FACULTATEA DE ELECTRONICĂ,TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI,

 UPB 2013

Cache Manager

Student:

Ungureanu – Ilie Alexandru – Florin

Grupa 431A

Cuprins:

 1.Principalele intrebuintari ale Cache Manager-ului

 2.Managementul memoriei virtuale a cache-ului

 3.Dimensiunea cache-ului

 4.Stucturi de date ale cache-ului

 5. Interfatarea cu fisierele sistemului

 6. Citirea si scrierea rapida a fisierelor cache: Fast I/O

 7. Modul de implementare a fisierelor de date citite si scrise

 8. Exemplu Windows Server 2003

 9. Concluzii

 10. Bibliografie

**1.Principalele intrebuintari ale Cache Manager-ului**

 **Caracteristici cheie ale Cache Managerului**

 Cache managerul are cateva caracteristici cheie:

* Suporta toate tipurile de fisiere de sistem (locale sau de network), astfel inlaturand necesitatea ca fiecare fisier de sistem sa isi implementeze propriul cod pentru managementul cache-ului.
* Foloseste managerul memoriei pentru a controla care parti din ce fisiere se afla in memoria fizica.
* Punele datele sub forma cache intr-o baza block virtuala, in contrast cu multe systeme care executa cache, care il efectueaza intr-o baza block logica. Astfel se permite efectuarea unui read-ahead inteligent si o viteza de acces mare catre cache fara a implica si driverele fisierelor de sistem.
* Suporta “aluzii” primite de catre aplicatii in momentul deschiderii fisierului.
* Suporta fisiere de sistem recuperabile pentru a recupera date dupa o eroare de sistem.

 **Un singur sistem cache centralizat**

 Unele sisteme de operare se bazeaza pe fiecare fisier de sistem individual pentru a face cache la date, un lucru din care rezulta ori in cache-uri duplicate si codul managementului memoriei in sistemul de operare sau in limitari in privinta tipurilor de date care pot fi cache-uite. In contrast, Windows ofera o facilitate centralizate de efecuare de cache-uri pentru toate datele stocate in exterior, fie ele pe hard disck-ul local, dschete, CD-ROM-uri etc.

**Managementul memoriei**

 Un aspect neobisnuit al cache manager-ului este acela ca el nu stie cata data cache este in memoria fizica, acesta datorandu-se faptului ca el acceseaza date prin maparea vizibilitatii fisierului in spatiul adreselor virtuale al sistemului. Pe masura ce adresele in aceste mapari sunt accesate, managerul memoriei demapeaza aceste pagini din cache si, daca datele s-au schimbat, pune in pagina datele inapoi in fisiere.

 Prin efectuarea de cache pe baza unui spatiu de adrese virtuale prin maparea fisierelor, cache managerul evita generarea de Request packets I/O scrise sau citite pentru accesarea datelor pentru fisierele asupra carora efectueaza cache. In schimb el copiaza data catre sau de la adresele virtuale de unde portiunile fisierelor cache-uite sunt mapate si se bazeaza pe managerul memoriei.

**Coerenta memoriei cache**

 O functie importanta a cache managerului este de a asigura ca oricarui proces care aceeseaza datele cache sa i se ofere cea mai recenta versiune a datelor respective. O problema poate aparea aunci cand un proces deschide un fisier in timp ce alt proces mapeaza fisierul direct in spatiul adresei. Aceasta potentiala problema nu apare in Windows pentru ca atat managerul cache cat si aplicatia care face maparea fisierelor in spatiul adresei foloseste acelasi servicii de management al memoriei pentru maparea fisierelor .

 De exemplu, daca Process1 are un view (View1) pe fisierul mapat in spatiul adreselor userului si Process2 acceseaza acelasi view via cache-ul sistemului, Process2 va vedea in timp real orice schimbare pe care Process1 o face. Orice proces care acceseaza un fisier sub Windows vede intotdeauna cea mai recenta versiune a acelui fisier, chiar daca anumite procese au fisierul deschis prin sisteme I/O si altele au fisierul mapat in spatiul adreselor lor prin Functia de mapare a fisierului Windows.

