

Balanced Memory Architecture for High I/O Intensive Information Services for Autonomous Decentralized System

Hironao Takahashi, Kinji Mori

Department of Computer Science,
Tokyo Institute of Technology
Tokyo, Japan

takahashi@mori.cs.titech.ac.jp, mori@cs.titech.ac.jp

Hafiz Farooq Ahmad

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
drfarooq@niit.edu.pk

Abstract—The increasing disparity between CPU speed and storage system require novel techniques to achieve high I/Os for computing systems with high assurance properties, such as timeliness, fault tolerance and online expansion. Proposed architecture is balanced memory architecture which is for high I/O(Input Output) intensive Autonomous Decentralized System using Data Transmission System memory concept in DF (Data Field) to achieve timeliness in information systems to solve today's High I/O intensive application market demand. The proposed architecture also implements two new memory caches in the system, namely L_3 that works at local memory level and L_4 , on top of HDD. Moreover, the proposed high I/O intensive distributed system concept called Autonomous Decentralized Data Transmission System (ADDTS) that is composed of triosubsystems with three autonomous decentralized remote storage nodes. These remote storage nodes have online expansion technique for storage system to improve system capacity. Therefore, the proposed system architecture solves previous issues and it achieves flexible data storage system that increases or decreases its capacity as per the requirements. The I/O performance evaluation shows significant enhancement over the traditional computing and ADS systems.

Keywords- *DTS (Data Transmission System) cache, Timeliness, Write back cache, Block I/O device, Locality of reference, IOPS, Bonnie Bench mark program, ADDTS (Autonomous Decentralized Data Transmission System).*

I. INTRODUCTION

Food, fuel, energy and pollution management are the global challenges for the human society currently. Rapid increase in human population on the globe demands growing economic process. Information Technology (IT) is the key-enabling factor to accelerate this economic growth. The demands of IT services such as Web services, online transactions are increasing rapidly and numerous rich interactive Web based applications are emerging. Some of the applications can be characterized as high I/O intensive such as video on demand, video services, and medical sciences including healthcare and digital imaging. A number of application domains such as data

mining need high I/O intensive system to access data from/to hard disk, as it cannot fit into the main memory.

These emerging applications require continuous operation, non-stop service system and timeliness to achieve high assurance [1,3] to meet Service Level Agreement (SLA). SLA is the explicit fulfillment of the requirement of the Quality of Service (QoS), such as reliability and timeliness. SLA requirement in the emerging applications on the Internet needs ADS (Autonomous Decentralized System) characteristics for information system [2,4]. A major challenge in the realization of QoS in the Web services is the requirement of high I/O transactions for information systems. ADS concept is high assurance computer system architecture proposed in 1977 to meet the requirement of the industry, especially control systems. Theoretical foundations of ADS rely on the principles of autonomous controllability and autonomous coordinatability. These two properties assure timeliness, online expansion, fault tolerance and online maintenance of the system. But, traditional Autonomous Decentralized System does not perform high I/O intensive services for information systems on the Internet. Moreover, information system computing facility in data centers consumes high energy, mostly for its storage systems. Therefore these systems emit huge amount of CO₂ by heat dissipation. There is need to consider how to design information system to achieve efficiency for IO and energy consumption. To these goals, this paper focuses on system characteristics for I/O requests. Generally, CPU utilization ratio on the server is 10 to 15%. Therefore, people try to use CPU more effectively by Virtual OS with multiple application software on the same computer hardware. But these services place high I/O intensive requests and transactions on the server. I/O intensive applications with large, sustained workloads of various I/O sizes have special management challenges [7]. High I/O environments are commonly thought of as a large SAN issue, however small and medium-sized storage environments can be I/O intensive, if there is a large amount of active workload and I/O processing. An environment with a relatively small number of servers and storage, for example, an OLTP, database server or vertical application for CRM or ERP can be I/O intensive. I/O intensive

applications may have large, medium or small amounts of storage, yet have active workloads with either a large number of I/O requests of various sizes. For large I/O sizes, huge amount of data is moved and enormous bandwidth is utilized, while smaller I/O request consumes less bandwidth when processed. Therefore an I/O intensive application system have to be designed to support many transactions, support a large amount of bandwidth, or both. Moore's law gives rise to extremely high capacity circuits, such as large storage and powerful processors [5]. Overall performance of CPU has been increasing 50% /year, while for storage it has been merely increasing 7% / year. The access time of a disk drive is many orders of magnitude slower than CPU cycle time. The increasing disparity between CPU speed and storage system leads toward performance degradation of the computing systems.

