Energy Management and Adaptive Behavior

Tarek Abdelzaher

University of Illinois at Urbana Champaign

Energy in Data Centers

- Data centers account for 1.5% of total energy consumption in the US

  (Equivalent to 5% of all US housing)

  According to the U.S. EPA Report, 2007:

- The cost of energy already accounts for at least 30% of the total operation cost in most data centers.

  According to BroadGroup (independent market research firm)

The Energy Optimization Problem

- Requires a holistic approach
- Local optimization of individual knobs is not equivalent to global optimization

Problem: Composability of Adaptive Behavior

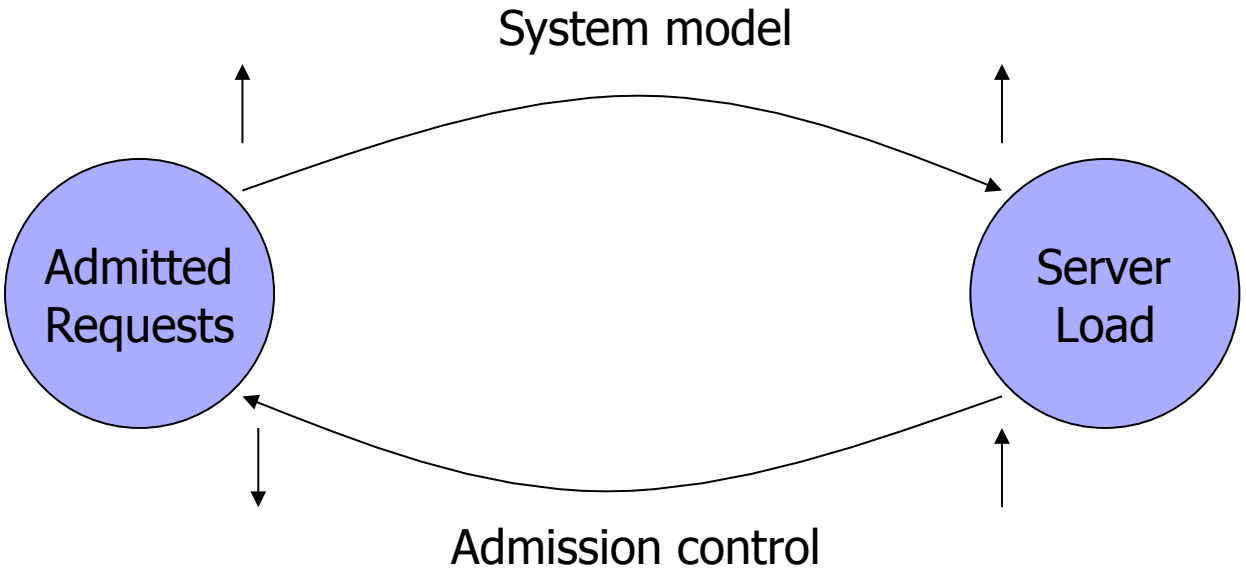
- Modern time-sensitive and performance-sensitive systems are getting more complex
  - → Manual tuning becomes more difficult,
    hence: automation
  - → Automation calls for adaptive capabilities
    (e.g., IBM's autonomic computing initiatives)
    hence: adaptive components
  - → Emerging challenge
    Composition of adaptive components
    (Locally stable but globally unstable systems?)

Locally Stable – Globally Unstable Preliminary Insights

- Positive feedback versus negative feedback

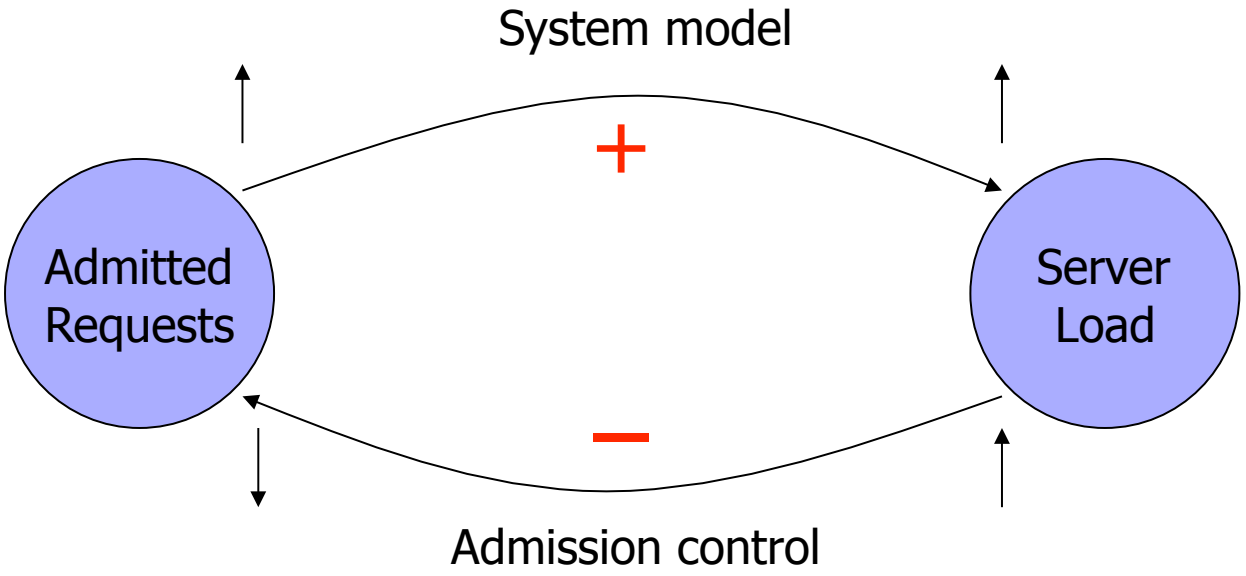
Locally Stable – Globally Unstable Preliminary Insights

