Overview and Brief Analysis of Multicores Memory Hierarchy

CSCI 6461 Computer System Architecture
Yiang Hu
G46086039
hyaruixi@gwu.edu

Contents

1. Introduction....3
2. Symmetric Cache Structure....4
1.1 Streaming Memory....8
1.2 Coherent Cache....8
1.3 Comparing performance of the two architectures....9
3. Asymmetric Cache Structure....10
3.1 Non-Uniform Cache Architecture....10
3.2 Static NUCA structure....11
3.2.1 Switched Channel Based....12
3.2.2 Private Channel Based....12
3.3 Dynamic NUCA structure....13
3.3.1 Data mapping....14
3.3.2 Data locating....16
3.3.3 Data locating....16
4. Dance Hall Structure....17
5. Cache in Middle Structure....18
5.1 Sharing degree....18
5.2 Mapping algorithm....19
5.3 Moving strategy....20
6. Cooperative Cache Structure....20
7. Consistency of Caches and its correlated solutions....23
7.1 The consistency problem in caches....23
7.2 Solutions for cache consistency....23
7.2.1 Hardware control....23
7.2.2 Software control....25
7.3 Deadlock processing mechanism....26
8. Conclusion....27
Reference....28

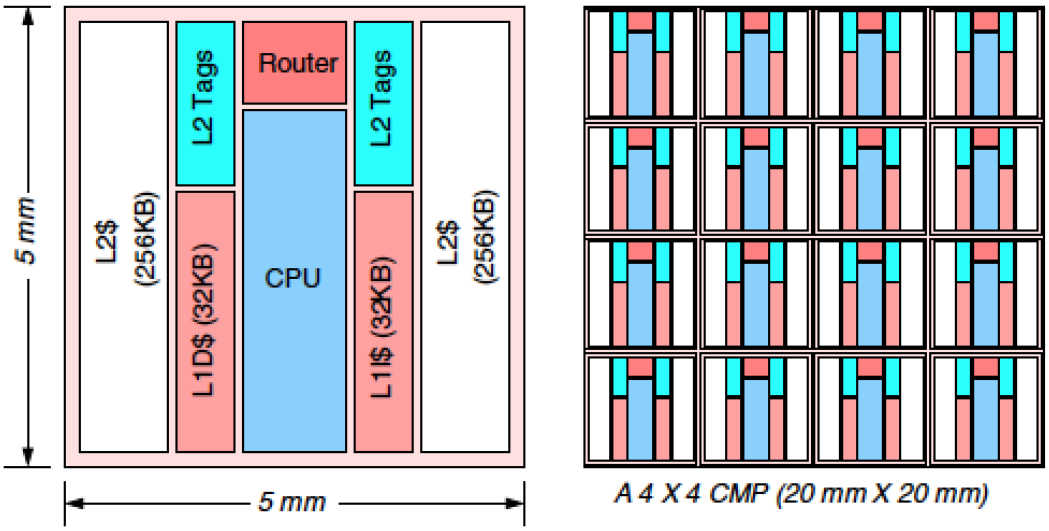
1. Introduction

Microprocessor architecture has dramatically developed in past decades, according to the highly respected Moore's law, the increasing rate would make billions of micro-transistors condensed on a single chip sooner more than imagine. Facing the reality needs and the impossibility of continuous developing, for the fact that integrating all needed micro-transistors as former methodologies would produce the heats even exceeding the surface of the sun, conventional uniprocessors are in need of a revolutionary design and hence give birth to multicore processors. By the satisfying performance preliminary multicore processors displayed and power they consumed, major companies like Intel and AMD brought an overwhelming wave of dual-core microprocessors since the year of 2005.

However, the more developed microprocessors become, the more complicated cache system they possess, and this works would be mainly aim at researching prevalent on-chip cache system design and their advantages respectively. Meanwhile, a brief evaluation regarding differentiated cache architecture of multicore processors conducted and finished by research groups recently would be presented as a reference of how innovative design develops and their relative trends.

Regarding the cache architecture, five major classes with multiple detailed classifications are discussed respectively. Some of the cache architectures would be tested in the following evaluations, which would be researched after understanding all prevailed classes of cache.

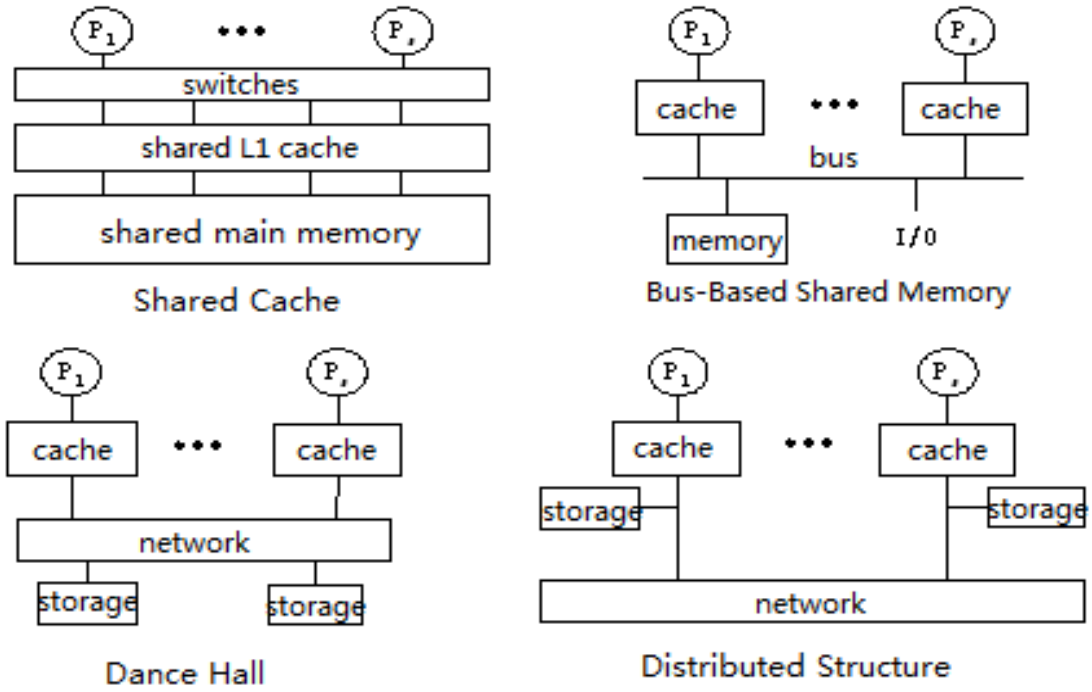
2. Symmetric Cache Structure



Graph 1. A typical Tiled CMP design in a 70 nm process [5]

Considering the universally application in major parallel processing systems of symmetric multiprocessors, it has the characters as below:

1. Every single processor has access to any storage location and related I/O device.
2. All the storage locations of the cache have a unified addressing method with coherent physical addresses.
3. The multilevel cache facilitates the locality of data while its consistency could be realized by hardware.
4. Communication among different processors is achieved by simple read/write instruction that would considerably save clocks in data exchanging.



