Workload Capacity Considering NBTI Degradation in Multi-core Systems

Jin Sun[*], Roman Lysecky[*], Karthik Shankar[*], Avinash Kodi[†], Ahmed Louri[*] and Janet M. Wang[*]

[*]ECE Department, University of Arizona, Tucson, AZ 85721, USA

[†]EECS Department, Ohio University, Athens, OH 45701, UDA

E-mail: {sunj, rlysecky, karthik1, louri, wml}@ece.arizona.edu, kodi@ohio.edu

Speaker : Janet M. Wang

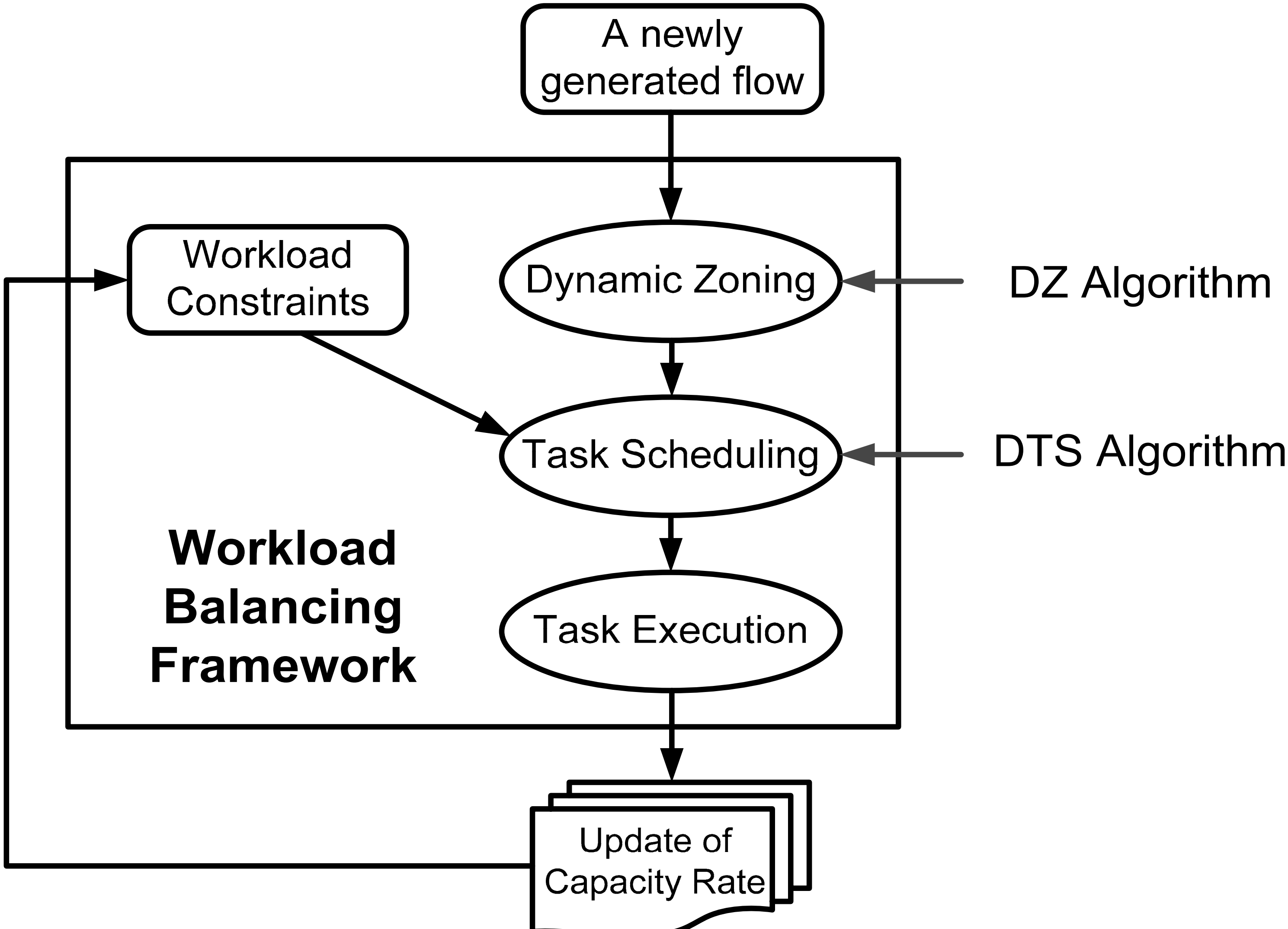
Talk Outline

- Motivation
- Fractional NBTI device model
- NBTI-Aware workload balancing
  - Dynamic Zoning (DZ)
  - Dynamic Task Scheduling (DTS)
  - Dynamic workload balancing
- Experimental Results
  - System performance evaluation
  - Device life-span evaluation
- Conclusions

