¿Cómo afecta Docker al consumo de energía? Evaluación de cargas de trabajo dentro y fuera de los contenedores de Docker

Eddie Antonio Santos∗ Carson McLean∗ Christopher Solinas∗ Abram Hindle∗

Abstracto

Contexto: Las máquinas virtuales proporcionan aislamiento de los servicios a costa de los hipervisores y más uso de recursos. Esto impulsó el crecimiento de sistemas como Do-

creación de instancias de una imagen. Tradicionalmente, la virtualización se ha implementado a través de máquinas virtuales, en las que una máquina puede albergar varios sistemas de operación invitado. ing Sin embargo, la intervención de la hi--

1

cker que permiten a los hosts individuales aislar varios aplica--

pervisor,

significa que las aplicaciones deben

Cationes Similar Para Vms Dentro a desbado ab- straction Llamado Contenedores.

arXiv:1705.01176v1 [cs. DC] 2 de mayo de 2017

Motivación: Aunque los contenedores promocionan un bajo rendimiento superior, ¿todavía tienen bajo consumo de energía - ción?

Metodología: Este trabajo compara estadísticamente (t- test, Wilcoxon) el consumo de energía de tres cargas de trabajo de ap- plicación en Docker y en Linux bare-metal.

Resultados: En todos los casos, hubo un aumento estadísticamente sig- nificant (t-test y Wilcoxon p < 0.05) en el consumo de energía al ejecutar pruebas en Docker, principalmente debido al rendimiento de las llamadas al sistema de E/S.

Palabras clave: virtualización, docker, containeriza- ción,, consumo de energía, cloud computing, mi- croservice.

1 Introducción

La virtualización proporciona una serie de ventajas al implementar software, como el aislamiento de procesos y el control de código fuente. El aislamiento de procesos significa que los desarrolladores de software pueden hacer suposiciones sólidas sobre el estado del sistema, incluida la configuración de sys- tem operativa y tener las de- pendencies exactas de software necesarias para el sistema. La virtualización de diez permite el control de recursos de modo que los opera- tors puedan configurar con precisión la cantidad de CPU, mem- oryo acceso a las interfaces de red que tiene una aplicación particular. Las plataformas de virtualización a menudo usan im- edades, instantáneas del sistema completo necesario para ejecutar una aplicación, , por lo que la implementación de una aplicación es tan fácil como

∗ DepartamentoDepartment de Ciencias de la Computación, Universidad de Alberta, Edmonton, Canadá

utilizar dos núcleos, directamente a través de la operación invitada -

sistema de ing, e indirectamente a través del hipervisor, al acceder a recursos como la red y el almacenamiento-edad. Esto puede considerarse un sobre-cabeza indeseable. Esto motivó la necesidad de una máquina virtual de baja sobrecarga. Recientemente, las sofisticadas características del kernel de Linux, a saber, los espacios de nombres y los grupos de control, hicieron posible una nueva forma de virtualización ization de baja sobrecarga: la contenedorización. Los contenedores son una alternativa ligera a las máquinas virtuales, ya que de aislamiento fer (procesos, sistema de archivos, red) y control de recursos (CPU, memoria, disco) sin la sobrecarga de un kernel adicional. Container man- software de envejecimiento como Docker [9], LXC [2] y rkt [6], están desplazando rápidamente las máquinas virtuales como la solución de virtualización de elección [1, 12].

Dado el ritmo de la adopción de la con- tainerización,, ¿cuál es el impacto de la contenedorización en el consumo de energía? Los cambios en el software tienen diferencias significativas y medibles en el consumo de energía y energía [10, 14, 15, 27, 34]. Dado que la contai- nerización, en principio, carece de la sobrecarga de las máquinas virtuales, claramente debe consumir una cantidad similar de energía como una configuración de metal desnudo. figuration.

