مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (4) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (4) 2015

التخزين المؤقت لصفحات الويب الديناميكية

جمال أبو شاهين*

(تاريخ الإيداع 2 / 6 / 2015. قُبِل للنشر في 4/ 8 / 2015)

□ ملخّص □

نبحث في هذه الدراسة اقتراح واختبار خوارزمية أمثلية من حيث الأداء والسرعة تصلح للتخزين المؤقت لأغراض الويب ذات المحتوى الديناميكي من خلال دراسة الخوارزميات التقليدية المعروفة في مجال التخزين المؤقت Caching لصفحات الويب ودراسة مدى موائمتها للتخزين المؤقت لصفحات الويب ذات المحتوى الديناميكي لما لهذه الصفحات من أهمية وانتشار كبيرين في مواقع الويب وما تسببه من حمل كبير على مخدمات الويب بهدف الوصول إلى الخوارزمية المطلوبة والتي ستحقق أداء أمثلياً في التعامل مع هذه الصفحات.

الكلمات المفتاحية: التخزين المؤقت، مخدم الويب، صفحات الويب، صفحات الويب ذات المحتوى الديناميكي

* ماجستير – الجامعة الافتراضية السورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (4) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (4) 2015

Caching dynamic data for web applications

Jamal Abu Shaheen *

(Received 2 / 6 / 2015. Accepted 4 / 8 / 2015)

□ ABSTRACT □

We study in this research proposing and testing a new optimal algorithm in performance and speed is suitable for caching of web objects with dynamic content through studying the conventional classic algorithms that are common in caching web pages and studying how they can deal with caching web pages that have dynamic contents due to their great importance and spread in web sites and what they cause of overload on web servers to get a new algorithm that performs an optimal performance in dialing with this type of web pages.

Keywords: Caching, Web server, web pages, dynamic web pages

* **Master – Syrian Virtual University**

مقدمة:

إن شعبية الإنترنت ازدادت بشكل كبير على مر السنوات القليلة الماضية ، فقد ازداد عدد المستخدمين كما ازدادت كمية البيانات المتاحة والمنقولة عبر شبكة الانترنت. وعلى الرغم من أن عدد المخدمات في الإنترنت ازداد مع ارتفاع حجم عرض الحزمة فيما بينها ، إلا أن العديد من هذه المخدمات غير قادرة على التعامل مع أعداد ضخمة من طلبات البيانات التي يولدها المستخدمون وبذلك فإن العديد من الاتصالات تزدحم بكميات ضخمة من البيانات المنقولة.

إن العديد من تلك المشاكل يمكن أن تحدد و تعنون بـ "التخزين المؤقت لأغراض الويب" التخزين المؤقت هو عبارة عن تقنية مستخدمة بشكل واسع أينما يتم الوصول إلى البيانات. مثلاً باعتبار لدينا ذاكرة الكاش لمعالج ما و ذاكرة الكاش لنظام ملفات ما وغيرها ، فإنها تعمل من خلال تخزين البيانات من الأجهزة الأبطأ في ذاكرة أسرع ولكن أصغر نسبياً ، وبذلك فإن زمن الوصول للبيانات المخزنة بشكل مؤقت يقل بشكل ملحوظ. وهكذا إذا تم الاختيار المناسب للبيانات التي يتم تخزينها فإنه من الممكن الحصول على سرعة كبيرة جداً عند الوصول للبيانات.

عند استخدام هذا المفهوم في تخزين البيانات من الانترنت فإنه يشار إليه بـ التخزين المؤقت لأغراض الويب "web caching ".

في معظم الحالات تعمل ذواكر الكاش لأغراض الويب من خلال مقاطعة الطلبات التي يولدها مستخدمو الانترنت. وعندما يتم التخزين المؤقت للبيانات بشكل فعلي فإن ذاكرة الكاش تلبي طلب المستخدم ، وفقط في الحالة التي لا تكون فيها البيانات مخزنة في ذاكرة الكاش فإنه يتم طلبها من الموقع الأصلي وتخزينها فيها ومن ثم إرسالها إلى المستخدم[3].

هنالك العديد من فوائد التخزين المؤقت لأغراض الويب ، فعادة عندما يتم طلب البيانات فإنها تضطر إلى التنقل بين العديد من أجهزة الحاسب قبل أن تصل إلى أجهزة المستخدمين ، ولكن في حال وجود ذاكرة كاش متوضعة بالقرب من المستخدم فإن البيانات تحتاج إلى العبور في هذا المسار مرة واحدة فقط ويتم خدمة كل الطلبات اللاحقة إليها من قبل ذاكرة الكاش دون اللجوء إلى الموقع الأصلي مرة أخرى. هذا له العديد من التأثيرات الجانبية ، بدايةً فإن حمل المخدم يوزع على العديد من ذواكر الكاش مما يؤدي إلى تقليل زمن الاستجابة لطلب المستخدم. كما أنه يقلل استخدام عرض الحزمة بما أن البيانات لا تحتاج لأن يتم إرسالها في كل مرة من المخدم الأصلي إلى ذاكرة الكاش[16].

وفي حال وجود ذاكرة الكاش بالقرب من المستخدم فإن زمن نقل المعطيات يقل أيضاً. وبالتالي فإن ذاكرة الكاش تساعد في تقليل استخدام عرض الحزمة، زمن الحساب ، زمن النقل ، حمل المخدم والتأخير خاصةً عندما يتم إدراكها من قبل المستخدم،

على الرغم من وجود العديد من نقاط التشابه بين التخزين المؤقت التقليدي والتخزين المؤقت لأغراض الويب ، فإن الاختلافات بينها أكثر باعتبار 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 و صفحة 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 و حمل المخدم.

أهمية البحث وأهدافه:

هدف البحث:

إن عدم تعامل الكاش بشكل جيد مع ما يسمى صفحات الويب ذات المحتوى الديناميكية أو اختصاراً صفحات الويب الديناميكية أثار قلق الكثير من الناس المختصين في مجال الويب ونشر المحتوى. حيث ان هذه الصفحات تعتبر ديناميكية لأن محتوى هذه الصفحات قد يختلف عند كل طلب لهذه الصفحات. المعلومات الحساسة للزمن، مثل أسعار السلع أو نشرات الأحوال الجوية تندرج منطقياً في فئة الصفحات الديناميكية. وكذلك الصفحات المخصصة للمستخدم والتي تتضمن اسمه والتي تسمى الملف الشخصي بروفايل Profile أو الاعلانات المبوبة تندرج أيضاً ضمن فئة الصفحات الديناميكية.

تاريخياً كان التعامل مع الصفحات الديناميكية في مجال التخزين المؤقت والكاش صعباً جداً ولا يخلو من المشاكل لأن بعض البرمجيات والخوارزميات المقترنة بالكاش لها استرتيجيات كاش قاسية نسبياً.

وخلال السنوات السابقة تطرق الكثير من الباحثين والدارسين إلى موضوع خوارزميات واستراتيجيات الاستبدال والكاش في مجال التخزين المؤقت لصفحات الويب وهذه الاستراتيجيات المختلفة جميعاً تتميز في أدائها من خلال ثلاثة نقاط أساسية وهي:

1 - عدد الأغراض أو الصفحات الموجودة في الكاش (hit)

2 - حركة التبادل على الشبكة traffic والتي تم تخفيف حجمها من خلال استدعاء الأغراض المعلمة ذات الإشارات المرجعية في الكاش

3 -زمن الاستجابة الذي تم توفيره من خلال هذه الاستراتيجيات.