- Positive feedback versus negative feedback



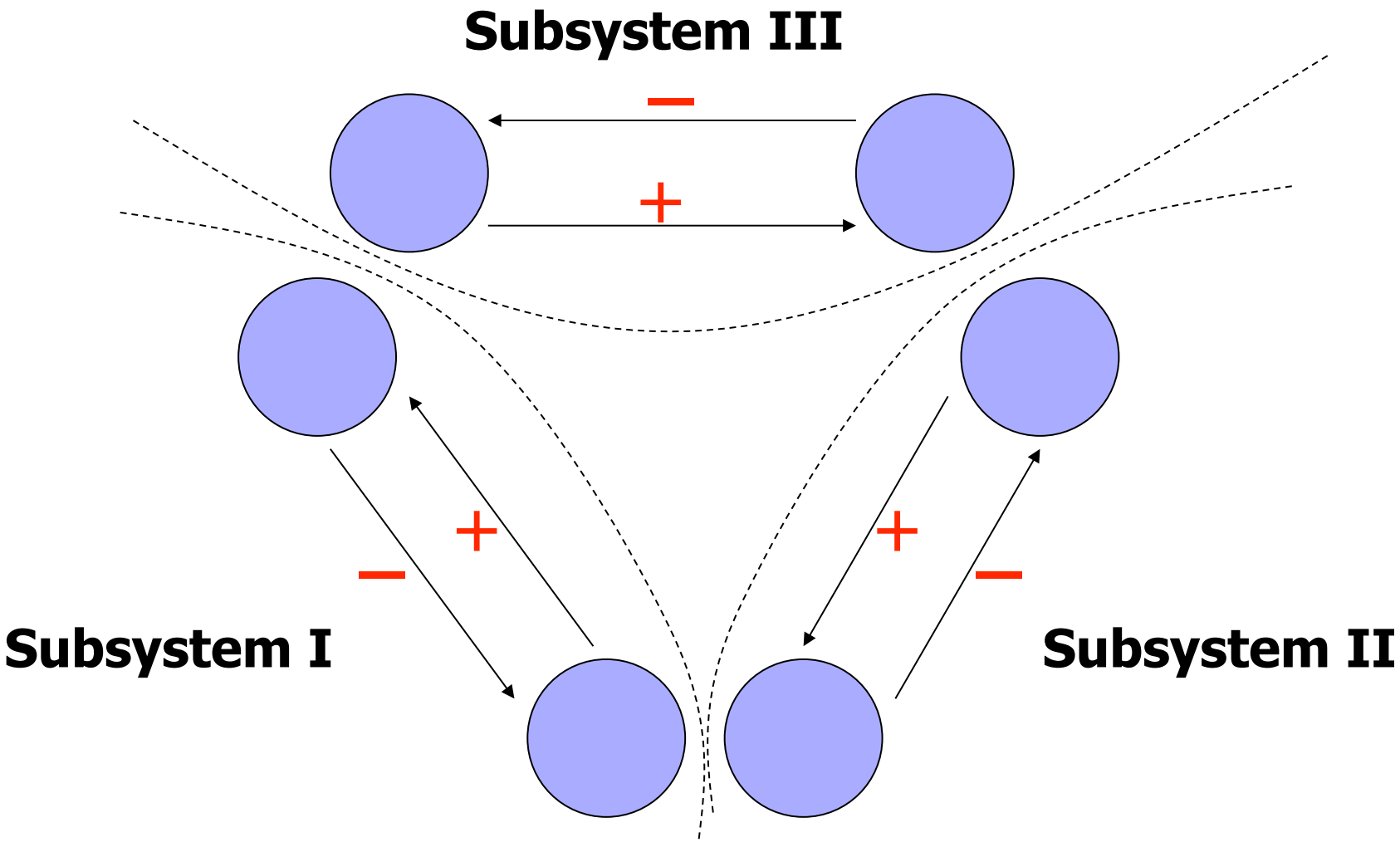
Locally Stable – Globally Unstable Preliminary Insights

