Universitatea Electronică, Telecomunicații şi Tehnologia Informației, Politehnica Bucureşti

Proiect Rețele de Calculatoare şi Internet

Tehnologia Web Cache

Perianu Dan - Claudiu

Master IISC an 2

ianuarie 2012

CUPRINS

Utilizarea cache-ului de rețea

Deşi volumul traficului pe web este stagnant, un mare procent din acest trafic este redundant – mai mulți utilizatori pentru un site dat au nevoie cam de acelaşi conținut. Aceasta înseamnă că un procent important din infrastructura WAN transportă un conținut identic (şi cereri identice pentru acesta). Eliminarea unui volum important de încărcare redundantă în telecomunicații înseamnă o oportunitate enormă de eficiență pentru companii şi provideri de servicii.

Caching web se ocupă cu stocarea locală a informațiilor de pe web, pentru a servi mai repede cererilor redundante ale utilizatorilor, fără a trimite cererile şi răspunsurile peste WAN.

Cache-ul de rețea este o tehnică de a ține informațiile des accesate într-o locație aproape de utilizator. Un cache web stochează paginile web şi conținutul acestora pe un dispozitiv de stocare care este fizic sau logic aproape de utilizator – mai aproape şi mai rapid decât o accesare pe web. Prin reducerea traficului pe legăturile WAN şi pe serverele supraîncărcate, cache-ul aduce beneficii importante ISP-urilor, companiilor de cablu şi utilizatorilor.

Beneficii principale

- *Reducerea costurilor datorită scăderii necesarului de lărgime de bandă pe WAN* – ISP-urile pot plasa noduri cache în puncte strategice din rețea pentru a îmbunătăți timpul de răspuns şi a scădea necesarul de lățime de bandă din rețeaua centrală. ISP-urile pot plasa soluții cache în puncte strategice de acces din WAN pentru a servi cererilor web de la un disc local mai degrabă decât de la servere web de la distanță. În rețelele companiilor, reducerea semnificativă din utilizarea lățimii de bandă prin folosirea cache-ului permite folosirea unei rețele WAN cu lățime de bandă mică (preț mai mic) pentru a servi aceiaşi utilizatori. Alternativ, o companie poate adăuga utilizatori sau mai multe servicii pentru a folosi lățimea de bandă liberă pe rețeaua WAN curentă.
- *Creşterea productivității pentru utilizatori finali* – răspunsul unul cache web local este de obicei de trei ori mai rapid decât timpul de descărcare al aceluiaşi conținut de pe WAN. Utilizatorul final poate vedea îmbunătățiri dramatice în timpul de răspuns, iar implementarea este complet transparentă pentru acesta.
- *Controlul accesului şi monitorizare securizate* – tehnologia cache oferă administratorului de rețea o metodă simplă şi securizată pentru a aplica o politică de acces a site-urilor prin filtrarea URL-urilor.
- *Logare operațională* – administratorul de rețea poate afla ce URL-uri au fost accesate, câte cereri pe secundă a servit cache-ul, ce procent de URL-uri a fost servit din cache şi alte statistici operaționale.

