Adding Memory Resource Consideration into Workload Distribution for Software DSM Systems

Presented by Tyng-Yeu Liang

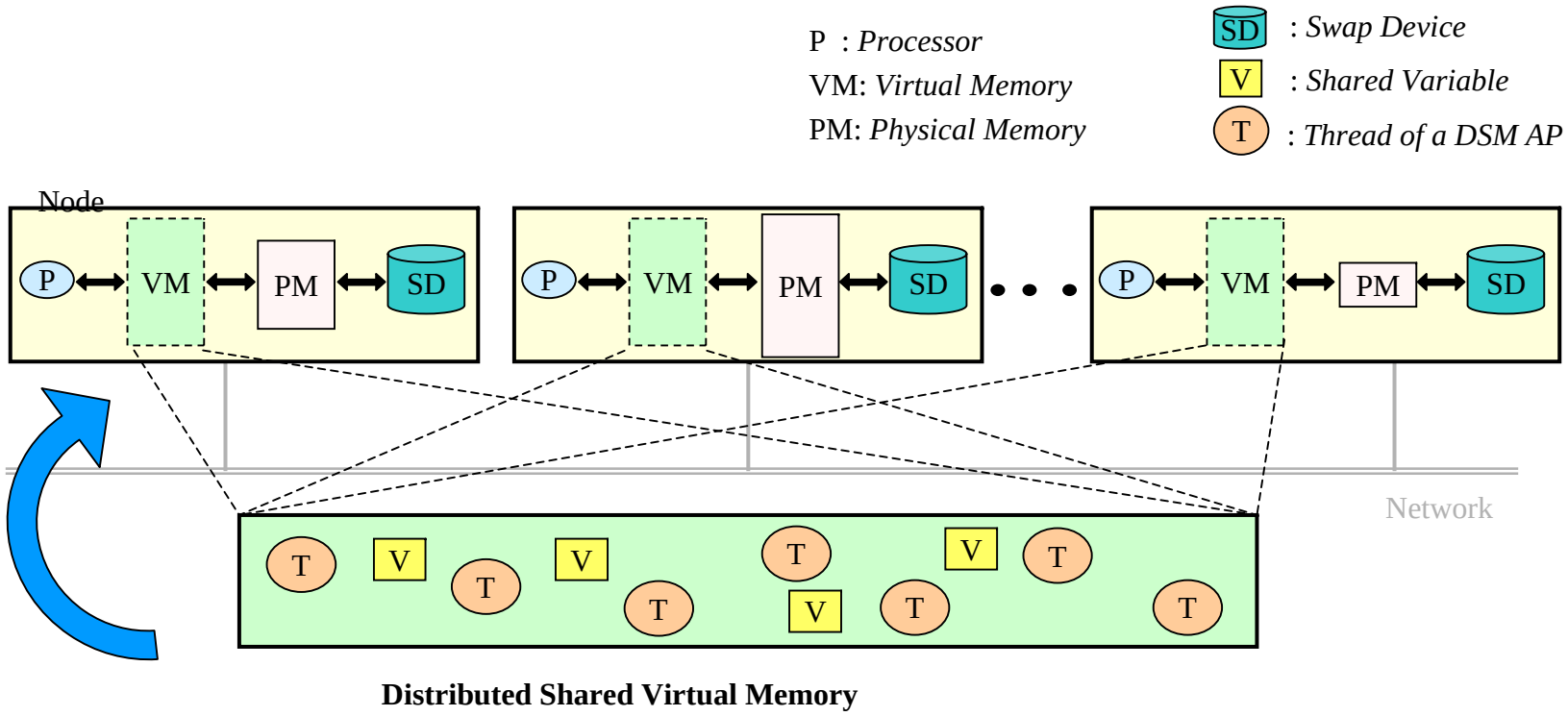
Department of Electrical Engineering,
National Kaohsiung University of Applied Science,
Kaohisung, Taiwan

Outline

- Introduction
- Our Work
- Performance
- Conclusions and Future Work

Introduction

1. Distributed Shared Memory System (DSM)
   - a run time system that emulates a virtual shared memory abstract on computer network via. software technology



Introduction (Conti.)

- Advantages of Software DSMs
  - *easy programming*
  - *high scalability*

-Important issues of DSM-applications performance
  - *minimization in data-consistency communication*
  - *workload distribution*

Introduction (Conti.)

1. Related Work

   - Load balance
     goal : minimal computation time
     considerations:
       computational powers of processors
       computational needs of threads

   - Reduction on internode Data Sharing
     goal: minimal communication time
     consideration: thread access pattern

Introduction (Conti.)