There have been a number of efforts to improve I/Os performance however these are substantially different from our work. In [12], the author proposes Unified Buffer Cache (UBC), the focus is to unify the file system and virtual memory caches of file data to improve I/O transactions. However it is an unmanaged one level cache that is very much different from the DTS cache though the objective is to improve I/Os. Similarly [13], [14] provide solution for high I/Os, based on solid-state disk, and is altogether different from DTS layered cache. The concept is to use non-mechanical devices to achieve high I/Os. Moreover, this solution is highly expensive compared to the DTS technologies.

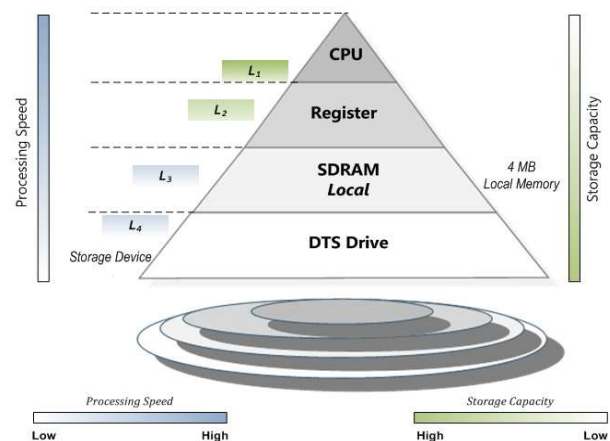
In this paper firstly a novel layered cache architecture consisting of 4 levels cache hierarchy has been proposed inline with memory hierarchy. Secondly ADS based ADDTS system has been proposed to achieve online expansion in the storage system on the Internet. The proposed system enhances I/O performance to achieve timeliness, and optimize the energy usage in storage systems by increasing CPU utilization and reducing frequent HDD access respectively. The rest of the paper is organized as follows. Section 2 describes the DTS multi level cache concept and subsystem architecture for I/O intensive information systems. Section 3 gives the detail of ADDTS system design based on ADS concept. Section 4 describes system evaluation using benchmarking tools. Section 5 describes DTS and ADS integration benefits, and Section 6 is conclusions and the future work.

II. PROPOSED BALANCED MEMORY ARCHITECTURE

A. The Concept of Architecture

Storage systems need to provide high I/O transactions in addition to their capacity and availability requirements [7, 9]. Generally large and medium size storage systems consist of disk arrays and associated caches to improve the I/O performance. Multi level cache hierarchy requirement for computing systems has been recognized as early as 1989[5]. Caches at various layers of memory hierarchy in computing systems are more fast but very small compared to the adjacent lower level of storage device. This paper proposes a novel concept of layered cache based system (Figure 1). This is Data Transmission System (DTS) concept based on hierarchical layered memory architecture for information systems. The

memory layers consist of hierarchical caches starting from local memory to hard disk (Figure 1 a). Proposed concept consists of 4 levels cache (L_1, L_2, L_3, L_4) to store important data for read and write on various levels in close proximity to the CPU. The key requirement for the size of the cache at lower level L_{j-1} is that it should be smaller than the size of the cache at higher level L_j to maintain all the data of lower level cache layer at higher cache layer. This requirement fulfills the inclusion property dictating that the contents of lower level cache are subset of higher-level cache unlike the non-inclusion where this property does not hold[15]. This property gives rise to balanced layered memory architecture as shown in Figure 1(a).



(a) Balanced computing system memory hierarchy

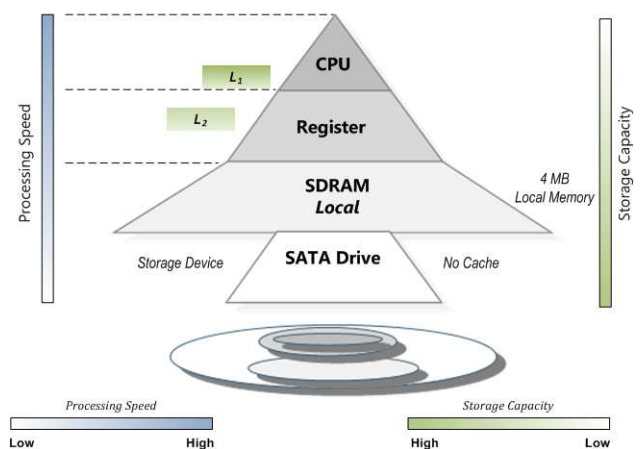


Figure 1. (b) Unbalanced computing system memory hierarchy

Figure 1(a) is DTS based abstract system architecture for high I/O intensive information system to achieve timeliness. The local memory connected to CPU via DMA is converted into block I/O cache device, namely DTS cache by dividing it into two areas [16]. This DTS cache concept is very unique and it brings excellent potential in I/O intensive application services. The L_4 cache is placed in the specially designed hybrid HDD using DTS memory concept. DTS L_4 cache based hybrid HDD has 1GByte SDRAM as cache with HDD as

