walter, 2004.

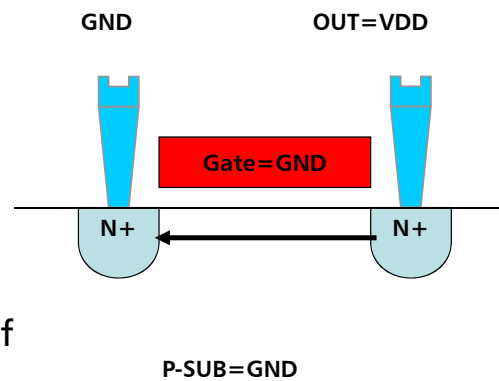


CS2410: Computer Architecture

University of Pittsburgh

# Sub-threshold leakage

- Caused by weak inversion
- Grows exponentially with the lowering of  $V_{TH}$
- Grows exponentially with increasing temperature
- Grows linearly with total widths of transistors



$$I_{sub} = K_1 W e^{-V_{th}/nkT} (1 - e^{-V/kT})$$

Use high Vth

Disconnect the power supply

CS2410: Computer Architecture

University of Pittsburgh

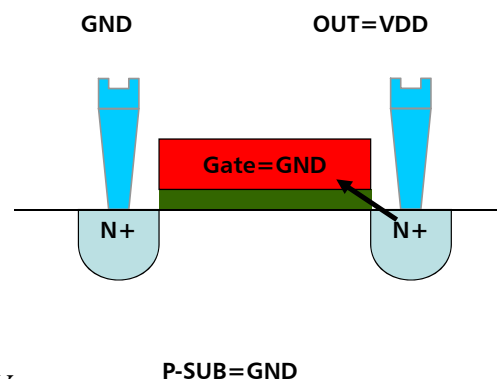
# High $V_{TH}$ vs. low $V_{TH}$

- Technology scaling results in low- $V_{TH}$  devices
  - Faster but leaky
  - Lowering  $V_{DD}$  is slower
- High-speed logic implementations
  - Use low- $V_{TH}$  devices for performance critical parts
  - Use high- $V_{TH}$  devices for others

# Gate-oxide leakage

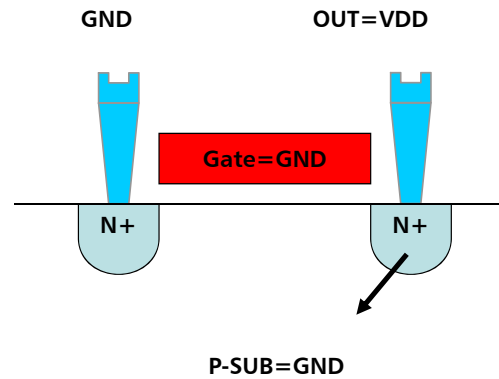
- Increases exponentially with the decreasing of  $T_{ox}$
- Can be solved by high-k materials
- Proportional to gate width

$$I_{ox} = K_2 W \left( \frac{V}{T_{ox}} \right)^2 e^{-\alpha T_{ox} / V}$$



# Reverse-biased PN leakage

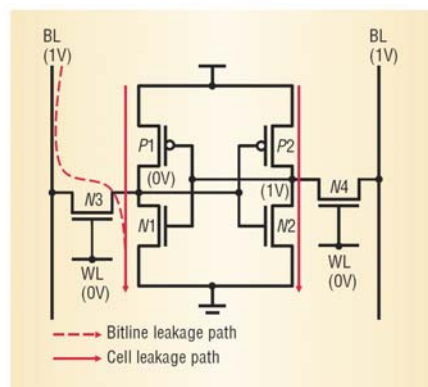
- PN junction in reverse bias
- Leakage proportional to diffusion area
- Exponentially sensitive to high temperature and high voltage



$$I_{pn} = J_{leakage, p+n} \left( e^{\frac{qV}{kT}} - 1 \right) A$$

# Memory leakage

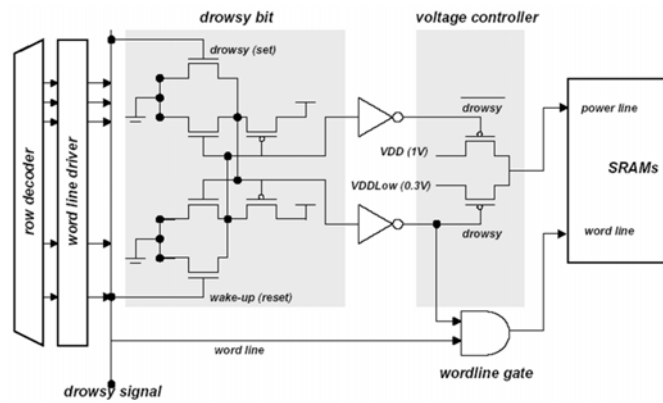
- Leakage constitutes 70% of cache power
- Cell leakage, bitline leakage



- State-preserving vs. state-destructive circuit techniques

# State-preserving technique

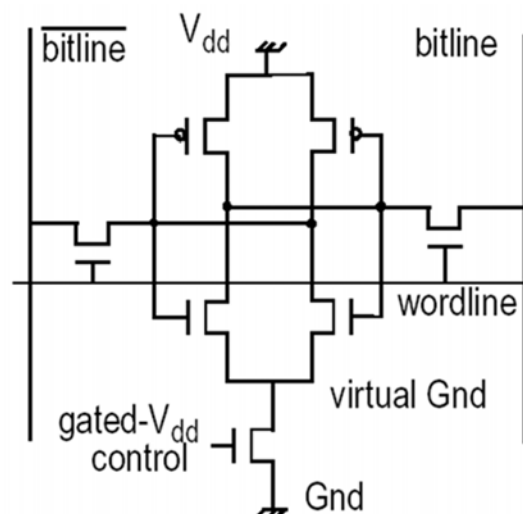
- Dual  $V_{DD}$



- Adaptive Body Bias (ABB)
  - Apply body bias voltage

# State-destructive technique

- Gated- $V_{DD}$



# Comparison of different techniques

	Advantages	Disadvantages	Leakage power in low power mode
<b>DVS</b>	<ul style="list-style-type: none"><li>• Retains cell information in low-power mode.</li><li>• Fast switching between power modes.</li><li>• Easy implementation.</li><li>• More power reduction than ABB-MTCMOS.</li></ul>	<ul style="list-style-type: none"><li>• Process variation dependent.</li><li>• More SEU noise susceptible.</li></ul>	6.24nW
<b>ABB-MTCMOS</b>	<ul style="list-style-type: none"><li>• Retains cell information in low-power mode.</li></ul>	<ul style="list-style-type: none"><li>• Higher leakage power.</li><li>• Slower switching between power modes.</li></ul>	13.20nW
<b>Gated-V<sub>DD</sub></b>	<ul style="list-style-type: none"><li>• Largest power reduction.</li><li>• Fast switching between power modes.</li><li>• Easy implementation.</li></ul>	<ul style="list-style-type: none"><li>• Loses cell information in low-power mode.</li></ul>	0.02nW

## Final remarks

- Computer architecture is a changing sub-field of computer science
  - New technologies are continuously developed and deployed
  - More demanding applications emerge
  - Higher performance and affordability requirements
- Current and future technical focuses
  - Optimized multicore hardware designs
    - That can be programmed easily
  - Specifying and enforcing application's quality of service
  - Virtualization – esp. server consolidation
  - Low power and energy
  - Reliability