Analyzing the Characteristics of Memory Subsystem on Two Different 8-Way NUMA Architectures

Qiuming Luo[1,2], Yuanyuan Zhou[1], Chang Kong[1], Guoqiang Liu[1], Ye Cai[1,2,*], and Xiao-Hui Lin[3]

[1] National High Performance Computing Center (NHPCC), SZU, China
[2] College of Computer Science and Software Engineering, SZU, China
[3] Department of Communication Engineering, SZU, China
{lqm,caiye}@szu.edu.cn,
clarkong89@gmail.com

Abstract. Two NUMA architectures with different memory subsystems are experimentally analyzed in this paper. By applying the benchmark with various access patterns, it shows much different characteristics of memory system between Xeon E5620 with Global Queue and LS 3A with typical crossbar switch. The experiment results reveal the fact that LS 3A and Xeon E5620 have some similar features. Our study also showed some other diverse features of these two platforms: due to the different contention locations and mechanisms, the memory access model for E5620 doesn't fit for LS 3A. Through comparing, we find that one advantage of LS 3A is that it can obtain steady bandwidth on both local and remote thread, and it is more fair for local and remote access under some circumstances. Another fact is that LS 3A is not such sensitive to remote access, compared with E5620, so there will be no obvious performance degradation caused by non-local memory access.

Keywords: Memory subsystem, NUMA, Global Queue, Crossbar switch.

1 Introduction

With the increasing number of processor cores on one single machine, memory bandwidth has become the main bottleneck of computer system. Instead of using faster and bigger processor caches, NUMA has reduced the memory bandwidth issue by using asymmetric hierarchical memory model. A NUMA system contains some processors, caches, memory controllers and memory banks, using connection technologies provided by AMD's HT (HyperTransport)[1] or Intel's QPI (Quick Path Inter connect) [2] to connect with each other. Due to the NUMA factor, when a process access local memory or remote memory, they have different access delay and bandwidth. To obtain the optimal performance, previous optimization work on NUMA platform often schedule the process, move data from remote to local to maximize the

[*] Corresponding author.

local accesses. Some of these are based on analysis after execution [3][4][5], while others can dynamic deal with it during execution [6][7][8][9][10].

Instead of just observing local or remote memory access, the studies have made deep analyze of the inner memory controller architecture in recent two years. And they found that under some circumstance decreasing data locality may procure better performance [11][12]. This kind of study involves the detail behaviors and architecture of memory controller and it is more applicable to real application environment. After all, entirely local or remote access is not common in real applications. When we study the memory bandwidth on NUMA platform we need to consider both memory layout and the contention of local and remote memory access. We need to understand its architecture features through experiments to take full advantage of these CPU's underlying hardware power.

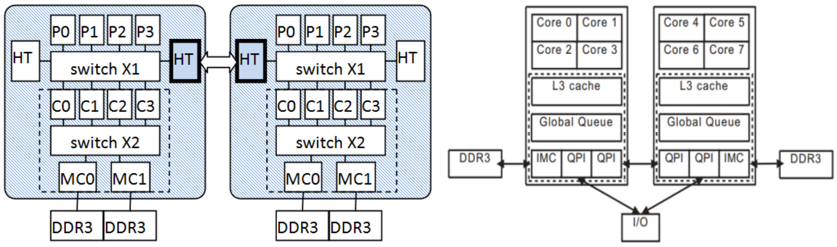
In this paper, we test two NUMA platforms, consisted of LS 3A processors and Intel Xeon E5620 processors, to characterize their memory performance. Then we analyze their similarities and differences, merits and shortcoming based on their difference architectures. At last, we propose a guideline to obtain best performance when using NUMA system consisted of these two processors.

2 Experimental Setup

In this section we describe the architecture details of the evaluation systems, the benchmark programs, and the experimental methodology used.