Graph 2. Cache Architectures of Expanded Symmetric Multi-Processor Device

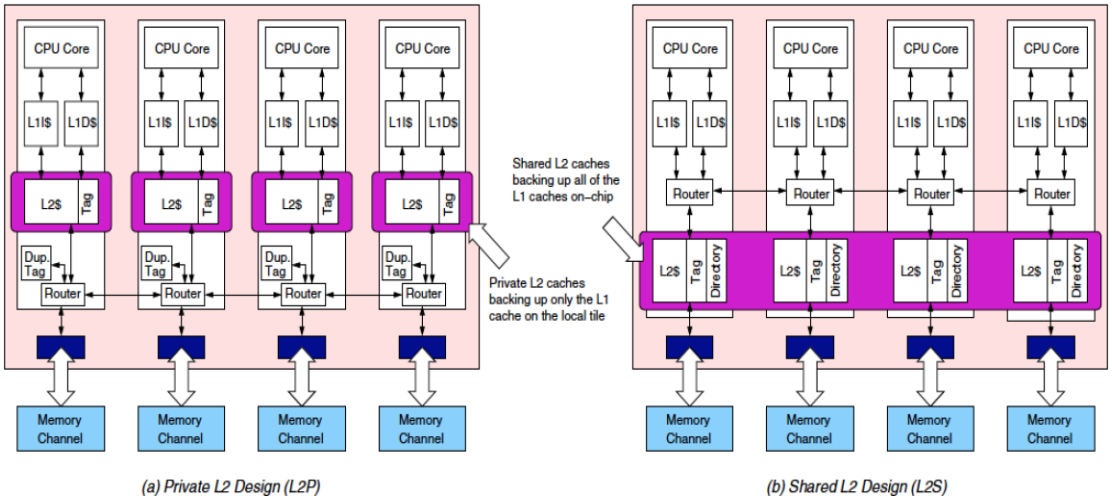
Since caches consume a large aggregate of energy and space in CMP system, it is appropriate to use multi-level cache to maximize its locality.

L1 caches reside in the nearest location of the data path toward processors, it taking care of most of processor requests with the shortest delay and quickest access speed. However, it may prolong the access delay for multi-core processors if merely enlarge the capacity of L1 caches with multiple accessible ports for processors use due to the sequence of access caused by read and write inflict and needs of realizing consistency. Therefore the private L1 cache in processor core is usually small, meanwhile the privacy guarantees that there is no need to go through the internal bus to achieve data from L1 cache.

As for K2 cache, there are two different types of schemes: private L2 cache, and shared L2 cache. Private L2 Cache Scheme: Evenly assigned the spaces to every

processor with each of them could access their own K2 cache without the necessity of going through the bus. The process could be deemed as shrinking a previous processor with multi-chips into one condensed chip. If L2 cache misses, processor has to visit L2 caches in other processors through internal bus to check if they have the needed data, if not, try storages outside the chip.

Shared L2 Cache Scheme: All processors share the bus between L1 cache and L2 cache, L2 cache guarantees the data consistency of L1 cache. If L1 cache misses, search the L2 cache; if L2 misses too, and then search storages outside the chip.



Graph 3. The two L2 cache strategies [1]

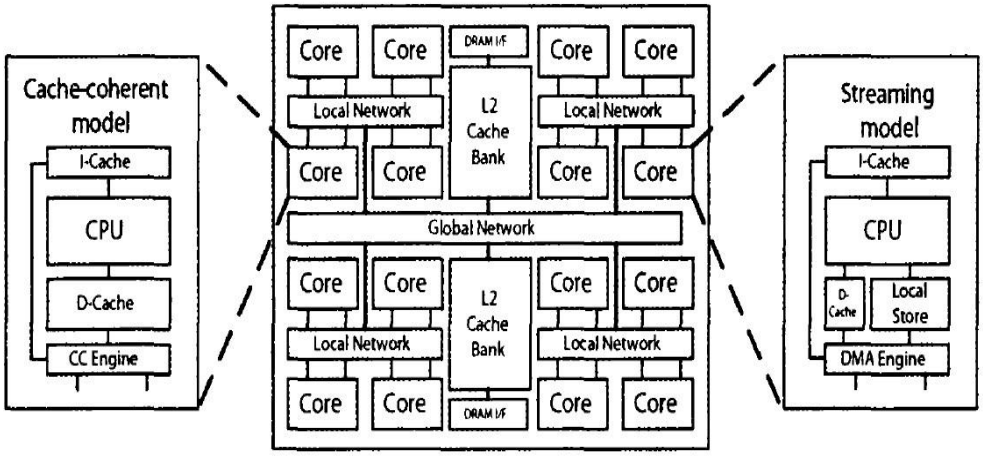
Both scheme have their merits while suffers from different disadvantages.

In private L2 cache, the cache unit is close to the processor and it provides convenience especially when the most visit which do not need to make request from the bus by saving the delay and bus contention. However, the same problem is nonexistent in shared L2 cache strategy. It is reasonable to regard the shared L2 cache as a huge entire cache in which no copy of data block contained, similarly, there is no unbalanced loading issue as well.

Traditional shared L2 cache structure treated the L2 cache as an entirety that ultimately causes a large dynamic consumption when processing access operations. Currently we divide the L2 cache into multiple isolated independently addressed blocks, which connected and shared through cross addressing method.

In multi-core and multi-threading processors, some specific caches are assigned for processor resource utility improvement and visiting loads reducing, i.e. introducing Trace cache concept in multi-core multithreading processors to make the physically unconnected blocks be continuous, so as to facilitate the speed of preloading. Hence a larger instruction stream amount could be provided and reduced the pressure of bandwidth from storage level. Improve the predicting hit rate of cache by applying address-transferring cache.