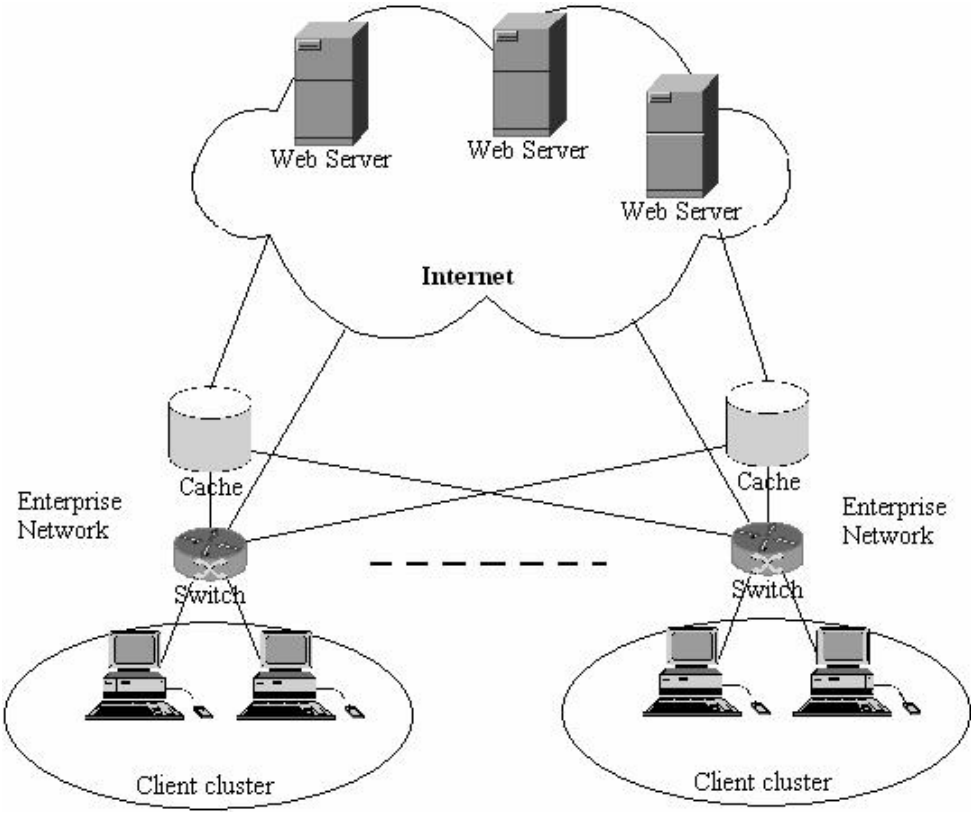

Fig 1. Un model de sistem web cache

Modul de funcționare a cache-ului web [2]

1) Un utilizator accesează o pagină web. 2) Rețeaua analizează cererea, apoi bazându-se pe anumiți parametri, redirectează transparent către un cache al rețelei locale. 3) Dacă cache-ul nu conține pagina web, va trimite o cerere web proprie către serverul web original. 4) Serverul original trimite conținutul către cache, care trimite conținutul către client, în timp ce salvează datele în memoria proprie. 5) Mai târziu, un alt utilizator accesează aceeaşi pagină web, rețeaua analizează cererea, şi după anumiți parametri, redirectează transparent către cache-ul rețelei locale. În loc de a trimite cererea peste Internet sau Intranet, cache-ul rețelei îndeplineşte cererea local. Acest proces accelerează livrarea conținutului.