En este artículo, probamos empíricamente este asumo- ción contra numerosas cargas de trabajo medidas, ejecutar con y sin contenedorización. En la práctica, los proveedores de contenedores, como Docker, agregan ventajas excesivas adicionales, como el sistema de archivos AUFS y una capa de red ab-stracted. We Buscamos cuantificar el impacto que estos gastos generales tienen en la eficiencia energética. Comparamos el consumo de energía de varios sce- narios funcionan en Linux de metal desnudo, es decir, la aplicación-

1 Utilizamos el término hipervisor para cualquier máquina virtual moni-tor que esté alojada encima de un sistema operativo existente, o que sea un módulo del kernel del sistema operativo host, como KVM [16].

Las ciones se ejecutan en un núcleo, sin ningún virtu- alization en absoluto - en contraste con los con- tainers administrados por Docker , utilizando imágenes de Docker "off-the-shelf". Utilizamos el consumo total de energía del sistema (o "potencia de pared") para estimar el consumo total de energía. We Ejecutamos varias iteraciones de cada experimento, cuyos resultados presentamos y explicamos por qué vemos diferencias en el consumo de energía entre Linux y Docker. Este trabajo sugiere que no hay almuerzo gratuito para la contenedorización en términos de consumo de energía. La contenedorización implica un equilibrio entre energía y mantenimiento, y depende de las personas o equipos encargados de la implementación determinar cuál es

más costoso en su escenario particular.

2 Trabajo previo

El trabajo anterior se ha centrado en la energía de la máquina virtual y el consumo de energía. Xu et al. [33] midió la CPU y el uso total de energía en hipervisores Xen y KVM. Encontraron que Xen generalmente tiene una mayor sobrecarga de energía que KVM al procesar el tráfico de red, atribuido a "re-búsquedas de interrupción excesivas". Encontraron que a medida que la carga se distribuye más uniformemente entre las máquinas virtuales, el consumo de energía - tion aumenta. Este artículo explica el efecto de Docker en el consumo de energía de la red.

Algunos trabajos han comparado las máquinas virtuales con los contenedores directamente. Morabito [18] comparó el uso de energía de la máquina virtual tradicional hypervi- sors (KVM, Xen) con la virtualización basada en contenedores (Docker, LXC). En todos los casos, el estilo de contenedor vir- tualization utilizaba marginalmente menos potencia, pero en general ninguno de los dos métodos de virtualización mostró dif significativo - ference. Morabito ences no consideró que el tiempo de ejecución difería, , élnce este trabajo no puede sacar conclusiones sobre el consumo de energía general. Además, no hubo comparación con el rendimiento de Linux sin metal. Ambas of preocupaciones se abordan en nuestro trabajo. Van Kessel et al. [26] utilizaba sensores de hardware internos para des- termine la diferencia en el consumo de energía de Xen contra Docker. Encontraron que Docker es más eficiente - competente en cargas enlazadas a la CPU y de disco. En con- trast, nuestro trabajo se compara con la potencia de la pared de medición de Linux de metal desnudo en lugar de la energía interna sen- sors para cuantificar las abstracciones proporcionadas por Docker. Shea et al. [23]] comparó el consumo de energía de las transacciones de red utilizando virtualización como KVM, Xen y OpenVZ,, en contraste con un sistema de metal desnudo. Solo OpenVZ se puede considerar virtualización basada en contenedores. Midieron la potencia de ambos muros

y la potencia de la CPU mediante el límite de energía promedio de funcionamiento (RAPL) de Intel. Los autores encontraron que el poder mea- asegurado a través de RAPL era always una fracción de la potencia de pared medida. Encontraron una diferencia en los gastos generales de energía de las transacciones de red en plataformas de virtualización ferent dif. Sin embargo, llegaron a la conclusión de que los gastos generales eran aturdidos. Nuestro trabajo se centra únicamente en el virtubasado en contenedores de Docker: la alización.. Medimos la potencia de la pared solamente, porque queríamos capturar el uso total de energía del sistema. Ad- ditionally, medimos más escenarios que sólo transacciones netas de trabajo.