**Blocuri virtuale ale memoriei cache**

 Cache managerul Windows foloseste o metoda cunoscuta ca si “virtual block caching”, in care managerul cache tine evidenta care fisiere sunt in cache. Cache Manager-ul este capabil sa monitorizeze aceste porțiuni de fișiere prin maparea a 256 KB de fișiere în sistemul de spații de adrese virtuale , folosind rutine cache speciale sistemului situate în managerul de memorie. Aceasta abordare are urmatoarele beneficii cheie:

* Ofera posibilitatea de a efectua procese de “citire inainte” (read-ahead) inteligente, deoarece cache-ul urmareste care parti din ce fisiere sunt in cache, poate prezice unde in apelant s-ar putea duce in continuare.
* Permite sistemelor I/O de a trece pe bypass sistemul de fișiere pentru solicitările de date care sunt deja în cache.

**Stream-Based Caching**

Cache managerul este construit si pentru a executa ‘stream caching’, in opozitie cu file caching. Un stream este o secventa de bytes intr-un fisier. Unele fisiere de system, cum ar fi NTFS, permite unui fisier sa contina mai mult de un singur stream ; cache managerul acomodeaza astfel de fisiere de sistem prin cache-uirea fiecarui stream in mod independent.

**Nota:** Intern, cache managerul nu este constient de numele streamurilor de fisiere dar foloseste pointeri catre aceste obiecte.

**Fisiere de sistem suport recuperabile**

 Fisierele de system recuperabile cum ar fi NTFS sunt construite pentru a reconstrui structura de volum a discului dupa o eroare de sistem. Aceasta capabilitate inseamna ca operatiunile I/O in progress la momentul respectiv trebuie sa fie ori complet terminate ori complet indepartate de pe disc cand sistemul etse restartat. Operatiunile I/O pe jumatate terminate pot corupe volumul discului sau chiar a face un volim intreg inaccesibil. Pentru a evita aceasta problema, un fisier de sistem recuperabil pastreaza un fisier log in care inregistreaza fiecare update pe care intentioneaza sa il faca structurii fisierului de sistem.

 Pentru a garanta recuperarea cu succes a volumului, fiecare inregistrare de fisier log care documenteaza un update de volum trebuie sa fie scris complet pe disc inainte ca auto-update-ul sa fie aplicat volumului. In mare, urmatoarele actiuni au loc:

1. Fisierul sistem scrie un fisier log care inregistreaza documentatia pe care metadata intentioneaza sa o faca.
2. Fisierul de sistem apeleaza cache managerul pentru a trage fisierul log pe disc.
3. Fisierul de sistem scrie update-ul volumului in cache.
4. Cache managerul trage metadatele alterate pe disc, updatand structura volumului.

 Cand un fisier de sistem scrie data in cache, poate sa furnizeze o secventa logica de numere (LSN) care identifica inregistrarea in fisierul log, care corespunde update-ului cache-ului. Cache managerul patreaza evidente aceptor numere, inregistrand cel mai mic si cel mai mare LSN asociat fiecarei pagini din cache.

 **2.Managementul memoriei virtuale a cache-ului**

 Datorita faptului ca cache managerului sistemului Windows cache-uieste date in mod constant, foloseste regiuni din spatiul virtual al adreselor (in locul memoriei fizice) si le manageriaza in structuri numite blocuri de control al adreselor virtuale , sau VACBs. VACBs defineste aceste regiuni ale spatiului adreselor in sloturi de 256 KB numite views. Cand cache managerul initializeaza in timpul procesului de bootare, aloca un array intial de VACBs pentru a descrie memoria cache.

 La prima operatie I/O a fisierului, cache managerul mapeaza un view de 256KB dintr-o regiune aliniata de 256KB a fisierului ce contine datele cerute intr-un slot liber in spatiul adreselor cache ale sistemului. De exemplu, daca 10 bytes incep cu un offset de 300.000 bytes ar fi cititi intr-un fisier, view-ul care ar fi mapat ar incepe la un offset de 262.144 si s-ar extinde la 256KB.