target drive. DTS L₄ cache based hybrid HDD is managed by memory layer control device called DTS chip inside the drive. The cache algorithm is 100% write back policy. To achieve reliability in this context, the drive uses an uninterruptible power supply inside HDD. The DTS intelligent function loads the most frequently used block I/Os and maintain these on the cache. One example is quick boot, where a system boots up using this HDD and it pre-reads OS boot up sectors and application programs at the previous boot up process. Therefore DTS L₄ cache based hybrid HDD improves boot up time for any Operating System (OS) such as Windows XP, Windows2003, Linux and UNIX. It also shows significant performance in computer memory consuming environments such as virtual OS software as VMware and XEN. Eventually the I/O performance will be 10 to 100 times faster than traditional HDD depending on the size of data.

The balanced memory architecture facilitates efficient resources utilization while unbalanced memory hierarchy creates performance bottleneck (Figure 1b). The proposed system concept provides high I/Os as explained in the Section 2.2.

B. Subsystem Architecture

The proposed subsystem architecture is realized by creating two new caches L₃ and L₄ as L₁ and L₂ are already part of the computing system design. Therefore, we focus on how to design and implement L₃ and L₄ caches. In our proposal L₃ cache is realized by converting local memory into block I/O device [16]. An appropriate cache policy is applied to manage data on this newly created cache. Write back cache policy is the most appropriate choice for transferring data on DTS cache and subsequently storing it to the HDD.

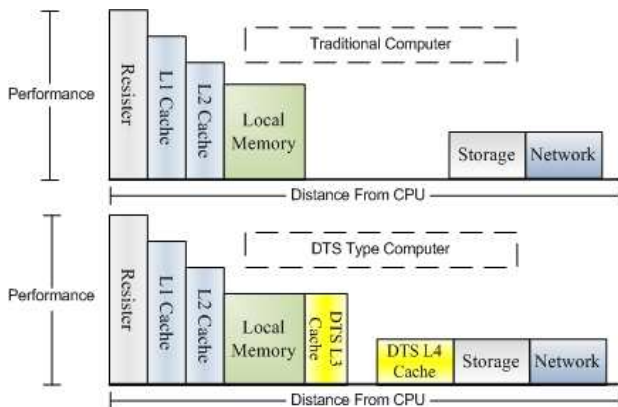


Figure 2. DTS Computing System vs Traditional Computing System architecture

Two-way I/O transactions i.e. write and read are performed on local memory in DTS type computing system. Therefore, DTS is the technology for data transmission management, which has intelligent caching function. DTS cache technology is independent of the physical device characteristics and therefore it can use any block device as DTS cache. However, it represents slow speed device at low-level layer (Figure. 2).

For instance, if user needs more high I/O performance, then select the most powerful block I/O device as cache. The local memory comprising of SDRAM is the most appropriate choice for this purpose. DTS technology manages to keep the most frequently used I/O block data in the DTS cache. It proposes efficient search technique to quickly search the DTS cache. This further enhances the search time within the DTS cache, called random index search. It helps to find the required block quickly compared to the conventional search in the simple cache. A technique has been proposed in [16] to reach quickly the desired location of the DTS cache called jumping algorithm. It also enhances network transfer speed through DTS cache utilization.

Figure 3 proposes DTS based Subsystem Architecture. The proposed architecture has DTS based memory in each subsystem node and caches the block data from HDD device. The user program on the subsystem uses 4GB in case of 32bit OS. These user programs are located in memory or HDD depending on the situation of memory in the subsystem. Therefore, there is no guarantee that the cache stores all the data from HDD. Proposed memory architecture maintains I/O data for each user if data is required frequently.

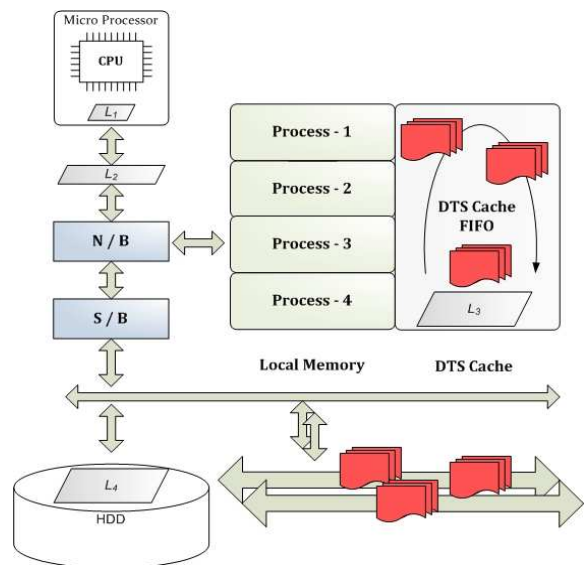


Figure 3. Proposed DTS based subsystem Architecture

In this section though the focus is on the DTS cache in a single subsystem, but in the next section, we present some details how this cache can be used to create distributed DTS cache system using ADS principles.

