Energy- and Thermal-Aware Scheduling for Heterogeneous Datacenters

Hongyang SUN, Patricia STOLF, Jean-Marc PIERSON, Georges DA COSTA

IRIT, Toulouse

9th Scheduling for Large-Scale Systems Workshop
July 4th, 2014@Lyon, France

Background

- Energy consumption in datacenters has increased significantly over the years
  - Responsible for 1-2% of global energy
  - A large portion is spent on cooling related activities (up to 50%)
- Resource management in datacenters needs
  - Performance-, Energy-, and Thermal-Aware
- “CoolEmAll” (http://www.coolemall.eu/)
  - EU funded project (2011-2014) to design models, tools and algorithms to improve datacenter energy efficiency

Outline

- Cooling and Energy Model for Datacenters
- Hardware Placement
  - Static Server Placement for Minimizing Max. Temperature
- Software Placement
  - Dynamic Job Scheduling for Energy-Performance Tradeoff
- Performance Evaluation
- Conclusion and Future Work

Cooling and Energy Model for Datacenters

Typical Datacenter Layout

- Racks of servers are organized in rows, with alternating cold aisles and hot aisles
- Heat is removed by computer room air conditioning (CRAC) unit, or heating ventilation air conditioner (HVAC)

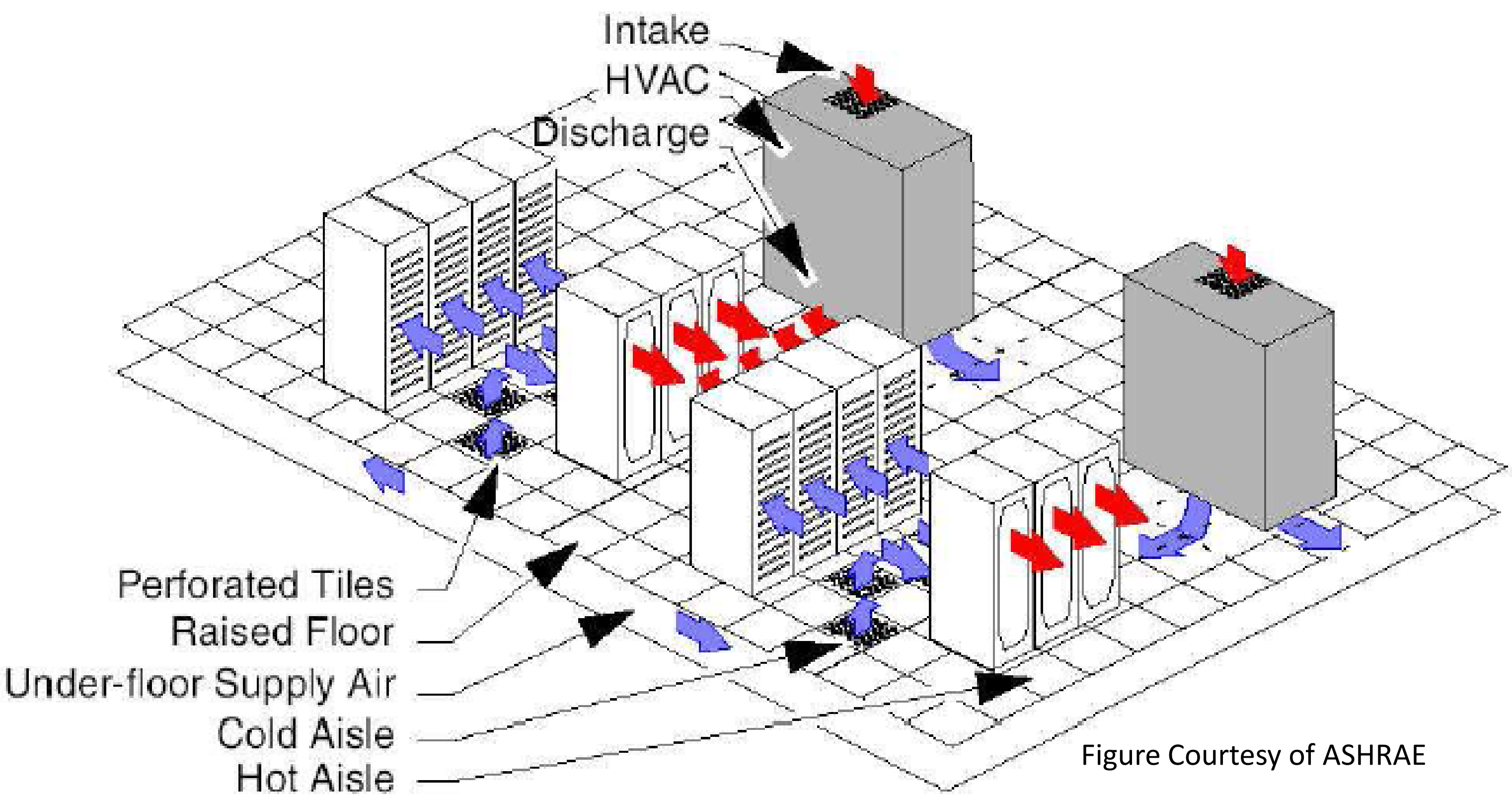
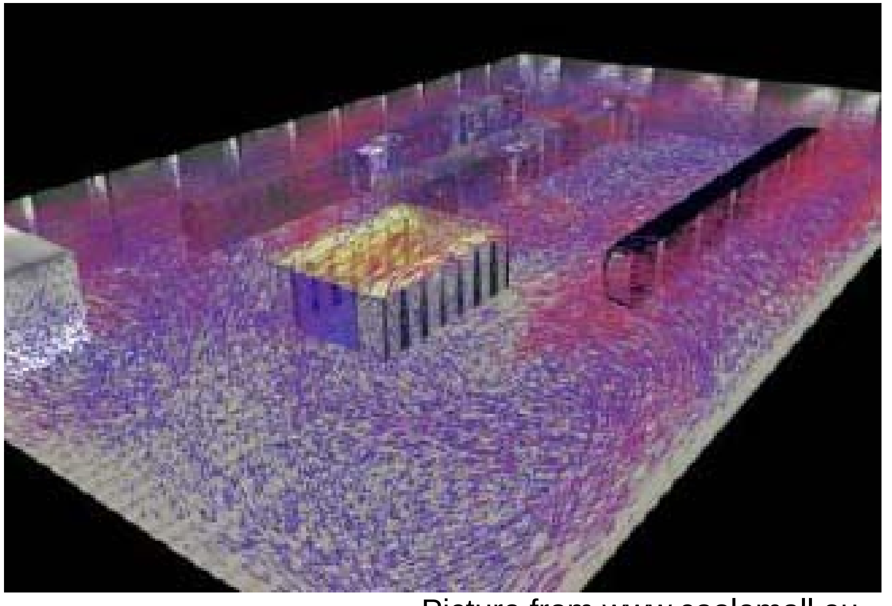