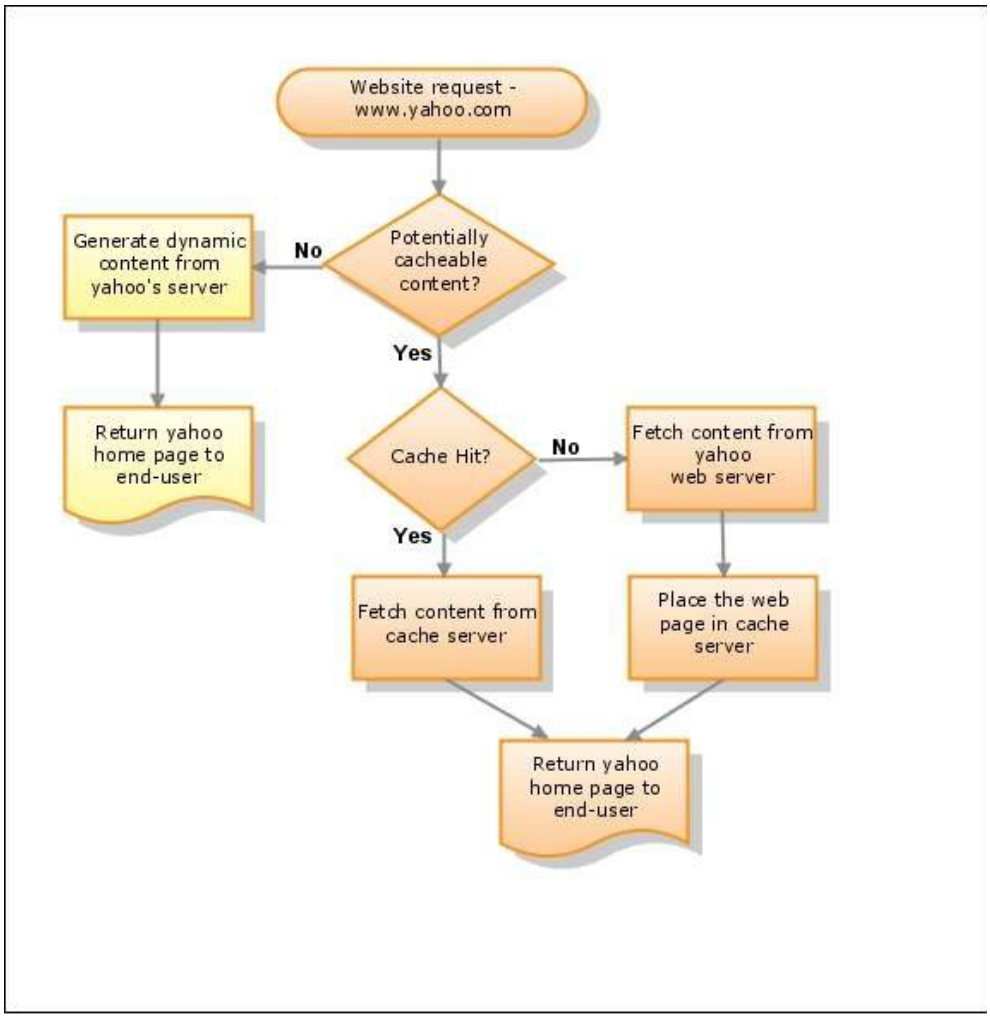

Fig 2. Traseul unei cereri web din perspectiva end-user

Sarcina importantă de a asigura că datele sunt curente este soluționată într-o varietate de căi, depinzând de arhitectura sistemului. Implementarea tehnologiei cache localizează modelele de trafic şi adresează problemele de supraîncărcare a traficului în rețea prin următoarele căi:

- Conținutul este adus utilizatorului la rate accelerate
- Optimizarea utilizării lățimii de bandă
- Administratorul de rețea poate monitoriza traficul cu uşurință

Serverele proxy

Serverele proxy sunt aplicații software ce rulează pe hardware şi sisteme de operare de uz general. Un server proxy este plasat pe un hardware care este din punct de vedere fizic între o aplicație client, cum este un browser web, şi un server web. Proxy-ul

se comportă ca un portar care primeşte toate pachetele destinate serverului web şi examinează fiecare pachet pentru a determina dacă nu cumva poate el însuşi să indeplinească acea cerere; dacă nu, va crea o cerere proprie către serverul web. Serverele proxy pot fi folosite şi pentru filtrarea cererilor, de exemplu, pentru a preveni angajaţii de a accesa un set specific de site-uri web.

Din păcate, serverele proxy nu sunt optimizate pentru cache şi nu sunt scalabile sub încărcare mare în rețea. În plus, fiindcă proxy-ul este în calea întregului trafic a utilizatorilor, apar două probleme: tot traficul este încetinit pentru a permite proxy-ului examinarea fiecărui pachet, şi erorile software sau hardware ale proxy-ului care determină ca toţi utilizatorii să piardă accesul la rețea. Proxy-urile au nevoie de configurare pentru browser-ul fiecărui utilizator – o sarcină de management nescalabil pentru furnizorii de servicii şi pentru companiile mari. În plus, serverele proxy care sunt aranjate într-o ordine ierarhică formează încă un strat în rețea, contrazicând planurile de a converge rețelele disparate într-o singură rețea unificată.