1. Insufficient physical memory space
   - processors need to consecutively execute page replacement for data caching while executing the threads
   - the execution of these threads will be delayed by the latency of executing page replacements

Our Work

Experimental Environment, Teamster

based on Intel personal computer connected with Ethernet

features of Teamster

- *multiple consistency protocol sequential and eager update*

-multithreading per node

-auto-reconfiguration

Our Work

Goal

This paper is aimed at adding memory resource consideration in workload distribution and evaluating the effectiveness of this new method in minimizing the execution time of the applications running on DSM systems.

Our Work

Scope of our Work

-loop applications

-regular data access patterns

Steps of our work

- analysis

- implementation

- performance evaluation

Our Work

1 Analysis

greedy method :

the method implements the optimal workload distribution solution for each iteration such that the total execution time is minimized.

the finish time of a given iteration is determined by the longest finish time of any node working within that iteration as follows.

$$TI = Max \{T^1, T^2, T^3, ..., T^N\}$$

Our Work

1. Analysis
   - T^x : the execution time of node x

$$T^x = T^x_{comp} + T^x_{comm} + T^x_{mem}$$

   - T^x_{comp} : Computation time is the time that node x executes the computational work of local threads.
   - T^x_{comm} : Communication time is the time that node x communicates with other nodes for maintaining data consistency
   - T^x_{mem} : Memory time is the time that node x executes page replacements to cache the data accessed by its local threads.

Analysis (Conti.)

Computation time

$$T_{comp}^{x} = \sum_{i \in S_x} t_{comp}^{ix}$$

1. Let
   - S_x be the set of threads running on node x,
   - t_{comp}^{ix} be the computation time of thread i running on node x

Analysis (Conti.)

Communication time

$$T_{comm}^{x} = \sum_{y=1, y\neq x}^{N} \sum_{k=1}^{P} C_{xyk},$$

$$\begin{cases} C_{xyk} = \left\lceil \dfrac{\phi(diff_{xk})}{Packetsize} \right\rceil t_{packet}, if\ \ node\ \ x\ and\ \ node\ \ y\ \ share\ \ \ page\ \ k \\ C_{xyk} = 0, otherwise \end{cases}$$

T_{comm}^{x} , is the time the node x spends in the propagation of its update to shared data pages for maintaining data consistency.

N denotes the number of execution nodes

P represents the number of data pages and t_{packet} is the time of transferring one message packet in network.

Analysis (Conti.)

Memory time

$$T_{mem}^{x} = \begin{cases} 0 & , if \quad M_x \geq \sum_{i \in S_x} m_i \\ t_{mem}^{x} & , if \quad M_x < \sum_{i \in S_x} m_i \end{cases}$$

1. Let
   - t_{mem}^{x} be the time of executing page replacements for caching the data on node x.
   - M_x be the maximum memory space which node x is able to afford for thread memory demand
   - m_i be the memory space requested by thread i

Analysis (Conti.)

$$t_{mem}^{x} = \sum f^{x} \times \left(t_{spi}^{x} + t_{spo}^{x} \right)$$

- f^{x} : the number of page replacements executed by node *x* in caching the data pages required by its local threads.
- t_{spi}^{x} : the average time of swapping in pages on node x
- t_{spo}^{x} : the average time of searching LRU data pages and swapping them out on node x

Performance

1. Applications

	Problem size (floating points)	computation demand per thread(s)	memory demand per thread	The average no. of threads sharing the same page
SOR	6144*6144	20~25sec	9.375MB	1.2
MM	2560*2560	20-25sec	9.375MB	0
VQ	2048*2048	47~50sec	3.382MB	8.3

