Optimization of Uncore Data Flow on NUMA Platform

Qiuming Luo, Yuanyuan Zhou, Chang Kong, Mei Wang, Ye Cai

Optimization of Uncore Data Flow on NUMA Platform

Qiuming Luo[1,2], Yuanyuan Zhou[1], Chang Kong[1], Mei Wang[3]
Ye Cai[1,2*]

[1]Guangdong Province Key Laboratory of Popular High Performance Compters
[2]College of Computer Science and Software Engineering, SZU, China
[3] School of Computer Engineering, Shenzhen Polytechnic, China
lqm@szu.edu.cn, clarkong89@gmail.com, caiye@szu.edu.cn

Abstract. Uncore part of the processor has a profound effect, especially in NUMA systems, since it is used to connect cores, last level caches (LLC), on-chip multiple memory controllers (MCs) and high-speed interconnections. In our previous study, we investigated several benchmarks' data flow in Uncore of Intel Westmere microarchitecture and found that the data flow of Global Queue (GQ) and QuickPath Home Logical (QHL) has serious imbalance and congestion problem. This paper, we aims at the problem of entries' low efficiency in GQ and QHL we set up an M/M/3 Queue Model for GQ and QHL's three trackers' data flow, and then design a Dynamic Entries Management (DEM) mechanism which could improve entries' efficiency dramatically. The model is implemented in Matlab to simulate two different data flow pattern. Experiment results shows that DEM mechanism reduces stall cycles of trackers significantly: DEM reduces almost 60% stall cycles under smooth request sequences; DEM mechanism reduces almost 20~30% stall cycles under burst request sequences.

Keywords: NUMA, Uncore, Data flow, DEM

1 Introduction

The number of processors or cores of PC servers are increasing with the growing demand of performance. The memory conflict becomes more and more serious. Instead of using faster and bigger processor caches, NUMA have solved the memory bandwidth issues to some extent by using asymmetric hierarchical memory model and has become the mainstream of modern server architecture. In a NUMA system, the memory chips are distributed in physics and all this memory shares a global address space. Accessing the remote memory needs processor interconnection technology. There are two main processor interconnection technologies currently. They are, respectively, AMD's HT (Hyper Transport) [2] and Intel's QPI (Quick Path Inter connect) [3]. Obviously, accesses to the local memory are faster than accesses to remote memory, because accesses to remote nodes must traverse the interconnection.

* Corresponding author

Because of this, previous optimization work on NUMA platform often focus on scheduling the process and moving data from remote to local to maximize the local accesses. Some of these are based on performance profile after execution [4][5][6], while others can deal with it during execution dynamically [7][8][9][10][11].

But in recent two years, the studies[12][13][14][15] have shown that microarchitecture have a great effect on optimizing the memory performance on NUMA platforms, even under some circumstance decreasing data locality may procure better performance. In our previous work [14], two 8-way NUMA architectures with different memory subsystems are experimentally analyzed, this two 8-way NUMA systems have diverse memory access feature because of their different microarchitecture. Dashti's work [15] also shows that remote delays are not the most important source of performance overhead, congestion on interconnection links and in MCs can dramatically hurt performance.

In our previous studies [1], we have made deep analyze of the inner architecture of uncore in Westmere processor and explored the data flow of Uncore in NUMA systems and discussed the unbalance and congestion of Uncore traffic, we will describe this later in this article. That unbalance might challenge the fixed entry number of GQ and QHL, which infer a dynamic entries management.

In this paper, we propose DEM (Dynamic Entries Management) mechanism and verified it efficiency by simulating the data flow using Matlab. We conclude that the DEM mechanism improve entries' efficiency dramatically.

The rest of the paper is organized as follows. In section II, we descript uncore modeling. Section III focuses on DEM algorithm description. Section IV presents our experiment results and analysis. Our conclusion is in section V.

