Decentralized allocation of CPU capacity for clustered web services

Shrutivandana Sharma

University of Michigan, Ann Arbor

Asser Tantawi, Malgorzata Steinder, Michael Spreitzer

Service Management Middleware Group, IBM T. J. Watson Research Center

August 22, 2008, Hawthorne, NY, USA

Outline

WebSphere XD - a service management middleware

What is WebSphere XD

- A middleware for quality control of web service provisioning

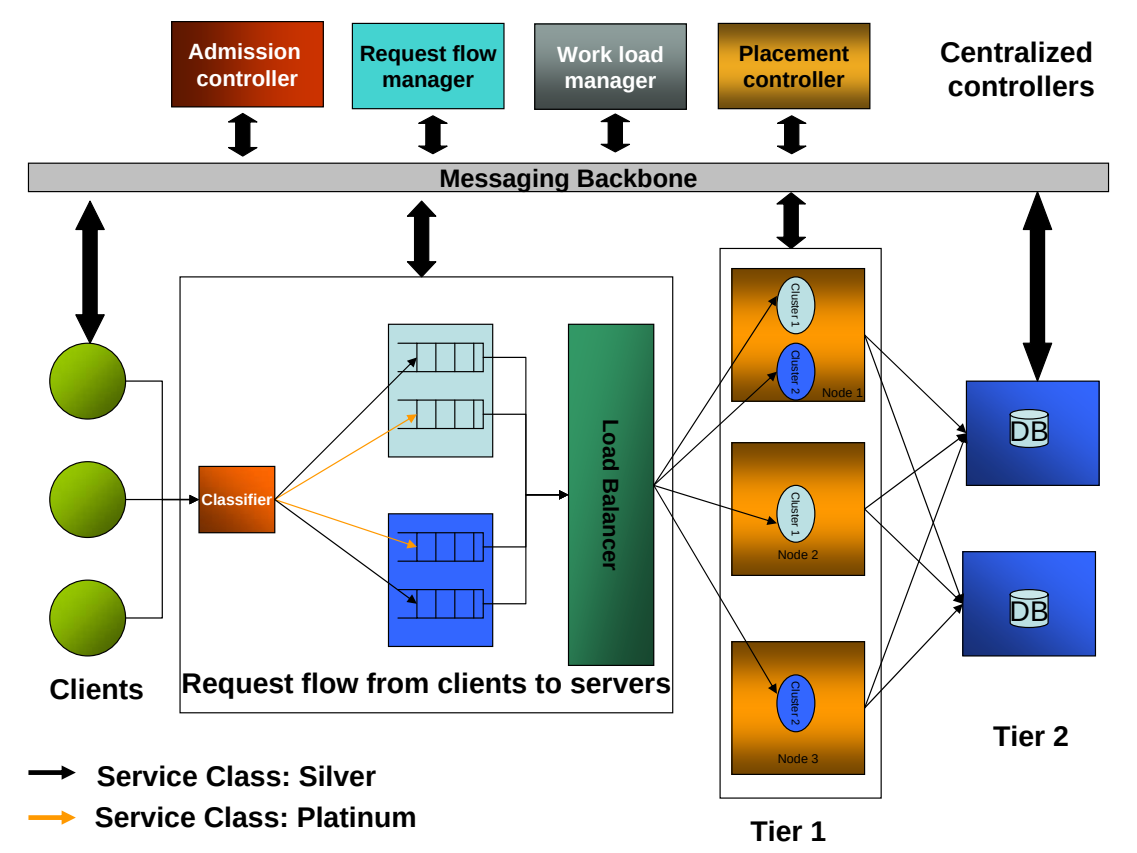
Why is it desirable

- Web applications/services are of numerous types
- Service users have different Quality of Service (QoS) requirements
- Service providers have limited resources

Goal

- Given the available infrastructure,
  - satisfy maximum possible service demand
  - best meet the users' QoS requirements

Websphere XD model



- Various controllers for different tasks –
  admission control, request flow control, workload balancing, placement control
- All controllers are centralized

Motivation for the project

Difficulties with centralized control

- Requires global information of the system
- Not scalable
- One failure can lead to the breakdown of entire system

Decentralized control is desirable

- Each controller would operate with little information
- System size can be scaled
- System is less vulnerable to failure

CPU power allocation: System model

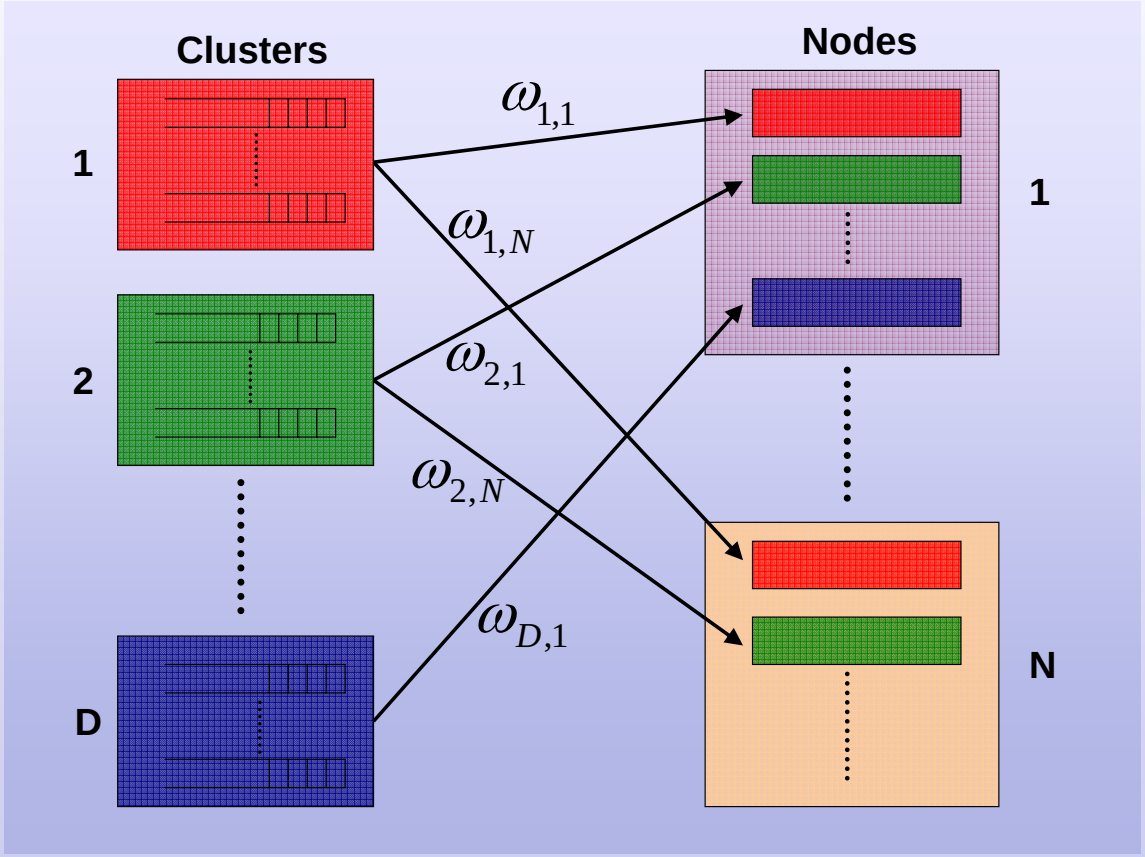


Figure: CPU power allocation for clusters on heterogeneous nodes

- $\omega_{d,n}$ – CPU power for cluster d on node n, $d \in \{1, 2, \ldots, D\}$, $n \in \{1, 2, \ldots, N\}$.
- Each node has a CPU power capacity: Ω_n, $n \in \{1, 2, \ldots, N\}$.
- Each cluster's received QoS is represented by a utility: $U_d(\sum_{n=1}^{N} \omega_{d,n})$

