مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

Evaluating the performance of the static content serving for the web servers

Dr. Radwan Saleh Dandeh*
Ibraheem Issa Halloum **

(Received 9 / 7 / 2017. Accepted 21 / 12 / 2017)

□ ABSTRACT □

The services provided by the Internet are increasing and the number of users, which they use it, is increasing, and that causes a tremendous pressure on the data centers to provide the servers needed to operate these services. This highlights the need to use all the resources of the server the optimal way to save the expenses and reduce the waste.

The web service is considered as one of the most popular types of web services and that because of the web pages being the most online resource-intensive, and hence the importance of choosing the right operating system and the right web server to provide the services comes.

In this research, the evaluation of three servers (nginx, apache, IIS) is done. Ubuntu operating system, and windows operating system are used to run these servers, and Tests were performed in an Error-free network.

Keywords: Network operating systems, Network stack, Network performance, web servers.

***Professor – computer Systems and network department – information engineering faculty – Tishreen University – Latakia – Syria**
****Postgraduate student – computer Systems and network department – information engineering faculty – Tishreen University – Latakia– Syria**

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

تقييم أداء تخديم المحتوى الثابت لدى مخدمات الويب

الدكتور رضوان صالح دنده*

إبراهيم عيسى حلوم**

(تاريخ الإيداع 9 / 7 / 2017. قُبِل للنشر في 21/ 12 / 2017)

□ ملخّص □

تزداد الخدمات التي تقدمها شبكة الانترنت ويزداد عدد مستخدميها لتسبب ضغطاً هائلاً على مراكز البيانات لتأمين المخدمات اللازمة لتشغيل هذه الخدمات، ومن هنا يبرز الحاجة لاستخدام كامل موارد المخدم بالطريقة الأمثل لتوفير المصاريف وتقليل الهدر.

تعتبر خدمة الويب من أكثر أنواع الخدمات الشبكية انتشاراً وذلك لأن صفحات الويب تعتبر من أكثر الموارد طلباً عبر شبكة الانترنت ومن هنا تأتي أهمية اختيار نظام التشغيل ومخدم الويب المناسبين للتخديم.

في هذا البحث قمنا بتقييم عمل ثلاث مخدمات وهي apache، nginx و Internet Information Services (IIS)، وقد استخدمنا لتشغيل هذه المخدمات نظامي تشغيل Ubuntu Server وServer Windows وتمت الاختبارات في شبكة خالية من الأخطاء.

الكلمات المفتاحية: نظم التشغيل الشبكية، مكدس الشبكة، أداء الشبكة، مخدمات الويب.

* أستاذ – قسم النظم و الشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.

** طالب دراسات عليا (ماجستير) – قسم النظم و الشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.

مقدمة:

تعتبر تكنولوجيا TCP/IP أساس وجود شبكة الانترنت والتي تربط أكثر من 2.4 بيليون مستخدم وهي واحدة من أكبر الشبكات انتشاراً في العالم. مجموعة البرمجيات التي تحقق جميع البروتوكولات التي نحتاجها للتواصل عبر شبكة الانترنت تدعى بمكدس TCP/IP (TCP/IP Stack) [1] أو مكدس بروتوكول الانترنت (internet Protocol stack). في أنظمة التشغيل المتجانسة (monolithic operating systems) مثل جنو/ لينوكس (GNU/Linux)، عائلة BSD وميكروسوفت ويندوز (Microsoft Windows) جميع العمليات الخاصة بالنقل عبر الشبكة تتم في مجال نواة التشغيل.

أهمية البحث وأهدافه:

تعتبر خدمة الويب من أكثر الخدمات الشبكية انتشاراً في شبكة الانترنت كما تعتبر ملفات وصفحات الويب من أكثر أنواع الملفات طلباً عبر شبكة الانترنت، لذلك اعتمدنا على بروتوكول HTTP [2] في دراسة مكدس TCP (TCP stack) وأداءه في أنظمة التشغيل المختلفة (نظام Ubuntu Server 16.04 [3] ونظام Windows Server Standard 2012 R2 [4])، وتأتي أهمية دراسة مكدس TCP من كونه الاساس لأي خدمة شبكية وينعكس أداء عملياتها على أداء الخدمة الشبكية سلباً أو ايجاباً.

تنقسم أهداف هذه الدراسة الى قسمين اثنين:

- الأول تقييم أداء مكدس TCP لدى نظامي التشغيل Ubuntu Server 16.04 و Windows Server Standard 2012 R2 من خلال تقييم أداء مخدم الويب لدى النظامين حيث تم اعتماد IIS [5] مخدم ويب على نظام Windows Server Standard 2012 R2 و مخدمي الويب Apache [6] و Nginx [7] على نظام Ubuntu Server 16.04.
- الثاني تقييم أداء مخدمي الويب Apache و Nginx على نظام Ubuntu Server 16.04.

طرائق البحث ومواده:

1. البروتوكولات:

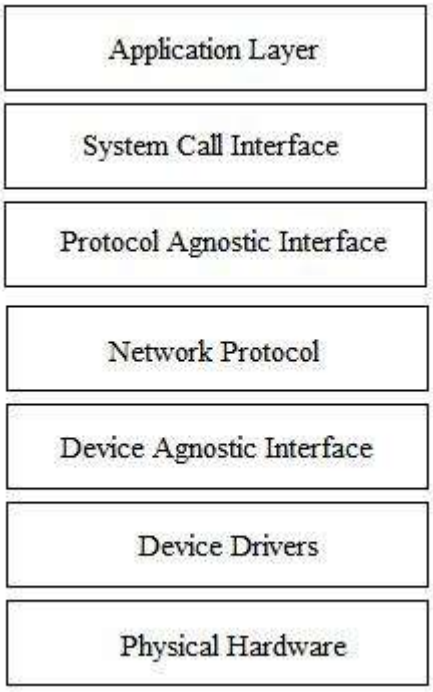
1.1. مكدس TCP:

كل نظام تشغيل شبكي لديه نوع واحد من مكدس الشبكة Network Stack، يسمح للتطبيقات أن تكون قادرة على الوصول إلى الشبكة من خلال أجهزة الشبكة الفيزيائية والتي يمكن أن تكون المودم، أجهزة ISDN، Wi-Fi، أو بطاقات إيثرنت... الخ، بغض النظر عن الوسط الفيزيائي physical medium المستخدم، مكدس الشبكة لا يتغير بتغير الوسط الفيزيائي.