There are two major types of symmetric multiprocessors currently, Coherent Cache and Streaming Memory. In comparing the two models, we should clear that they share the same core structure, and the relative parameters shows in the form below.



Graph 4. CMP Structure with 16 multi-cores [3]

	Cache-Coherent Model (CC)	Streaming Model (STR)
Core	1, 2, 4, 8, or **16** Tensilica LX cores, 3-way VLIW, 7-stage pipeline **800MHz**, 1.6GHz, 3.2GHz, or 6.4GHz clock frequency 2 FPUs, 2 integer units, 1 load/store unit	
I-cache	16KB, 2-way associative, 32-byte blocks, 1 port	
Data Storage	32KB, 2-way associative cache 32-byte blocks, 1 port, MESI Hardware stream prefetcher	24KB local store, 1 port 8KB, 2-way associative cache, 32-byte blocks, 1 port DMA engine with 16 outstanding accesses
Local Network	bidirectional bus, 32 bytes wide, 2 cycle latency (after arbitration)	
Global Crossbar	1 input and output port per cluster or L2 bank, 16 bytes wide, 2.5ns latency (pipelined)	
L2-cache	**512KB**, 16-way set associative, 1 port, 2.2ns access latency, non-inclusive	
DRAM	One memory channel at 1.6GB/s, **3.2GB/s**, 6.4GB/s, or 12.8GB/s; 70ns random access latency	

Chart 1. Parameter of two types of caches [3]

1.1 Streaming Memory

In Streaming Memory structure, part of on-chip storage resources is assigned to independent and program-addressable local storage, and the data exchange with storages not on chips needs controls of programs and DMA operations.

In this very model, the core of a processor reads and retrieves data from local storage, which then makes the exchange with outside storages through DMA operations.

Through the given table, we notice that the structure of I-cache is same as cache consistency model whereas its L1 storage was divided into 24KB local storage and 8KB D-Cache. A relatively small D-Cache for stack and external variable storage would be very helpful for simplifying stream programming.

1.2 Coherent Cache

All the on-chip resources are assigned to be either part of private cache in the processor or shared cache among different processors, and maintain the consistency of data by hardware. The core of processor implicitly operating read and writes processes

in cache, whose controller then reads and replaces data in cache by units of blocks from storages that are not on-chip (memory).

All processes of cache data retrieving and storing, access synchronizing and data consistency are realized by hardware, which deprived the interference from programmers. Meanwhile, it is the memory that the programs can directly addressing.

1.3 Comparing performance of the two architectures

For parallel data applications with massive data reuse, both streaming memory and coherent cache have an efficient quality and expandability. Cache and software control could effectively save the delay of storage accessing in the local storage by data locality. If the cache consistency model applied the same non-writing strategy, streaming memory may not has the advantage in energy saving as they both applying applications in other types.

However, streaming memory structure has a better performance for its preloading character especially in the capacity of processors. Meanwhile, coherent cache can assign the preloading function to external hardware so as to achieve similar quality. There do exist some occasions that streaming model performs worse than coherent cache.

Regarding storage structures, streaming model does not exceed coherent cache which based mainly on hardware, they have the same quality and consumption. The programming work for streaming model may require a careful coordination in communication and local details.

3. Asymmetric Cache Structure

The introduction of cache is intended to fix the delay of accessing internal storages in processing operations in need of speed. By instructions, locality of space and time in data streams, the performance could be dramatically improved. However, since the capacity of cache is the key factor influencing the efficiency of cache and it is currently limited by manufacturing technologies, protocols like replacement algorithm, reducing the hit time of cache, preloading etc. are invented.

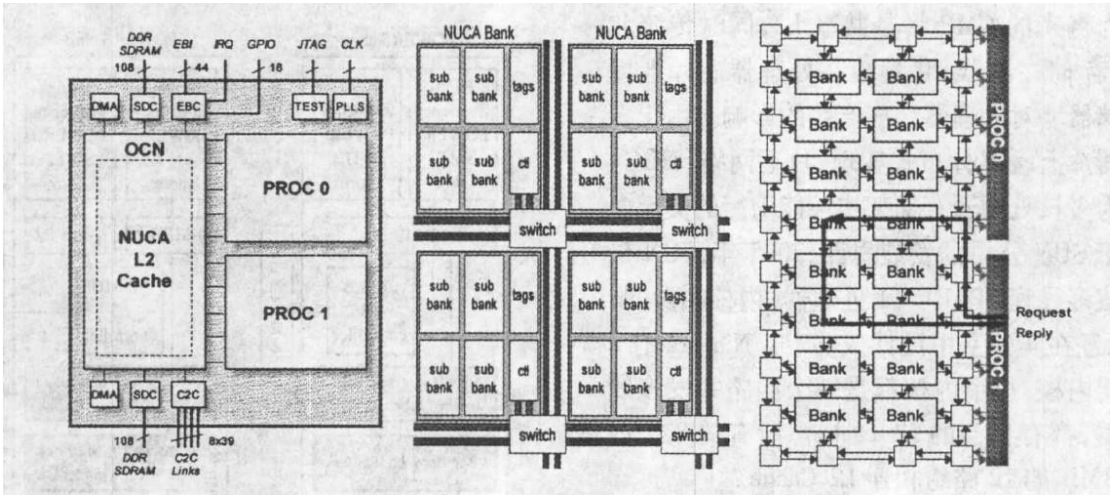
Asymmetric cache structure firstly presented by Changkyu Kim and his co-workers in paper of Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches for ameliorating the cache dilemma.

3.1 Non-Uniform Cache Architecture

With the expansion of Cache, the access delay got unprecedentedly influenced by the whereabouts of it is stored. Those who are close to processors get a much more ideal response performance than those who reside far from processors, i.e. in 50mm manufacture circumstances, the L2 cache in 16M chip, the difference between accessing the nearest and far most data units could be 43 clocks![2] In previous uniformed architecture structures, the delay is purely decided by the time needed to reach the far most point, which would be 47 clocks, and it would considerably interfere the overall performance of the processor by wasting time in waiting for responses.[2]

The structure of Non-Uniform Cache Architecture is designed specifically caters to this situation. The NUCA re-organizing the L2 cache into multiple cache banks with

each of which have a different time to reach the processor. Thus when the processor needs to reach a nearer location on the cache, the delay would not counted as the far most location, hence the response time could be saved by the distance where data located and time would be much more shorter facing a massive processing needs.