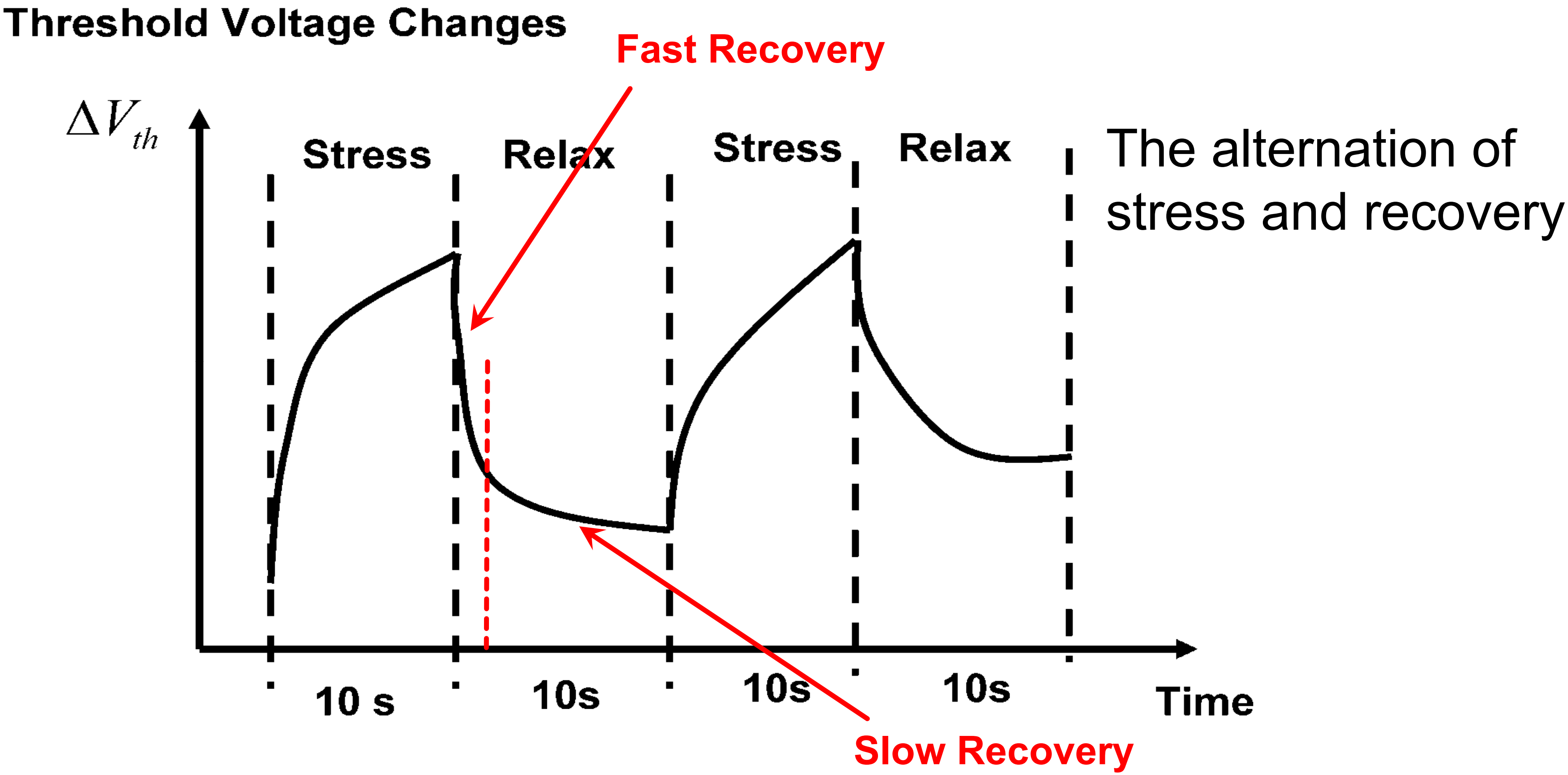
Motivation

- Permanent fault such as Negative Bias Temperature Instability (NBTI) affects system life-span.
- Very little attention has been paid to device stress and its impact.
- A meaningful approach: relaxing cores when they are stressed long before complete wear-out.
  - how to assess a core is stressed
  - how to assign workload to relax stressed cores
  - how to avoid additional performance cost

The General Flow



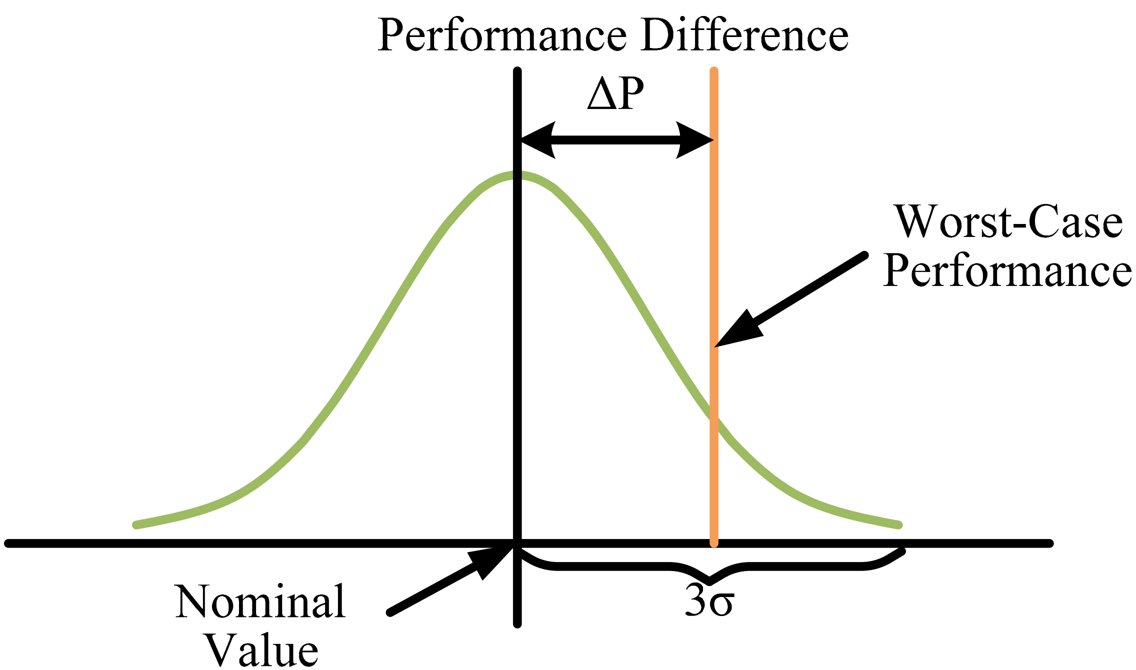
NBTI Concept



- A fast recovery and a slow recovery
- Recovery and stress periods are fairly symmetric.