يتألف مكدس الشبكة من سبع طبقات لكل منها وظيفة محددة لتمكن نظام التشغيل من الاتصال عبر الشبكة كما يظهر الشكل (1) [8]، ويتحدد عمل المكدس بطريقتين أساسيتين: الأولى معالجة البيانات لإرسالها عبر الشبكة والثانية استقبال البيانات من الشبكة ومعالجتها، تصنف هذه الطبقات ضمن ثلاث مجموعات: فضاء المستخدم User Space، فضاء النواة Kernel Space، الطبقة الفيزيائية Physical Layer، كالتالي:

- الطبقة العليا (طبقة التطبيقات) هي جزء من فضاء المستخدم.

• الطبقات الخمسة التالية هي جزء من فضاء النواة.
• الطبقة الأخيرة (العتاد الفيزيائي) هي الطبقة الفيزيائية.



الشكل 1 : طبقات مكدس الشبكة

طبقة التطبيقات Application Layer تمثل فضاء المستخدم حيث توجد التطبيقات التي يتعامل معها المستخدم مباشرة مثل متصفح الانترنت برامج المحادثة ..... الخ.

الطبقة الثانية واجهة استدعاءات النظام System Call Interface والتي يتم فيها انشاء الاستدعاءات من طبقة التطبيقات الى النواة.

طبقة Protocol Agnostic Interface وهي الطبقة التي يتم فيها انشاء Socket تستخدم في عملية الاتصال.

طبقة Network Protocol المسؤولة عن كيفية ارسال البيانات واستقبالها وتتحكم هذه الطبقة بالمعلومات التي تمكن البينات من عبور الشبكة.

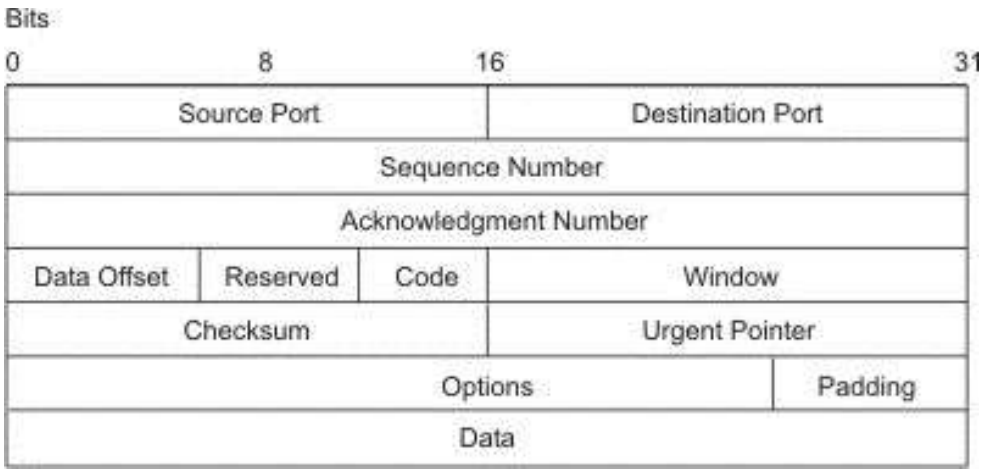
طبقة Device Agnostic Interface والتي تقوم بإرسال البيانات من نواة نظام التشغيل إلى العتاد الصلب لأجهزة الشبكة.

طبقة Device Drivers والتي تحتوي على توابع واجراءات تمكننا من التعامل مع العتاد الصلب لجهاز الشبكة الذي سيتم ارسال البيانات عبره.

طبقة Physical Hardware وهي العتاد الصلب لجهاز الشبكة التي يتم فعليا ارسال واستقبال الرزم من خلاله.

يوضح الشكل (2) ترويسة بروتوكول TCP ويظهر حقل window الذي يحدد حجم النافذة عند عملية تأسيس الاتصال [9].

و في محاولة لتحسين أداء عمل المكدس أنجز الباحثون [10] Erik Nordmark, Sunay Tripathi, Nicolas G. Droux. براءة اختراع لصالحة شركة Google بعنوان SHARED AND SEPARATE NETWORK STACK INSTANCES و سجلت بوثائق براءات الاختراع الأمريكية عام 2014 و اعتمدت فكرة الاختراع على انشاء مكدس شبكة وهمي مشترك بين عدة اجهزة شبكة وهمية مما يمكن نفس الجهاز الموجود على نفس الشبكة باستخدام خيارات مختلفة في طبقة النقل و الشبكة، فوجود عدة مكدسات شبكة وهمية و عدة أجهزة شبكة وهمية سيضيف امكانيات اكثر للجهاز للتواصل عبر الشبكة حيث يتم التفريق بين كل مكدس من خلال معلومات مميزة له تحتويها كل الرزم الخاصة بهذا المكدس كما تمكن الباحثون ما استخدام مكدس شبكة واحد لأكثر من جهاز شبكة وهمية و ذلك يمكن من ان يكون لمكدس الشبكة اكثر من إعدادات خاصة بطبقة الشبكة و طبقة النقل .