Figure Courtesy of ASHRAE

Heat Recirculation

- Some hot air from the server *outlets* recirculates in the room, raising the temperature of the server *inlets*
- Recirculation is characterized by *heat distribution matrix* **D** [Tang et al. 2008]
  - $d_{j,k}$: temperature increase for the inlet at position *j* per unit of power consumed by the server at position *k*
  - Asymmetric, but relatively stable 🡐 verified by CFD simulations



Picture from www.coolemall.eu

Cooling Model

- Redline temperature T^{red} for the inlet of any server
- CRAC adjusts supply temperature T^{sup} to satisfy the bound

$$T_j^{in}(t) = T^{sup}(t) + \boxed{\sum_{k=1}^{m} d_{j,k} \cdot U_k^{comp}(t)}$$

By heat recirculation

$$T^{sup}(t) = T^{red} - \boxed{\max_{j=1..m} \sum_{k=1}^{m} d_{j,k} \cdot U_k^{comp}(t)}$$

Peak temp. increase

- The cooling cost is related to total power consumption and the supply temp.

$$U^{cool}(t) = \frac{\sum_{j=1}^{m} U_j^{comp}(t)}{\mathrm{CoP}(T^{sup}(t))}$$

- CoP (Coefficient of performance) is defined as the ratio of heat to be removed to energy consumed for cooling
  - Increasing (super-linear) function of supply temp.

Energy Model

- The total energy consumption over interval [t_1, t_2]
  - Due to computing $$E_{comp} = \int_{t_1}^{t_2} \sum_{j=1}^{m} U_j^{comp}(t)dt$$
  - Due to cooling $$E_{cool} = \int_{t_1}^{t_2} U^{cool}(t)dt$$
- To reduce the total energy consumption
  - Reduce the computing energy
  - Reduce the cooling energy
    - 🡐 Raise supply temperature T^{sup}
      - 🡐 Minimize peak temp. increase due to heat recirculation

Static Server Placement

Problem Statement

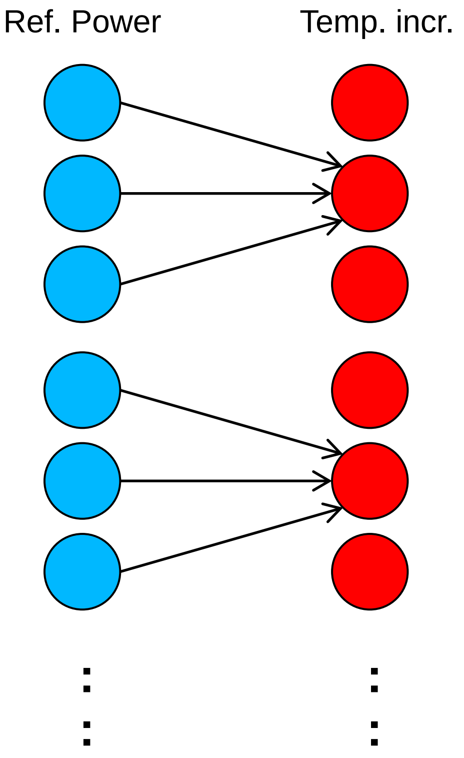
- Input
  - A set of *m* heterogeneous servers, each characterized by a reference power U_j^{ref}, e.g., at average or full load
  - A set of *m* rack slots/positions, characterized by a heat distribution matrix **D**
- Output
  - *One-to-one mapping σ* between servers and slot positions so as to minimize the maximum temperature increase at any server inlet

$$\text{minimize } \max \ \mathbf{D} \cdot \mathbf{U}_\sigma^{ref}$$

$$\mathbf{U}_\sigma^{ref} = [U_{\sigma(1)}^{ref}, U_{\sigma(2)}^{ref}, \cdots, U_{\sigma(m)}^{ref}]^T$$