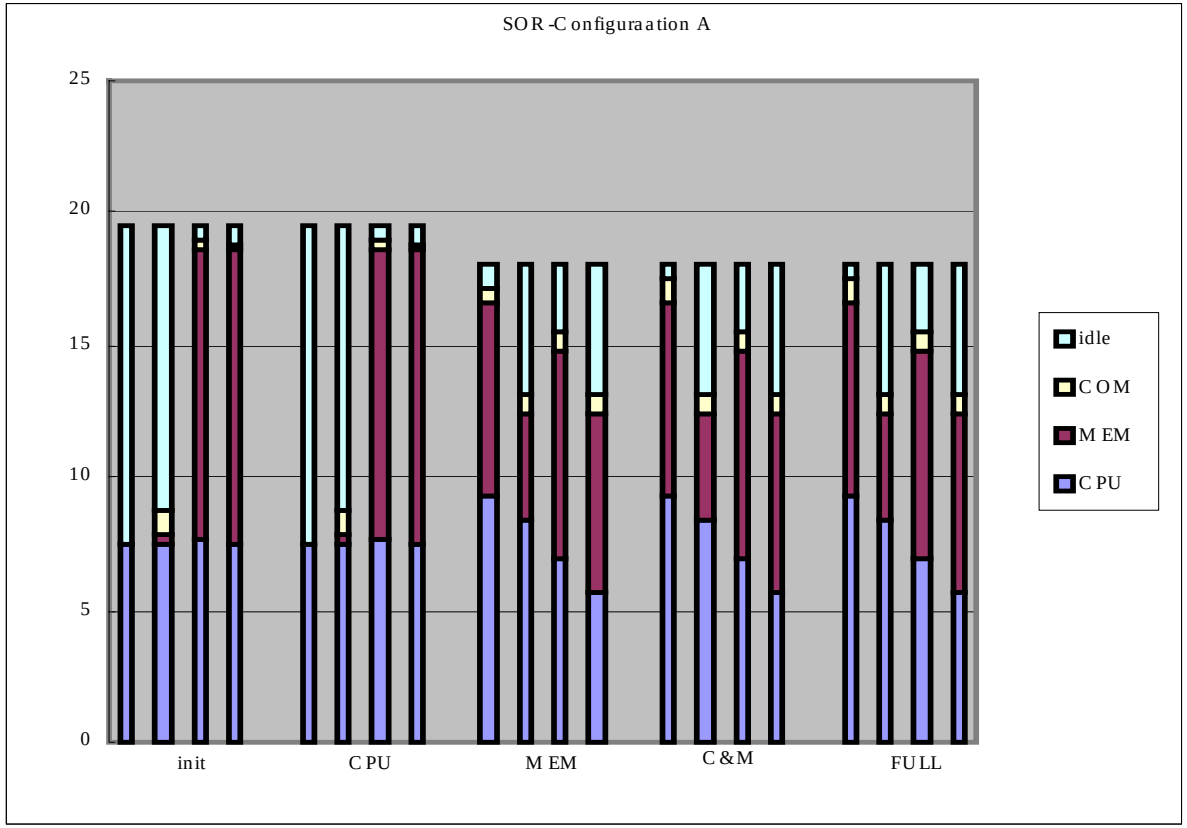
Performance (Conti.)

Cluster configurations

	node 0	Node1	node2	node3
A	500Mhz 90MB	500Mhz 75MB	500Mhz 60MB	500Mhz 45MB
B	500Mhz 80MB	400Mhz 80MB	400Mhz 80MB	300Mhz 80MB
C	500Mhz 90MB	400Mhz 75MB	400Mhz 60MB	300Mhz 45MB

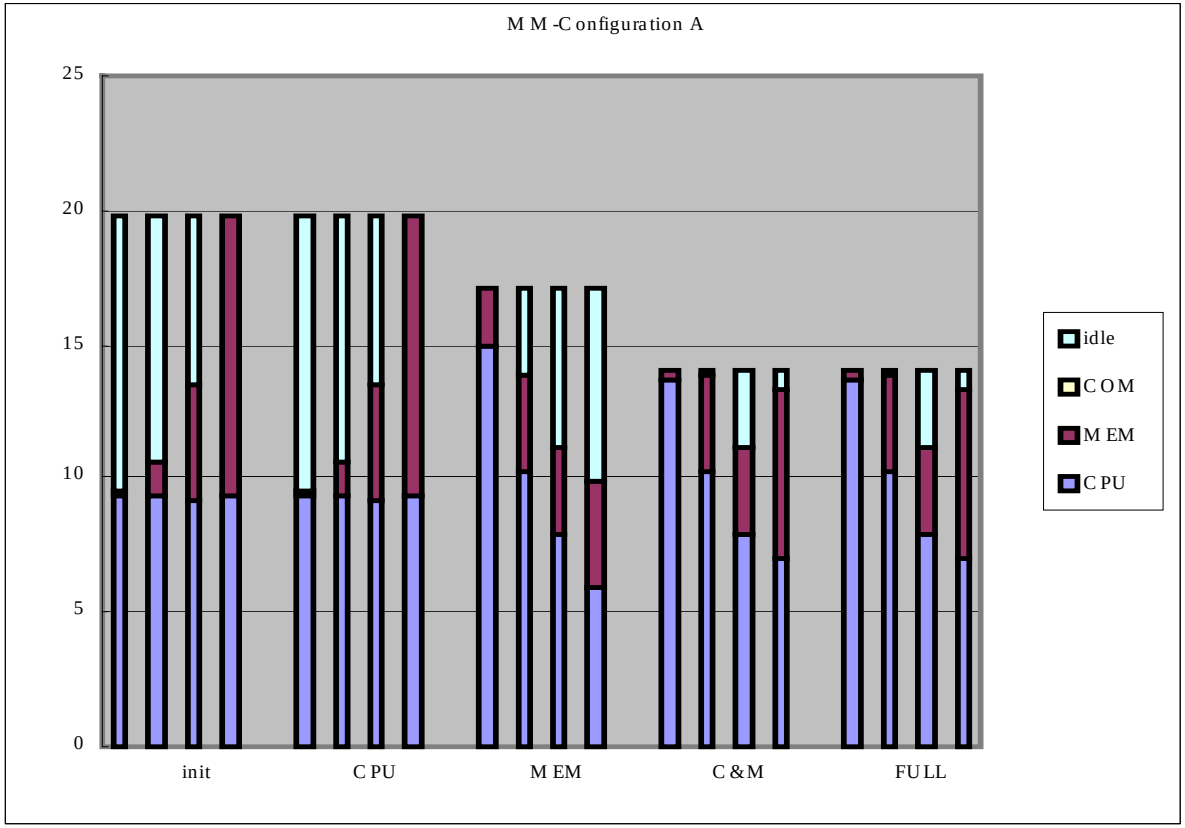
Performance (Conti.)