الشكل 2 : ترويسة بروتوكول TCP

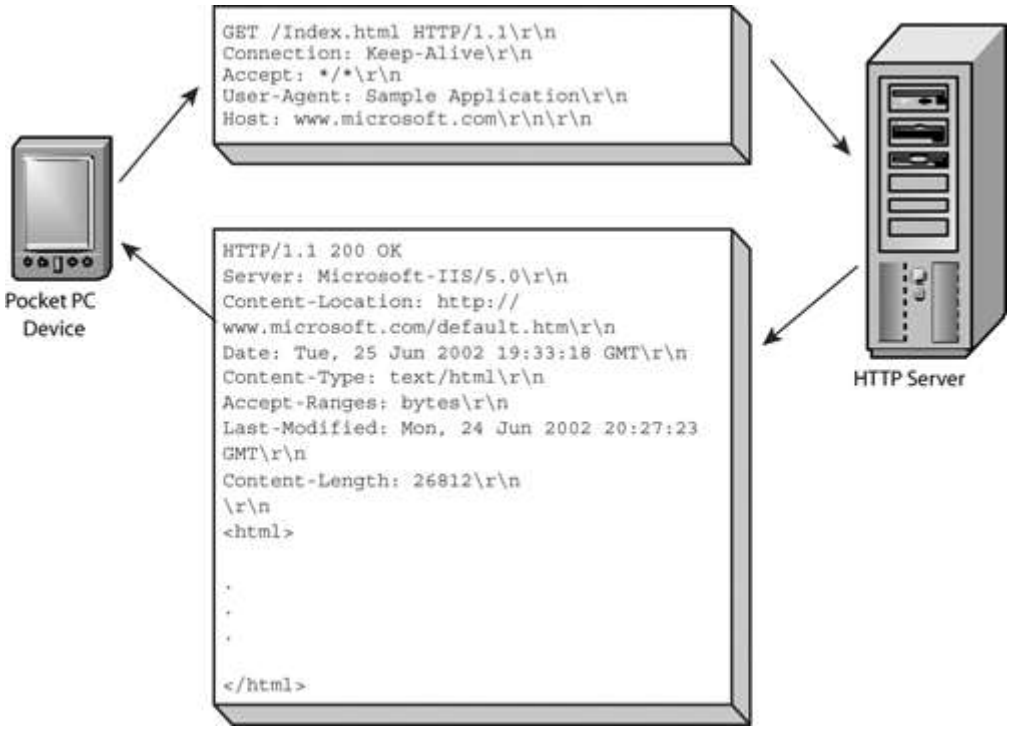
1.2. بروتوكول HTTP:

تعتمد شبكة الإنترنت في عملها على مجموعة من بروتوكولات قياسية تستخدمها الحواسيب لتبادل البيانات. ولم يكن ممكناً لشبكة الإنترنت أو لأي شبكة أخرى أن تنشأ من دون هذه البروتوكولات، وعلى الرغم من أن الشبكة تسمح بالعديد من البروتوكولات، إلا أنها تحتوي على بروتوكول أساسي يسمى بروتوكول نقل النص المتشعب HTTP Hypertext Transfer Protocol هذه هي اللغة المتعارف عليها لنقل النصوص والصور بين جميع أجهزة الحاسب على الإنترنت. فعندما يبدأ عنوان موقع بكلمة HTTP يعني ذلك أنك تطلب من جهازك أن يقوم بإحضار صفحة إنترنت.

لقد بدأ استخدام برتوكول HTTP في الشبكة منذ عام 1990، والإصدار الأول (HTTP/0.9) كان إصداراً بسيطاً ولكن الإصدارات التي تلتهُ (HTTP/1.0)و (HTTP/1.1) أصبحت أكثر تعقيداً [11] .

لقد تم تصميم البروتوكول HTTP لتأمين عملية نقل ملفات HTML إلى العميل الذي طلبها، وتمثل HTML لغة تقوم بعملية إنشاء الملف المتشعب Hypertext، حيث إن الملف المتشعب Hypertext document يحوي ارتباطات إلى ملفات أخرى تحتوي على معلومات إضافية عن الموضوع. وبالتالي عندما يستقبل مستعرض الشبكة ملف

HTML، فإن هذا الملف يترجم ويحلل، فإذا كان يحوي ارتباطات أخرى فإنها تطلب من جديد. وعند الحصول عليها تتم عملية تنسيق لكامل الملف للحصول في النهاية على صفحة الإنترنت المتعارف عليها، ومن الممكن أن تقود هذه الارتباطات إلى ملفات صوتية أو صور أو مقاطع فيديو أو غيرها.



الشكل 3 : مثال عن طلب و استجابة HTTP

إن البروتوكول HTTP مبني على أساس طلب واستجابة (Request–Response) كما هو موضح في الشكل (3) [11]، حيث يطلب العميل البيانات من الخادم في صورة رسالة تسمى هذه العملية طلب (Request) ثم يفسر الخادم هذه الرسالة ويرد على العميل برسالة وتسمى هذه العملية استجابة (Response) حيث يقوم العميل بتأسيس وصلة مع الخادم ويرسل الطلب ضمن هذه الوصلة، وبدوره يقوم الخادم بالاستجابة لهذا الطلب بسطر الحالة Status line والذي يحوي نسخة البروتوكول HTTP المستخدمة ورمز لفشل أو نجاح العملية ومن ثم بعض الترويسات عن الخادم والاستجابة ومن ثم محتوى الرسالة.

يمكننا تقسيم عمل البروتوكول HTTP إلى أربع مراحل وهي:

- إنشاء الوصلة (Connection) مع الخادم.
- إرسال الطلب إلى الخادم.
- استلام الاستجابة من الخادم.
- إغلاق الوصلة.

يعد برتوكول HTTP من الطبقة الخامسة وهي طبقة التطبيقات. يعتمد برتوكول HTTP على برتوكول TCP/IP في إرسال البيانات والمنفذ الرئيسي هو 80 كما يمكن استخدام منافذ أخرى. كما أن البروتوكول HTTP في النسخة (HTTP/1.0) هو بروتوكول عديم الحالة (Stateless)، ويعني ذلك أنه لا يحافظ على حالة الوصلة، فعلى

فرض أنه تم طلب صفحة ويب تحتوي على صورتين فإنه عملياً يتم إنشاء ثلاثة وصلات (Connections) الأول من أجل صفحة HTML والثاني من أجل الصورة الأولى والثالث من أجل الصورة الثانية، حيث تأتي الارتباطات التي تدل على تلك الصور من خلال ترجمة نص HTML في مستعرض الشبكة المستخدم عند العميل، أما في النسخة (HTTP/1.1) تتضمن نوعاً من الربط يسمى الوصلة الدائمة(Persistent Connection) فتستخدم الوصلة نفسها لأكثر من طلب في وقت واحد، ولقد حققت الوصلة الدائمة العديد من الفوائد ومن أهمها الحد من عدد الوصلات المنشأة على البروتوكول TCP.

يحتوي الطلب في بروتوكول HTTP على عدة طرق يشار لها بطرق البروتوكول (HTTP Methods)، ومنها:

GET طلب عرض، وهو أكثر وسيلة مستخدمة اليوم على الشبكة.

HEAD تطلب رداً مطابقاً لذلك الذي يرجعه طلب GET، ويفيد هذا في الحصول على معلومات عن المطلوب دون نقل على كامل المحتوى.

POST تستخدم لإرسال معلومات من العميل إلى الخادم، حيث ترسل هذه المعلومات ضمن جسم الرسالة (body).

