A Good Data Allocation Strategy on Non-Uniform Memory Access Architecture

Xiaomei Guo
School of Computer
Communication University of China
Beijing, China
Email: gxmqin@163.com

Haiyun Han
Communication Strategy of Co-innovation Center
Communication University of China
Beijing, China
Email: 353614438@qq.com

***Abstract*—It is important to Choose a good data distribution to the performance of applications on Non-Uniform Memory Access (NUMA) shared memory multiprocessors. In this paper, we have investigated memory accesses behavior over a AMD Opteron cache coherent Non-Uniform Memory Access (ccNUMA) platform with multicore processors. We get important conclusions that it is important to keep data local on node where it is being accessed and it is important to avoid sharing of data between threads running on different cores.**

Keywords—NUMA; multiprocessor; threads; memory access

I. INTRODUCTION

NUMA (Non-Uniform Memory Access) technology combines programmability of SMP (symmetric multiprocess) system and the scalability of MPP (Massively Parallel) system, and has become one of the mainstream architectures today. At present, many well-known foreign server manufacturers have developed a NUMA-based high-performance server architecture.

Today, both Intel and AMD use the NUMA architecture in the multiple CPU chipset to achieve the best performance at a relatively low price. system designers use non-uniform memory access to improve processor speed without increasing the processor bus load. NUMA system is characterized by multiple CPU modules, each CPU module consists of multiple cores (such as six), and has a separate local memory, I/O slots and so on. Each node can access the entire system's memory, because its nodes can connect and exchange information through interconnected modules. But the access time of thread to local and far memory is different.

In this paper, we have investigated memory accesses behavior over a AMD Opteron cache coherent Non-Uniform Memory Access (ccNUMA) platform with multicore processors. The memory accesses behavior investigation has been based on three types memory operations (read access, write access and read/write access), how datas were accessed (regular, irregular and random accesses) and how work was distributed to threads. The differences of local access and remote access are analyzed. Our results have shown that memory accesses behavior is related to data and threads placement on the machine nodes.

This paper is supported by the National Natural Science Foundation of China(Grant No.61502437).

II. NUMA ARCHITECTURE

The traditional model for multiprocessor support is Symmetric Multi-Processor (SMP). In this model, each processor has equal access to memory and I/O. As more processors are added, the processor bus becomes a limitation for system performance. Simplified flowchart of an SMP system is shown in Fig. 1.

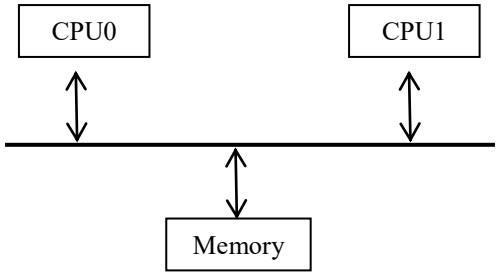

Fig. 1. Simplified flowchart of an SMP system

System designers are now using non-uniform memory access (NUMA) to increase processor speed without increasing the load on the processor bus. The architecture is non-uniform because each processor is close to some parts of memory and farther from other parts of memory. The processor quickly gains access to the memory to which it is close, while it can take longer to gain access to memory that is farther away. Simplified flowchart of an NUMA system is shown in Fig. 2.

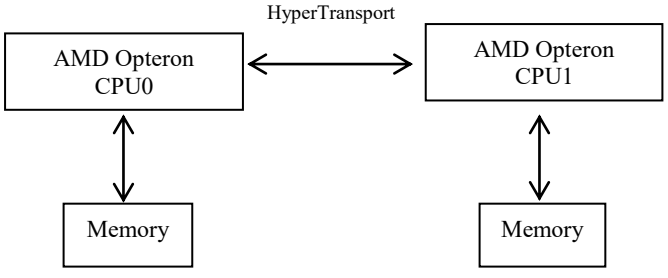

Fig.2.Simplified flowchart of a NUMA system

978-1-5090-5507-4/17/$31.00 ©2017 IEEE
ICIS 2017, May 24-26, 2017, Wuhan, China

In a NUMA system, CPUs are arranged in smaller systems called nodes. Each node has its own processors and memory, and is connected to the larger system through a cache-coherent interconnect bus.

Today, both Intel and AMD use the NUMA (Non-Uniform Memory Access) architecture in the multiple CPU chipset to achieve the best performance at a relatively low price.

In the NUMA system, the performance of application mainly is impacted as follows:

1) Remote memory access. The times of a processor accessing remote memory directly affects the performance of an application. One strategy to improve the performance NUMA system is to reduce the times of remote accesses. The process should run at the local node as much as possible.

2) The effects of interconnected HyperTransport bus bandwidth.