- Positive feedback versus negative feedback

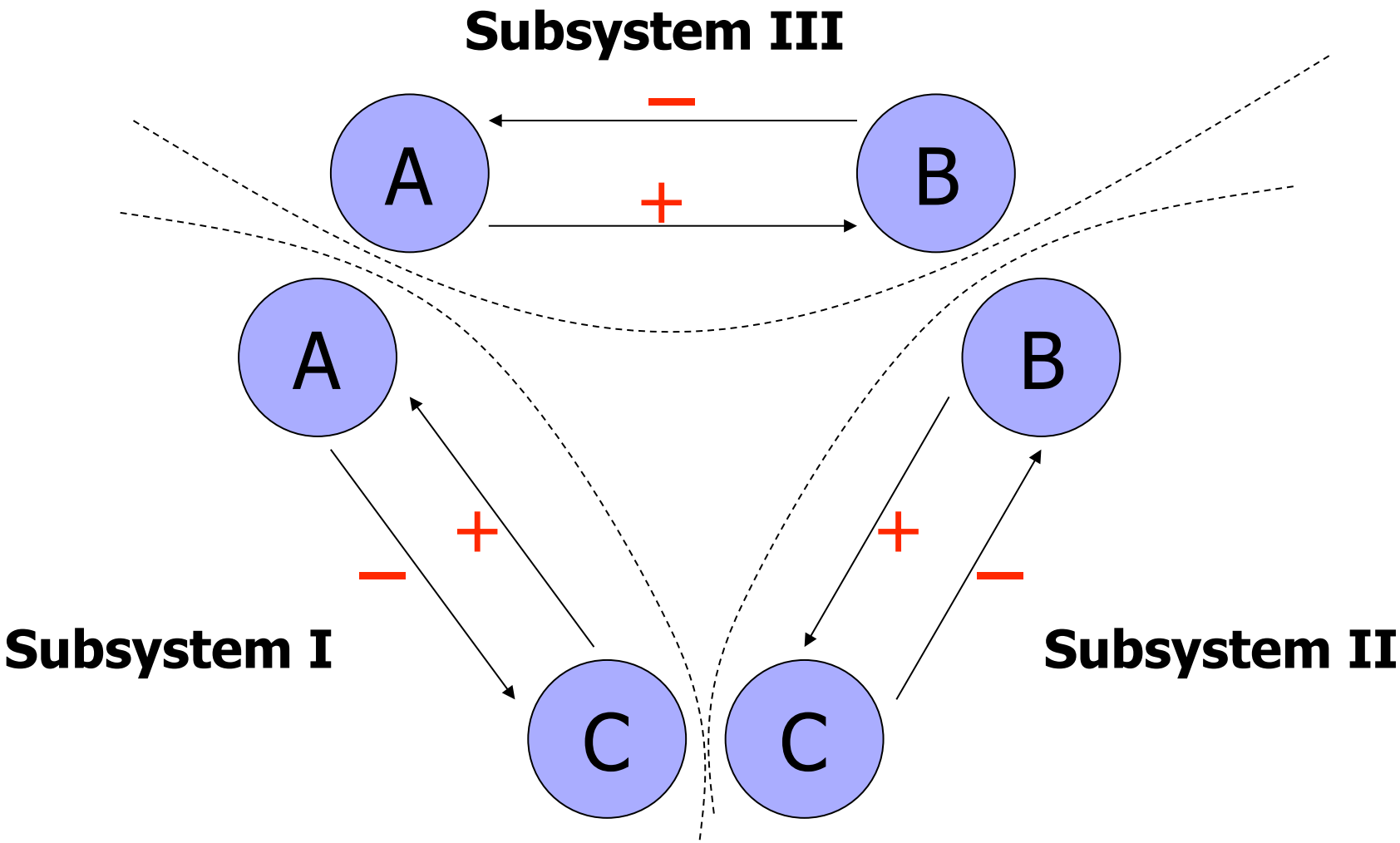


All stable feedback is negative

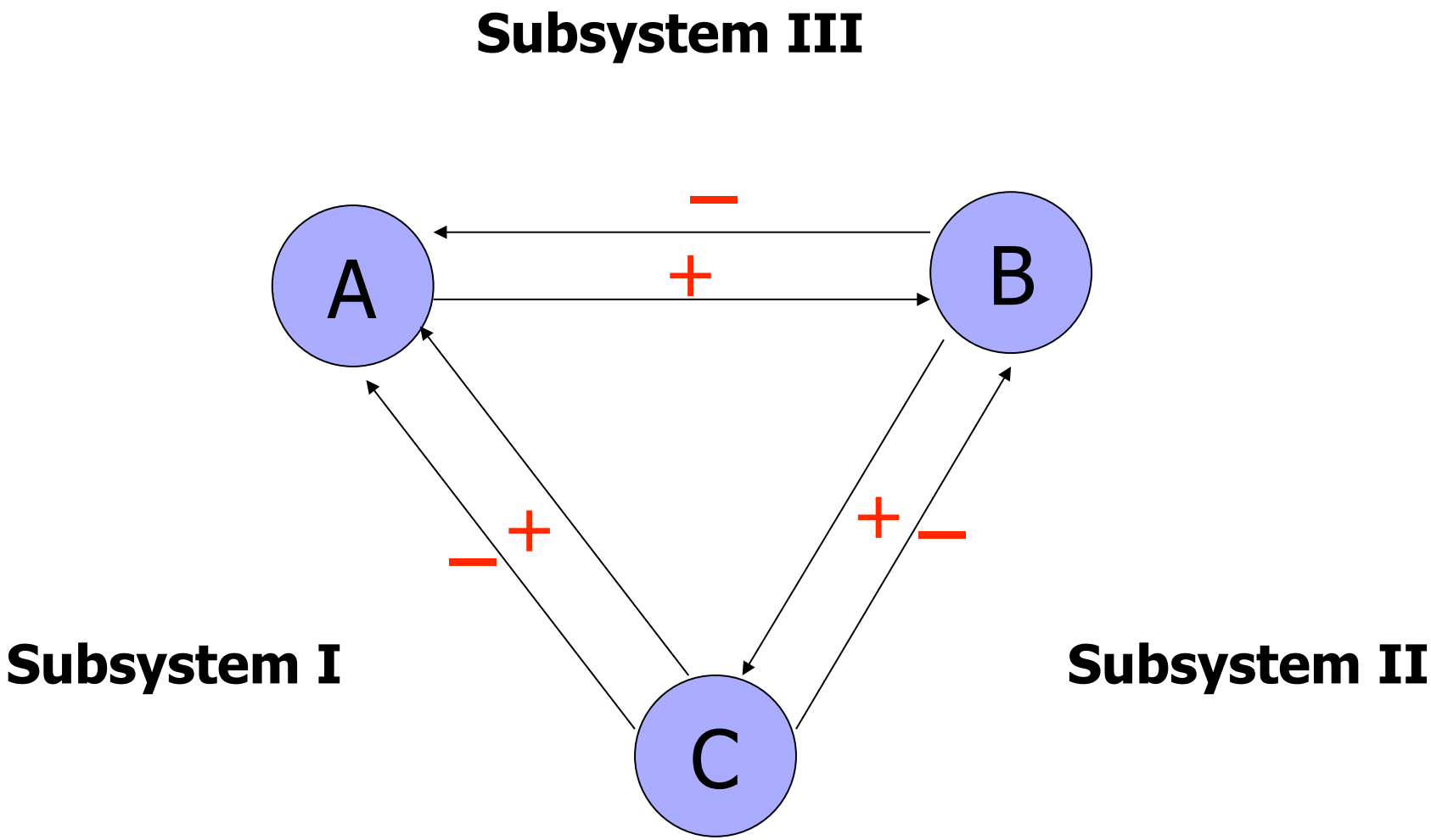
Composition of Adaptive Systems



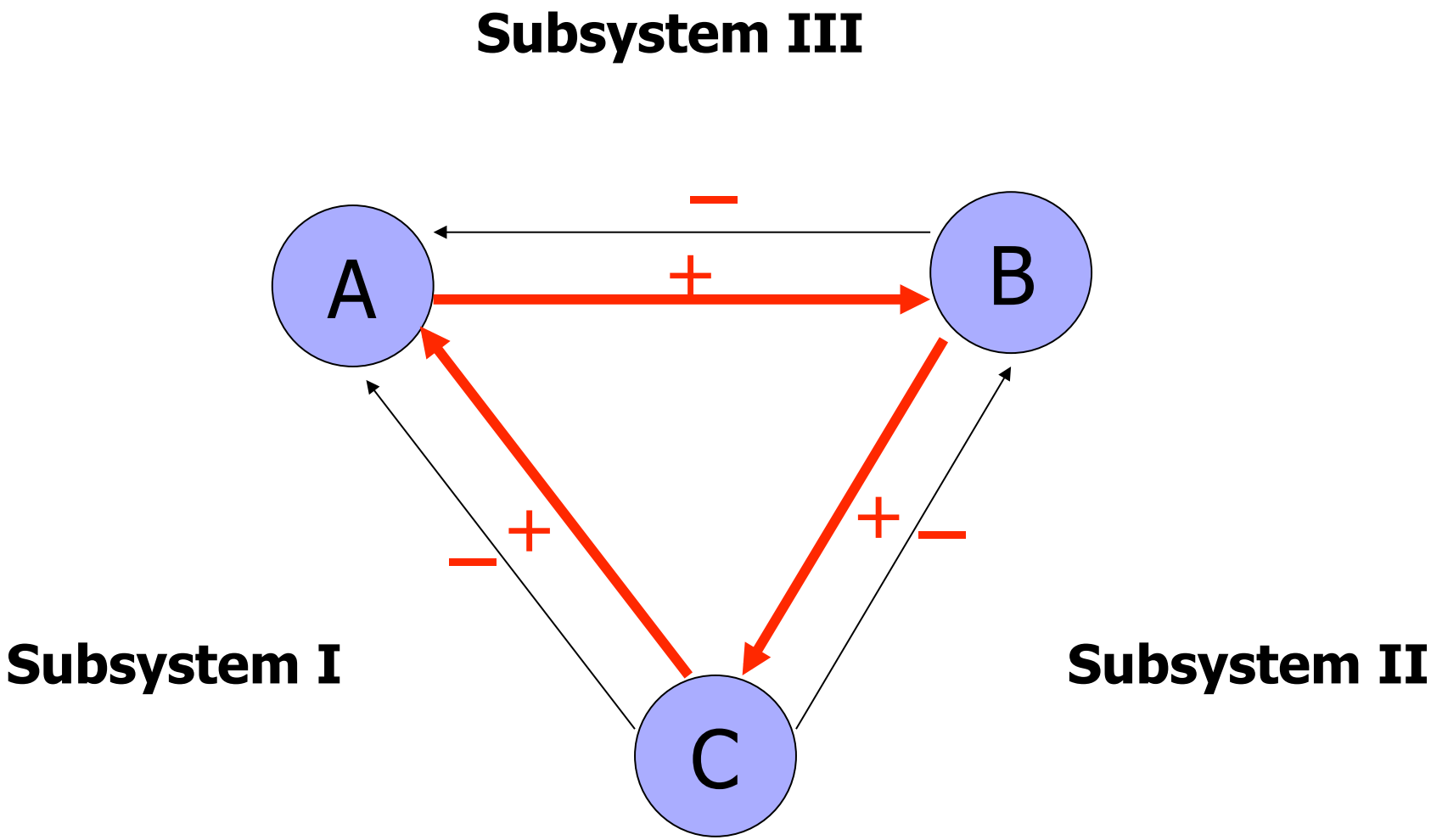
Composition of Adaptive Systems



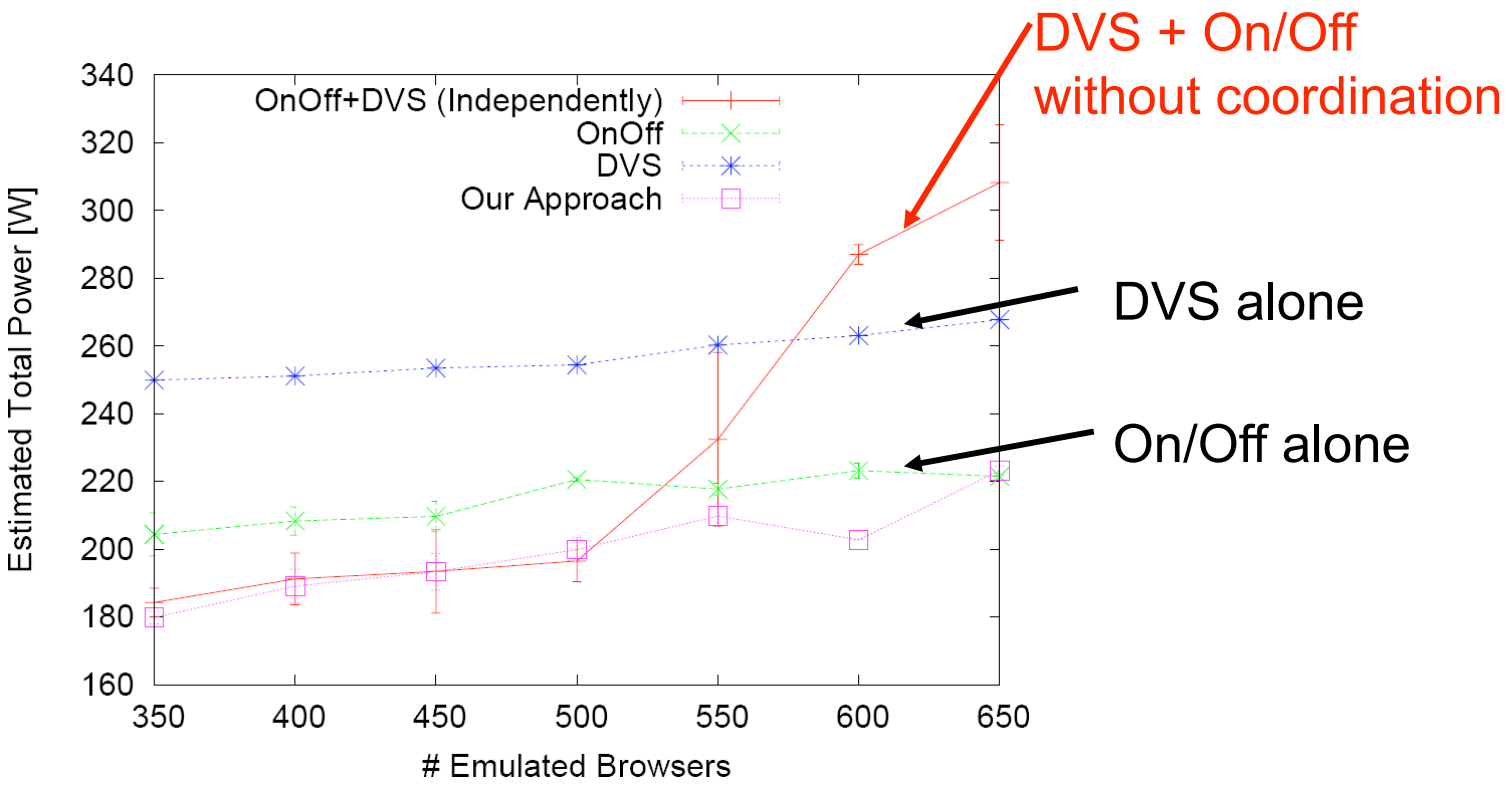
Composition of Adaptive Systems



