مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (40) العدد (5) 2018
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (40) No. (5) 2018

مقارنة اداء توابع الاستبدال لخوارزميات كاش القرص الصلب وكاش الويب

الدكتور علي سليمان*

(تاريخ الإيداع 7 / 8 / 2018. قُبِل للنشر في 17/ 9 / 2018)

□ ملخّص □

تم في هذه الورقة مقارنة خوارزميات الكاش الأساسية من حيث الأداء والسرعة لأغراض التخزين المؤقت لأغراض الويب ذات المحتوى الديناميكي وأغراض التخزين المؤقت في القرص الصلب، من خلال دراسة الخوارزميات التقليدية في هذا المجال، بهدف تحديد مدى الاستفادة من الخوارزميات الأساسية في مجال التخزين على الأقراص الصلبة وفي مجال كاش الويب، بينت النتائج أن الخوارزميات ذات توابع الاستبدال التي تعتمد المؤشرات الأساسية (مثل **LRU**، **LFU**) تعطي نتائج أفضل في التخزين المؤقت لأغراض التخزين في الأقراص الصلبة في حين تحتاج خوارزميات كاش الويب إلى معايير إضافية لعمل تابع الاستبدال للحصول على مؤشرات أداء عالية. كما تبين أن خوارزميات الكاش في الويب تعطي أداء أخفض منه في القرص الصلب وعليه تبرز الحاجة إلى تطوير خوارزمية كاش الويب بشكل دائم.

الكلمات المفتاحية: كاش – كاش القرص الصلب – كاش الويب – توابع الاستبدال

*أستاذ مساعد –قسم الهندسة الطبية– كلية الهندسة الميكانيكية والكهربائية– جامعة تشرين– اللاذقية– سورية

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (40) العدد (5) 2018
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (40) No. (5) 2018

Comparison of The Performance of The Replacement Functions for Hard Disk Cache and Web Cache algorithms

(Received 7 / 8 / 2018. Accepted 17 / 9 / 2018)

□ ABSTRACT □

In this research paper, we compare the basic cache algorithms in terms of performance and speed for the purposes of web caching for dynamic content and hard disk buffering purposes, by studying the traditional algorithms in this field, in order to determine the utilization of the basic algorithms in disk storage in the field of web caching. The results shows that algorithms with replacement functions that rely on basic indicators (such as LRU, LFU) give better results in storage for storage purposes in hard drives, while web caching algorithms need additional benchmarks for replacement work to get high performance indicators, Web Cache algorithms also show lower performance then that hard drive, so the need to constantly develop the Web cache algorithm.

Keywords: cache, hard disk cache , web cache , replacement functions

مقدمة

إن العديد من عوامل الأداء المتعلقة بنظم استرجاع البيانات، يمكن أن تحل من خلال تقنية التخزين المؤقت، التخزين المؤقت هو عبارة عن تقنية مستخدمة بشكل واسع، عندما يتم الوصول إلى البيانات من قبل أنظمة المعلومات*[1]* . مثلاً باعتبار لدينا ذاكرة الكاش لمعالج ما وذاكرة الكاش لنظام ملفات ما وغيرها، تعمل هذه الذواكر على تخزين البيانات من الذواكر الأبطأ في ذاكرة أسرع ولكن أصغر نسبياً، وبذلك فإن زمن الوصول للبيانات المخزنة بشكل مؤقت يقل بشكل ملحوظ.

إذا تم الاختيار المناسب للبيانات، التي يتم تخزينها في ذواكر الكاش، فإنه من الممكن الحصول على سرعة كبيرة جداً عند الوصول للبيانات.

مشكلة البحث

على الرغم من وجود العديد من نقاط التشابه بين التخزين المؤقت التقليدي، والتخزين المؤقت لأغراض الويب، فإن الاختلافات فيما بينها هو أكبر باعتبار أن التخزين المؤقت لأغراض الويب أكثر تعقيداً.

على سبيل المثال باعتبار لدينا ذاكرة كاش لمعالج ما تستخدم لتخزين البيانات من الذاكرة الأساسية، فإن البيانات المخزنة لها حجم موحد وزمن استجابة الذاكرة الأساسية ثابت دوماً. في حين أن البيانات المخزنة بشكل مؤقت من الويب تختلف في حجمها، مثلاً الاختلاف في الحجم بين ملف فيديو وصفحة *html*.

ولتكون الأمور أكثر تعقيداً أيضاً فإن التخزين المؤقت لأغراض الويب يحتاج إلى تخزين البيانات من مصادر عديدة مع الاختلاف الشديد لزمن نقل البيانات بالاعتماد على عرض الحزمة، حجم العبور الكلي وحمل المخدم.

أهمية البحث وأهدافه:

أهمية البحث

على الرغم من أن سياسات الاستبدال المناسبة تعمل بشكل جيد عند تطبيقها على ذواكر كاش المعالجات، إلا أنها ليست ناجحة بنفس الدرجة عند تطبيقها على التخزين المؤقت لأغراض الويب. والسبب أنها ليست معدة من أجل التعقيد الأعلى في هذا النوع من التخزين.

من أجل كل خوارزمية تخزين مؤقت لأغراض الويب فإنه من الضروري أن يؤخذ بعين الاعتبار حجم الملفات المخزنة بشكل مؤقت، كما أن الاستخدام الكفؤ للذاكرة يؤثر بشكل حرج على أداء ذاكرة الكاش.

باعتبار لدينا خوارزمية تقييم الملفات على أساس الزمن اللازم لتحميلها، ولدينا ملفان للتخزين يختلفان بشكل ملحوظ في الحجم، فإن الملف الأكبر سيكون له قيمة تقدير أكبر من الملف الصغير، وبالتالي عندما يكون الاسترداد ضروري فإن الملف الصغير سيسترد محرراً قدر صغير من الذاكرة.

أهداف البحث

يهدف البحث إلى تقييم أداء مجموعة من سياسات الاستبدال المختلفة لخوارزميات الكاش وتحديد الأفضل منها، والتي ستحسن وضع الخدمات التي تقدمها نظم المعلومات، وتسريع للعمل على أرض الواقع بما يتوافق مع نماذج طلبات ووصول المستخدمين *user request and access patterns* إلى هذه النظم والمقارنة بين أداء هذه الخوارزميات في التخزين المؤقت للقرص الصلب والتخزين المؤقت لصفحات الويب.

