مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(3) 2021
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (3) 2021

تقييم أداء موازني الحمل: Varnish و HAproxy في تخديم صفحات الويب الثابتة و الديناميكية باستخدام الخوارزمية الدائرية

د. م. أحمد محمود أحمد*

(تاريخ الإيداع 24 /3 /2021 . قُبِل للنشر في 18 /5 /2021)

□ ملخّص □

تعتبر خدمات الويب أساسية في عالم الإنترنت اليوم، حيث يقوم أي زبون يستخدم الإنترنت بطلب صفحات الويب من المخدمات. تكون بعض هذه الصفحات ثابتة، بمعنى أن المحتوى جاهز على المخدم الذي يقوم فقط بإرسال المحتوى الى الزبون، بينما يكون بعضها الآخر ديناميكياً، أي تحتاج تلك الصفحات تنفيذ برنامج ما على المخدم كي يعيد النتيجة للزبون. من هنا، فإن مخدم واحد غير قادر على تخديم العدد المتزايد من الزبائن على الإنترنت، لذلك لا بد من تجميع عدد من المخدمات فيما يسمى عنقود (cluster) من أجل تخديمهم. لكي نكون قادرين على استخدام العنقود بشكل فعال في التخديم لا بد من استخدام موازن حمل كي يقوم بتوجيه الطلبات إلى المخدم الذي يتضمن الصفحات المطلوبة. يقدم هذا البحث دراسة موازني حمل هما: Varnish و HAproxy باستخدام الخوارزمية الدائرية، يتميز Varnish عن HAproxy بأنه يقوم بتخبئة Cache البيانات قبل إرسالها للزبون كي يستطيع استخدامها لاحقاً في تسريع الطلبات. نلاحظ أن عملية التخبئة هذه تساعد في تحسين الأداء بشكل كبير بالنسبة للصفحات الثابتة و الصفحات الديناميكية من حيث تقليل زمن الاستجابة و رفع معدل الطلبات مع تقليل احتمال الأخطاء.

الكلمات المفتاحية: خدمات الويب، عنقود المخدمات، موازن الحمل، HAproxy، Varnish، Apache، التخبئة المؤقتة، Locust .

* عضو هيئة تدريس في قسم النظم الشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – سوريا.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(3) 2021
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (3) 2021

Performance Evaluation of Varnish and HAproxy Load Balancers in serving static and dynamic web pages using Round-Robin Algorithm

Dr. Ahmad Mahmoud Ahmad *

(Received 24/ 3 / 2021 . Accepted 18/ 5/ 2021)

□ ABSTRACT □

Web services are essential in today's internet world, as any customer who uses the Internet will request web pages from the servers. Some of these pages are static, which means the content is ready on the server that only sends the content to the customer. The other type of pages is dynamic, that is, those pages need to implement a program on the server in order to return the result to the customer. In all cases, one server is not able to serve the increasing number of customers on the Internet, so it is necessary to collect a number of servers in the so-called cluster in order to serve them. To use the cluster efficiently, it is necessary to use a load balancer to direct the requests to the server that contains the required pages. This paper presents a study of two load balances: Varnish and HAproxy using round-robin algorithm. Varnish is distinguished from HAproxy in that it Caches data before sending it to the customer, so that it can be used later to speed up requests. Our results denote that this caching process greatly improves the performance by reducing response time and increasing the rate of served requests while reducing the possibility of errors, all that when requesting static and dynamic pages.

Keywords: Web services, cluster, load balancer, HAproxy, Varnish, Apache, web Cache, Locust

* Lecturer – Department of Computer Systems and Networks – Faculty of Informatics engineering – Tishreen University – Syria.

1. مقدمة

إن مخدمات الويب هي أساس كل نشاط يستخدم الإنترنت تقريبًا، حيث يقوم مخدم ويب بالرد على طلبات الزبون و وضع الخدمة في المتناول، حيث تقوم خدمة الويب (Web service) بتقديم خدمات إلكترونية بين التطبيقات أو الأنظمة من خلال الرد على التطبيق عند طلبه الخدمة (Service Request). يتم تعريف قابلية التوسع في خدمة الويب على أنها القدرة على التعامل مع الحمل المتزايد للزبائن [1]، حيث يسعى الباحثون لأن تكون العلاقة خطية بين عدد طلبات الزبون و بين قابلية التوسع. بكافة الأحوال، فإن العلاقة ليست خطية لا بل إن تحقيق قابلية التوسع يعاني من مشكلة ازدياد عدد الزبائن و ازدياد عدد العناصر المطلوبة.

جاءت فكرة تعدد المخدمات لحل مشكلة ازدياد طلبات الخدمات من جهة و ازدياد عدد الزبائن من جهة أخرى، لذلك كان لابد من توزيع الحمولات بين المخدمات، و بعبارة أخرى، تحقيق منهجية للاستخدام الفعال للموارد الممثلة على شكل عنقود يربط مجموعة من المخدمات مع بعضها، تسمى تلك المنهجية بموازن الحمل. موازن الحمل هو جهاز أو أداة برمجية مصممة لتوزيع الحمال بين المخدمات المختلفة، حيث تم تطوير العديد من الخوارزميات لتحقيق ذلك معتبرة زمن الاستجابة كمقياس الأداء الأهم في ظروف ازدياد عدد الطلبات، الذي قد يترافق أو لا يترافق مع ازدياد عدد الزبائن . تأخذ موازنات الأحمال أهمية كبرى مع ظهور مواقع الويب الحديثة ذات عدد الزيارات المرتفع الذي يتطلب معالجة آلاف الطلبات المتزامنة من العملاء، و إعادة الردود المتمثلة بنصوص أو صور أو فيديو أو بيانات التطبيق أو جميعمها مع بعضها أو جزء منها، كل ذلك بطريقة سريعة ومحددة. لتحقيق ذلك، ظهر HAProxy كحل مجاني وسريع للغاية وموثوق يحقق التوافرية العالية و موازنة الأحمال و يعمل كوكيل للتطبيقات المعتمدة على TCP و HTTP، و هو مناسب بشكل خاص لمواقع الويب عالية الحركة للغاية حيث يستخدم في عدد كبير من المواقع الأكثر زيارة في العالم . يتم تضمينه الآن ضمن معظم توزيعات Linux السائدة ، وغالبًا ما يتم نشره افتراضيًا في الأنظمة الأساسية السحابية [2].