Fractional NBTI model (1/2)

- We consider temperature changes w.r.t time.
- The changes in V_{th} in turn affect core performance.
- Fractional NBTI device model:



Capacity Rate:

$$CR = 1 - \frac{\Delta P}{3\sigma}$$

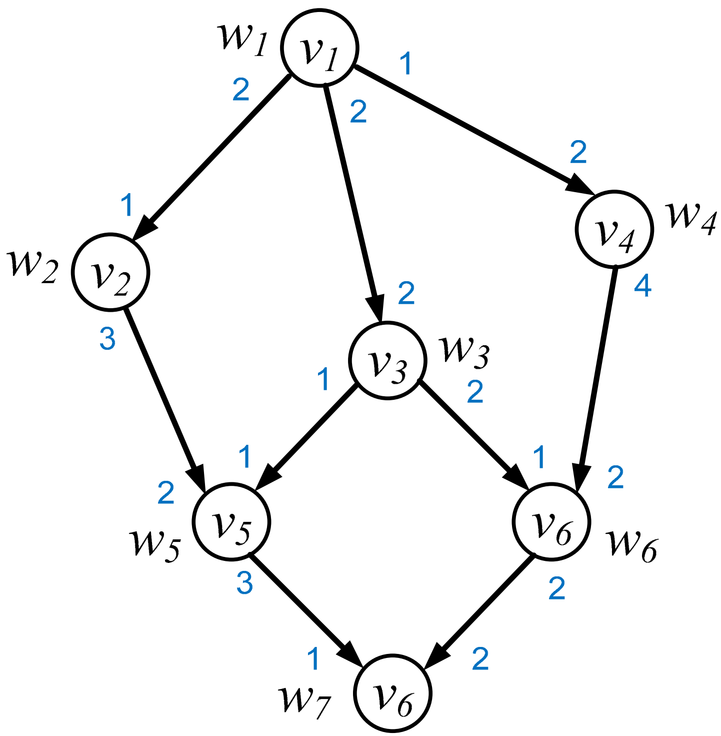
Fractional NBTI model (2/2)

- This model averages leakage and delay impact on core performance.
- We have transferred from single PMOS model to core capacity rate interpretation.
- The capacity rate will be regarded as an upper bound limit for core workload.
- NBTI impact on V_{th} changes with regard to time, so as the capacity rate.

Dynamic Zoning (DZ)

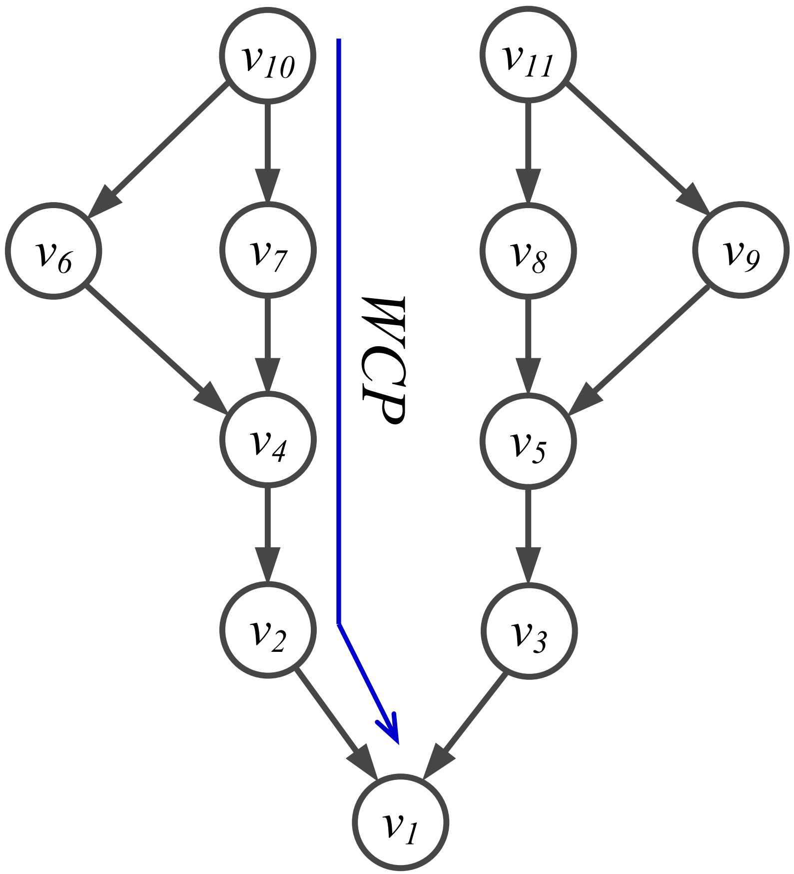
- The process of one task flow is limited to a zone.
- A zone: a group of cores physically adjacent to each other.
- Three factors:
  - size of the zone (number of cores)
  - total Manhattan distances
  - Zone capacity

Task Graph



- Directed Acyclic Graph $G = (V, E)$
  - $V = \{v_1, v_2, \cdots, v_n\}$: a set of tasks to be executed.
  - $E = \{(i, j)\}$: precedence relationships.
  - w_i : task weights
  - The number of tokens consumed (at input) or produced (at output).

Estimated Workload



- Estimated workload:

$$Workload = \frac{\sum_i w_i, \quad v_i \in V}{\sum_j w_j, \quad v_j \in WCP}$$

  - *WCP*: Worst-Case Path

- the minimally required capacity rate for a zone to process it.

$$
\begin{aligned}
&\text{find} && Z_k \\
&\min && \sum_{(i,j)} d(i,j), \quad \forall v_i, v_j \in Z_k \\
&s.t. && \sum_i CR_i \geq Workload, \quad \forall v_i \in Z_k
\end{aligned}
$$

Algorithm 1 Dynamic Zoning

Input: a task flow to be allocated;
a multi-core system with core capacity rate identified

