Software TLB Management Method Based on Balanced Binary Tree

Chen Hongyu
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 15771900781@189.cn

Zhao Li
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: zhaoli1998@163.com

Zhang Yuke
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 17792012345@189.cn

Ai Jian
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: aijianup@163.com

***Abstract*—Purpose: with the development of the computer system, there is a higher request to reduce the failure times of Translation Look—aside Buffer and relieve the failure influence, normal solution is deal with TLB failure by software or hardware, it find the page table and then implement the index operation to locate the page want. Method: In order to satisfy the needs of software management method for mapping speed from virtual address to physical address and to expand the size of TLB, our team design a software management method based on balanced binary search tree. Page management is implemented in software management, TLB is managed by operating system based on abstract model, and MMU (memory management unit) is no longer used Unit, our team build a balanced binary tree to search TLB. Result: Binary search tree has the advantage of pruning in the interval search problem under the balanced state. Therefore, searching TLB by this method is conducive to the expansion of TLB capacity, making space for cache and other performance improvement designs on chip and reducing costs. However, the search speed is reduced and some time is sacrificed to free up CPU space. Conclusion: This method can free up CPU design space, also can expand more size of TLB and reduce the cost, its optimization algorithm is worthy of further research.**

Keywords-Component; TLB; Balance Binary Tree; Management of Software

I. INTRODUCTION

With the continuous development of the computer, the speed of CPU is faster and faster, but the speed of memory has not been improved. The development of TLB will help computer to process large virtual address. In this paper, our team design a TLB software management method of constructing a balanced binary search tree based on virtual page number. When the TLB capacity of balanced binary search tree is large, the method could deal with it quickly, for example, the number of virtual page numbers in balanced state is doubled, and the average retrieval times only need to be increased once.

Therefore, the retrieval speed is fast, the capacity increases, and the hit rate will also increase. Moreover, the time cost of tree building is evenly distributed in each search, and the average search time decreases. The balanced binary search tree is used to cut the interval search the advantage of branch and search is to quickly search the virtual page number and locate it to the corresponding location of the memory. Because the page number needs to be searched many times, it helps to deal with the large virtual space.

II. TLB

Most programs always visit a few pages many times, so TLB records frequent pages and their information, which can accelerate the mapping from virtual address to physical address.

The basic unit of TLB internal storage is the page table item, which corresponds to the page table item stored in RAM, the more TLB capacity, the more page table items can be stored, and increase the probability of TLB hit rate. Due to the limited capacity of TLB, RAM page table and TLB page table items cannot be one-to-one correspondence.

A. Location

TLB is used to cache some tab table entries. TLB can be between CPU and CPU cache, or between CPU cache and main memory, depending on whether cache uses physical addressing or virtual addressing. If the cache is a virtual addressing, the addressing request is sent directly from the CPU to the cache, and then the required TLB entries are accessed from the cache. If the cache uses physical addressing, the CPU will first perform for each memory operation and send the obtained physical address to the cache. Each method has its own advantages and disadvantages.

B. Common optimization

A common optimization of cache with physical addressing is parallel TLB search and cache access. The lower bits of all virtual addresses (e.g., the lower 12 bits in the virtual address when there is a 4KB tab in the virtual memory system) represent the address offset (in page address) of the requested address within the paging, and these bits will not change during the transition from the virtual address to the physical address. The process of accessing the CPU cache consists of two steps: use an index to find the corresponding entries in the CPU cache data store, and then compare the corresponding tags of the CPU cache entries found. If the cache is indexed by the same page address during the translation of addresses, the translation of higher bits of virtual and real address (i.e. page to page address / page number of pages) on TLB and the "index" operation of CPU cache can be performed in parallel. The page number of the physical address obtained from the TLB is then sent to the CPU cache. The CPU cache compares page number tags to determine whether the access is missing or missing. It is also possible to perform TLB search and CPU cache access in parallel, even if the CPU cache must use some bits that may change after address translation. Refer to the address translation section of cache entry for further details on cache and TLB under virtual addressing.

III. ALGORITHM OF BALANCED BINARY SEARCH TREE

Balanced binary tree has the following properties: it is an empty tree or the absolute value of the height difference between its left and right sub trees is no more than 1, and the left and right sub trees are both balanced binary tree.