طرائق البحث ومواده

تمّ اتباع المنهج التجريبي *Experimental Method* للتحقق من فرضيات البحث، حيث تمّ تثبيت المتغيرات المستقلة، ودراسة القيم التي تمّ الحصول عليها للمتغيرات التابعة، كما تمّ اتباع المنهج الوصفي *Descriptive Method* من خلال استخدام التحليل الإحصائي، وذلك للتحقق من مواصفات سجلات وصول المستخدمين إلى المخدمات، ومدى تطابقها مع مجموعات البيانات المستخدمة في الدراسات المرجعية.

مجتمع وعينة البحث

تم اختبار الخوارزميات من خلال تصميم محاكي يقوم بمراقبة الطلبات الواصلة إلى القرص الصلب في مخدم نظام معلومات ويدرس إمكانية تلبيتها من الخزن الجانبي *buffer* بدلا من استعادة المعلومات من القرص الصلب مباشرة، وذلك وفق سياسة تابع الاستبدال للخوارزمية المدروسة، كما تم اختبار خوارزميات كاش الويب باستخدام محاكي تم تصميمه على مخدم ويب خاص، حيث يقوم المحاكي بدراسة الطلبات الواصلة إلى مخدم الويب، وتحديد إمكانية تلبيتها من الكاش بدلاً من مخدم الويب حسب سياسة الاستبدال المحددة للخوارزمية.

خوارزميات الكاش الأساسية

1- خوارزمية LRU

LRU هي اختصار لـ *Least Recently Used*، والتي تعني الأقل استخداماً، وقد تمّ اشتقاقها مباشرةً من التخزين المؤقت لنظام الملفات *File System Caching*. تقوم خوارزمية *LRU*على مبدأ أن الصفحات الأقل استخداما سوف يتم إخراجها من ذاكرة الكاش، حيث أنّ الصفحات التي تم استخدامها بشكل كبير في الفترة الأخيرة يكون احتمال إعادة استخدامها مجدداً كبير أيضاً في المرات المقبلة *[2]* .

Pseudo-code for LRU

If p is in the buffer then LAST(p) = current time;
Else
i) Min = current time + 1;
ii) For all pages q in the buffer do
a) If (LAST(q) < min)
victim = q
Min = LAST(q)
iii) If victim is dirty then flush it to disk
iv) Fetch p into the buffer frame held by victim
LAST(p) = current time

2- خوارزمية LFU

LFU هي اختصار لـ *Least Frequently Used* والتي تعني الأقل تكراراً، وهي تقوم على مبدأ استبعاد الصفحات التي تملك أقلّ عدد مرات استخدام، بغض النظر عن حداثة استخدامها مقارنة مع الخوارزمية السابقة، إذ يتم في هذه الخوارزمية الاحتفاظ بعداد من أجل كل غرض (صفحة) ويتم طرد واستبعاد الأغراض التي تملك العداد ذو القيمة الأدنى وهذه العملية تتم عند اتخاذ القرار بإجراء عملية التبديل*[3]* .

LFU Pseudo-Code
Devide every value(Page) in cache by 2 if Page in cache value(Page) = R + value(Page) EXIT while cachefree<filesize(Page)

```
find and remove candidate in cache with smallest value
cachefree = cachefree + filesize(candidate)
insert value(FILENAME) = R
```

3-خوارزمية Greedy Dual

إنّ خوارزمية *GreedyDual* هي تعميم للخوارزمية المعروفة *LRU* مراعيةً احتياجات التخزين المؤقت لأغراض الويب.

تقوم الخوارزمية على مبدأ الاحتفاظ بقيمة تقديرية $H(p)$ لكل مستند أو صفحة p تم تخزينها. عندما يتم تخزين المستند تسند لقيمته التكلفة التي حصلت نتيجة تخزين p وبالتحديد $c(p)$. وعندما تبرز الحاجة إلى إزالة مستند من التخزين فسيتم اختيار المستند ذو قيمة H الأدنى، ويتم طرد هذا المستند ويتم إنقاص قيمة H لكل مستند متبقي في التخزين بمقدار قيمة H للمستند المطرود.

في أي وقت يتم فيه طلب مستند p تم تخزينه فإن قيمة $H(p)$ تعاد إلى $c(p)$ *[4]*.

4- خوارزمية GreedyDual-Size

هي خوارزمية معدلة عن خوارزمية *GreedyDual* ويكمن الفرق الجوهري بينهما في طريقة تركيب القيمة التقريبية H.

عندما نقوم بتخزين أو إعادة الوصول إلى مستند p، فإن قيمة $H(p)$ تصبح

$$\frac{C(p)}{S(p)} \qquad (1)$$

حيث أن $S(p)$ هو حجم المستند p وبالتالي يتم هنا أخذ حجم المستند بعين الاعتبار.

إنّ تأثير هذا التغيير في الخوارزمية الأصلية يظهر عند القيام بتخزين المستندات ذات الأحجام المتكافئة، حيث أن الكبيرة منها يرغب بإزالتها قبل الصغيرة منها (طبعاً بعد الأخذ بعين الاعتبار النفاذ إلى هذه المستندات وقيمة H) مما يزيد من كفاءة استخدام الذاكرة *[5]* .

Algorithm GDS

Initialize L=0
Process each requested document in turn:
Let p be the current document:
IF p is already in cache
K(p) = L + C(p)/S(p)
ELSE
WHILE there is not enough space for p
Let L = mink(qj), for all qj in cache
Evict q such that K(q) = L
END WHILE
Doposit p into cache and set
K(p) = L + C(p)/S(p)
END IF

5- خوارزمية GDSF

GDSF هي تطوير لخوارزمية *GreedyDual* السابقة من خلال إضافة تردد طلب المستند *Frequency* إلى قيم المفتاح *Key*.