SOR-Configuration A



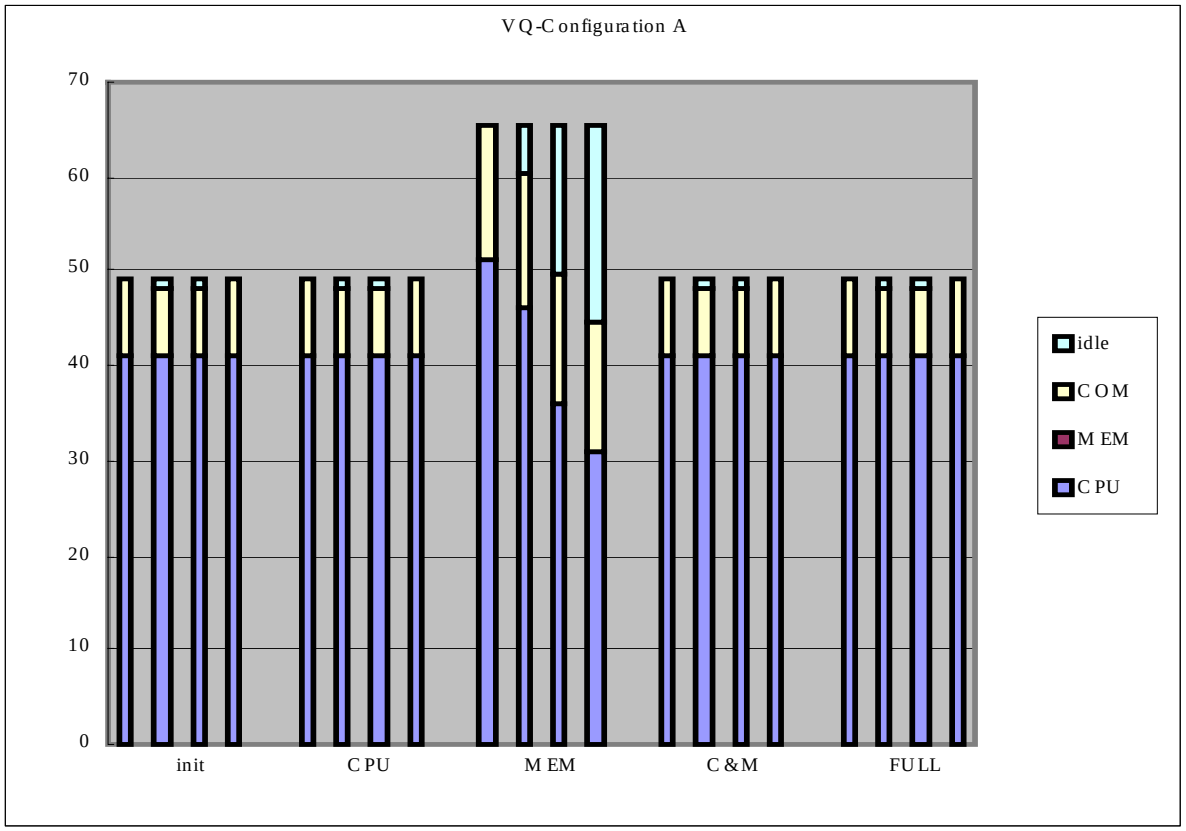
Performance (Conti.)