A. Insert operation

Inserting a new node into the balanced binary tree destroys the balance of the balanced binary tree. First of all, our team need to find the pointer of the root node of the minimum sub tree which is out of balance after inserting a new node. Then adjust the link relationship between the nodes in the sub tree to make it a new balanced sub tree. When the unbalanced minimum sub tree is adjusted to a balanced sub tree, all the other unbalanced sub trees do not need to be adjusted.

LL type adjustment: Insert a node on the left sub tree of point B. After insertion, the balance factor of the left sub tree of point B becomes 1 and that of node a becomes 2. In this way, it can see that the sub tree with node a as the root node is the minimum unbalanced sub tree. When adjusting, the left child B of a is rotated to the right instead of a as the root node of the original unbalanced sub tree, and the lower right rotation of the node of a is called the root node of the right sub tree of B, and the original right sub tree of B becomes the left sub tree of A.

In the binary search tree insertion and deletion operations, the advantage of using balanced tree is to make the tree structure better, so as to improve the speed of search operation. The disadvantage is that the insertion and deletion operations become more complicated, which reduces their operation speed. The operation of the imbalance caused by deleting a node in a binary search tree is more complex than that of inserting a node, so it will not be discussed here.

B. AVL Deletion

Like the insert operation, deleting a node may break the balance, which requires us to adjust the balance when deleting.

The deletion is divided into the following situations:

- First, search the whole binary tree for the node to be deleted. If it is not found, it will be returned without processing. Otherwise, the following operations will be performed.The node to be deleted is the current root node t.If the left and right sub trees are not empty. The deletion operation is implemented in the higher sub tree.
- The height of the left sub tree is greater than that of the right sub tree. Assign the largest

element in the left sub tree to the current root node, and then delete the node with the largest element value in the left sub tree.

- If the height of the left sub tree is less than that of the right sub tree, assign the smallest element in the right sub tree to the current root node, and then delete the node with the smallest element value in the right sub tree.
- If one of the left and right sub trees is empty, replace the current root node with the non empty sub tree or null.
- If the element value of the node to be deleted is less than the T value of the current root node, delete it in the left sub tree.
- Recursively, delete in the left sub tree.

This is to determine whether the current root node still meets the equilibrium condition.

If the equilibrium condition is satisfied, only the height information of the current root node T needs to be updated. Otherwise, rotation adjustment is required.

If the height of the left sub tree of the left child node of T is greater than the height of the right sub tree of the left child node of T, the corresponding single rotation is performed. Otherwise, double rotation is performed.

The element value of the node to be deleted is greater than the T value of the current root node. Delete it in the right sub tree.

C. Summary:

- Non leaf nodes have at most two child nodes.
- The value of non leaf node is larger than that of left child node and less than that of right child node.
- The difference in the number of levels on the left and right sides of the tree is no more than 1.
- There are no nodes with identical values.

IV. DESIGN SCHEME

The application of page number can be approximately regarded as a combination of mass or partially ordered intervals over time. When the CPU offer page number request, the operating system first searches the balanced binary search tree about page number in TLB, and compares the root node with the request page number. If the page number of the root node is greater than the request page number, the operating system first searches the balanced binary search tree of the TLB, Recursively search in the left sub tree. If the page number of the root node is less than the request page number, then it is recursively searched in the right sub tree. If the page number of the root node is exactly equal to the request page number, the query is considered successful. The corresponding physical block will read out, and the adder is used to splice the address in the page to convert it into the corresponding physical address.

A. Recursion exit

Therefore, there are two exits for the end of recursion.

- One is that the access is successful and the corresponding physical address is obtained;
- the other is that the left or right sub tree of the root node is empty. If the access fails, there is no corresponding page number in the TLB. The operating system will retrieve the page table in memory, call the corresponding page table entry into TLB, and insert it into the balanced binary search of the TLB according to the page table entry in the tree.

When the balanced binary tree constructed by TLB is full, it is necessary to replace the oldest page table entries. For deleting a page table item, it is necessary to set different flag bits for replacement operation which accord to the corresponding replacement algorithm. Deletion will also cause imbalance, it is necessary to balance the binary search tree first after deletion.