إنّ قيمة المفتاح للمستند P تحسب وفق المعادلة التالية :

$$K(p) = L + F(p) * C(p)/S(p) \qquad (2)$$

حيث $F(p)$ هو عدد مرات النفاذ للمستند. عندما يتم استعادة المستند p إلى الذاكرة لأول مرة فإن قيمة التردد الخاص به $F(p)$ تكون مساوية لـ 1.

إذا تم طلب النفاذ إلى p ، مجدداً في الذاكرة، فإن قيمة التردد الخاص به تزداد بـ 1:

$$F(p) = F(p) + 1 \qquad (3)$$

حيث أن $S(p)$ هو حجم المستند p وبالتالي يتم هنا أخذ حجم المستند بعين الاعتبار.[6]

نلخص خطوات الخوارزمية وفق الأتي:

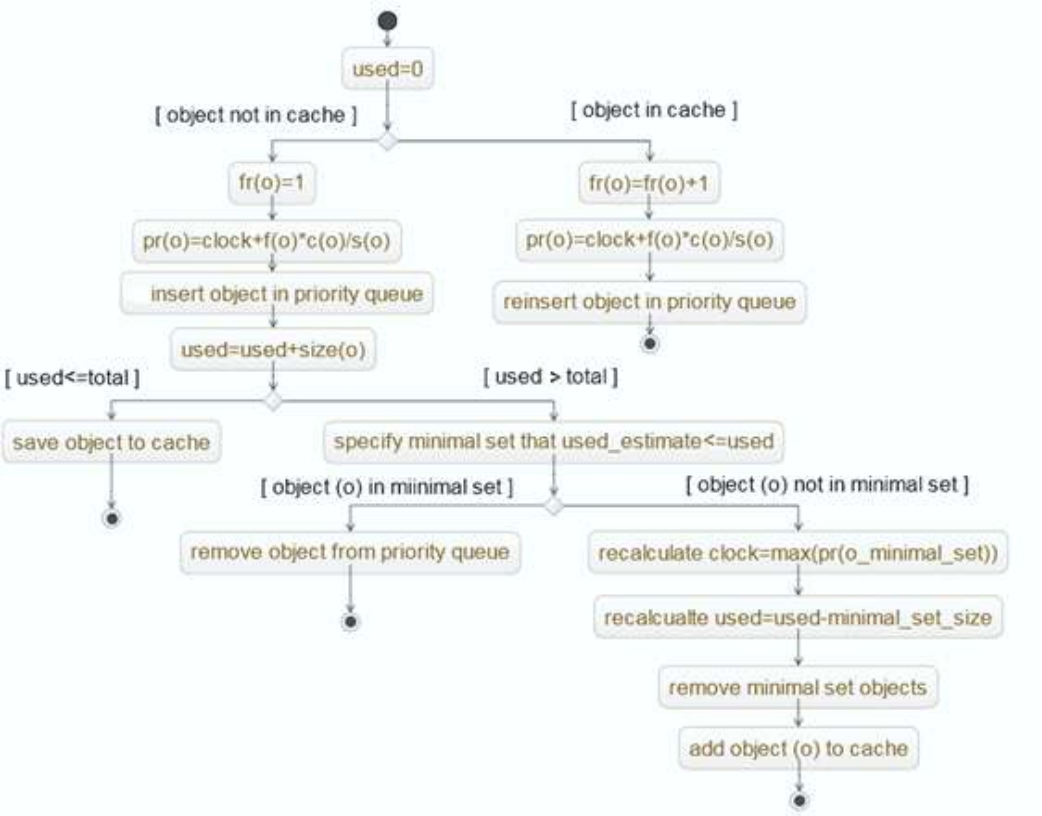
توجد ذاكرة كاش بحجم معين، يوجد بها جزء مستخدم وفي البداية يكون صفر، لكل غرض في الكاش عداد يحدد عدد مرات الوصول إليه، عند إدخاله للمرة الأولى يسند له القيمة 1، عند الحاجة لحشر غرض ولم يعد هناك حجم فارغ كافي سيتم إنشاء رتل أولوية وفق المعادلة 2 حيث L هى clock ساعه للرتل تبدأ من الصفر ويتم تحديثها من أجل كل غرض يتم استبداله إلى قيمة مفتاح الاولية له في رتل الاولويات، عند الحصول على أية غرض من الكاش عندها يزداد عداد تردد الغرض بمقدار 1، لكل غرض قياس للحجم والكلفة.

1. إذا كان الغرض المطلوب موجود في الكاش عندها يتم إيصال الغرض إلى الزبون من الكاش وهنا كمية الكاش لا تتغير، تردد الغرض يزداد بمقدار 1، مفتاح الأولية يحسب من المعادلة 2، clock لايتغير، يعاد حشر مفتاح الأولوية المحوب في رتل الأولوية).
2. إذا كان الغرض المطلوب غير موجود في الكاش عندها يتم إيصال الغرض للزبون من المخدم الأساسي (يسند 1 لتردد الغرض، يحسب مفتاح الأولوية من المعادلة 2، بحشر الغرض في رتل الأولوية ويعاد تقييم الاستخدام).

هنا نجد حالتين:

a. وجود مساحة كافية في الكاش لتخزين الغرض وعندها لا يحدث أي استبدال لأي غرض في الكاش ويتم تخزين الغرض في الكاش.
b. لا توجد مساحة كافية في الكاش لتخزين الغرض ويوجد مجموعة من الأغراض يتوجب استبدالها (يتم تحديد مجموعة أصغرية مع أولوية اصغرية ومجموع حجومها كافية للغرض المحشور ونحذفها ونحشر الغرض).

3. إذا كان الغرض الأصلي الذي نريد إدخاله إلى الكاش ليس بين الأغراض الموجودة فيها تجرى مجموعة من الحسابات (تحسب clock، وكمية الكاش المستخدمة، ويتم وضع الغرض في الكاش).
4. إذا كانت أولوية الغرض الأصلي الذي نريد إضافته إلى الكاش بين أولويات الأغراض الموجودة عندها لا يتم تخزين الغرض في الكاش ولا يتم حذفه من الرتل، لا يتم حذف غرض آخر من الكاش. ترسم هذه الخوارزمية كما هو في الشكل (1).