 Cache managerul garanteaza ca un view este mapat atat timp cat este activ(desi view-urile pot ramane mapate dupa ce devin inactive). Un view este marcat activ numai in timpul unei operatii de citire sau scriere in sau dintr-un fisier. Numai daca un proces deschide un fisier prin specificarea flag-ului FILE\_FLAG\_RANDOM\_ACCES in call pentru a Crea un Fisier (CreateFile), cache managerul demapeaza viewurile inactive ale unui fisier cat timp mapeaza noi viewuri pentru fisier daca detecteaza ca fisierul ar fi accesat in mod secvential. Paginile pentru viewul nemapat sunt trimise in liste standby sau modificate si datorita faptului ca managerul de memorie exporta o interfata speciala pentru managerul cache, acesta poate directiona paginila pentru a fi plasate la inceputul sau sfarsitul acestor liste. Paginile care corespund viewurilor fisierelor deschise cu flagul FILE\_FLAG\_SEQUENTIAL\_SCAN sunt mutate in fata listelor, pe cand toate celelalte sunt mutate la sfarsit. Aceasta schema incurajeaza reutilizarea paginilor care apartin fisierelor cu citire secventiala si impiedica in mod specific o operatiune mare de copiere de fisiere sa afecteze mai mult de o mica parte din memoria fízica.

 Daca managerul cache are nevoie sa mapeze un view al unui fisier si un mai exista sloturi libere in cache, va demapa cel mai recent view inactiv mapat si va folosi acel slot. Daca niciun vew nu e disponibil, o eroare I/O este returnata, indicand ca nu sunt suficiente resurse de sistem disponibile pentru a executa aceasta operatie. Dat fiind faptul ca viewurile sunt marcate ca fiind active numai in timpul operatiilor de scriere sau citire, exista putine sanse ca acest scenariu sa se intample deoarece mii de fisiere ar trebui sa fie accesate simultan pentru ca aceasta situatie sa aibe loc.

 **3.Dimensiunea cache-ului**

**Dimensiunea virtuala a cache-ului**

 Pe un sistem Windows de 32 de biti marimea virtuala a sistemului cache este limitata doar la cantitatea de spatiu virtual al adreselor pentru modul kernel si la registrul cheie SystemCacheLimit care poate fi configurat optional. Asta inseamna ca marimea cacheului este plafonata de spatiul adreselor sistemului de 2GB, dar este in mod tipic semnificant mai mic. Marimea maxima a cacheului virtual este de 1.024GB pe un Windows de 64 de biti.

**Dimensiunea setului de lucru a cache-ului**

 Cum am mentionat anterior, una din diferentele cheie in designul managerului cache in Windows fata de alte sisteme de operare este delegarea de management a memoriei fizice catre managerul memoriei globale. Din aceasta cauza, codul existent care se ocupa de extinderea setului de lucru si de trimming, precum si de controlul listelor de standby si modificate, este folosit si pentru controlul marimii sistemului cache.

**Dimensiunea fizica a cache-ului**

Setul de lucru al sistemului include cantitatea de memorie física care este mapata in viewuri in spatiul adreselor virtuale a cacheului, un reflecta neaparat cantitatea totala de fisiere de date care sunt cachuita in memoria fízica. Poate exisa o discrepanta intre cele doua valori din cauza fisierelor de date aditionale care pot exista in listele de pagini modificate sau in standby-ul managerului de memorie.

 Un exemplu ilustreaza cum managerul cache poate cauza mult mai multe fisiere de date decat cele continute in setul sistemul functional care trbeuiesc cachuite in memoria fizica. Sa consideram un sistem care se comporta ca un server de fisiere dedicat. O aplicatie client acceseaza fisiere data din intreaga reta., pe cand un server, cum ar fi driverul de fisiere server foloseste interfete ale cache managerului pentru a scrie sau citi date in numele clientului. Daca clientul citeste prin cateva mii de fisiere de 1MB fiecare, managerul cache va trebui sa inceapa sa refoloseasca viewuri cand ramane fara spatiu de mapare. Pentru fiecare fisier citit ulterior, cache managerul demapeaza viewuri si le remapeaza pentru fisiere noi. Cand cache managerul demapeaza un view, managerul memoriei nu renunta la fisierele de date din setul in functiune al cacheului care corespunde viewului , ci muta datele in lista de standby. In absenta oricarei alte cereri de memorie fizica, lista de standby poate sa consume aproape toata memoria fizica care ramane in afara setului in functiune al sistemului. Cu alte cuvinte, in mod virtuale toata memoria fizica a serverului va fi folosita pentru a cachui datele fisierului, cum este prezentat in figura urmatoare:

 Deoarece cantitatea totala de fisiere de date cachuite include si setul functional al sistemului, liste de pagini modificate si liste de standby este in fond managerul cache real. Subsistemul managerului cache ofera interfete convenabile pentru a accesa fisierele data prin manegeruld e memorie.