PUT لتحميل المعطيات إلى الخادم.

DELETE تطلب من الخادم حذف المصدر المحدد.

TRACE تسمح للعميل بإمكانية معرفة عدد المرات التي طلبت فيها هذه الرسالة من قبل الخادم.

OPTION تستخدم لمعرفة الميزات التي يتمتع بها خادم الشبكة.

2. مخدمات الويب:

يعتبر مخدما الويب Apache وNginx الأكثر شهرةً من بين الخوادم المفتوحة المصدر في عالم الشبكة العنكبوتية، على اعتبار أنهما مسؤولان عن خدمة وتأمين نصف تدفق البيانات على الإنترنت، وعلى مقدرة على تولي مختلف حجوم الأحمال، والعمل مع برمجيات أخرى في سبيل توفير حزمة من خدمات الويب الشاملة والكاملة لتلبية مختلف الاحتياجات.

بينما يَتشارك المخدمان Apache وNginx العديد من الميزات والخاصيات، فلا يمكن اعتبارهما متشابهان، أو التفكير بأن أحدهما هو بديل عن الآخر، فكل منهما يتفوق عن الآخر بشيءٍ ما، وعلى مدير النظام فهم وإدراك لماذا يجب اختيار أحدهما دون الآخر، فهذا الدليل يهدف إلى مناقشة كيف أن كل مخدم يتميّز ويبرز في شيء ويخفق في آخر[12] .

2.1 مخدم Apache:

أَنشأَ "روبت ماك كول" (Robert McCool) "مخدم HTTP أباتشي" (Apache HTTP Server) في عام 1995، وتم متابعة تطويره تحت مظلة "منشأةُ برمجيات أباتشي" (Apache Software Foundation) منذ العام 1999، وكان هذا المخدم هو المشروع الأساسي للمنشأة والأكثر شهرةً عن باقي البرمجيات، فأصبح ببساطة يشار إليه بالاسم "Apache".

يعتبر مخدم الويب أباتشي الخادم الأكثر استخدامًا على الإنترنت منذ العام 1996، وبسبب هذه الشهرة، استفاد أباتشي من توثيق ودعم باقي مشاريع البرمجيات الأُخرى.

يختار مدراء الأنظمة الخادم أباتشي غالبًا بسبب مرونته، وقدرته على التحمل، وتوفر دعمه العالي والمنتشر، كما يحسب له قابليته على التوسع عبر نظام الوحدات (modules) الديناميكية، واستطاعته على معالجة عدد كبير من اللغات المفسرة (interpreted languages) من دون الحاجة إلى برمجية مستقلة وسيطة.

2.2. مخدم Nginx:

بدأ "ايجور سيسيوف" في عام 2002 العمل Nginx للإجابة وحل المشكلة المعروفة بالاسم C10K، والتي كانت تشكل تحدياً كبيراً لمخدمات الويب لتصبح قادرة على تولي عشرة آلاف اتصال متزامن (concurrent) وذلك في تلبية حاجة الويب الحديث، وقد تم إنتاج الإصدار الأولي والمتاح للعموم في عام 2004 مقدمة حلاً لهذه المُشكلة، وذلك بالاعتماد على معمارية مدفوعة بالحالة (event-driven) ولا متزامنة.

انتشر Nginx بشكل كبير منذ إصداره، بمقتضى خفته واستخدامه الخفيف للموارد، وقدرته على التوسع وتحمل الضغط العالي وبأقل عتاد مُمكن، وتفوّق Nginx بتأمين المحتوى الثّابت (static content) بسرعة، وبتصميمه لتمرير الطلبات الديناميكيّة (المتغيرة) إلى برمجيات أُخرى قادرة على معالجة هذا النوع من المحتوى.

يختار مدراء الأنظمة الخادم Nginx غالبًا بسبب استهلاكه الأمثل للموارد، واستجابته العالية مع الضغط المتزايد[13] .

النتائج والمناقشة:

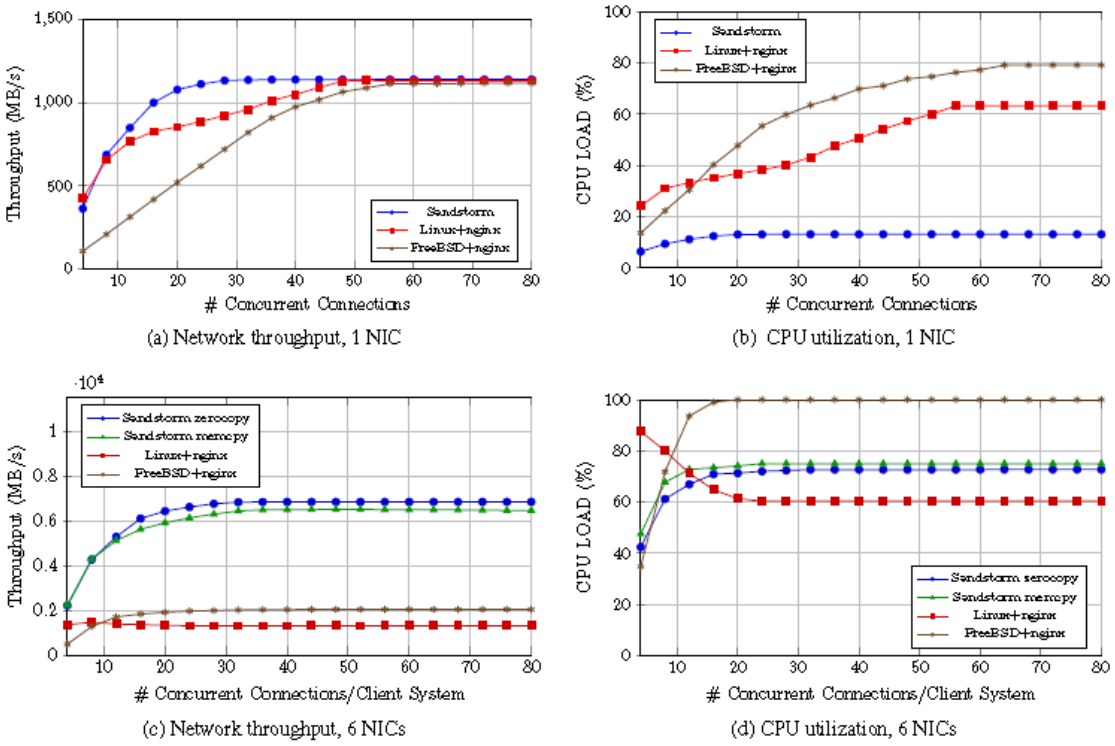
تضمنت الدراسة العملية تقييم أداء مخدمات الويب IIS، Apache، Nginx وتأثير اختلاف مكدس TCP وذلك ضمن شبكة مخبر خالية من الأخطاء.