Composition of Adaptive Systems



Example
(A Tale of Two Policies)



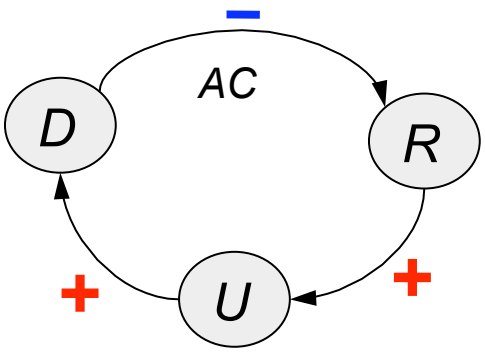
Empirical measurements from a 30-machine 3-tier testbed of a shopping site

Composability of Adaptive Behavior

- Many adaptive policies may perform well in isolation, but conflict when combined
- Example: DVS enabled QoS-aware Web server
  - DVS policy and admission control policy (AC)
  - In an underutilized server, DVS decreases frequency, hence increasing delay
  - AC responds to increased delay by admitting fewer requests
  - Unstable cycle - throughput diminishes

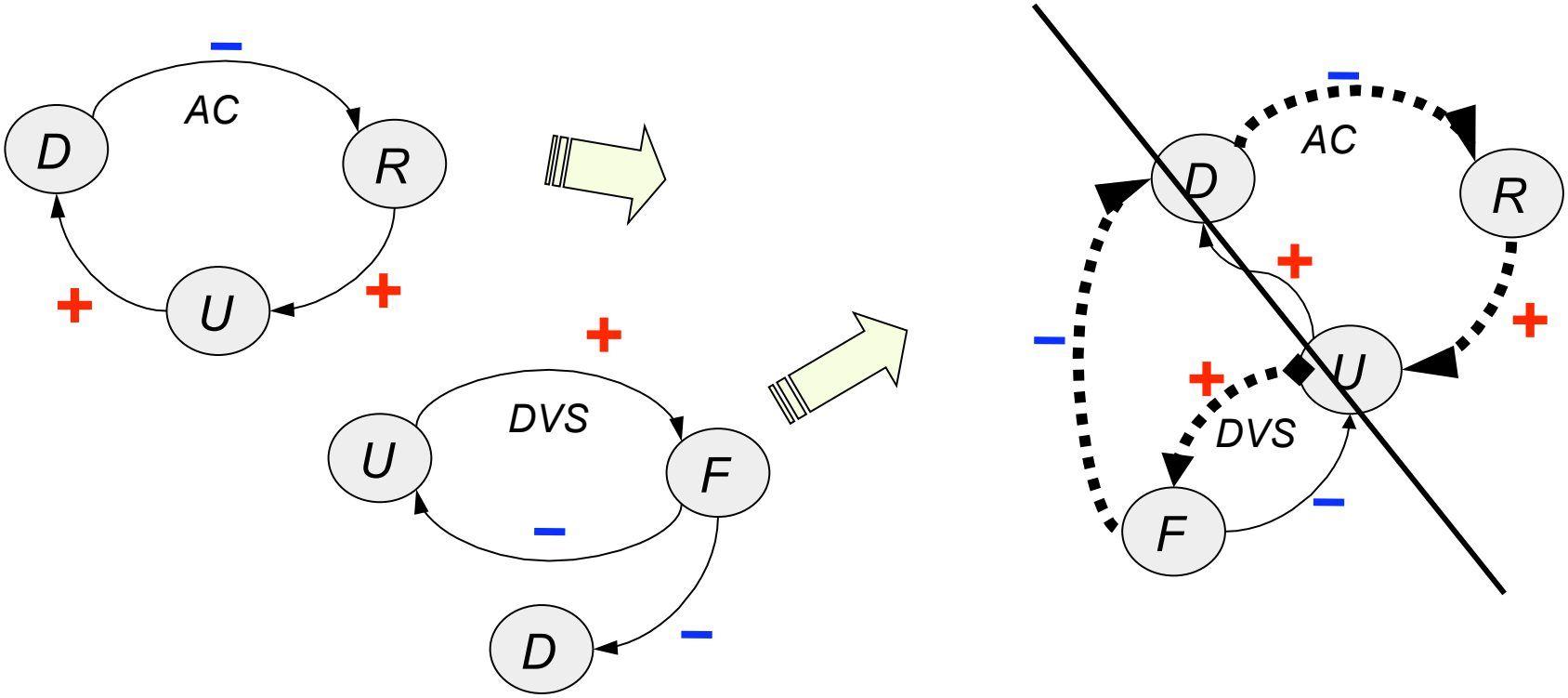
Detection of Potential Conflicts:
Introduction to Adaptation Graphs