2 Unbalance and Congestion of Data Traffic of Uncore

We have used the Intel Westmere as the target NUMA platform. In this section we will introduce the Uncore structure briefly and detail the unbalance and congestion of data traffic of Uncore in Intel "Westmere".

2.1 Microarchitecture of Uncore

Uncore is a term used by Intel to describe the functions of non-in-core parts in a microprocessor. The uncore unit in Intel Westmere is shown in Fig.1. It includes LLC, QPI, QMC and other components. Cache line requests from the local cores or from a remote package or the I/O hub are handled by uncore's GQ. There are 3 trackers in GQ to deal with the requests. One for write requests with 16 entries is called write tracker, the other one is read tracker for read requests with 32 entries, and the last one with 12 entries for other socket requests delivered by the QPI named peer probe tracker. Data access requests that miss the local LLC are sent to the QHL (QPI home logic) unit to retrieve the data. Such requests are speculative by nature, as a hit (m) response to a snoop requests to the other caching agents may return the line more quickly and supersede the request to the local QHL. Again, QHL has 3 queues for requests from

local, remote sockets and IOH, and entries for each queue are 24, 16 and 24, respectively.

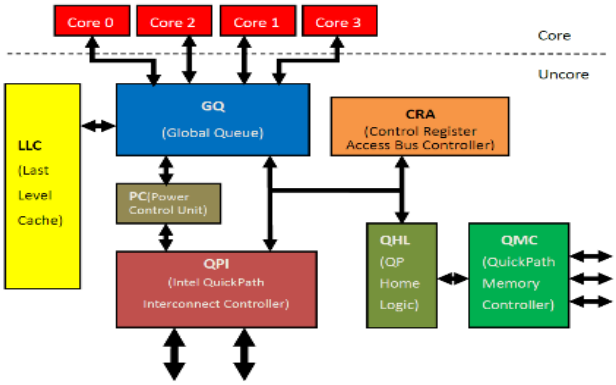

Fig. 1 Details of Westmere Uncore

2.2 Unbalance and Congestion of Data Traffic

In our previous work [1], we used Likwid [18] to measure several hardware performance events which can be classified into two groups: GQ Full and QHL Full, and Full means entries in trackers are used up and new requests should wait for empty entries. We measured the unbalance and congestion of data flow in uncore with using NAS Parallel Benchmark [19] and STREAM Benchmark [20].

After a series of experimental measurements and studies, we found that the unbalance of GQ's and QHL's trackers is serious and the usage rate of entries is low both for GQ and QHL. Such as GQ, Fig.2 shows the unbalance of GQ's three trackers. Each NPB applications run in Class C with 8 threads. We set GQ peer probe tracker's full cycles as 1 and normalized the other trackers' full cycles. Ten is the exponent in Y-axis. The X-axis lists the ten NPB applications. We can see that the unbalance of GQ's three trackers is really serious, the biggest unbalance rate is more than 800 times happened in FT's read and write trackers; the smallest unbalance rate is less than 1/100 times happened in CG's read and write trackers. And similar things happened in QHL.

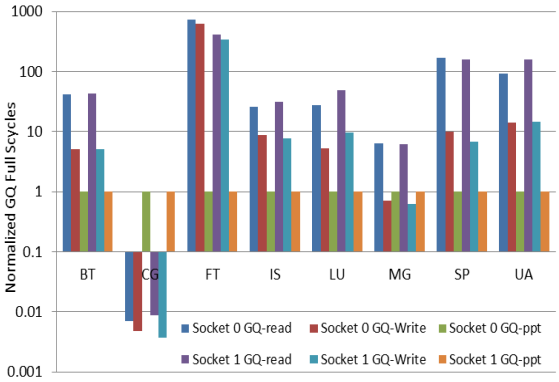

Fig. 2 The unbalance of GQ's three trackers

And also the rate of entries does not match the rate of full cycles of every application because each application's memory access pattern is not the same. As for the congestion of data flow, we found that the Full cycles growth patterns of three trackers were not the same for both GQ and QHL with threads number increase from 2 to 8.