Queueing model for a flow

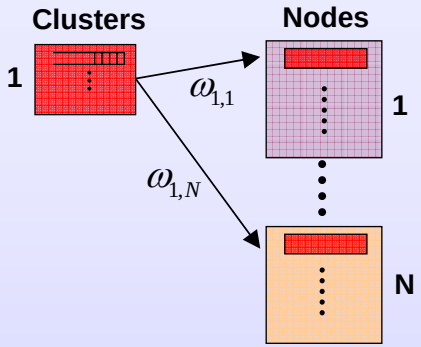


Figure: Queueing model for deriving response time of requests resulting from resource allocation

Queueing model for a flow

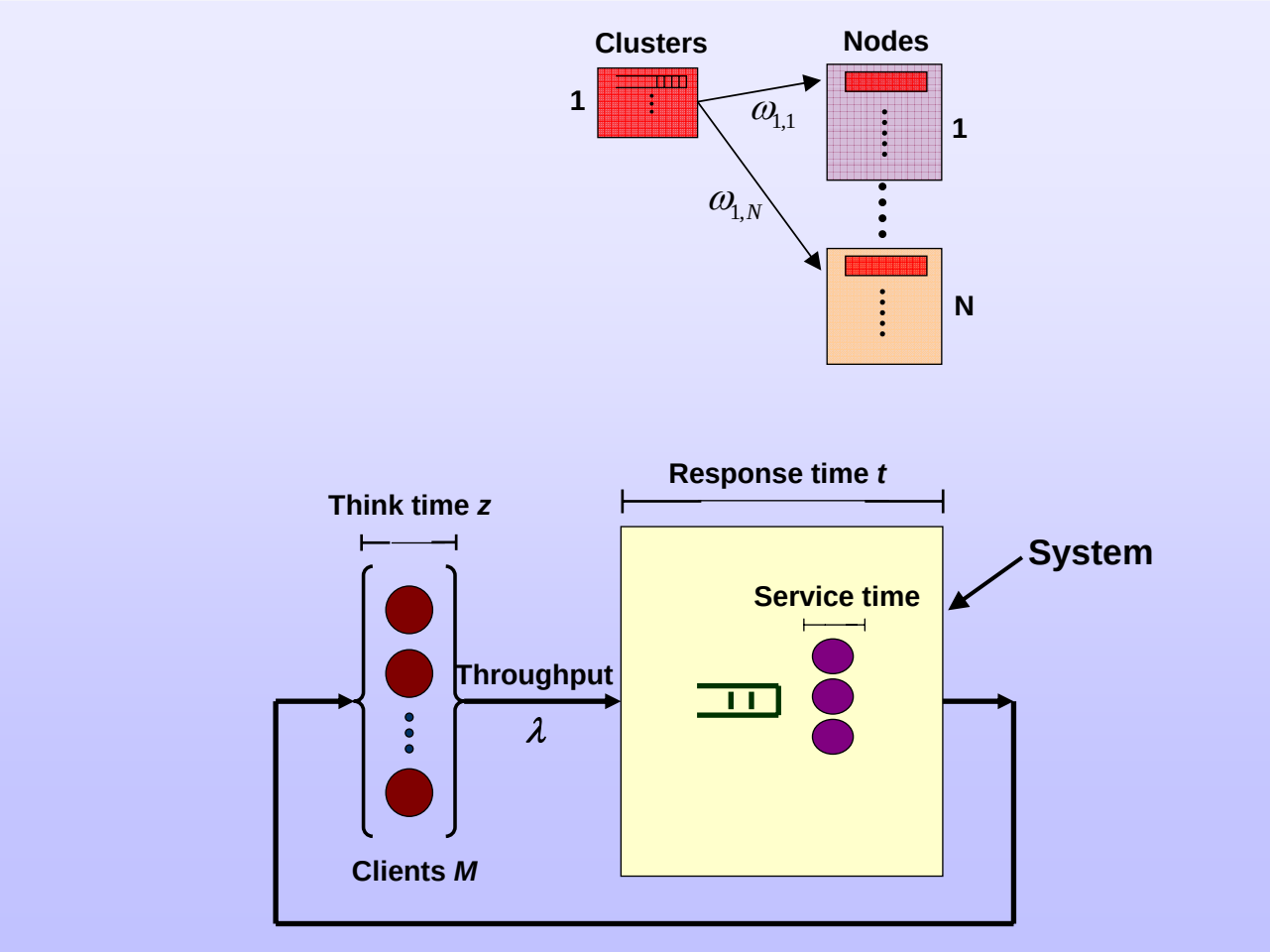


Figure: Queueing model for deriving response time of requests resulting from resource allocation