بزيادة عدد التجهيزات الحاسوبية، التي تمكن من الوصول الى خدمات الويب من أي مكان و في أي وقت، أصبح هاجس المطورين هو تمكين الزبائن من تحقيق وصول أسرع وأسهل إلى البيانات بأنواعهما المختلفة. لذلك أتت فكرة التخزين المؤقت (Caching) لتكون أحد ركائز تحسين أداء الويب، و يظهر تأثيرها بشكل جلي عند استخدامها مع موازنات الأحمال. ظهر Varnish [3] ليكون أحد الحلول و هو مسرع لتطبيقات الويب، حيث يمكن استخدامه للتخزين المؤقت لكل من المحتوى الديناميكي والثابت، مما يجله فعالاً ليس فقط لزيادة سرعة مواقع الويب و إنما أيضًا لأداء المخدمات. يتم استخدام Varnish اليوم في مواقع الويب من جميع الأنواع ، من Facebook و Wikia و Slashdot[4].

نقدم في هذا البحث تقييماً لأداء موازني الحمل: Varnish و HAproxy في تخديم صفحات الويب الثابتة و الديناميكية. تم توزيع ما تبقى من هذه الورقة البحثية على أربعة محاور أساسية: يعرض المحور الأول مشكلة البحث، يلي ذلك أهميته. أما المحور الثاني فيعرض الدراسات المرجعية التي تستخدم موازنات حمل مختلفة. أما المحور الثالث فيتطرق الى خوارزميات موازنة الحمل المستخدمة في HAproxy و Varnish و من ثم عرض مميزات التخبئة التي يقدمها Varnish. أما في المحور الرابع و الأخير فيتم تحقيق التقييم و التجارب و استخلاص و تحليل النتائج و عرض الخلاصة و التطلعات المستقبلية.

2. مشكلة البحث

إن تحقيق قابلية التوسع بالإضافة لموازنة الأحمال يمثل تحديًا للباحثين من أجل تلبية متطلبات وصول الزبائن المتعددة والمتزامنة، ومع الانتشار الواسع لخدمات الويب، أصبح من الضروري جداً العمل على تسريع عملية تخديم طلبات الزبائن من مواقع الويب، يتطلب هذا الأمر مخدمات ذات مقدرات كبيرة جداً من ناحية وحدة المعالجة المركزية و الذاكرة الرئيسية. إن استخدام مخدم واحد كبير لمواقع الويب يجعل منه نقطة فشل وحيدة، أي بمعنى أنه إذا توقف هذا المخدم عن العمل، سيوقف الموقع كاملاً، كما أن استخدام مخدم واحد يجعل عملية التوسع صعبة في المستقبل. يتم عادة استخدام عدة مخدمات و موازنة الحمل بينها عن طريق موازن حمل متخصص، لكن تمرير جميع الطلبات للمخدمات من أجل تخديمها سيزيد من العبء عليها لذلك كان لا بد من استخدام التخبئة المؤقتة (caching)، مع العلم أن تقنيات التخزين المؤقت التقليدية تعاني من قيود فيما يتعلق بصفحات التخزين المؤقت الديناميكية [1]. ظهرت فكرة تحقيق التخزين المؤقت ضمن موازن الحمل لتقليل العبء على المخدمات، لذلك فإن مشكلة البحث تتلخص في التأكد من مدى كفاءة استخدام التخبئة المؤقتة في موازنات الحمل.

3. أهمية البحث

يهدف هذا البحث إلى دراسة عملية التخبئة بالإضافة إلى موازنة الحمل و كيف يمكن أن تلعب دوراً في تقليل زمن الاستجابة من ناحية، و تقليل العبء على مخدمات الويب من ناحية أخرى. سنستخدم من أجل تحقيق هذا الأمر موازني حمل هما: HAproxy ، و Varnish و هما من أكثر موازات الحمل استخداماً، لكن الموازن الأول لا يدعم التخبئة المؤقتة بينما يدعمها الآخر، حيث لا توجد في الأدبيات البحثية، حسب علمنا، أي مقارنة بينهما.

4. الدراسات المرجعية

قام الباحثون في [5] بنشر ورقة بحثية عن استخدام مخدم الويب و موازن الحمل المعروف Nginx مع أنظمة لينكس معزولة (Linux Containers) في موازنة الحمل لعنقود ويب يتألف من عدة مخدمات باستخدام خوارزمية موازنة حمل ديناميكية و محسنة، أظهرت النتائج تحسن في زمن الاستجابة حوالي 52.5% و 46.4% مقارنة مع الخوارزمية الدائرية (Round-Robin) و خوارزمية الإتصالات الأقل (Least-Connections). من جهة ثانية، قام الباحثون في [6] بتقديم خوارزمية محسنة عن خوارزمية الإتصالات الأقل بحيث تقوم باستبعاد أي مخدم جديد من عملية الجدولة ريثما يتم إرسال طلبات لباقي المخدمات بعد انضمامه، و ذلك لتجنب إغراق المخدمات الجديدة بالطلبات عند أول انضمامها. أظهرت تلك الخوارزمية تحسناً كبيراً في الأداء. كما قام بعض الباحثين في جامعة ديبونيغورو في إندونيسيا [7] بدراسة أداء كل من Heartbeat و HAproxy باستخدام خوارزميات موازنة الحمل المختلفة، حيث توصلوا إلى أن خوارزمية الإتصالات الأقل تتفوق على الخوارزمية الدوائرية و خوارزمية المصدر في معدل الإتصال، زمن الاستجابة، و معدل الأخطاء. كما أظهرت الورقة البحثية [8] تحسناً كبيراً في تخديم صفحات ويكيبيديا عند استخدام التخبئة لأكثر الصفحات زيارة، حيث تم استخدام مجموعة بيانات متوفرة للعامة لأكثر من 1000 صفحة يتم طلبها في