3 Dynamic Entries Management

In this regard, we propose DEM (Dynamic Entries Management) mechanism as shown in Fig.5 and Fig.6, all the entries for the three Tracker are managed uniformly. Compared with FEM (Fixed Entries Management), DEM will improve the efficiency of the usage of the entries and reduce the number of the stall cycles.

3.1 Modeling Trackers of GQ/QHL

The M/M/3 queuing system is a model of the process that customers get service from server desk, the two "M" represents the interval of the customers arrive and the service time needed by customers are exponentially distributed (no memory, random, or Markov property), and the "3" refers to there are three Server Desk. We use the M/M/3 queuing system to simulate the flow of data which go through the GQ or QHL. Both GQ and QHL have three Server Desks to provide services for three Trackers, for GQ, the trackers are Read tracker, Write tracker and Peer Probe tracker. There are Local tracker, Remote tracker and IOH tracker for QHL. The data requests and responses are the customers. Cores, LLC, QPI and QHL will generate customers to GQ. And for QHL, GQ, QPI and QMC will produce customers. Each customer will be stored in an entry through a Tracker, then wait for getting service from one of the three Sever Desks.

In Intel Westmere, the number of entries for each tracker is fixed, as show in Fig.3 and Fig.4. If there is no free entries for one tracker and the newly arriving customers (data requests and responses) who want to enter into that tracker will have to wait. At

the same time, the components generate customers have to stall cycles until a new free entry is available, though the other two Trackers might have free entries. We call this FEM mechanism, as Fig.7 shows.