 Pentru a incerca sa reflectam in mod precis cantitatea totala de fisiere de date cachuite intr-un sistem, Task Managerul arata o valoare numita Cache in viwul de performanta care reflecta marimile combinate ale setului in functiune a sistemului, lista de standby si lista de pagini modificate. Valoarea cachului in Task Manager mai include si valorile Paged WS, Kernel WS si Driver WS care figureaza in Process Explorer. Cand aceste valori au fost alese, vasta majoritate a System WS a provenit din Cache WS. Acest lucru nu mai este de actualitate astazi, dar anacronismul ramane in Task Manager.

**4.Stucturi de date ale cache-ului**

 Managerul cache foloseste urmatoarele structuri de date pentru a tine evidenta fisierelor cachuite :

- Fiecare slot de 256 KB din sistemul cache este descris de un VACB

- Fiecare fisier cache deschis separat are o harta cache privata, care contine informatia folosita pentru a controla citirea in avans.

- Fiecare fisier cachuit are o singura structura de harta cache impartita, care indica catre sloturile din sistemul cache care contine viewuri mapate ale fisierului.

**Strucuri de date cache valabile in tot sistemul**

Cum a fost descris in pasajele anterioare, cache managerul tine evidenta starii viewurilor in sistemul cache cu ajutorul unui array de structuri de date numit virtual adress control block (VACB) array care este stocat intr-o plaja de valori neindexate. Pe un sistem de 32 de biti, fiecare VACB are dimensiunea de 32 de bytes, si un array VACB este de 128 KB, rezultand in 4096 de VACB per array. Pe un sistem de 64 de biti, un VACB este de 64 bytes, rezultand in 2048 VACB per array. Cache managerul aloca array-ului VACB initial in timpul initializarii sistemului si il leaga la lista systemwide de arrayuri VACB numita CcVacbArrays. Fiecare VACB reprezinta un view de 256 KB in sistemul cache, cum este rpezentat in figura 11-5. In figura 11-6 este rpezentata structura unui VACB.

In mod adicional, fiecare array VACB este compus din doua tipuri de VACB: VACB cu prioritate de mapare mica si VACB cu prioritate de mapare mare. Sistemul aloca initial 64 de VACBuri initiale cu prioritate mare pentru fiecare array VACB. VACBurile cu prioritate mare au in mod distinctiv viewuri prealocate din spatiul adreselor sistemului. Cand managerul memoriei un are viewuri pentru a le oferi cache managerului in timpul maparii unor date si daca cererea de mapare este marcata ca avand prioritate mare, cache managerul va folosi unul din viewurile prealocate prezentate ca fiind VACBuri cu prioritate mare.

 In timpul unei operatii I/O intr.-un fisier, contorul de referinta VACB al fisierului este incrementat si apoi decrementat cand operatia I/O s-a terminat. Cand contorul de referinta este diferit de zero VACBul este activ. Pentru acces la fisierul metadata al sistemului, contorul activ reprezinta cate drivere de fisiere de sistem au paginile in acel view inchis in memorie.

**Structura cache-ului de date per-fisiere**

 Fiecare fisier cachuit are o structura de harta cache share-uita care descrie starea fiecarui fisier cache, incluzand dimensiunea si lungimea valida a datelor. Harta cache shareuita indica deasemenea catre section ojbect, lista de harti cache privata asociata cu fisierul respectiv si orice VACB care descrie curentele viewuri mapate ale fisierului in sistemul cache. Aceste relatii per structuri de fisier de date cache sunt ilustrate in figura 11-7.

 Cache managerul trebuie sa afle daca un view al fisierului la adresa dorita este mapat in sistemul cache. Daca niciun VACB un contine offsetul dorit, data ceruta un e mapata in lod curent in cacheul sistemului.