2.1 Hardware

LS 3A used in our work is an experimental CPU made by Institute of Computing Technology (Chinese Academy of Sciences). The NUMA platform made by LS 3A is consisted of dual-processors and it use HT to connect with each other. As shown in Fig.1(a), for each processor, the first level crossbar switch X1 connected with four 64-bit superscalar GS464 high performance processor cores (P0-P3) ， four 1MB shared second level cache (we can also call it LLC, Last Level Cache), the two ports of HT used to connect with IO or other processors. The second level crossbar switch X2 used to connect LLC and 2 memory controllers MC0 and MC1, which also called IMC(Integrated Memory Controller). Each MC is connected with 2GB DDR2 memory, and the total memory of the system is 8GB. The frequency and width of the HT on this main board is about 200MHz and 8-bit (it can work on 800MHz and 16-bit). Two 64-bits 400MHz DDR2 controller can provide total bandwidth about 6.4GB/s. The 8 processor cores will share the L2 cache. When doing remote memory access, they go through the remote crossbar switch X1 and X2. Intel named LLC, local and remote arbitrate queue and IMC as uncore unit, and it corresponds to the part surrounded by dotted line square in Fig.1.



(a) LS 3A (b) E5620

Fig. 1. NUMA system consisted of dual-processors

In this architecture, a memory access may reach different components, such as local cache, remote LLC (L2), local memory or remote memory. And IMC also can deal with request from other nodes. So they need the support of request queuing and arbitrating mechanism. According to the official material, LS 3A use the HT port to send remote memory access request and it will contend the X1 switch network with native memory access request issued by the 4 local processor cores (include direct access to DDR and those access the LLC). And there is no detail explanation about how it arbitrates the request, so we need to figure out its inner mechanism through experiments. But from the architecture we can tell it is much different about remote memory access and local memory access path from Intel Xeon E5620. The later NUMA system is illustrated as Fig.1(b), the QPI connection, processor core and LLC layout is different from LS 3A. In E5620, the processor cores shared the L3 cache, GQ (Global Queue) is used for dealing with request from L3 cache, IMC and QPI. IMC have 4 channels and is used to connect 4 DDR3-1066 (2GB each, the total memory is 8GB). It can provide 25.6GB/s peak bandwidth. Two QPIs separately used for connecting with I/O and processors, the unidirectional peak bandwidth is 11.72GB/s and the bidirectional peak bandwidth is 23.44GB/s

The uncore unit in Intel E5620 processor is shown in Fig.2. It includes 3 components, LLC, IMC and QPI, and connected with each other with GQ. In GQ component (based on crossbar switch), there are 3 queues to deal with the miss request and update request from L2 cache and the remote access request from QPI. These queues' length is 32 items, 16 items and 12 items respectively[13]. Since every memory request goes through GQ, there are many contentions among local memory request and remote memory request on it. And GQ can reserve about 50% bandwidth for remote memory access request (This is the maximum bandwidth of remote memory access). Early version of E5500 reserves less than 50% of total bandwidth.

PMU (Performance Monitoring Unit) can used for observing and analyzing performance issues in Intel E5620 [13]. This paper used Intel E5620 for comparison, the same tests will perform on both platform.

Cache plays an important role in memory tests. In order to measure the bandwidth, we need to take advantage of cache and reduce the interference of cache. The detail information of cache on LS 3A is described in [14]. LS 3A have implement a

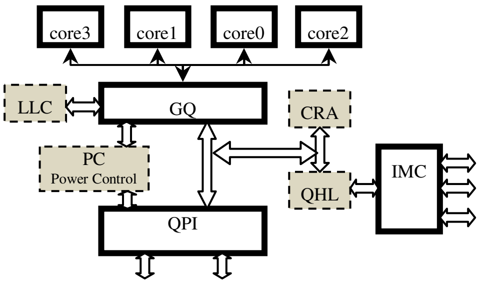


Fig. 2. Intel E5620 uncore unit

non-blocking cache technology. It can support up to 24 misses in LS 3A's cache. The Westmere (Nehalem's successor) core of Intel E5620 has private L1 and L2 cache, four Westmere cores of E5620 shared the L3 cache (LLC). Its non-blocking technology can support up to 16 caches misses. Since LS and Intel have different cache levels, capacity and pre-fetch mechanism, we need to avoid their influence about measuring the bandwidth.