ويكيبيديا، ساهمت التخبئة في تقليل زمن الاستجابة بشكل كبير و تخفيف الحمل على المخدمات. تم تقييم أداء خوارزمية تخبئة جديدة تعتمد على الجمع بين الأقل استخداماً مؤخراً LRU و اختيار الكائنات المبني على العلامة score based object selection، في الورقة البحثية [9]، حيث أظهرت النتائج تحسن في معدل الإصابة hit rate باستخدام الخوارزمية الجديدة بمقدار 10−20% مقارنة مع خوارزمية الأقل استخدام مؤخراً لوحدها، تم إجراء الاختبارات بالاستعانة بالبيانات العامة حول أكثر من 1000 صفحة يتم طلبها على الويكيبيديا. تم في [10] دراسة خوارزمية تبديل كاش جديدة تعتمد على التنقيب في الويب، تعتبر خوارزمية التبديل الجيدة ضرورية جداً، و خصوصاً عندما تكون الذاكرة ممتلئة و نحتاج إلى حذف بعض البيانات لوضع بيانات جديدة مكانها. أظهرت هذه الخوارزمية الجديدة تحسن كبير في معدل الإصابة hit rate للبيانات المخبأة.

قام الباحثون في [11] بتنفيذ خوارزميات موازنة الحمل على عناقيد ويب غير متجانسة تستخدم HAProxy. أظهرت النتائج أنه يتم الحصول على أفضل أداء من خلال خوارزمية round−robin فيما يخص استخدام الموارد والاتصال بالشبكة وطلب الخدمات، بينما ظهرت أسوأ النتائج عند استخدام خوارزمية الأقل اتصالًا. من جهة أخرى، تم في [12] تقييم أداء الخوارزمية الدائرية على موازني الحمل HAproxy و Nginx، خلص الباحثون الى أن استخدام الاعدادات الافتراضية في كلا موازني الحمل يقود الى تفوق HAproxy، لكن عند تغيير اعدادات الخوارزمية Nginx، فإن ذلك يقود الى ظهور أداء عالي. استخدم الباحثون الخوارزمية الدائرية لكونها الأنسب عند استخدام مخدمات متشابهة.

انطلاقاً من الورقة البحثية [1] التي أشارت الى أن تقنيات التخزين المؤقت التقليدية تعاني من قيود فيما يتعلق بصفحات التخزين المؤقت الديناميكية، و [11] التي خلصت الى تفوق الخوارزمية الدائرية عند بارامترات محددة و [12] التي أشادت باستخدام الخوارزمية الدائرية لكونها الأنسب عند استخدام مخدمات متشابهة، فإننا نقدم دراسة لمقارنة موازني الحمل المذكورين سابقاً، حيث احدهما يستخدم التخبئة المؤقتة بينما لا يستخدمها الآخر، منطلقين من فرضية أن المخدمات المستخدمة متماثلة.

5. موازنة الحمل و التخبئة ضمن HAproxy و Varnish:

تستهدف هذه الورقة البحثية مخدمين معروفين في موازنة الحمل و هما HAproxy و Varnish.

1.5. Haproxy:

هو موازن حمل عالي التوافرية (High Availability)، يستخدم في توزيع الحمل على عدة مخدمات كي تقوم بتخديم طلبات الزبائن، مكتوب بلغة البرمجة C، و يستخدم من قبل عدد كبير من الشركات مثل twitter, github و instagram ....الخ. يعتمد في عمله كموازن حمل على مفهومين هما [6]:

- قوائم التحكم بالوصول (Access Control List): يتم استخدام هذه القوائم من أجل معالجة الطلبات القادمة و استخلاص معلومات منها كي يتم استخدامها في موازنة الحمل و اختيار الواجهة الخلفية (Backend) التي سيتم استخدامها لإرسال الطلب إليها.
- الواجهة الخلفية (Backend): تتضمن الواجهة الخلفية مجموعة مخدمات سيتم إرسال الطلبات إليها، يتم تعريف هذه المخدمات من خلال عنوان IP و منفذ، يتم ايضاً استخدام خوارزمية موازنة الحمل من أجل اختيار أحد المخدمات كي يتم إرسالها إليه.

2.5. Varnish:

يعتبر Varnish مسرع تطبيقات ويب، و يسمى أيضاً وسيط HTTP العكسي (HTTP reverse proxy)، مكتوب بلغة ال C، و يقوم بتخزين الإستجابة التي تصل من مخدمات الويب من أجل استخدامها في تخديم الطلبات التالية، يقوم بتسريع الاستجابة بشكل كبير جداً في حال تم إعداده بالشكل الصحيح. يتميز Varnish بالأداء المذهل، فبإمكانه تخديم طلبات من رتبة الغيغا بت في الثانية الواحدة، كما يتميز أيضاً بالمرونة و سهولة الإعداد حيث يستخدم لغة مجال متخصصة (Domain Specific Language)، تسمى لغة إعداد Varnish (Varnish Configuration Language) أي VCL، تسمح هذه اللغة بتعريف إعدادات كثيرة لـ Varnish تتحكم بكيفية اختيار المخدم الذي سيتم إرسال الطلب إليه و كيفية تخبئة الاستجابة من أجل أن يتم إرسالها لاحقاً، يتم ترجمة ملفات الـ VCL باستخدام مترجم C متخصص و ربطها مع عملية Varnish على المخدم مما يعني أن اللغة لا تسبب أي تأثير سلبي على الأداء.

6. خوارزميات موازنة الحمل في Varnish و HAproxy:

تعتبر موازنة الحمل أمر أساسي جداً في تخديم مواقع الويب و ذلك بسبب العدد الكبير والمتزايد من زبائن الانترنت، لذلك يستخدم عدد كبير من المخدمات من أجل تخديم العدد المتزايد من الزبائن. يجب أن يتم وضع موازن الحمل بين الزبون و المخدمات المختلفة التي تستضيف مواقع الويب، و ذلك لأن الزبون لا يجب أن يقوم بطلب الخدمات من المخدمات المختلفة مباشرةً، و إنما هو بحاجة لعنوان ثابت يقوم بطلب جميع الخدمات منه، و هنا يقوم موازن الحمل باختيار أحد المخدمات المختلفة لتخديم الطلب. تسمى عملية اختيار المخدم الذي سيقوم بتخديم الطلب بخوارزمية موازنة الحمل، هنا سنقوم بدراسة عدد من الخوارزميات المستخدمة في عالم الويب [6]:

