Involving Memory Resource Consideration into Workload Distribution for Software DSM Systems

Presented by Yen-Tso Liu

Department of Electrical Engineering,
National Chung Kung University,
Tainan, Taiwan

Outline

- Introduction
- Related Works
- Analysis
- Implementation
- Performance
- Conclusions and Future Work

Introduction

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

Introduction (Conti.)

P : *Processor*
VM: *Virtual Memory*
PM: *Physical Memory*

SD : *Swap Device*
V : *Shared Variable*
T : *Thread of a DSM AP*

Distributed Shared Virtual Memory

Related Works

- Past Methods
  - adapted the number of threads assigned on each processor to make load balance
  - never cared about whether processors have enough memory capability to meet thread memory demand

Introduction (Conti.)

- 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 absolutely delayed by the latency of executing page replacements

Analysis

- T^x is the execution time of node x

$$T^x = T^x_{comp} + T^x_{mem} + T^x_{comm} \quad \text{... Eq.(1)}$$

- T^x_{comp} is the time that node x executes the computational work of local threads.
- T^x_{comm} is the time that node x communicates with other nodes for data sharing purposes.
- T^x_{mem} is the time spent by node x in performing page replacements to cache the data accessed by its local threads.

Analysis (Conti.)

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

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

Analysis (Conti.)

$$T_{mem}^{x} = \begin{cases} \sum_{i \in S_x} t_{mem}^{i}, & if \bigcup_{i \in S_x} m_i > M_x \\ 0, & if \bigcup_{i \in S_x} m_i \le M_x \end{cases}$$

- Let
  - t_{mem}^{i} be the memory access latency of thread i 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.)

$$t_{mem}^{i} = \sum_{i \in S_x} f^{i} \times \left(t_{spo}^{x} + t_{spi}^{x} \right)$$

- f^{i} : the number of page replacements that node x executes in caching the data needed by thread i
- 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

Analysis (Conti.)

$$T_{comm}^{x} = \sum_{y=1, y \neq x}^{N} \sum_{k=1}^{P} C_{xyk} \quad \text{where} \quad C_{xyk} = \left\lceil \frac{\phi\left(\bigcup_{i \in S_x} diff_{ik} \right)}{Size_{packet}} \right\rceil \times t_{packet}$$

if node x and node y share page k, and $C_{xyk} = 0$ otherwise.

- N is the number of execution nodes
- P is the number of total data pages
- $Size_{packet}$ means the network packet size
- t_{packet} is the average time of transferring one message packet in network

Workload Distribution Algorithm

1. Identify the node with the longest finish time and the node with the shortest finish time, and designate these as the source node and the destination node, respectively. Specify the length of the program's critical path to be the same as the finish time of the source node.
2. Migrate a thread from the source node to the destination node.
   a. Use Eq.(1) to predict the length of the program's critical path, and determine if the length of the critical path is reduced. If this is the case, repeat Step (2) until the critical path can not be reduced any more. Then the process will go to Step (1).
   b. If no thread can be migrated between this pair of execution nodes, the system will try to designate other nodes as the destination node and repeat Step (2).
   c. If no thread can be migrated between any pair of execution nodes, the process of adapting the workload distribution will come to a halt.

Algorithm (Conti.)

Performance

- Test bed (Teamster)
  - Teamster is a DSM system built on a cluster of Intelx86 PCs connected with Ethernet network
- Three iterative DSM applications
  - SOR, Matrix Multiplication (MM), Gaussian Elimination (GE), Jacobi, N-Body, MPEG4 Encoder (MPEG4) and Vector Quantization (VQ)

Performance (Conti.)

- Environment
  - 4 Nodes
    - Intel-Pentium III Xeon 500 MHz processor
    - 512MB DRAM
    - Sun Solaris 8 operating system
  - Connected by 100-Mbps Ethernet
  - Auxiliary tool to control available free physical memory

Performance (Conti.)

Group 1	Node 0 (Mhz - MB)	Node 1 (Mhz - MB)	Node 2 (Mhz - MB)	Node 3 (Mhz - MB)
A	500 - 100	500 – 85	500 - 70	500 - 55
B	500 - 90	500 – 75	500 - 60	500 - 45
Group 2	Node 0	Node 1	Node 2	Node 3
A	500 - 100	400 - 100	400 - 100	300 - 100
B	500 - 80	400 - 80	400 - 80	300 - 80
Group 3	Node 0	Node 1	Node 2	Node 3
A	500 - 90	400 - 75	400 - 60	300 - 45
B	500 - 45	400 - 60	400 - 75	300 - 90
C	500 - 60	400 - 75	400 - 75	300 - 60

The organization of computer clusters used for executing the test applications

Performance (Conti.)

	Problem Size (Double)	Number of Threads	Average Memory Demand per Thread (MB)
SOR	6144*6144	32	9.275
MM	2560*2560	32	9.375
GE	6144*6144	32	9.604
Jacobi	6144*6144	32	9.727
N-Body	61440	32	0.092
MPEG4	640*480	32	0.124
VQ	2048*2048	32	3.382

Parameters of test applications

Performance (Group1)

Performance (Group2)

Performance (Group3)

Conclusions

- This paper has confirmed the importance of taking memory resource considerations into account when specifying a suitable workload distribution to optimize the performance of DSM applications.
- The impact of memory resource to the performance of DSM programs has been analyzed, and the analysis results have been successfully added into the workload distribution mechanism on the Teamster test bed.
- The results of the present study have shown that a workload distribution method with including memory resource is more effective in minimizing the execution time of DSM programs than the previously proposed methods without including memory resource.

Future Work

- In the future, we will develop an advanced workload distribution method for DSM systems which is clustered with SMP (symmetric multiple processors) machines. Moreover, we also study the problem of workload distribution for multiple applications simultaneously running on DSM systems.