تتألف صفحة الويب بشكل عام من مجموعة من المكونات أو الأجزاء تدعى Fragments وفي هذا البحث سنقدم طريقة جديدة للتخزين المؤقت لأجزاء الصفحات الديناميكية من اجل تحسين أداء وفعالية مخدمات الويب وتطبيقاتها.

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 اجل 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 تقيم الملفات على أساس الزمن اللازم لتحميلها، ولدينا ملفين للتخزين يختلفان بشكل ملحوظ في الحجم ، فإن الملف الأكبر سيكون له مواءمة أكبر من الملف الصغير. وبالتالي عندما يكون الاسترداد ضروري فإن الملف الصغير سيسترد محرراً قدر صغير من الذاكرة. وفي حال كون قيم الملفات متساوية بشكل قريب فإنه من المفيد أن يتم استرداد الملف الأكبر.

حيث من الممكن إعادة تخزينه بنفس سرعة تخزين الملف صغير ولكنه يحرر في حال الاسترداد قدر كبير من ذاكرة الكاش ، مما يحسن الأداء الكلي بشكل ملحوظ. اعتماداً على تابع التقدير فإنه من الممكن أن يتم تحديث قيم تقدير كل الملفات عند طلب ملف ما أو تخزينه بشكل مؤقت. وبما أن ذاكرة كاش الوكيل تخدم عدد كبير من الطلبات

وتخزن عدد ضخم من الملفات فإن هذا قد يكون مكلف جداً حسابياً. أيضاً عندما يتم استرداد ملف ما فإن كل الملفات يجب أن تقارن ببعضها البعض من أجل تحديد الملف الذي يجب استرداده[7].

بسبب الحمل الكبير على مخدم ذاكرة كاش الوكيل فإن هذا يعتبر مشكلة أخرى لدى منفذي ذاكرة كاش الوكيل. من أجل تحديد وعنونة المشاكل السابقة تم تشكيل خوارزميات خاصة من أجل التخزين المؤقت لأغراض الويب.

طرائق البحث و مواده:

الدراسات السابقة:

تم الاطلاع على عدد من الدراسات في مجال خوارزميات التخزين المؤقت الكاش في مجالات الذاكرة والويب وصفحات الويب الديناميكية ونعرض فيما يلي لمحة عن بعض منها.

الدراسة Differentiated Web Caching A Differentiated Memory Allocation Model on Proxies للباحث Zheng [15]

تتحدث هذه الدراسة عن خوارزميات الكاش المعروفة وأهميتها في مجال صفحات الويب وقد قدمت هذه الدراسة اسهامين كبيرين، حيث تمت دراسة حمل العمل على مخدم التخزين المؤقت وتمت مقارنة هذا الحمل في حالة خوارزميات الكاش التقليدية مع الخوارزمية المقترحة من قبل الدراسة والمطورة عن الخوارزمية LRU-K والتي تأخذ بالاعتبار معلومات التكرار التقريبية بالإضافة إلى معلومات الحداثة الزمنية لكل وصول.

الاسهام الثاني كان اقتراح هيكلية بسيطة وفعالة تعتمد الخوارزمية المقترحة من قبل الدارسين والتي تم تسميتها webLRU-2 لتحقيق الهدف المرجو من الخوارزمية من اجل مختلف انواع طلبات الويب.

من خلال محاكاة عمل الخوارزمية في وضع افتراضي وفي الوضع الحقيقي تمت مقارنة خوارزمية webLRU-2 مع الخواررزميات المعروفة وهي LRU وLRU-K و LFU.

ومن الدراسات المهمة أيضاً الدراسة:

New Web Cache Replacement Algorithm للباحثين Bhattacharjee, Debnath [16]

تقدم هذه الدراسة خوارزمية استبدال عشوائية جديدة حيث تقوم باستخدام صناديق خزن Bins والتي يمكن اعتبارها تقسيمات منطقية حيث تقوم الخوارزمية باختيار أحد هذه الصناديق عشوائياً وبعد البحث عن صفحة قديمة بشكل عشوائي في الصندوق المختار يتم استبدال الصفحة القديمة. وفي حالة عدم وجودها يتم اضافة الصفحة الجديدة. وإلا يتم اختيار صندوق آخر بشكل عشوائي أيضاً ويستمر بنفس الطريقة.

إن هذه الخوارزمية قابلة للتحقيق بسهولة دون الحاجة لاستخدام أية بنية معطيات معقدة، كما أنها تقلص زمن الاستجابة بشكل فعال وهي فعالة جداً في مجال التخزين المؤقت لصفحات الويب.

خوارزمية المواءمة المقترحة Suggested Algorithm

من خلال ما تقدم تبرز الحاجة إلى خوارزمية جديدة وسياق عمل مناسب لإنجاز عملية التخزين المؤقت للأجزاء الديناميكية لصفحات الويب المولدة ديناميكياً.

الأهداف المطلوبة من الخوارزمية:

إن الهدف هو إيجاد حل للتخزين المؤقت للأجزاء الديناميكية لصفحات الويب المولدة ديناميكياً، ولتحقيق هذا الهدف يجب تحقيق المتطلبات التالية:

- **التخزين المؤقت لأقسام الصفحة:** تخزين اقسام الصفحات المولدة مسبقاً لتجنب إعادة توليدها مجدداً لتجنب التكلفة العالية لهذه العملية وهذا يوفر العديد من المصادر مثل الوصول إلى قاعدة البيانات وإجراء الحسابات المكلفة.
- **التوزيع:** يجب أن تكون الأجزاء Fragments في بيئة موزعة وتبقى محدثة بشكل دائم
- **السرية:** يجب الحفاظ على السرية طالما يتم استخدام هذه الأجزاء.

بنية وفوائد الخوارزمية:

من أجل حل المشكلة نقترح إنشاء هيكلية تخزين مؤقت موزعة من أجل تطبيقات الويب. إن الفائدة المرجوة من التخزين المؤقت تكمن في إعادة استخدام الموارد بدلاً من إعادة طلبها مرة أخرى وهذا يتم من خلال الاحتفاظ بهذه الموارد في الذاكرة بعد استخدامها.

إن جزء الصفحة هو عبارة عن قسم من الكود المكون للصفحة. وكل جزء من الصفحة يمكن تخزينه كملف ASCII يحتوي على الكود المكون (والذي قد يكون HTML).

تملك أجزاء الصفحة معلومات توصيفية مثل تاريخ الانشاء وعدد مرات الزيارة والتي يمكن استخدامها من أجل المزامنة أو الاستبعاد من الكاش.

لا يقوم الزبائن Clients بالوصول مباشرةً إلى مخازن البيانات. بل يحصلون على المعلومات المخزنة في المخازن من خلال واجهة تخزين مؤقت.