- Adaptation graphs determine which adaptive policies conflict (if they do)
- Adaptation graphs
  - Graphical representation of causal effects among performance control knobs and system performance metrics
- A affects B: *A* → *B*
  - Changes in A cause changes in B
  - Direction of change (+, -)
  - Natural consequences or programmed behavior
  - The sign of a cycle: multiplication of the signs of all edges



Adaptation graph for Qo S-aware Web Server

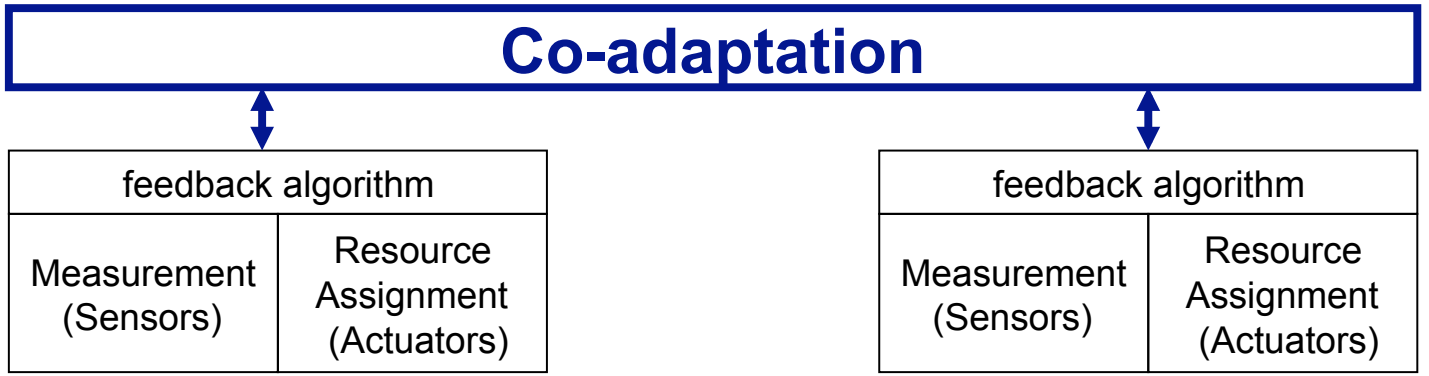
Example: DVS-enabled QoS-aware Web Server



Individual adaptation loop for each policy is stable (negative)

Combined together, unstable positive loop across policy boundaries → Use co-adaptation!!

Co-adaptation Design Methodology



Adaptive policy (software component) 1

Adaptive policy (software component) 2

Co-adaptation guides you to design a shared co-adapt module - Outputs knob settings that increases utility

Constrained optimization (Necessary condition) + Feedback control

Co-adaptation Cont.

- Step1: Casting the objective
  - Find a common objective function – minimize cost or maximize utility

- Step2: Formulating optimization problems
  - Decision variables: settings of adaptation "knobs"
  - Subject to two types of constraints
    - resource constraints
    - performance specifications

$x_1, \ldots x_n$: adaptation knob settings for policy *I*

$j = 1, \ldots, m$: resource and performance constraints

$$\min_{x_1,\ldots,x_n} \quad f(x_1, \ldots, x_n)$$

$$\text{subject to} \quad g_j(x_1, \ldots, x_n) \leq 0, \quad j = 1, \ldots, m$$

Co-adaptation Cont.

- Step3: Derivation of necessary conditions
  - Lack of accurate model for computing systems
  - Augmented by feedback to move closer to the point that increases utility
  - Use the Karush-Kuhn-Tucker (KKT) optimality condition

Γx_i ←

$$\frac{\partial f(x_1,\ldots,x_n)}{\partial x_i} + \sum_{j=1}^{m} \nu_j \frac{\partial g_j(x_1,\ldots,x_n)}{\partial x_i} = 0$$