2.2 Software and Benchmarks

The Linux kernel on LS 3A platform is Linux 2.6.36, the compiler version is GCC 4.4.5 and the compile parameter is gcc –fopenmp –O3. The Intel platform use Linux 2.6.32 as its kernel and the compiler version is also GCC 4.4.5, the compile parameter is gcc –fopenmp –O3 –mcmodel=medium.

This paper use STREAM [15] to test the memory access performance of each processor core, each processor and the whole system. Traid has an intensive memory access cycle. It accesses 3 arrays (a[], b[] and c[]). Their sizes all exceed the size of LLC, so they can issue enough memory access requests. The kernel cycle of Triad is showed in Fig.3.

```
for (i = 0; i < ARRAY_SIZE; i++)
{
   a[i] = b[i] + SCALAR * c[i];
}
```

Fig. 3. kernel cycle of Triad

In fact, an instance or thread is not able to saturate the maximum bandwidth of IMC, so we can't just use one Triad program to study the memory access characteristic of Intel or LS NUMA systems. Besides, we need to study the memory contention of local request and remote request. We must use at least 2 Triad instances to fulfill our work. So we made some modification on Triad and made it can produce multiple processes or threads to execute concurrently. Through using OpenMP, we can use

For direction and static schedule direction to parallelize it, the threads can share the 3 arrays and access their data set alone at the same time. The threads share the arrays, but the processes didn't share any data.

The advantage of Triad is described as below: The first one is that it is cache-starve-style[17] application and the cache will have steady missing rate, each Triad thread will not influence the cache missing of LLC[16]. The second advantage is that 94% to 99%'s read operation will reach the main memory [12], it will not be impacted by confliction on cache line.

2.3 Measurement and Methodology

We use numactl [18] tool to control the data distribution on every node's memory and bind the threads to corresponding nodes.

To study the interference and contention on IMC and cross-process connection (HT/QPI), we defined three configurations, as fig.4 shows, about the layout of threads and data. The first configuration put threads in both L(Local, on node0) site and R(Remote, on node1) site, and put all data in L site. The second configuration put threads and data on both nodes in a cross accessing style. The third configuration put threads on L site and issue memory access to both nodes.

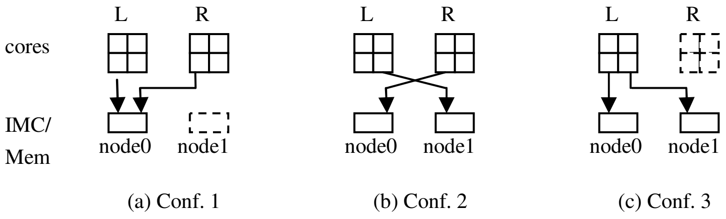


Fig. 4. Threads and data layout configurations

When measuring the memory access bandwidth, it needs to obtain the total number of memory access that arrived IMC and the time elapsed. But we can't get the total amount of memory access direct for the program or source code. The Intel's processor hardware counter can give accurate cache line size, missing rate and processor frequency [12]. But we haven't got the proper tools for LS 3A to do this, so we use the execution time of Triad as a reference. This is mainly due to Triad has a steady memory access workload, and every data only accessed once, this kind of application's execution time is decided by memory access operation[13]. In addition, both LS 3A and E5620 processor's L1 instruction cache have enough capacity to hold Triad's main cycle instructions, which means fetching instruction will not occupy the memory controller's bandwidth. So we can compute the bandwidth as below:

$$\text{Bandwidth}_{\text{total}} = \text{Bandwidth}_{\text{local}} + \text{Bandwidth}_{\text{remote}}$$

$$\text{Bandwidth}_{\text{local(or remote)}} = \frac{\text{Memory Access Size}}{\text{Exe Time}}(\text{MB/s})$$

Due to local process and remote process need different time to finish the workload, the contention will not exist when one of them exit. We execute the processes repeatedly to keep the bandwidth contention. Therefore, the date we get is under contention.

