مقارنة بين virtual machine based System و

container based system

(taken from “**Analysis of Horizontal Reactive Scaling on Container Management Systems 2016” paper**)

مقدمة:

الهدف هو زيادة أداء النظام مع تخفيض الكلف التي نحتاج إليها لتشغيل هذا النظام ولكن الأداء في تغير دائم مع تغير أعداد المستخدمين المتصلين بهذا النظام وبشكل أمثلي يجب على النظام توقع عدد المستخدمين الممكن إتصالهم و إلا باء بالفشل في حالة وصول أعداد غير متوقعة ممّا سيؤدي الى توقف الخدمة وحدوث فشل حيث يتم إيقاف الخدمة لزيادة العتاد المطلوب و المناسب ممّا يشوش تقديم هذه الخدمة.

وبنفس الوقت اذا قام النظام بالعمل بشكل دائم بأقصى طاقته فسيكون لدينا ضياع في جهود النظام في الأوقات التي ينخفض بها أعداد الطلبات القادمة من المستخدمين.

للتحكم بهذا التغير الدائم في أعداد الطلبات للخدمات تم إبتكار مفهوم جديد

وهو الCLOUD COMPUTING

الحل اﻷول الذي يُستخدم فيها هو ال virtualization

الطريقة التقليدية هي استخدام virtual machines بحيث تمكن المطورين من زيادة أعدادها أو إنقصاها(التوسع أو التقلص) وفق الحاجة بشكل تلقائي, ففي حال كان الضغط مستمر بالزيادة على الخدمة يكون الحل بزيادة أعدادها و إذا انخفض الطلب على الخدمة نقوم بإلغاء بعض هذه الأجهزة اﻷفتراضية.

على الرغم من أن الوقت المتاح للزيادة أعداد ال virtual machines ليس بالكبير(ربما دقائق معدودات) إلا أنّه في بعض الحالات قد يتزايد الطلب على الخدمة بشكل لا يسمح لها بزيادة أعدادها بالوقت المناسب مما سيؤدي إلى فشل الكثير من الطلبات.

هنا يأتي الحل الثاني وهو إستخدام ال containers والتي عمليا يمكن زيادة أعدادها وتشغيلها في ثواني قليلة والتي ستمكن النظام المخدم بالتوسع او التقلص تلقائيا في وقت قصير مما يمنح المستخدم زمن إستجابة أفضل في حال زاد الطلب على الخدمة بشكل كبير و غير متوقع بالإضافة إلى إستخدام أفضل للموارد الموجودة دون ضياع كبير.

الهدف الآن هو مقارنة أداء و تكاليف كلا من الحلين

Virtual Machines Container based system

التكنولوجيا المستخدمة في تجربة الحلين هي

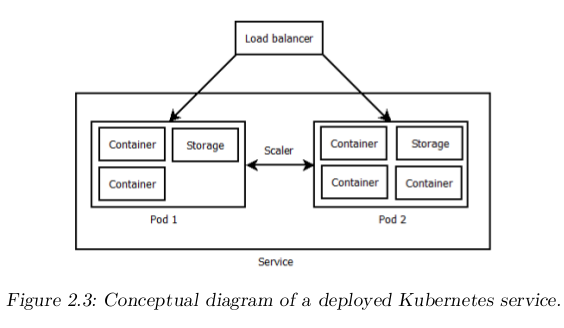
Amazon Web Services Kubernetes

شرح مبسط لبنية kubernetes

Kubernetes يستخدم nodes تحوي كل منها pod والتي تمثل وحدة البناء الأساسية لمستخدم Kubernetes ويحوي كل pod حاويةواحدة container أو أكثر.

من أهم ما يحتويه kubernetes هو (HPA(horizontal pod auto-scaler الذي يقوم بتغير أعداد ال pods وفقا لأرقام محددة مسبقا.

ومن ميزات Kubernetes أنها تمكنك من البدء وإنشاء(Deploy) بنية الخدمة كاملة باستخدام ملف إعدادات واحد مكتوب بصيغة JSON أو YAML.