NP-Hardness Proof

- 3-Partition Problem
  - For a set $S=\{v_1,v_2,\ldots,v_n\}$ of $n = 3k$ positive integers with a total value of kB, can S be partitioned into k subsets $S_1, S_2, \ldots, S_k$ such that the sum of the numbers in each subset is equal (to B)?
  - Remains NP-complete even if each subset is restricted to contain exactly 3 numbers.
- Reduction
  - $m = n = 3k$, $U_j^{ref} = v_j$
  - **D** matrix: every 3 positions contribute only and equally to the temperature increase at one of these positions.
  - Can we achieve a maximum temperature increase of $\sum U_j^{ref}/k = B$?



A Heuristic

- Greedy

  1. Sort the servers by non-increasing reference power

  2. For each server

  3. Assign it to a remaining position that gives the lowest maximum inlet temperature

  4. Update the temperature increase of all inlets

  5. EndFor

- Runtime complexity $O(m^3)$

A Heuristic

- Greedy is *Θ*(*m*)-approximation
  - *O*(*m*): max. temp. contributed by each server < OPT
  - Ω(*m*): $U^{ref} = \{\underbrace{1, 1, \ldots, 1}_{m/2}, \underbrace{\varepsilon, \varepsilon, \ldots, \varepsilon}_{m/2}\}$, $\mathbf{D} = \left.\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ & & \cdots & \\ & & \cdots & \\ 1+\varepsilon & & & \\ & 1+\varepsilon & & \\ & & \cdots & \\ & & & \cdots \end{bmatrix}\right. \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} m/2 \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} m/2 \end{matrix}$

    Greedy > *m/2*, OPT ~= *1+ε*

  - *Any heuristic* is *O*(*Δ*)-approximation, $\Delta = \max U_j^{ref} / \min U_j^{ref}$

Dynamic Job Scheduling

Problem Statement

- Motivated by Online Scheduling for HPC Applications
  - A set of *m* heterogeneous servers (already placed) and heat distribution matrix. Each server has a number of available processors
  - A set of *n* (rigid) parallel jobs arrive over time. Each job has a processor requirement, server-dependent processing time and power consumption
  - Scheduler makes online assignment of jobs to servers, without knowledge of future job arrivals. Processor sharing and job migration are not allowed.
  - Optimize total energy (due to computing and cooling) and/or performance (e.g., average response time)

Scheduling Framework

- Greedy

  1. For each arriving job

  2. Assign it to a server with minimum cost according to some cost function and sufficient remaining processors

  3. If all servers are short of processors, queue the job and reschedule it later when some server becomes free

  4. EndFor

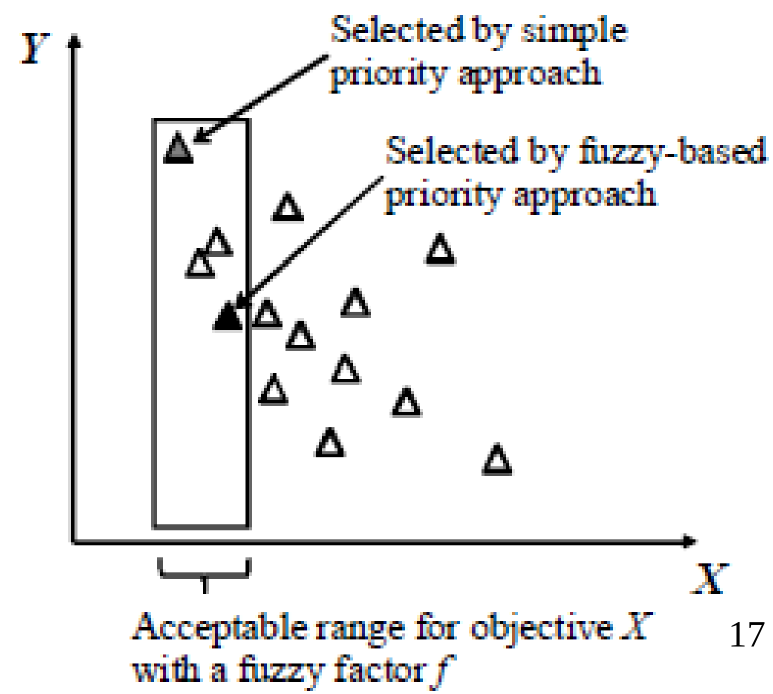
- Different cost functions depending on the objective
  - **Performance-Aware**: cost = response time
  - **Energy-Aware**: cost = energy consumption
  - **Thermal-Aware**: cost = max. inlet temperature

Energy-Performance Tradeoff

- Common Approaches for Two Objectives (e.g., X & Y)
  - **Simple priority**: optimize X first, followed by Y
  - **Constraint optimization**: optimize X subject to a bound on Y
  - **Pareto front**: gives all possible non-dominant solutions
  - **Weighted sum**: optimize $\alpha X + \beta Y$