الشكل (*1*) مخطط خوارزمية *GDFS*

6-خوارزمية *LRU-K*

خوارزمية *LRU-K* هي خوارزمية استبدال صفحات في الذاكرة مستقلة تماماً وقادرة على إنجاز مهامها بشكل كامل ومتقن. هذه الخوارزمية مشتقة من الخوارزمية التقليدية المعروفة *LRU*. وقد كانت تستخدم لإدارة مساحات التخزين الموجودة في نظم إدارة قواعد البيانات *[7]* .

تقوم هذه الخوارزمية بدمج كل من مبدأي الحداثة *Recency* والتكرار أو التردد *Frequency* عند اتخاذ قرارات التبديل والإزاحة في الذاكرة، وبما أن خوارزمية *LRU* تقوم على مبدأ إزاحة الصفحة، التي لم يتم استخدامها منذ زمن طويل، وذلك عند بروز الحاجة إلى مكان هذه الصفحة، وبالتالي، فهي تقيد نفسها بزمن آخر عملية نفاذ، وهي لا تميز بوضوح بين الصفحات المتكررة وغير المتكررة في اللائحة خلال فترة زمنية طويلة.

تقوم فكرة خوارزمية *LRU-K* على مبدأ أن تعقب آخر مرجع لـ *K* الخاص بصفحات الويب المستعرضة من قبل المستخدم، عندما يمتلئ المخزن المؤقت تقوم *LRU-K* باستبدال الصفحة ذات *K* الرئيس، ولكن قد تظهر فرصة لمجموعة أخرى من الصفحات للحصول على الطول الأعظمي من صفحة الويب الرئيسة، في هذه الحالة ولمعرفة أي صفحة يجب اختيارها، فإن اعتبارات مختلفة يمكن اخذها بالحسبان. تقوم خوارزمية *LRU-K* بالتقاط صفحات الويب المستخدمة أخيراً، والخيارات الأفضل يمكن ان تتخذ بمساعدة الصفحات المستخدمة مؤخراً، وبهذه الطريقة تستطيع *LRU-K* تحديث وتحسين عملية التخصيص هذه.

7-خوارزمية *LFU-Aging*

هي شكل من أشكال الخوارزمية *LFU*، وهي تأخذ بالاعتبار عاملين أساسيين هما:

- تردد الوصول إلى الغرض *(Frequency)*

- عمر الغرض في الذاكرة *(Age)*

تقوم خوارزمية *Aging* بفهرسة وعنونة تلوث التخزين، والذي قد يحصل عند تحول مجموعة الأغراض الشائعة ويتم حل هذه المشكلة كما يلي:

تتألف خوارزمية *Aging* من بارامترين أساسيين هما:

Mref: وهو العدد الأعظمي الذي يستطيع أي غرض أن يصل اليه بعدد مرات الوصول اليه. إذا وصل العداد المرجعي لغرض في التخزين إلى هذا الرقم فإن أي طلب لهذا الغرض لن يزيد هذا العداد أبداً. يسعد هذا البارامتر يساعد على منع وصول العداد المرجعي للغرض إلى أعداد كبيرة*[8]* .

البارامتر الثاني *Amax* : وهو المعدل الأعظمي لعدد العدادات المرجعية لجميع الأغراض في التخزين *cache*. عندما يتم الوصول إلى هذه القيمة فإن العداد المرجعي لكل غرض يتم انقاصه بنسبة معينة.

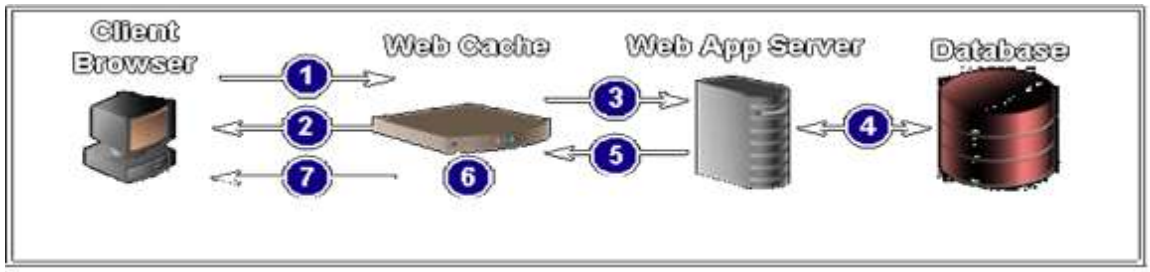
وبالتالي فإن الأغراض الشائعة السابقة لن تحافظ على مكانتها بعد ذلك، حيث أن عداداتها المرجعية سيتم إنقاصها، وبالتالي ستعود لها إمكانية الاستبدال في حال وجود أغراض أكثر شيوعاً منها.

التخزين المؤقت لصفحات الويب

يعرّف كاش الويب أو التخزين المؤقت لصفحات وأغراض الويب، بأنّه نظام حاسوبي مخصص لمراقبة طلبات الأغراض (أغراض الويب) وتخزين هذه الأغراض، بحيث يتم استردادها من المخدم، حيث سيقوم الكاش بتسليم الأغراض من المخزن بدلاً من إعادة تمرير الطلب إلى المخدم المعني في الطلبات اللاحقة [9].

هنالك العديد من فوائد تطبيق خوارزميات كاش الويب، ذلك أنّ طلب البيانات وتخديمها من قبل الكاش دون العودة إلى الموقع الأصلي للبيانات يوزع الحمل ويخففه عن المصادر الأساسية، التي هي عادة مخدمات تتعرض لضغط من ناحية محرك قواعد البيانات أو عرض الحزمة المستخدمة أو الموارد الأخرى كأقراص التخزين والذواكر، حيث يمكن استخدام مخازن كاش متزامنة *concurrent cache stores*، وبالتالي تسريع زمن الاستجابة لطلب المستخدم بالإضافة إلى إمكانية تصميم الكاش بطريقة تستفيد فيها من بنية الشبكة لتقليل عرض الحزمة المستخدم على كامل الشبكة. يبين الشكل (2) نموذج مخدم – زبون مع كاش ويب حيث تتم عملية الطلب والإجابة من المخدم وفق الخطوات التالية[10]:

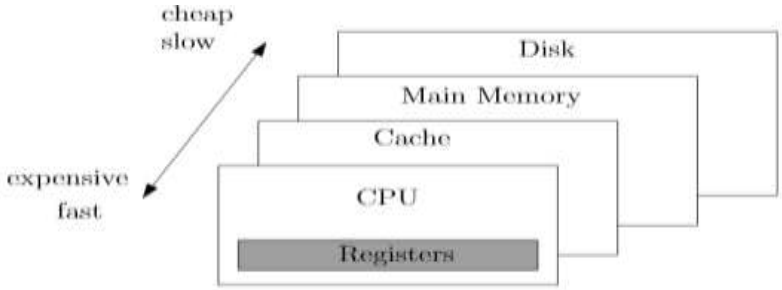
- يقوم الزبون بإرسال طلبات *HTTP*.
- يستجيب مخدم كاش الويب مباشرةً في حال كان الغرض متاحاً في الكاش.
- يتم تحديث البيانات التي تصف العناصر المخزنة في مخدم الكاش.
- إذا لم يكن الغرض في الكاش يقوم كاش الويب بطلب الغرض من مخدم التطبيقات.
- يقوم مخدم التطبيقات بتنفيذ المطلوب والاستجابة لكاش الويب.
- إذا أمكن تخزين الاستجابة في الكاش، فإنّه يتم الاحتفاظ بها من أجل طلبات لاحقة، ويتم إخراج العناصر من الكاش وفق سياسة الاستبدال للخوارزمية المحددة لعمل المخدم.
- يقوم كاش الويب بتحضير الصفحات من أجل الاستجابة لطلبات الزبائن وفق الخوارزمية المحددة له.



الشكل (2) نموذج مخدم – زبون مع تقنية كاش الويب.

التخزين المؤقت للقرص الصلب

يبين الشكل (3) هرمية الذاكرة في نظام الحاسوب وموقع القرص الصلب ضمن هذه الهرمية، وحيث أنه هو وسيط التخزين الأكثر بطئا، فان الحاجة إلى وجود كاش للقرص الصلب مع سياسة استبدال ناجحة يعتبر أمرا ضروريا.

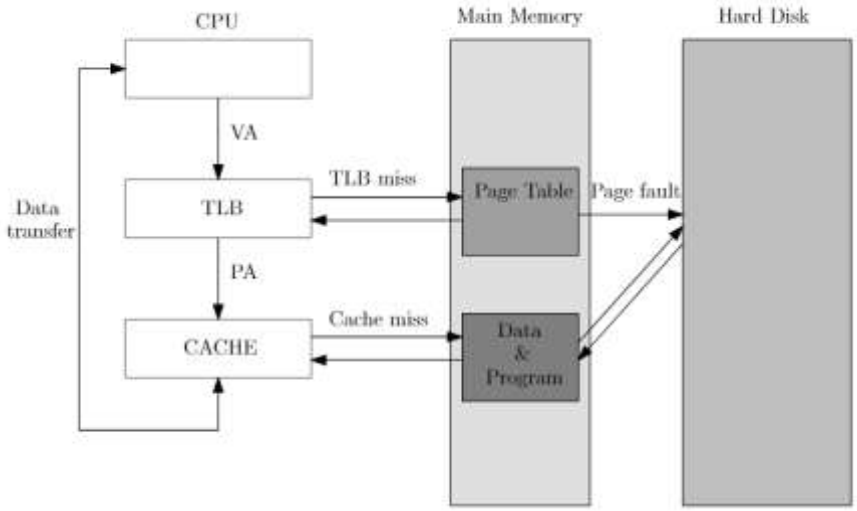


الشكل (3) البنية الهرمية للذواكر في نظام الحاسب[13].

يتبين ان كل مستوى من هرمية الذاكرة يتضمن مجموعة جزئية من المعلومات المخزنة في المستوى الأدنى

$$CPU \subset Cache \subset Main\ Memory \subset Disk$$

يبين الشكل (4) التعامل بين القرص الصلب والذاكرة الرئيسة في نظام الحاسب التي هي *RAM [11]*.



الشكل (4) العلاقة بين الذاكرة الرئيسية والقرص الصلب في نظام الحاسب[13].

تقييم أداء خوارزميات الكاش

تمّ اعتماد نسبة الإصابة *Hit Rate*، ونسبة الإصابة للبايت *Byte Hit Rate*، وزمن تحميل الصفحة لقياس أداء الخوارزميات المدروسة، تعرّف نسبة الإصابة على أنها النسبة المئوية من الطلبات التي يمكن تلبيتها من الكاش مقارنة مع عدد الطلبات الكلي، وهي تكتب بالمعادلة (4) *[12]*:

$$Hit\ Ratio = \frac{\sum_{i \in R} h_i}{\sum_{i \in R} f_i} \quad (4)$$

عندما تكون نسبة الاصابة مرتفعةً، فإنّ ذلك يدلّ على أنّنا استطعنا استرجاع نسبة عالية من الطلبات من التخزين المؤقّت، وهو عادةً النتيجة المرجوّة لمعظم مديري النظم.

كما تم اعتماد معدل الإصابة للبايت *Byte Hit Ratio* والذي يقيس حجم البيانات التي تم استعادتها من الكاش مقدرة بالبايت مقارنة مع حجم البيانات الكلي الذي تم استعادته *[12]*.