Queueing model for a flow

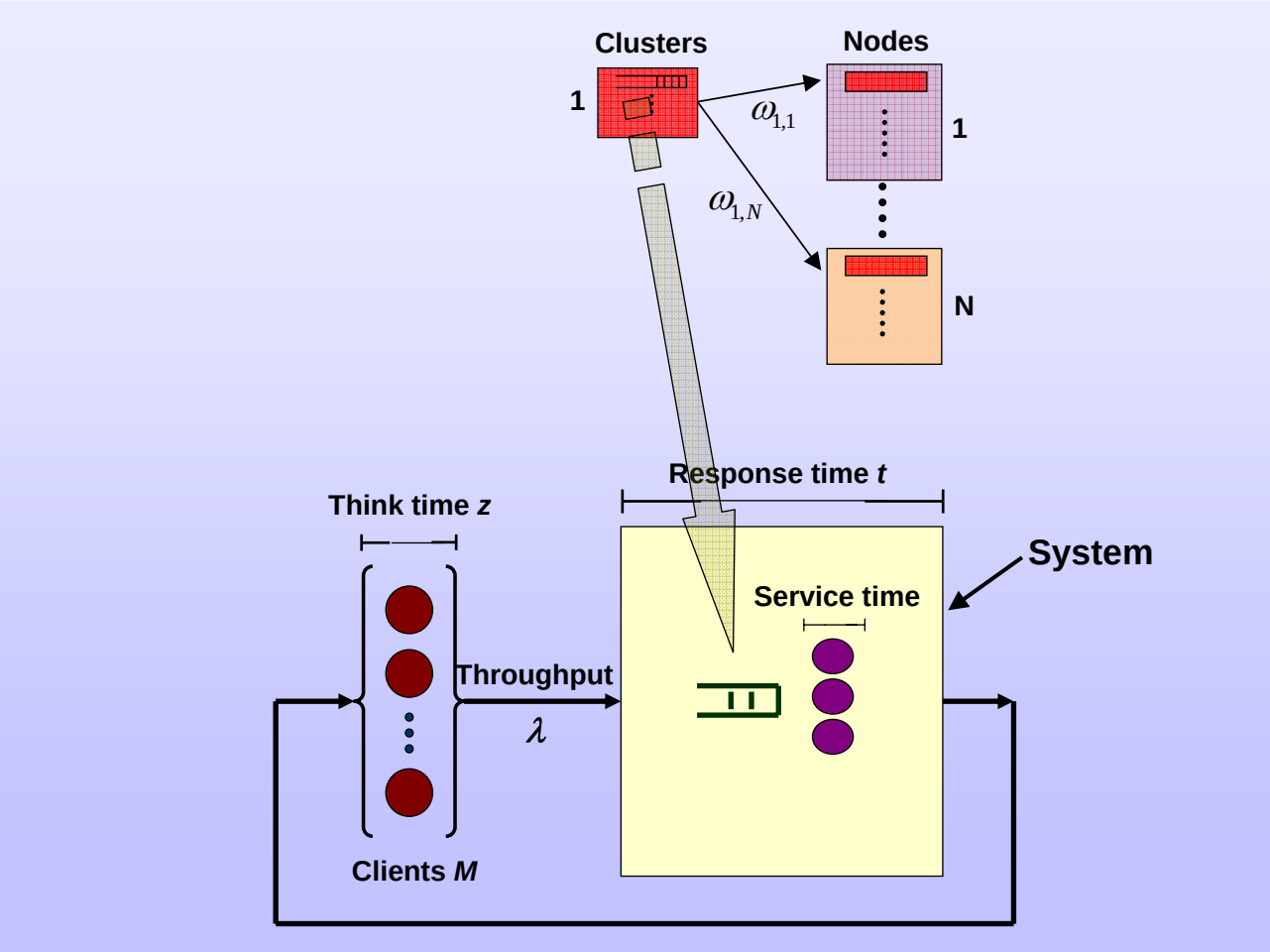


Figure: Queueing model for deriving response time of requests resulting from resource allocation

Queueing model for a flow

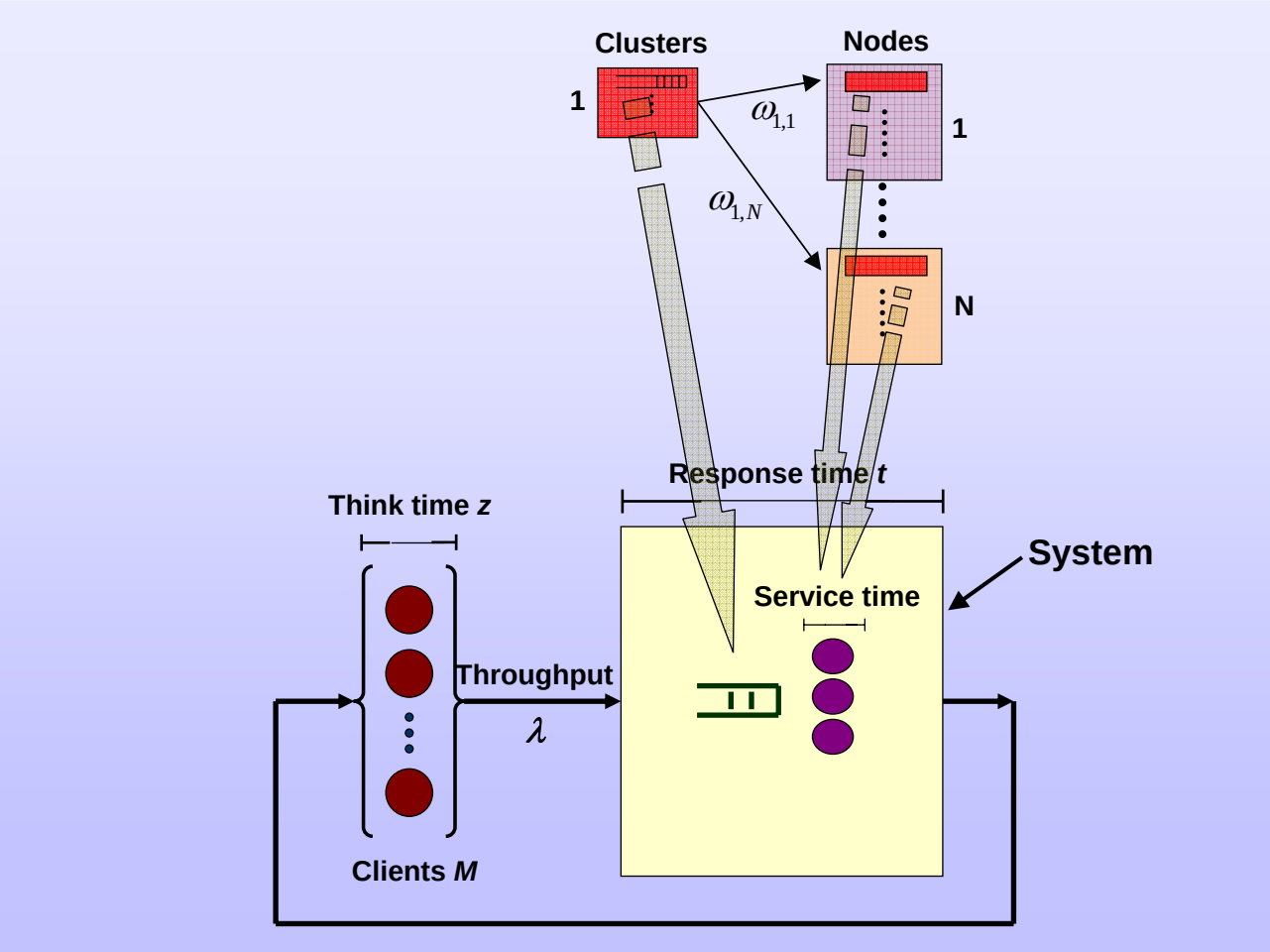


Figure: Queueing model for deriving response time of requests resulting from resource allocation