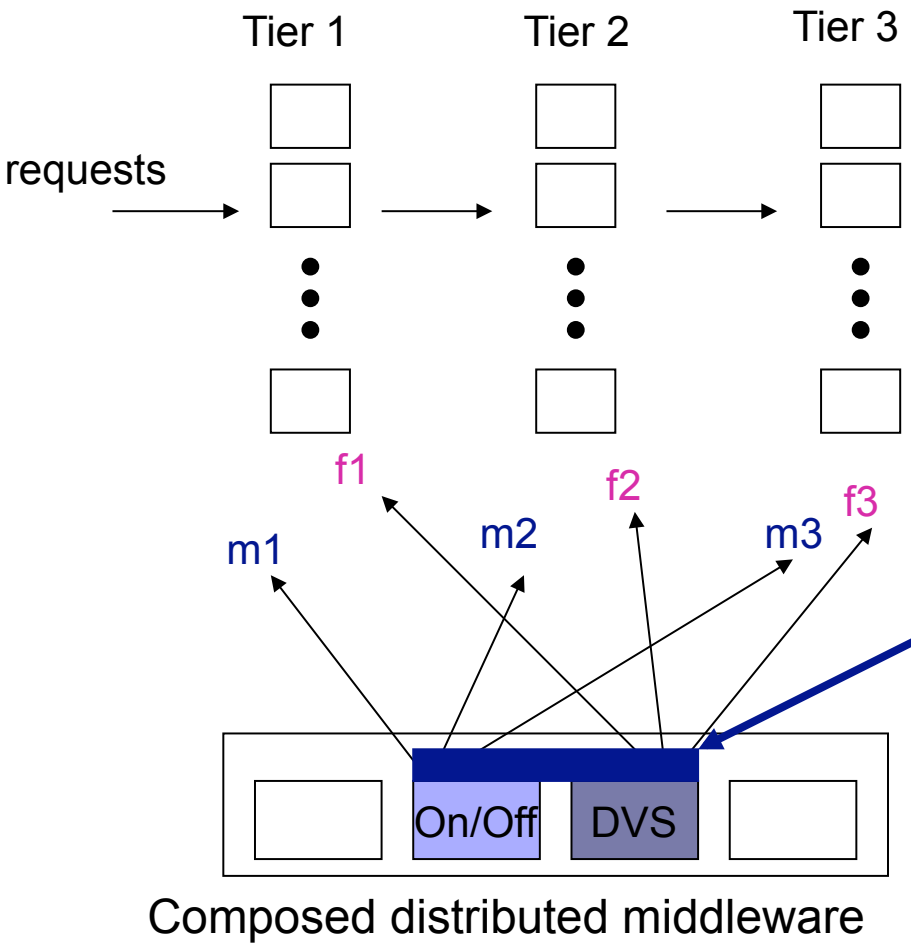
  - Necessary condition $\Gamma x_1 = \ldots = \Gamma x_n$
    - Define $\Gamma x = (\Gamma x_1 + \ldots + \Gamma x_n)/n$

Co-adaptation Cont.

- Step4: feedback control
  - Measurement based – periodic measurement to estimate Γx_i
  - Try to meet the necessary condition $\Gamma x_1 = \ldots = \Gamma x_n$ by Hill climbing
    - Pick one with the largest or smallest value of Γx_i
    - Search through the neighboring knob settings (values of Xi)
      - Reduce the error ($\Gamma x - \Gamma x_i$)
      - Maximum increase in utility subject to constraints

A Server Farm Case study

Energy Minimization in Server Farms

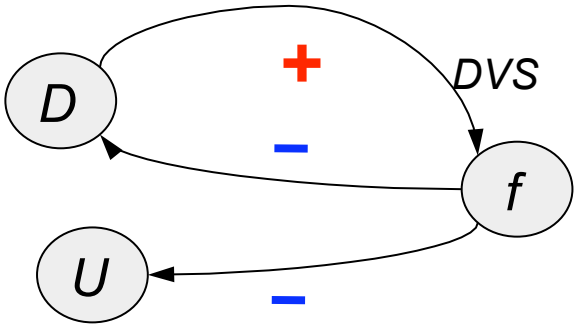


Two policies were shown to be in conflict by adaptation graph analysis: yielding more energy consumption

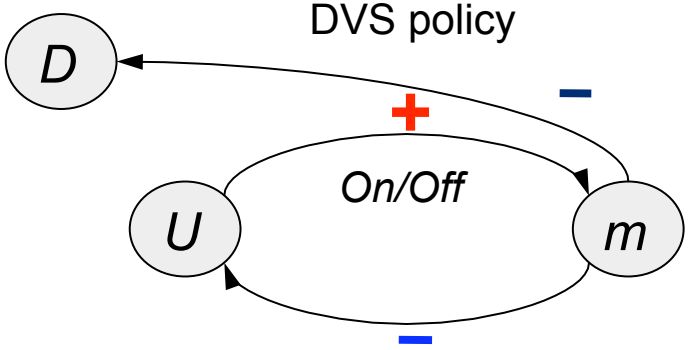
Co-adaptation finds knob settings, (m1, m2, m3, f1, f2, f3) in the direction where energy consumption is reduced

1. Incompatibility Detection:

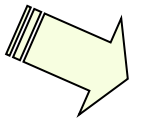
- Two adaptive policies
  - DVS policy
  - On/Off policy

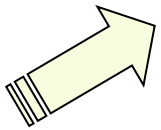


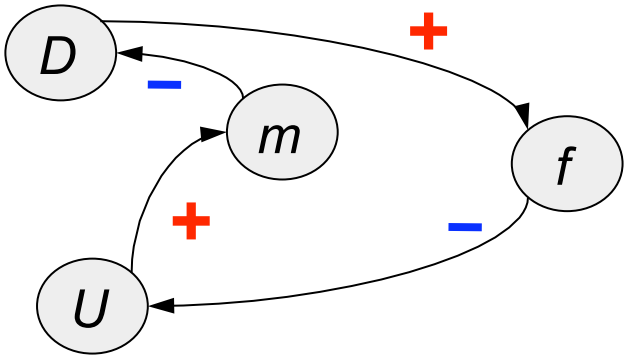
DVS policy



On/Off policy







(c) Adaptation graph for combined DVS and On/Off policies: possible interference in the control of *D* ♪

2. Design of a Co-adaptive Energy Minimization Policy

Formulate constrained optimization

$$P_i(f_i) = A_i \cdot f_i^p + B_i$$

Power estimation of a machine at tier i

$$U_i = \frac{\lambda}{\mu} = \frac{\lambda_i / m_i}{f_i} = \frac{\lambda_i}{m_i f_i}$$

Queuing equation using number of machines and arrival rate

$$P_i(U_i, m_i) = A_i \cdot \left(\frac{\lambda_i}{U_i m_i}\right)^p + B_i = \frac{A_i \lambda_i^p}{U_i^p m_i^p} + B_i$$