MM-Configuration A



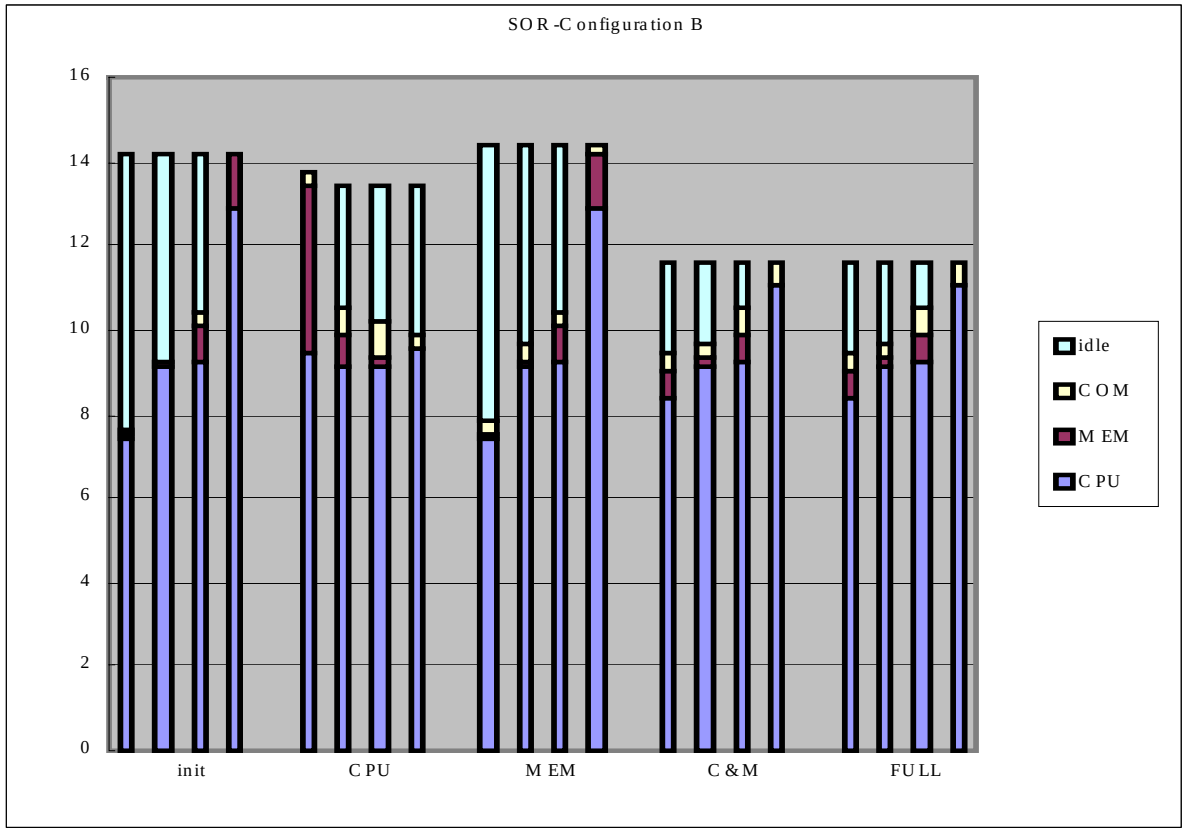
Performance (Conti.)

VQ-Configuration A



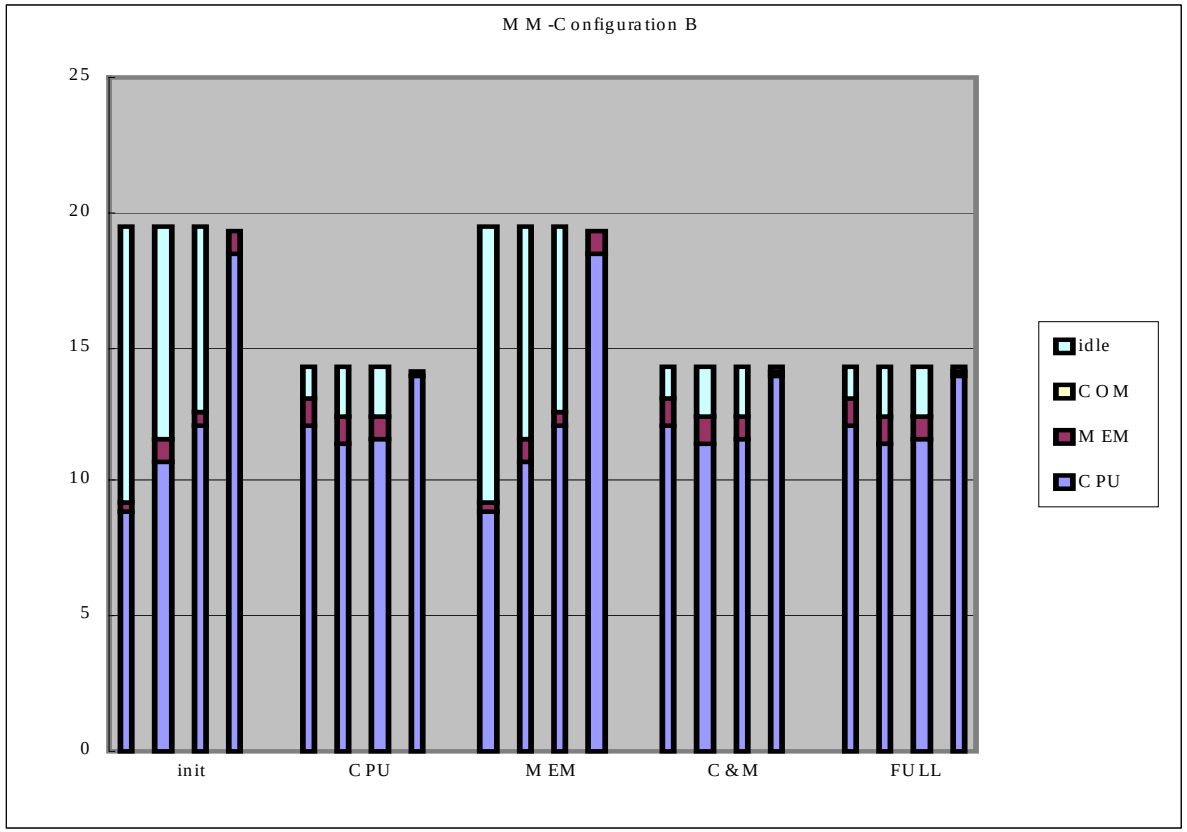
Performance (Conti.)