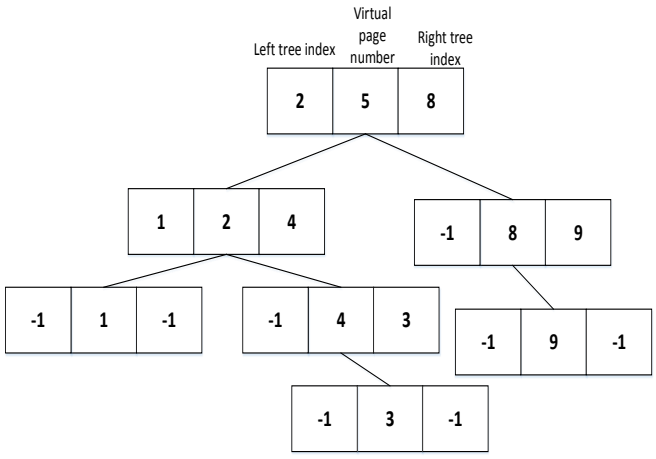

Figure 1. Model

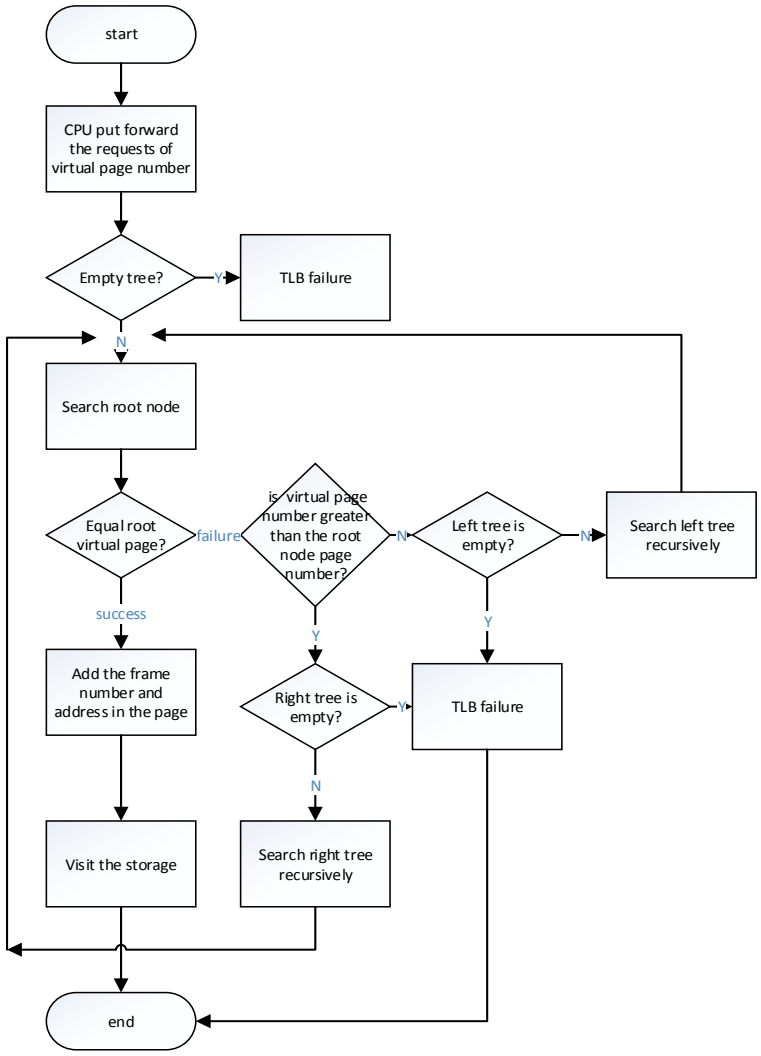

Figure 2. Search step

V. TLB STORAGE

A typical page table entry includes page frame number, protection bit, modification bit (dirty bit), access bit, cache forbidden bit and on / off bit.

TLB has virtual page number, significant bit, modification bit, protection bit, page frame number, root node index bit of left and right sub tree, balance factor and so on.

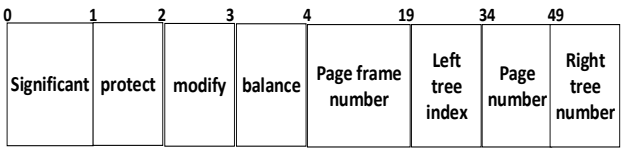

Figure 3. Item

A. Composition

The virtual page number marks the page corresponding to the table item, and the significant bit records whether the table entry is used or not, whether the table item has been modified, the read / write and execution permissions of the protection bit record, and the setting of index bit and balance factor bit of the root node of the left and right sub trees are determined by comparing the size of the root node with that of the root node to determine whether the search of the left and right sub trees is recursive .