The Intel's Nehalem micro architecture use MESIF protocol to keep cache coherent, different access states in cache line will cause different access delay. The states of LS 3A are INV 、SHD and EXC, L2 cache use directory to find cache stay in which processor core. Since the modified Triad code doesn't share data, cache line will only exist M, E and I state, we don't have to take the delay variation into account.

At last, Nehalem micro architecture' SpeedStep and Turbo Burst technology [13] can change the clock frequency according to power consumption and fever situation. Therefore we close this function in BIOS to exclude its influence and provide a steady easy measure system environment.

3 Experimental Data of Memory Performance

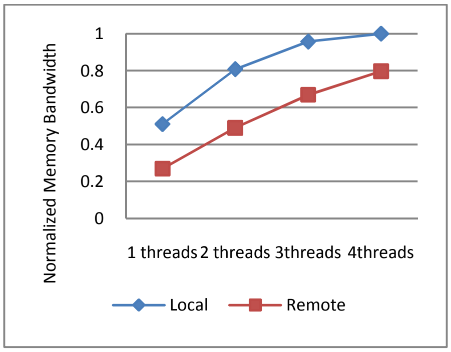
We divide the work into several parts. The first test, using Conf.1 configuration, is to measure the local or remote bandwidth from threads on one single node, on both LS and Intel platform. The second one (still using Conf.1) then combines the local and remote access together to show the contention on one IMC. The third test is based on Conf.3 which focus on the total bandwidth of whole system based on the same configuration as previous tests. The forth test is based on Conf.2 which setup a cross accessing scenery. The last one study the confliction of local and remote access issued from same node, using Conf.3. The details are described in following sections.

3.1 Local and Remote Bandwidth for One Single IMC

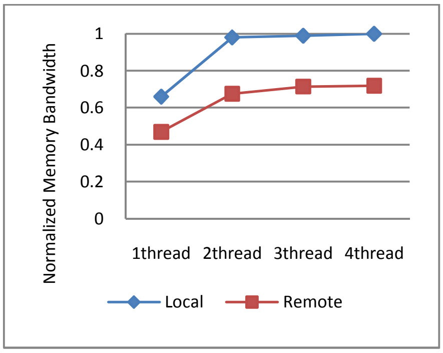
In order to study the behavior of IMC serving the local and remote requests, the first test using conf. 1. In this section (xL,yR) represents the threads binding, x represents the number of local threads (on node0) and y represents the number of remote threads (on node1). There will be x+y threads executing concurrently.

The setup of (xL,0R), which means no remote access, is used to measure the local accessing. From Fig.7(a) we can see the bandwidth of serving the local access on LS platform continues to increase till it reaches the maximum thread number on signal node. And the remote memory bandwidth (with setup of 0L,yR) is about 20% less than local access bandwidth. On Intel platform, it nearly reaches the maximum bandwidth with 2 threads, and the maximum bandwidth of remote memory is 20%~30% less than local memory bandwidth.

The data show that LS can obtain increasing bandwidth with more threads, but Intel can reach the maximum bandwidth with fewer threads.