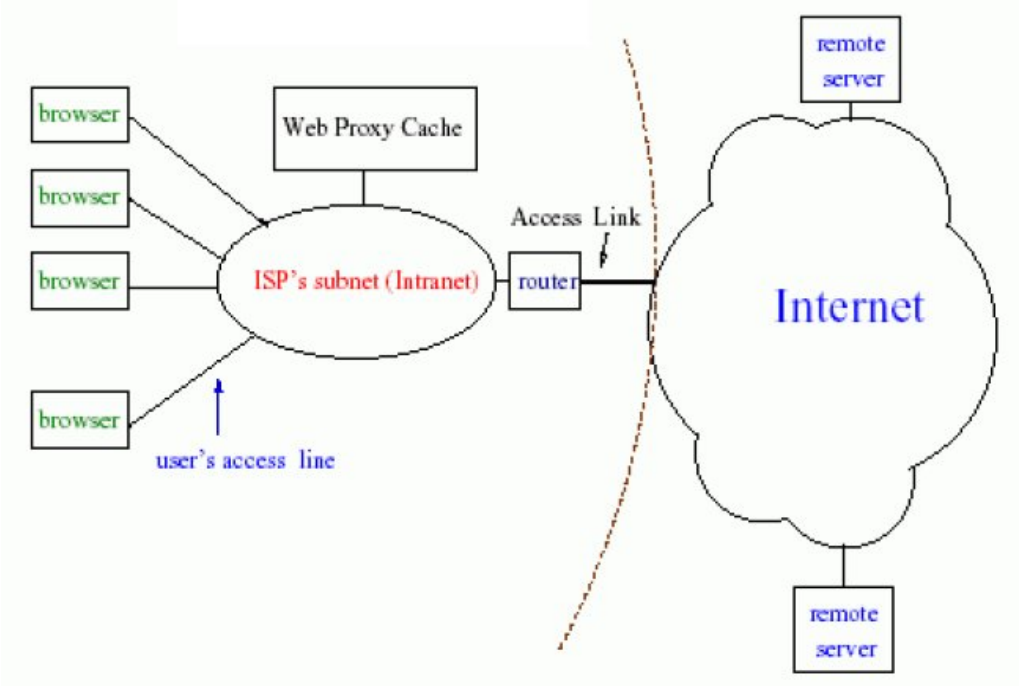

Fig 3. Infrastructură de interconectare a unei sub-rețele ISP (Intranet) de rețeaua Internet

Ca răspuns pentru problemele serverelor proxy, unii dezvoltatori au creat cache de sine stătător. Aceste aplicaţii software şi hardware orientate pe cache sunt construite pentru a creşte performanţa prin îmbunătăţirea soft-ului cache şi eliminarea altor aspecte mai puţin eficiente ale implementărilor serverelor proxy. În timp ce acesta este un pas în faţă, aceste cache-uri de sine stătătoare nu sunt integrate în rețea, rezultând un cost mai mare pentru folosirea lor şi fiind astfel mai puţin atractive pentru o implementare pe scară largă.

Cache de sine stătător

Aplicațiile browser-elor web permit utilizatorilor individuali să stocheze pagini web (imagini şi text HTML) pe discul local al acestora. Un utilizator poate configura mărimea spațiului pe disc dedicat cache-ului. Acesta este benefic în cazul în care utilizatorul accesează un site mai mult decât o singură dată. Prima dată când un utilizator accesează un site web, conținutul acestuia este salvat ca fişiere într-un subdirector pe hard disk-ul calculatorului. Data viitoare când utilizatorul accesează acest site web, browser-ul preia conținutul acestuia din cache fără să mai acceseze rețeaua. Utilizatorul poate observa că elementele din pagină – în special obiecte grafice mari ca butoanele, icon-urile şi imaginile – apar mult mai repede decât la prima accesare a paginii. Această metodă ajută mult un utilizator, însă alți utilizatori de pe aceeaşi rețea care accesează acelaşi site web nu vor avea nici un beneficiu.

Probleme specifice implementării cache-ului web

Calitatea serviciilor devine o proprietate din ce în ce mai importantă pentru utilizatorii World Wide Web. Fiind una dintre cele mai populare aplicații care rulează pe Internet, World Wide Web creşte în dimensiune, ceea ce produce congestie în rețea şi supraîncărcarea severelor. Web-ul este folosit în diferite moduri pentru a accesa o varietate de conținut, incluzând date multimedia, care este de timp real, şi pune o importanță mare pe întârzierea datelor (ex. Voice over IP, Video over IP) şi date de tip e-business, care au nevoie de răspuns rapid şi pierderi minime de date. Totuşi, datorită creşterii rapide în conținut şi număr de utilizatori, Internetul poate furniza doar servicii "best-effort" către utilizatori.

Web caching, o tehnică de stocare temporală a obiectelor Web (cum ar fi documentele Hypertext) pentru o accesare ulterioară, a fost introdusă peste tot în Internet şi s-a dovedit a fi o cale eficientă de a combate congestia de trafic şi supraîncărcarea serverelor, aşadar crescând disponibilitatea datelor web şi timpul de răspuns către client. Cache-u proxy este plasat la granița rețelelor organizațiilor, la capătul legăturilor cu întârzieri mari, sau la schimburi între sisteme autonome mari. Serverele web sunt, de altfel, replicate pentru a obține load-balancing. Totuşi, este inevitabil lipsa datelor din cache din cauza limitării de capacitate a acestuia şi schimbarea datelor de lucru, deci cache-ul este doar un serviciu de stocare în rețea de tip best-effort.