Figure 4 shows DTS based cache memory configuration for high I/O intensive system architecture, where subsystems communicate through logical DF inspired by ADS [8]. In the context of ADS, each subsystem broadcasts data with Content Code (CC) to every other subsystem, and this memory area i.e. DTS cache at level L₃, and level L₄, is used as common I/O area. This DTS cache is reserved as a common area for all I/Os in the DF in ADS. The target drive is internal HDD or remote storage connected via high-speed network. The

protocol between subsystems to remote storage is fast data transmission protocol such as iSCSI.

This system architecture brings autonomous storage expansion as per the system requirements for the application. One of the subsystems broadcast write event result on DF, the second subsystem receives it and executes this event with CC. Each subsystem has write-back policy and it does not need to wait the write operation to secondary storage. Consequently, each subsystem carries out the execution of the write operation in the local memory, such as SDRAM as cache level L₃, and cache L₄ on top of DTS based hybrid memory HDD. The read/write transactions response time is enhanced due to both of these caches acting as virtual hard disk and also due to write back policy of the cache. If there is no DTS cache on subsystem, renewal of data and reading data cannot be maintained on local memory area because it is unified memory space defined by OS policy [10, 11].

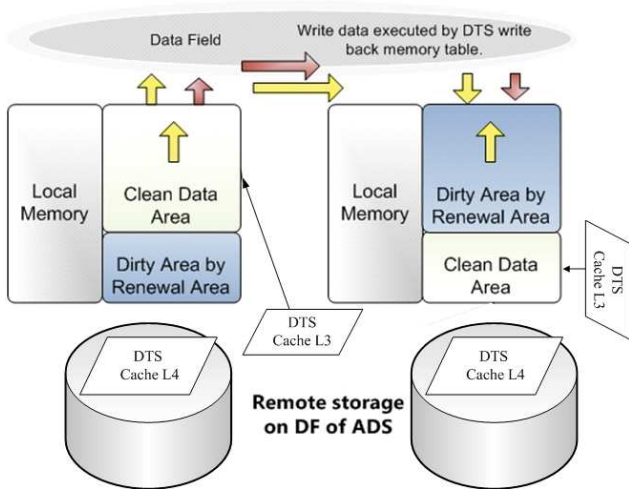


Figure 4. DTS basis cache memory configuration for high I/O intensive system architecture using Data Field of ADS.

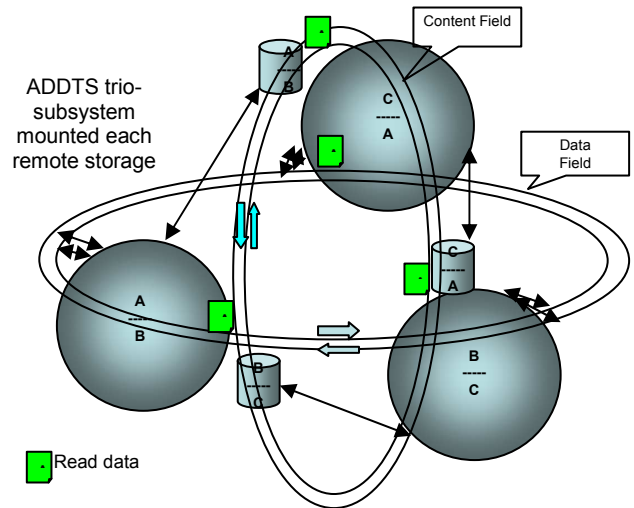
III. AUTONOMOUS DECENTRALIZED DATA TRANSMISSION SYSTEM (ADDTS): CONCEPT AND DESIGN

ADDTS (Autonomous Decentralized Data Transmission System) system architecture is proposed based on ADS data field principles on DTS balanced memory architecture presented in this paper in section 2. ADDTS concept is realized from three subsystems like trio- group node with two field networks.

One field is traditional Data Field (DF) to distribute control commands to request services in the system. The second one is Content Field (CD), which is a storage data network based on high-speed transmission protocol such as iSCSI protocol. ADDTS enhances I/O performance by utilizing multi level DTS cache in the subsystem. There is no need to access remote storage because DTS cache maintains write requests in DTS cache by write back cache policy. Read requests from outside systems, such as the Internet are also performed by DTS cache policy using subsystem cache. Finally, these subsystems on DF in ADS perform High I/O performance for

I/O intensive application services through L₃, and L₄, caches at very high speed.