(a) LS 3A NUMA

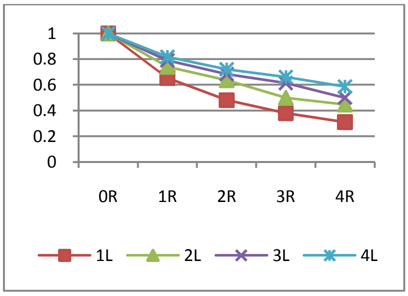


(b) Intel E5620 NUMA

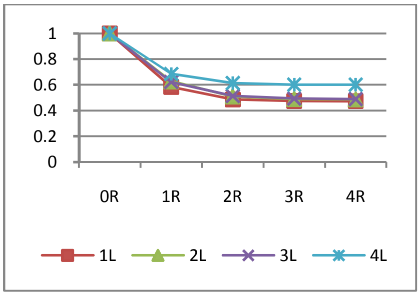
Fig. 5. The memory bandwidth of local and remote access with signal node

3.2 Local and Remote Memory Contention on One IMC

Now we test the memory contention for local and remote memory access on one node under configuration 1. In order to check out how remote access interferes with local bandwidth, we set xL from 1 to 4 with yR increased from 0 to 4 and illustrate it as Fig.6. From the figure we can see the bandwidth decreased steady with more remote access on LS 3A NUMA. But on Intel E5620 the local bandwidth decreased dramatically with 1 remote thread, and the local bandwidth decreased little when there are more than 1 remote threads. Another difference is that more local threads will obtain more local bandwidth on LS platform, while Intel platform remains the same except a little increment for 4 local threads.



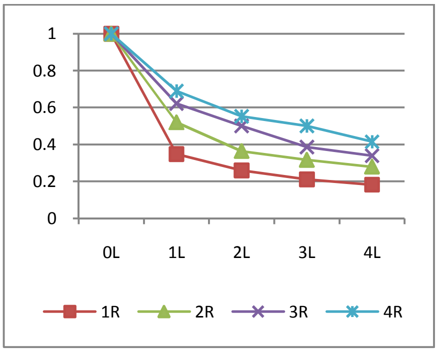
(a) LS 3A NUMA



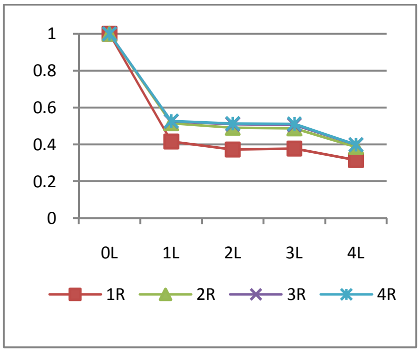
(b) Intel E5620 NUMA

Fig. 6. The impact of remote access to local bandwidth

We also measure the remote bandwidth impact by local threads. The details are showed in Fig.7, where remote thread number varies from 1 to 4 with local thread number varies from 0 to 4. It can say that this case is similar to previous case.



(a) LS 3A NUMA



(b) Intel E5620 NUMA

Fig. 7. Remote bandwidth impact by local memory contention

3.3 Total Memory Bandwidth

Table 1 is the total memory bandwidth with different (xL,yR) combination under configuration 1. The total memory bandwidth in this table is the normalized sum of local and remote bandwidth.

These 2 tables show that on both platform the maximum bandwidth is not at (4L,4R). The maximum for LS platform is at (4L, 3R) and the Intel platform is at (3L,2R). If there are 6 threads, the maximum bandwidth for LS is not at (4L,2R) but at (3L,3R), it is similar on both platforms. From the data we can find that on both NUMA platforms, maximizing local access can't obtain the maximum bandwidth, and when move some threads to remote node may increase the total bandwidth.

Table 1. Normalized total bandwidth with different (xL, yR) combination

(a) LS 3A NUMA

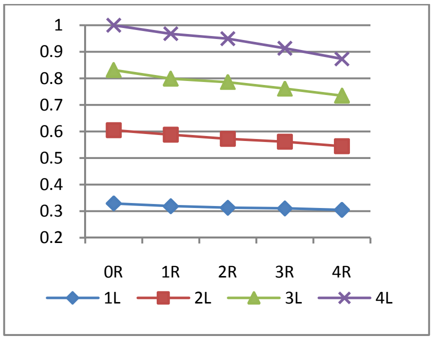
	0R	1R	2R	3R	4R
0L	0	0.268604753	0.489641936	0.669026213	0.796305483
1L	0.510155182	0.76435979	0.93206135	1.071639386	1.167414268
2L	0.807668329	1.077136439	1.21477007	1.061157669	1.327363384
3L	0.957279374	1.221851671	1.294655035	1.428549427	1.026310918
4L	1	1.244672139	1.395384964	1.480886933	1.458042771