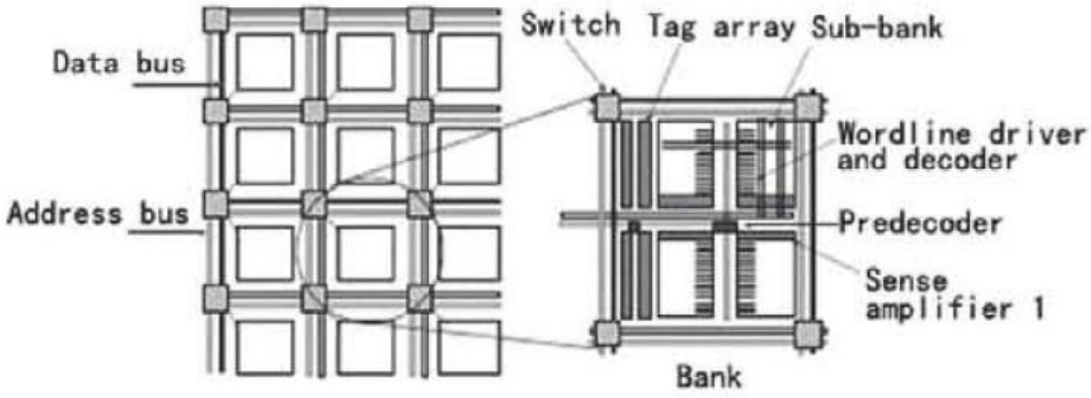
Graph 5. TRIPS chip of NUCA structure [3]

3.2 Static NUCA structure

In static Non-Uniform Cache Architecture, every cache bank is capable of independent addressing and data could be mapped into certain banks according to its static addresses in lower storages. With different location of data could be achieved by different time and addresses, the advantage of applying NUCA is acknowledged by its ability of parallel processing and low access delay. Static NUCA structure has two different types according to its accessing ways: Private Channel Based and Switched Channel Based.

3.2.1 Switched Channel Based

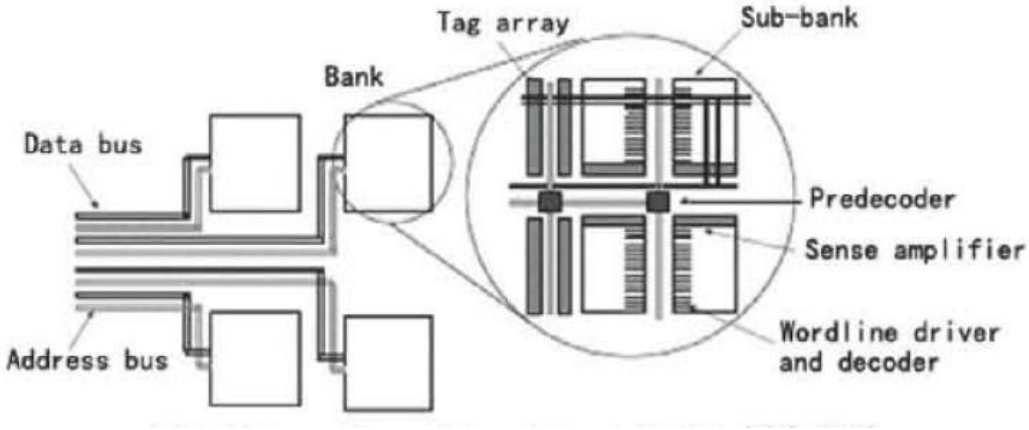
In switched channel base, there is no specific channel for any certain bank, instead, it is connected by a 2-D mesh network and exchange data between cache controller and banks by packet switching. Saving spaces for chips, it may cause jam and network loading serving for application with high data density.



Graph 6. Switched Channel Based Static NUCA Structure [4]

3.2.2 Private Channel Based

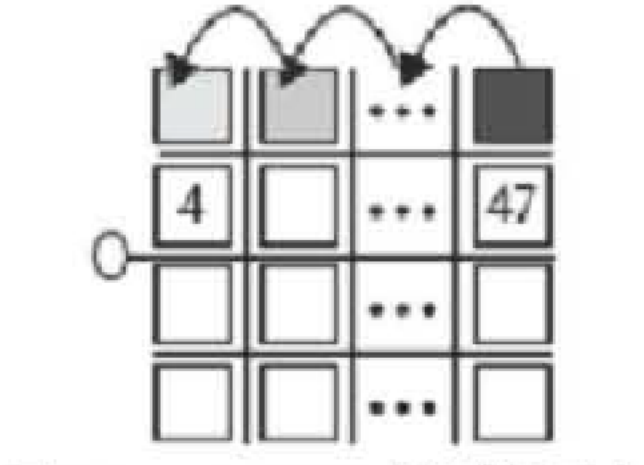
In private channel structure, there are two specific data channels for each cache bank so as to operate read and write, therefore the independence of read and write of each bank get guaranteed while the parallelism of data retrieving gets improved. However when the division of caches is too small, there have to be numerous of data channels for bank catering, hence the space of chips have to be further expanded.



Graph 7. Private Channel Based Static NUCA Structure [4]

3.3 Dynamic NUCA structure

In dynamic NUCA, data is not stored in a static cache bank, it follows a certain strategy to monitor its location and exchange from one cache bank to another so as to make sure that popular data resides in cache banks near the controller. In the very process, three aspects should be taken into account: mapping, locating and movement.



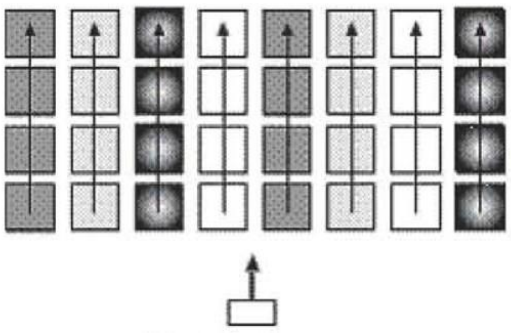
Graph 8. Data Moving in Dynamic NUCA Structure [4]

3.3.1 Data mapping

Date mapping allows data imported from lower storage devices is available in cache banks and make decision of its whereabouts. It may perhaps completely confine the data in one specific location and stored just like static NUCA structure, or it could be continuously switching locations that makes addressing and hardware consumption maximized.