3) The effect of memory competition. When many processors access a memory unit at the same time, there will be memory competition, memory competition will increase the memory response time, reduce the execution efficiency of the program. Appropriate data allocation strategy reduces memory contention.

4) Effects of memory bandwidth.

5) The effect of cache. Cache plays an important role in the performance of NUMA multiprocessor system. If the processor does not find the required data in the local cache, the remote memory is accessed.

III. Experiments and Results

In this research, The NUMA architecture workstation we use is a Quartet system having four 2.2 GHz E1 Dual-Core AMD Opteron™ processors. Each processor had 2x1GB DDR400 DRAM memory. The structure is as fig. 3.

Each processor has one bidirectional HyperTransport link that is dedicated to I/O and two bidirectional coherent HyperTransport links that are used to connect to two other dual-core processors.Each node has its own on-chip memory controller and is connected to its own memory.

When the thread runs and accesses the memory at the same node, it is called a local access or 0-hop access. When the thread is running on a node, accesses to memory in the other node, it is called remote memory access. In remote access, if the node that the thread runs and the memory access node is directly connected to each other, it is called a 1-hop access. If the node that the thread is running is not directly connected to the memory access node, it is called a 2-hop access. In Figure 3, thread runs in node 0, the access to 1, 2 and 3 nodes is remote memory access. The access to node 1, 2 is called 1-hop access, the access to node 3 is called 2-hop access.

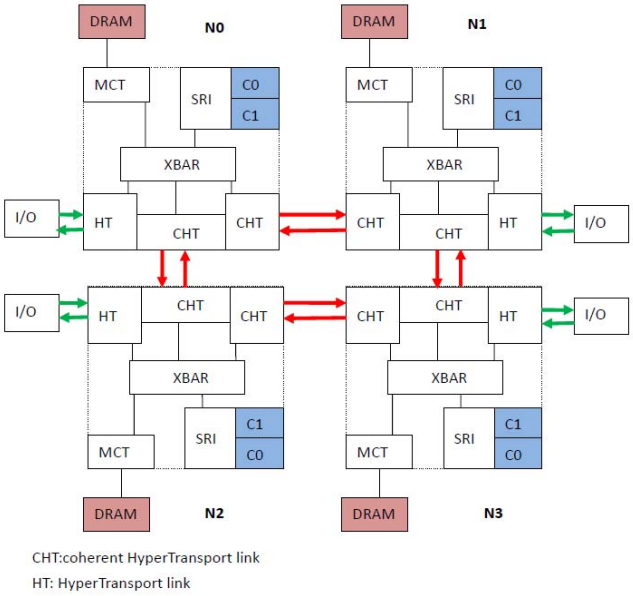

Fig. 3. NUMA structure

A. *Difference of local and remote memory access*

We took a serial program to use a thread to write the same 5M data to different node memory to test the access time, and each time we repeated the experiment 2000 times. The statistical results are shown in Fig.4. The access time of local memory is the shortest, the average is 2166μs; and it will require a longer time for the remote memory access, the average access time to the node 1 and 2 is 2445μs, and for the node 3,the distance is the farest, the average access time is the longest. The time is up to 3032μs.

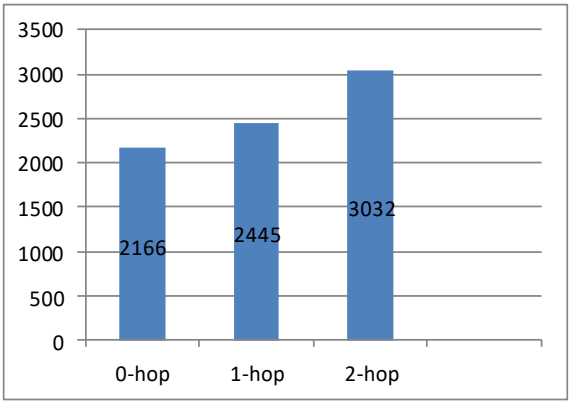

Fig. 4. Access time for local and remote node

From this experiment, we can see that in the NUMA system, access to remote memory is slower than the access to local memory. In this case, the access time of 1-hop distance is 1.13 times of the local access, the access time of 2-hop distance is 1.4 times of the local access. The longer distance thread accesses, the more time it needs. In our application, we should try to keep the data local.

B. *The study of read-only and write-only memory of single thread*

What is the difference between local and remote access when memory access is read-only or write only? We use the following examples to study the influence.