و في دراسة مشابهة قام الباحثون [14] Kenichi Yasukata, Michio Honda, Douglas Santry, and Lars Eggert من جامعة Keio University بنشر ورقة بحثية بعنوان StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs في مؤتمر USENIX Annual Technical عام 2016 و قد قدموا فيها StackMap و التي تتفوق على أفضل أداء لشبكة kernel-bypass networking من خلال خدمات لينوكس الشبكية الجديدة ذات تأخير قليل و التي تعتمد على تحقيق مكدس شبكة بمواصفات عالية، من خلال تكريس واجهات الشبكة (network interfaces) للتطبيقات و تقديم نسخة موسعة من netmap API الداعمة لخاصية zero-copy، و تحفيف ضغط على مسار البينات (data path) مع الاحتفاظ بالـ Socket API من مسار التحكم (control path) .

من أجل الرسائل الصغيرة وأعباء العمل (transactional workloads) تتفوق StackMap على لينوكس بنسبة من 4% الى 80% في التأخير وبنسبة من 4% الى 39% في الانتاجية (throughput). كما تحقق أداء مشابه مع Seastar وهو عبارة عن مكدس TCP/IP يعمل في فضاء المستخدم بالاعتماد على Data Plane Development Kit (DPDK).

كما قام الباحثون [15] Ilias Marinos , Robert N.M. Watson , Mark Handley من جامعة كامبردج و لندن بنشر ورقة بحثية بعنوان Network Stack Specialization for Performance في مؤتمر Sigcpmm 2014 حول العمل على تطوير و تخصيص مكدس الشبكة بدلا من استخدام المكدس العام، طرحت هذه الورقة البحثية مخدم ويب بمكدس شبكة مخصص يعمل في فضاء المستخدم مع بعض التعديلات في فضاء النواة

بهدف زيادة أداء الشبكة وتخفيف الحمل على المعالج من خلال تقليل عمليات النسخ المستخدمة بين الفضاء المستخدم و النواة، وتضمن القسم العملي تصميم مخدم الويب بالاعتماد على نظام frssBSD واستخدم netmap التي توفر إمكانية الوصول إلى جهاز الشبكة بشكل مباشر و التعامل معه من خلال فضاء المستخدم، و اعتمدوا بشكل أساسي على تقليل عمليات النسخ بين فضاء المستخدم و النواة واعادة تصميم مكدس البروتكول لتحسين الأداء، أما Metrics فكانت الحمل على المعالج و أداء الشبكة و زمن الوصول في الشبكة و يوضح الشكل (4) تقيم أداء المخدم .



الشكل 4 : مقارنة أداء Sandtorm مع Nginx

1. خطة العمل:

- الأدوات المستخدمة:
  - ✓ Apache Jmeter 3.2 [16] يتم تشغيله على نظام Windows 7 SP1 ultimate
- البيئة:
  - ✓ نظام التشغيل:
    - Xenserver 7.1
  - ✓ أنظمة التشغيل الافتراضية المستخدمة:
    - Ubuntu Server 16.04.2 amd64
    - Windows Server Standard 2012 R2
  - ✓ مخدمات الويب:
    - IIS 7 يتم تشغيله على مخدم بنظام التشغيل Windows Server Standard 2012 R2.
    - Apache يتم تشغيله على مخدم بنظام التشغيل Ubuntu Server 16.04.2 amd64.
    - Nginx يتم تشغيله على مخدم بنظام التشغيل Ubuntu Server 16.04.2 amd64.
- العتاد الصلب:
  - ✓ المخدم:
    - CPU : intel core i7 2,2 GHz
    - RAM : 6GB
  - ✓ الزبون:
    - CPU : intel core i7 2,2 GHz
    - RAM : 6GB
- الشبكة:
  - ✓ Fast Ethernet.
  - ✓ شبكة مخبر خالية من الأخطاء.

- آلية التقييم:

يتضمن هذا السيناريو جهازين اثنين (مخدم وزبون) متصلين ببعضهم عبر شبكة Fast Ethernet:

– جهاز الزبون يعمل بنظام تشغيل Windows 7 SP1 ultimate، ويستخدم لتشغيل برنامج Jmeter لتقييم أداء المخدمات موضح في الشكل (5).

الشكل 5 : هيكلية الزبون

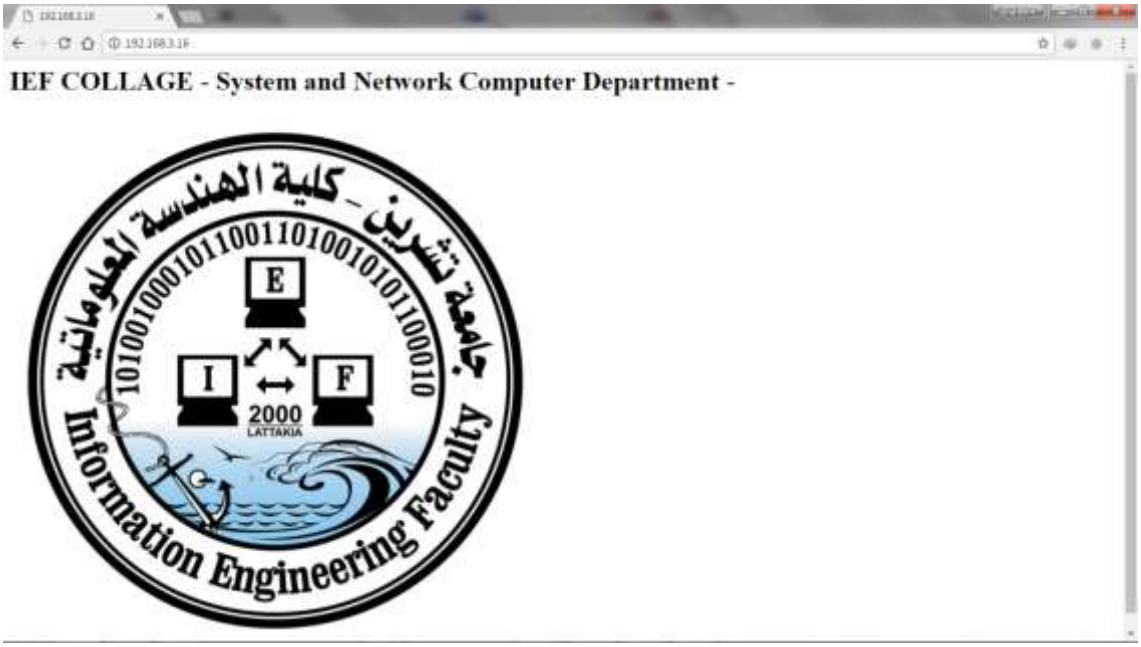
– المخدم يعمل بنظام تشغيل xenserver الذي يتيح لنا تشغيل عدة مخدمات افتراضية كما هو موضح في الشكل (6) وهي كالتالي:

الشكل 6 : هيكلية المخدم

- مخدم IIS يعمل بنظام تشغيل Windows Server 2012 ويشغل مخدم الويب IIS.
- مخدم Nginx يعمل بنظام تشغيل Ubuntu Server 16.04.2 ويشغل مخدم الويب Nginx.
- مخدم Apache يعمل بنظام تشغيل Ubuntu Server 16.04.2 ويشغل مخدم الويب Apache.

2. تنفيذ السيناريو:

تم انشاء صفحة HTML الموضحة في الشكل (7) تضمن عبارة نصية وشعار كلية الهندسة المعلوماتية في جامعة تشرين ورفعت نفس الصفحة على المخدمات الثلاث وهي التي سيتم طلبها في عملية الاختبار.



الشكل 7 : صفحة الاختبار

تضمن السيناريو سبعة اختبارات:

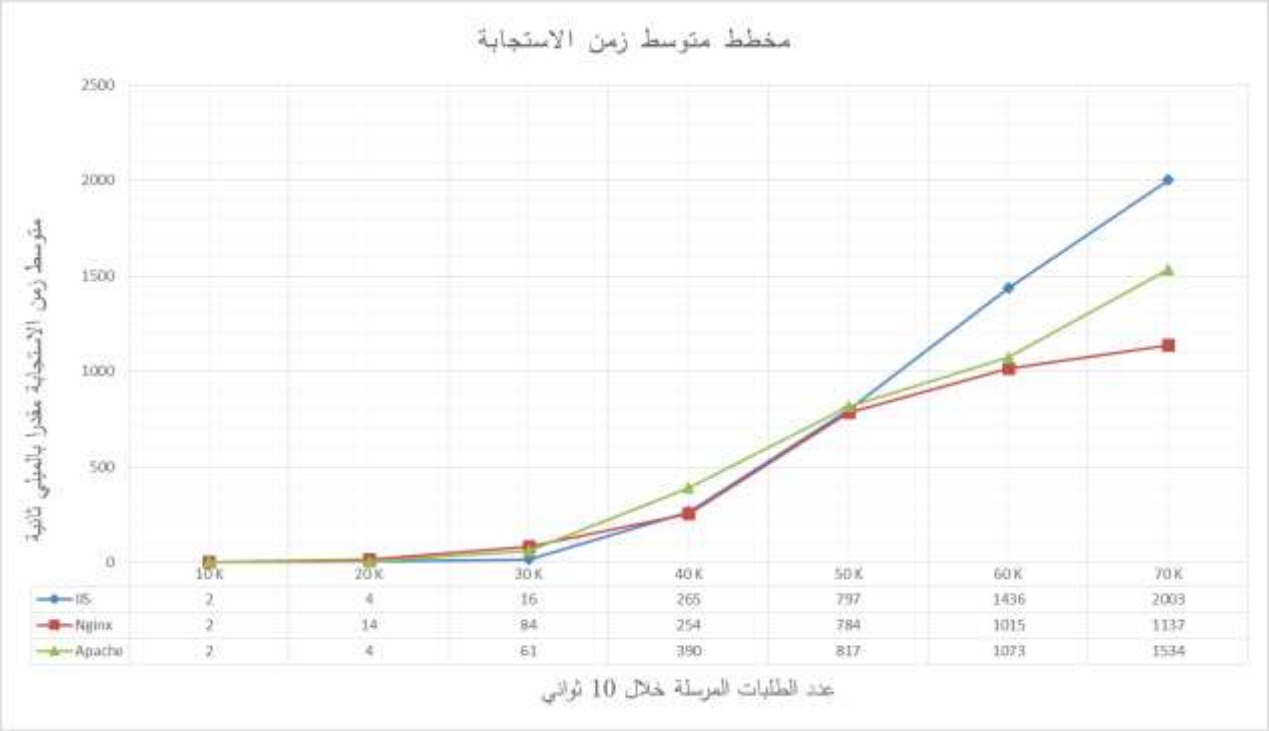
- الاختبار الأول تضمن توليد 10 ألاف طلب HTTP.
- الاختبار الثاني تضمن توليد 20 ألف طلب HTTP.
- الاختبار الثالث تضمن توليد 30 ألف طلب HTTP.
- الاختبار الرابع تضمن توليد 40 ألف طلب HTTP.
- الاختبار الخامس تضمن توليد 50 ألف طلب HTTP.
- الاختبار السادس تضمن توليد 60 ألف طلب HTTP.
- الاختبار السابع تضمن توليد 70 ألف طلب HTTP.

هذه الاختبارات السبعة نفذت على المخدمات الثلاث وفي كل اختبار تم حساب المقاييس التالية (زمن الاستجابة Response Time، نسبة الأخطاء، الانحراف المعياري) تم الاعتماد عليها في عملية المقارنة.

2.1. نتائج السيناريو:

- متوسط زمن الاستجابة:

يقصد فيه الزمن الفاصل بين ارسال طلب HTTP واستقبال الاستجابة ويقدر بالميلي ثانية.