The node stores the index of the left and right child nodes, and then stores the balance factor (the balance factor is only 1,0, -In three cases, if the absolute value exceeds 1, then the balanced binary search tree is judged to be unbalanced).

When the TLB is full, some page table items need to be eliminated. Therefore, according to the corresponding page replacement algorithm, different flag bits are flexibly set to mark the page table items that should be eliminated; the row significant bit is that when the system starts, each TLB row is empty, and the information in it is invalid In order to show whether the information in the TLB row is valid, each row has a significant bit. By clearing the significant bit of the row, the corresponding page table item is eliminated.

Note: setting the global bit in a page directory/table entry will prevent that entry from being flushed. This is useful for pinning interrupt handlers in place.

B. Alternate method

An alternate method is to use instruction, which should be used instead of the above method when doing small mapping modifications (creation, removing, changing.) instruction is mostly used in page not mapping and remapping routines in order to invalidate a previous cached translation. If instruction or some other TLB flush method had not been used, the mapping would remain cached, producing undefined consequences.

However, please note that the instruction instruction was introduced in the i486 ISA and is not part of the i386 ISA, thereby requiring a properly written i386-compatible kernel to use conditional inclusion of relevant code at compilation time depending on the target machine. The above is more complicated in the multiprocessor case. If another processor could also be affected by a page table write (because of shared memory, or multiple threads from the same process), it must also flush the TLB on those processors. This will require some form of inter-processor communication.

VI. MATHEMATICAL VERIFICATION

Time complexity: the difference between height of the left sub tree and the right sub-tree is no more than 1. Our team assumed that NH is the minimum number of nodes in the balanced binary tree with depth H.

From the method of recursion recursion, the result is NH = NH-1 + NH-2 + 1 (1 is the root node), and the minimum search length is deduced.

Therefore, the equation can be obtained by the characteristic equation method.

$$\lambda^2 = \lambda + 1$$

The result is $\lambda = \frac{1 \pm \sqrt{5}}{2}$

Then our team assign the special solution is n = C, replace it into the former formula, and the result is C = - 1, so the general term formula can be obtained.

$$C_1 \lambda_1^n + C2 \lambda_2^n - 1$$

Therefore, the equation can be obtained.

$$(1 + \frac{2}{\sqrt{5}}) (\frac{1+\sqrt{5}}{2})^n + (1 - \frac{2}{\sqrt{5}})(\frac{1-\sqrt{5}}{2})^n - 1$$

And then figure the equation

$$n < \log \frac{1+\sqrt{5}}{2} (N+1) < \frac{3}{2} \log_2 (N+1)$$

According to the characteristics of the tree, the maximum depth of the balanced binary tree with n nodes is log2N, and the average search length is log2N approximately.

VII. MODERN OPERATING SYSTEM

In modern processor, software uses virtual address to access memory, and MMU unit of processor is responsible for converting virtual address to physical address. In order to complete this mapping process, software and hardware jointly maintain a multilevel mapping page table. When the processor finds that the page table cannot be mapped to the corresponding physical address, it will trigger a page missing exception and suspend the error process. The operating system software needs to handle the page missing exception.

A. Content

TLB is specifically used to cache page table entries in memory, usually inside the MMU unit. TLB is a very small cache, and the number of TLB entries is relatively small. Each TLB entry contains information about a page, such as significant bit, virtual page number, modified bit, physical page frame number, etc. When the processor wants to access a virtual address, it will first query in the TLB.

- If there is no corresponding entry in the TLB table entry, it is called TLB miss, then need to access the page table to calculate the corresponding physical address. If there is a corresponding entry in the TLB table entry, the physical address is directly obtained from the TLB table entry, which is called TLB hit.

- The basic unit of TLB internal storage is TLB table entries. The larger the TLB capacity, the more TLB entries can be stored, and the higher the TLB hit rate. However, the capacity of TLB is limited. At present, the Linux kernel uses 4KB small pages by default. If a program uses 512 small pages, it needs at least 512 TLB entries to ensure that TLB miss will not occur. However, if a 2MB page is used, only one TLB table entry is needed to ensure that TLB miss will not occur. For large applications that consume memory in gigabytes, large pages in 1GB can also be used to reduce TLB miss.
- Because accessing the page table in memory is relatively time-consuming, especially when multilevel page tables are widely used nowadays, multiple memory accesses are required. In order to speed up the access, the system designer has designed a hardware cache TLB for page table.