Otros trabajos han evaluado las métricas de rendimiento del contenedor, como el tiempo de ejecución, el uso de CPU y la utilización de la red. Felter et al. [11] comparó cpu, mem- ory,, E/S y rendimiento de red de Docker y KVM con Linux sin metal. En los casos most, Do- cker añade poca sobrecarga, y casi siempre supera-forma KVM. También probaron cargas de muestra en Re- dis y MySQL. They Encontraron que, en algunos casos, como el ejemplo de Redis, Docker realiza compa- rably a bare-metal cuando se configura correctamente. Los autores encontraron que la abstracción de sistema de archivos UnionFS de tem Docker tiene sanciones de rendimiento negativas en comparación con un sistema de archivos Linux estándar. En con- trast, nuestro trabajo mide directamente el consumode energía - ción de ejecutar puntos de referencia similares, tanto en Linux bare- metal en comparación con dentro de un contenedor Docker. En general,el tiempo de ejecución más rápido se correlaciona con un menor consumo de energía; sin embargo, la potencia también debe medirsed junto con el rendimiento para observar el consumo energético general de una tarea.

3 Metodología

Queremos comparar el consumo de energía de la ejecución de ning una carga de trabajo dentro de un contenedor administrado por Docker (elthe tratamiento)) con la ejecución de la misma carga de trabajo en "bare-metal" (el control). Para estimar el consumo de energía de una carga de trabajo, ejecutamos un servidor (el sistema de bajo prueba o SUT) con el software de inter- est; we ejecutamos un sistema externo para iniciar pruebas en el SUT y registrar las mediciones de potencia (el corredor de pruebas); y usamos un medidor de potencia para medir la potencia instantánea consumida por el SUT. Configuramos los sistemas para ejecutar el software deseado, ya sea iniciando ing el servicio (Linux de metal desnudo) o iniciando un nuevo con- tainer (Docker) que ya se ha construido. A continuación, iniciamos las pruebas en el ejecutor de pruebas, lo que in- duce una carga de trabajo en el SUT después de una pausa de dos minutos. Durante la prueba, recopilamos root-mean-squared

System-Under-Test

Dell PowerEdge R710

a la toma de corriente de la pared

CPU 2- Intel Xeon X5670 de seisnúcleos a 2,93 GHz

RAM 72 GiB ECC DDR3

Conexión Gigabit Ethernet de red

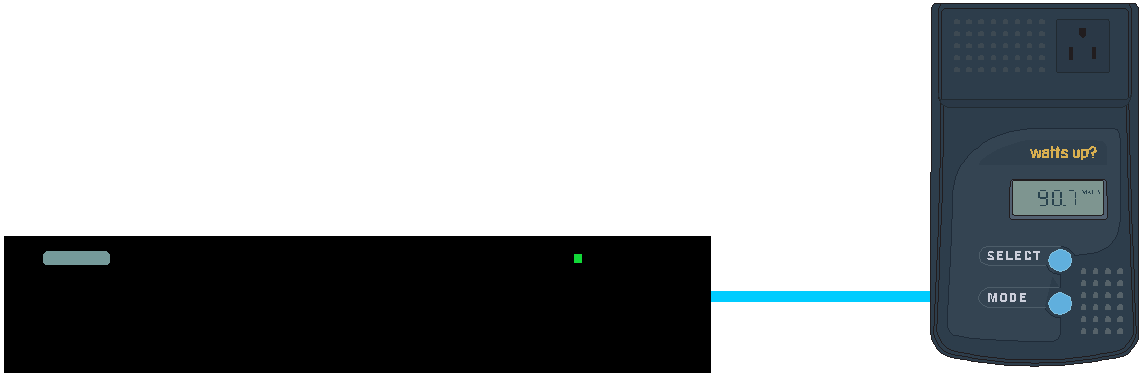
Disco duro SAS de 146 GB de almacenamiento a 15000 RPM

Fuente de alimentación 870 vatios (120 voltios ∼ 12A a 60 Hz)

Corredor de pruebas

Dell PowerEdge R710

Usb



Medidor de potencia

¿Watts Up? PRO

Tabla 1: Configuración de hardware del sistema- Under-Test y el ejecutor de pruebas.