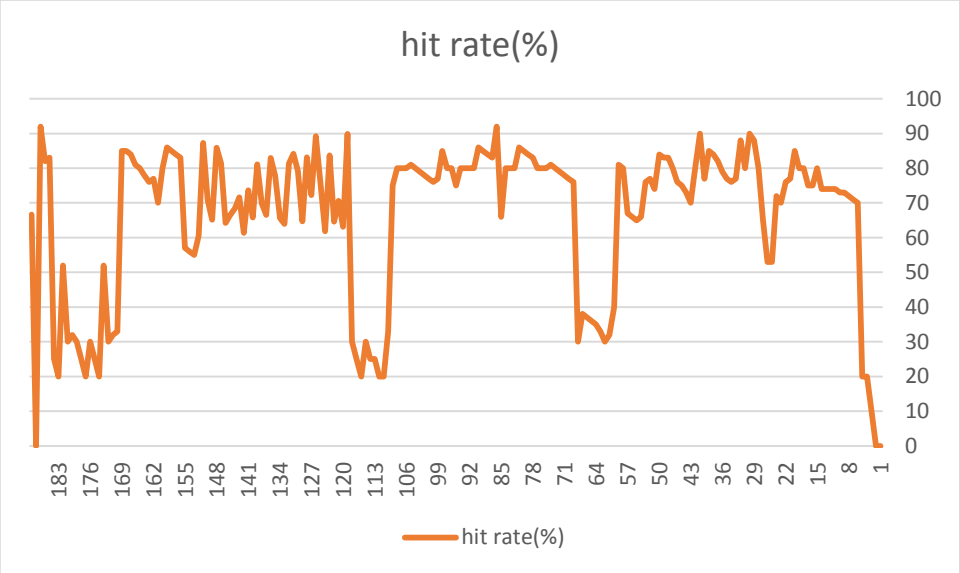
$$Byte\ Hit\ Ratio = \frac{\sum_{i \in R} S_i h_i}{\sum_{i \in R} S_i f_i} \quad (5)$$

حيث أن:

- R هي مجموعة الاغراض التي تم الوصول اليها.
- S_i هي قياس الغرض i
- f_i اجمالي عدد الطلبات للغرض i
- h_i اجمالي عدد الاصابات للغرض i

النتائج والمناقشة

تم قياس معدل الأصابة للخوارزميات المدروسة ورسم المخططات البيانية التي تعبر عن القيم كما هو مبين كمثال في الشكل (5) لخوارزمية *LRU*. الجدول (1) يبين القيم المقاسة كمتوسطات لمعدل الإصابة للخوارزميات المدروسة.

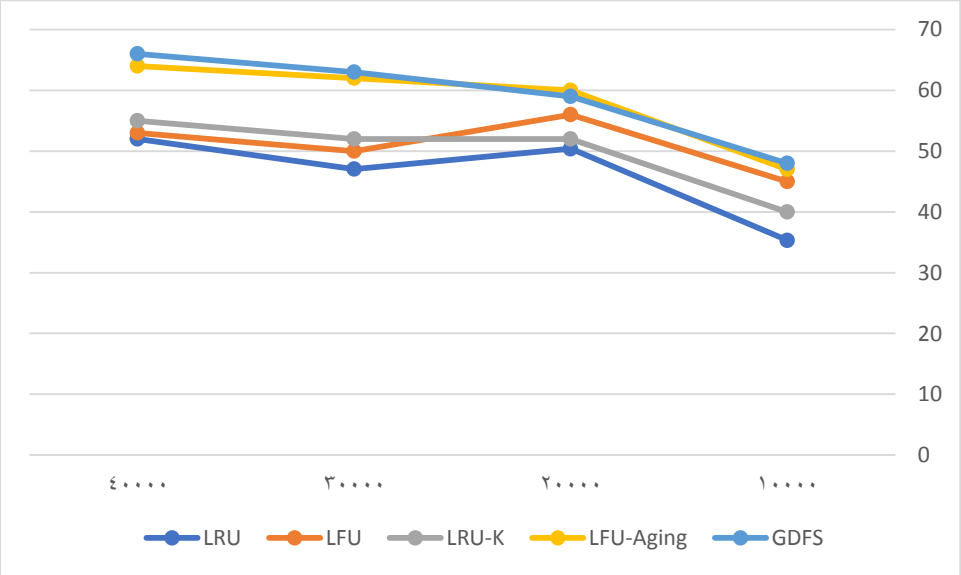


الشكل (5) العلاقة بين الذاكرة الرئيسية والقرص الصلب في نظام الحاسب لخوارزمية *LRU* من أجل عدد الطلبات.

الجدول(1) قيم متوسطات معدلات الإصابة للخوارزميات المدروسة، من أجل عدد طلبات مختلف.

متوسط معدل الاصابة %					Num. of requests
GDFS	LFU-Aging	LRU-K	LFU	LRU	
48	47	40	45	35.33333	10000
59	60	52	56	50.38542	20000
63	62	52	53	47.04167	30000
66	64	55	56	52	40000

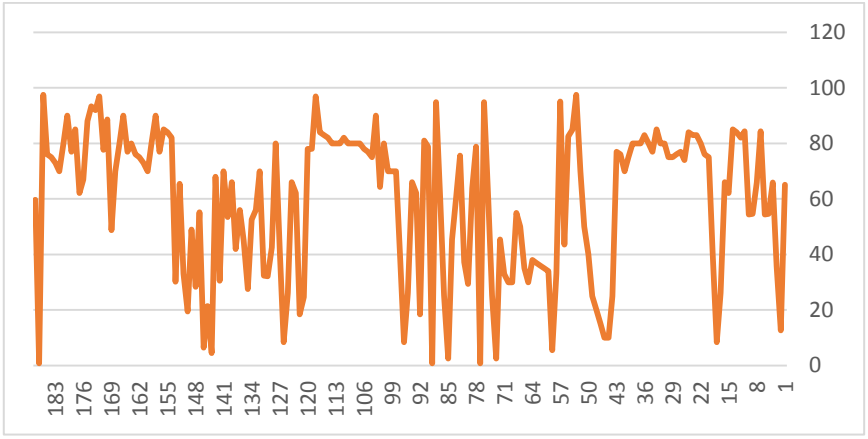
يبين الشكل (6) المخطط البياني للقيم المبينة في الجدول (1) ، ويمكن أن يستنتج من المخطط أن الخوارزمية وحيدة المعيار *LRU, LFU* حصلت على معدلات إصابة أدنى من النسخ المحسنة من الخوارزميات كما أن خوارزمية *GDFS* حققت أعلى معدل إصابة ولكن بنسبة ضئيلة عن الخوارزمية المحسنة *LFU-Aging*.



الشكل (6) العلاقة بين ذاكرة رام والقرص الصلب في نظام الحاسب عند تطبيق الخوارزمية المدروسة.