تتألف البنية المقترحة من مكونين أساسيين هما:

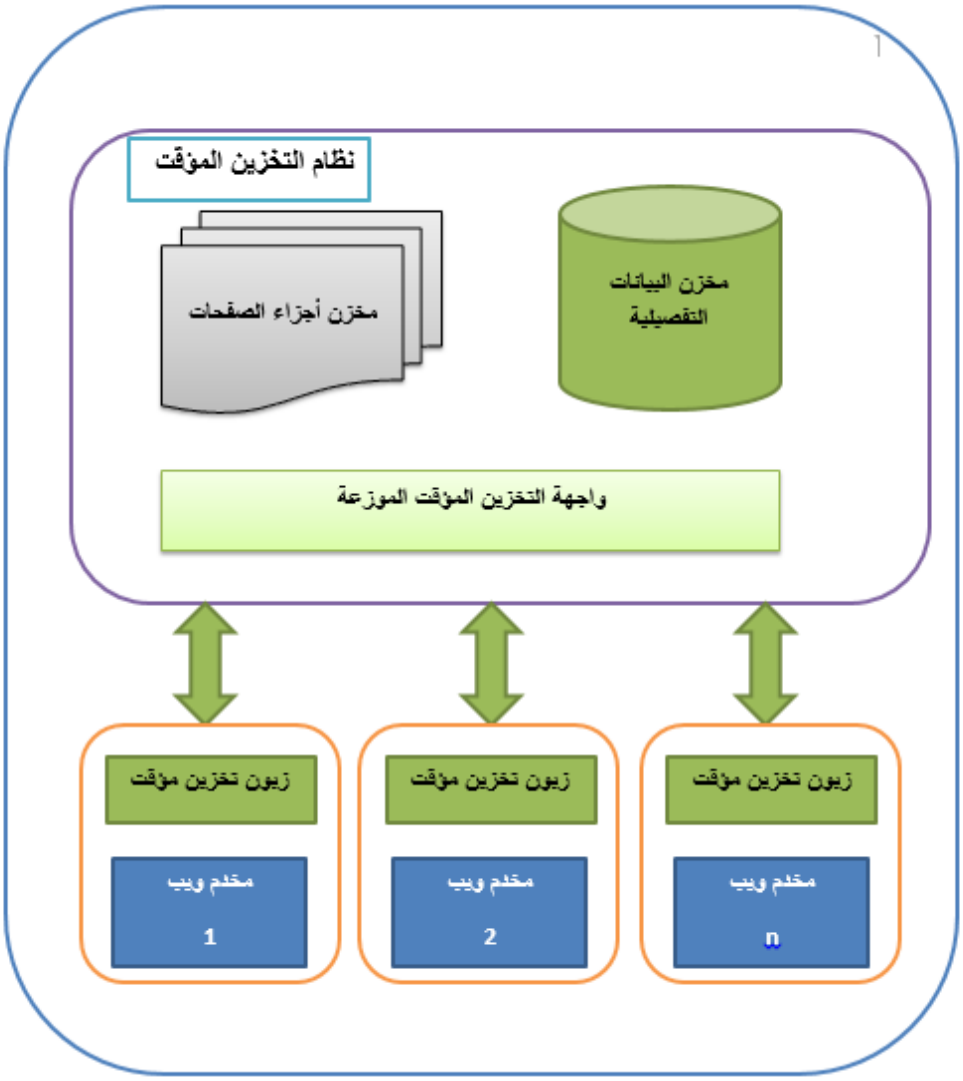
- **مخزن البيانات السلبية:** ويحتوي على أجزاء الصفحة Fragments وأيضاً على معلومات هذه الأجزاء Metadata حيث تخزن جميعها في مخدم التخزين المؤقت ولا يمكن الوصول اليها بشكل مباشر من قبل المستخدمين العاديين للنظام.
- **واجهة التخزين المؤقت:** تسمح هذه الواجهة بالوصول إلى محتويات الكاش أو المخزن المؤقت من خلال استخدام الميتا داتا وذلك بغرض إجراء أغراض المزامنة أو الاستبعاد.

أما الزبائن فهم عبارة عن تطبيقات ويب أو مخدمات ويب وهذه التطبيقات تستخدم الكاش من خلال تطبيق زبون Caching Client والذي يقوم بدوره بالاتصال مع مخدم التخزين المؤقت Caching Server من خلال واجهة التخزين المؤقت ذات البيئة الموزعة.

ليس هنالك من تبعية بين مخدمات الويب الحقيقية والتي تحوي على مواقع الويب وبين مخازن الكاش. حيث أن جميع المحادثات والاتصالات تتم من خلال واجهات ذات بنية موزعة معرفة مسبقاً. هذه الواجهات يمكن تعريفها كخدمات Services.

بنية النظام:

يظهر الشكل التالي (الشكل 1) المخطط الهيكلي للحل المقترح:



الشكل(1) المخطط الهيكلي لبنية التخزين المؤقت المقترحة

يشكل مخدم التخزين المؤقت Caching Server المكون الأساسي للنظام المقترح. وهو يتألف من المكونات التالية:

- **مخزن أجزاء الصفحات:** هو عبارة عن مخزن من نوع ملفات النظام يقوم بتخزين الملفات التي تحتوي على أقسام الصفحة الفعلية (وهي عبارة عن ملفات ASCII تحتوي على أجزاء من كود الـ Fragment والذي قد يكون HTML)
- **مخزن البيانات التفصيلية Metadata:** وهو عبارة عن قاعدة بيانات تحتوي على الميتاداتا المقترنة بجزء ديناميكي من الصفحة. وهذه الميتاداتا تحتوي على معلومات مثل تاريخ آخر تحديث للجزء أو عدد الزيارات hit وهذه المعلومات مهمة للغاية من أجل إجراء عمليات الموازنة والاستبعاد لهذه الأجزاء.

