A Network Memory Architecture Model and Performance Analysis

Li Liu†‡, Mingyu Chen†,Yungang Bao†‡, Jianwei Xu†‡, Jianping Fan†
† Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
‡ Graduate School of Chinese Academy of Sciences, Beijing, China
{ liuli,baoyg,xjw}@ncic.ac.cn {cmy,fan}@ict.ac.cn

1. Introduction

High-speed electrical circuit and optical interconnection technique bring us high-bandwidth and low-delay network. It is possible to use network memory to satisfy the memory bandwidth and capacity requirements in multiprocessor system. Previous researches have proposed that the optical interconnection technology can innovate memory architecture [1,2]. This paper presents the performance study of a network memory architecture based on the idea of Dynamic Self-organized Architecture based on Grid-component (DSAG) [1].

But the long latency caused by remote memory access makes the imbalance between CPU and memory even harder, which will incur system slow-down. Most software based network memory systems were only used as network block devices and large amount of local memory were still needed.

The motivation for the research of this paper came from the questions: Is remote memory feasible if remote assess delay can be decreased to certain low level? How does the remote access delay influence application performance? How to use cache and prefetching to improve the performance?

To answer these questions, we assume an ideal network memory architecture and make a performance model to analyze the slowdown caused by the remote access delay.

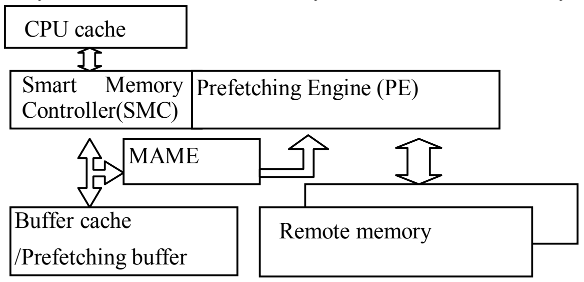
CPU cache
Smart Memory Controller(SMC)
Prefetching Engine (PE)
MAME
Buffer cache /Prefetching buffer
Remote memory

Figure 1: Remote memory architecture

2. Network Memory Architecture and Performance Model

In Figure 1 we present a Network Memory Architecture. The Smart Memory Controller (SMC) can access remote memory by hardware directly. The Memory Access Monitor Engine (MAME) monitors all memory access and provides hint to Prefetching Engine (PE) to do prefetching. Local memory is only used as local page buffer cache and local prefetching buffer. The cache and prefetching pages are all indexed by virtual page address and cache line size is 4K also.

If the accessed page is in local page buffer cache, SMC access it directly, else SMC does a remote memory access operation to get it and put it into the local page buffer cache. When the page is replaced out of the cache, SMC writes it back to the remote memory if it is dirty, or put it into the prefetching buffer if needed ;otherwise discards it.

The application slowdown caused by remote memory is evaluated through a performance model based on some real system memory access traces including Linpack, Blast, Quicksort and SPEC2000 got by HMTT [4].

The memory trace collection tool HMTT is able to monitor the DIMM slot to get the memory access trace data. The trace contain following information (process id (16bit), address (physical/virtual, cache line level, 16bit), read/write bit, duration time (ns level, 17 bit)).

Application slowdown and remote access ratio are defined to evaluate the remote memory system. We also defined some delay parameters in the model including local buffer cache access delay (100ns), local prefetching buffer access delay (100ns) and set remote memory as 5us, 10us, 20us, 40us and 80us respectively to evaluate the performance. The local memory configed for each processor is set as 1MB, 2MB, 4MB, and 8MB respectively because the memory on the test platform is restricted within 512MB.

To get the baseline performance we test the influence of remote access delay on 4MB cache and the influence of page cache size under 5us delay. The result shows that local buffer cache can reduce the remote access ratio greatly and larger local cache can work better. But for some application as Linpack, Quicksort and most SPEC2000 benchmarks as Swim, increasing cache size only can't always improve the performance greatly and remote access delay affects system performance gravely.

Influence Parameter (IP) is defined to evaluate influence.

$IP=|(\text{Slowdown Ratio})_m - (\text{Slowdown Ratio})_n| / (m - n)$.

For remote delay, it is computed when delay is increased from $5us*2^m$ to $5us*2^n$. For cache size, it is computed when cache size is increased from $1MB*2^m$ to $1MB*2^n$.

IP result shows that remote access delay influences harder than cache size for applications as Linpack, Quicksort, which have regular memory page access patterns. So for the applications that have regular memory page access pattern, as Linpack, it is the key to make the remote access delay decrease to a low level that can be acceptable.