أما الشكل (7) يبين معدلات الإصابة لخوارزمية *LRU* عند تطبيق الخوارزمية على مخدم الويب، ونلاحظ التفاوت الكبير في معدلات الإصابة من أجل كل طلب مقارنةً مع الشكل (5) بسبب طبيعة الطلبات على صفحات الويب، والتي تختلف عنها في نظام الحاسب، وهي تتميز بما يلي:

- التفاوت في قياس الأغراض *sizes*.
- التفاوت في كلف جلب الأغراض *fetching costs*.
- التفاوت في البنية الهرمية لكاش الويب *caching hierarchy*.
- نموذج الوصول إلى البيانات *access patterns*، والذي لا يولد من قبل عدد محدود من الخدمات المبرمجة، وإنما يرتبط بشكل أساسي بسلوك إنساني متفاوت بشكل كبير.



الشكل (7) العلاقة بين الذاكرة الرئيسة والقرص الصلب في نظام الحاسب.

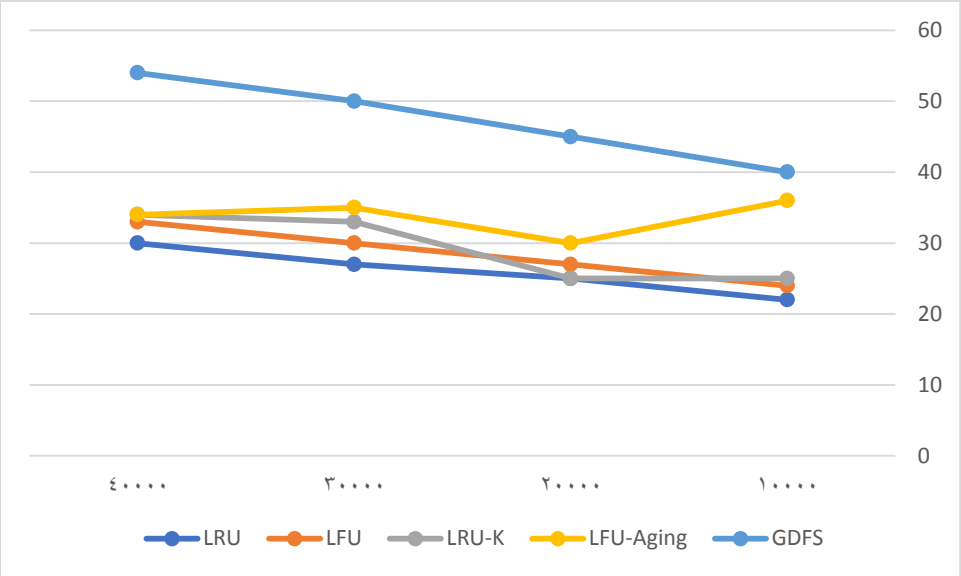
تم قياس متوسطات معدلات الإصابة للخوارزميات المدروسة في مخدم الويب، كما هو مبين في الجدول(2).

الجدول(2) قيم متوسطات معدلات الإصابة للخوارزميات المدروسة، من أجل عدد طلبات مختلف في مخدم الويب.

متوسط معدلات الإصابة %					Num. of requests
GDFS	LFU-Aging	LRU-K	LFU	LRU	
40	36	25	24	22	10000
45	30	25	27	25	20000
50	35	33	30	27	30000
54	34	34	33	30	40000

ويبين الشكل (8) معدلات الإصابة للخوارزميات المدروسة من أجل مخدم الويب، حيث يمكننا أن نرى أن الخوارزميات المحسنة أعطت أداء أفضل من الخوارزميات الأساسية، إلا أن معدل الإصابة للخوارزميات الأساسية في مخدم الويب كان أخفض من معدل الإصابة في نظام القرص الصلب، كما نلاحظ أن خوارزمية *GDFS* أعطت أداء أفضل من باقي الخوارزميات بشكل ملحوظ، حيث تساعد العوامل التي تم إدخالها إلى تابع الاستبدال للخوارزمية في تحسين تعامل الخوارزمية مع نموذج الطلب المتغاير من قبل المستخدمين.

تم قياس نسبة التحسين التي حصلت باستخدام الخوارزميات الأساسية والخوارزميات المحسنة، كما تم قياس نسبة التحسين بين خوارزمية *GDFS* والخوارزميات الأساسية كما هو مبين في الجدول (3).

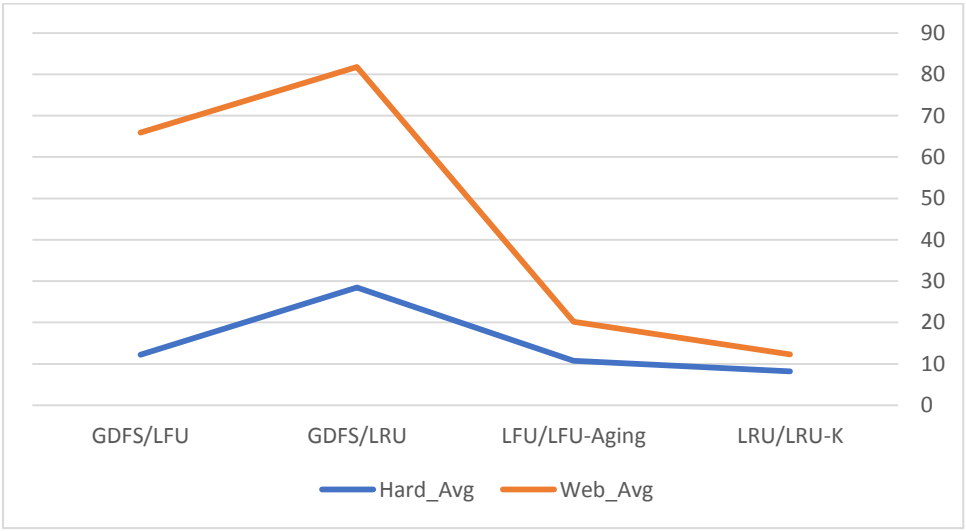


الشكل (8) العلاقة بين الذاكرة الرئيسة والقرص الصلب في نظام الحاسب.

الجدول(3) معدلات التحسين في نسب الإصابة للخوارزميات المدروسة.