Kim proposed a compromising strategy that combines traditional associated mapping with NUCA structure. That is, to divide all banks into multiple groups, each data set could only be mapped into one bank group and each byte should be in different bank, and the pending data switching could only be achieved inside the bank group. The demand of mapping strategy correlated with bank amounts in every group differed into three types of mapping: simple mapping, fair mapping and shared mapping.

a. Simple mapping

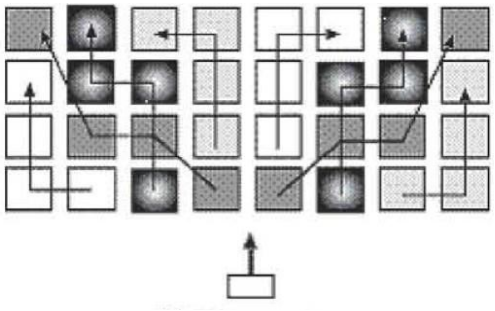


Graph 9. Scheme of Simple Mapping [4]

The cache bank is arrayed into two-dimensional array with each column serve as a bank group. When processing data search operations, controller could get the needed data by order of group to bank to byte and retrieve. However, there is no guarantee that

the number of rows is matched with group associated mapping strategy and their accessing delay still varies.

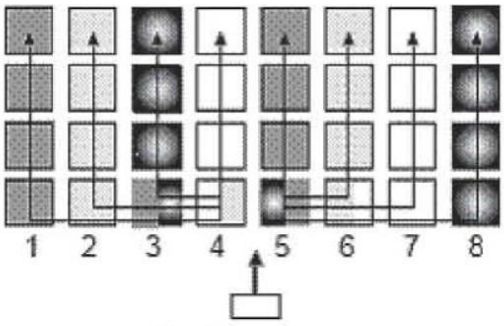
b. Fair mapping



Graph 10. Scheme of Fair Mapping [4]

Catering to the shortcomings in simple mapping, idea of fair mapping comes for equaling every the average accessing delay of each bank group. With unified bank group delay, it suffers from a complicated read and write path.

c. Shared mapping



Graph 11. Scheme of Shared Mapping [4]

Shared mapping strives to achieve the shortest access delay by allowing data switching in the banks that are closest to the controller. It comes with effective performance whereas the rows and columns of bank arrays have to be strictly matched with mapping strategy.

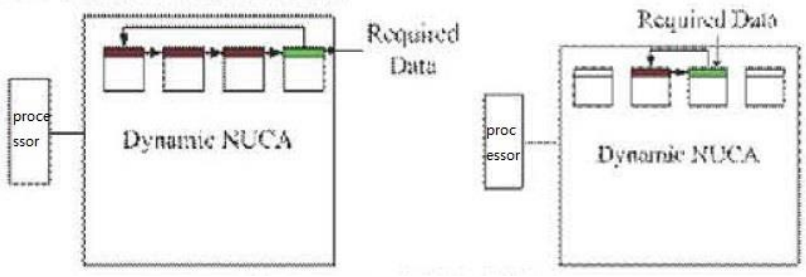
3.3.2 Data locating

Data locating refers to pinpoint the location of data in cache, there are two major types of searches: a. Incremental search, search from the nearest location of cache controller until get the data, or gives a miss signal by the end of banks; b. Multicast search, the addressing signal is broadcasted into multiple or all banks in bank groups and operating the search procedure.

3.3.3 Data locating

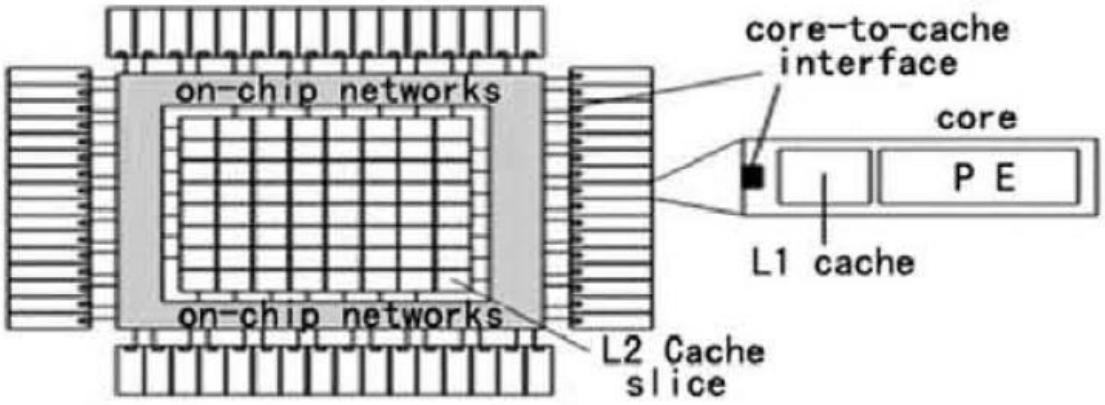
Dynamic NUCA aims in making all cache accesses occur in cache bank with shortest distance with cache controller, hence movements inside the cache should make sure that recent accessed data are resided in nearest banks. The commonly applied conventional LRU algorithm that saves the recently used data in nearest cache controller may not be applicable, because it would cause too many data switches.

According to traditional LRU algorithm, if the far most data needs to be retrieved, all the data should be associated to move, which would bring loads of pressures to the chip networks. Therefore, the promoted LRU algorithm proposed by Kim allows the hit data moves only one bank toward the controller. It somehow reduce the delay by moving it closer step by step while avoid to pressure too much on networks.