 Pentru a tine evidente caror viewuri pentru un fisier dat sunt mapate in sistemul cache, managerul cache pastreaza un array de pointeri catre VACB, cunoscut ca VACB index array. Prima intrare in VACB index array face referinta catre primi 256KB ai fisierului, a doua intrare la cei de-ai doilea 256KB si asa mai departe. Diagrama din figura 11-8 arata patru diferite sectiuni apartinand a trei fisere diferite care sunt mapate in prezent in sistemul cache.

 Cand un proces acceseaza un fisier in mod particular intr.-o locatie specificata, cache managerul se uita in intrarea adecvata in VACB index array-ul fisierului pentru a vedea daca datele cerute au fost mapate in cache. Daca arrayul de intrare este diferit de zero, sectiunea fisierului la care s-a facut referinta este in cache. VACBul in schimb arata catre locatia din sistemul cache unde viewul fisierului este mapat. Daca intrarea este zero, cache managerul trebuie sa gaseasca un spot disponibil in sistemul cache pentru a mapa viewul cerut.

 Urmatoarea figura (Figure 11-9) este necesara pentru a manipula fisiere rare care pot avea dimensiuni de fisiere extrem de Mari cu doar o mica fractiune de date valide datorata faptului ca destul din array este alocat pentru a se descurca cu viewurile mapate curent ale unui fisier. De exemplu un fisier rar de 32GB pentru care doar 256KB sunt mapati in spatiul virtual al adreselor cachea ar cere un array VACB cu trei indecsi de array alocati deoarece o singura ramura al arrayului are o mapare si un fisier de 32GB cere un array de trei nivele. Daca cache managerul un a folosit index arrayul VACB multinivel optimizat pentru acest fisier, va trebui sa aloce un index array VACB cu 128000 de intrari, sau echivalentul a 1000 VACB index arrayuri.

1. **Interfatarea cu fisierele sistemului**

 Prima data cand datele unui fisier sunt accesate pentru o operatiune de scriere sau citire, driverul fisierului sistemului este responsabil cu determinarea daca o parte din fisier este mapat in sistemul cache. Daca un este, driverul fisierului sistemului trebuie sa apeleze functia CcInitializeCacheMap pentru a setastructurid e date per fisier.

 Odata ce un fisier este setat pentru acces cache, driverul fisierului sistemului apeleaza una din mai multe functii pentru a aceesa datele din fisier. Sunt trei metode primare pentru a accesa date cache, fiecare pentru o anume situatie:

- Metoda de copiere copiaza datele userului intre bufferi cache in spatiul sistemului si un proces buffer in spatiul userului.

- Metoda de mapare si fixare se foloseste de adresele virtuale pentru a citi si scrie date direct din bufferi cached

- Accesarea memoriei fízice foloseste adresele fizice pentru a scrie si citi date direct din bufferi cached.

 Driverele fisierelor de sistem trebuie sa asigure doua versiuni ale operatiei de citire fisier –cache si noncache- pentru a preveni o bucla infinita cand managerul memoriei proceseaza o pagina cu probleme. Cand managerul memoriei rezolva problema unei pagini prin apelarea fisierului sistemului pentru a readuce date din fisier, trebuie sa specifice aceasta operatie de citire noncache prin setarea flagului “no cache” in IRP.

 Figura 11-10 ilustreaza interactiunile tipice intre cache manager, managerul memoriei si driverele fisierului de sistem ca raspuns la operatiile I/O de citire sau scriere ale userului. Cache managerul este invocat de catre fisierul de sistem prin interfete de copiere. De exemplu, pentru a procesa o citire CcFastCopyRead sau CcCopyRead cache managerul creaza un view in cache pentru a mapa o portiune din fisierul citit si citaste datele in bufferul userului prin copierea din view. Operatiunea de copiere genereaza greseli de pagina in timp ce acceseaza fiecare pagina invalida precedente din view, si ca raspun managerul memoriei initiaza operatiuni I/O noncached in driverul fisierelor de sistem pentru a readuce datele corespunzatoare partii de fisier mapat al paginii cu probleme.

 Urmatoarele trei sectiuni explica aceste mecanisme de accesare cache, scopul lor si cum sunt folosite.