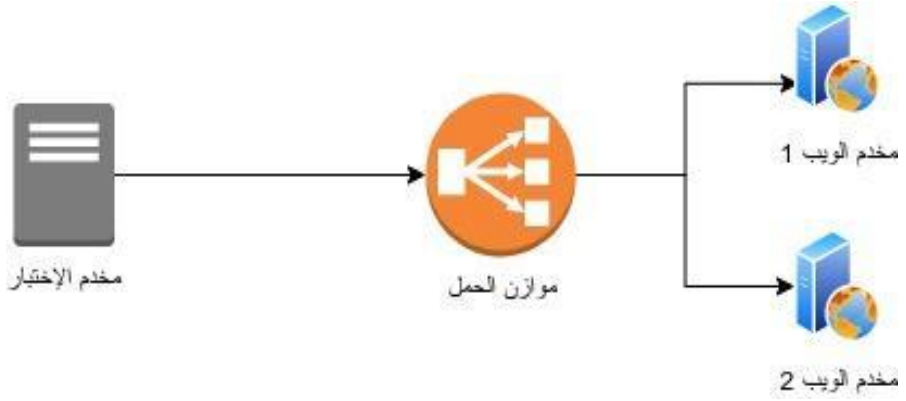
- الخوارزمية الدائرية (Round Robin): يتم في هذه الخوارزمية اختيار المخدم الذي سيقوم بتخديم الطلب بشكل دائري من قائمة مخدمات مختلفة، بحيث يتم ترتيب المخدمات و الإختيار بينها الواحد تلو الآخر. تقدم هذه الخوارزمية أداء جيد، لكن في حال كان زمن معالجة الطلبات مختلف بشكل كبير بين زبون و آخر فإن الأداء سيتراجع بشكل كبير و بعض المخدمات سيكون عليها حمل أكبر من المخدمات الأخرى.
- الخوارزمية الدائرية الموزونة (Weighted Round Robin): تشبه هذه الخوارزمية الخوارزمية السابقة مع الاختلاف بأنه يتم تعيين وزن لكل مخدم، هذا الوزن يتناسب طرداً مع قدرة المخدم على معالجة الطلبات، فلو كان لدينا مخدمين مثلاً، الأول ذو مواصفات أفضل من الثاني بمقدار 4 أضعاف، فإنه يتم تعيين وزن 4 للأول و وزن 1 للثاني، و في هذه الحالة سيتم إرسال أول 4 طلبات للمخدم الأول، ثم يتم إرسال طلب واحد للمخدم الثاني، ثم نعود للمخدم الأول، نستطيع بهذه الطريقة إرسال طلبات أكثر للمخدم ذو المواصفات الأفضل.
- خوارزمية الإتصال الأقل (Least Connection Algorithm): تعتبر خوارزمية ديناميكية، حيث تقوم بتخزين عدد الإتصالات التي تم إنشاؤها مع كل مخدم، و عند ورود طلب جديد فإنه يتم

إرساله للمخدم ذو الإتصالات الأقل، بحيث لا يتم إغراق أي مخدم يقوم بمعالجة طلب لفترة طويلة بطلبات أخرى والذي يسبب في زيادة زمن الاستجابة بشكل كبير.

- خوارزمية الإتصالات الأقل الموزونة (Least Connection Algorithm): تشبه هذه الخوارزمية سابقتها، إلا أنها تختلف في تعيين وزن لكل مخدم، هذا الوزن يتناسب طرداً مع مواصفات المخدم، يتم هنا إرسال أكثر من طلب للمخدمات ذات الوزن الأكبر مقارنة مع المخدمات ذات الوزن الأقل، تعتبر هذه الخوارزمية بالنسبة لخوارزمية الإتصالات الأقل مثل الخوارزمية الدائرية الموزونة بالنسبة للخوارزمية الدائرية.

7. القسم العملي:

تم إعداد كل من HAproxy و Varnish باستخدام الخوارزمية الدائرية، كما تم استخدام مخدمي Apache و الذي سيتضمن صفحة الويب الثابتة و الأخرى الديناميكية من أجل إجراء الإختبار عليهما. يقوم Varnish، بالإضافة لعمله كموازن للحمل، بالقيام بتخبئة الاستجابة المستقبلة من Apache كي تستخدم لاحقاً، مما يسرع عملية إرسال الاستجابة للزبون بشكل كبير، في حين يقوم مخدم HAproxy فقط بعملية موازنة الحمل وإرسال الطلب لمخدمات Apache في كل مرة، وهذا ما حفزنا لدراسة تأثير عملية التخبئة مع عملية موازنة الحمل على أداء النظام. لتنفيذ التجارب العملية، قمنا باستخدام موازن حمل (HAproxy تارةً و Varnish تارةً أخرى)، مخدمي ويب و مخدم اختبار، تم ترتيب البنية كما هو موضح في الشكل (1) والذي يوضح البنية التي تم إجراء التجارب عليها.



الشكل (1) البنية المستخدمة في التجارب.

يوضح الجدول (1) مواصفات المخدمات الأربعة المستخدمة في التجارب:

الجدول (1) مواصفات المخدمات المستخدمة في التجارب.

المخدم	الذاكرة	المعالج	القرص الصلب
مخدم الإختبار	16 غيغا	ثماني النواة	320 غيغا SSD
موازن الحمل	8 غيغا	رباعي النواة	160 غيغا SSD
مخدم الويب 1	8 غيغا	رباعي النواة	160 غيغا SSD
مخدم الويب 2	8 غيغا	رباعي النواة	160 غيغا SSD

تم استخدام الأداة locust في الاختبار، و التي تستخدم مكتبات متخصصة من أجل توليد حمل كبير على النظام المراد اختباره و إغراقه بالطلبات، و تقوم أيضاً بحفظ نتائج الإختبار ضمن ملفات CSV من أجل أن يتم معالجتها لاحقاً. تستخدم هذه الأداة لغة البرمجة بايثون من أجل توصيف الاختبارات المراد القيام بها. على الرغم من وجود أدوات أخرى ممكنة الاستخدام كـ Gatling و Apache Jmeter ، إلا أن locust لديه أعلى معدل نقل، وأقل زمن استجابة وأقل وقت تنفيذ يتبعه JMeter ، ثم يأتي Gatling و ذلك طبقاً لبعض السيناريوهات [13].