SOR-Configuration B



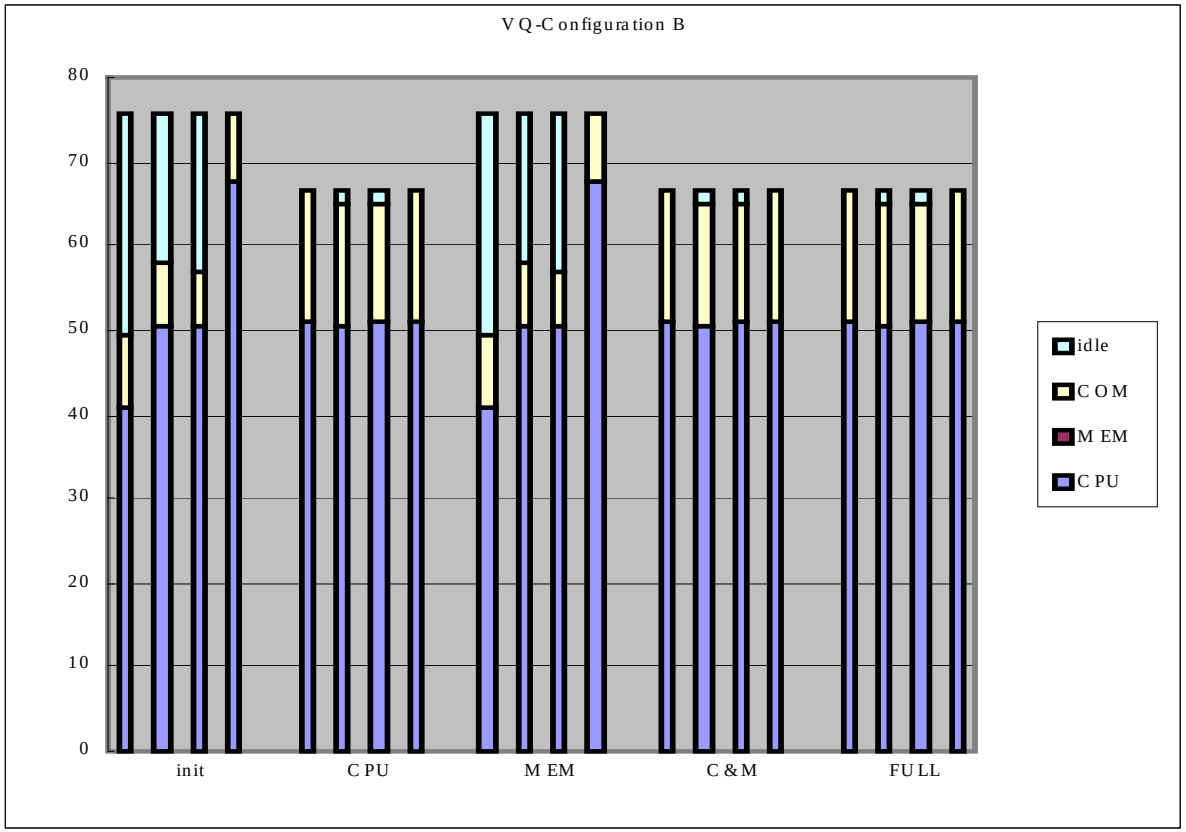
Performance (Conti.)