The CPU will first look it up in TLB, because it is very fast to find it in TLB. The reason why TLB is fast is that it contains a small number of entries. On the other hand, TLB is integrated into the CPU. It can run almost at the speed of the CPU.

If an entry (TLB hit) containing the virtual address is found in the TLB, the corresponding physical address can be obtained directly from the entry.

Otherwise, unfortunately, TLB miss will occur, and the page table of the current process (paging structure caches may be used here). At this time, another part of the MMU, the table walk unit, is called out. The table in the MMU is page table.

The method of using table walk unit hardware unit to find page table is called hardware TLB miss handling, which is usually adopted by CISC architecture processor (such as IA-32).

It can't be found in page table, and it will be handled by software (operating system) when page fault appears.

In contrast, software TLB miss handling, which is usually adopted by RISC architecture processors (such as alpha), does not involve the CPU after TLB miss. The operating system searches the page table through software. The way to use hardware is faster, while the way to use software is more flexible. IA-64 provides a hybrid mode, which can take into account the advantages of both.

If the P (present) bit of the entry corresponding to the virtual address is found in the page table, it indicates that the physical page corresponding to the virtual address currently resides in memory, that is, page table hit. There are still two things to do.

Since it can't find it in TLB, it's natural to update TLB.

Check the permissions, including read / write / executable permissions, user / supervisor mode permissions, etc. If it do not have the correct permissions, SIGSEGV (segmentation fault) will be triggered.

If the P bit of entry corresponding to the virtual address is 0, page fault will be triggered. There may be several situations.

B. Address

The virtual address has never been accessed after it has been allocated (for example, if there is no space allocated after allocation, the physical memory will not be allocated). After the page fault is triggered, the physical memory is allocated, that is, demand paging. After a certain demand is available, the system will allocate the P position to 1.

The content of the corresponding physical page is swapped out to the external disk / flash. At this time, the page table entry stores the temporary location of the page in the external swap area. It can switch it back to physical memory, establish the mapping again, and then set P position 1.

If the virtual address does not exist in the page table of the process, it means that the virtual address is not in the address space of the process. At this time, segmentation fault will also be triggered. The CPU's MMU locates the page directory for the process using the special register mentioned above.

The page directory index (from the first 10 bits of the virtual address) is used to locate the PDE that identifies the page table needed to map the virtual address to a physical one. The page table index (from the second 10 bits of the virtual address) is used to locate the PTE that maps the physical location of the virtual memory page referenced by the address.

The PTE is used to locate the physical page. If the virtual page is mapped to a page that is already in physical memory, the PTE will contain the page frame number (PFN) of the page in physical memory that contains the data in question. (Processors reference memory locations by PFN).

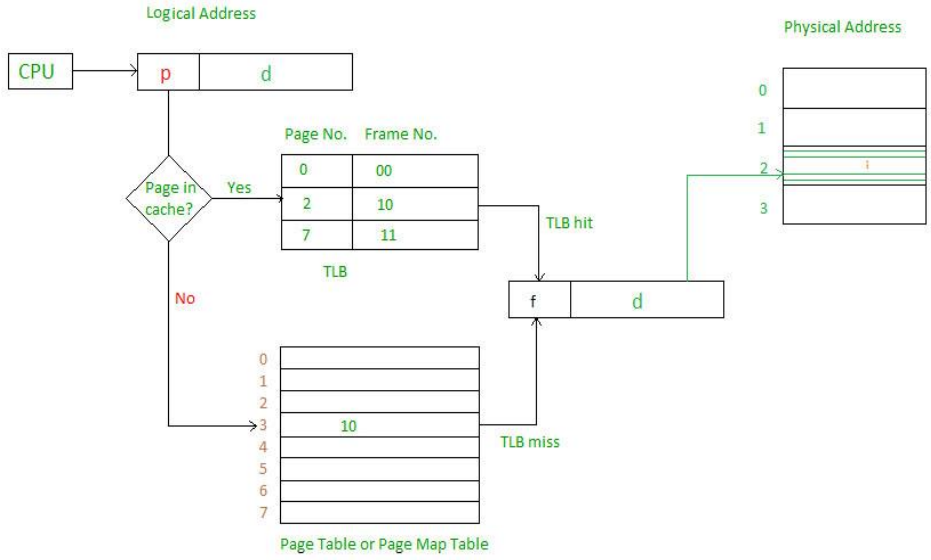

Figure 4. process

C. *Steps in TLB hit:*

- CPU generates virtual address.
- It is checked in TLB (present).
- Corresponding frame number is retrieved, which
- now tells where in the main memory page lies.