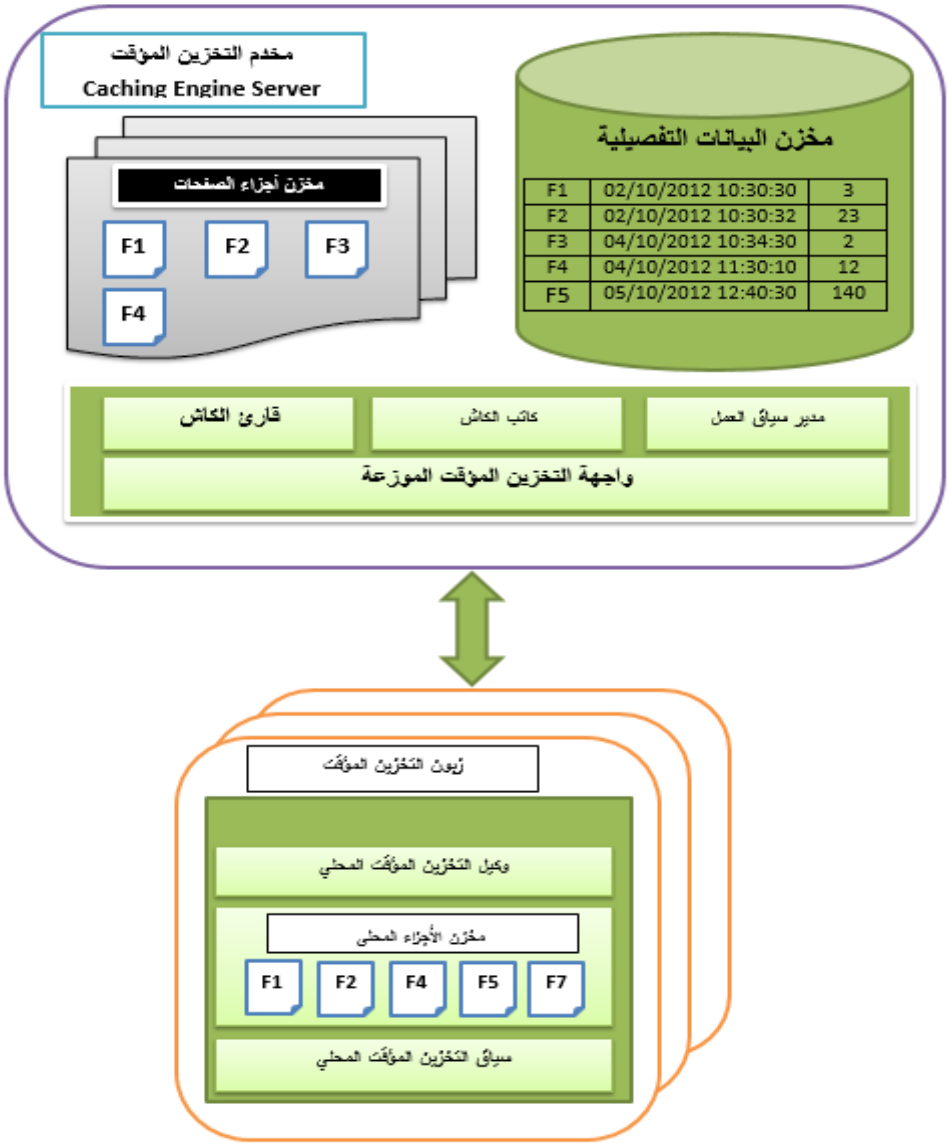
- **واجهة التخزين المؤقت الموزعة:** تقوم هذه الواجهة بالكشف عن محتويات الكاش في وحدات منفصلة.

إن زبائن الكاش هم عبارة عن مخدمات ويب موزعة (كما يظهر في الشكل السابق نجد مجموعة من مخدمات الويب تعمل سويةً مع مخدم التخزين المؤقت). يمكن ان توضع مخدمات الويب هذه في شبكات موزعة مهما اختلفت جغرافيتها. ولكن الشرط الأساسي هو ان تستطيع الاتصال مع مخدم التخزين المؤقت Caching Server. كل مخدم يحتوي على زبون تخزين مؤقت Caching Client وهو عبارة عن التطبيق الذي يقوم بالتفاعل مع واجهة التخزين المؤقت الموزعة.

البنية التفصيلية للنظام:

لقد قمنا بعرض لمحة عامة عن بنية النظام المقترح. سنقوم الآن بعرض البنية التفصيلية للنظام من الداخل وشرح كيفية عمله بشكل تفصيلي.

الشكل التالي (الشكل 2) يظهر بنية اكثر تفصيلاً للنظام:



الشكل(2) المخطط التفصيلي لنظام التخزين المؤقت المقترح

سنقوم في الفقرات الجزئية اللاحقة بشرح كل جزئية من المخطط السابق بشكل تفصيلي متعرضين إلى كافة مكونات النظام:

مخدم التخزين المؤقت Caching Server:

إن مخدم التخزين المؤقت هو المكون المسؤول تخزين ومعالجة أجزاء الصفحات الديناميكية. إنه ليس مخزناً سلبيا بل يحوي أيضا على واجهة التخزين المؤقت الموزعة والتي تسمح بمعالجة ومشاركة محتويات المخزن المؤقت من خلال تشارك مجموعة من المكونات الداخلية .

يشمل مخدم التخزين المؤقت مجموعة من المكونات عالية التخصص والتي تم تحديد مسؤولياتها ووظائفها بدقة تامة و حرص كبيرين.

مخزن أجزاء الصفحة Page Fragment Store:

إن مخزن أجزاء الصفحة هو عبارة عن مخزن سلبي أي أن مهمته تنحصر في عملية التخزين فقط دون أي معالجة للبيانات الموجودة داخله. يحتوي هذا المخزن على أجزاء الصفحة الفعلية وهو عبارة عن مخزن من نوع ملفات النظام بحيث يستطيع تخزين أجزاء الصفحات على شكل ملفات ASCII.

لا يمكن الوصول إلى محتويات المخزن بشكل مباشر من خارج المخدم (حيث لا يستطيع أي مكون النفاذ اليه باستثناء الواجهة).

يتم كتابة هذه الأجزاء من خلال مكون يدعى Cache Writer أما القراءة فتتم من خلال مكون آخر يدعى Cache Reader والذين سنتطرق لهما لاحقاً. كل جزء Fragment له معلومات تفصيلية Metadata مرتبطة به. وهذه المعلومات التفصيلية Metadata تخزن في مخزن البيانات التفصيلية للكاش.

مخزن البيانات التفصيلية Metadata للكاش:

إن مخزن البيانات التفصيلية هو مخزن سلبي أيضا يحتوي على المعلومات التفصيلية أو التوصيفية Metadata المرتبطة بجزء الصفحة الذي تتم عملية الكاش له.

تمثل هذه المعلومات التوصيفية المفتاح الأساسي لتحقيق وظائف الكاش الأساسية في الاستعادة والاستبعاد. هذه المعلومات التوصيفية المرتبطة بجزء الصفحة تحتوي على معلومات مثل تاريخ آخر تحقيق لهذا المكون ونقصد به تاريخ آخر ظهور لهذا المكون على الصفحة كما تحتوي على عدد الضربات Hit والذي يمثل في الحقيقة عدد مرات زيارة الجزء.

يستطيع زبون الكاش Caching Client من خلال هذه المعلومات تحديد فيما إذا كانت النسخة الموجودة لديه عن الجزء الديناميكي للصفحة (هذه النسخة تكون موجودة في مخزن أجزاء الصفحة المحلي الموجود عند الزبون) قديمة. كما أننا نستطيع استخدام هذه المعلومات لتحديد فيما إذا كان المكون Fragment قابلاً للاستبعاد.

لا يمكن تعديل المعلومات التوصيفية metadata بشكل مباشر من خارج مخدم الكاش Caching Server. ولكن يمكن تعديلها من خلال واجهة التخزين المؤقت الموزعة عندما يتفاعل الزبائن مع الكاش (عندما يقوم Cache Reader بقراءة جزء Fragment أو عندما يقوم Cache Writer بكتابة جزء) أو بشكل داخلي عندما يتم طرد جزء من الكاش.

واجهة التخزين المؤقت الموزعة:

تسمح واجهة التخزين المؤقت الموزعة باستخدام ومعالجة محتويات الكاش في مخازن مخدم التخزين المؤقت Caching Server. وهي تمثل المكون الفعّال الذي يقدم إمكانية النفاذ والوصول إلى العناصر السلبية وهي المخازن الموجودة في مخدم الكاش.