ADDTS system architecture assigns three subsystems as trio-node group mounted on three remote storage nodes, having dual storage area. The first subsystem mounts storage partition A and B, the second node has storage partitions B and C. The third node mounts storage partitions C and A. Two of them maintain same data to achieve fault tolerance. Therefore, if one of the three nodes fails, the other subsystem maintains same date without any switching time to assure high IOs in the system.



(a) ADDTS system configuration using two separate fields (Read process).

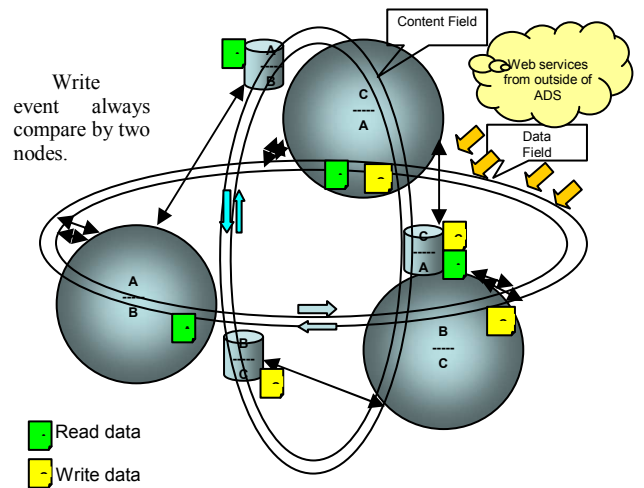


Figure 5. (b): ADDTS system configuration using two separate fields (Write process).

ADDTS trio-subsystem concept is depicted in Figure 5 (a) and (b), where the three nodes are correlated to each other and mounted through iSCSI protocol on the Internet. When one of the nodes fails, another node reproduces same Internet Protocols to connect remote storage by iSCSI. If remote storage fails, other owner of storage volume in the trio of node

reproduces it through the data transmitted via CF network. Therefore, there is no overhead in DF transactions traffic and system reproduces new storage volume by storage volume.

Each subsystem has high I/O performance feature based on DTS balanced memory architecture. ADDTS also has remote storage expansion by DTS cache based policy. To enhance the write performance including small packets from other subsystem on the network, ADDTS executes high I/O response by write back memory policy. These subsystems broadcast data to other subsystems with CC, so huge amount of data is generated in DF whenever I/O intensive applications execute. DTS cache reserves one common area for I/Os data in DF. Virtual memory with OS is assigned 4GB area for each user processes. But it is unified memory architecture therefore it cannot maintain I/O data of DF. DTS cache area is common I/O cache area for outside I/Os from the network. Therefore, each process may access separately DTS cache. It also gives highly improved response time to maintain I/O intensive applications from multiple processes (OS based virtual memory does not provide this feature). Therefore, ADS concept application to DTS technology enhances timeliness in the computing system. ADDTS read/write process can be described as follows:

- 1) When the service request is broadcast on DF, some of the subsystems execute this event by CC data.
- 2) Minimum number of subsystems required is three and each subsystem executes the service request separately.
- 3) When the read operation is required, each subsystem executes this operation separately. The first subsystem executes read operation then broadcasts the result of read to other subsystems. Thus, read event is always executed by the first receiving subsystem on the DF, and reply to the requester. Moreover, if all requests are read, ADDTS cache size will be two times bigger by block cache meta data on each subsystem from request event.
- 4) Once the write operation executes, the result of these requests are compared by each node through broadcast data with CC in DF. If compared results are same, then one of the nodes replies the acknowledgement for the write event to requester. But if results are not matched, then the second subsystem broadcast the message that the write operation failure (negative acknowledgement). Requester should retry the write event again till both subsystems write result is matched.
- 5) Once compared results are matched, the written data result assigned as dirty block data and will execute to reflect to remote storage as renewal data via CF as shown in Figure 6.