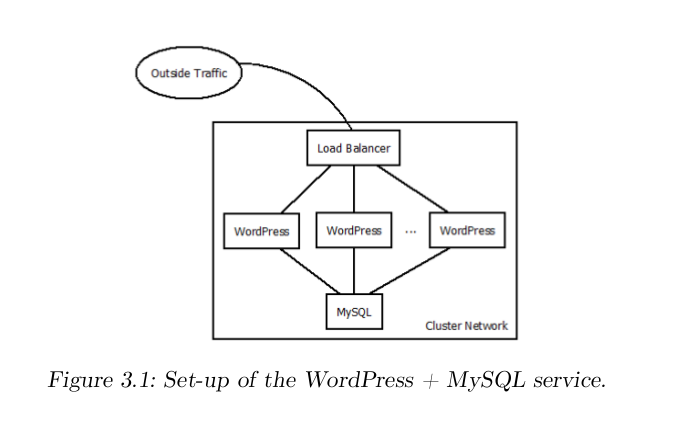
إختيار التطبيق المستخدم في التجارب

على التطبيق المستخدم للتجربة أن يوضح الفرق بين إستخدمه في الحاويات (Containers) أو في

الvirtual machines وعليه أن يحتوي على مستوى جيد من المعالجة مع وجود عامل زيادة وتوسع يوضح توسع الخدمة لكي تتمكن من الإستجابة للمستخدمين بشكل جيد.

التطبيق المختار للتجارب هو wordpress

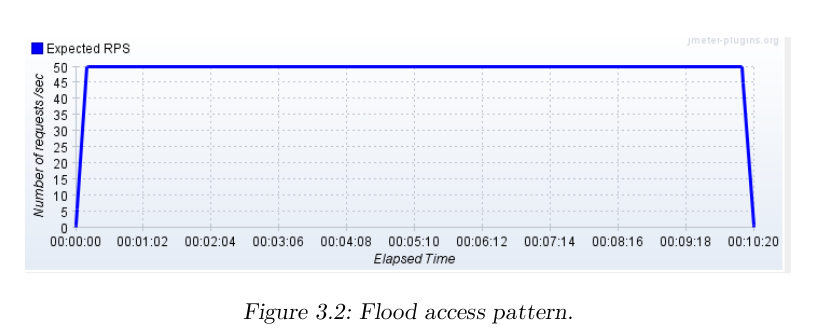
وهو موقع يستخدم لإنشاء صفحات أو موقع ويستخدم php و mySQL كمحرك قاعدة البيانات الخاصة به.

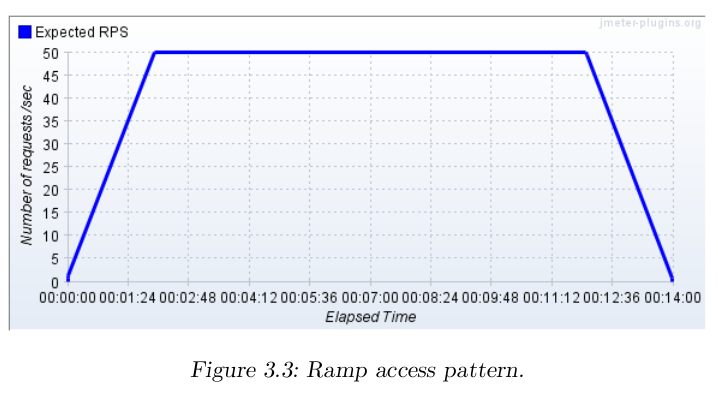


أقسام الإختبارات:

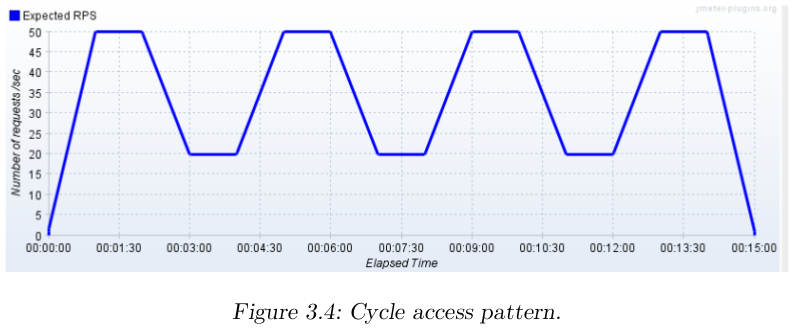
أولا: القياس من وجهة نظر المستخدم النهائي باستخدام JMeter

أهم ما يتم قياسه هو زمن الاستجابة الوسطي للخدمة التي تتطلب من قبل المستخدم النهائي حيث يلاحظ ثلاث أنماط لوصول المستخدمين للخدمة.