MM-Configuration B



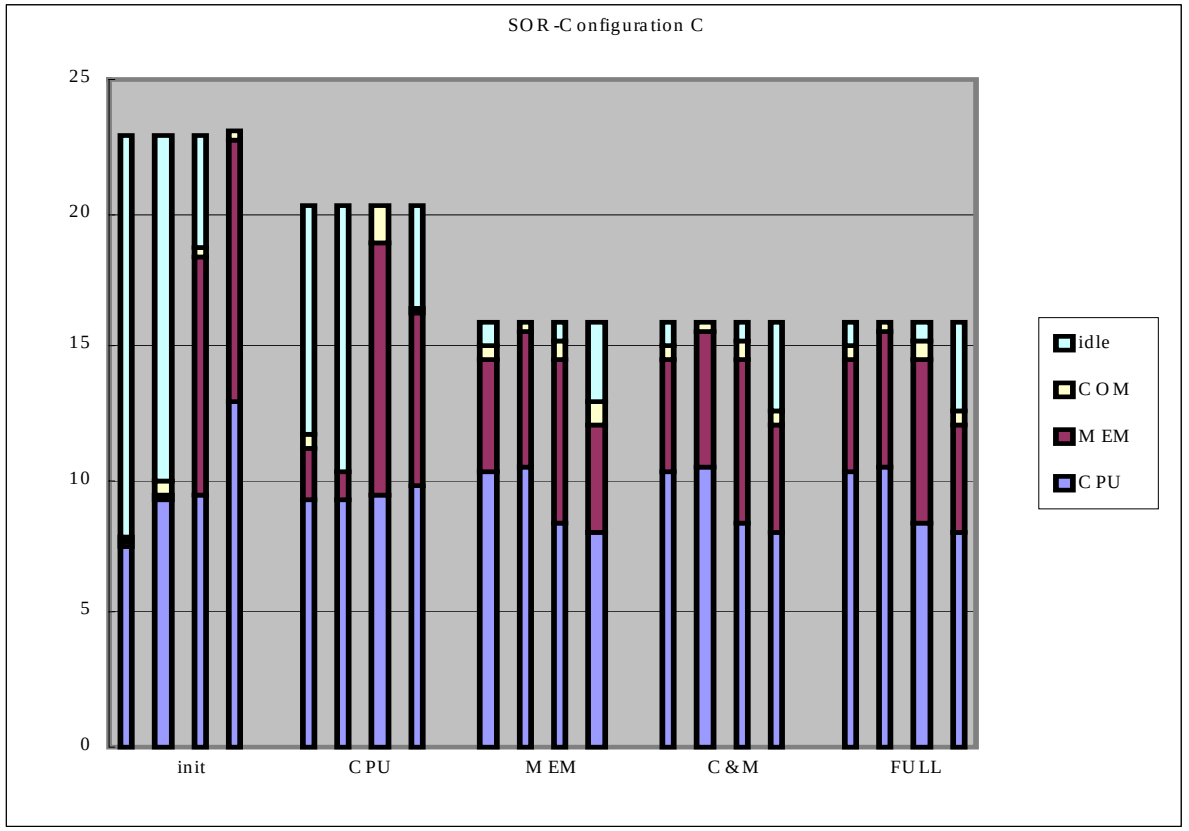
Performance (Conti.)

VQ-Configuration B



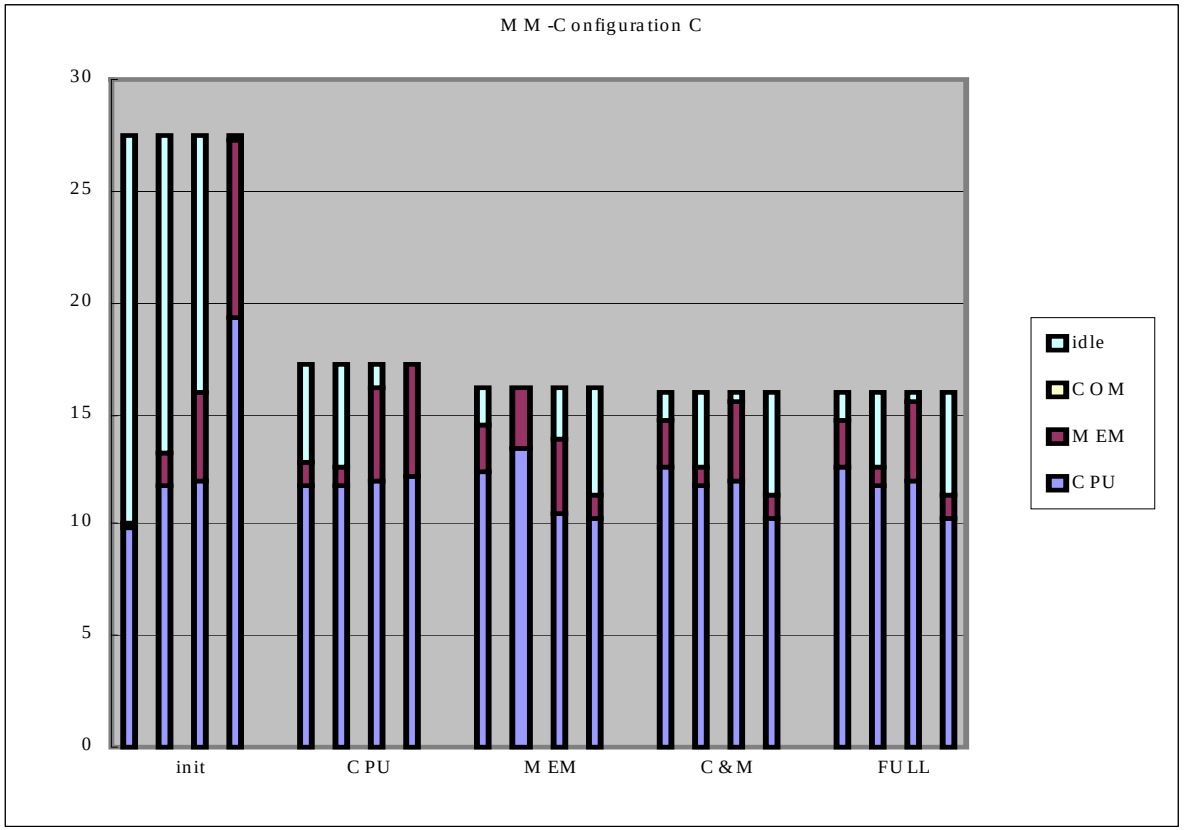
Performance (Conti.)