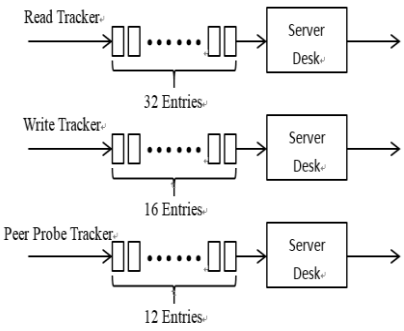

Fig. 3 GQ FEM M/M/3 queuing model

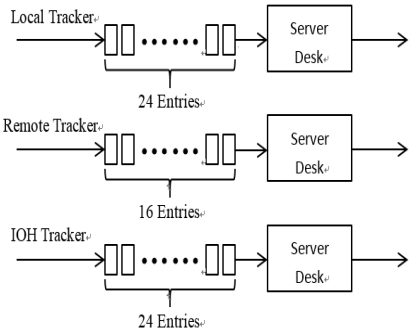

Fig. 4 QHL FEM M/M/3 queuing model

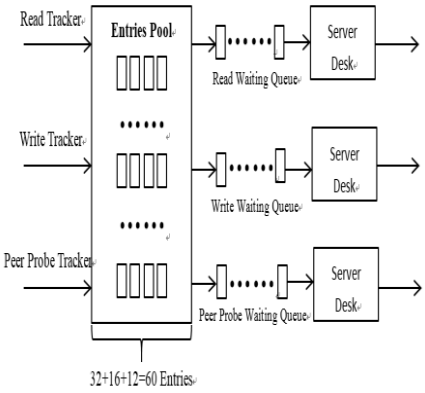

Fig. 5 GQ DEM M/M/3 queuing model

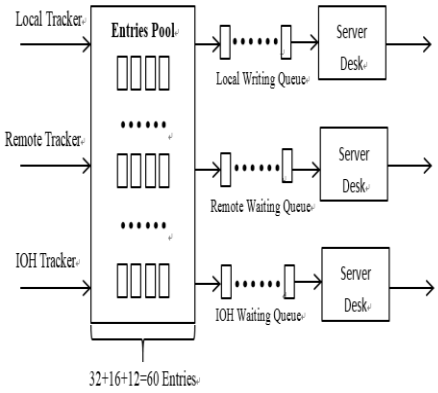

Fig. 6 QHL DEM M/M/3 queuing model

3.2 The implementation of Dynamic Entries Management

In this section we will detail the FEM and DEM algorithm and we take the GQ as an example to illustrate these algorithms. Cores, LLC, QPI and QHL generate data requests or data responses to GQ, we call this a new customers' coming. The customer with different data access type will be stored in free entries through the different tracker. Such the customer with data-read request will go through the Read tracker, and the customer with data-write request will go through the Write tracker. For FEM, the entries divide into three parts according the type of the tracker, the number of entries for read tracker is 32, 16 for write tracker and 12 for peer probe tracker. Therefore, the customers with different type of data access can use the corresponding entries only. For instance, the customer with the type of data read can only be store in one of the 32 entries. All the customers stored in the entries will get

service (the data requests or responses will be handled by GQ) from corresponding Server Desk according FIFO. Flow chart of the FEM algorithm is showed as Fig. 7.

For DEM all the entries are uniform to customer. We treat all the entries as in an entries buffer pool. In order to make the customer get service from the right Server Desk, when a new customer comes, a free entries from the entries poll is assigned to the customer and must mark this assigned entries with the label of the Tracker where the customer from. Then, the customers with the same label form a queue to get the service from Server Desk according FIFO. At the end of the service, the entrie which has been marked must be removed the mark and put back into the entries pool. Flow chart of the FEM algorithm is showed as Fig.8. In this case, just when all the entries in the pool are used up will produce Stall Cycles.

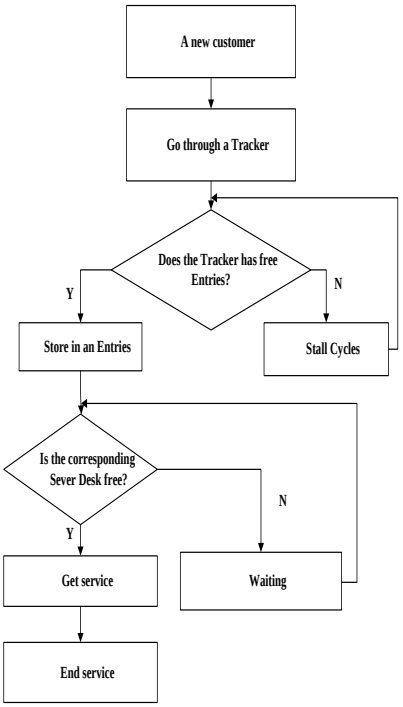

Fig. 7 FEM algorithm

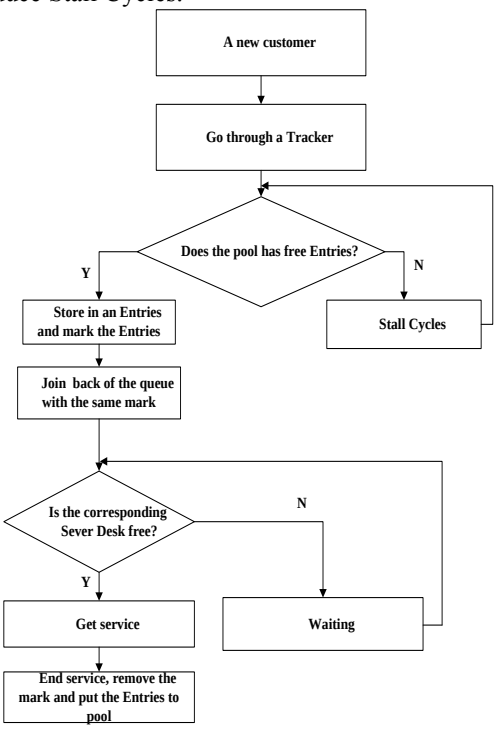

Fig. 8 DEM algorithm

4 Experiment Results

We adopt M/M/3 system to simulate the flow of data which through the GQ or QHL by Matlab. We have assumed that the customers arrive according to a Poisson process, which means the interval time of arrival is exponential distribution. The probability density function of exponential distribution is showed in Formulas (1) and the average of probability density function of exponential distribution $E[x] = 1/\lambda$. We create a parameter **Parameter1** (the value of $1/\lambda$) to represent the average