تم تشغيل 20 اختبار، مدة كل اختبار هي 5 دقائق، و تم إعادة الإختبارات 6 مرات من أجل الحصول على متوسط النتائج و لتجنب أي نتيجة شاذة قد تظهر في بعض الاختبارات أحياناً. تم استخدام بارامترين في تحديد كل اختبار و هما:

- عدد الزبائن الكلي: يحدد العدد الأعظمي من الزبائن الذين سيقومون بطلب صفحات الويب المراد اختبارها و ذلك خلال فترة 5 دقائق.
- معدل توليد الزبائن: يدل على عدد الزبائن الذين سيتم إضافتهم كل ثانية للنظام.

فمثلاً، تدل الثنائية 10000.200 الى أن عدد الزبائن الكلي هو 10000 بينما معدل توليد الزبائن هو 200.

عندما تبدأ الأداة بالعمل، لا يتم إرسال طلبات من جميع الزبائن دفعة واحدة، بل يتم إضافة عدد محدد من الزبائن للنظام كل ثانية من أجل أن يقوموا بإرسال الطلبات، ينتظر كل زبون بين 5 و 10 ثواني لإرسال الطلب التالي. سنستخدم من أجل تقييم الأداء ثلاثة معايير و هي:

1. متوسط زمن الاستجابة.
2. عدد الطلبات المخدمة في الثانية.
3. معدل الأخطاء.

سيتم تقييم أداء النظام في الحالات الثلاثة التالية:

1. مخدم Apache مستقل.
2. HAproxy، والذي لا يستخدم تخبئة (Caching)، مع مخدمي Apache .
3. Varnish والذي يستخدم تخبئة (Caching)، مع مخدمي Apache .

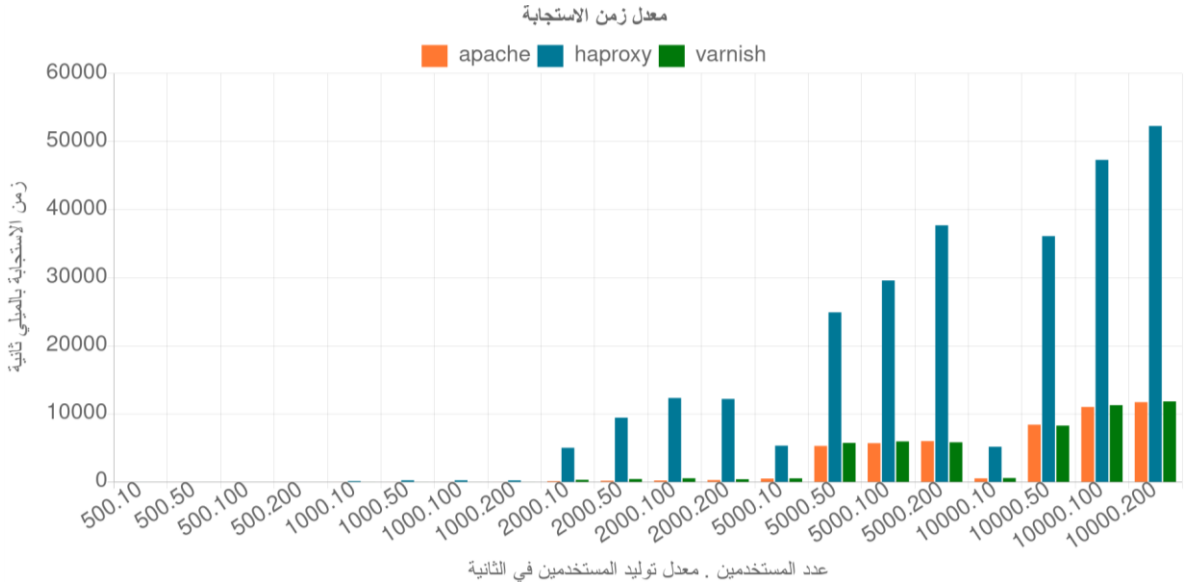
سنقوم بتنفيذ الاختبارات في حالتين:

1. عند تخديم صفحة HTML واحدة ثابتة.

2. عند تخديم صفحة ديناميكية تتضمن جزء ديناميكي يتم توليده على المخدم.

1.7. تقييم الأداء على صفحة HTML واحدة ثابتة:

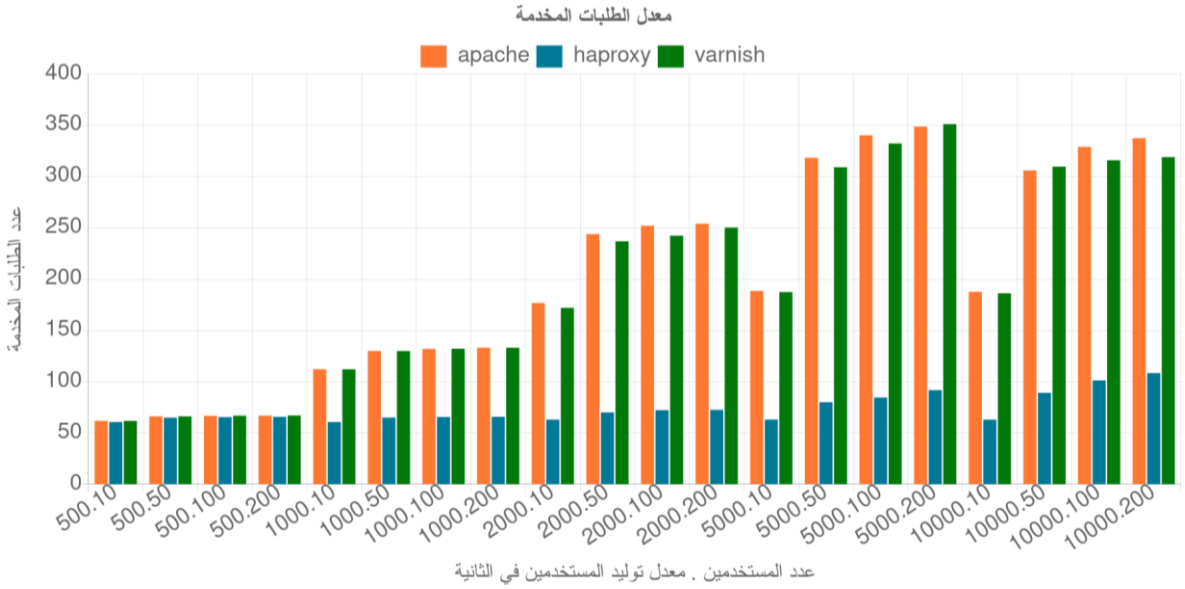
يوضح الشكل (2) متوسط زمن الاستجابة عند تخديم صفحة ويب ثابتة.