الوصول الفائض: يتمثل بوصول أعداد هائلة للمستخدمين من أول لحظة إلى أخر لحظة لعمل النظام.

الوصول التدريجي: تتدرج طلبات الخدمات إلى أن تصل إلى الحد الأقصى لقدرة النظام ثم تعود لتنخفض تدريجيا.

الوصول المتقطع المتكرر: تتم فيه زيادة أعداد المستخدمين لفترة قصير ثم تنقص في الفترة التي تليها و يستمر هذا التكرار طيلة حياة النظام في هذه النمط.



في الصور التي سترد لاحقا يتم إختبار النظام المخدم لفترة 10 دقائق و يتم تجربة أنماط الوصول الثلاثة المبينة سابقا

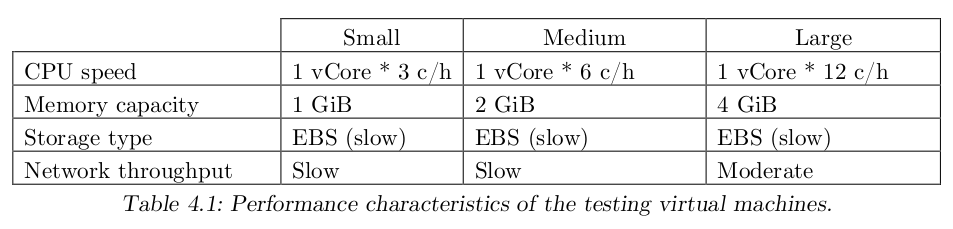
ثانيا: القياس من وجهة نظر النظام المخدم بإستخدام برنامج مكتوب يدويا

سيتم تحديد توسع و زيادة أعداد الcontainers و virtual machines عند زيادة إستخدام المعالج

على 50%

أي سيتم فيه قياس أداء المعالج و مشاهدة كيفية توسع النظام وفقا له.

التجربة: سيتم إستخدام ثلاثة أنواع مختلفة من الأجهزة مختلفة في قدرتها:

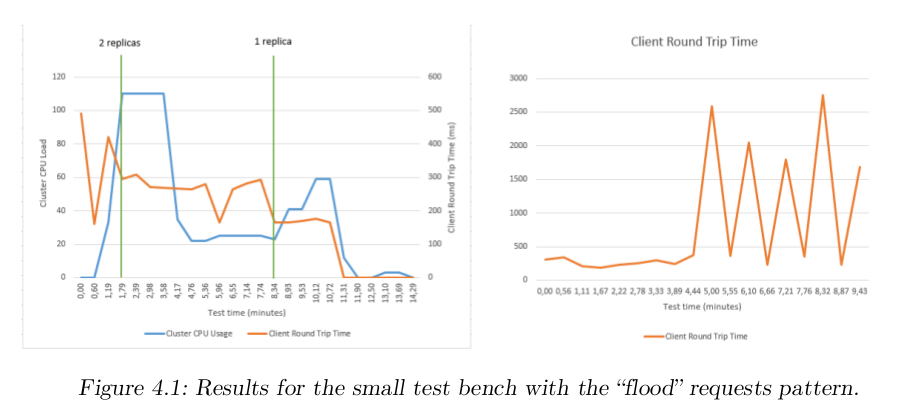


حيث يتم حجز 5 أجهزة ل kubernetes واحد فقط للmaster node و اﻷخرى للslave nodes

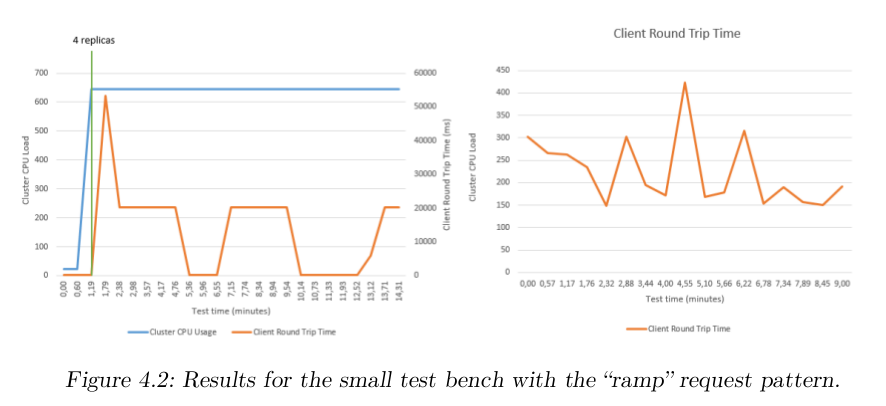
ويتم حجز 5 أجهزة للتجربة على virtual machine بحيث يتم حجز و تشغيل واحد فقط و الإعتماد على خاصية التوسع التلقائي في حال تتطلبت الخدمة المزيد من الأجهزة دون الزيادة على 5 أجهزة.