interval time of arrival, the greater of the value of **Parameter1**, the heavier the pressure of the customer's request. So in this simulation, we adjust the customer's request pressure by set the value of **Parameter1**.

$$f(x) = \lambda e^{-\lambda x} . \quad (1)$$

We also assumed that the time each customer gets service is an exponential distribution, we use a similar parameter **Parameter2** to simulate each customer's average service time, we choose a relative middle value 0.6 for **Parameter2**.

In order to make the comprehensive comparison of the FEM and DEM algorithms, we simulate two scenarios. One scenario is that the customers are generated gently, it means the customer's arrival request pressure is relative stable, another scenario is that the customers arrive with burst mode. The following content is the result of the simulation and analysis of the results.

4.1 Smooth request sequences

For this scenario, we used two sets of parameters, one for the situation that the customer's arrival request pressure is relative low and the entries for each tracker will not be used up, another for the situation that the request pressure is relative high and the entries for some trackers will be used up (for FEM). The parameters shows in Table 1 and Table 2, where the **Customer Number** represents the number of customers, and **Tracker1**, **Tracker2**, and **Tracker3** represent the three Trackers in GQ or QHL respectively.

Table 1 Parameters of low pressure.

	Parameter1	Parameter2	Customer Number
Tracker1	0.5	0.6	200
Tracker2	0.5	0.6	120
Tracker3	0.5	0.6	80

Table 2 Parameters of high pressure.

	Parameter1	Parameter2	Customer Number
Tracker1	0.9	0.6	200
Tracker2	0.9	0.6	120
Tracker3	0.9	0.6	80

The result of the simulation about low pressure is showed in Fig.9. We define the time of the customer's arrival with no stall cycles as the theoretical arrival time or the ideal arrival time. In fact, when the entries for customers is used up , the customer will have to stall some cycles and the following customer's arrival time will be pushed back. We define this arrival time as actual arrival time. The X axis is the sequences of the customers; the Y axis is the time of the customer's arrival in cycles. The red curve is the customer's theoretical arrival time; the blue curve is the customer's actual arrival time using FEM and the green curve for using DEM. As the picture shows, the three curves are coincided, therefore we just can see one green

curve. That means under low pressure, whether use the FEM or DEM, there is no stall cycles occur and the efficiency of this two arithmetic are the same.