الشكل (2) متوسط زمن الاستجابة عند تخديم صفحة ويب ثابتة.

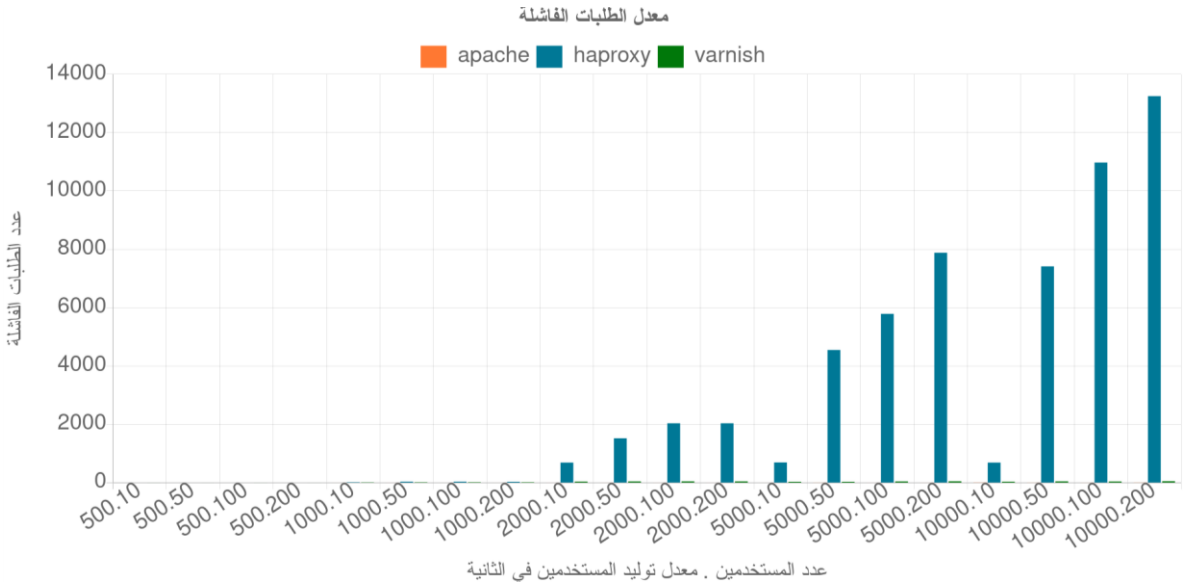
نلاحظ من الشكل (2) أنه في جميع الحالات الثلاثة المذكورة سابقاً، يزداد زمن الاستجابة بزيادة عدد الزبائن و بازدياد معدل توليد الزبائن. نلاحظ أيضا تقارب في الأداء بين Apache و Varnish حيث تكون الزيادة في زمن الاستجابة مقبولة نسبياً، في حين أن أداء HAproxy يتراجع بشكل كبير مع تزايد عدد الزبائن.

يوضح الشكل (3) معدل الطلبات المخدمة في الثانية الواحدة، حيث نلاحظ أنه في الحالات الثلاثة، يكون عدد الطلبات المخدمة ثابتاً طالما أن عدد المستخدمين الكلي أقل من 1000 مستخدم. لكن و بعد زيادة عدد المستخدمين و ما يرافقه من زيادة عدد الطلبات المرسلة، يجعلنا نلاحظ بأن عدد الطلبات المخدمة يزداد بشكل ملحوظ ابتداء من 1000 مستخدم بما يخص Apache و Varnish حيث يبين الشكل تقارب في الأداء بينهما كما يبين ضعف في أداء HAproxy .



الشكل (3) معدل الطلبات المخدمة في الثانية عند تخديم صفحة ويب ثابتة.

بينما يوضح الشكل (4) معدل الأخطاء عند تخديم الزبائن. يعبر معدل الأخطاء عن عدد الطلبات الفاشلة، أي التي لم يتم تخديمها. نلاحظ معدلاً مرتفعاً للأخطاء في حالة HAproxy و شبه انعدام وجود طلبات فاشلة في الحالتين Apache و Varnish، أي أن أداءهما متقارب.