Output: the optimal zoning result to execute the given flow

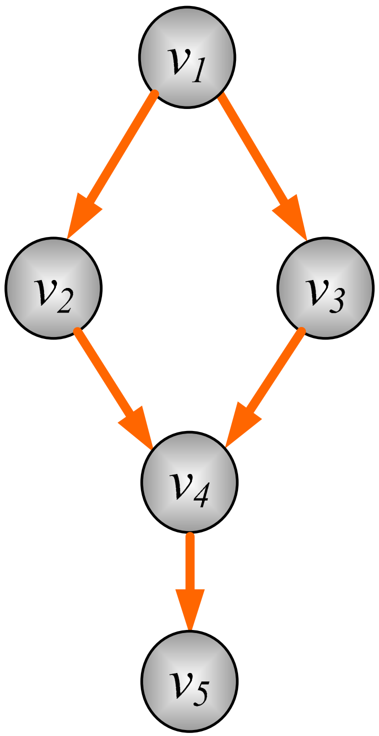
```
S ⇐ Find_Max_Region ()
DZ_opt ⇐ Initial_Rectangle (S)
Dist_opt ⇐ Manhattan_Distances (DZ_opt)
k ⇐ 0
while k < N and Dist_opt > D_th do
    DZ_new ⇐ Perturbation (DZ_opt)
    Dist_new ⇐ Manhattan_Distances (DZ_new)
    if Dist_new < Dist_opt then
        DZ_opt ⇐ DZ_new;  Dist_opt ⇐ Dist_new
    end if
    ΔDist ⇐ Dist_new–Dist_opt
    if exp(–ΔDist/N) > random (0,1) then
        DZ_opt ⇐ DZ_new;  Dist_opt ⇐ Dist_new
    end if
    k ⇐ k+1
end while
return  DZ_opt
```

Dynamic Task Scheduling (DTS)

- Objective:
  - maximizing system utilization
  - minimizing communication cost
- Workload Constraints:
  - Core utilization should not exceed its capacity rate.
- Solution: Mixed Integer Programming (MIP)

Task-Core Mapping Matrix

- Assign tasks to particular cores (within a zone):



Task Graph

	Task 1	Task 2	Task 3	Task 4	Task 5
Core 1	1	0	0	0	0
Core 2	0	0	1	0	1
Core 3	0	1	0	1	0

Binary Mapping Matrix

Decision Variables (1/3)

- A binary matrix M (mxn):
  - m : the number of available cores in a zone
  - n : the number of nodes from the task graph
  - $M_{ij}=1$: task v_i mapped onto j-th core
  - Constraint on M_{ij}'s:

$$\sum_{i=1}^{m} M_{ij} = 1, \quad \forall j = 1, \cdots, n$$

	t1	t2	t3	t4	t5
C1	1	0	0	0	0
C2	0	0	1	0	1
C3	0	1	0	1	0

Decision Variables (2/3)

- Starting time for each task:

$$\{S_1, S_2, \cdots, S_n\}$$

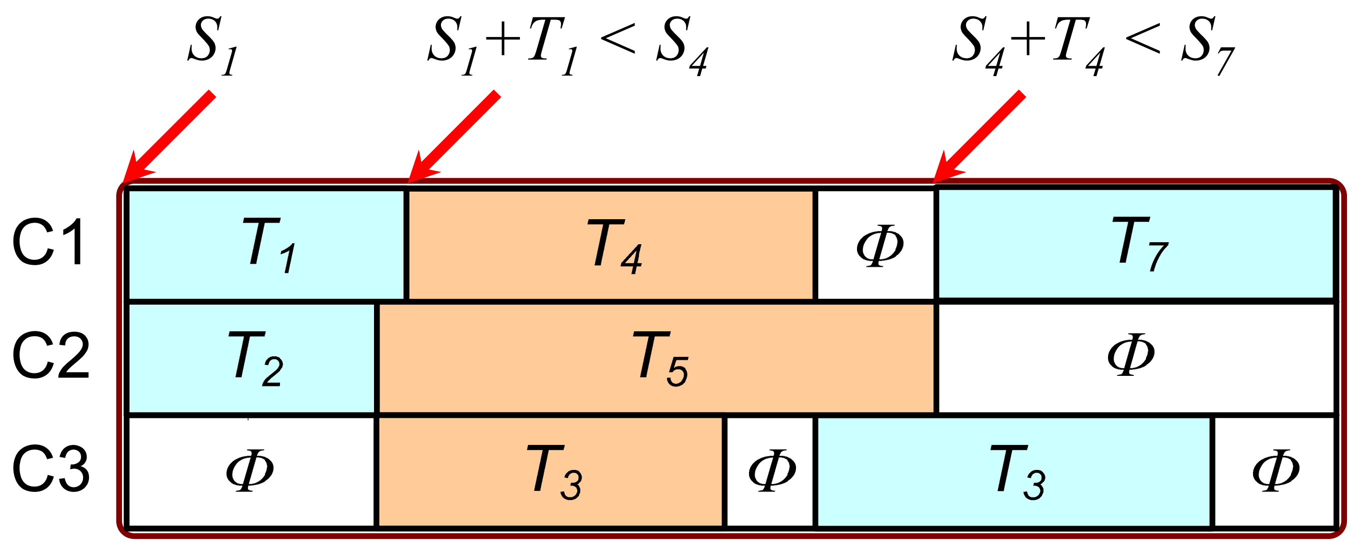
- To keep precedence relationships:
  - For each arc *(i, j)*:

$$S_i + T_i + T_{comm}^{(i,j)} \leq S_j$$

  - T_i: the execution time for task v_i.
  - T_{comm}: the communication time between v_i and v_j.

Decision Variables (3/3)

- M and $\{S_1, S_2, \ldots, S_n\}$ determine the schedule:



- We need to identify M matrix and $\{S_1, S_2, \ldots, S_n\}$ for the optimal schedule.