تتألف الواجهة بدورها من مجموعة من المكونات الداخلية وهي:

- **قارئ الكاش Cache Reader**: يقوم هذا المكون بقراءة جزء الصفحة من الذاكرة المؤقتة ويقوم بتحديث البيانات التوصيفية المرتبطة Metadata مثل عدد مرات الزيارة Hit.
- **كاتب الكاش Cache Writer**: يقوم هذا المكون بتخزين جزء الصفحة في ذاكرة الكاش وتحديث المعلومات التوصيفية Metadata المرتبطة بها (على سبيل المثال إذا كان الجزء غير موجود في ذاكرة الكاش من قبل يقوم الكاتب بإنشاء الميتاداتا المرتبطة بهذا الجزء بينما إذا كان الجزء موجود مسبقاً هنا يقوم الكاتب بعملية تعديل هذه المعلومات التوصيفية).
- **مدير سياق العمل Context manager**: يقوم بإنشاء نسخ من Caching Context ويعيدها إلى الزبائن. بالإضافة إلى ذلك يقوم بأخذ نسخ من سياق الزبائن ومن ثم تتم عملية المقارنة لتحديد أي الأجزاء هي منتهية الصلاحية وتحتاج للمزامنة عند الزبون.
- **الواجهة الموزعة Distributed interface**: تزودنا هذه الواجهة بطريقة وصول سهلة ومرنة إلى كل من قارئ الكاش، كاتب الكاش، بالإضافة إلى مدير السياق. وهذه الوصول يتم من خلال أغراض موزعة او من خلال واجهة معينة تعتمد على خدمة ويب.

زبون التخزين المؤقت Caching Client:

يمثل زبون الكاش المكون الذي يسمح لتطبيقات الزبائن باستخدام مخدم التخزين المؤقت وهو صلة الوصل بينهم. يتألف الزبون من ثلاثة مكونات أساسية: وكيل التخزين المؤقت المحلي، مخزن الأجزاء المحلي وسياق التخزين المؤقت المحلي.

وكيل التخزين المؤقت المحلي Local Caching Proxy:

وهو المكون المسؤول عن الاتصال والتفاعل مع المخدم من خلال واجهة المخدم الموزعة مقدماً بذلك واجهة محلية للاتصال مع المخدم. انها تستطيع إدخال البيانات إلى مخدم التخزين المؤقت Caching Server كما تستطيع سحب البيانات من المخدم.

مخزن الأجزاء المحلي Local Fragments Storage:

يحتوي هذا المخزن على أجزاء الصفحة المحملة من المخدم. وهي الأجزاء التي يتم استخدامها من قبل التطبيقات.

إن مخزن الأجزاء المحلي هو عبارة عن مخزن مبني على نظام الملفات وهو شبيه بالمخزن الموجود على المخدم (مخزن أجزاء الصفحات).

يتم تحميل هذه الأجزاء Fragments إلى هذا المخزن باستخدام وكيل التخزين المؤقت المحلي كما ذكرنا سابقاً. عندما يستخدم تطبيق معين هذا الجزء يتم استدعاؤه من هذا المخزن.

سياق التخزين المؤقت المحلي Local Caching Context:

يحتوي على المعلومات التفصيلية Metadata والتي تقوم بتوصيف أجزاء الصفحة. يتم استخدام هذه المعلومات Metadata في خوارزمية المزامنة من قبل وكيل التخزين المؤقت المحلي.

إن المعلومات المخزنة في وكيل التخزين المؤقت المحلي هي عبارة عن دمج بين المعلومات التوصيفية Metadata الموجودة في مخزن الـ Metadata الموجودة على المخدم والمتعلقة بجزء الصفحة وبين المعلومات Metadata الموجودة محلياً حول نفس الجزء (تاريخ الانشاء المحلي، عدد مرات الزيارة المحلي).

خوارزمية المزامنة الطارئة Emergent Synchronization Algorithm:

إن فكرة المزامنة في نظامنا المطروح تقوم على الفكرة التالية:

ليس هنالك من مدير تخزين مؤقت مركزي أو مكون تزامن يقوم بالتاكد من أن جميع الزبائن (وهم في حالتنا هنا مخدمات الويب) هي متزامنة مع ذاكرة التخزين المؤقت الكاش.

بدلاً من ذلك فإن المزامنة هنا تنشأ من خلال المواءمة بين مجموعة من القواعد البسيطة يمكننا سردها كما يلي:

1 - تقوم صفحة ويب بطلب جزء Fragment

2 - يقوم وكيل التخزين المؤقت المحلي بالتحقق من كون الصفحة موجودة في مخزن الأجزاء المحلي

a. إذا لم تكن الصفحة موجودة يقوم الوكيل بالحصول عليها من واجهة التخزين المؤقت الموزعة الموجودة على المخدم ثم يقوم بتخزينها في مخزن الأجزاء المحلي ويقوم بعدها بتحديث سياق التخزين المؤقت المحلي

b. إذا كانت الصفحة موجودة يقوم بالتحقق فيما إذا كان تاريخ الملف هو أقدم من آخر تحديث للجزء في المخدم (ويتم هذا من خلال الواجهة الموزعة). إذا كان الملف أقدم يتم تحميله مجدداً من المخدم ويتم استبدال الملف المنتهي الصلاحية في مخزن الأجزاء المحلي ويتم تعديل البيانات التوصيفية Metadata المرتبطة به في سياق التخزين المؤقت.

3 - يقوم الوكيل المحلي بالحصول على جزء الصفحة من الكاش المحلي ويقوم باستخدامه

الشكل (3) Pseudo الخاص بخوارزمية المزامنة المقترحة

نلاحظ أن مخدم التخزين المؤقت Caching Server يقوم فقط بتزويد المعلومات ولكن لا يتدخل في كيفية استخدام هذه المعلومات.

نلاحظ أيضاً من خلال الخوارزمية السابقة أنه لا يوجد أي مكون من المكونات السابقة يعلم شيء عن المزامنة. حيث أن جميع هذه المكونات تقوم بتنفيذ احداث بسيطة.

النتيجة المذهلة هنا ان جميع الزبائن (مخدمات الويب) أصبحت متزامنة أوتوماتيكياً دون الحاجة إلى مراقب عام لها.

إن هذه العملية ملائمة جداً في شبكة الويب حيث أننا نستطيع إضافة مخدمات جديدة أو إزالة أخرى دون الحاجة لإجراء أي تغيير على مخدم التخزين المؤقت Caching Server. في أي وقت يتم فيه إضافة زبون جديد فإنه سيقوم بتنفيذ هذه الخوارزمية وسوف يصبح متزامناً.

ومن النتائج الإيجابية أيضاً أن عملية التزامن ليست عملية إجبارية وإنما يتم تنفيذها فقط عند الحاجة إليها مما يقلل من جهود إدارة المخدم.