Queueing model for a flow

- Utility of the flow

$$U(t) = \frac{\tau - t}{\tau}$$

- τ is target response time

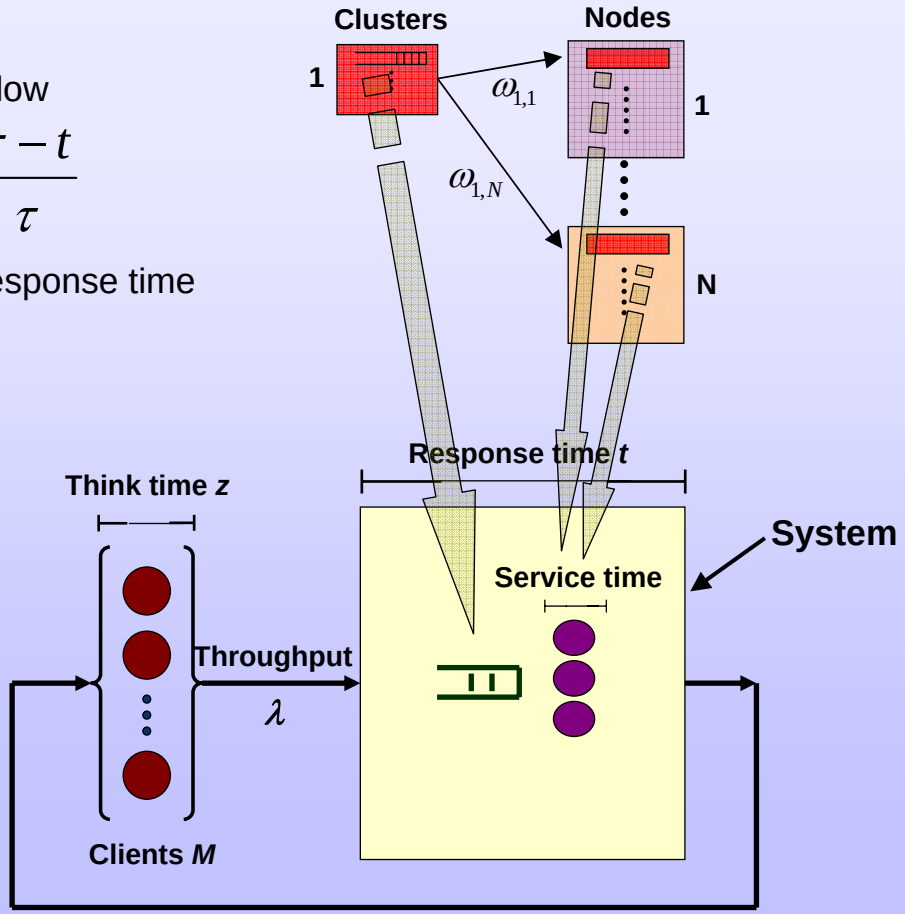


Figure: Queueing model for deriving response time of requests resulting from resource allocation

Utility of a flow as a function of CPU power

- The response time t_f to guarantee utility U for flow f, $f \in \{1, 2, \ldots, F\}$ is,

$$t_f = (1 - U)\tau_f \tag{1}$$

  - Corresponding throughput for flow f is,

$$\lambda_f = \frac{M_f}{(z_f + t_f)} \tag{2}$$

  - CPU power required to sustain throughput λ_f for flow f is,

$$\omega_f = \alpha_f \lambda_f \tag{3}$$

    α_f is the work factor for flow f

- Total CPU power required by cluster d to guarantee utility U for each flow in cluster d is,

$$\omega_d(U) = \sum_{f=1}^{F} \alpha_f \frac{M_f}{(z_f + (1 - U)\tau_f)} \tag{4}$$

- $U_d(\omega)$ is obtained by inverting the function $\omega_d(U)$.
- $U_d(\omega)$ thus obtained is concave in ω.

Optimization problem

Problem (P)

$$\max \quad \sum_{d=1}^{D} U_d \left(\sum_{n=1}^{N} \omega_{d,n} \right) \tag{5}$$

$$\text{s.t.} \quad \sum_{d=1}^{D} \omega_{d,n} \leq \Omega_n, \qquad \forall\, n \in \{1, 2, \ldots, N\} \tag{6}$$

$$0 \leq \omega_{d,n} \leq \overline{\Omega}_{d,n} \quad \forall\, d \in \{1, 2, \ldots, D\} \tag{7}$$

- Centralized solution of Problem (P) requires information of,
  - All capacity constraints in (6)
  - All individual cluster constraints in (7)
  - Utility functions of all the clusters in (5)

Breaking up Problem (P)

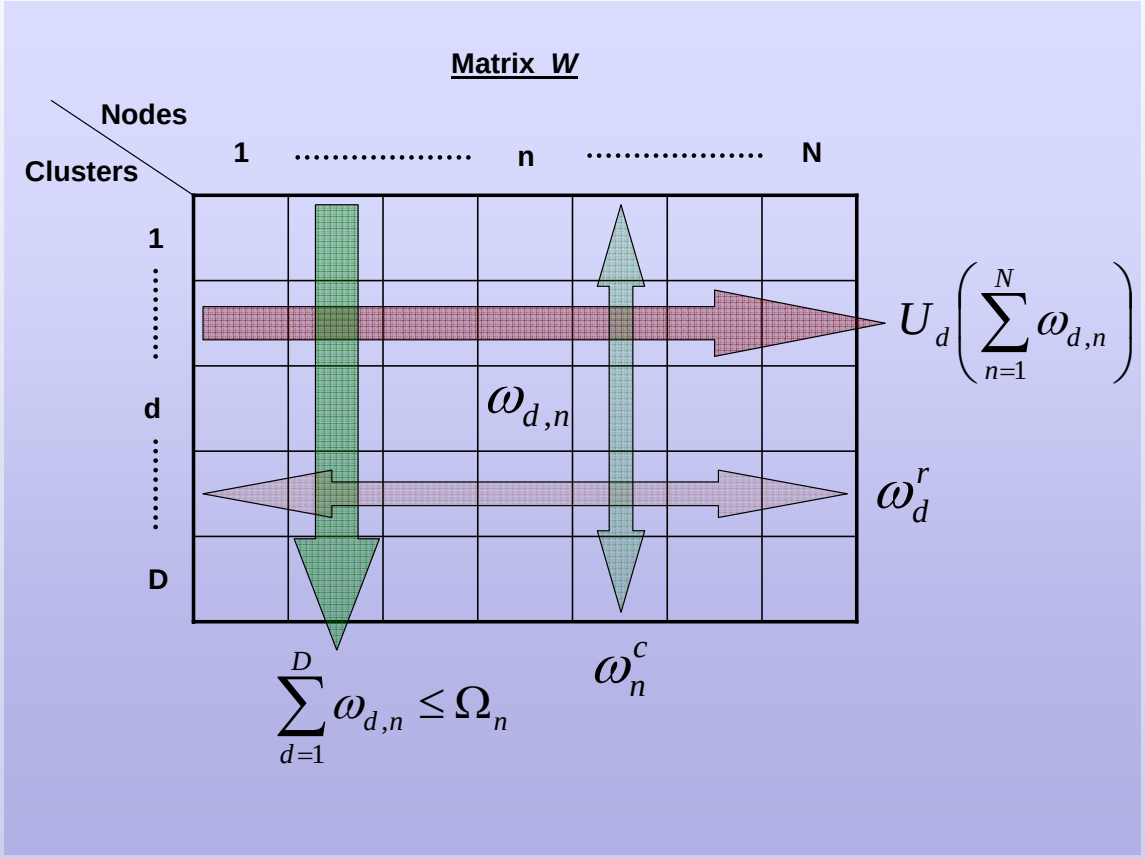


Figure: Variables influencing the clusters and the nodes

- $W = [\omega_{d,n}]_{D\times N}$, matrix of CPU power variables
- $R = [r_{d,n}]_{D\times N}$, row update matrix with rows r_d^r – updated by clusters
- $C = [c_{d,n}]_{D\times N}$, column update matrix with columns c_n^c – updated by nodes

Decentralized algorithm for CPU power allocation

Step 0) Initialization: $k = 0$

- Sequence of penalty parameters $\{\tau^{(t)}\}_{t=1}^{\infty}$ is chosen s.t.