الشكل 8 : مخطط متوسط زمن الاستجابة

نلاحظ من المخطط متوسط زمن الاستجابة الشكل (8)، في الاختبارات الثلاث الاولى كان أداء IIS أفضل بكثير لكن ما لبث أن تدهور الأداء ليصبح الفارق كبير في الاختبار السابع بين IIS ومخدمي الويب الأخرين، يعود سبب هذا التغير الى حجم نافذة الازدحام (congestion window) في مكدس TCP، حيث يبدأ حجم النافذة في مكدس TCP لدى أنظمة ويندوز من 64 KB، أما الحجم الاعظمي فهو 16 MB بينما يبدأ حجم النافذة لدى أنظمة لينوكس من 6 KB فقط والحجم الأعظمي 16 MB.

يؤثر حجم النافذة على مقدار البيانات المرسلة عبر TCP قبل انتظار استلام ACK وبالتالي الويندوز بدأ بقيمة نافذة كبيرة سمحت له خلال وجود عدد قليل من الطلبات بالتخديم بشكل أفضل، لكن مع ازدياد عدد الطلبات وحصول ضغط على الشبكة وحصول الأخطاء سبّب ضعف بالأداء نتيجة الحجم الكبير للنافذة، لأن عدم وصول اشعار الوصول او ضياع احدى الرزم المرسلة سيتعين إعادة ارسال من جديد، وبسبب الحجم الكبير للنافذة سوف يتم ارسال بيانات بحجم أكبر من حجم البيانات المرسلة عندما يكون حجم النافذة صغير.

تبدأ أنظمة لينوكس بحجم نافذة 6 KB وبالتالي تحتاج عملية نقل ملف ما لعدد دورات أكثر ولكن في حال ضياع احدى الرزم وضياع إشعار(ACK) سيتم اعادة ارسال حجم أقل لذلك عند حصول ضغط على المخدم وضياع للبيانات سوف يظهر بأداء أفضل.

بالنسبة للمقارنة بين مخدم الويب Apache ومخدم الويب Nginx نلاحظ في البداية تفوق في الاداء لصالح Apache لكن عند ازدياد عدد الطلبات المرسلة أظهر Nginxأداء أفضل وأكثر استقراراً، يعود هذا السبب لوحدة معالجة الاتصال أو الطلبات الخاصة بمخدم الويب حيث المخدم Apache يستخدم بشكل افتراضي mpm_prefork كما هو موضح بالشكل (9).

```
Server version: Apache/2.4.18 (Ubuntu)
Server built:   2017-05-05T16:32:00
Server's Module Magic Number: 20120211:52
Server loaded:  APR 1.5.2, APR-UTIL 1.5.4
Compiled using: APR 1.5.2, APR-UTIL 1.5.4
Architecture:   64-bit
Server MPM:     prefork
  threaded:     no
    forked:     yes (variable process count)
Server compiled with....
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D DYNAMIC_MODULE_LIMIT=256
 -D HTTPD_ROOT="/etc/apache2"
 -D SUEXEC_BIN="/usr/lib/apache2/suexec"
 -D DEFAULT_PIDLOG="/var/run/apache2.pid"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_ERRORLOG="logs/error_log"
 -D AP_TYPES_CONFIG_FILE="mime.types"
 -D SERVER_CONFIG_FILE="apache2.conf"
```

الشكل 9 : اعدادات MPM في Apache

هذه الوحدة او الموديول (module) تستنسخ وتولد عمليات (processes) باستخدام سلسلة وحيدة (single thread)، على أن تكون كل عملية لمعالجة طلب، ويستطيع كل ابن معالجة اتصال واحد في نفس الوقت، وطالما أن عدد الطلبات أقل من عدد العمليات، فستكون هذه الوحدة سريعة جداً، ولكن ينخفض الأداء بسرعة بعد تخطي الطلبات عدد العمليات، ولذلك لا يعتبر هذا النمط بالخيار الجيد للعديد من السيناريوهات، فكل عملية لها تأثير بليغ على استهلاك الذاكرة (RAM)، ولهذا السبب تصعّب هذه الوحدة من عملية التوسع بطريقة ملائمة، ومع هذا تبقى هذه الوحدة خيارًا جيدًا إن تم استخدامها مقترنةً مع مقومات (components) أُخرى لم تبنى بالأساس مع الأخذ بعين الاعتبار السلاسل (threads)، فلغة PHP ليست سلسلة آمنة (thread–safe)، ولذلك ينصح بهذه الوحدة باعتبارها الطريقة الوحيدة الآمنة للعمل مع mod_php والتي هي وحدة من وحدات أباتشي لمعالجة هذا النوع من الملفات.

أتى المخدم Nginx إلى عالم خوادم الويب بعد قدوم أباتشي، مبنيًّا ومعدًّا لمشاكل الاتصالات المتزامنة (concurrency) التي تواجه المواقع عند التوسع ومحملاً بهذه المعرفة صمم Nginx من جذوره ليستخدم خوارزمية معالجة اتصالات مدفوعة بالحالة (event–driven).

يستنسخ ويولد Nginx من العمليات العاملة (worker processes)، ويستطيع كل منها معالجة آلاف الاتصالات، وتنجز العمليّات العاملة ذلك عن طريق تنفيذ آلية حلقيّة سريعة، والتي تفحص بشكل مستمر حالات العمليّات (processes events)، وإن فصل المهمة الفعلية عن الاتصالات يسمح لكل عامل بالاهتمام بنفسه باتصال فقط عند بدء حالة جديدة.

يسمح هذا الأسلوب من معالجة الاتصال Nginx بتحمل الضغط العالي بشكل لا يصدّق وبأقل المَوارد المتاحة، وباعتبار أنّ المخدم ذو سلسلة وحيدة (single–threaded) ولا تتولد العمليات لتعالج كل اتصال جديد، فإن استهلاك المعالج (CPU)، والذاكرة يبقى ثابتا نسبياً، حتى في أوقات الذروة.

- الأخطاء:

نلاحظ من مخطط نسبة الأخطاء الشكل (10) كيف تتزايد نسبة الأخطاء عند مخدم الويب IIS مع ازدياد عدد الطلبات المرسلة للمخدم بشكل كبير عن مخدمي الويب Apache، Nginx، من أحد اسباب هذا الفرق بين المخدمات هو الاختلاف السابق الذي تحدثنا عنه بين مكدس TCP لدى أنظمة لينوكس ومكدس TCP لدى أنظمة مايكروسوفت ويندوز، بسبب اختلاف حجم النافذة، الازدحام يبدأ تأثيره عند وجود ضغط بالشبكة وحصول بعض الأخطاء.