معدل التحسين في نسبة الإصابة %				النتيجة
GDFS/LFU	GDFS/LRU	LFU/LFU-Aging	LRU/LRU-K	
12.18722	28.44833	10.71354	8.180385	Hard_Avg
65.90909	81.75084	20.20202	12.29798	Web_Avg

تم رسم معدلات التحسن في الخوارزميات في الشكل(9)، ويتبين من الشكل أن التحسن في الخوارزميات الأساسية باستخدام تطوير وحيد لمعيار تابع الاستبدال كان منخفضاً لكل من خوارزميات الكاش في القرص الصلب ومخدم الويب، كما يتبين أن التحسن باستخدام خوارزمية *GDFS* كان هو الأعلى وخصوصاً من أجل مخدم الويب، ذلك أن خوارزميات الكاش التقليدية مثل خوارزميتي *LRU* و*LFU* تظهر أداءً منخفضاً عند التعامل مع الصفحات ذات الحجوم الكبيرة. وفي هذه الأيام فإن أكثر المستندات أو الصفحات التي يتعامل بها الناس على شبكة الانترنت هي في الحقيقة ملفات الصوت والصورة (الفيديو) ، والتي تكون في الحقيقة كبيرة الحجم.



الشكل (9) العلاقة بين الذاكرة الرئيسة والقرص الصلب في نظام الحاسب.

الاستنتاجات والتوصيات:

الاستنتاجات

- تعطي خوارزميات الكاش وحيدة المعيار أداء أقل من نفس الخوارزمية عند إدخال معايير إضافية عليها.
- التفاوت في معدلات الإصابة للخوارزميات لدى تطبيقها على الويب أكبر من التفاوت لدى تطبيقها على القرص الصلب بسبب طبيعة الطلب المتعدد على الويب، وهو مايصعب على تابع الاستبدال للخوارزمية اتخاذ القرار الأفضل لاستخدام ذاكرة الكاش.
- تعطي خوارزميات الكاش في الويب أداء أخفض منه في القرص الصلب بسبب التفاوت المذكور وعليه تبرز الحاجة إلى تطوير خوارزمية كاش الويب بشكل دائم.
- التحسن المدخل على خوارزمية *GDFS* في مجال الويب كان أفضل منه في مجال القرص الصلب وعليه فإن هذه الخوارزمية من الأفضل استخدامها في مجال الويب كونها أكثر تعقيداً من الخوارزميات وحيدة المعيار ومن الصعب تحقيقها على مستوى القرص الصلب.

التوصيات:

- استخدام الخوارزميات وحيدة المعيار فقط من أجل سهولة تطبيق الخوارزميات.
- استخدام خوارزمية *GDFS* الفعلي في مجال الويب، حيث أعطت تحسين أفضل على الخوارزميات منها في القرص الصلب.
- في مجال القرص الصلب ينصح باستخدام الخوارزميات الأساسية أو إدخال معيار تطوير وحيد لها.

المراجع

[1]K,Arashi, Taha, R Ezaei-H, Achesu P Eyman , G Aderi Leila, Acta Medica, "Designing And Evaluating The Web-Based Information System Of Primary, Health Care In Accordance With The Electronic Health Records Of Iran", Mediterranea, 2016, 32: 2051

[2]Moruz, Gabriel; Negoescu, Andrei; Neumann, Christian; W, Volker;" Engineering Efficient Paging Algorithms";Goethe University Frankfurt am Main. Robert-Mayer-Str. 11-15, 60325 Frankfurt am Main, Germany.2015

[3]Saemundsson, Trausti ;"An experimental comparison of cache algorithms"; Research Methodology, Reykjavik University, September 30, 2013

[4] Jitendra Singh Kushwah, & Dr. Sitendra Tamrakar, "An extensive Review of Webs Caching Techniques to Reduce Cache Pollution", Department of Computer Science & Engineering, AISECT University, Bhopal, India mperial Journal of Interdisciplinary Research (IJIR) Vol-3, Issue-1, 2017 ISSN: 2454-1362

[5] J. Li, J. Wu, G. Dan, A. Arvidsson, M. Kihl, "Performance Analysis of Local Caching Replacement Policies for Internet Video Streaming Services", IEEE 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCom) 2014.

[6] Li, Jie; Arvidsson, Åke; Dán, György; Wu, Jinlong;."Performance Analysis of Local Caching Replacement Policies for Internet Video Streaming Services";Networking and Transmission Laboratory Acreo Swedish ICT AB, Kista, Sweden,2016.

[7] A. Radhika Sarma and R.Go vindarajan "An Efficient Web Cache Replacement Policy",Supercomputer Education and Research Center, Indian Institute of Science,

Bangalore, 560012,INDIA, In the Proc. of the 9th Intl. Symp. on High Performance Computing, (HiPC-03), Hyderabad, India, Dec. 2003.
[8] Dynamic Semantic LFU Policy with Victim tracer (DSLV): A Customizing Technique for Client Cache; Krishnan GeethaEmail authorN. Ammasai Gounden; Arabian Journal for Science and Engineering;February 2017, Volume 42, Issue 2, pp 725–737
[9]Shamsuddin, SitiMariyam; Ismail, Abdul Samad; Ali, Waleed ;" A Survey of Web Caching and Prefetching";Int. J. Advance. Soft Comput. Appl., Vol. 3, No. 1, March 2011 ,ISSN 2074-8523; Copyright © ICSRS Publication, 2013 www.i-csrs.org .
[10]Chauhan, Amit K. ; Chauhan, Vineet;" Exploring the Web Caching Method To Improve The Web Efficiency";International Journal of Computer Applications in Engineering Sciences VOL I, ISSUE IV , DECEMBER 2014 ISSN: 2231 -4946 Page 401.
[11] R. Nair, "Evolution of Memory Architecture," in *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1331-1345, Aug. 2015.doi: 10.1109/JPROC.2015.2435018
[12] Timothy Y. Chow, Terence P.Kelly, Daniel M. Reeves,"Estimating Cache Hit Rates from the Miss Sequence", Enterprise Systems and Software Laboratory,HP Laboratories Palo Alto ,HPL-2007-155 September 17, 2017.
[13] Steven Przybylski*,"Cache and Memory Hierarchy Design",1st Edition, imprint: Morgan Kufmann,Published:2014.*