Energy-Performance Tradeoff

- Common Approaches for Two Objectives (e.g., *X* & *Y*)
  - **Simple priority**: optimize *X* first, followed by *Y*
  - **Constraint optimization**: optimize *X* subject to a bound on *Y*
  - **Pareto front**: gives all possible non-dominant solutions
  - **Weighted sum**: optimize $\alpha X + \beta Y$
- **Fuzzy (Relax) Priority Approach**
  - Optimize *X* followed by *Y*
  - A (fuzzy) factor *f* specifies range for acceptable *X* ; optimize *Y* as long as *X* is acceptable



Energy-Performance Tradeoff

- Fuzzy Priority Rule for Ordering Two Servers

$$H_{i,j_1}^{X,Y} < H_{i,j_2}^{X,Y} \iff$$

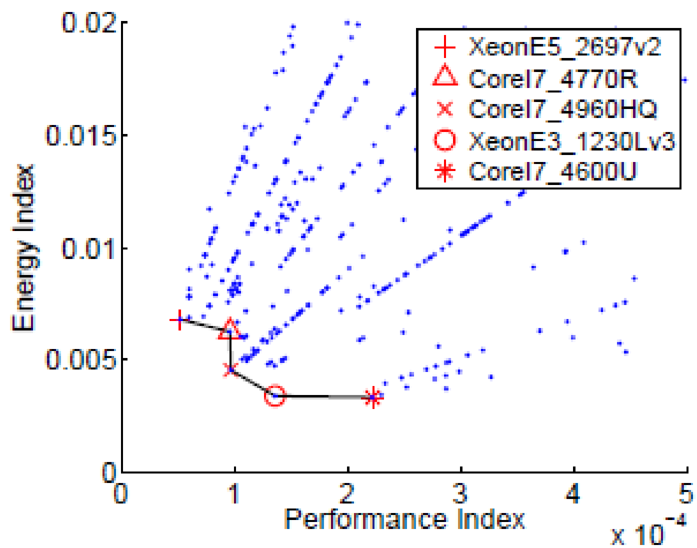
- $\overline{H}_{i,j_1}^X \le f < \overline{H}_{i,j_2}^X$, or
- $\overline{H}_{i,j_1}^X \le f$ and $\overline{H}_{i,j_2}^X \le f$ and $H_{i,j_1}^Y < H_{i,j_2}^Y$, or
- $\overline{H}_{i,j_1}^X < \overline{H}_{i,j_2}^X \le f$ and $H_{i,j_1}^Y = H_{i,j_2}^Y$, or
- $f < \overline{H}_{i,j_1}^X < \overline{H}_{i,j_2}^X$, or
- $f < \overline{H}_{i,j_1}^X = \overline{H}_{i,j_2}^X$ and $H_{i,j_1}^Y < H_{i,j_2}^Y$.

- Can be extended to include more objectives

Performance Evaluation

Simulation Setup

- Small datacenter with 50 servers, each with 18 processors.
- 5 types of processors from Intel, non-dominating in terms of performance and energy
- Heat recirculation matrix is from measurement of a datacenter at ASU [Tang et al. 2008]



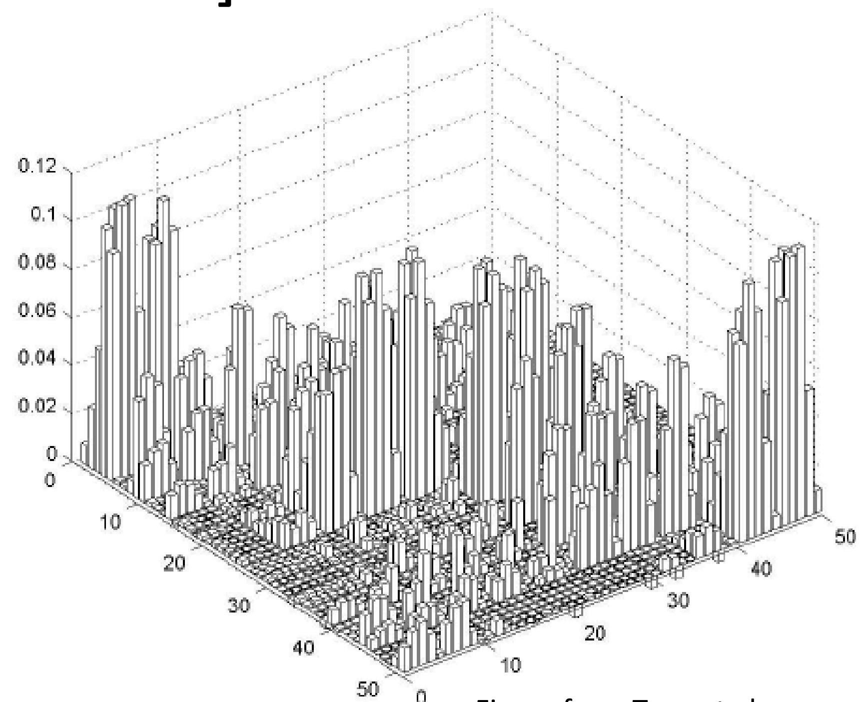


Figure from Tang et al.