**Copierea in si din memoria cache**

Deoarece cacheul sistemului este in spatiul sistemului, este mapat in spatial adreselor la fiecare process. Aplicatiile de scriere si citire ale userului in fisiere cached trebuie sa fie servite de catre rutinele kernel-mode care copiaza date intre bufferi cacheului in spatiul sistemului si aplicatiile bufferuluic are isi au locatia in spatiul adreselor de proces.

**Caching cu interfetele de mapare si fixare**

 Asa cum aplicatiile user citesc si scriu date in fisiere pe disc, driverele fisierelor system trebuie sa citeasca si sa scrie date care descriu insusi fisierele. Datorita faptului ca driverele fisierelor sistem ruleaza in mod kernel, ele ar putea totusi, daca managerul cache a fost informat corespunzator, sa ,odifice date direct in sistemul cache. Pentru a permite aceasta optimizare, managerul cache furnizeaza functii ce permit driverului de fisiere a sistemului sa gaseasca unde in memoria virtuala fisierul de sistem metadata isi are resedinta, astfel permitand sa aibe loc modificari directe fara a utiliza bufferi intermediari.

**Caching cu interfetele de acces direct la date**

 In plus fata de maparea si pastrarea interfetelor folosite pentru a accesa metadata direct in cache, cache managerul furnizeaza o a treia itnerfata catre date cache: direct memory acces (DMA). Functiile DMAului sunt folosite pentru a citi din sau scrie in pagini cacheuite fara interventia bufferilor, ca in cazul in care o retea de fisiere de sistem face un transfer prin retea.

 Interfata DMA returneaza fisierului sistemului adresa fizica a datelor cache ale userului, care poate fi utilizat apoi pentru a transfera date direct din memoria fizica intr-un device de retea.

1. **Citirea si scrierea rapida a fisierelor cache: Fast I/O**

 Oricand este posibil, citirile si scrierile in fisiere cached sunt tratate de un mecanism high-speed numit fastI/O. Fast I/O este un mijloc de scriere sau citire a unui fisier fara a mai trbui generat un IRP. Cu ajutorul Fast I/O, managerul I/O apeleaza driverul fisierului de sistem rutina I/O rapida pentru a vedea daca I/O poate fi satisfacuta direct din cache manager fara a mai fi generat un IRP.

 Fast I/O nu se petrece intotdeauna. De exemplu, prima citire sau scriere intr-un fisier seteaza fisierul pentru cache. De asemenea, daca apelantul specifica o citire sau scriere asincrona, fast I/O nu este folosit deoarece apelantul poate fi intarziat in timpul paginarii operatiilor I/O necesare satisfacerii bufferului de copiere din sau in sistemul cache astfel nefurnizand operatia asincone de I/O ceruta. Dar chiar si cu un I/O asincron driverul fisierului de sistem poate decide ca nu poate sa proceseze operatiunea I/O prin utilizarea mecanismului de fast I/O, sa zicem de exemplu ca fisierul la care facem referire are o raza de bytes blocata. Deoarece cache managerul nu stie care parti din ce fisiere sunt blocate, driverul fisierului de sistem trebuie sa verifice validitatea citiri si scrieri, lucru care solicita generarea unui IRP. Procesul de decizie pentru fast I/O este prezentat in figura 11-11.

 Acesti pasi sunt necesari pentru a furniza o citire sau scriere cu fast I/O:

1. Un thread desfasoare o operatie de scriere sau citire.
2. Daca un fisier este cache si I/O este sincron, cererea trece mai departe catre punctul de intrare al fast I/O al stivei driverului de fisiere sistem. Daca fisierul nu este cache, driverul fisierului de sistem seteaza fisierul pentru a i se efectua operatia de cache pentru ca data viitoare fast I/O sa poata fi executat pentru a indeplini cererea de citire sau scriere.
3. Daca rutine de fast I/O a driverului fisierului de sistem determina ca fast I/O este posibil, face apel la rutina de citire sau scriere a managerului cache pentru a accesa datele fisierului direct in cache.
4. Cache managerul translateaza offsetul fisierului furnizat intr-o adresa virtuala in cache.
5. Pentru citiri, managerul cache copiaza data din cache in bufferul procesului care a adresat cererea; pentru scrieri copiaza datele din buffer in cache.
6. Una din urmatoarele actiuni are loc:

-Pentru citiri unde FILE\_FLAG\_RANDOM\_ACCES nu a fost specificat cand fisierul a fost deschis, informatia read-ahead harta cache privata a apelantului este updatada.