Graph 12. Scheme of Data Movement Strategy [4]

4. Dance Hall Structure



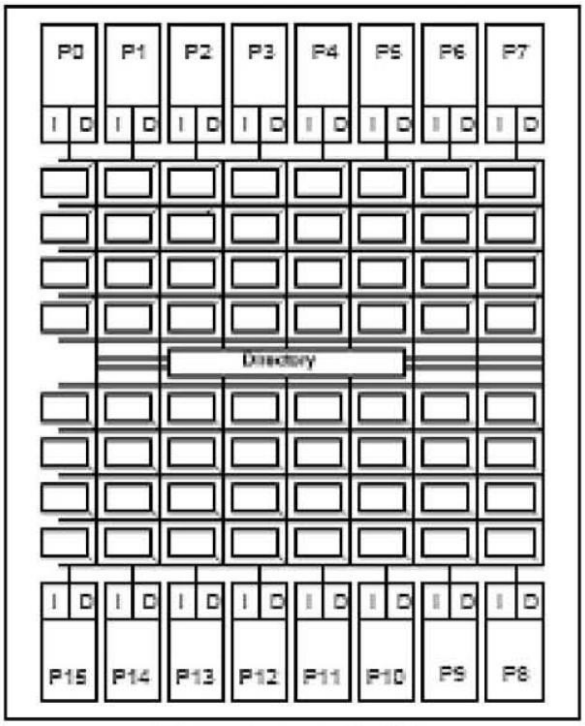
Graph 13. Dance Hall Structure [4]

The dance hall structure is a completely shared L2 cache structure. It seems concise and simple as it allows operation of “direct write” and “write behind” to maintain the consistency of L1 and L2 cache data. It is possible to apply Directory Access Protocol to insure the consistency of L1 cache. In face of the complete sharing of L2 cache for all processors, the best utilization of L2 spaces would achieve a high hit rate.

However, the wire delay may occur during the read operation from L2 caches with different distances to the processor. In dance hall structure, the delay caused by wire delay is considerably high and it pulls the overall efficiency of the cache. Beckmann and his colleagues presented a method to reduce the influence of wire delay in his paper <Managing Wire Delay in Large Chip-Multiprocessor Caches>. In the strategy he proposed, control of data block movement places the frequently used data block in location that is near the controller. However, this methodology would somehow alleviate the situation as it seems, merit of the problem remains the same because the processor could not anticipate the popular blocks. [6]

5. Cache in Middle Structure

In research of Jachyuk Huh <A NUCA Substrate for Flexible CMP Cache Sharing>, a new idea of dividing L2 cache into multiple cache banks and connected through internet with processors with cache arrays located in center of the chip surrounded by processor core. [7]



Graph 14. Scheme of Cache in Middle [4]

Focus of multi-core L2 cache is different from mono-core NUCA in various aspects.

5.1 Sharing degree

Sharing degree decides when different caches have been distributed in banks, every processor core's authority of accessing all banks or limited numbers of banks.

When sharing degree set as 1, each processor core could only access the L2 cache for their own, which could be treated as each processor core has a private cache; when sharing degree is 2, two processor cores could share part of cache banks; hence when the sharing degree equals to number of processor cores, every core have the authority to access all cache banks.

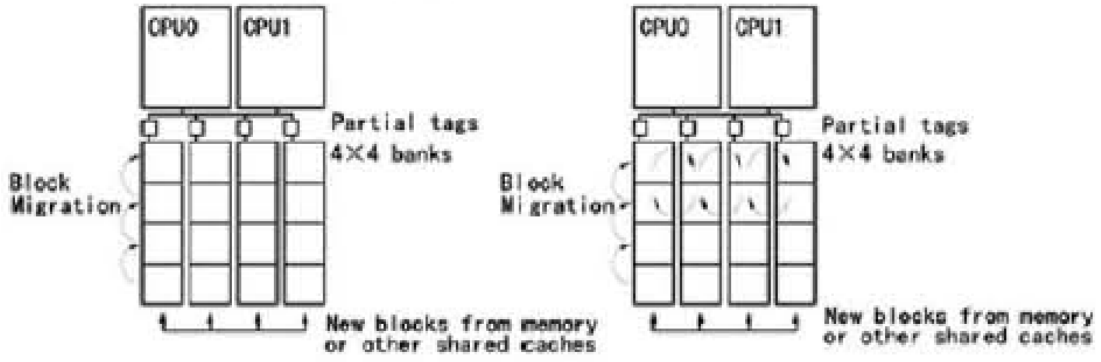
Protocol of sharing degree mainly influences the performance of cache in its hit rate, hit latency, inter-core data transmission and complexity of consistency maintaining. Reduced sharing degree would alleviate accessing delay while a high sharing degree would improve hit rate and reduce inter-core communication. The maintaining the consistency of L2 would be complex if sharing degree decrease whereas it would be complex for L1 if sharing degree increases.

Besides, Jaehyuk Huh and his colleagues have proved that when sharing degree is 4, the overall performance of the system achieve the best optimized result and all parameters could reach a balanced status.

5.2 Mapping algorithm

While dynamic mapping could complement the shortcoming in static mapping strategy in mono-core NUCA, multi-core NUCA has its problem in dynamic mapping, i.e. when multiple cores trying to access the same data, moving inflict may occur and data would shifting between two cores; besides, traditional central tag may not applicable on multi-core processors for the impossibility of closing to every core in tag arrays.

5.3 Moving strategy



Graph 15. Strategy of Date Moving

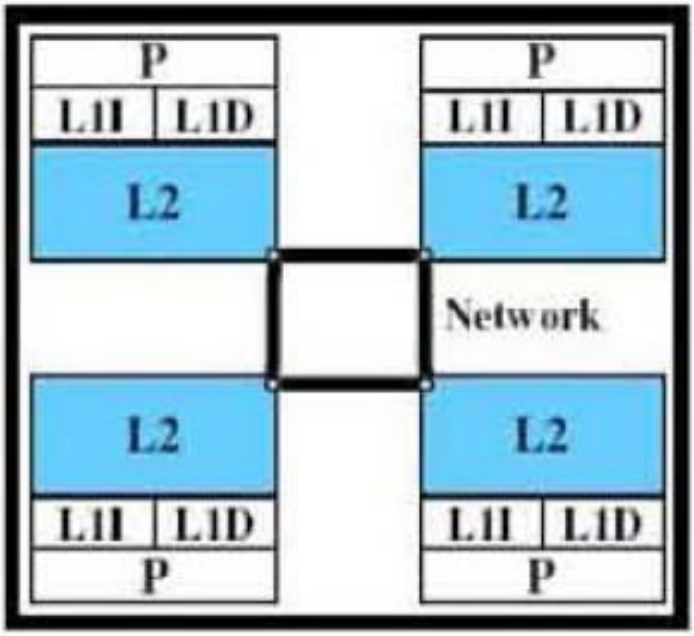
There is only one allowed data moving direction in mono-core NUCA because the mono-direction movement would be sufficient for controlling the distance of data and controller. However in multi-core situation, data blocks would move in numerous axes according to requests from processors, hence the moving strategy should be redesigned and ameliorated to make this demand met.