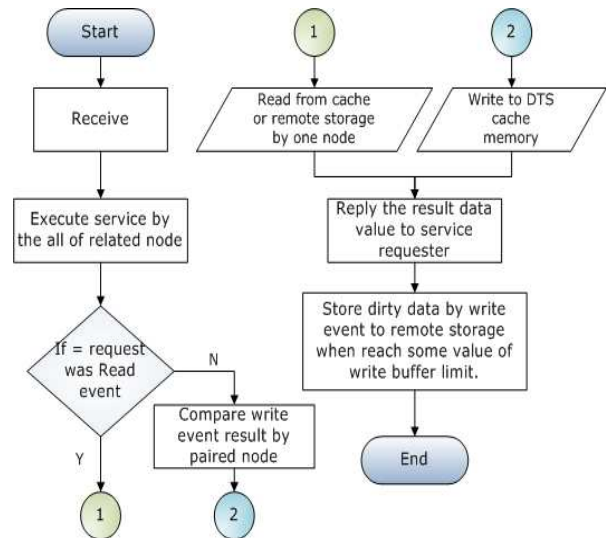


Figure 6. ADDTS event process flow chart

IV. PERFORMANCE EVALUATION

A. Theoretical Evaluation

There are two different scenarios to be considered for the evaluation.

Write event: DTS write back policy replies acknowledgement rapidly. Therefore I/O response time is equal to SDRAM (DT|S cache).

$$S(DTS\ write) = S(DRAM) \text{ EXP}r(t-t_0)$$

If write data capacity is less than DTS cache memory area then:

$$S(DTS\ write) \approx S(DRAM)$$

Read event: There are two situations of read events and the performance should be considered in two different scenarios as follows:

The first read access always performed on slow device located at the bottom of storage hierarchy such as HDD. The response time in this case

$$S(DTS\ read) \approx S(HDD)$$

Once read event starts frequently then read event may hit DTS cache area and read performance will be enhanced. Therefore read performance will be improved in DTS cache as follows.

$$S(DTS\ read) = M / (1 + (M/S_0 - 1) \text{ EXP}^{-r}(t-t_0))$$

This is logistic formula because the behavior varies i.e. increase and decrease over the time. Under these circumstances, ADDTS cache concept always meets the demand for data capacity in DTS cache memory area to maintain I/Os performance. If ADDTS actually effects, the read data is always less than cache memory and therefore:

$$S(DTS\ read) \approx S(DRAM)$$

This is the most important phenomenon to enhance I/Os performance under ADDTS system architecture.

It should be pointed out that when the data size which is running on ADDTS is smaller than DTS cache, there exists the possibility of cache miss hit by the first read operation.

Therefore, the first read cache penalty time is likely to be slower than HDD and I/Os performance shows unstable behavior before the data is on the DTS cache. The first-read cache penalty shows unpredictable behavior on ADDTS. I/Os performance behavior with respect to the cache capacity in ADDTS is based on logistic formula and it functions on the principle of feedback to the node. Therefore, when very large size of data runs on DTS cache, the behavior of the cache is unstable. The write event is acknowledged from DTS cache, and it maintains high performance until cache reaches its threshold value. But read event consumes more time initially as the data has to be read from slow storage device such as HDD. However, when read data is stored on DTS cache, then read speed increases rapidly. If L_4 cache is at the bottom device on top of HDD, the performance of I/O should increase through L_3 , and L_4 caches.

Generally I/O performance can be modeled as follows:

$$X_{t+1} = aX_t (1 - X_t) : (3.57 \leq a \leq 4, 0 \leq X_0 \leq 1)$$

This type of behavior also appears whenever size of the data exceeds DTS cache size. Therefore, ADDTS tries to maintain actual data within DTS cache size by block I/O analysis technique. In this paper, we carry out theoretical evaluation of I/Os performance after the first read penalty. There are multiple factors that need to be considered. The first is appropriate probability distribution, representing data access pattern. The second is locality principle, with respect to spatial locality and temporal locality. The third one is overhead for the processes involved.

The probability distribution can be considered as standard distribution as follows:

$$F(x) = (1/\sqrt{2\pi\sigma^2}) \int \exp(-(x-m)^2/2\sigma^2) dt$$

Consider that the value of σ is higher than 2 due to high queue I/O model demands from Web services. Therefore the ratio of I/O intensiveness is fairly very high. When the behavior of the DTS cache reaches stable phase of read operation in ADDTS, the response time evaluation of ADDTS will be equal to SDRAM based DTS cache VS HDD access time.