Workload Constraint

- A stressed core will be assigned low workload.
- Assigned workload for each core

$$WL_i = \frac{\sum_j \alpha_{ij} T_j, \quad \forall M_{ij} = 1}{T_s}$$

  - α_{ij} : the frequency scaling ratio
  - T_s : the length of schedule
- Core utilization must not exceed its capacity rate:

$$WL_i \leq CR_i, \quad \forall i = 1, \cdots, m$$

Communication Cost (1/2)

- Transmitting cost:
  - $N_c(i, j)$: the number of token transmitted on (i, j)
  - $c(i, j)$: time it takes to transmit one token on (i, j)

- Buffering cost:
  - $N_b(i, j)$: the number of token buffered on (i, j)
  - $b(i, j)$: time it takes to buffer one token on (i, j)

Communication Cost (2/2)

- The total communication cost:

$$T_{comm} = T_{tran} + T_{buff}$$

  - T_{tran} : total transmitting cost

$$T_{tran} = \sum_{(i,j)} n_c(i,j)c(i,j)$$

  - T_{buff} : total buffering cost

$$T_{buff} = \sum_{(i,j)} n_b(i,j)b(i,j)$$

The Optimization Model

- A mixed-integer program:

$$\min \quad \alpha \cdot \left(\sum_{i=1}^{m} (1 - U_i) \right) + \beta \cdot T_{comm}$$

$$s.t. \quad \sum_{j=1}^{m} M_{ij} = 1, \quad \forall j = 1, \cdots, n$$

$$S_i + T_i + T_{comm}^{(i,j)} \le S_j, \quad \forall (i, j) \in E$$

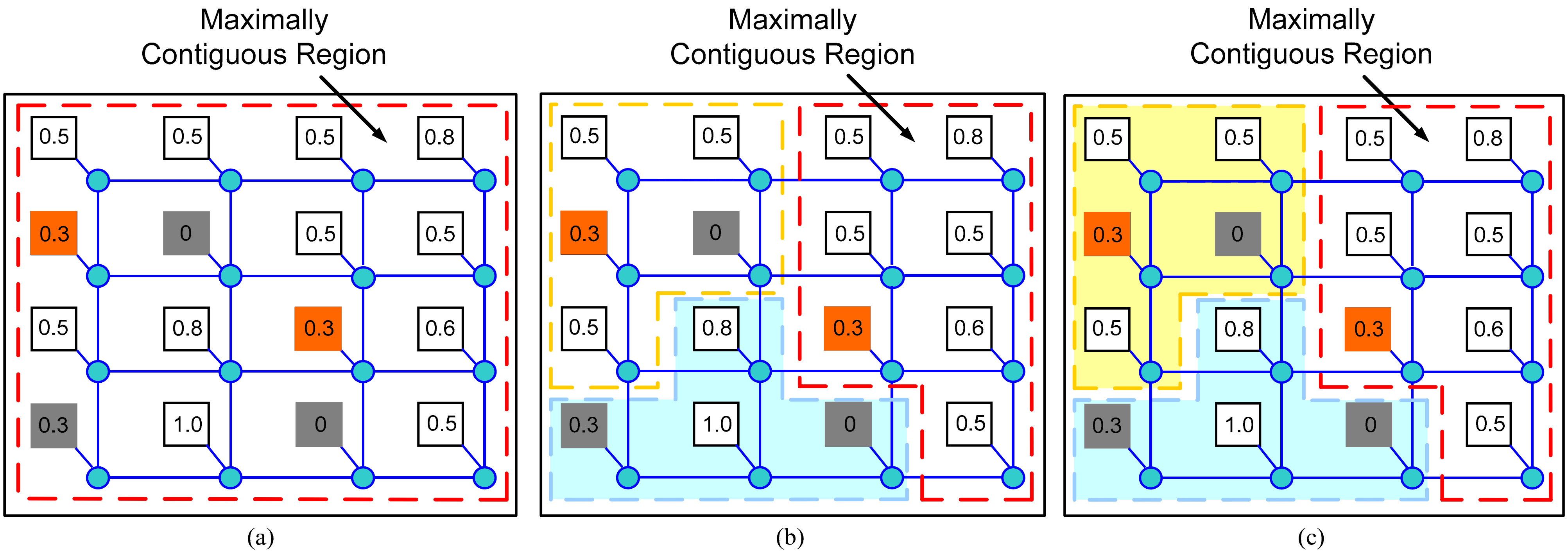
$$WL_i(M, S) \le CR_i, \quad \forall i = 1, \cdots, m$$