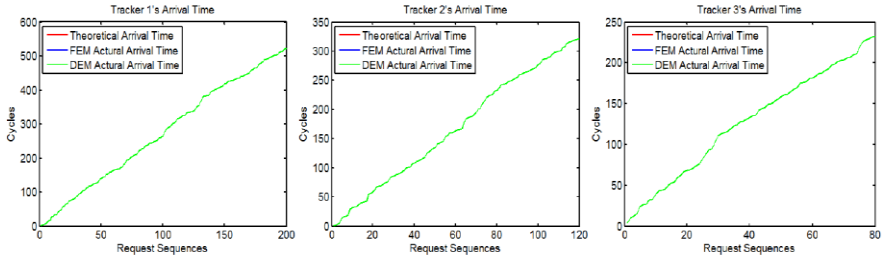

Fig. 9 The time of customer's arrival of the three Trackers under low pressure

Fig.10 shows the results of the simulation about high pressure. We can clearly see that stall cycles occurred for all the three Trackers, because the red curve is lower than other two curves in some time. For Tracker1, both FEM and DEM algorithm can provide enough entries for customers at the beginning, so the green curve covers the blue and red curve in the beginning. Then the blue curve begin to separate upward, this indicates that the entries is slowly be used up with the increase of request of customers and some stall cycles begin to appear for using DEM algorithm. But for FEM, the number of entries for Tracker1 is relative more (32) and it can continue to meet the requests, so the blue curve is coincided with the red curve. With the further increase of the requests of customers, the blue curve begin to separate upward and the growing is faster than green curve. This indicates that DEM algorithm begins to perform better than the FEM algorithm. It is because the other Tracker may have free entries, DEM algorithm can use these free entries for Tracker1. For **Tracker2** and **Tracker3**, because of the entries for them is relative few, the DEM performs decisive advantage over FEM.

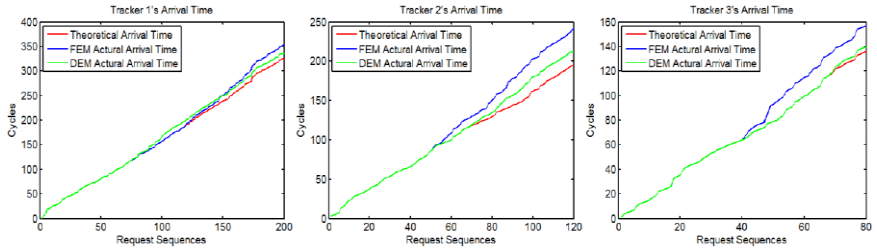

Fig. 10 The time of customer's arrival of the three Trackers under high pressure

We have also counted the average number of stall cycles of each customer of the three Trackers. The result is showed in Fig. 11. Compared with FEM, the DEM reduced the average number of stall cycles by 60 percent.

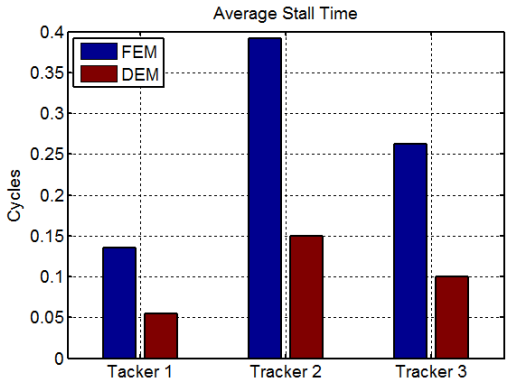

Fig. 11 Average stall time using FEM and DEM of Smooth request sequences

4.2 Burst request sequences

We adjusted the **Parameter1** dynamically to simulate the burst scenario. The lowest black arrow in Fig. 12 shows the trend of customers' arrival when we adjust **parameter1** and the number below the black arrow is the size of **parameter1**. Larger **parameter1** indicates the interval of customers' arrival is smaller, so the decreases of the slope of the curves mean burst requests occur. The parameters setting is shows in the table 3.

Table 3 Parameters of burst mode.

	Parameter1	Parameter2	Customer Number
Tracker1	0.9（10.0）	0.6	200
Tracker2	0.9（10.0）	0.6	120
Tracker3	0.9（10.0）	0.6	80

Compared with Fig.10 and Fig.12, we can find that the trend of the curves is similar to Fig.12. For **Tracker1**, in the same way as high pressure situation, because the entries of **tracker1** is relative more for FEM, the curve of DEM actual arrival time separated upward later earlier than of FEM. For the three trackers, at the end of the burst, the number of stall cycles of FEM begins to overtake those of DEM. what is different is that the blue curves and green curves separate upward earlier in Fig. 12 than in Fig.10. Such as the **Tracker2**, the blue curve separates almost at 30 cycles, while in Fig.10 it separates almost at 100 cycles. This is because the burst occur at beginning of **Tracker2**, so the entries for Tracker2 is used up rapidly.