الفوائد المحققة من النظام المقترح:

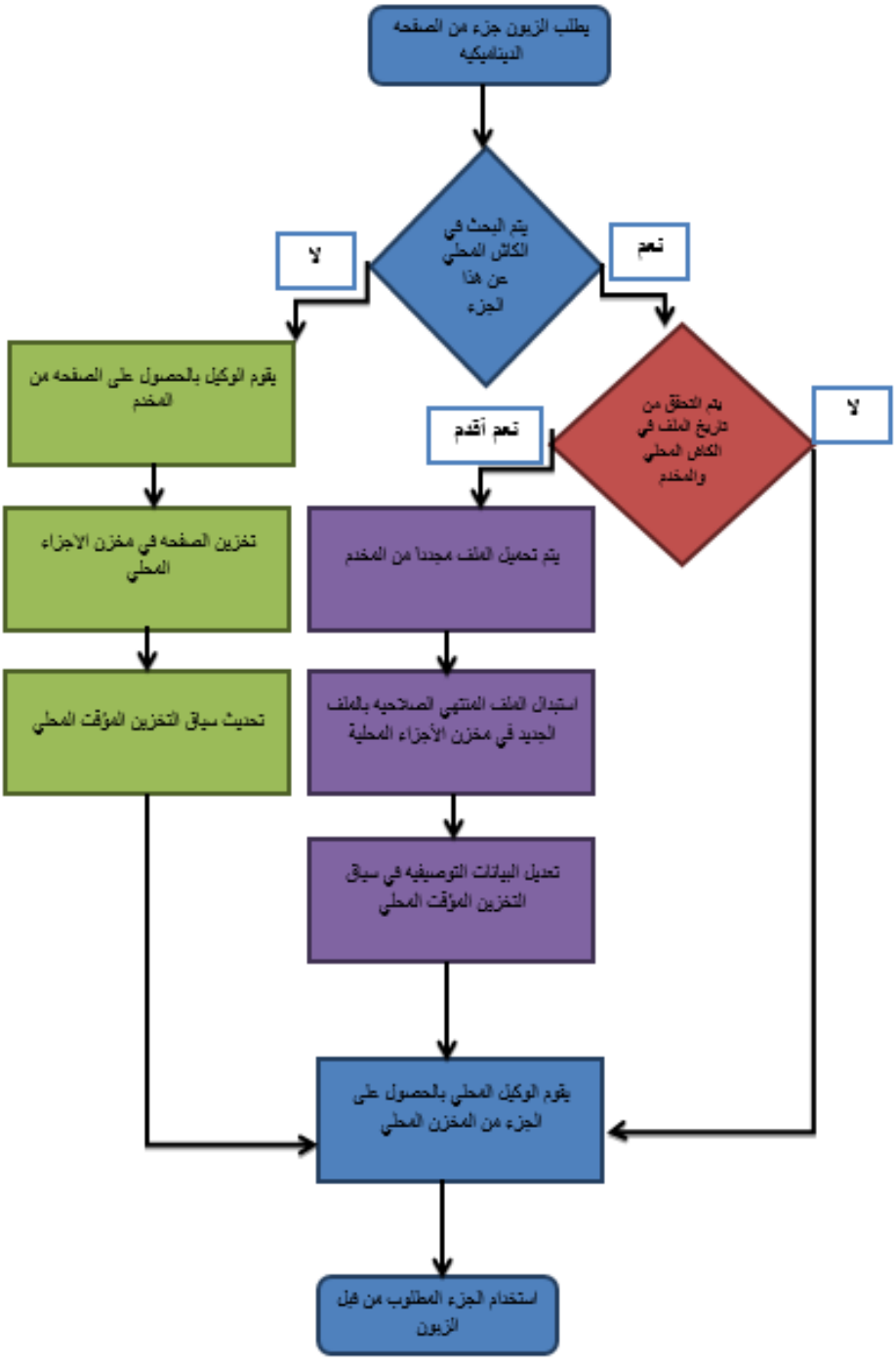
يقدم نظامنا المقترح تخزيناً مؤقتاً موزعاً لأجزاء الصفحات الديناميكية ليحل بذلك مشكلة التعامل مع صفحات الويب ذات المحتوى الديناميكي المتغير.

ويمكننا سرد هذه الفوائد كما يلي:

- التخزين المؤقت لأجزاء الصفحات: يتم تخزين أجزاء الصفحة في مخزن أجزاء الصفحات ويمكن تحميلها واستخدامها من قبل أي زبون من خلال واجهات مترابطة فيما بينها.
- الشفافية: يستطيع المدير استعراض مخزن أجزاء الصفحة في المخدم أو مخزن أجزاء الصفحة المحلي عند الزبون في أي وقت يريده. كما يمكنه الاستعلام عن كون المعلومات التفصيلية Metadata منتهية الصلاحية او لا لجزء Fragment على المخدم ومقارنته بالجزء الموجود على المخزن المحلي عند الزبون. كما يستطيع المدير استعراض مخزن الأجزاء المحلي عند أي زبون ويفحص محتويات جزء الصفحة المستخدم. وهذا يسمح للمدير بتحديد حالة ذاكرة التخزين المؤقت في أي وقت يريده.
- التوزيع: يقوم الزبائن بعملية المزامنة بشكل طبيعي وليس طارئ. وبالتالي لا ضرورة لا لآلية مزامنة مركزية ولا لطرق أخرى معقدة. مما يسمح بإضافة مخدم جديد ليستخدم الكاش دون الحاجة إلى تسجيله أو اعلام مخدم التخزين المؤقت بوجوده.
- التشخيص **Personalization**: حيث يمكن إظهار وعرض أجزاء الصفحة للمستخدم وفقاً لدوره ووظيفته.
- الفعالية: من أجل زيادة الفعالية تم استخدام التخزين المؤقت للمعلومات metadata وفحص التزامن.
- التوفر Availability: طالما أن جزء الصفحة المخزن في الكاش متزامن مع الزبون فإن المصدر الأصلي للجزء لا حاجة له حتى يتم مسح الكاش أو حتى تنتهي صلاحية هذا الجزء ويتم تحميل جزء جديد بإصدار أحدث.
- الاستخدام الأفضل للموارد: إن تحقيق هذه الخوارزمية يمكننا من الاستخدام الأفضل للموارد: ان المعلومات تجلب فقط عندما تكون متوفرة وملائمة حيث تخزن بعدها في بنية ذاكرة سريعة الوصول مما يزودنا بإمكانية وصول فعالة لها. كذلك عملية بناء الصفحات الديناميكية المعقدة تكون محضرة مسبقاً وبالتالي يتم تقليل استهلاك الموارد من جهة المخدم. بالإضافة لذلك يتم تخفيض تحميل الأغراض مما يقلل من استهلاك موارد الذاكرة وخصوصاً في البيئات التي تعتمد نظم إدارة الذواكر أوتوماتيكياً مثل J2EE و NET..
- البساطة: إن بنية نظام التخزين المؤقت لأجزاء الكاش سهل الاستخدام وإن السيناريوهات الرئيسية فيه (قراءة من الكاش و كتابة إلى الكاش) مصممة لتعمل بشكل طبيعي وبسيط وليس بشكل إجباري وقسري.