$$0 < \tau(t+1) \leq \tau(t), \qquad \forall\ t \geq 1 \tag{8}$$

$$\lim_{t \to \infty} \tau(t) = 0 \tag{9}$$

$$\lim_{k \to \infty} \sigma^{(k)} = \lim_{k \to \infty} \sum_{t=1}^{k} \tau(t) = \infty \tag{10}$$

- $\overline{W}^{(k)} = [0]_{D \times N}$

Step 1) Row and Column updates:

$$r_d^r(k+1) = \underset{\omega_d^r\,:\,0 \leq \omega_d^r \leq \overline{\Omega}_d^r}{\arg\max} U_d\Big(\sum_{n=1}^{N} \omega_{d,n}\Big) - \frac{1}{\tau^{(k+1)}} \|\omega_d^r - \overline{\omega}_d^r(k)\|^2 \tag{11}$$

$$c_n^c(k+1) = \underset{\omega_n^c\,:\,0 \leq \omega_n^c,\ \sum_{d=1}^{D} \omega_{d,n} \leq \Omega_n}{\arg\max} 0 - \frac{1}{\tau^{(k+1)}} \|\omega_n^c - \overline{\omega}_n^c(k)\|^2 \tag{12}$$

$$\tag{13}$$

Matrix R is sent to the nodes and matrix C is sent to the clusters.

An example with two clusters and two nodes

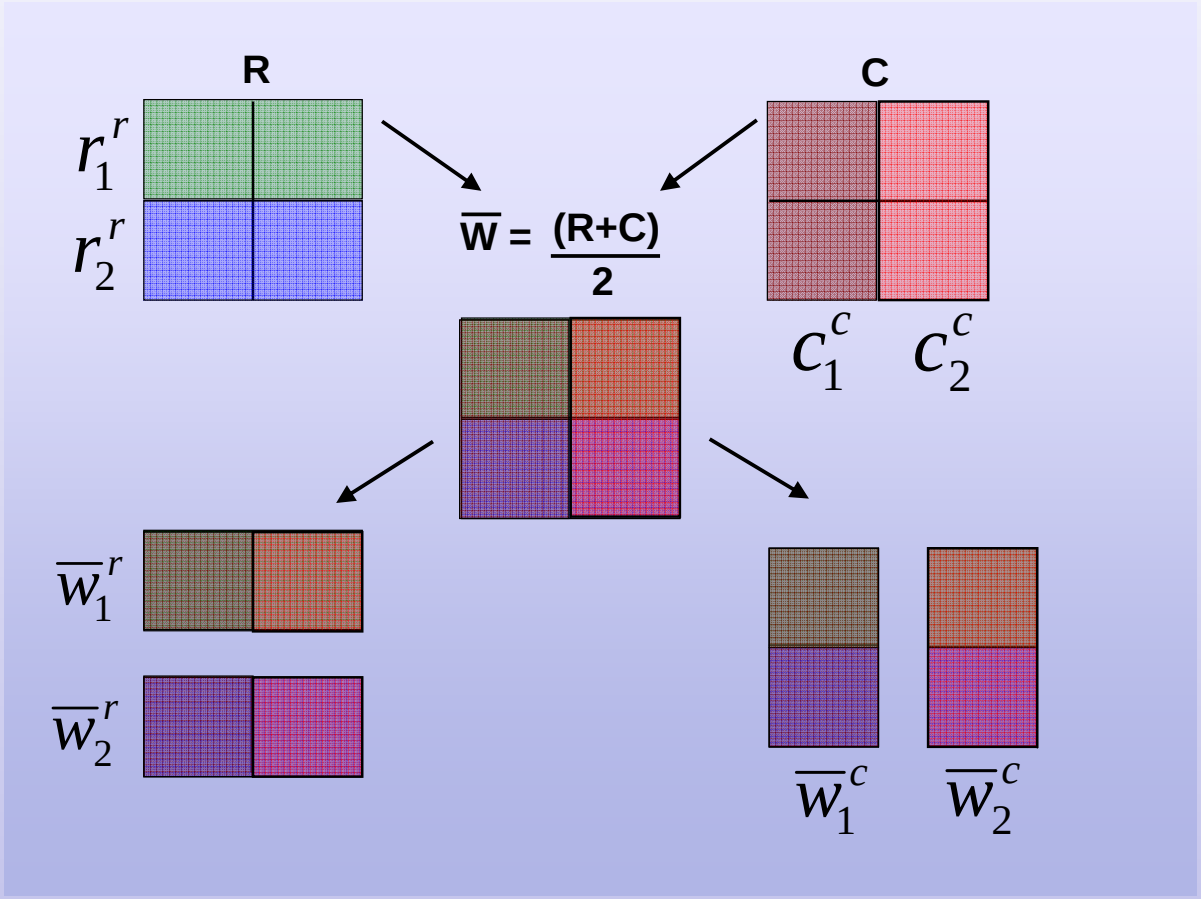


Figure: Row and column updates by the clusters and the nodes

Progression of decentralized algorithm

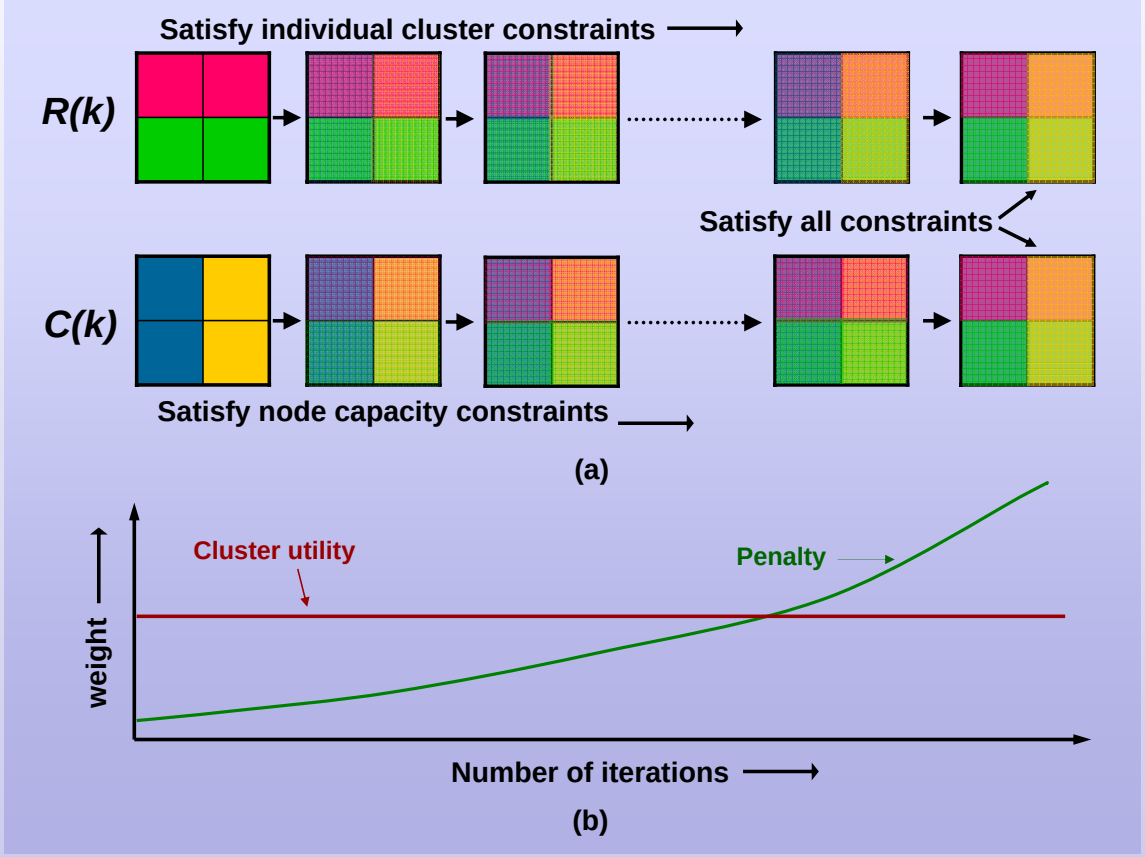


Figure:

(a) Sequences $R(k)$ and $C(k)$ of row and column update matrices

(b) The weight of cluster utility and penalty terms in cluster optimization

Decentralized algorithm for CPU power allocation (cont')

Step 2) After receiving the updates from the nodes (respectively the clusters), the clusters (respectively the nodes) calculate the following averages.

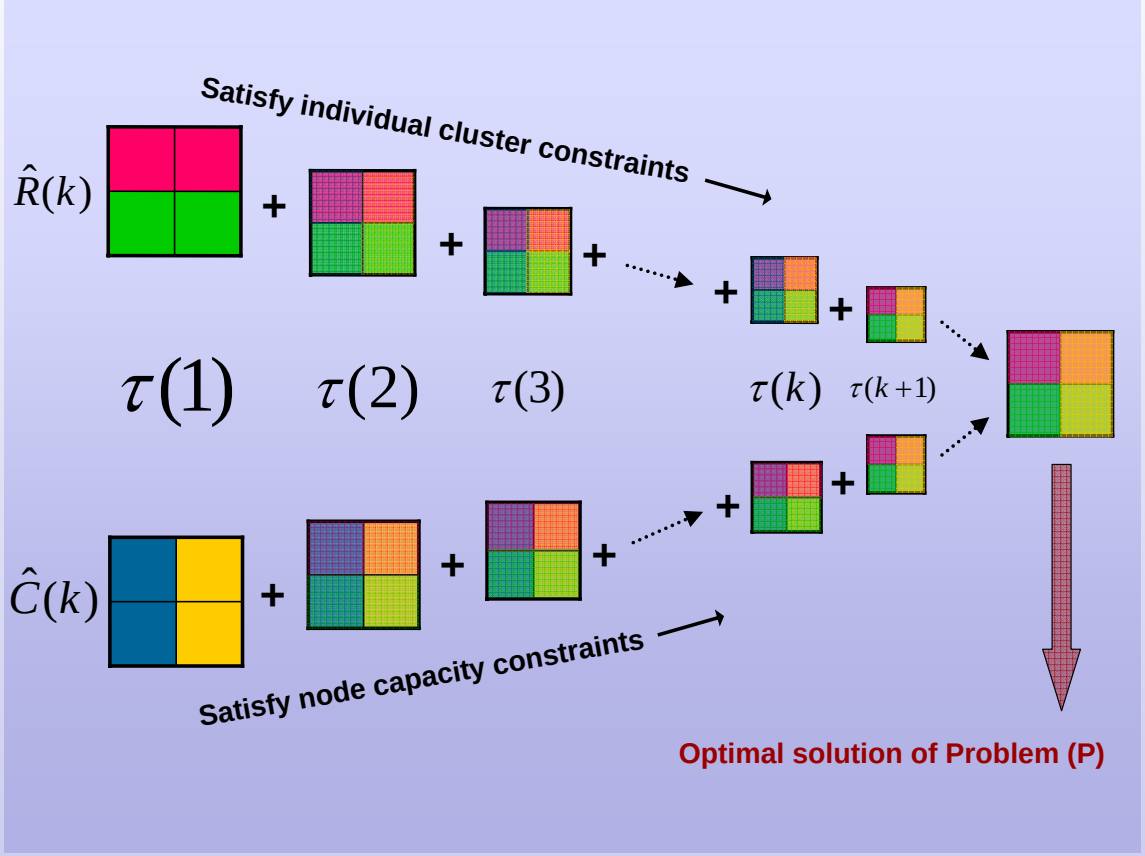
$$\overline{W}(k+1) = \frac{1}{2}[R(k+1) + C(k+1)] \quad (14)$$

$$\widehat{R}(k+1) = \frac{1}{\sum_{t=1}^{k+1}\tau(t)} \sum_{t=1}^{k+1} \tau(t)R(t) \quad (15)$$

$$\widehat{C}(k+1) = \frac{1}{\sum_{t=1}^{k+1}\tau(t)} \sum_{t=1}^{k+1} \tau(t)C(t) \quad (16)$$

$$\widehat{W}(k+1) = \frac{1}{\sum_{t=1}^{k+1}\tau(t)} \sum_{t=0}^{k} \tau(t+1)\overline{W}(t) \quad (17)$$

Time averaging of row and column update matrices



- Time average sequences lie in the respective convex and compact sets, therefore obey the respective constraints.
- Time average sequences converge to the same matrix
- The matrix of convergence is a feasible solution of Problem (P).

Result

Theorem

The sequences $\{\widehat{R}^{(k)}\}_{k=1}^{\infty}, \{\widehat{C}^{(k)}\}_{k=1}^{\infty}$, and $\{\widehat{W}^{(k)}\}_{k=1}^{\infty}$ all converge to the optimum solution of Problem (P).

- The above theorem has been proved using convex analysis.
- Convergence to the optimum solution of Problem P is guaranteed by the decentralized algorithm.

Numerical results

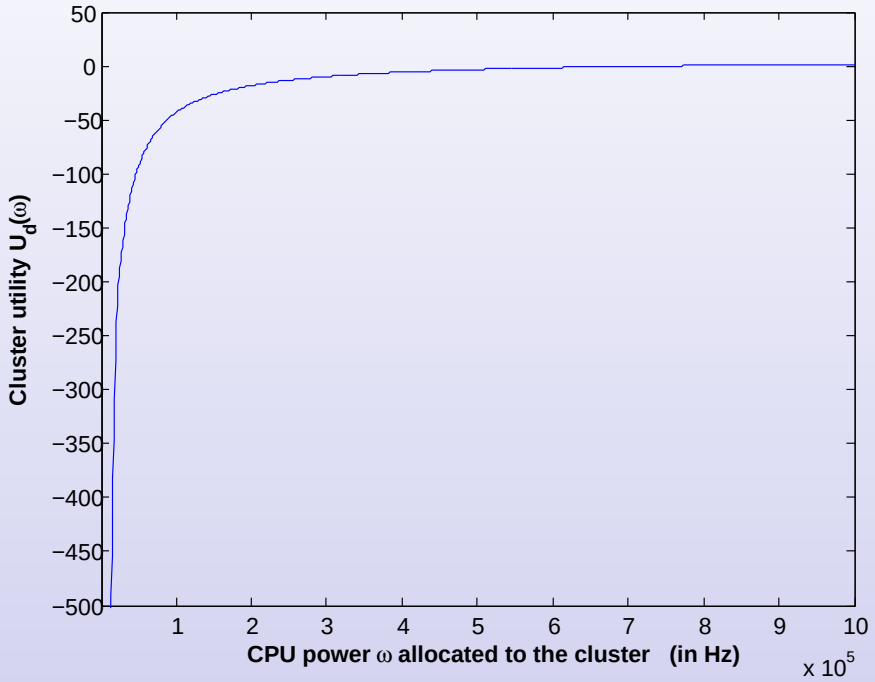
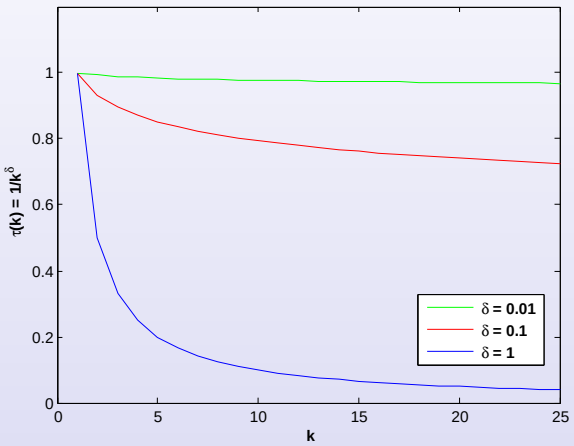