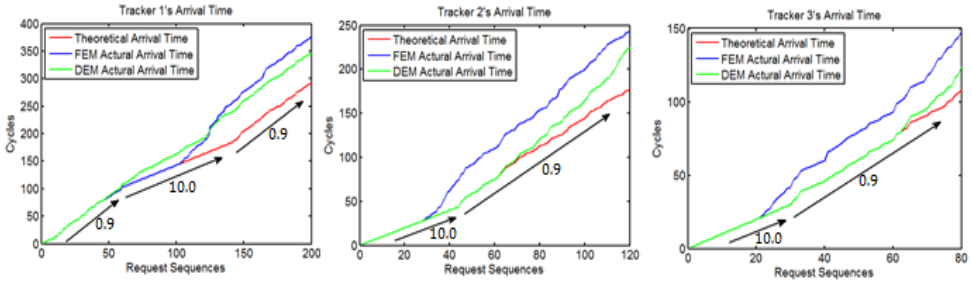

Fig. 12 The time of customer's arrival of the three Trackers under dynamical pressure

For burst mode scenario, the DEM reduced the average number of stall cycles by 20% to 30%. The statistical result is showed in Fig.13.

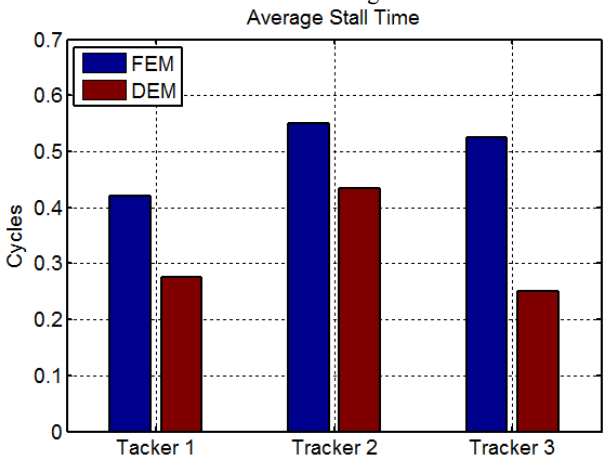

Fig. 13 Average stall time using FEM and DEM of Burst request sequences

5 Conclusion

Uncore component is important for the whole NUMA system, because it is the data transport center. We have found that serious traffic unbalance or congestion happens here when using the current FEM mechanism in our previous studies. In this paper, we proposed DEM mechanism and we used the Matlab to simulate two different data flow pattern of GQ and QHL. According to the experimental results and our analysis , we conclude that the DEM reduce almost 60% stall cycles under smooth request sequences; and reduce almost 20~30% stall cycles under burst request sequences DEM mechanism. Compared with fixed entries number in each tracker, we think that dynamic distributing entries according to each tracker's pressure have more advantages. The DEM could improve entries' usage rate and reduce trackers' stall time, therefore reduce stalling cycles of the preceding out-of-order instruction execution pipeline and improve the performance of the whole NUMA system.

In addition, the implementation of DEM may need some additional hardware to mark the entries and select the entries with the same label to get service by FIFO.

Future work: Our next work is to obtain the memory traces of various multithreaded applications, so we can testify DEM efficiency on "real" memory request sequence other than a Poisson process. Then we are going to design DEM trackers for GQ and QHL in VHDL to verify its functionality and assess its hardware overhead.