-Pentru scrieri, bitul eronat al oricarei pagini modificate in cache este setat astfel incat scrierea lenesa va stii sa o arunce discului.

-Pentru write-through fisiere, orice ,odificare este aruncata discului.

1. **Modul de implementare a fisierelor de date citite si scrise**

 In aceasta sectiune se va analiza cum implementeaza cache managerul scrierile si citirile de fisiere de date un numele driverelor de fisiere de sistem . A se retine faptul ca cache managerul este implicat in fisierele I/O doar cand un fisier este deschis fara flagul FILE\_FLAG\_NO\_BUFFERING si apoi citit sau scris cu functiile I/O ala Windowsului. Fisierele mapate un trec prin cache manager si nici fisierele deschise cu flagul FILE\_FLAG\_NO\_BUFFERING setat.

**Inteligent Read-Ahead**

 Cache managerul foloseste principiul de localizare spatiala pentru a executa read-ahead inteligent prin prezicerea ce date sunt predispuse a fi citite de catre procesal de apelare bazandu-se pe datele care sunt curent citite. Deoarece sistemul cache este bazat pe adrese virtuale un conteaza daca sunt juxtapuse in memoria física. Fisierele read-ahead pentru prinderi in blocuri logice sunt mai complexe si necesita cooperari stranse intre driverele fisierelor de sistem si blocul cache deoarece sistemul cache este bazat pe pozitionarea relativa a datelor accesate pe disc, si fisierele un sunt neaparat stocate invecinat pe disc. Se paote observa activitatea de read-ahead prin utilizarea Cacheului: contorul de performanta Read Aheads/sec sau variabila de sistem CcReadAheadIos.

 Citirea urmatorului bloc a unui fisier care este accesat in mod secvential ofera in mod evident o imbunatatire a performantei, cu dezavantajul ca va genera head seeks.Daca se poate determina un pattern a citirilor aparent randomizate ale apelantului, cache managerul il extrapoleaza. De exemplu, daca apelantul citaste pagina 4000 si aopi pagina 3000, cache managerul presupune ca urmatoarea pagina pe care apelantul o va necesita va fi pagina 2000 si o va citi in avans.

 Pentru a face read-ahead si mai eficient, functia CreateFile a Win32 ofera un flag care indica acces la fisier secvential: FILE\_FLAG\_SEQUENTIAL\_SCAN. Daca acest flan este setat, cache managerul un pastreaza historia citirii in cache a apelantului pentru prezicere, dar in schimb executa read-ahead secvential. Cu toate acestea cat fisierul este citit in working setul cacheului, managerul cache demapeaza viewuri ale fisierului care nu mai este active si, daca este nemodificat, directioneaza managerul memoriei pentru a plasa paginile apartinand viewurilor nemapate inaintea listei de standby astfel incat sa poata fi refolosit rapid.

8.Exemplu:

 Virtual, fiecare sistem de operare modern cacheuieste datele fisierelor de sistem pentru a optimiza performantele. majoritatea programelor acceseaza date care sunt localizate in cateva fisiere locale. Daca sistemul de operare aduce aceste fisiere in memorie si le pastreaza acolo pe durata rularii programului, atunci numarul de citiri si scrieri pe disc pe care sistemul trebuie sa le imbunatateasca e minimizat. Fara operatia de cache, un program are cerinte pentru operatiile pe disc relativ costisitoare de fiecare data cand programul acceseaza fisiere de date.

 In mod implicit, Windows Server 2003 trece toate operatiile de citire si scriere pe disc prin memoria cache. Cache Managerul lucreaza indeaproape cu managerul memoriei si al fisierelor de sistem pentru a trece prin memoria cache fisierele sistemului de date in memorie.

 Microsoft Windows Server 2003 foloseste urmatoarele tipuri de sistem de stocare a datelor in memoria cache:

* Trecerea in memoria cache a blocului logic: aceasta operatie trece datele sectorului discului in meorie din memori cache
* Trecerea in memoria cache a blocului virtual: acesta operatie trece datele de la nivelul fisierului de sistem in locul nivelului de disc.