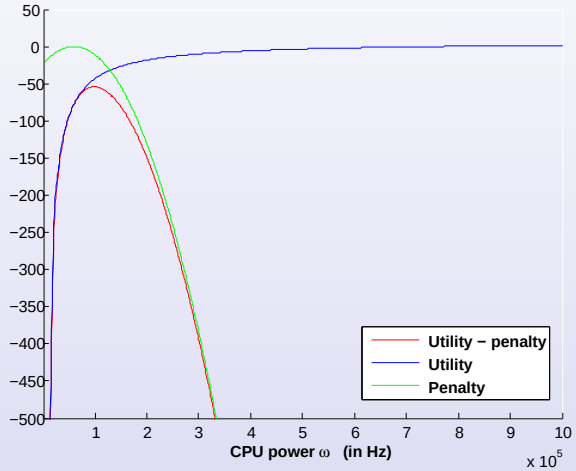
Figure: Utility of a cluster consisting of three flows characterized by the parameters given below.

flow f	–	1	2	3
Work factor α_f (kc)	–	5.648	4.252	4.600
Population M_f	–	6	4	1
Think time Z_f (ms)	–	113.9	109.4	99.9
Target τ_f (ms)	–	10	15	20

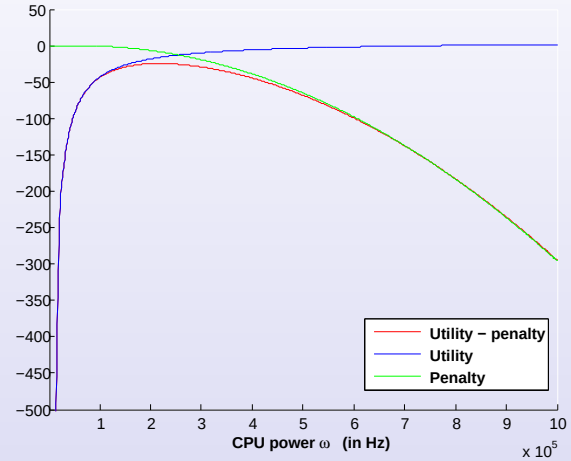
Numerical results: effect of penalty parameter sequence



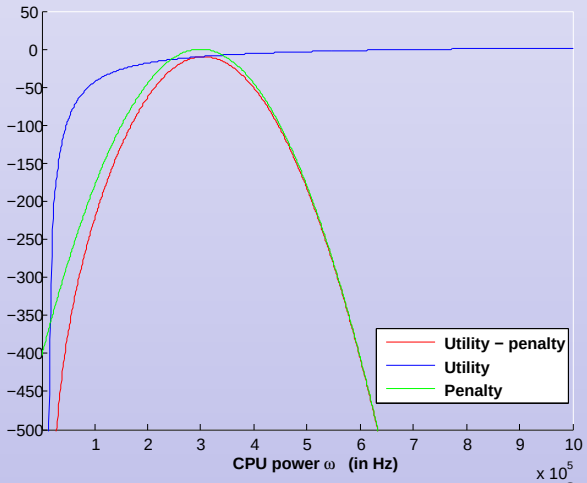
(a) $\tau(k) = 1/k^{\delta}$ vs. k for $\delta = 1, 0.1, 0.001$



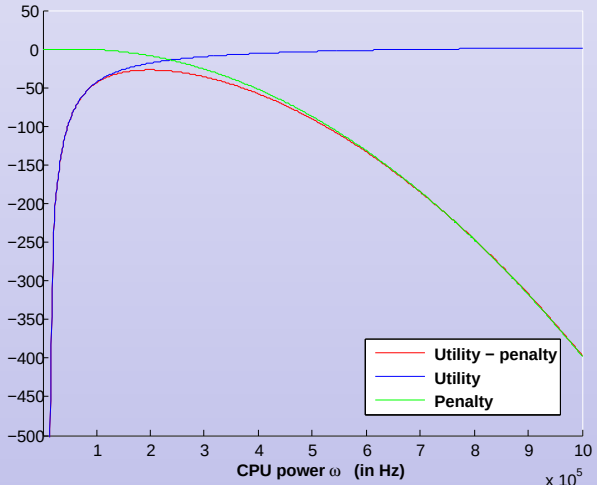
(b) Utility and penalty vs. ω for $\tau(k) = \dfrac{10^9}{k}$



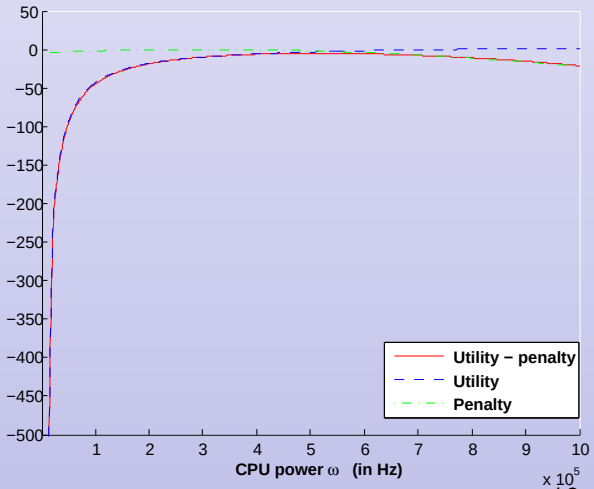
(c) Utility and penalty vs. ω for $\tau(k) = \dfrac{10^9}{k^{0.001}}$