Acknowledgement

The research was jointly supported by the following grants: China 863-2012AA010239, NSF-China-61170076, Foundation of Shenzhen City under the numbers JCYJ20120613161137326, JCYJ2012061310222457, and Shenzhen Polytechnic Foundation 601422K20008

References

1. Qiuming Luo, Chang Kong, Yuanyuan Zhou, etc. Understanding the Data Traffic of Uncore in Westmere NUMA Architecture [C] 22th Euromicro International Conference on Parallel, Distributed and network-based Processing. Turin:IEEE, 2014
2. Advanced Micro Devices. AMD HyperTransport Technology-based system architecture [EB/OL]. Sunnyval, CA: AMD. [2002-05]. http://www.amd.com/us/Documents/AMD_HyperTransport_Technology_ based_System_Architecture_FINAL2.pdf
3. Maddox R A, Singh G, Safranek R J. A first look at the Intel QuickPath Interconnect [EB/OL]. Hillsboto, OR: Intel Corporation. [2009-04-28]. http://www.intel.com/intelpress/articles/A_First_Look_ at_the_Intel(r). QuickPath_Interconnect.pdf
4. Li H, Tandri S, Stumm M, Sevcik K C. Locality and loop scheduling on NUMA multiprocessors [C] International Conference on Parallel Processing(ICPP). New York: IEEE, 1993
5. Marathe J, Mueller F. Hardware profile-guided automatic page placement for ccNUMA systems [C] Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP). New York: ACM, 2006
6. McCurdy C, Vetter J C. Memphis: Finding and fixing NUMA-related performance problems on multi-core platforms [C] International Symposium on Performance Analysis of Systems & Software (ISPASS). New York: IEEE, 2010
7. Ogasawara T. NUMA-aware memory manager with dominant-thread-based copying GC [C] Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications (OOPSLA). New York: ACM, 2009
8. Tikir M M, Hollingsworth J K. NUMA-aware Java heaps for server applications [C] Proceedings. 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS). Colorado: IEEE, 2005
9. Tikir M M, Hollingsworth J K. Hardware monitors for dynamic page migration [J]. Journal of Parallel and Distributed Computing, 2008, 68(9):1186–1200
10. Verghese B, Devine S, Gupta A, et al. Operating system support for improving data locality on CC-NUMA computer servers [C] Proceedings of the seventh international conference on Architectural support for programming languages and operating systems(ASPLOS). New York: ACM, 1996
11. Wilson K M, Aglietti B B. Dynamic page placement to improve locality in CC-NUMA multiprocessors for TPC-C [C] Proceedings of the 2001 ACM/IEEE conference on Supercomputing(SC). New York: ACM/IEEE, 2001
12. Awasthi M, Nellans D W, Sudan K, et al. Handling the problems and opportunities posed by multiple on-chip memory controllers [C] 19th International Conference on Parallel Architecture and Compilation Techniques (PACT). Vienna: ACM, 2010
13. Majo Z, Gross T R. Memory System Performance in a NUMA Multicore Multiprocessor [C] Proceedings

of the 4th Annual International Conference on Systems and Storage (SYSTOR). New York: ACM, 2011
14. Qiuming Luo, Yuanyuan Zhou, Chang Kong, etc. Analyzing the Characteristics of Memory Subsystem on Two different 8-way NUMA Architectures [C] the 10th International Conference on Network and parallel Computin. Guiyang: Springer, 2013
15. M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, etc. Traffic management: A holistic approach to memory placement on NUMA systems [C] the 18th International Conference on Architectural Support for Programming Languages and Operating Systems. Houston:ACM, 2013
16. Intel Corporation. Intel 64 and IA-32 Architectures Optimization Reference Manual [EB/OL]. Intel Corporation. [2010-04] .http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf
17. R. Yang, J. Antony, A. Rendell, D. Robson, and P. Strazdins. Profiling directed NUMA optimization on Linux systems: A case study of the Gaussian computational chemistry code. [C] the 25th IEEE International Parallel and Distributed Processing Symposium, Anchorage:IEEE, 2011
18. J. Treibig, M. Meier, G. Hager, and G. Wellein. Poster - LIKWID: Lightweight performance tools. [C] the 2011 High Performance Computing Networking, Storage and Analysis. Seattle:ACM, 2011
19. Nas Prallel Benchmark [CP]. http://www.nas.nasa.gov/publications/npb.html
20. STREAM Benchmark [CP]. http://www.streambench.org/