D. *Steps in Page miss:*

- CPU generates virtual address.
- It is checked in TLB (not present).
- Now the page number is matched to page table residing in main memory (assuming page table contains all PTE).
- Corresponding frame number is retrieved, which now tells where in the main memory page lies.
- The TLB is updated with new PTE (if space is not there, one of the replacement technique comes into picture i.e either FIFO, LRU or MFU etc).

E. *Conclusion*

Effective memory access time(EMAT) : TLB is used to reduce effective memory access time as it is a high speed associative cache.

EMAT = h*(c+m) + (1-h)*(c+2m)

Where, h = hit ratio of TLB

m = Memory access time

c = TLB access time

If the page is not in physical memory, the MMU raises a page fault, and the Windows page fault–handling code attempts to locate the page in the system paging file.

If the page can be located, it is loaded into physical memory, and the PTE is updated to reflect its location.

If it cannot be located and the translation is a user mode translation, an access violation occurs because the virtual address references an invalid physical address. If the page cannot be located and the translation is occurring in kernel mode, a bug check(also called a blue screen) occurs.

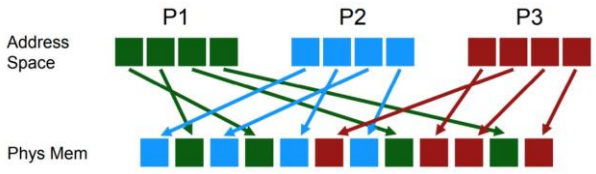

Figure 5. mapping relation

VIII. CONCLUSION

According to the characteristics of the tree, the maximum depth of balanced binary tree with n nodes is log2N, and the average search length is approximately log2N.

In this paper, our team discuss the software TLB management method based on balanced binary search tree through the abstract model. This management method takes advantage of the advantages of balanced binary search tree in pruning and searching, which can quickly complete the retrieval of virtual page number and physical page .

When the TLB expands, the number of TLB failures decreases, and the cost of tree building is shared equally in each search. To a certain extent, it can improve the performance of computer operating system in processing large virtual memory space, accelerate the conversion from virtual address to physical address, and make room for other designs on chip.

However, due to the lack of experimental environment and sufficient theoretical knowledge, our team only verify theoretical time compleity by math and statistic, did not test in the actual environment, especially not verify the program model ,which is based on balanced binary search tree, our team did not test reliability under the condition of hardware instability or storage space disorder, like no electrical power and other special circumstances.

A reliable software model which can be used in engineering must consider all kinds of special circumstances. Stability and reliability is the most important aspect of a software model. Balanced binary search tree as the search scheme of this model has the possibility of further optimization.

The algorithm designed in the future research should not only consider the running speed, but also consider the difficulty of implementation in practical engineering and the future maintenance work. It is necessary to do further study of complexity of maintenance work and the stability of the software model.

Engineering and theory are the unity of opposites. Theory is the source of engineering. The development of theory needs the support of engineering.

REFERENCES

[1] Marin G, Mellorcrummey J. Cross-architecture performance predictions for scientific applications using parameterized models[C]. measurement and modeling of computer systems, 2004, 32(1): 2-13.

[2] Boncz P, Manegold S, Kersten M L, et al. Database Architecture Optimized for the New Bottleneck: Memory Access[C]. very large data bases, 1999: 54-65.

[3] G.M.Adelson-Velsky,E.M.Landis, “An algorithm for the organization of information” 1962 30(1): 7-13

[4] Deb, Kalyanmoy, and Ram Bhushan Agrawal. "Simulated Binary Crossover for Continuous Search Space.." Complex Systems 1995.

[5] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[6] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[7] Talluri M, Hill M D. Surpassing the TLB performance of superpages with less operating system support[C]. architectural support for programming languages and operating systems, 1994, 29(11): 171-182.

[8] Sleator, Daniel D., and Robert E. Tarjan. "Self-adjusting binary search trees." Journal of the ACM 32.3 1985

[9] Rajwar R, Herlihy M, Lai K K, et al. Virtualizing Transactional Memory[C]. international symposium on computer architecture, 2005, 33(2): 494-505.

[10] Menon A, Santos J R, Turner Y, et al. Diagnosing performance overheads in the xen virtual machine environment[C]. virtual execution environments, 2005: 13-23.