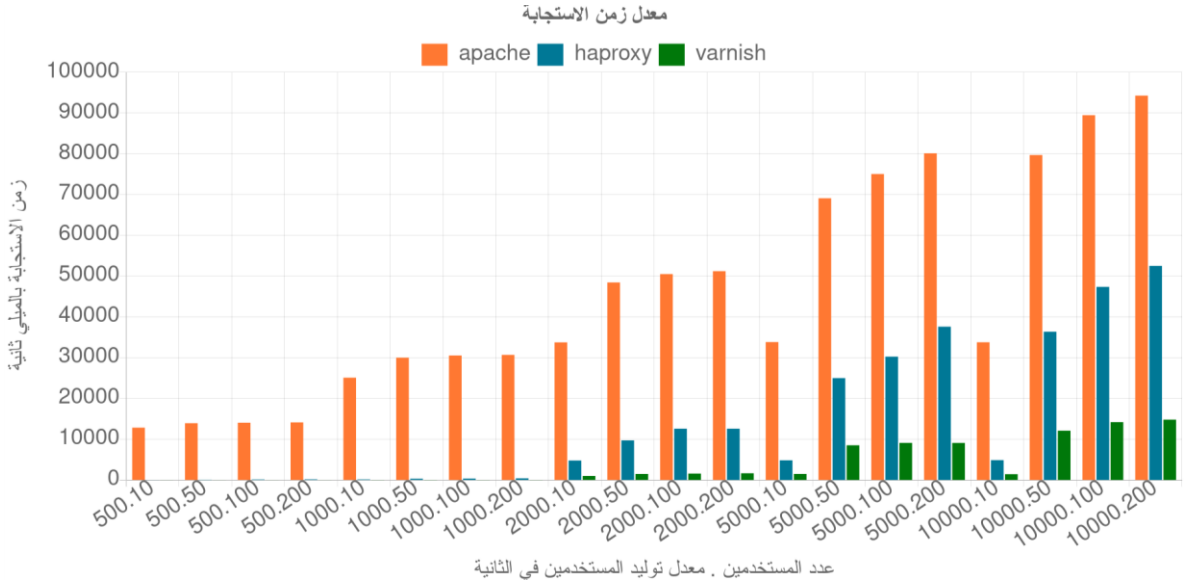


الشكل (4) معدل الأخطاء عند تخديم الزبائن و ذلك عند تخديم صفحة ويب ثابتة.

نلاحظ من الاختبارات الثلاثة السابقة أن أداء HAproxy كان أسوأ بكثير من Apache و Varnish، و ذلك بسبب اعتماده على توليد نيسب جديد لتخديم كل طلب و قيامه أيضاً بإرسال كل طلب لمخدم الويب دون استخدام أي ذاكرة مخبئية (Cache) في حين أن Varnish يستخدم ذاكرة مخبئية للتخديم تجعله يوازي Apache في الأداء.

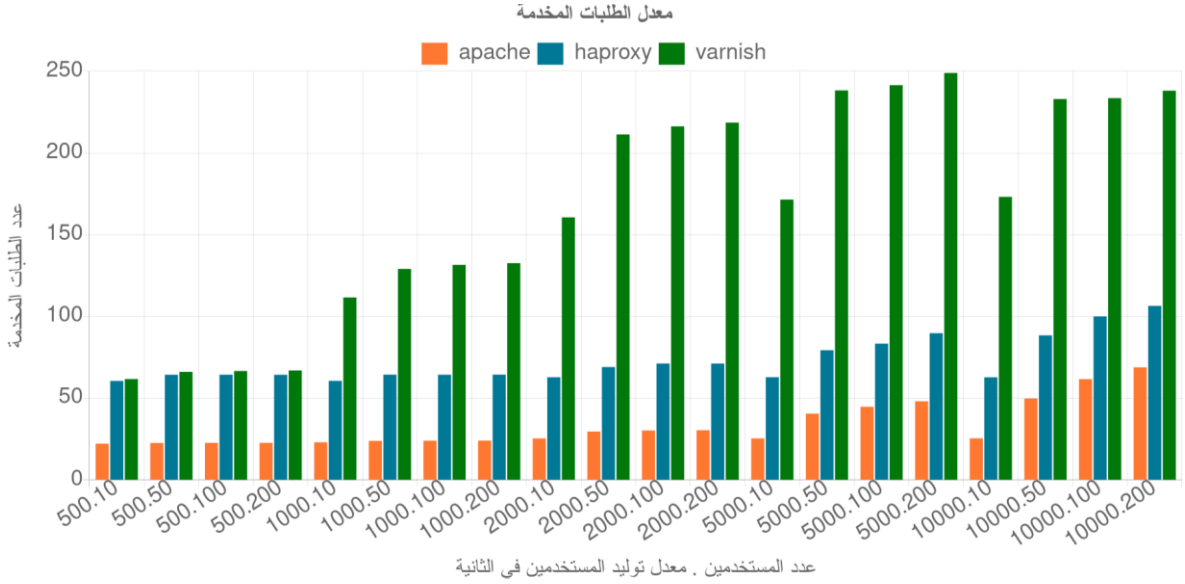
2.7. تقييم الأداء عند تخديم صفحة ديناميكية:

تمت الاختبارات السابقة على صفحة HTML واحدة ثابتة، فيما يلي سنعرض النتائج عند تخديم صفحة ديناميكية تتضمن جزء ديناميكي يتم توليده على المخدم عن طريق تنفيذ كود PHP و جزء آخر ثابت يتم طلبه مباشرة من المخدم. أي سنعرض فيما يلي النتائج التي تتعلق بتخديم موقع ديناميكي يتضمن بعض الأجزاء الثابتة مثل ملفات CSS و JavaScript. يوضح الشكل (5) متوسط زمن الاستجابة.



الشكل (5) متوسط زمن الاستجابة عند تخديم صفحة ديناميكية.

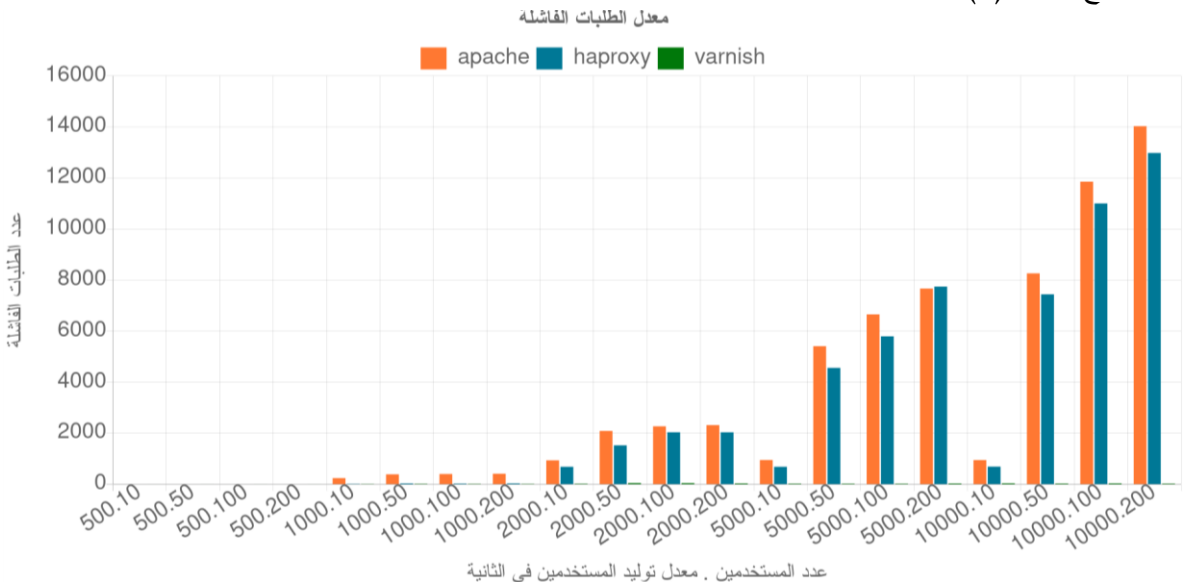
بينما يوضح الشكل (6) معدل الطلبات في الثانية.