(d) Utility and penalty vs. ω for $\tau(k) = \dfrac{10^8}{k^{0.1}}$

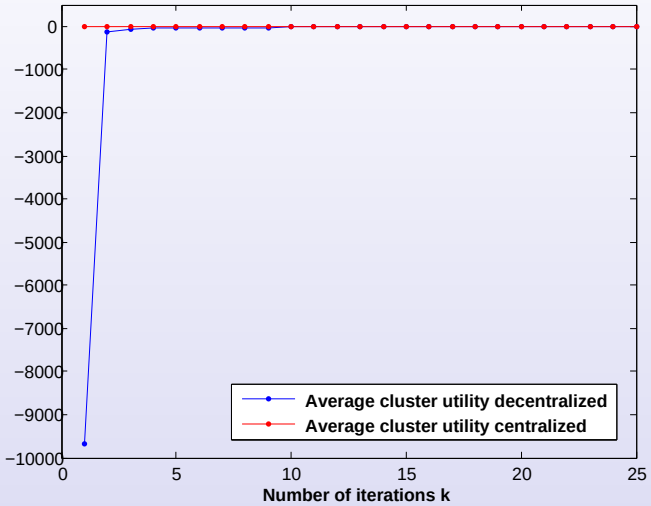


(e) Utility and penalty vs. ω for $\tau(k) = \dfrac{10^9}{k^{0.1}}$

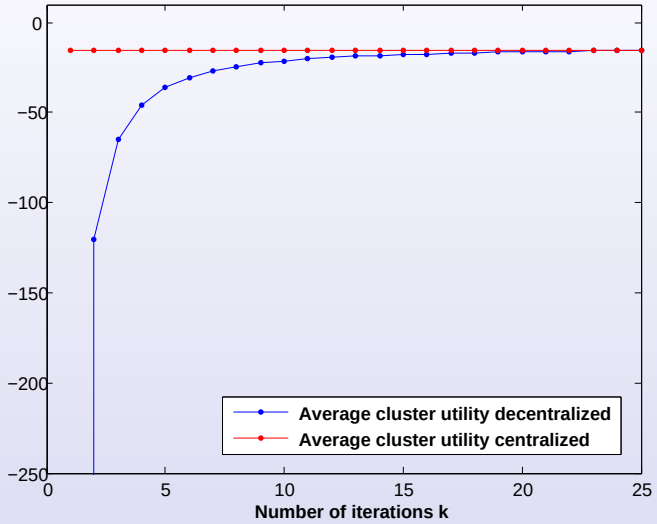


(f) Utility and penalty vs. ω for $\tau(k) = \dfrac{10^{10}}{k^{0.1}}$

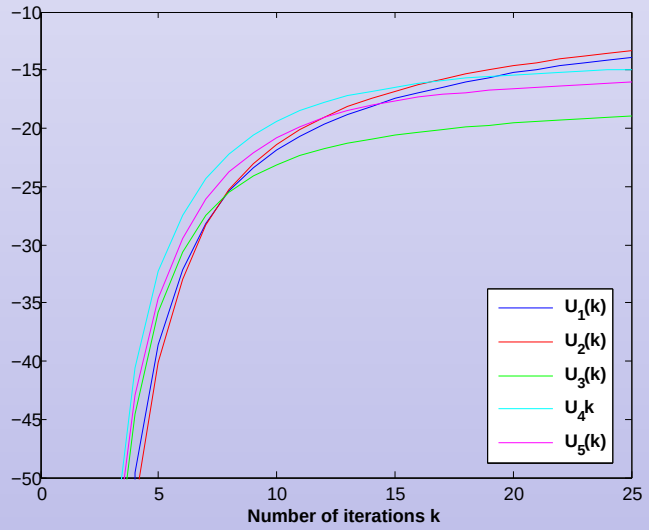
Numerical results: Performance of decentralized algorithm



Average cluster utility, centralized and decentralized, vs. number of iterations, $D = 5$, $N = 3$, $F = 3$

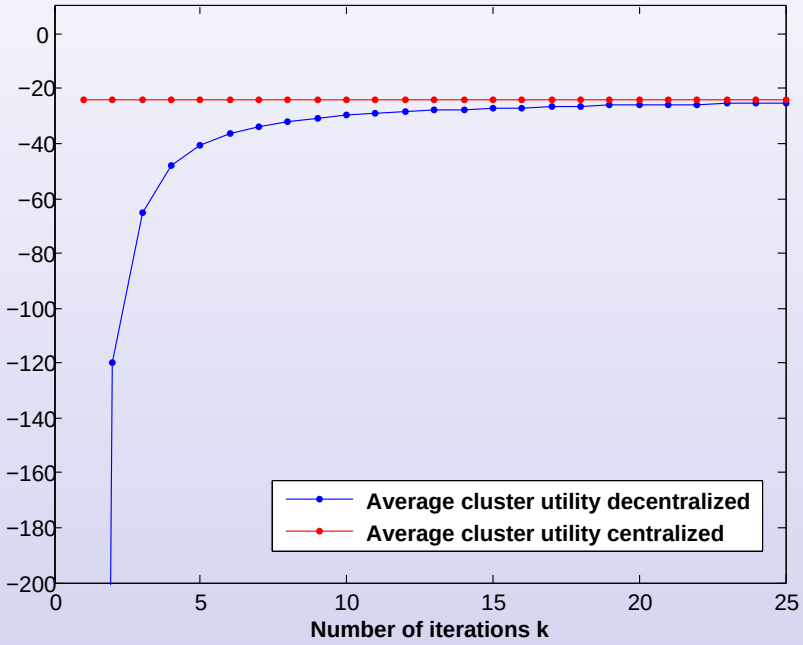


Average cluster utility, centralized and decentralized, vs. number of iterations, $D = 5$, $N = 3$, $F = 3$



Individual cluster utilities from decentralized algorithm vs. number of iterations, $D = 5$, $N = 3$, $F = 3$

Numerical results: Large system



Average cluster utility, centralized and decentralized, vs. number of iterations

System size: $D = 20, N = 12, F = 3$.

Summary

- Decentralized algorithm is designed for CPU power allocation to multiple clusters on heterogenous nodes.
- The optimization/control task is spilt among the clusters and the nodes.
- Each cluster needs to control only the variables that affect its own utility, and each node needs to control only the variables that affect its own utilization.
- The algorithm guarantees convergence to the optimum centralized solution.
- For the test data with five clusters and three nodes, the decentralized algorithm shows fast convergence.
- The algorithm shows similar convergence results when the system size is increased to twenty clusters and twelve nodes.

Summary

- Decentralized algorithm is designed for CPU power allocation to multiple clusters on heterogenous nodes.
- The optimization/control task is spilt among the clusters and the nodes.
- Each cluster needs to control only the variables that affect its own utility, and each node needs to control only the variables that affect its own utilization.
- The algorithm guarantees convergence to the optimum centralized solution.
- For the test data with five clusters and three nodes, the decentralized algorithm shows fast convergence.
- The algorithm shows similar convergence results when the system size is increased to twenty clusters and twelve nodes.
- **It provides a step forward towards scalability of WebSphere XD!**

Extensions of work (in progress)

- Designing decentralized algorithm for CPU power allocation to clusters so as to achieve fair utility distribution.
- Designing decentralized algorithm for CPU resource allocation to clusters so as to achieve fair utility distribution as well as load balancing.

Acknowledgements

- Abhishek Dubey, University of Vanderbilt/Summer Intern, IBM
- Ian Whalley, IBM T. J. Watson Research
- Rahul Jain, IBM T. J. Watson Research

THANKS!

Contact for further details:

- **email:** *svandana@umich.edu*
- **Web:** *http://www.umich.edu/∼svandana*