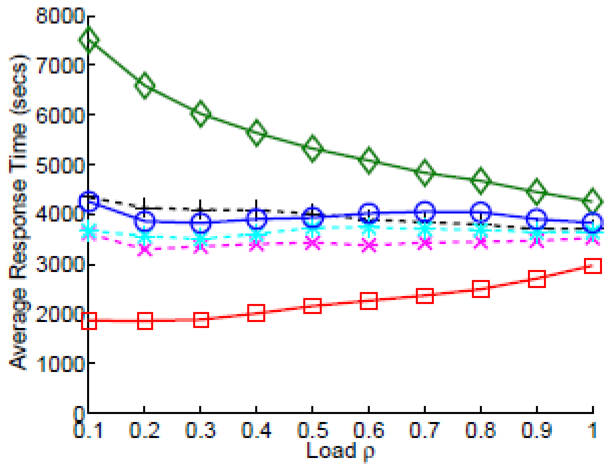
Simulation Setup

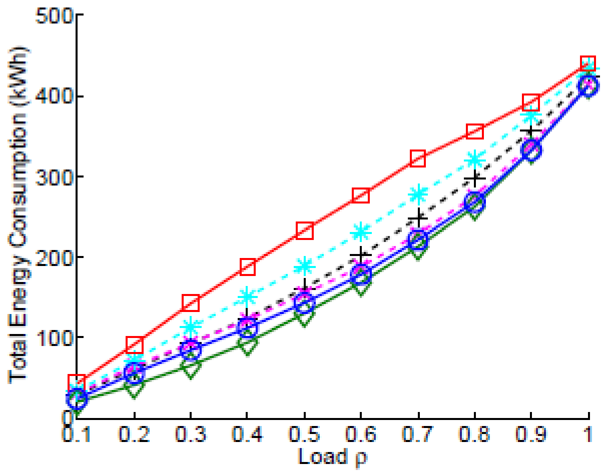
- CoP is from measurement of a water-chilled CRAC [Moore et al. 2005]
  - CoP$(T) = 0.0068T^2 + 0.0008T + 0.458$
  - Workload consists of some HPC apps, e.g., *FFT*, *C-Ray*, *Abinit*, *Linpack*, *Tar*, with profiled time and power info.
- Redline temperature $T^{red} = 25^{o}$C / 77^{o}F
- Simulation is conducted using *Data Center Workload and Resource Management Simulator* (*DCWorms*) [Kurowski et al. 2013]

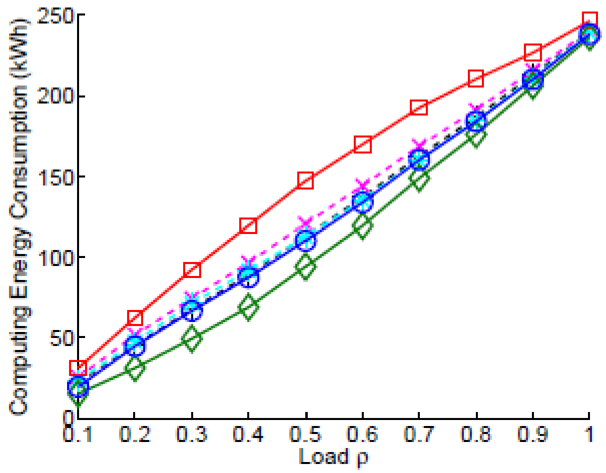
Figures Courtesy of Tang et al. 2008

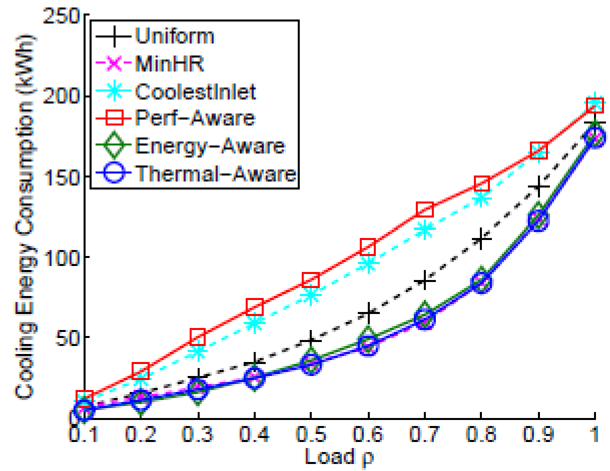
Simulation Results – Job Scheduling

- Heuristics for Single Objective
  - **Perf-**, **Energy-**, and **Thermal-Aware**
  - **Uniform**: Assign jobs randomly/uniformly
  - **CoolestInlet**: Assign jobs to coolest node
  - **MinHR**: Assign jobs to node with least heat recirculation contribution