To further improve performance a prefetching engine based on stream analysis is given under the architecture.

3. Page Frame Stream Prefetching

To find the application memory access pattern, the virtual page traces for regular access application, such as Linpack, Quicksort, swim, were researched. It can be seen from figure2 that Linpack and Quicksort have linear page stream pattern when access to the memory that caused the page cache misses. The stream is caused by the for() cycle or matrix computing that is common in the scientific applications. So a stream-prefetching engine is presented to get the pages ahead.

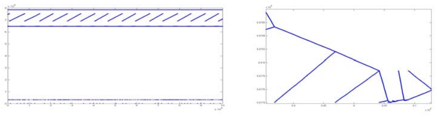

Figure 2. Linpack , Quicksort memory page access

This paper gives a stream check algorithm that is similar to Tushar Mohan's algorithm [3]. We use local buffer cache miss data to check stream to avoid the address repeat pattern problem and use the timestamp to define an aging policy to help creating and deleting the stream. When local buffer cache miss PE adds the page frame address into check buffer.

The algorithm is described as follows:

1: add new page frame information (address, time stamp) into checking window; tag the node as available, goto 2;

2: To check whether the address is in a stream. If it is in a stream, update the stream (length, time stamp), tag the node as unavailable. If the node is in multi streams, repeat to update every stream, goto 1; else, goto 3;

3: To check new node address A with older available node B and C in the checking window. If the address A and B can satisfy A-B=B-C and C is older than B, a new stream is found. So tag A, B, C as unavailable and add the stream into stream pool; goto 1;

If accessed page is in a stream the stream updates its information and prefetchs next page. If the prefetched page is in local memory, we avoid the prefetching and adjust the node location to prefetching buffer if it is deleted in the cache by LRU, else send remote access request to prefetch it.

Simulation shows that this algorithm will not cost too much resource (mostly live streams <60) and most applications have high stream covering ratio (>90%) except Blast and Lucas because of their memory access characters.

Many prefetching can't be done before the page be accessed under singlestep prefetching policy (only get next page) because stream time stride may be less than remote access delay. So a multistep policy (get next n pages) based on time stride is given to solve the problem.

We also compare the performance and communication costs for sequence prefetching, stream prefetching and seq+stream that mix the two algorithms in Figure3, 4.

The results show that with the help of cache and stream prefetching, for applications that have regular memory access patterns, the performance slowdown is less than 2(for Linpack it is 1.4; for SPEC it is 2.56 average, most less than 2) and cause no more communication cost when remote access delay decreases to 5us.

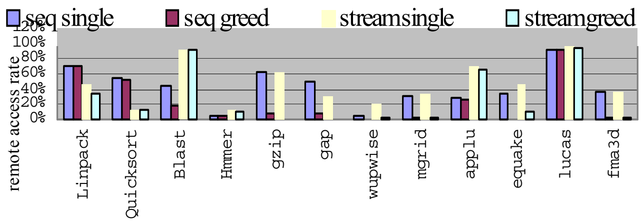

Figure 3. Remote access rate comparison on 4MB cache, 5us remote delay

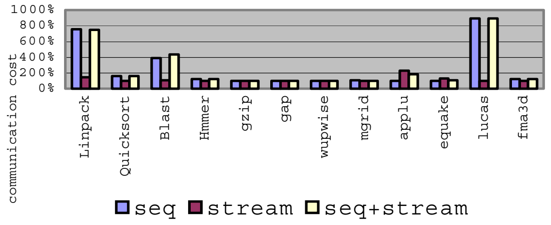

Figure 4. Communication cost comparison

4. Conclusion

It is feasible to build flexible extended remote memory architecture to break the memory capacity restrict for some memory-bound applications with a little performance decrease and the memory can be extended easily and unlimitedly.

References

[1] Jianping Fan, Mingyu Chen. "Dynamic Self-organized Architecture based on Grid-component DSAG". Journal of Computer Research and Development. 2003 Vol.40 No.12.

[2] Y. Katayama and A. Okazaki. "Optical Interconnect Opportunities for Future Server Memory Systems". HPCA-13

[3] Tushar Mohan, Bronis R. de Supinski, Sally A. McKee, Frank Mueller, Andy Yoo, Martin Schulz, "Identifying and Exploiting Spatial Regularity in Data Memory References", Supercomputing Conference 2003, November 2003.

[4] Yungang Bao, Mingyu Chen, Yuan Ruan, Li Liu, Jianping Fan, Qingbo Yuan, Bo Song, Jianwei Xu, "HMTT: A Platform Independent Full-System Memory Trace Monitoring System", SIGMETRICS'08.