 **Marimea memoriei cache virtuale**

 Arhitectura IA-32 limiteaza marimea spatiului de adrese virtuale la 32 bits. Daca computerul este configurat cu switchul de 3GB, spatiul adreselor virtuale de 3GB  este rezervat pentru operatiile userului si doar 1GB e disponibil pentru operatiile in mod kernel. Dimensiunea memoriei virtuale depinde de urmatorii factori:

* Memoria fizica a computerului
* Valoarea de intrare in LargeSystemCache in urmatorul subkey de registrii:
* KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\MemoryManagement
* Setarea registrilor subkey ai PagedPoolSize in timpul initializarii
* Utilizarea a 3GB switch
* Bootarea dimensiunii paginii
* Marimea spatiului sesiunii care este necesar prntru configurarea terminalului  serverului

 In tabelul urmator este indicat modul in care se calculeaza :

* dimensiunea memoriei cache virtuale:

|  |  |
| --- | --- |
| Architecture | Virtual cache size |
| IA-32 | 128 megabytes (MB) + (Physical memory - 16 MB) / 4 MB x 64 MB |
| IA-64 | 128 MB + (Physical memory - 16 MB) / 8 MB x 64 MB |

* limita minima si maxima a dimesiunii memoriei cache:

|  |  |  |
| --- | --- | --- |
| Architecture  | Minimum virtual cache size | Maximum virtual cache size  |
| IA-32 (LargeCacheSize value = 0)  | Not available  | 512 MB  |
| IA-32 (LargeCacheSize value = 1)  | Not available  | 960 MB  |
| IA-64  | 128 MB  | 1 terabyte |

 **Dimensiuena fizica a memoriei cache**

Nu exista limita a dimensiunii fizice a memoriei cache la arhitecturile IA-32 si IA-64. Windows, pentru a aloca memorie cache se bazeaza pe urmatorii factori:

* Memoria cache de date virtuala
* Fisierele input/output din cache
* Alte cereri pe care sistemul le face pentru memoria fizica

 Memoria fizica care suporta memorie cirtuala cache de date paote fi golita si folosita pentru alte scopuri, asta depinzand de cerintele sistemului. Windows-ul confirma golirea momoriei cache de date de pe disc inainte ca Windows-ul sa returneze memoria fizica pentru alte scopuri. Cu cat cererea pentru cresterea memoriei de date virtuale cache este mai mare, cu atat mai multa meorie fizica este alocata pentru aceasta.

 **Golirea frecventa a memoriei cache**

 Daca Cache Managerul un incearca sa scrie fisiere modificate inapoi in fisier, si memoria libera devine insuficienta, thread-ul modificat de scriere al managerului de memorie scrie datele care nu au fost scrise inapoi in fisier. Sistemul nu se bazeaza pe managerul de memorie pentru a trimite datele de fisier inapoi pe disc. In schimb, Managerul Cache incearca sa scrie datele inapoi intr-un depozit nevolatil in timp util prin folosirea procesului de 'lazy writing'. Pe masura ce programul modifica datele de fisier, cahe managerul tine evidenta a cat de multe date sunt modificate, sau 'murdare'(compromise). Managerul Cache scrie inapoi o optime din datele modificate ale cachului inapoi pe disc la fiecare secunda

**9.Concluzii**

 Cache managerul ofera un mecanism high-speed si inteligent pentru reducerea de disk I/O si cresterea in mare a tranzitiilor sistemului. Prin cachuirea pe baza de blocuri virtuale, cache managerul poate executa read-ahead inteligent. Prin faptul ca se bazeaza pe fisierele primitive mapate ale managerului memoriei globale pentru a accesa date, managerul cache poate oferi mecanismul special fase I/O pentru a reduce timpul necesar CPUului pentru operatiile de scriere si citire si lasa si lucrurile legate de managementul memoriei fizice managerului memoriei Windows globale, astfel reducand duplicatiile de cod si crescand eficiente.

10. Bibliografie

* Microsoft Windows Internals 5th Edition

Mark E. Russinovich, David A. Solomon, Alex Ionescu

* <http://support.microsoft.com/kb/837331>
* <http://codesurgeonblog.com/2009/11/windows-cache-manager-quick-overview.html>
* <http://windowsitpro.com/systems-management/inside-cache-manager>