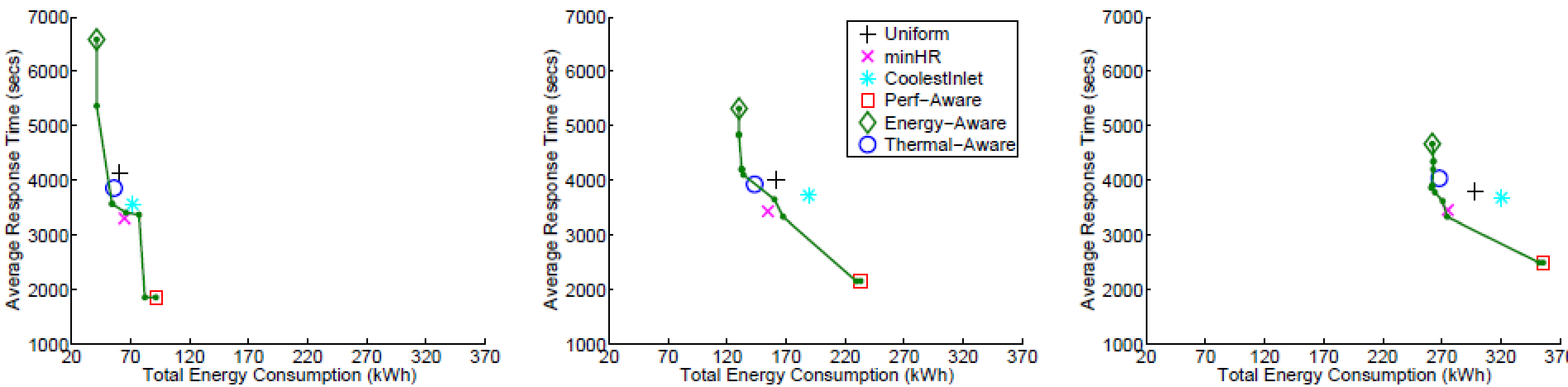






Simulation Results – Job Scheduling

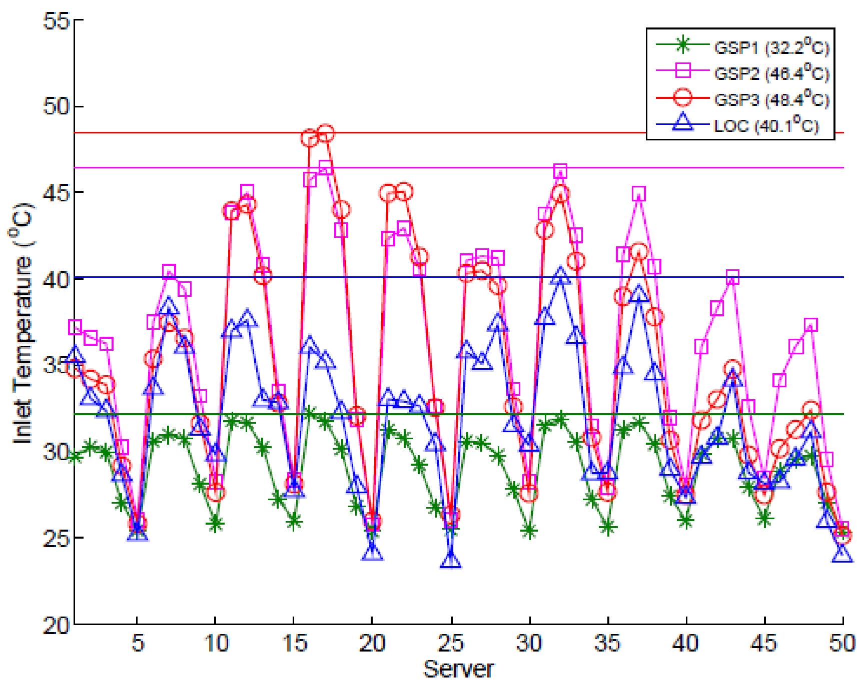
- Energy-performance tradeoff
  - Optimize <energy(*f*), time> and vary fuzzy factor *f* in [*0, 1*]
  - Significant performance gain with little loss in energy
    - 🡐 fuzzy (relaxed) priority