$$\text{var} \quad M = [M_{ij}]_{mxn}, \quad S = \{S_1, \cdots, S_n\}$$

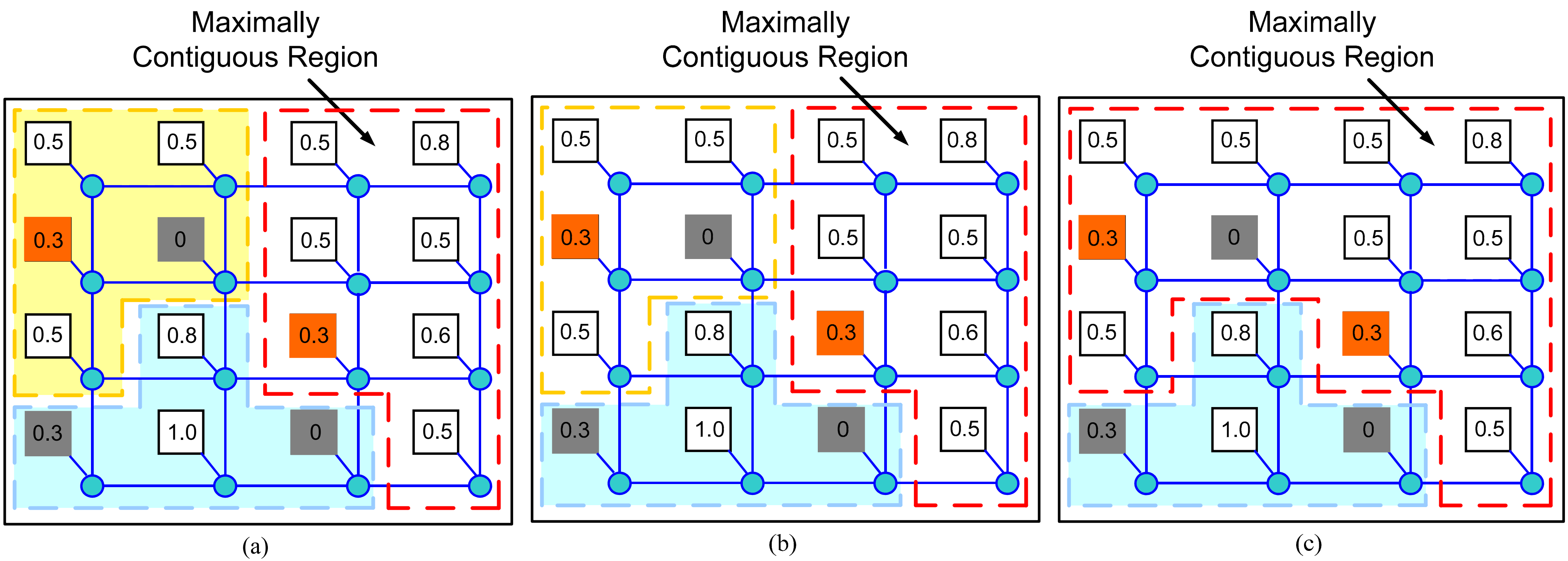
NBTI-Aware Workload Balancing

- Adaptive to the frequent update of capacity rates
- Generate a new zone when a new flow comes in
- Relax an existing flow when a flow is finished execution

Generation of A New Zone



Relaxation of An existing Zone



Results (1/5): Scheduling results

Stressed Core Index	Capacity Rate	Assigned Workload
9	0.7	0.6248
12	0.5	0.5000
28	0.5	0.3892
33	0.4	0.4000
37	0.4	0.4000
39	0.5	0.4473
50	0.8	0.6482
57	0.9	0.8079

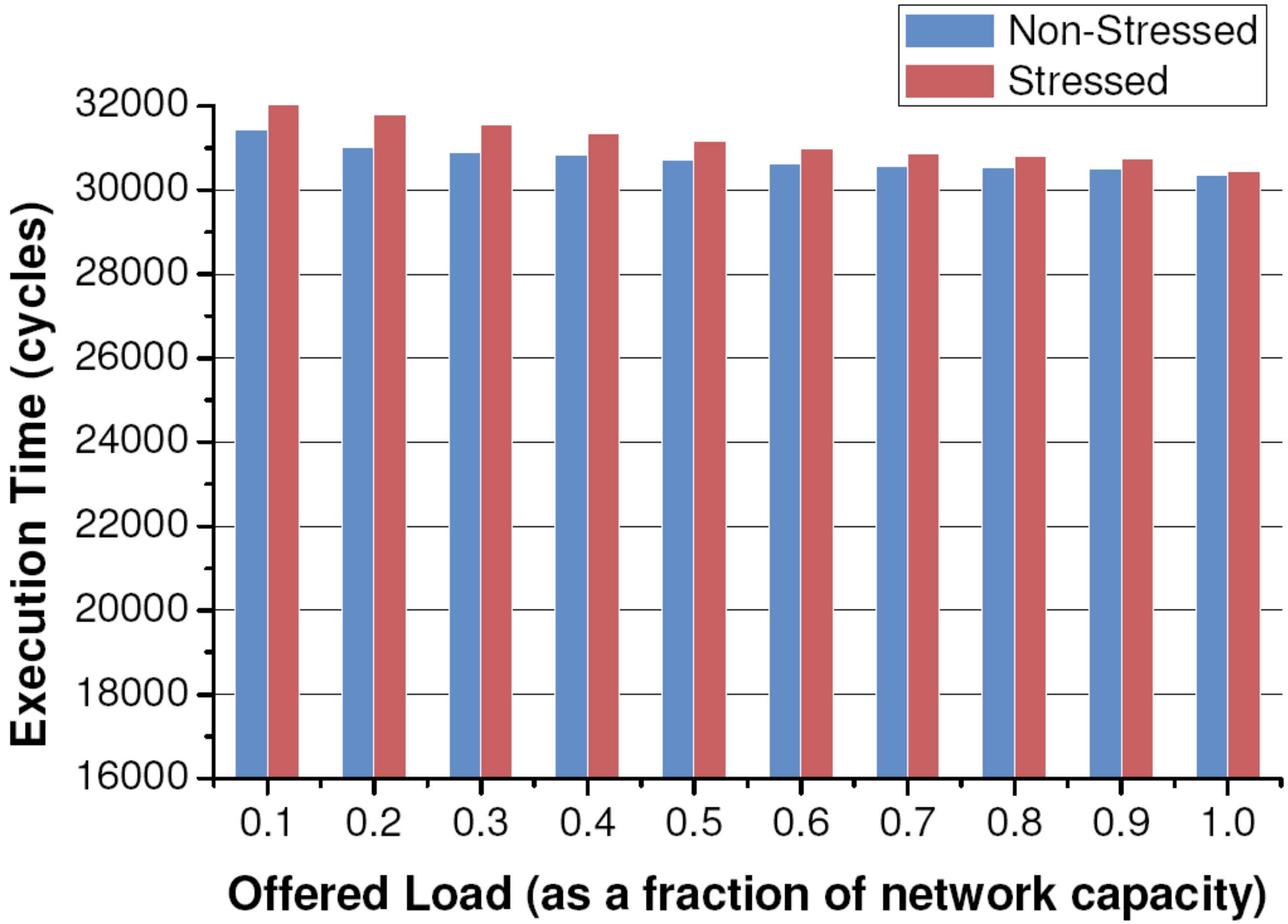
At 90 seconds

Stressed Core Index	Capacity Rate	Assigned Workload
14	0.4	0.3514
25	0.4	0.4000
31	0.5	0.5000
38	0.5	0.3892
41	0.7	0.5864
44	0.3	0.3000
52	0.6	0.5536
58	0.5	0.4293
59	0.2	0.2000