النتائج:

Small profile test results



|  |  |  |
| --- | --- | --- |
| F 4.1 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 417,76 | 1051,65 |
| نسبة نجاح الطلبات | 100% | 97% |



|  |  |  |
| --- | --- | --- |
| F 4.2 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 220,43 | 834,23 |
| نسبة نجاح الطلبات | 100% | 96% |



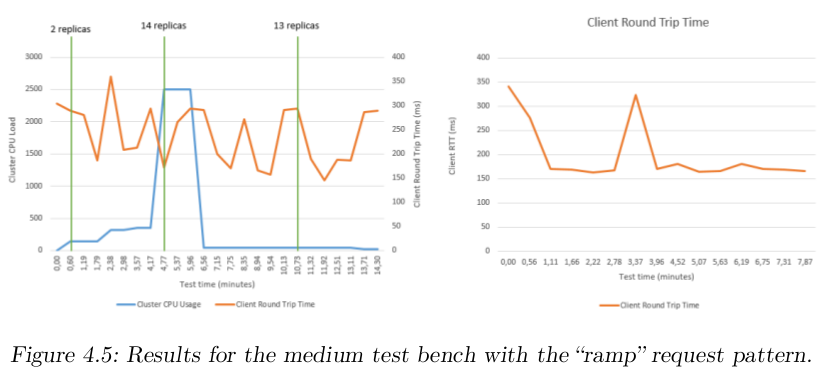
|  |  |  |
| --- | --- | --- |
| F 4.3 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 969,85 | 1505,70 |
| نسبة نجاح الطلبات | 100% | 95% |

يمكن تفسير ما جرى من ضعف أداء بإستخدام الcontainers بأن kubernetes تتطلبت حمل إضافي بحد ذاتها والذي لم يتم إستحماله من قبل جهاز بمواصفات ضعيفة.

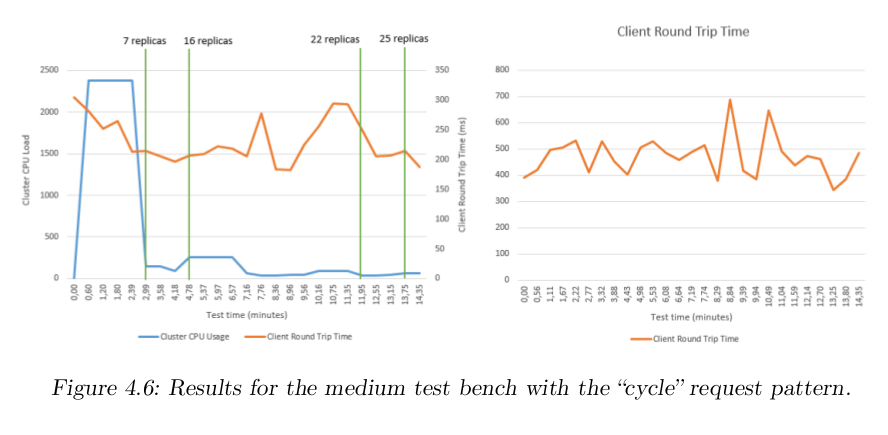
Medium profile test results



|  |  |  |
| --- | --- | --- |
| F 4.4 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 545,17 | 224,37 |
| نسبة نجاح الطلبات | 100% | 100% |



|  |  |  |
| --- | --- | --- |
| F 4.5 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 227,16 | 211,54 |
| نسبة نجاح الطلبات | 100% | 100% |



|  |  |  |
| --- | --- | --- |
| F 4.6 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 541,34 | 232,42 |
| نسبة نجاح الطلبات | 100% | 100% |