The worst scenario of DTS cache is when σ value becomes "1". It is assumed that the probability of existing data in one hour on DTS cache is 20% /24hours = 0.83% from total storage capacity. Let us consider the capacity of total mounted storage is 250GB. In this scenario, necessary DTS cache size is 250GB * 0.83% = 2.07GB. Therefore, 2.07GB data pass on each subsystem within one hour. I/O process time consists of CPU process time plus storage I/O access speed. Let us consider

Total execution/process time = T (T)

$$T (total) = T (cache device speed) + T (target mounted storage)$$

If local SDRAM memory is cache device, and target storage is HDD, total time of each storage access is as follows:

$$T (SDRAM) = 45 \mu s = 0.045ms$$

$$T (HDD) = 9ms (seek time + overhead)$$

The time of the system is

$$\begin{aligned} T (system) &= (1/2.07) \times 0.045 + (1 - (1/2.07)) \times 9 \\ &= 48.3 \times 0.045 + (1 - 0.48) \times 9 \\ &= 0.022 + 4.68 \\ &= 4.692 (ms) \end{aligned}$$

The worst scenario is

$$9/4.692 = 1.92 \text{ times faster than HDD.}$$

Once ADDTS architecture is in stable phase, the I/O performance should reach as that of SDRAM performance.

$$\text{Ratio of Speed} = T (HDD) / T (SDRAM) = 200$$

Therefore, the total ratio of expected speed gain with DTS cache over HDD is 200 times faster access. Amount of enhanced speed is in best-case scenario; in fact actual speed will be less 200 times due to other overheads.

B. Performance Evaluation using I/O meter benchmark program

This paper investigates I/O performance of DTS multi level cache architecture and standard computing system architecture in Linux operating system. The results show significant improvement in the response time for the proposed system architecture. For instance, 512 100% Write0% Read Random result is 211.51 IOPS (Input-Output per Second) but ADDTS based is 33354.32 IOPS under eight multiple accesses situation. Also 1K 80%Write20%Read Random result is 237.14 IOPS but ADDTS based is 32020.75 IOPS under eight multiple accesses situation (Figure 7).

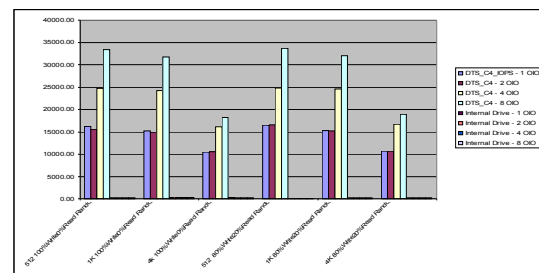


Figure 7. ADDTS type vs. Traditional computing system IOPS test results.

The setup for the measurement is as follows:

(Intel Xeon 2.4 GHz, 4GB System memory, 4GB Working Table, Target File Size = 2GB

Write Back Policy, Windows2000 Advance SP4 Server Target Disk size 4GB) Test tool is I/O meter 2004.

C. ADDTS node I/O performance vs traditional ADS node

Using section 4.2 test results, we evaluate node I/O overhead on ADS and ADDTS system architecture.

The number of nodes is $N(node)$, and communication value is

$$CV = \sum k=n(n-1)/2$$

$$\begin{aligned} \text{I/O overhead of ADS subsystem in DF (4KB/ 64KB)} \\ &= 1/DF(4KB/ 64KB) \text{ speed (MB/s)} \times N(node) \times (N-1) / 2 \\ &= TN(total) (ms) \end{aligned}$$

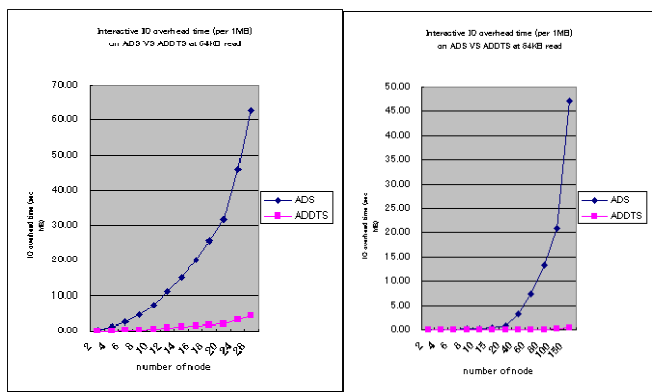


Figure 8. I/O intensive application response time evaluation under ADS VS ADDTS.

Both of the evaluation results (Fig 8) show how I/O intensive applications spend the time under ADS broadcasting on the DF. For example, 64KB random read event total 1MB size event on ADS under 24-nodes, the result is 63msec, but ADDTS shows 4.2msec. Let us see the comparison of interactive I/Os. The result 20% read, 80% write 1KB random access with 150-nodes is 47.12 seconds under ADS but ADDTS executes this operation in 0.35 second. The ratio between the two is 134.6, which means that ADDTS is 134.6 times faster than ADS model. Therefore, ADDTS has significant potential to enhance this service model. Moreover, the evaluation results show that the I/O process overhead is extremely large for I/O intensive services in ADS. The general cache memory architecture is always best effort model. But ADDTS concept always maintains data on DTS cache memory with automatic block I/O analyzing architecture including the write processes. Only the first read penalty is the best effort behavior when DTS cache reads data from HDD, the results shows unpredictable behavior as discussed in Section 4.1. The fundamental point of I/O performance under ADDTS design is that all I/Os should execute on DTS cache. If one of the node caches is full, the total performance will drop down rapidly. Therefore, ADDTS cache capacity is controlled by autonomous expansion/reduction of node(s) policy to maintain required number of nodes in the system.