Problemele de calitate a serviciilor din sisemul de cache Web derivă din două surse. Prima, capacitatea cache-ului este foarte limitată comparativ cu interesul variat şi bogat al utiliazatorului. În al doilea rând, timpul CPU de execuție pentru fiecare proces este crucial în timpul supraîncărcării. S-au încercat diferite metode pentru a creşte performanța cache-ului. Acesta a fost distribuit în variate arhitecturi de sistem (ierarhic, distribuit total, hibrid), cu diferite protocoale de operare (ICP, Cache Digest, Summary Cache etc) pentru a distribui cache-ul mai eficient între clienți. Problemele de optimizare ca şi plasarea proxy în ierarhie, cache routing/ switching şi algoritmi de alocare/ înlocuire au fost studiate pentru a obține o performanță generală mai bună.

Soluții de implementare

Serviciile Intergrate (IntServ) şi Serviciile Diferențiate (DiffServ) sunt două metode folosite de Internet Engineering Task Force (IETF) pentru a adăgua Quality of Service rețelelor IP. Prin soluția IntServ, ruterele verifică traseul fiecărui pachet cu nivele diferite de servicii. Rețeaua acceptă sau refuză traficul bazat pe disponibilatatea resurselor cerute pentru a garanta nivelele de servicii pentru fiecare flux de date. Acest model de servicii suferă de probleme majore de scalabilitate, cost şi implementare. Mulți cercetători au trecut din această cauză la DiffServ, care se axează pe agregarea fluxurilor de date a diferitelor clase mai degrabă decât asigurarea managementului calității fiecărui flux.

Performanța proxy-urilor Web prin infrastuctura cache este crucială pentru asigurarea calității serviciilor. Aceasta depinde în mare măsură de felul cum algoritmul se potriveşte cu modelul schimbător de acces a proxy-ului web. Algoritmii cache sunt văzuți ca probleme de sortare care variază în cheia de sortare folosită. Ei provin din algoritmi clasici pentru cache din sistemele de fişiere sau baze de date, care folosesc atribute de bază a documentelor cum ar fi chei de sortare, recența în LRU (Least Recently Used), frecvența în LFU (Least Frequently Used) şi combinații ale celor două în LRU-K. După cum în Internet încărcarea de lucru este diferită față de cea din sistemele de fişiere sau bazele de date, unii algoritmi cache Web iau în considerare diferite caracteristici, spre exemplu, mărimea fişierului în GDS (Greedy Dual-Size) şi LRU-SIZE, costul de rechemare a obiectelor pierdute de la server la proxy în LRV (Least Relative Value). Din păcate, este greu de găsit un algoritm cache optimal pentru o variație largă de încărcări. Un algoritm cache trebuie să aibă capacitatea de a se adapta şi în acelaşi timp să țină o contabilitate eficientă. Un astfel de model este ACME, care combină eforturile mai multor algoritmi cache pe un server de date şi alege în mod dinamic cel mai potrivit algoritm conform sarcinii de lucru variabile. Modelul este adaptiv la diferite tipuri de sarcini de lucru şi topologii cache. Totuşi, deşi acest model deține virtual mulți algoritmi cache, suferă de o contabilitate greoaie.

LRU-K este un algoritm auto-adaptiv la schimbarea sarcinii de lucru. Acest algoritm a fost propus pentru buffer-e în Database Management Systems pentru a obține o rată de găsire mai mare, şi s-a dovedit a fi "optim față de algoritmii adaptivi bazați pe informația stohastică a referințelor trecute". LRU-K este auto-adaptiv şi necesită puțină contabilitate.

Există ideea că spațiul pe disc nu este momentan factorul limitativ în performanța serverelor proxy, datorită creşterii susținute a capacității discurilor şi viteza de răspuns a proxy-urilor. Aceasta, împreună cu faptul că majoritatea algoritmilor nu pot încă să se adapteze la toate sarcinile de lucru sau contabillitatea acestora este prea mare, determină ca cel mai popular algoritm cache din sistemele existente să fie încă LRU.

Totuşi, se poate observa că rata de găsire a cache-ului Web este proporțională cu logaritmul mărimii cache-ului, iar politicile de înlocuire diferă substanțial în performanțe.

Algoritmul de înlocuire LRU-K [1]

LRU-K este un algoritm de înlocuire a paginilor de sine stătător, derivat din clasicul Least Recently Used (LRU). A fost propus pentru a administra zonele de buffer în sisteme de baze de date. Acesta încorporează şi recenţa şi frecvenţa informaţiilor când ia decizii de înlocuire. Din cauză că algoritmul de buffering LRU aruncă pagina care nu a fost accesată pentru cel mai mult timp în momentul în care bufferul este necesar, algoritmul se limitează numai la timpul ultimei referinţe. Aşadar, LRU nu face diferenţa între referirile mai frecvente sau mai rare ale paginilor şi va menţine un timp îndelungat paginile referenţiate mai puţin frecvent. LRU-K este un algoritm optimal între toate modelele de algoritmi bazaţi pe informaţia stohastică a referirilor trecute.