الشكل (6) متوسط زمن الاستجابة عند تخديم صفحة ديناميكية.

نلاحظ من الأشكال (5) و (6) تفوق Varnish في متوسط زمن الاستجابة و معدل الطلبات المخدمة في الثانية بشكل كبير على كل من HAproxy و Apache و تراجع Apache بشكل كبير في هذه الناحية، ذلك لأن Apache يعمل على إنشاء عملية جديدة لتخديم كل طلب لصفحة الـ php مما يسبب عبء كبير على المخدم و ضعف في الأداء في حين أن قيام Varnish بعملية التخبئة يحسن الأداء بشكل كبير من جديد.

يوضح الشكل (7) معدل الأخطاء.



الشكل (7) معدل الأخطاء عند تخديم صفحة ديناميكية.

نلاحظ من الشكل (7) تقارب بين أداء كل Apache و HAproxy مع ازدياد معدل الأخطاء لهما بشكل ملحوظ مقارنة Varnish، لكن يجب ألا ننسى أن معدل الطلبات التي خدمها Varnish أكبر بكثير من الطلبات التي خدمها Apache مما يجعل الفرق في الأداء أكبر و أكبر و ذلك بسبب قيامه بتخبئة الاستجابة و استخدامها في الطلبات اللاحقة. يمكننا القول و بثقة بأن Varnish قد تفوق في جميع النتائج.

النتائج والتوصيات المستقبلية

قمنا في هذا البحث بدراسة موازني الحمل HAproxy و Varnish باستخدام خوارزمية الجدولة الدائرية. يتميز Varnish عن HAproxy بإمكانية تخبئة المحتوى إضافةً لعمله كموازن للحمل، لذلك أتت هذه الدراسة من أجل معرفة تأثير التخبئة على أداء النظام فيما يتعلق بزمن الاستجابة، عدد الطلبات المخدمة و معدل الأخطاء. أظهرت النتائج، بما يتعلق بزمن الاستجابة، تحسناً ملحوظاً عند استخدام Varnish مقارنة مع HAproxy حيث انخفض زمن الاستجابة عند استخدام التخبئة بمقدار 83 % بالنسبة للصفحات الثابتة، و بمقدار 89 % بالنسبة للصفحات التي تتضمن محتوى ثابت و ديناميكي، و ذلك عند تحميل النظام في الحالتين السابقتين بأكبر عدد من الزبائن مترافقاً مع أعلى معدل توليد للزبائن. في حين أن الزيادة الأعظمية

لمعدل الطلبات المخدمة الأعظمي قد ارتفعت عند استخدام التخبئة بمقدار 4 أضعاف تقريباً بالنسبة للصفحات الثابتة و 3 أضعاف تقريباً بالنسبة للصفحات المختلطة. ترافق كل ذلك مع تقليل كبير في معدل الأخطاء.

استخدمت التجارب التي قمنا بها عدد قليل من المخدمات بسبب قلة الموارد المتاحة، يمكن إجراء تجارب مستقبلية باستخدام عدد أكبر من المخدمات و بمواصفات مختلفة من أجل مقارنة موازنات حمل مختلفة مع التخبئة و بدونها و تقييم أداء الخوارزميات المختلفة المضمنة عليها.

المراجع

[1] Muhammad Hasnain, Muhammad Fermi Pasha, Imran Ghani, "*Drupal core 8 caching mechanism for scalability improvement of web services*". Software Impacts 3, 2020, 100014.

[2] "haproxy." [Online]. Available: http://www.haproxy.org/. Accessed: Nov. 12, 2020.

[3] "Varnish Cache." [Online]. Available: https://Varnish-Cache.org/. Accessed: Nov. 28, 2020.

[4] Poul-Henning Kamp, "*Think you've mastered the art of server performance? Think again*". ACM Queue, 2010.

[5] R. Li, Y. Li, and W. Li, "*An Integrated Load-balancing Scheduling Algorithm for Nginx-Based Web Application Clusters*". J. Phys. Conf. Ser., vol. 1060, no. 1, 2018.

[6] G. Singh and K. Kaur, "*An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems*". pp. 1950–1955, 2018.

[7] A. B. Prasetijo, E. D. Widianto, and E. T. Hidayatullah, "*Performance comparisons of web server load balancing algorithms on HAproxy and Heartbeat*". 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 393–396, 2016.

[8] G. Hasslinger, M. Kunbaz, F. Hasslinger, and T. Bauschert, "*Web Caching evaluation from Wikipedia request statistics*". 2017 15th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wirel. Networks, WiOpt 2017, pp. 0–5, 2017.

[9] G. Hasslinger, K. Ntougias, F. Hasslinger, and O. Hohlfeld, "*Performance evaluation for new web Caching strategies combining LRU with score based object selection*". Comput. Networks, vol. 125, pp. 172–186, 2017.

[10] V. Sathiyamoorthi, "*A novel Cache replacement policy for Web proxy Caching system using Web usage mining*". Int. J. Inf. Technol. Web Eng., vol. 11, no. 2, pp. 1–13, 2016.

[11] Fatma Mbarek; Volodymyr Mosorov, "*Load balancing algorithms in heterogeneous web cluster*". IEEE, International Interdisciplinary PhD Workshop (IIPhDW), 2018.

[12] Luthfan Hadi Pramono;Robby Cokro Buwono;Yanuar Galih Waskito "*Round-robin Algorithm in HAProxy and Nginx Load Balancing Performance Evaluation: a Review*". International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2018.

[13] S. Shrivastava and P. P. Sb, "*Comprehensive Review of Load Testing Tools*". no. May, pp. 3392–3395, 2020.