At 100 seconds

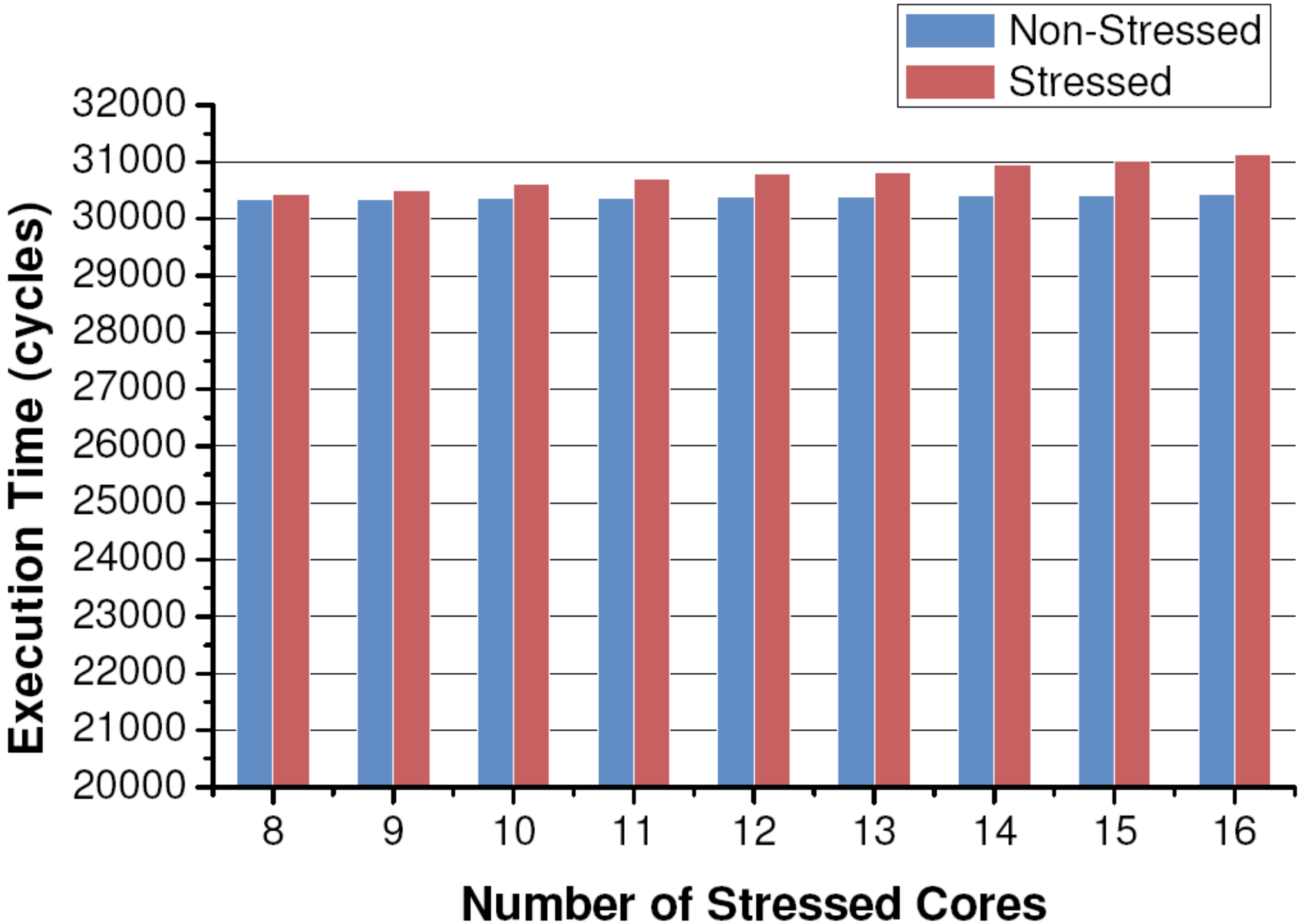
- Assigned workload is well bounded by capacity rate.
- Stressed cores have been recovered.

Results (2/5): Performance Comparison



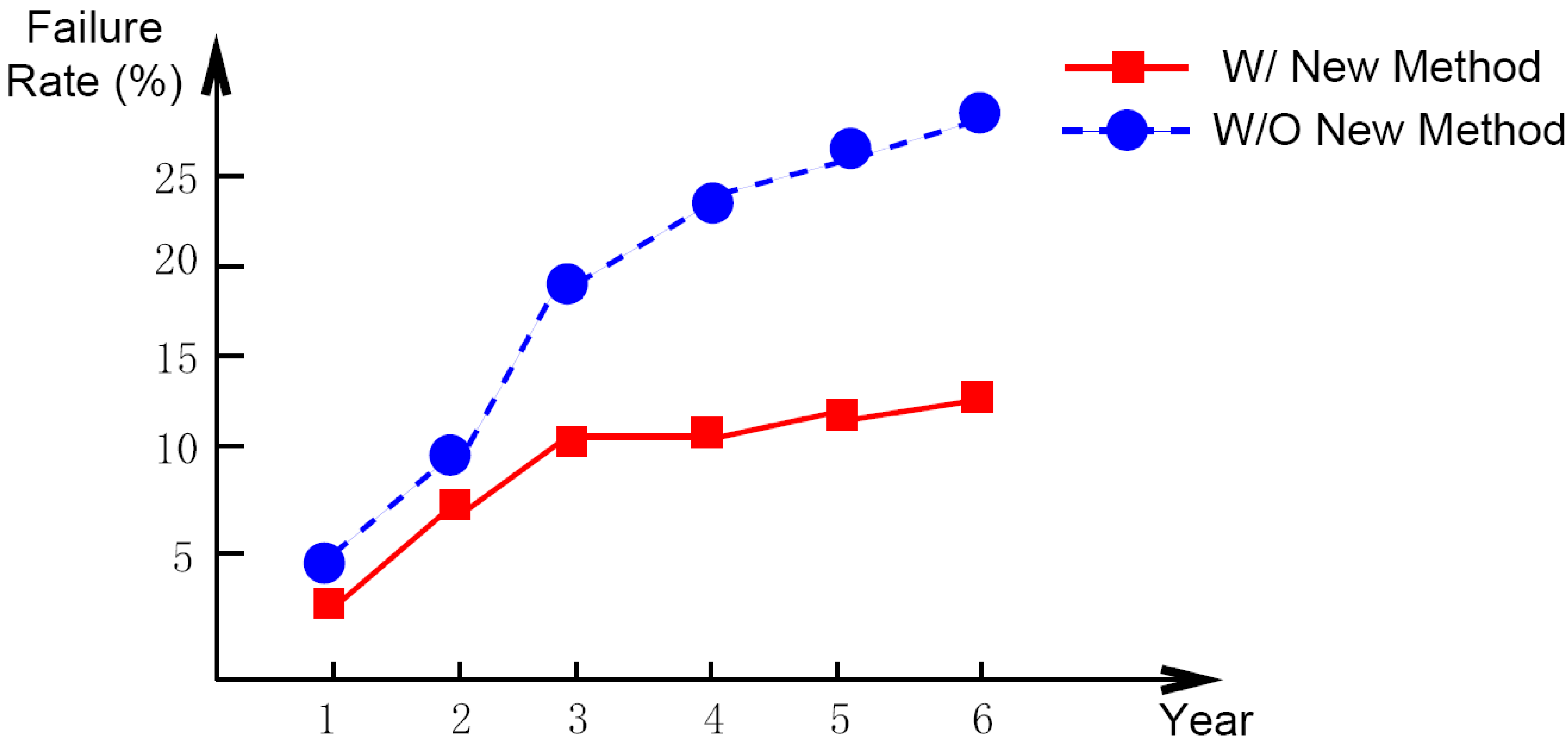
- An insignificant increase in execution time with different offered load (<2% performance degradation)