(b) Intel E5620 NUMA

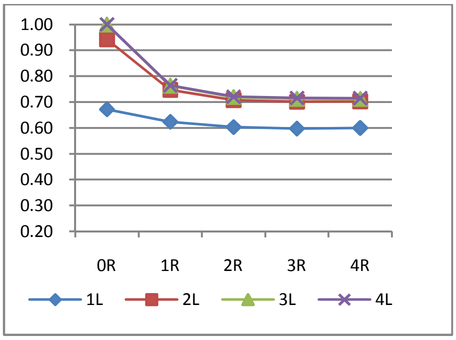
	0R	1R	2R	3R	4R
0L	0	0.469004798	0.675381934	0.714149411	0.718478895
1L	0.660250795	1.034590501	1.073531914	1.056589576	1.047122689
2L	0.980231839	1.205814996	1.253073808	1.245351318	1.237903597
3L	0.989410094	1.18881594	1.260403052	1.252844415	1.243145613
4L	1	1.059521274	1.057766793	1.029016999	1.021363832

3.4 Cross Pattern

We place the threads and the data they access on different nodes according to configuration2, which is defined by in Fig.4(b), to check out how the remote access of node0 and node1 interfere with each other. The detail is showed in Fig.8. On Intel's platform, when there is one opposite thread, the referenced threads' bandwidth will decrease about 30% (except for the case of 1 local threads), and there is not bandwidth lost when the remote thread number increase further. And when there is only one referenced thread, the bandwidth will decrease less than 5%. On LS Platform, the opposite thread impact is much less, with the maximum no more than 10%. This test shows that LS's remote access's influence is much smaller than that of Intel.



(a) LS

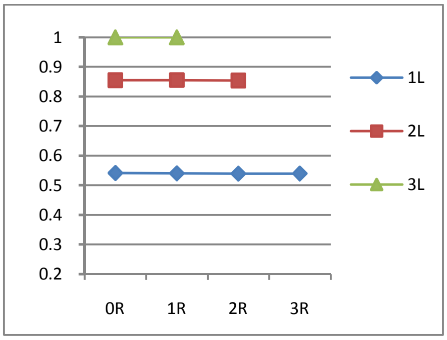


(b) Intel

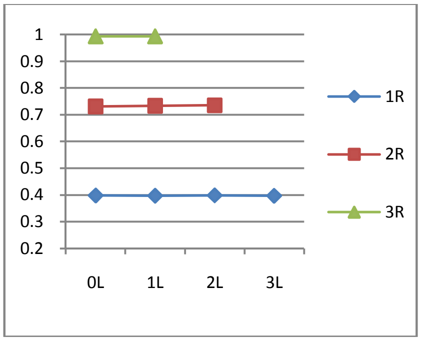
Fig. 8. Bandwidth for cross pattern

3.5 Split Pattern

The last test setup is according to configuration 3 in Fig.4(c), and we want to check out how the local and remote access of the threads on the same node interferes with each other. We call this access pattern as split pattern. The result is showed in Fig.9.