Având în vedere limitările lui LRU clasic, cel mai bun estimat pentru timpul de sosire este intervalul de timp relativ la ultima referire, iar paginile cu cel mai mic astfel de interval sunt cele ce vor fi păstrate în buffer. Ideea de bază a LRU-K este să ţină cont de timpul pentru ultimele K referinţe ale paginilor populare, folosind aceste informaţii pentru a estima statistic timpul de sosire a referinţelor de tip pagină-cu-pagină.

Să presupunem că avem un set de pagini date, numite după un şir de numere întregi pozitive N={1,2,…n}, iar sistemul bazei de date pe care îl studiem face o succesiune de referinţe la aceste pagini specificate de şirul de referinţe r_1, r_2 ,... r_i ..., unde r_i = p(pЄN) means that r_i este o referinţă la pagina p. Desigur, fiecare pagină p are un timp de referinţă la sosire estimat, care este timpul între diferite apariţiile ale lui p în şirul de referinţă. Sistemul va menţine în buffer-ul de memorie doar paginile cu cel mai mic timp de accesare, sau echivalent probabilitatea cea mai mare de referinţă. Algoritmul LRU clasic poate fi considerat ca având o astfel de proprietate statistică, ţinând în memorie doar acele pagini care par au cel mai mic timp de sosire.

Algoritmul LRU-K specifică o politică de înlocuire a paginilor când un slot buffer este cerut pentru o nouă pagină de pe disc, astfel algoritmul specifică pagina ce va fi aruncată ca fiind cea cu distanţa Backward K, $b_t(p,K)$ este maximă între toate paginile din buffer. $b_t(p,K)$ este definit ca distanţa în urmă până la referinţa numărul K cea mai recentă a paginii p:

$$b_t(p,K) = \begin{cases} x & \text{dacă } r_{t-x} \text{ are valoarea p şi au trecut exact K-1 alte valori i cu t-x<i<t} \\ \infty & \text{dacă p nu apare de măcar K ori în } r_1, r_2, \ldots r_t \end{cases}$$

Singurul moment în care această alegere este ambiguă este atunci când mai mult de o pagină are distanţa egală cu infinit. În acest caz, o sub-politică poate fi aplicată, cum ar fi LRU, pentru a înlocui una din paginile cu distanţa Backward K infinită.

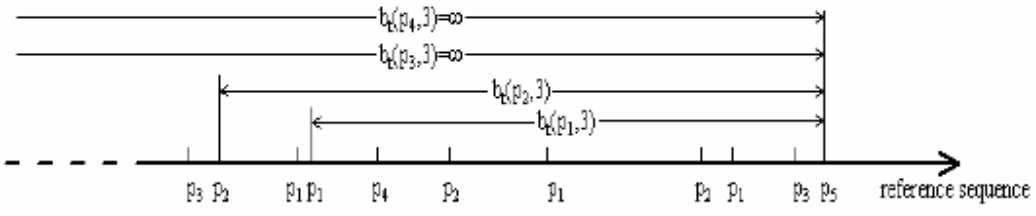

Fig 4. Un exemplu simplificat de distanță Backward K (K =3)

Figura 3 prezintă un exemplu simpificat de LRU-3 pentru o secvență de accesări pentru paginile p_1, p_2,..., p_n. Când o cerere pentru o pagină absentă p_5 ajunge, iar buffer-ul este încărcat, o pagină de aruncat este aleasă după distanța backward K de la punctul noii accesări. În acest exemplu, şi p_3 şi p_4 au această distanță un infinit, deci o politică subsidiară trebuie să facă diferența.

LRU-K ia în considerare mai mult decât istoria accesului fiecărei pagini pentru a face diferențierea între paginile care ar trebui ținute în cache, bazat pe presupunerea că şirul de caractere de referință este o secvență de accesări aleatoare cu distribuție Zipfian, şi fiecare pagină de disc p are o probabilitate bine definită, , să fie următoarea pagină referențiată de sistem. Schimbarea modalității de acces poate altera probabilitatea de referențiere a acestor pagini, dar probabilitatea are perioade relativ lungi de valori stabile şi se presupun independente de t. Din definiția lui LRU-K, este conjunctural ca pentru K>2, algoritmul LRU-K va avea o performanță mai mare decât LRU-2 pentru modele de acces stabile, dar este mai puțin flexibil la schimbări în modelul de acces, care poate fi foarte important pentru unele aplicații.