Versión de software Docker Image

Distribución Ubuntu Server 16.04.1 LTS Kernel Linux 4.4.0

Docker 1.12.1

Apache 2.4.10 php:5.6-apache

Figura 1: Configuración de prueba de hardware: un servidor de montaje en rack System-Under-Test; y un corredor de pruebas. Power mea- los aseguramientos se recogieron con un Watts Up? pro.

(RMS) power mediciones de energía, y las registró. Utilizamos used las mediciones power para estimar el consumo total de energía en el SUT en dos escenarios: el software que se ejecuta en Linux sin metal en comparación con el software que se ejecuta dentro de un contenedor Docker.

Es importante destacar que el System-Under-Test no es la misma máquina que el ejecutor de pruebas; por lo tanto, iniciar las pruebas (ejecutor de pruebas) se aísla de la ejecución de la prueba (SUT). Por lo tanto, un ergy servidor independiente se utiliza como ejecutor de pruebas tanto para iniciar pruebas como para registrar estadísticas de uso desde el medidor de potencia.

Esta sección describe el hardware y el instrumen- tation que usamos para ejecutar tareas y recoger power sam- ples. En la Figura 1 se proporciona una descripción general de nuestra configuración completa. Nuestra configuración de hardware consistió en un servidor de montaje en rack como nuestro System-Under-Test (Sección 3.1), un medidor de potencia digital (Sección 3.2) para recoger muestras de energía y un ejecutor de pruebas (Sección 3.3) para iniciar las cargas de trabajo.

3.1 System-Under-Test

System-Under-Test (SUT) es un servidor de montaje en rack Dell PowerEdge R710. Un resumen de su hardware se enumera en la Tabla 1. Aunque el R710 está diseñado para usarse con fuentes de alimentación redundantes, múltiples interfaces de red y almacenamiento RAID redundante, solo utilizamos una fuente de alimentación, una interfaz de red (una conexión Ethernet gigabit) y un disco duro para nuestras pruebas. Los 2 Intel Xeon X5670 contienen 6 núcleos cada uno, con un total de 12 núcleos reales, y con hyper-threading habilitados aparecen como 24 proces lógicos - sors a Linux.

Un resumen del software instalado se enumera en

PHP 5.6.24 php:5.6-apache

MySQL 5.7.15 mysql:5.7.15

WordPress 4.6.0 wordpress:4.6-apache

Redis 3.2.3 redis:3.2.3

PostgreSQL 9.5.4 postgres:9.5.4

Tabla 2: Versiones de software utilizadas en el sistema- Sub-prueba

Cuadro 2. Docker se instaló en la prueba System-Under-Test. Para las versiones de metal desnudo de Apache, PHP, Word- Press, MySQL, y PostgreSQL, utilizamos apt-get. Redis se instaló desde el origen en Linux de metal desnudo. Todo el software de aplicación de Docker se ejecutó dentro de contenedores administrados por Docker. Al instalar soft- ware en Docker, usamos la imagen oficial alojada en Docker Hub [8]. Tenga en cuenta que la imagen de WordPress hereda de la imagen php:5.6-apache, que se detiene tanto PHP como Apache. Por lo tanto, la única imagen que tuvimos que instalar explícitamente era la que contenía WordPress.

3.2 Mediciones de potencia

Este documento se centra en comparar la energía necesaria para realizar varias tareas. Sin embargo, no podemos mea- energía segura directamente. En su lugar, medimos la potencia de pared en monttána extraída por la prueba System-Under-. Para esto, ¿usamos un Watts Up? medidor de potencia profesional [29].