We took a serial program to use a thread to read and write the same 60M data to different node memory to test the access time,the size was much larger than the size of cache. Thread ran on C0 of node 0. And for the system only one thread was running. The data access of thread is divided into the following:

- Local access to node 0.(0-hop)
- Remote access to node 1.(1-hop)
- Remote access to node 2.(1-hop)
- Remote access to node 3.(2-hop)

Including:

- 0.0.R.0—Thread running on node 0/core0 does read-only accesses to memory resident on node 0.
- 0.0.R.1—Thread running on node 0/core0 does read-only accesses to memory resident on node 1.
- 0.0.R.2—Thread running on node 0/core0 does read-only accesses to memory resident on node 2.
- 0.0.R.3—Thread running on node 0/core0 does read-only accesses to memory resident on node 3.
- 0.0.w.0—Thread running on node 0/core0 does write-only accesses to memory resident on node 0.
- 0.0.w.1—Thread running on node 0/core0 does write-only accesses to memory resident on node 1.
- 0.0.w.2—Thread running on node 0/core0 does write-only accesses to memory resident on node 2.
- 0.0.w.3—Thread running on node 0/core0 does write-only accesses to memory resident on node 3.

In this case, the comparison of the results of the access time is shown in Fig.5 and fig.6. With the increase of access distance, the access time of read and write is increased. In each case, the access time to write is longer than read because the write operation produces more memory bandwidth load. However, the access time of read and write increases with the increase of access distance.

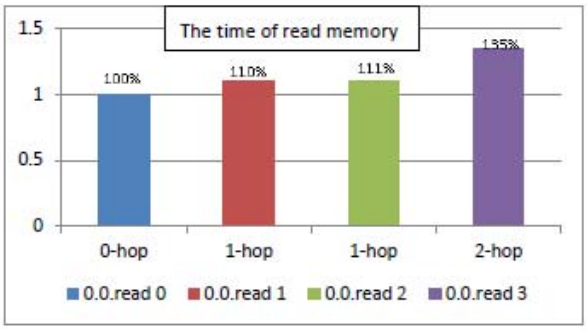

Fig. 5. The thread runing on node 0 in the idle system, and reading the memory of nodes 0, 1, 2, and 3

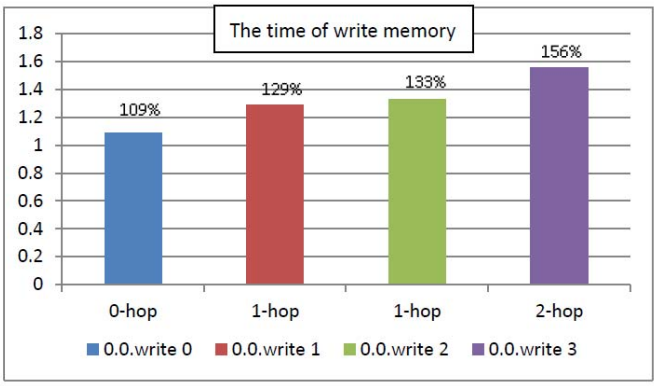

Fig. 6. The thread runing on node 0 in the idle system, and writing the memory of nodes 0, 1, 2, and 3

C. *Research the access between nodes and inside the node*

Multiple threads scheduled between nodes have the following constraints:

- Whether the system is idle, that is, the system has no other load;
- Whether multiple threads access to their own private data.
- Whether multiple threads access to the shared datas.

When the system was idle, We studied threads only accessed the private data of the node.

We used two threads to write, each thread wrote the 60M data and accessed the local memory. The first method was that the threads were scheduled between nodes, that was core 0 on node 0,core 1 on node 1; The second method was that the threads were scheduled in a node, that was core 0 on node 0,core 1 on node 0. Comparing the running time of the two methods. The scheduling time between nodes is relatively fast.

IV. CONCLUSION

In this letter, We get important conclusions through the three experiments. The most important recommendation for most applications is to keep data local on node where it is being accessed. And the second recommendation is to avoid sharing of data between threads running on different cores.

REFERENCES

[1] Bircsak J, Craig P, Crowell R. Extending OpenMP for NUMA machines. Donnelley J. Proceedings of the 2000 IEEE/ACM Conference on Supercomputing: High Performance Networking and Computing. United States: IEEE Computer Society, 2000:48-78 .

[2] B. Falsafi, D.A. Wood, Reactive NUMA: a design for unifying S-COMA and CCNUMA, in: Proceedings of the 24th Annual International Symposium on Computer Architecture, 1997, pp. 229-240 (June).

[3] J. R. McCombs, R. T. Mills, and A. Stathopoulos. Dynamic load balancing of an iterative eigensolver on networks of heterogeneous clusters. In Proceedings of the 17th International Parallel and Distributed Processing Symposium, to appear, 2003.

[4] M. Corrˆea, R. Chanin, A. Sales, R. Scheer, and A. F. Zorzo. Multilevel Load Balancing in NUMA Computers. Technical Report TR 049, PUCRS, Porto Alegre, 2005. http://www.inf.pucrs.br/tr/tr049.pdf.