Trebuie să se țină seama de două cazuri speciale pentru K≥2 pentru a asigura un comportament corespunzator al cache-ului. Primul, numit Early Page Replacement, apare în situații când pagina abia citită în buffer-ul de memorie nu merită a fi păstrată în buffer după criteriile standard LRU-K, de exemplu are valoarea bt(p,K) aproape de infinit. Vom dori desigur să ştergem această pagină din cache cât mai repede, pentru a salva resurse de memorie pentru pagini mai importante. Totuşi trebuie să acceptăm că o pagină care nu este în mod general populară poate avea un grup de referințe corelate imediat după prima referențiere a sa. Algoritmul LRU-K adresează această problemă cu un parametru de expirare a corelării, astfel încât sistemul să nu piardă o pagină imediat după referențierea sa, ci să o țină pentru o perioadă de referențiere corelată (determinată de un parametru).

A doua situatie este faptul că nu este nevoie să se rețină istoricul referințelor pentru obiectele care nu sunt în acest moment prezente în cache. Aceasta este o distanțare față de algoritmii actuali de înlocuire a paginilor, şi este numită Page Reference Retained Information Problem. Algoritmul LRU-K adresează această problemă cu un parametru, Retained Information Period, astfel încât sistemul menține informația istorică a oricărui obiect pentru această perioadă după cel mai recent accces al său. Elimină posibilitatea referențierii repetate a unei pagini imediat după ce a fost ştearsă negăsindu-se nici o înregistrare a unei referențieri trecute şi o şterge fiindcă de fiecare dată distanța K este estimată ca fiind infinit.

Internet Cache Protocol (ICP) [4]

ICP este un format de mesaje folosit pentru comunicarea între cache-urile Web. Deşi cache-urile Web folosesc HTTP pentru a transfera datele, ele pot beneficia printr-un protocol de comunicație mai simplu. ICP este folosit de obicei în rețele mesh pentru a localiza obiecte specifice din Web în cache-urile învecinate. Un cache trimite o cerere ICP către vecinii săi. Aceştia trimit răspunsuri ICP, indicând "HIT" sau "MISS". O cerere sau un răspuns ICP trebuie să vină foarte repede, de obicei într-o secundă sau două. Un cache nu poate aştepta mai mult de atât înainte de a începe returnarea obiectului. Neprimirea unui mesaj de răspuns înseamnă cel mai probabil că drumul pe rețea este ori congestionat ori blocat. În oricare dintre cazuri nu se doreşte selctarea acelui vecin. Ca şi indicare a condițiilor de alăturare în rețea între cache vecine, ICP peste un protocol ca UDP este mai bun decât cel cu supraîncărcare ca TCP-ul.

În plus față de utilizarea ca un protocol de localizare a obiectelor, mesajele ICP pot fi folosite pentru selectarea cache. Neprimirea unui mesaj de răspuns de la un cache poate indica o eroare de rețea sau de sistem. Răspunsul ICP poate include informație care asistă selectarea a celei mai apropiate surse de la care să se returneze un obiect.

Formatul mesajului ICP

Formatul mesajului ICP este format dintr-un header fix de 20 octeți plus o mărime variabilă de supraîncărcare.

Opcod	Versiune	Lungime mesaj
Număr cerere		
Opțiuni		
Date opțiuni		
Adresă emițător		
Încărcătură utilă		

Fig 5. Formatul mesajului ICP

Opcod – unul din opcod-urile de mai jos

Valoare	Nume
0	ICP_OP_INVALID
1	ICP_OP_QUERY
2	ICP_OP_HIT
3	ICP_OP_MISS
4	ICP_OP_ERR
5-9	UNUSED
10	ICP_OP_SECHO
11	ICP_OP_DECHO
12-20	UNUSED
21	ICP_OP_MISS_NOFETCH
22	ICP_OP_DENIED
23	ICP_OP_HIT_OBJ