¿El Watts Up? pro es un dispositivo con una toma de corriente alterna tipo B. It Muestra la tensión y el consumo de alquiler del aparato eléctrico conectado a su toma de corriente. Dado que la potencia es tensión multiplicada por corriente, el medidor puede informar de la potencia instantánea de la edad de un aparato eléctrico, en nuestro caso, un servidor de montaje en rack como nuestro System-Under-Test. Puesto que estamos interesados en el uso total de energía del sistema de neumáticos en - incluyendo la CPU, pero también la memoria, almacenamiento, interfaces de red, periféricos, refrigera-

ing, e incluso la sobrecarga due a la fuente de alimentación, optamos por medir la potencia de la pared, en lugar de utilizar la medición a bordo, como EL RAPL de Intel para mea- asegurando el uso de energía de la CPU solo. El Watts Up? pro calcula la raíz-media-cuadrada (RMS) de las arenas de las muestras en el transcurso de un segundo [30]. Trabajos anteriores de McCullough et al. [17] ¿Encontró que la recopilación de mediciones DE RMS a una frecuencia de una medición por segundo de un Watts Up? medidor de potencia es suficiente para una estimación precisa del consumo de energía [17].

Usamos una versión modificada del software yyongpil's’s wattsup2 para recuperar las medidas de potencia de watts Up? pro y guardarlos en el corredor de prueba. Cada segundo,, el vataje utilizado por el System-Under-Test se extrae del Watts Up? pro, transferido a través de USB al corredor de pruebas, y luego escrito a stdout. Los scripts de recopilación en el ejecutor de pruebas controlaban las ejecuciones de prueba para cada uno de nuestros estudios de caso y las mediciones registradas para cada ejecución de prueba con el fin de recopilar datos de potencia junto con marcas de tiempo. Esta información se guardó en una base de datos SQLite3 local en el ejecutor de pruebas.

Sin embargo, la energía no es energía. La energía es la in- tegración del poder a lo largo del tiempo. El Watts Up? pro produce muestras de potencia RMS de un segundo de duración: varias mediciones de potencia instantánea promediadas en un segundo. Dada una marca de tiempo inicial (ti) y una marca de tiempo final (tf ), podemos usar la suma de muestras de energía para estimar la energía necesaria para completar una tarea. Nos aproximamos en- ergy utilizando una suma de muestras de energía, tomadas a una frecuencia regular. Esto es análogo al uso del método rectángulo de aproximación de una integral con una duración

∆t de 1 segundo (Ecuación 1).

para restablecer la máquina a su estado de ejecución inactivo, como Chowd- hury et al. [4] descubrió que la ejecución de pruebas en rápida sucesión puede alterar el estado de potencia de la máquina, sesgando artificialmente los resultados. Estos resúmenes de energía se comparan, agrupados por caso práctico, para Linux de metal desnudo frente a Docker.

3.3 Corredor de pruebas

Para iniciar las pruebas y registrar la alimentación sam- ples, utilizamos un servidor de montaje en rack Dell PowerEdge R710, idéntico en la especificación de hardware y con- figuración que el SUT. Escribimos scripts de colección en Python que inician las pruebas (descritas en la Sección 4) en el System-Under-Test a través de solicitudes de red, mientras que si registramosmultanosamente las estadísticas de energía de Watts Up? pro a través de USB con vatiosup de yyongpil. . Grabamos marcas de tiempo para cada muestra de potencia.

Para cada experimento:

1. Iniciamos el servicio en el System-Under-Test (siif corresponde).). En Docker,, iniciamos uno o varios contenedores nuevos a partir de sus respectivas imágenes Do- cker.

2. En el corredor de pruebas, iniciamos un lote de carreras de prueba.

3. Para cada ejecución de prueba, el ejecutor de pruebas opcionalmente por-formado una inicialización por prueba.

4. El corredor de pruebas entonces utes dormiría durante dos minutos. .

5. El corredor de pruebas indujo entonces una carga de trabajo en el

System-Under-Test a través de solicitudes de red.

6. Durante cada prueba, el corredor de pruebas registró las mediciones de potencia instantáneas del SUT y la marca de tiempo cada segundo.