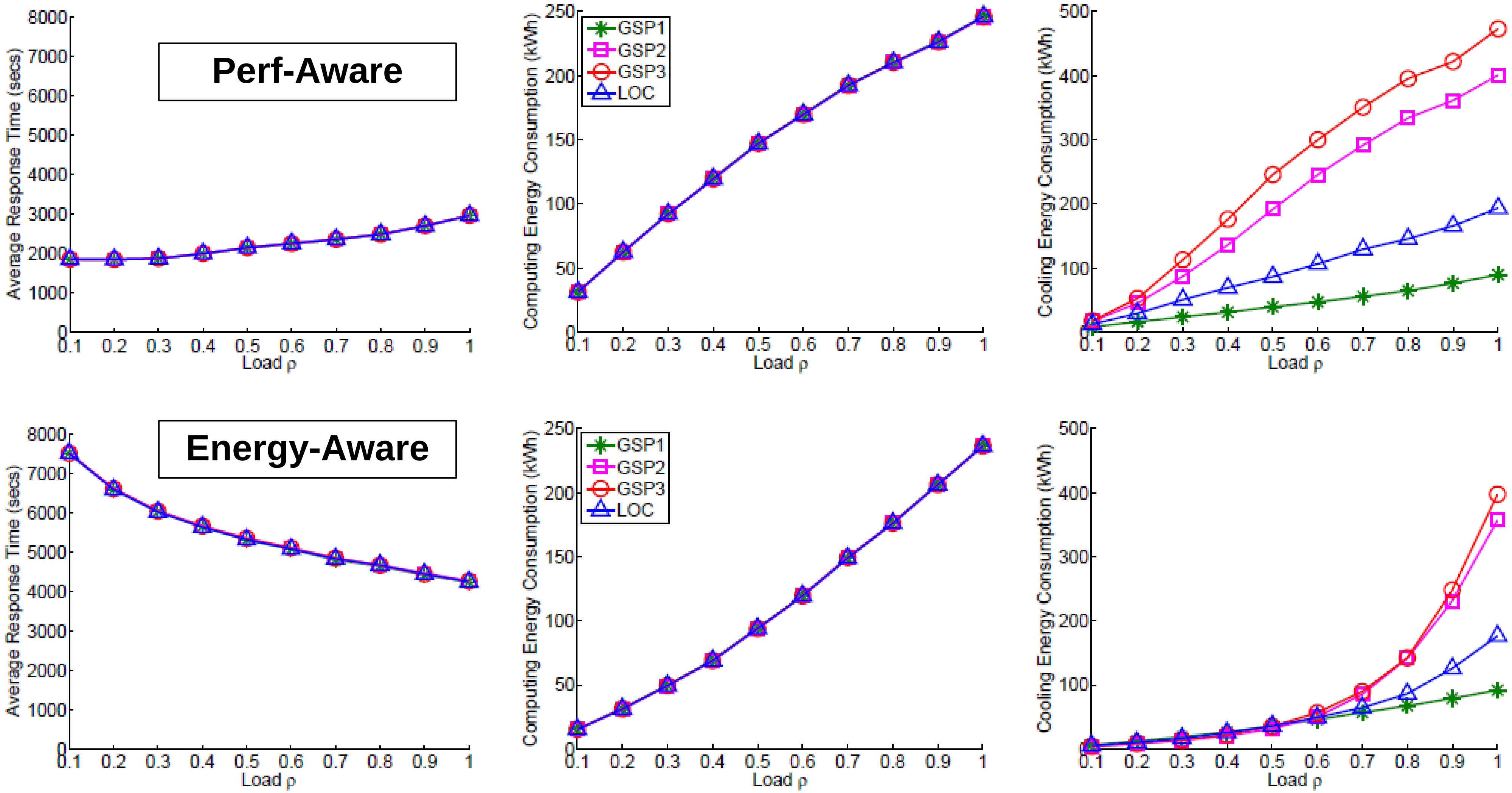
(a) Load $\rho = 0.2$ (b) Load $\rho = 0.5$ (c) Load $\rho = 0.8$

Simulation Results – Server Placement

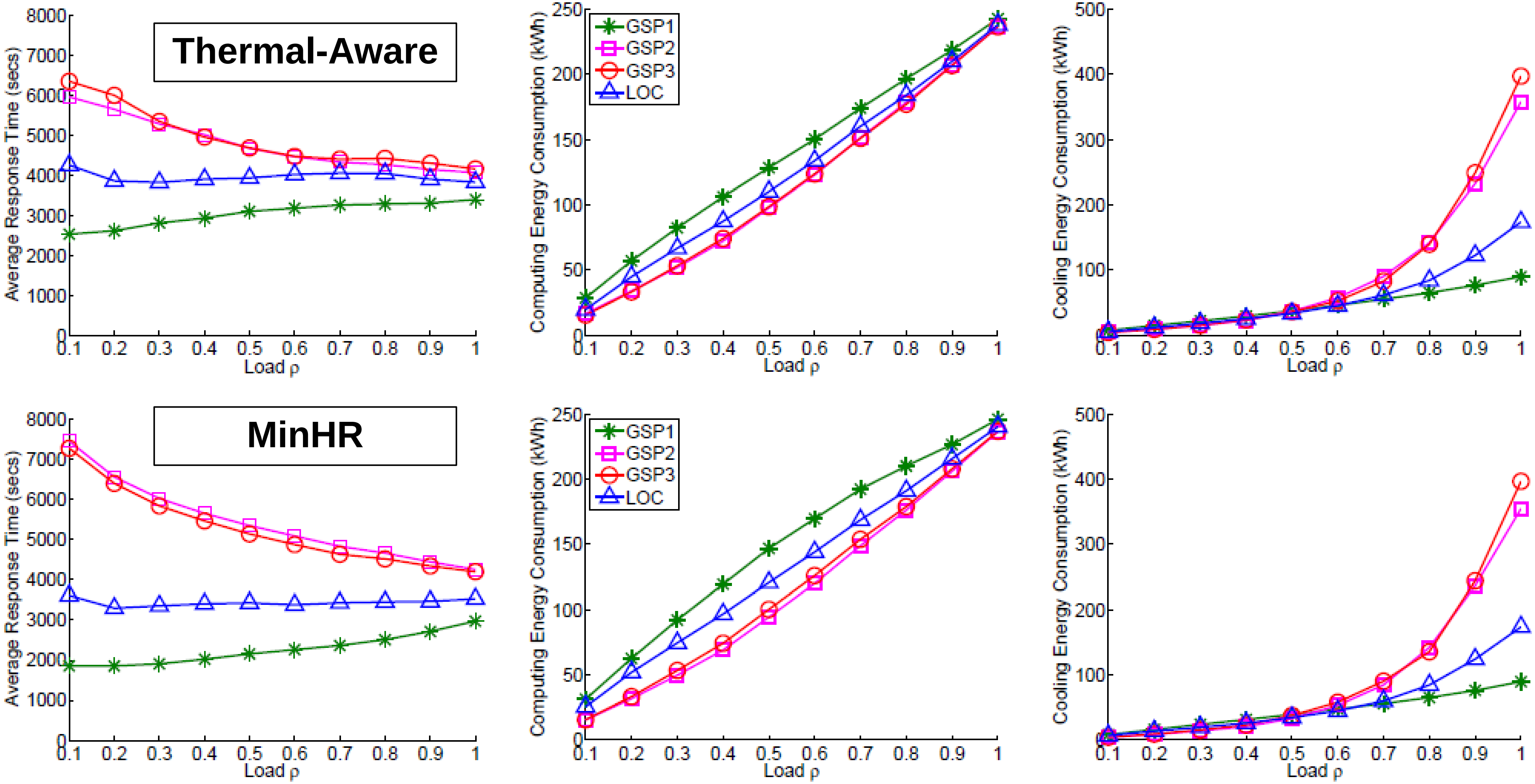
- To illustrate that server placement makes a difference
  - GSP1: Greedy Server Placement as described
  - GSP2: Sort servers in *increasing* power instead of decreasing
  - GSP3: Place servers to *maximize* max. inlet temp. instead of minimize
  - LOC: Place same type of servers in contiguous locations