Versiune – numărul versiunii protocolului ICP.
Lungime mesaj – lungimea în octeți a mesajului ICP; nu poate depăşi 16.384 bytes.
Număr cerere – un identificator opac. Când se răspunde unei cereri, această valoare trebuie copiată în mesajul de răspuns.
Opțiuni – un câmp de 32 biți pentru flag-uri de opțiuni, care permit extensia versiunii protocolului în moduri stabilite.
Date opțiuni – un câmp de patru bytes care menține opțiuni suplimentare.
Adresa emițător – adresa Ipv4 a calculatorului care trimite mesajul ICP. Acest câmp probabil nu ar trebui să aibă întâietate peste ce se returnează prin getpeername(), accept() şi recvfrom(). Există puțină ambiguitate asupra scopului original al acestui câmp. Nu este folosit în practică.
Încărcătură utilă – conținutul acestui câmp variază depinzând de opcod, dar cel mai probabil conține un şir de caractere URL terminat în null.

Extinderea protocolului ICP original [5]

Au fost propuse trei metode de a extinde protocolul original pentru a îl adapta noii metode de transmisie şi stocare a datelor.

1. *Includerea PPSP (Peer-to-Peer Streaming Protocol) în protocolul ICP*

Se poate folosi un identificator pentru a arăta dacă protocolul este ICP original sau unul nou, cu PPSP inclus în el. Se poate folosi doar unul din cei 8 biți ai Opcod-ului pentru identificare. De exemplu, când identificatorul este setat pe 0, se consideră mesajul ca fiind unul ICP, iar când este setat pe unu, este un mesaj PPSP.

Se poate extinde Opcod-ul folosind valorile nefolosite pentru exprimarea informațiilor PPSP:

Valoare	Nume
5	FIND
6	CHUN_AVAILABILITY
7	PROPERTY_QUERY
8	TRANSPORT_NEGOTIATION
12	SUCCESSFUL (OK)
13	INVALID_SYNTAX
14	VERSION_NOT_SUPPORTED
15	MESSAGE_NOT_SUPPORTED
16	MESSAGE_FORBIDDEN

Când identificatorul de 1 bit este setat pe unu, valorile de desupra vor fi activate şi cache-ul care primeşte mesajul ICP se schimbă pentru interpretarea unui mesaj PPSP. De altfel, când folosim mesajul PPSP, mesajul de răspuns trebuie să conțină ID-ul obiectelor în corpul său, în loc de URL-ul din formatul ICP.

2. *Co-existența PPSP şi ICP în comunicația cache*

În această metodă, daca nodurile cache păstrează documente, când va fi apelat de alți utilizatori, cache-ul va folosi protocolul ICP pentru comunicație; dacă va stoca bucăți (chunk) de date, atunci cache-ul va comunica prin protocolul PPSP.

Totuşi, cache-urile au nevoie de o entitate funcțională, numită Probator de Tip de Date, pentru a identifica tipul datelor (document/ obiect). În acest scenariu, fiecare cache trebuie să informeze Probatorul despre tipul de date pe care îl stochează. Când se inițializează o comunicare, Probatorul va decide ce tip de mesaj se utilizează în concordanță cu tipul de date din cache.

3. Utilizarea PPSP pentru înlocuirea ICP

În acest caz se va folosi protocolul PPSP pentru a face schimbul de informații între cache-uri. Totuşi, trebuie să fie extins protocolul PPSP existent din moment ce nu susține localizarea datelor prin URL în prezent.

Acest caz seamănă cu cazul 1, doar că se va încerca rescrierea protocolului PPSP în loc de ICP. Se va introduce un nou membru în câmpul mesajului PPSP, numit "TYPE", pentru a indica tipul de date din cache. Se va extinde protocolul PPSP prin adăugarea Opcod-ului ICP în câmpul de metode.

Când TYPE va indica că datele din cache sunt de tip documet, aceste Opcod-uri se vor activa, altfel protocolul va fi un PPSP normal.

BIBLIOGRAFIE

1. Dong Zheng, *Differentiated Web Caching – A Differentiated Memory Allocation Model on Proxies*, July 2004
2. Brian D. Davison, *A Web Caching Primer*, IEEE, Volume 5, Number 4, July/August 2001, pages 38-45
3. *Web Caching – A cost effective approach for organizations to address all types of bandwidth management challenges*, A ViSolve White Paper, March 2009
4. D. Wessels, K. Claffy, *Internet Cache Protocol (ICP), version 2*, RFC 2186, Septmebrie 1997
5. Y. Zhang, L. Gui, J. Peng, C Schmidt, L. Xiao – *Extended Internet Cache Protocol to support today's content based cache exchange*, Internet-Drafts, 1 July 2011