Z tf

E ?

ti

F

P (t) dt á ∆t X PRMS (tk) (1)

k'i

7. Una vez finalizadas todas las ejecuciones de prueba de un lote, calculamos la energía por cada ejecución de prueba.

El ejecutor de pruebas se conectó al sistema-

Escribimos scripts de Python que implementaron la estimación anterior, tomando en los datos de prueba de las bases de datos SQL- ite3 en el corredor de pruebas, que tenían potencia en vatios con marcas de tiempo. Cada marca de tiempo se afirmó que estaba a aproximadamente un segundo de distancia, por lo que nuestro esti- mation válido. La suma produce una estimación de la energía total consumida para una sola ejecución de una prueba. Consideramos que cada prueba era una muestra de energía. Corrimos cada prueba 40 veces, lo que nos da 40 sam de energía - ples por caso de estudio por configuración. Antes de cada prueba, tuvimos la máquina durmiéndose durante dos minutos

2https://github.com/yyongpil/wattsup

Prueba sub-prueba a través de un interruptor gigabit.

4 Caso práctico

Se seleccionaron tres proyectos de software de código abierto para probar la diferencia en el consumo de energía de ejecutar la aplicación en Linux sin hardware en comparación con un contenedor administrado por Docker. Cada una de las aplicaciones hace hincapié en diferentes recursos de hardware y, en conjunto, proporciona información sobre el rendimiento y la energía sobre qué tipos de aplicaciones son más adecuadas para Docker. WordPress con MySQL representa un sitio web extremadamente popular

redis y PostgreSQL son soluciones de base de datos comunes, con diferentes casos de uso. Consid- ering la popularidad y la amplitud de las aplicaciones se- lected como estudios de caso, los resultados dan información relevante sobre el efecto de Docker en el consumo de energía en comparación con las instalaciones de metal desnudo.

4.1 Inactivo

Como línea de base, nos interesaba cualquier posible sobrecabezado head de ejecutar el servicio Docker sin colocar ninguna carga en el sistema. In Con el fin de estimar cuánta energía se espera que se utilice en reposo, el sistema se dejó inactivo durante exactamente 10 minutos, durante los cuales se registró el uso de energía. In Con el fin de ser con- sistent sistent con la metodología utilizada para los siguientes estudios de caso, insertamos 2 minutos adicionales de tiempo de inactividad antes de cada ejecución de prueba durante la cual no se registraron muestras de energía. Esta prueba se realizó

40 veces secuencialmente y puede considerarse una línea base para Linux y Docker sin metal.

"Inactivo" significa que el sistema ha estado funcionando el tiempo suficiente para lograr un estado estable con nada más que el sistema operativo base en funcionamiento, lo que significa que ninguno de los otros servicios bajo prueba (PostgreSQL, Redis, MySQL, Apache) se estaban ejecutando o estaban activos de ninguna manera. Al realizar la línea base de Docker, la única diferencia es habilitar el servicio en segundo plano de Docker. Cero contenedores se estaban ejecutando, por lo que mea- aseguró la sobrecarga de sólo el propio demonio de Docker. Dado que el tiempo se fija en esta prueba, cualquier diferencia en en- ergy debe debe ser debido a una diferencia en el consumo de energía -

ción.

4.2 WordPress

WordPress es un sistema de gestión de contenido de código abierto [31]. A partir de febrero de 2017, Docker Hub ha tenido más de 10 millones de tiradores de WordPress [8] y Word- Press potencia más de una cuarta parte de los 10 millones de sitios web más altos en todo el mundo [28].

Hemos instalado WordPress manualmente para la versión Linux de metal desnudo, según la documentation oficial de WordPress [32]. We Usamos Docker Compose [7] para instalar WordPress dentro de Docker. Ambos meth- ods instalaron las mismas versiones de WordPress, My- SQL, PHP y Apache, como se indica en la Tabla 2. En el sistema de metal desnudo, MySQL y Apache funcionaban como ser- vicios. Docker requería dos contenedores: un contenedor con Apache,, que ejecuta WordPress con modphp,mientras que otro contenía la base de datos MySQL. Estos se configuraron automáticamente y se conectaron mediante Docker

Componer. Generamos un blog utilizando el plugin de contenido amplio de WP Ex-1.3 [13], cuya base de datos se copió tanto en la instalación sin sistema operativo como en la instalación de Docker.