V. ADVANTAGES OF PROPOSING MEMORY CONCEPT IN ADDTS

This paper investigates DTS balanced memory architecture, and its integration with ADS concept to achieve high assurance properties in computing systems, especially for I/Os. DTS cache concept proposes two new caches in the computing systems namely L_3 , that is realized by dividing local memory into two portions and implements one portion as DTS cache L_3 . The second cache, called L_4 , is realized by implementing a cache on top of HDD. This multi level DTS cache is reserved for I/Os data for DF in ADS. This DTS cache concept is very unique and it brings excellent potential in I/O intensive applications. DTS cache dedicated area for just I/Os data for

DF enhances I/Os performance for ADS. It also gives high response time to maintain I/O intensive applications from multiple processes and nodes. Proposed balanced memory architecture creates more effective Autonomous Decentralized System configuration than traditional ADS [2]. Eventually, ADDTS reduces the total energy consumption and CO2 by avoiding access to HDD as the requests are fulfilled from the DTS caches.

REFERENCES

- [1] I.-L. Yen, R.Paul and K. Mori. Towards integrated methods for high-assurance systems. *IEEE Computer*, 31(4):32-34, April 1998.
- [2] K. Mori. Autonomous decentralized systems: concept, data field architecture and future trends. In *Proc. of ISADS, IEEE*, pages 28-34, 1993
- [3] H.F. Ahmad and Mori. *Autonomous Information Service System: Basic Concept for Evaluation*, IEICE Transactions on Fundamentals of Electronics and Computer Sciences, Vol. E83-A, No.11 pp.2228-2235, November 2000
- [4] Leguizamo, C.P. Kato, S. Kirai, K. Mori, K. Autonomous Decentralized Database System for Assurance in Heterogeneous e-Business. *Proceeding of COMPSAC*, p589-p595, IEEE, May 2000
- [5] S. Przybylski, M. Horowitz, J. Hennessy, Characteristics of performance-optimal multi-level cache hierarchies, *Proc of the 16th Annual International Symposium on Computer Architecture* pp. 114 - 121 (1989).
- [6] Moore's law www.intel.com/technology/mooreslaw/index.htm
- [7] Elizabeth Varki, Arif Merchant, Jianzhang Xu, and Xiaozhou Qiu. Issues and Challenges in the Performance Analysis of Real Disk Arrays. *IEEE Transactions on Parallel and Distributed Systems*, 15(6):559-574, 2004.
- [8] ADS protocol specification R3.0, MSTC/JOP 1101-1999/09/3
- [9] Dai Kobayashi, Akitsugu Watanabe, Toshihiro Uehara, Haruo Yokota, A high-availability software update method for distributed storage systems: *Research Articles, Systems and Computers in Japan*, Vol 37, Issue 10, Pp 35-46 (2006)
- [10] Daniel P. Bovet, Marco Cesati : *Understanding the Linux Kernel* , O'reilly Press, pp.422-498 (2001).
- [11] Maurice J. Bach : *The Design of The UNIX Operating system*, Bell Labs Press, pp.264-287 (1986)
- [12] Chuck Silvers, UBC : *An efficient Unified I/O and Memory Caching Subsystem for Netbsd*, *Proceedings of FREENIX Track: 2000 USENIX Annual Technical Conference*, San Diego, California, USA, June 18-23, 2000
- [13] White Paper, *Using Real-Time I/O Signature Analysis to Identify Performance Improvement Options for Database Applications*, http://www.soliddata.com/pdf/WP_IOSignatures_v2.pdf, Solid Data Systems Inc, July 2006.
- [14] Wade Tuma, *Comparisons of Drive Technologies for High-Transactions Databases*, http://www.soliddata.com/pdf/WP_Drive_Comparison_v2.pdf, Solid Data Systems, Inc. (August 2007)
- [15] Mohamed Zahran, Kursad Albayraktaroglu and Manoj Franklin, Non-Inclusion Property in Multi-level Caches Revisited, *IJCA*, Vol. 14, No. 2, pp.1-10, (2007)
- [16] Hironao Takahashi, Hafiz Farooq Ahmad, Kinji Mori, *Layered Memory Architecture for High IO Intensive Information Services to Achieve Timeliness*, 11th IEEE High Assurance Systems Engineering Symposium Nanjing, China, December 3 - 5, 2008