6. Cooperative Cache Structure

In research of Jichuan Chang, <Cooperative Caching for Chip Multiprocessors>, a new structure of cooperative cache (CC) has been proposed. The cooperative cache structure based on private L2 cache structure and allows every processor has access to every private L2 caches in the same chip. Thus by storing frequently used data in local private cache, it may store the needed data in L2 cache of other processors is local private cache has reach its extreme of hit rate. This very method could guarantee the frequently used data has a relatively low hit delay while achieve a satisfying hit rate by using other local caches from other processors, therefore the cost of retrieving data from

memory especially regarding the delay from memory compared with wire delay between caches. [8]

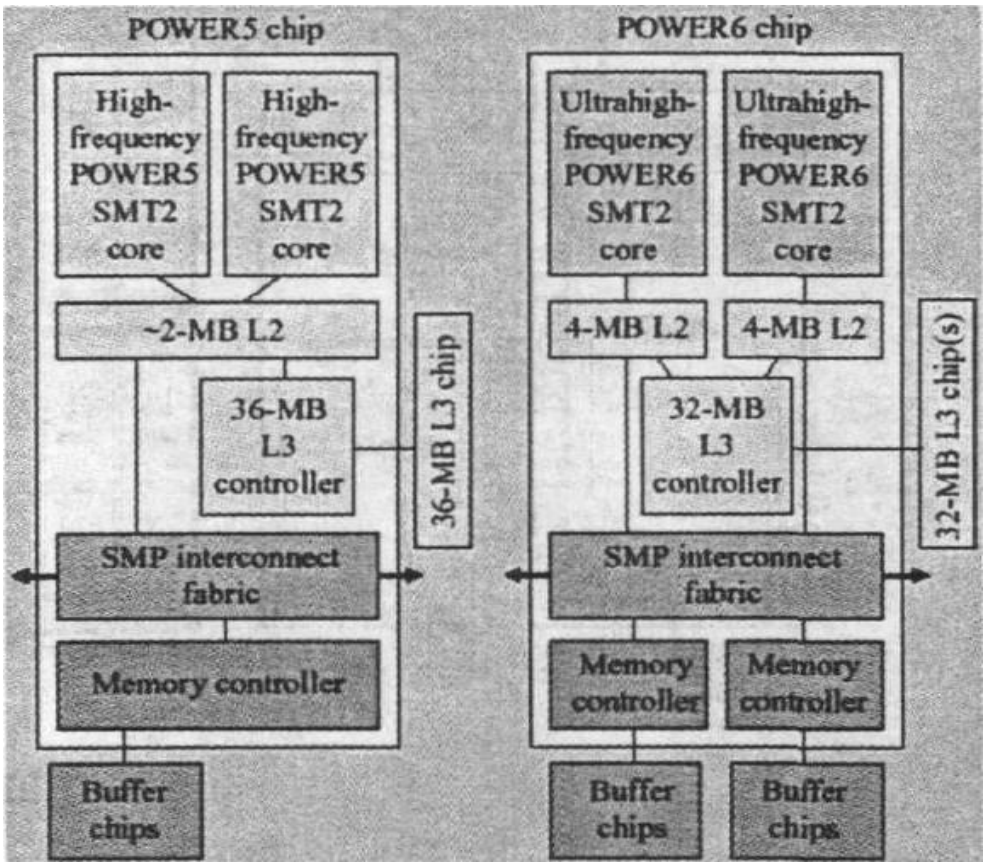
In Michael Zhang and his colleagues' paper <Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors>, an idea similar to cooperative cache structure while the operating detail differs, we consider that to be the same set of structural designs.



Graph 16. Cooperative Cache Structure

However, the balancing problem of cache resource sharing in dynamic cache still remains. Especially when processor A directly uses the local private L2 cache while the L2 resources in processor B is also in need, it would drastically influence the efficiency of processor B. In some certain situation, the coordinating strategy applied on cache system may leads to a thrashing, in which major part of system times are used on cache replacement. Has the system provides sufficient support in both software and hardware for cache spaces and storage band width allocation.

In research of Bradford M and his colleagues <Adaptive Selective Replication for CMP Caches>, there is an ameliorating method proposed. Because of the static nature of the algorithm in moving data to local L2 caches under Cooperative Cache structure, data moving may decrease hit delay whereas the increasing copies of data would not only decrease the hit rate of L2, but also increases the complexity of maintaining data consistency. [9] Hence applying static algorithm of data moving may perhaps drag down the overall performance. The ASR dynamic algorithm allows the data moving only in face of the occurrence of the overall performance of low wire delay outwitted the loss of low L2 hit rate. This very method improves the performance by complicated protocol, hence when the scale of CMP gets larger and larger, the shortcoming of this method appears more and clearer.



Graph 17. Structure of optimized chips of POWER 5 and POWER6 issued by IBM[3]

7. Consistency of Caches and its correlated solutions

7.1 The consistency problem in caches

The multi-processor contains shared and private data in its cache, the former used by single processor while the latter used by multiple processors. For any change in cache, if the related data belongs to shared data, it may has copies in other caches, if the newly updated data has not synchronized in other caches timely, there may be error when other processors access the copies of data. Therein lies the issue of data consistency.

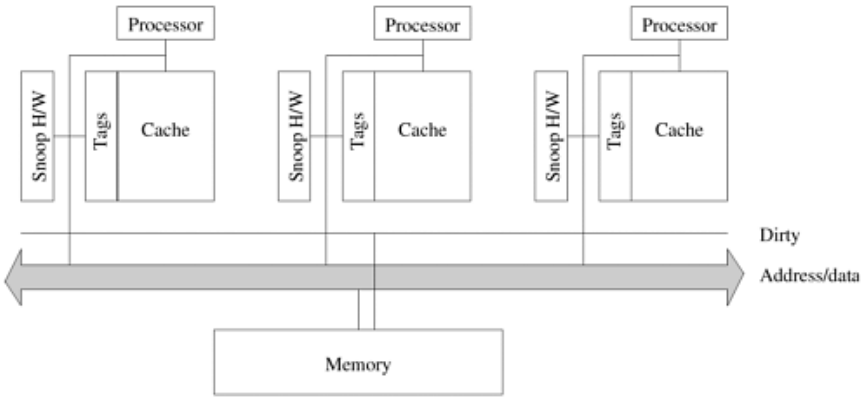
7.2 Solutions for cache consistency

7.2.1 Hardware control

The small-scale processor maintains consistency of cache by introducing a protocol called Cache-coherence Protocol on hardware. There are two types of prevalent protocols applied most frequently.

a. Snoopy Cache Protocol

Snoopy Cache Protocol applied on bus-based multiprocessor systems. The protocol shared the signal from main memory to buses of every cache monitors. There are two main types in designing the protocol: the first is Write Through cache and Write Back cache; another one is Write Invalidate, which usually seen as MSI and MESI etc., and Write Update protocol, which often used like Dragon. The Write Invalidate Protocol is the most prevalent protocol in Snoopy Cache Protocol.