Usamos Tsung 1.6.0 [25] para realizar una prueba de esfuerzo de carga HTTP en el servidor de WordPress para el que el corredor de pruebas estaba monitoreando el uso de energía. Tsung, corriendo en el corredor de pruebas, creó clientes virtuales que simu- tarde un gran número de usuarios que visitan la página principal de WordPress y navegar al azar por el sitio. Cada prueba duraba exactamente 15 minutos. A partir de la no carga, la prueba agregó 100 usuarios simulados por segundo. Cada usuario cargó el contenido de la página de inicio de WordPress, que a su vez requería consultas de base de datos con el fin de recuperar las publicaciones y otro contenido. Realizamos el test completo 40 veces secuencialmente, con el fin de producir 40 muestras en - ergy, con 2 minutos de tiempo de inactividad entre las pruebas para garantizar la precisión de las mediciones de energía.

4.3 Redis

Redis es un almacén de claves de código abierto en memoria que se puede utilizar como base de datos, caché o mensaje bro- ker [21]. En febrero de 2017, Docker Hub ha tenido más de 10 millones de tiradores de Redis [8]. Elegimos Redis para probar la sobrecarga de una carga de trabajo que es predominantemente memoria, CPU y red enlazada (hace accesos mínimos al almacenamiento).

Redis se instaló en Docker con la versión especificada ified en Table2. En Linux de metal desnudo, Redis fue construido a partir de la fuente. Para Docker, usamos la imagen oficial para crear un único contenedor que tenía el servidor de Redis. La imagen oficial descargada de Docker Hub elimina la persistencia periódica de la base de datos en memoria al almacenamiento permanente, por lo que también deshabilitamos esto en la configuración de bare-metal.

El conjunto de referencia de Redis, redis-benchmarkseutilizó para crear una carga de trabajo de 1000 clientes paralelos mak- un total de 1,5 millones de solicitudes. Esto implica una gran cantidad de tráfico de red desde el servidor que ejecuta los clientes, además de realizar una gran cantidad de accesos de memoria. Realizamos la prueba completa 40 veces secuencialmente, que produjo 40 muestras de energía, con dos minutos de tiempo de inactividad entre cada muestra.

4.4 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de objetos (DBMS) de código abierto [19]. A partir de Febru- ary 2017, PostgreSQL ha sido retirado más de 10 millones de veces [8].

Caso práctico Correlación de tamaño de efecto normal (rEt)

Cliff 's d Cohen's d Linux Docker Idle No 0.80

WordPress No 1.00 0.83 0.99

Redis Sí 11.31 0.98 0.98

PostgreSQL Sí 1.55 0.99 0.95

Tabla 3: Resumen de los resultados obtenidos para cada exper- iment. "Correlación"Correlation” se refiere a la correlación lineal entre la energía estimada con el tiempo transcurrido de la ejecución de la prueba. Nota: para el experimento"inactivo", , calcu - correlación de la energía con el tiempo de ejecución no tiene sentido porque el tiempo transcurrido es fijo.

PostgreSQL incluye pgbench para la evaluación comparativa de rendimiento. PostgreSQL se instaló en los servidores SUT y test runner con la versión especificada en la Tabla 2. En Linux demetal desnudo, ejecutamos PostgreSQL como un servicio, mientras que Docker tenía los procesos de base de datos en un solo contenedor. Es importante tener en cuenta que la versión de Ubuntu 16.04 habilita SSL de forma predeterminada, mientras que la instalación de Docker no lo hace. Hemos contabilizado esto deshabilitando SSL en la instalación de Linux PostgreSQL sin sistema operativo. También realizamos una prueba en la configuración de bare-metal con SSL habilitado, para comparar la sobrecarga de Docker con el exceso de consultas de We cifrado. En Docker, la imagen predeterminada de PostgreSQL crea un volumen montado en el host (es decir, al escapar del contenedor) para conservar los datos. Por lo tanto, las escrituras no tienen acceso a la capa de almacenamiento AUFS de Docker.