المخطط التدفقي للخوارزمية:

يبين الشكل (4) المخطط التدفقي للخوارزمية المقترحة:



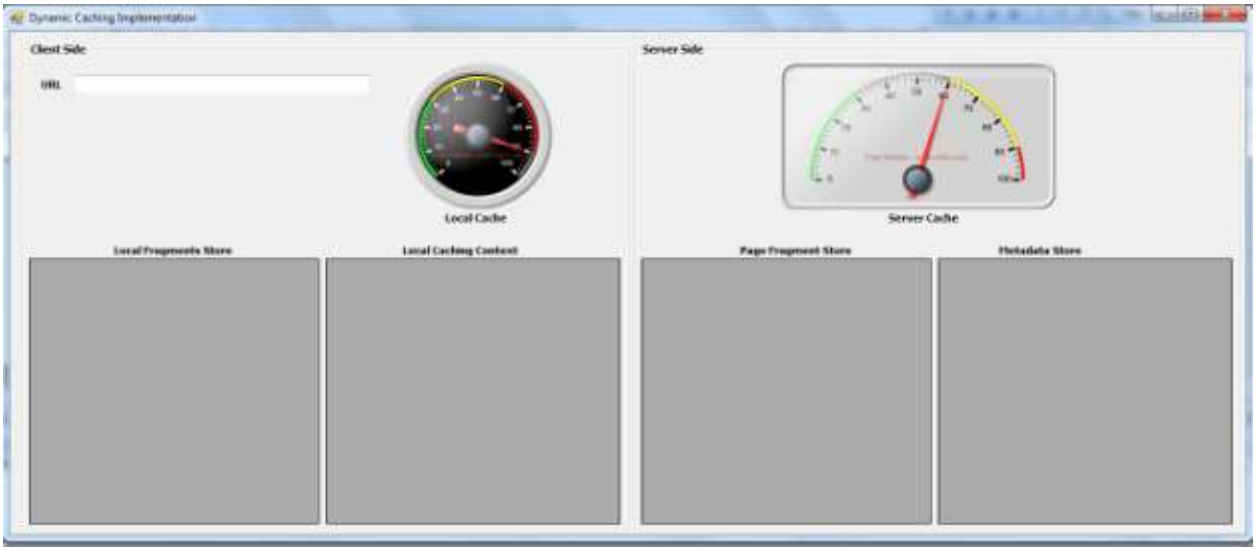
الشكل(4) المخطط التدفقي للخوارزمية المقترحة

الاستنتاجات والتوصيات:

قمنا بمحاكاة عمل الخوارزمية المقترحة من خلال افتراض وجود مخدم ويب يقوم بتوليد الصفحات الديناميكية وتخزينها في قاعدة البيانات السابقة حيث يتم تخزين اجزاء الصفحات التي تم توليدها ديناميكياً في المخزن السلبي كما يتم تخزين المعلومات الموصفة لهذه الأجزاء في الجدول Server_Metadata_store بينما عناوين الصفحات التي تمت عليها عملية التخزين المؤقت تخزن في الجدول المسمى Server_URL.

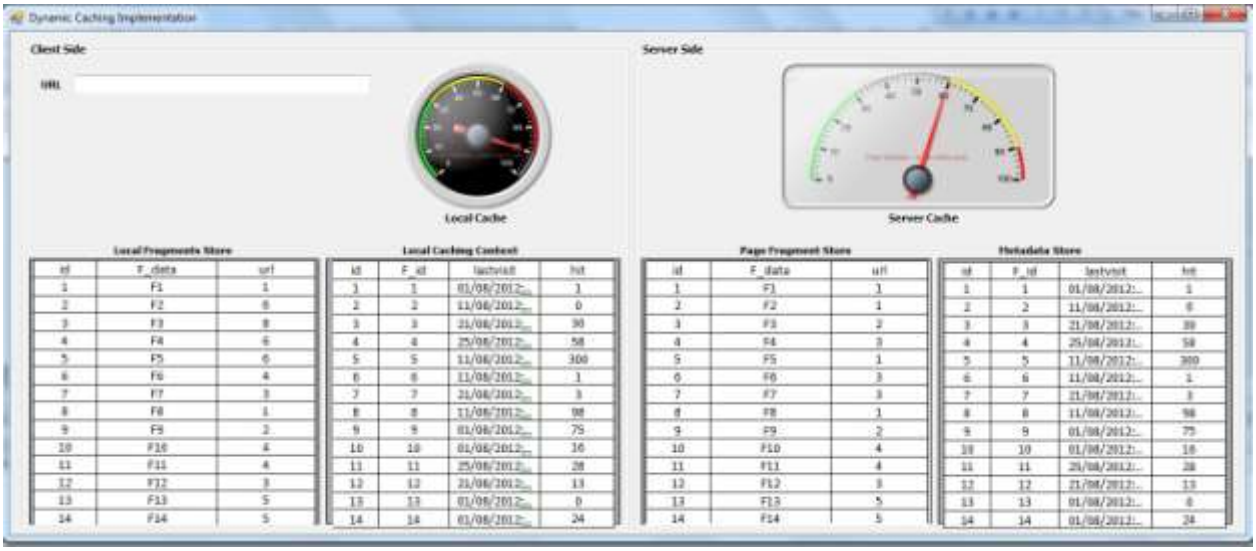
قمنا بتصميم برنامج محاكاة الخوارزمية المقترحة باستخدام لغة C#.Net تحت بيئة التطوير المشهورة Microsoft Visual Studio 2008.

ويبين الشكل (5) لقطة للبرنامج قبل تجريب البيانات:



الشكل(5) برنامج محاكاة الخوارزمية

وبعد ملء قاعدة البيانات بالبيانات التجريبية يظهر البرنامج كما في الشكل (6):



الشكل(6) برنامج محاكاة الخوارزمية بعد تنفيذ الخوارزمية

طريقة عمل البرنامج:

يقوم البرنامج بمحاكاة عمل الزبون او مخدم الويب الذي سيقوم بطلب الصفحات المولدة من قبل المخدم الرئيسي وهذا الزبون لديه جداوله الخاصة التي تحتوي على معلومات الكاش الخاصة به مثل أجزاء صفحات الويب التي تمت عملية التخزين المؤقت لها عند زيارتها من قبل هذا الزبون حيث ستخزن في جدول قاعدة البيانات السابقة المسمى Client_fragment_store كما تخزن البيانات التوصيفية الخاصة بها في الجدول المرادف المسمى client_metadata_store والذي تتم عملية تحديثه تبعا لتحديث عملية الوصول إلى الجزء fragment الموافق له.

عندما يطلب الزبون عنواناً لصفحة ويب معينة يقوم البرنامج بالبحث عن أجزاء الصفحات المقترنة بهذا العنوان والموجودة في كل من الجدولين server_url و client_url وعندها تتم عملية طلب كل جزء من هذه الأجزاء المكونة للصفحة الديناميكية كما يلي:

- يتم أولاً البحث عن الجزء في الكاش المحلي للزبون وعند وجوده يتم اختبار تاريخه ومقارنته مع تاريخ الصلاحية الموجود على المخدم وفي حالة المطابقة يتم تحميل هذا الجزء من كاش الزبون وليس من كاش المخدم.
- إذا لم يكن التاريخ متطابقاً تتم عملية إعادة جلب الجزء من كاش المخدم إلى كاش الزبون عبر تحميل الجزء الجديد الموجود في الجدول server_fragment_store إلى الجدول client_fragment_store مع حذف الجزء المنتهية صلاحيته.
- كما تتم عملية تحديث البيانات التوصيفية لهذا الجزء والموجودة على الكاش المحلي وكذلك البيانات التوصيفية له على المخدم الرئيسي من عدد مرات الطلب إلى تاريخ آخر طلب.
- إذا لم يكن الجزء موجوداً على الكاش المحلي تتم هنا عملية تحميل الجزء من المخدم إلى كاش الزبون وتحديث البيانات التوصيفية الخاصة به.