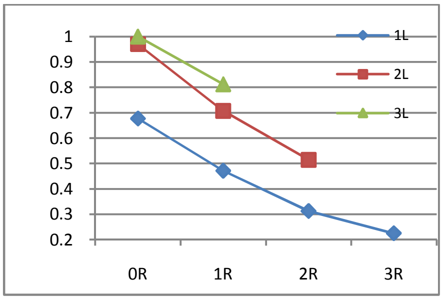


(a)LS local

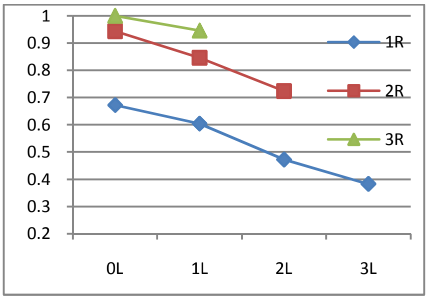


(b) LS remote

Fig. 9. Bandwidth for split pattern



(c) Intel local



(d) Intel remote

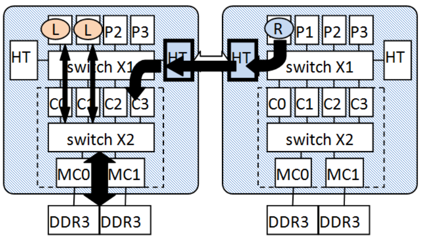
Fig. 9. (*continued*)

The Fig.9(a) and Fig.9(b) showed that LS NUMA platform have little impact on each other. But Fig.9 (c) and Fig.9(d) shows that, on Intel NUMA platform, there is continually impact on local accesses when increasing the remote accesses. The result from split pattern apparently shows that the optimization on Intel's platform will not fit for LS platform.

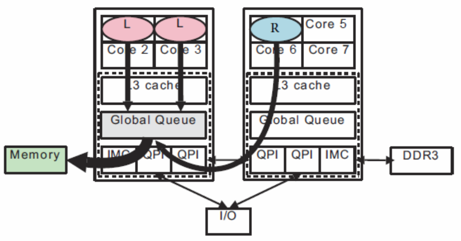
4 Discussion and Conclusion

In this section, we first discuss the architectural differences between these NUMAs concerning the memory access, and then try to explain their diverse behavior.

Local threads in E5620 only need to go through GQ and IMC to access the memory, and the remote threads must go through GQ, QPI and IMC to access the memory, as illustrated in Fig.10(b). The LS 3A local access need to go through X1, X2 and IMC, and the remote access must go through X1, HT, X2 and IMC, as shown in Fig.10(a).



(a) LS 3A



(b) E5620

Fig. 10. Local and remote access channels (2L,1R)

When considering the rate to issue memory access, the individual LS core is not such powerful as E5620 core, because the later can saturate the IMC by only two threads/cores as Fig.5 shown. The remote access is much slower than local access, on both platforms, due to the overhead of cross-processor interconnection.

When local and remote accesses compete for the same IMC, the LS demonstrates a equally free competition and Intel demonstrates a arbitrated behavior. As Fig.6 and Fig.7 shows, the local bandwidth on LS will decrease as more remote competitive threads and vice versa, but the Intel limit the tense of competition to some degree. The arbitrative action on Intel platform will reserve 50%~60% IMC bandwidth for local accessing and 40%~50% for remote accessing, not matter how many threads/cores take part in the competition.

The total bandwidth of two IMCs demonstrates the characteristics that previous study have point out[12]. As Tab.1 shows, the best performance occurs at (3L,3R)for LS with 6 threads.

For the cross pattern case, there is little contention on LS platform because of the full duplex HT connections, but an obvious contention on GQ in Intel platform. That explain why there is a dramatic drop for Intel and a slightly decrease for LS in Fig.8.

For the last case of split pattern, there is not variation on LS platform due to the crossbar switch routing, but there is a nearly linear decrease on Intel platform due to the queuing on GQ.

Because of the different philosophies of designing, these two NUMA systems demonstrate apparently diverse behavior of memory subsystem. With GQ might make Intel platform get a better bandwidth and a little longer delay. And the GQ's arbitrative action will contain the competition between local and remote threads. With barely crossbar switch LS allow the free competition and overcome the performance degradation of unnecessary arbitrating (as the cases of cross pattern and split pattern).