Results (3/5): Performance Comparison



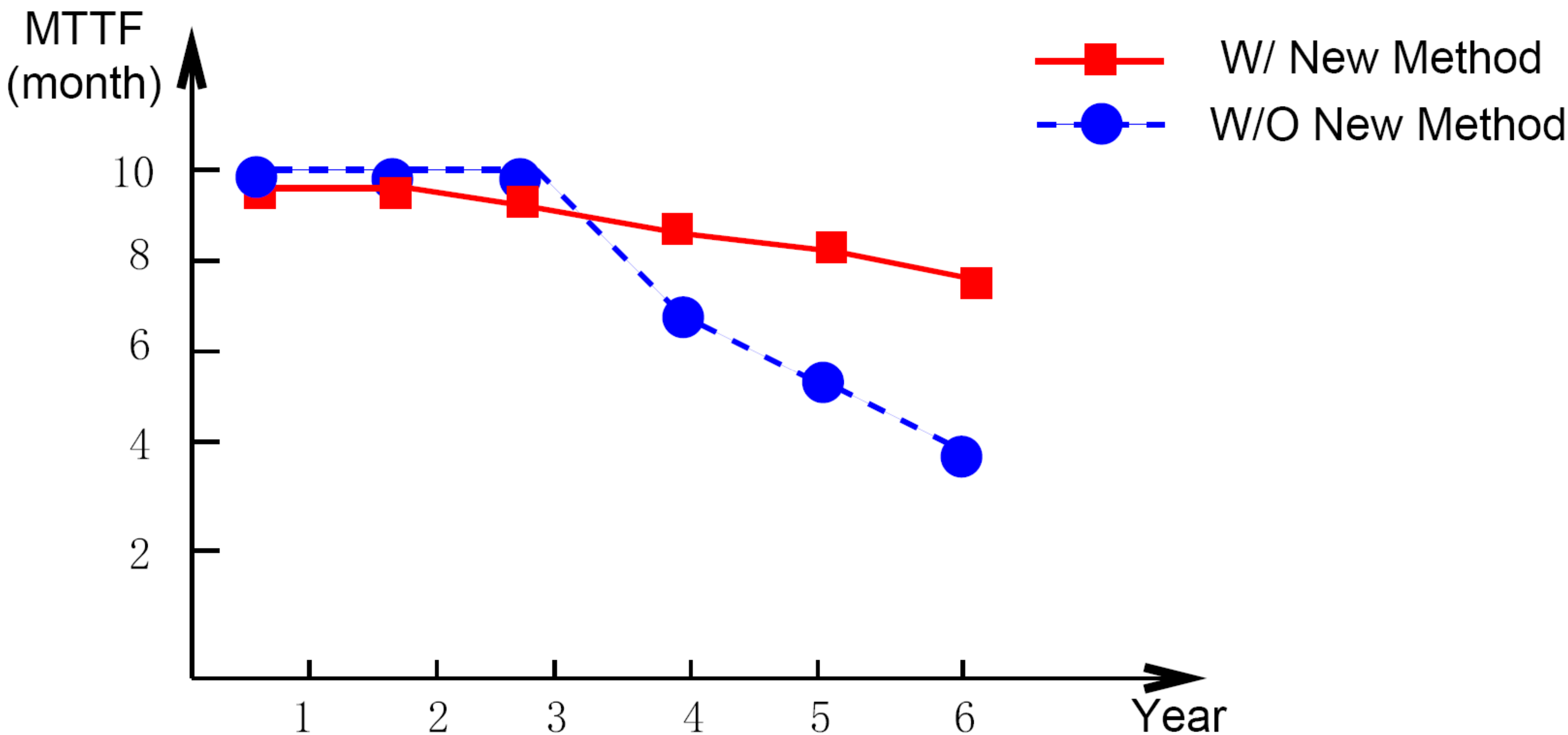
- An insignificant performance drop with increasing number of stressed cores (~3% when there are 16 stressed cores).

Results (4/5): Yield Comparison



- The differences in terms of yield becomes obvious after 2 years and starts to widen.

Results (5/5): MTTF Comparison



- After about 3 years both cases observe decreases in MTTF.
- The new strategy shows about 30% less changes.

Thank you!

Q&A