SOR-Configuration C



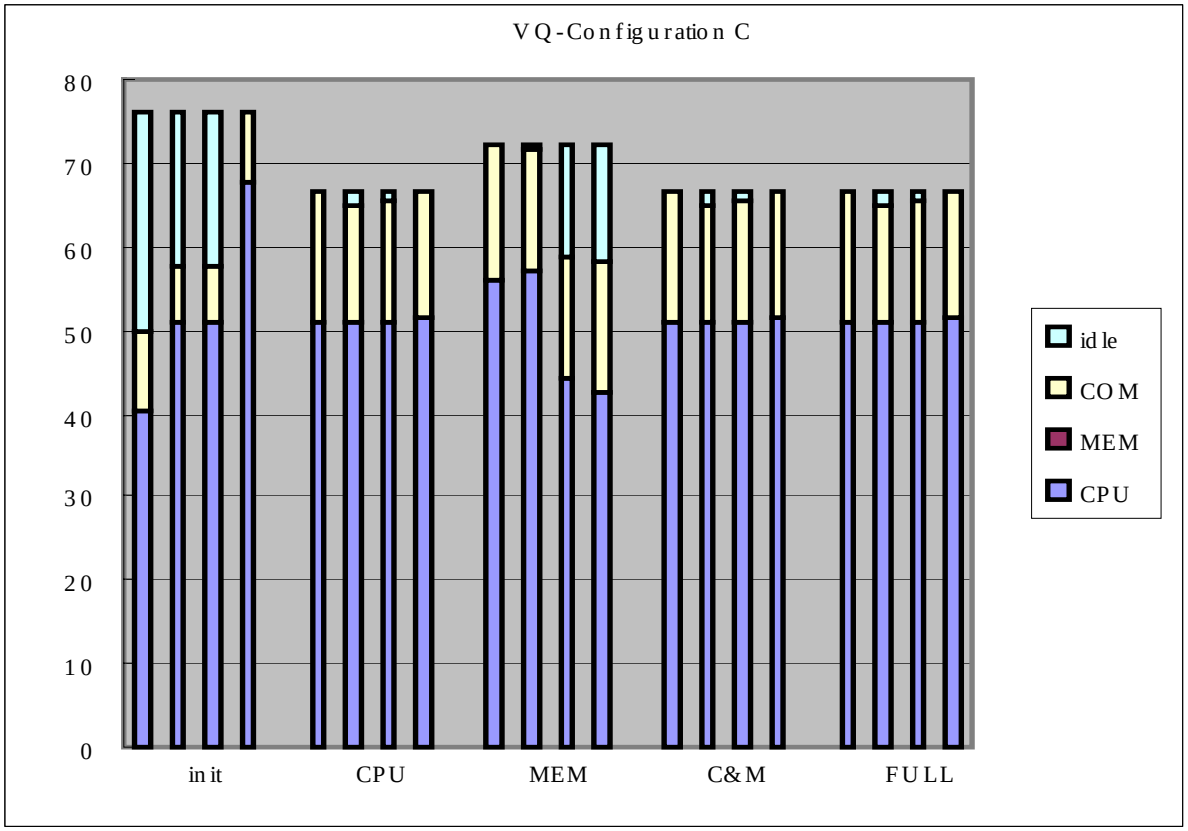
Performance (Conti.)

MM-Configuration C



Performance (Conti.)

VQ-Configuration C



Conclusions and Future Work

1 Conclusions

- Memory resource is an important consideration for the workload distribution problem of software DSM systems.
- The experimental results show that adding memory resource consideration in workload distribution is significant for minimizing the execution times of the applications running on DSM systems.

Conclusions and Future Work

1. **Future Work**

   - an advanced workload distribution method for DSM systems, which is clustered with SMP (symmetric multiple processors) machines.

   - the problem of workload distribution for simultaneously running multiple DSM applications on the same computer cluster