عندما يمتلئ الكاش أو مخزن الأجزاء المؤقتة سواء على المخدم الرئيسي أو على الزبون فإننا نستخدم خوارزميتي LRU و LFU في استبعاد الأجزاء ذات الاستخدام القليل التكرار والتي مضى على استخدامها فترة زمنية طويلة نسبياً من خلال الاستفادة من حقلي المعلومات التوصيفية الموجودين في جداول الميتاداتا في كل من المخدم والزبون وهما عدد مرات الزيارة وتاريخ آخر طلب.

من خلال عمل برنامج المحاكاة لخوارزميتنا المقترحة لاحظنا أن الخوارزمية الجديدة توفر الكثير من الحمل الزائد على المخدم من خلال اختصار العديد من عمليات التحميل عند طلب أجزاء تكون موجودة أصلاً في الكاش المحلي وهذا التوفير لا يقتصر فقط على الحمل على المخدم بل يتعداه إلى زمن الاستجابة المستغرق عند طلب الصفحة وهذا يزيد من فعالية وأداء مخدمات الويب.

إن هذه الخوارزمية ملائمة جداً لمخدمات الويب التي تحتوي على مواقع تولد الصفحات الديناميكية وغير الديناميكية وهي سهلة التحقيق كما شاهدنا في برنامج المحاكاة الخاص بنا. علاوة على ذلك فإن هذه الخوارزمية تتسم بالشفافية حيث أن أي مخدم ويب يطلب الصفحة يقوم تلقائياً بمزامنتها مع المخدم الرئيسي وتحديث بيانات الكاش لديه وكلما كان حجم الكاش لديه أكبر كلما زادت فعالية الخوارزمية أكثر وكلما خف الحمل وقل وقت الاستجابة على المخدم أكثر.

ونظراً لكل ما سبق من توفير للحمل الزائد على المخدمات واتسام الخوارزمية المقترحة بالشفافية حيث أن أي مخدم ويب يطلب الصفحة يقوم تلقائياً بمزامنتها مع المخدم الرئيسي وتحديث بيانات الكاش لديه وكلما كان حجم الكاش

لديه أكبر كلما زادت فعالية الخوارزمية أكثر وكلما خف الحمل وقل وقت الاستجابة على المخدم أكثر لذلك نوصي باعتماد هذه الخوارزمية في مخدمات الويب فهي مناسبة لجميع أنواع الصفحات الديناميكية وغير الديناميكية.

المراجع:

[1] Abdulla, G; Fox, E. A; Abrams, M; Williams, S. *Www proxy traffic characterization with application to caching*. Mar 1998.

[2] Abrams, M; Standbridge, C.R; Abdulla, G; Williams, S; Fox, E. A. *Caching Proxies: Limitations and Potentials*. Boston Conference, Dec 1995.

[3] Bestavros, A; Carter, R; Crovella, M; Cunha, C; Heddaya, A; Mirdad, S. *Application Level Document Caching in the Internet*. June 1995.

[4] Candan, K; Li, W. S; Luo, Q; Hsiung, W. P; Agrawal, D. *Enabling dynamic content caching for database-driven web sites*. May 2001.

[5] Cao, P; Irani, S. *Cost-aware WWW proxy caching algorithms***.** December 1997.

[6] Challenger, J; Iyengar, A; Dantzig, P. *A Scalable System for Consistently Caching Dynamic Web Data*. 1999.

[7] Challenger, J; Iyengar, A; Witting, K; Ferstat, C; Reed, P. *A Publishing System for Efficiently Creating Dynamic Web Data*. March 2000.

[8] Cohen, E; Kaplan, H. *Refreshment Policies for Web Content Caches*. April 2001.

[9] Cohen, E; Kaplan, H**.** *Exploiting regularities in Web traffic patterns for cache replacement*. May 1999.

[10] Datta, A; Dutta, K; Fishman, D; Ramamritham, K; Thomas, H; VanderMeer, D. *A Comparative Study of Alternative Middle Tier Caching Solutions to Support Dynamic Web Content Acceleration.* September 2001.

[11] Datta, A; Dutta, K; Fishman, D; Ramamritham, K; Thomas, H; VanderMeer, D. *Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic Web Page Generation*. May 2001.

[12] Datta, A; Dutta, K; Fishman, D; Ramamritham, K; Thomas, H; VanderMeer, D; Suresha.

Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. June 2002.

[13] Datta, A; Dutta, K; Ramamritham, K; Thomas, H; VanderMeer, D. *Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation*. June 2004.

[14] Duchamp, D; *Prefetching Hyperlinks*. October 1999.

[15] Zheng, D; *Differentiated Web Caching A Differentiated Memory Allocation Model on Proxies*. (2004).

[16] Bhattacharjee, A; Debnath, B. K**.** *A New Web Cache Replacement Algorithm,* 2005.

[17] Suresha; Jayant; Haritsa, R. *On Reducing Dynamic Web Page Construction Times***.** 2004.

[18] Suresha**.** *Caching Techniques for Dynamic Web Servers.* June 2007.

[19] Iyengar, A; Challenger,J. *Improving Web Server Performance by Caching Dynamic Data***.** 1997.

[20] Desai, K. *Bandwidth Optimization Using Content Aliasing Of Proxy Server***.** 2009**.**

[21] Bouchenak, S; Cox, A; Dropsho, S; Mittal, S; Zwaenepoel, W. *Caching Dynamic Web Content: Designing and Analyzing an Aspect-Oriented Solution***.** Springer Berlin Heidelberg,2006.

[22] Wikipedia. (2009, June) Static Web Page. [Online]. Available: http://en.wikipedia.org/wiki/Static_web_page.

[23] Wikipedia. (2009, June) Dynamic Web Page. [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_web_page.

[24] W3C, October 2009. HTTP/1.1, part 6: Caching, http://tools.ietf.org/html/draft-ietfhttpbis- p6-cache-08.

[25] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The Web Changes Everything: Understanding The Dynamics Of Web Content. In *Proc. WSDM*, 2009.

[26] H. Artail and K. Fawaz. A Fast HTML Web Change Detection Approach Based On Hashing And Reducing The Number Of Similarity Computations. *Data Knowl. Eng.*,66(2), 2008.

[27] M. Oita and P. Senellart. Archiving Data Objects Using Web Feeds. In *Proc. IWAW*, 2010.

[28] H. P. Khandagale and P. P. Halkarnikar. A Novel Approach For Web Page Change Detection System. *Intl. J.*

Comput. Theory Eng., 2(3), 2010.

[29] C. Kholschutter, P. Fankhauser, and W. Nejdi. Boilerplate Detection Using Shallow Text Features. In

Proc. WSDM, 2010.