Graph 18. Snoopy Cache Protocol Mechanism

All the data copies would be invalid in this monopoly access protocol, hence guarantees there is no other readable or writable copy. Besides, there would be only one processor gets the write authorization while data copies in another processor would be invalidated. The data copy must be reloaded before the processor failed to get write authorization previously could achieve the right. Therein lies the serialization character of Write Invalidation Protocol.

The limitation of Snoopy Cache Protocol lies in its applicability only on multi-processor system connected by bus, whose number is confined for the limit of efficiency of bandwidth and cache itself. Systems consist of multiple processors should apply other types of networks. Once it is impossible to broadcast the uniformed instructions to all caches through networks, the snoopy cache protocol is no long appropriate. Therefore the necessity of directory-based protocol rises.

b. Directory-based Protocol

Directory-based protocol most commonly applied on multi-processors system connected by Internets. By assigning the tracking and sharing caches of local storage in distributed cache multi-processor system, the protocol differs in three ways.

Full-map directory: In full-map directory, each node has a full-map occupies a large portion of their own storage spaces. Therein lies the implication that only small-scale multi-processor system and multi-computer system is proper to apply this very protocol.

Limited directory: In limited directory, the number of pointer for each directory column is limited regardless of capacity of the system. With reduced demand of space for cache, it is economic when the sharing of cache is limited; besides, when the multi-processor system possess locality, which means there are only a small part of processor accessing some certain data blocks in given times, the limited number of directory is sufficient to cope.

Chained directory: In chained directory, directory pointer achieves the track of shared data copying. Its major advantage is the expandability of its limited directories without regulations of the amount of shared data blocks; whereas the algorithm of maintenance is complicated and its hardware designing is even harder.

7.2.2 Software control

Preprogramming to analyze is major way of solving cache consistency problems. The easiest way to cope with the cache consistency issue is by software analysis, in which process data divided into applicable cache and inapplicable cache. All the shared data belongs to the latter and could not be stored in cache, whereas to many data, it would lose the mean of cache.

Albeit this very method is kind of conservative and runs the opposite way with the initial purpose of designing cache that it may worsen the performance of the entire system, it is the most guaranteed method especially for some important coding sections.

With development of compiling technology, combining the software and hardware's methodology plays a greater role by allowing hardware involves in parts that compiling could not conquer.

7.3 Deadlock processing mechanism

The true reason of deadlock is the dependency between requests. One simple mechanism of detecting deadlock and recovering from deadlock is to postulate that there is potential deadlock when detecting the overflow of requests of input and output cache. Then the system would no longer process the top of input stack and directly gives feedback a null response to requesting node until the potentiality got solved.

Another active recovering scheme is to change the tripartite transmission into bilateral transmission from requesting to response, that is, break the initial chain of requesting – interfering – responding into two chains of requesting – responding, and solve the deadlock by strict request-respond chains.

The third deadlock coping mechanism is by establishing limitations on scales of system and make the requests always acceptable so as to cancel the momentary status of the directory. This very mechanism ensures the request could be operated under every possible circumstance without receiving null responses.

8. Conclusion

This paper presents a general picture regarding current development and trends of contemporary caches and its problem of data consistency. In symmetric structure, coherent cache structure achieves its addressing and consistency by applying hardware interference; with reduced burden for programmers, it comes with the actual issue of a complicated hardware design. The two level cache strategy is prevalent and considerably important, a brief understanding of the notion makes it easy to identify the advantage and shortcomings of applying shared or private cache mechanism. Streaming memory structure provides more freedom for programmers with more authority of controlling the location of data storing as well as retrieving and their respect communication with processor cores. The very design eliminates the hardware expenses of core communication and data consistency while improving the complexity of program designing. Programmers have to control the communication and retrieving/storing process ostensibly. The non-uniform cache structure expresses its advantage.

Currently, all processor manufacturers are applying multiple cache technologies in their processors, some of which even need specific operation system to be implemented, or at least add more or less extra logic circuits to facilitate. Therefore, when it comes to cache development, extra consideration especially viewpoints from architecture and algorithm to make expectations is needed.

Reference

[1] Zhang M, Asanovic K. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors[C]//ACM SIGARCH Computer Architecture News. IEEE Computer Society, 2005, 33(2): 336-345.
[2] Hennessy J L, Patterson D A. Computer architecture: a quantitative approach [M]. Morgan Kaufmann, 2011.
[3] Jie Liu, Yan Ma, Wei Ye, Jiangang Gao: Analysis for Multicores Memory Hierarchy;
[4]Lisheng Wang, Chengjun Zhou, Liguo Chen. General Statement of Cache of Multi-core Processor Technology [J]. Journal of Technology Post, 2011 (21).
[5] Michael Zhang and Krste Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In 32nd International Symposium on Computer Architecture, Madison, Wisconsin, June 2005.
[6] Beckmann B M, Wood D A. Managing wire delay in large chip-multiprocessor caches[C]//Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on. IEEE, 2004: 319-330.
[7] Jaehyuk Huh J, Changkyu Kim C, Shafi H, et al. A NUCA substrate for flexible CMP cache sharing[J]. Parallel and Distributed Systems, IEEE Transactions on, 2007, 18(8): 1028-1040.
[8] Chang J, Sohi G S. Cooperative caching for chip multiprocessors[C]//Computer Architecture, 2006. ISCA'06. 33rd International Symposium on. IEEE, 2006: 264-276.
[9] Beckmann B M, Marty M R, Wood D A. ASR: Adaptive selective replication for CMP caches[C]//Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on. IEEE, 2006: 443-454.