When optimizing the memory performance on these platforms, the thread amount, thread binding, and memory layout should be considered with the memory subsystem's characteristics together. The best mapping of thread and data can be found according to the measured data other than barely maximizing the memory locality. Basically speaking, it needs more threads to compete for a higher bandwidth on LS architecture.

Acknowledgement. The research was jointly supported by project grant from Natural Science Foundation of China under the numbers NSFC61003272, NSFC61171071, grant from Foundation of Guangdong Province and Chinese Academy of Sciences (2011A090100037), and grants from Foundation of Shenzhen City under the numbers JCYJ20120613161137326, JCYJ2012061310222457.

References

1. Advanced Micro Devices. AMD HyperTransport Technology-based system architecture (EB/OL). AMD, Sunnyval (May 2002), http://www.amd.com/us/Documents/AMD_HyperTransport_Technology_based_System_Architecture_FINAL2.pdf

2. Maddox, R.A., Singh, G., Safranek, R.J.: A first look at the Intel QuickPath Interconnect (EB/OL). Intel Corporation, Hillsboto (April 28, 2009), http://www.intel.com/intelpress/articles/A_First_Look_at_the_Intelr_QuickPath_Interconnect.pdf
3. Li, H., Tandri, S., Stumm, M., Sevcik, K.C.: Locality and loop scheduling on NUMA multiprocessors. In: International Conference on Parallel Processing (ICPP). IEEE, New York (1993)
4. Marathe, J., Mueller, F.: Hardware profile-guided automatic page placement for ccNUMA systems. In: Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). ACM, New York (2006)
5. McCurdy, C., Vetter, J.C.: Memphis: Finding and fixing NUMA-related performance problems on multi-core platforms. In: International Symposium on Performance Analysis of Systems & Software (ISPASS). IEEE, New York (2010)
6. Ogasawara, T.: NUMA-aware memory manager with dominant-thread-based copying GC. In: Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA). ACM, New York (2009)
7. Tikir, M.M., Hollingsworth, J.K.: NUMA-aware Java heaps for server applications. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, Colorado (2005)
8. Tikir, M.M., Hollingsworth, J.K.: Hardware monitors for dynamic page migration. Journal of Parallel and Distributed Computing 68(9), 1186–1200 (2008)
9. Verghese, B., Devine, S., Gupta, A., et al.: Operating system support for improving data locality on CC-NUMA computer servers. In: Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, New York (1996)
10. Wilson, K.M., Aglietti, B.B.: Dynamic page placement to improve locality in CC-NUMA multiprocessors for TPC-C. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (SC). ACM/IEEE, New York (2001)
11. Awasthi, M., Nellans, D.W., Sudan, K., et al.: Handling the problems and opportunities posed by multiple on-chip memory controllers. In: 19th International Conference on Parallel Architecture and Compilation Techniques (PACT). ACM, Vienna (2010)
12. Majo, Z., Gross, T.R.: Memory System Performance in a NUMA Multicore Multiprocessor. In: Proceedings of the 4th Annual International Conference on Systems and Storage (SYSTOR). ACM, New York (2011)
13. Levinthal, D.: Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 Processors (EB/OL). Intel Corporation (2009), http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf
14. Wang, H., Gao, X., Chen, Y., Hu, W., et al.: Interconnection of Godson-3 Multi-Core Processor. Journal of Computer Research and Development 45(12), 2001–2010 (2008) (in Chinese)
15. McCalpin, J.D.: Memory bandwidth and machine balance in current high performance computers. IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, 19–25 (1995)
16. Intel Corporation. Intel 64 and IA-32 Architectures Optimization Reference Manual (EB/OL). Intel Corporation (April 2010), http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf
17. Charles, J., Jassi, P., Ananth, N.S., et al.: Evaluation of the Intel Core i7 Turbo Boost feature. In: Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC). IEEE, Washington (2009)
18. Kleen, A.: An NUMA API for Linux (EB/OL) (August 2004), http://www.halobates.de/numacpi3.pdf