[5] Geist G.A. and Sunderam V. S. The PVM System: Supercomputer level concurrent computation on a heterogenous network of workstations. In Proceedings of the Sixth IEEE Distributed Memory Computing Conference, 3, 1991.

[6] Laudon J, Lenoski D. The SGI Origin: A ccNUMA Highly Scalable Server. In: Proceedings of the 24th International Symposium on Computer Architecture (ISCA'97), Denver, Colorado, 1997:241-251

[7] Brock B C, Carpenter G D. Experience with Building a Commodity Intel-based ccNUMA System. IBM Journal of Research and Development, 2001, 45(2).

[8] Verghese B, Devine S, Gupta A. Operating System Support for Improving Data Locality on ccNUMA Compute Servers. Proc. of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge (USA), 1996:279-289.

[9] AMD. Bios and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron Processors. April 2004.

[10] AMD. HyperTransport Technology I/O Link: A High-Bandwidth I/O Architecture. July 2001.

[11] Don Anderson, Jay Trodden. HyperTransportTM System Architecture. Colorado Springs, Co: M indShare.Inc.2003.

[12] HyperTransport I/O Link Specification. http:// www.hypertransport.org/.

[13] Eric Krause. HyperTransportTM technology and infiniBand architecture:The complete high bandwidth IO so.http: //www.techonline. com/community/related_content/21070,2002:01-10.

[14] A. Joseph, J. Pete, and R. Alistair, "Exploring Thread and Memory Placement on NUMA Architectures: Solaris and Linux, UltraSPARC/FirePlane and Opteron/HyperTransport," 2006, pp. 338–352.

[15] C. P. Ribeiro, M. Castro, L. G. Fernandes, A. Carissimi, and J.-F. M´ehaut, "Memory Affinity for Hierarchical Shared Memory Multiprocessors," in 21st International Symposium on Computer Architecture and High Performance Computing SBACPAD. S˜ao Paulo, Brazil: IEEE, 2009.

[16] D. S. Nikolopoulos, E. Artiaga, E. Ayguad´e, and J. Labarta, "Exploiting Memory Affinity in OpenMP Through Schedule Reuse," SIGARCH Computer Architecture News, vol. 29, no. 5, pp. 49–55, 2001.

[17] F. Broquedis, N. Furmento, B. Goglin, R. Namyst, and P.A. Wacrenier, "Dynamic Task and Data Placement over NUMA Architectures: an OpenMP Runtime Perspective," in 5th International Workshop on OpenMP, IWOMP 2009, ser. Lecture Notes in Computer Science, vol. 5568. Dresden, Germany: Springer, Jun. 2009, pp. 79–92.

[18] C. Pousa Ribeiro and J.F. M´ehaut, "Minas: Memory Affinity Management Framework," INRIA, Research Report RR-7051, 2009. [Online]. Available: http://hal.inria.fr/inria-00421546/en/.

[19] C. P. Ribeiro and J.F. M´ehaut, "Minas Project Memory affinity management System," 2009. [Online]. Available: http://pousa.christiane.googlepages.com/Minas.

[20] M. Castro, L. G. Fernandes, C. P. Ribeiro, J.-F. M´ehaut, and M. S. de Aguiar, "NUMA-ICTM: A Parallel Version of ICTM Exploiting Memory Placement Strategies for NUMA Machines," PDSEC'09: Parallel and Distributed Processing Symposium, International, pp. 1–8, 2009.

[21] F. Dupros, C. P. Ribeiro, A. Carissimi, and J.-F. M´ehaut, "Parallel Simulations of Seismic Wave Propagation on NUMA Architectures," in ParCo'09: International Conference on Parallel Computing, Lyon, France, 2009.

[22] D.R.Kaeli, L.L.Fong, R.C.Booth, et a1.Performance analysis on a CC-NUMA prototype.IBM Journal of Research and Development,1997,41(3):205-214.

[23] R. Barr, and B. Hickman. Reporting computational experiments with paralle algorithms: Issues, measures and experts‘ opinions. ORSA Journal of Computer, 1993, (5) :2-18.

[24] Akl, S. G. Parallel Sorting Algorithms. Orlando, FL: Academic Press, 1985.

[25] Babb, Robert G. II. Introduction. In Robert G. Babb II, (ed.), Programming Parallel Processors, pages 1-6. Reading, MA: Addison-Wesley,1988.

[26] Tarek S. Abdelrahman and Thomas N. Wong. Distributed array data management on NUMA multiprocessors. In Proceedings of SHPCC, pages 551–559, May 1994.

[27] B. Falsafi, D.A. Wood, Reactive NUMA: a design for unifying S-COMA and CCNUMA, in: Proceedings of the 24th Annual International Symposium on Computer Architecture, 1997, pp. 229-240 (June).