فبمقارنة مخطط نسبة الأخطاء مع مخطط متوسط زمن الاستجابة نلاحظ تأثير الأخطاء العكسي على متوسط زمن الاستجابة ففي المرحلة التي بدأ فيها ظهور الأخطاء بدأ متوسط زمن الاستجابة بالتزايد ومع ازدياد الاخطاء ازداد متوسط زمن الاستجابة.

الشكل 10 : مخطط نسبة الأخطاء

من جانب أخر نلاحظ تفوق مخدم الويب Apache على مخدم الويب Nginx بنسبة الأخطاء لكن بفارق بسيط لا يذكر أمام تفوق Nginx بمتوسط زمن الاستجابة.

- الانحراف المعياري:

الانحراف المعياري (Standard deviation) [17] هو أحد أهم المقاييس الإحصائية التي تسمى بـ (مقاييس التشتت)، ويعرف علماء الإحصاء مقاييس التشتت بأنها: المقاييس التي تستخدم في قياس اختلاف مجموعة من البيانات أو تشتتها، وهذه المقاييس مكملة ومتممة لمقاييس النزعة المركزية التي تستخدم في إعطاء القيمة العددية التي تتجمع وتتركز حولها أكثر القيم والمشاهدات، حيث إن مقاييس النزعة المركزية لوحدها غير كافية دائما لإعطاء تصور واضح وكامل عن البيانات التي يتم تطبيقها عليها, ولذلك يستعمل الاحصائيون مقاييس التشتت إلى جانب مقاييس النزعة المركزية، فمقاييس النزعة المركزية تعطي القيمة الوسطية فقط، أما درجة تباعد البيانات وتشتتها حول هذه القيمة، فإن حسابها يتم عن طريق مقاييس التشتت.

الشكل 11 : مخطط الانحراف المعياري

إن المتوسط الحسابي لزمن الاستجابة لا يكفي لتحديد المخدم الويب الأفضل بشكل عام فالمتوسط الحسابي يخبرنا بالقيمة المتوسطة لزمن الاستجابة في حين ان الانحراف المعياري يخبرنا مدى بعد قيم العينة عن المتوسط، فعند اقتراب قيمة المتوسط من بعضها هنا يبرز دور الانحراف المعياري بتحديد الأفضل.

إذا قمنا بمقارنة قيمة الانحراف المعياري الموضح في الشكل (11) للمخدمات الثلاث في آخر ثلاثة اختبارات اي عند ارسال 50K و60 K و70 K طلب، نلاحظ نسبة التزايد في الانحراف المعياري عالية جداً لدى apache وIIS وقليلة عند Nginx على الرغم من تفوق IIS بقيمة الانحراف المعياري على Nginx، ولكن تفوق Nginx في

قيمة متوسط زمن الاستجابة وفي نسبة التزايد القليل للانحراف المعياري عند ارتفاع عدد الطلبات المرسلة يمنحه استقراراً أكثر في الأداء.

الاستنتاجات والتوصيات:

من خلال الاختبارات السابقة تبين صعوبة موضوع المقارنة بين المخدمات الثلاث، حيث يختلف الأداء تبعاً لظروف و هدف الاختبار، علماً أننا استخدمنا الإعدادات الافتراضية التي تأتي مع كل مخدم ويب و لم نقم بإجراء أي تعديل و ذلك لإنصاف المقارنة ففي حال تحميل أي موديل او ميزة للمخدمات يمكن أن يأثر على أداء التخديم إضافة لكون المقارنات تعتمد على تخديم المحتوى الثابت و لا يوجد هناك داعي للقيام باي اعدادات من اجل تخديم المحتوى الثابت حيث أن جميع المخدمات تقدم هذه الخدمة بالإعدادات الافتراضية ، و بالتالي وجدنا كيف أن مخدم الويب Nginx مناسب جداً لتحمل الضغط الكبير حيث أظهر استقراراً واضحاً من حيث متوسط زمن الاستجابة و الانحراف المعياري، في حين قدم Apache أداء رائعا مع الضغط المتوسط و حافظ على معدل أخطاء منخفض متفوقاً على المخدمين الآخرين.

في الدراسات القادمة من الممكن تقييم مخدمات الويب السابقة عن طريق طلب محتوى ديناميكي وليس محتوى ثابت فقط، اضافة للقيام بالاختبارات في ظروف شبكة تحتوي على أخطاء وذلك من خلال استخدام أدوات تقوم بالتنصت على الحزم المرسلة عبر الشبكة ورفض عدد من الرزم مولدة بذلك أخطاء لنحاكي بذلك ظروف الشبكة الواقعية.

المراجع:

[1] 30 June. 2017. <http://www.cubrid.org/blog/understanding-tcp-ip-network-stack>

[2] 29 June. 2017. <https://www.w3.org/Protocols/rfc2616/rfc2616.html>

[3] 1 July. 2017. <https://www.ubuntu.com/download/server>

[4] 2 July. 2017. <https://www.microsoft.com/en-us/licensing/product-licensing/windows-server-2012-r2.aspx>

[5] 1 July. 2017. <https://www.iis.net/>

[6] 2 July. 2017. <https://httpd.apache.org/>

[7] 3 July. 2017. <https://nginx.org/en/>

[8] 10 October 2017. < https://www.linux.org/threads/linux-network-stack.9065/ >

[9] 10 October 2017. <https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970>.

[10] Erik Nordmark, Sunay Tripathi, Nicolas G. Droux: SHARED AND SEPARATE NETWORK STACK INSTANCES 2014.

[11] 10 October 2017. <https://hpbn.co/brief-history-of-http/>.

[12] 1 July. 2017. <https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations>

[13] 2 July. 2017. <https://www.nginx.com/blog/nginx-vs-apache-our-view/>

[14] YASUKATA, K.; HONDA, M.; SANTRY, D.; & EGGERT, L. *StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs*. In USENIX Annual Technical Conference, 2016, June. (pp. 43-56).

[15] MARINOS, I.; WATSON, R. N.; & HANDLEY, M. *Network stack specialization for performance*. In ACM SIGCOMM Computer Communication Review, 2014, Aug. (Vol. 44, No. 4, pp. 175-186). ACM.

[16] 3 July. 2017. <http://jmeter.apache.org/>

[17] 3 July. 2017. <http://mathworld.wolfram.com/StandardDeviation.html>