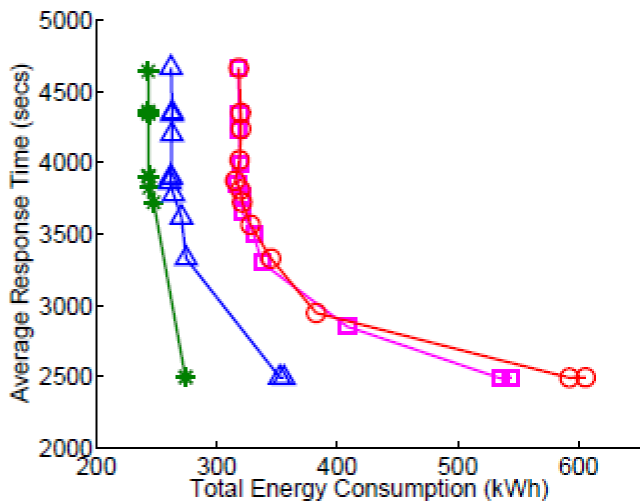
Simulation Results – Server Placement



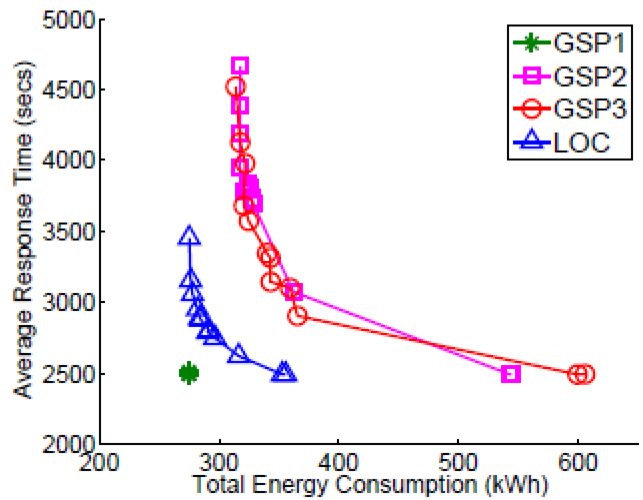
Simulation Results – Server Placement



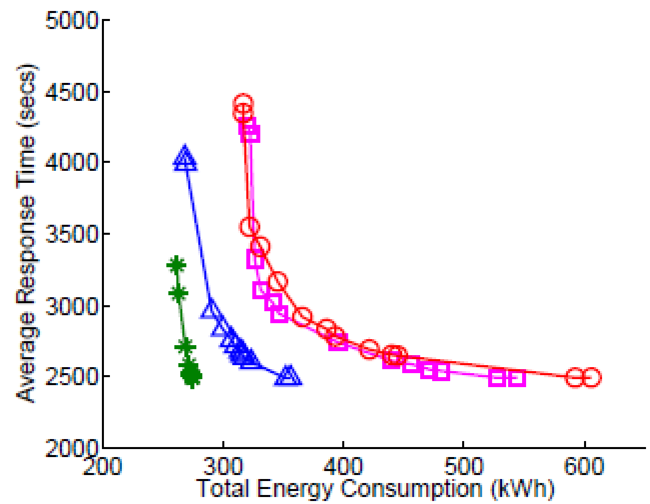
Simulation Results – Server Placement



<energy(f), time>



<HR(f), time>



<temp(f), time>

- Thermal-Aware Server Arrangement
  - (Always) reduces *cooling* energy
  - (Sometimes) introduces tradeoff between performance and *computing* energy
  - Improves overall energy-performance tradeoff

Conclusion and Future Work

- Conclusion
  - Static server placement: NP-hardness, Greedy heuristic
  - Dynamic job scheduling: Greedy framework, Fuzzy (relaxed) priority for energy-performance tradeoff
  - Simulations based on experimentally verified data
- Future Work
  - Static server placement: Better approximation algorithms (LP-based)
  - Dynamic job scheduling: power management techniques, e.g., DVFS, Switch Off