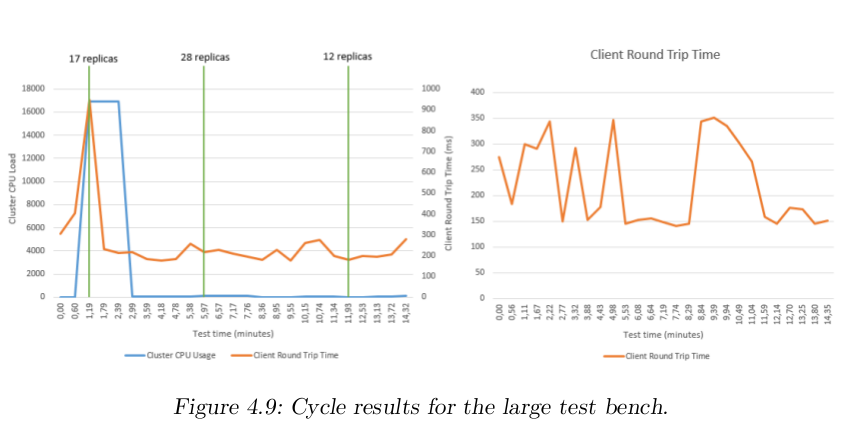
يمكنك ملاحظة تفوق الcontainers وخاصة في معدل زمن الإستجابة الوسطي (حيث أن نسبة نجاح الطلبات كان كاملا في الحلين) ممّا يعطي مستخدمين هذه الخدمة تجربة أفضل.

Large profile test results

|  |  |  |
| --- | --- | --- |
| F 4.7 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 303,11 | 208,17 |
| نسبة نجاح الطلبات | 100% | 100% |



|  |  |  |
| --- | --- | --- |
| F 4.8 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 188,77 | 193,65 |
| نسبة نجاح الطلبات | 100% | 100% |



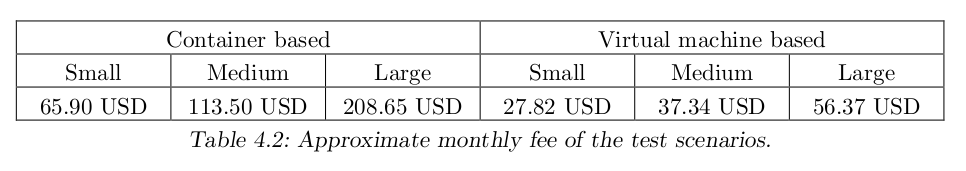
|  |  |  |
| --- | --- | --- |
| F 4.9 | VM(AWS) | Containers(Kubernetes) |
| زمن الإستجابة الوسطي in ms | 253,45 | 232,42 |
| نسبة نجاح الطلبات | 100% | 100% |

يمكنك ملاحظة تفوق الcontainers وخاصة في معدل زمن الإستجابة الوسطي (حيث أن نسبة نجاح الطلبات كان كاملا في الحلين) ممّا يعطي مستخدمين هذه الخدمة تجربة أفضل.

وعلى الرغم من إعداد الvirtual machine للتوسع التلقائي بعد مجاوزة حد 50% من المعالجة إلا أن النظام لم يتوسع أبدا و استكمل تلبية الطبات بجهاز إفتراضي واحد فقط.

حساب الكلفة:

بما أن الحل المستخدم للcontainers تفوق على الحل الأخر من حيث الأداء و زمن الإستجابة ممّا يعطي المستخدم النهائي تجربة أفضل إلا أن الحل المستخدم لvirtual machine لم يتطلب الكثير من الإعداد و بمقارنة الكلف المقابلة لكلا من الحلين فقد يناسب بعض أصحاب المصلحة إستخدام الvirtual machine حتى ولو كان ذلك على حساب تجربة المستخدم النهائي



لاحظ إرتفاع أسعار الكلف المستخدمة لحل الcontainers مقارنة مع الحل الأخر.

نجد أنّ الإختيار الأفضل يقع على عاتق المهندس للموازنة بين الكلفة و الهدف المطلوب تحقيقه من الخدمات المقدمة على هذا النظام.

النتيجة:

نلاحظ أنّ إستخدام الحاويات أفضل من إستخدام الأجهزة الإفتراضية من حيث الأداء إن كنا قابلين لإدارتها بشكل جيد إلا أن الكلفة يجب أن تؤخذ بعين الإعتبار قبل إتخاذ القرار النهائي علما أن الكلفة قد تتضمن تدريب طاقم العمل لديك قبل البدء بإستخدام هذا الحل في إيصال خدماتك.