Power estimation function of a machine at tier i

$$\min_{U_i \geq 0,\ m_i \geq 0} P_{tot}(U_i, m_i) = \sum_{i=1}^{3} m_i \left(\frac{A_i \lambda_1^3}{U_i^3 m_i^3} + B_i\right)$$

$$\text{subject to} \quad \sum_{i=1}^{3} \frac{m_i}{\lambda_i} \cdot \frac{U_i}{1 - U_i} \leq K,$$

$$\sum_{i=1}^{3} m_i \leq M$$

Find best composition of

(m_1, m_2, m_3, U_1, U_2, U_3)

Design of a Co-adaptive Energy Minimization Policy

- Derive necessary condition for optimality
  - Karush-Kuhn-Tucker (KKT) condition

$$\frac{\lambda_1^4(1-U_1)^2}{m_1^3U_1^4} = \frac{\lambda_2^4(1-U_2)^2}{m_2^3U_2^4} = \frac{\lambda_3^4(1-U_3)^2}{m_3^3U_3^4}$$

$$\Gamma(m_1, U_1) = \Gamma(m_2, U_2) = \Gamma(m_3, U_3)$$

Try to find (m_1, m_2, m_3, U_1, U_2, U_3) tuple that balance the condition.

Design of a Co-adaptive Energy Minimization Policy

$$\frac{\lambda_1^4(1-U_1)^2}{m_1^3U_1^4} = \frac{\lambda_2^4(1-U_2)^2}{m_2^3U_2^4} = \frac{\lambda_3^4(1-U_3)^2}{m_3^3U_3^4}$$
$$\Gamma(m_1, U_1) = \Gamma(m_2, U_2) = \Gamma(m_3, U_3)$$

- Feedback Control
  - Goal: balance the necessary condition in the direction to reduce energy consumption
  - *When delay constraint violated*: Pick the most overloaded tier (the lowest $\Gamma(m_i, U_i)$)
  - *Else*: Pick the most underloaded tier (highest $\Gamma(m_i, U_i)$)
  - Choose (m_i, U_i) pair that makes the error within a bound and yields the lowest total energy
    - Error = $\Gamma_x - \Gamma(m_i, U_i)$, where Γ_x is average of $\Gamma(m_i, U_i)$

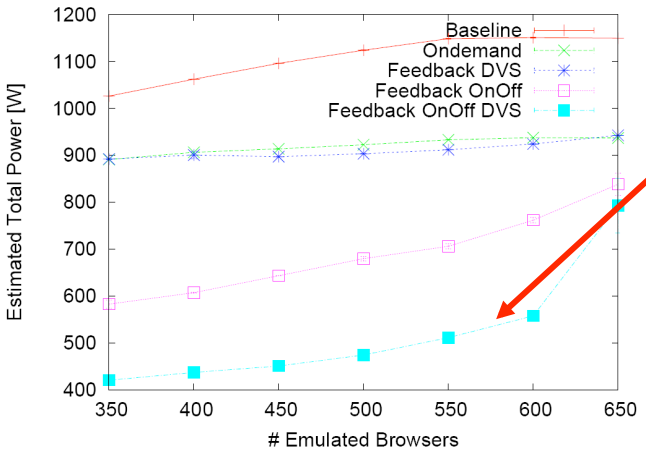
Evaluation on a Server Farm Testbed

- Energy minimization framework in 3-Tier Web server farms
  - Web tier (Web servers), application server tier (business logic), and database tier
  - Total 30 machines
  - Industry standard Web benchmark TPC-W

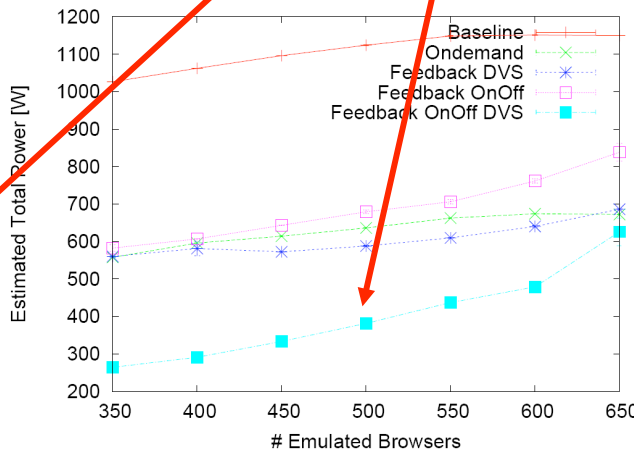
Evaluation

- Comparison with other mechanisms on different DVS settings
  - Baseline
  - Linux Ondemand
  - Feedback DVS
  - Feedback OnOff
  - Feedback OnOff DVS

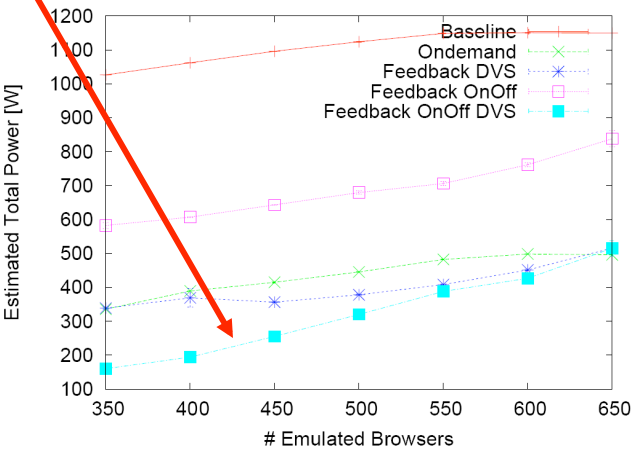
DVS + On/Off with co-adaptation: gives the best performance



δ=0.8, saving from DVS is small



δ=0.5



δ=0.3, saving from DVS is big

Conclusion

- Presented methods for composition of adaptive components
- Adaptation graph analysis to identify incompatibilities
- Co-adaptation design methodology for composition
- Web server farm case-study in the testbed with 30 machines

Questions?