La prueba consistió en ejecutar pgbench en el ejecutor de pruebas con 50 clientes, cada uno peforming 1000 transacciones de base de datos en el SUT de "un escenario que se basa libremente en TPC-B" [20, 24]. Realizamos 40 pruebas sequen- tial para producir 40 muestras de energía. Antes de cada prueba, ejecutamos pgbench -i para inicializar la base de datos, luego esperábamos dos minutos de tiempo de inactividad antes de comenzar- la prueba correctamente. La prueba completa se realizó tanto para Linux de metal desnudo como para Docker.

5 Resultados

Después de recoger todas las muestras de energía, estimar la energía

datos están disponibles en línea. 3

En primer lugar, determinamos si las muestras de energía en Linux y en Docker se distribuyeron normalmente mediante la prueba de normalidad Shapiro-Wilk. Luego, aplicamos varias pruebas para determinar si ambas muestras provenían de la misma distribución. Para los datos distribuidos normalmente, data, usamos una prueba t de Student emparejada. De lo contrario, aplicamos pruebas no paramétricas: una prueba de suma de rango Kruskal-Wallis y una prueba de suma de rango Wilcoxon por pares. En los dos experimentos de muestra, encontramos que la diferencia en las distribuciones de consumo de energía en Docker en comparación con Linux era estadísticamente sig- nificant, con un valor pcerca de cero,4 sin importar qué prueba usamos. Para cuantificar la diferencia, calcu- lated el tamaño del efecto. Para todas las pruebas, usamos el delta de Cliff, que simplemente compara la frecuencia con la que las muestras de una distribución son mayores que las muestras de la otra distribución. Como se muestra en la Tabla 3, para los experimentos de WordPress y Redis, las distribuciones de Do- cker son todas mayores que las observaciones de Linux con un delta máximo de Cliff de 1.0. Los otros dos experimentos también tuvieron un gran tamaño de efecto, según el delta de Cliff, con pequeñas superposiciones en las distribuciones. Fi- nally, calculamos la correlación lineal, el coeficiente de correlación de Pearson, de energía con tiempo de ejecución. Vuelva a llamar a que la energía es energía.× Por lo tanto, la energía debe estar fuertemente correlacionada con el tiempo (un valor r de +1.0). En todos los casos, - related encontramos que la energía estaba fuertemente relacionada con el tiempo, sin embargo, ya que el valor r de cada prueba no era exactamente 1.0, afirmamos que otros factores deben estar influyendo en la energía total en lugar de en- ergy siendo completamente explicado por el tiempo de ejecución.

Los resultados se presentan de dos maneras: resúmenes de los datos de energía se presentan en parcelas de violín (Fig- ures 2, 4, 8, 6) que pueden leer algo como tramas de caja donde cada "violín" representa una distribución. La anchura del violín en un punto dado representa la densidad de las mediciones observadas en ese punto. Para dar una sensación de tendencia, se dibuja una línea en la mediana de la distribución de la muestra. Los resúmenes de los datos de potencia se indican como gráficas de densidad (Figuras 3,

5, 10, 7), con contenedores hexagonales. Cada bin representa un grupo de observaciones en el momento dado y el vataje. Los agones hexagonalesmás oscuros representan una concentración más densa de observaciones.

por cada prueba, realizamos algunos análisis estadísticos sobre

los resultados para determinar si hay una diferencia significativa en el consumo de energía para ejecutar una tarea en Linux sin sistema operativo en comparación con un contenedor de Docker. En el Cuadro 3 figura un resumen de nuestros resultados. Nuestro crudo

3Disponible: https://archive.org/details/docker-linux- energy-feb-2017.sqlite3

4Si el valor p es inferior a 10x4 (y, por lo tanto, sólo se ex- presionado usando notación exponencial), entonces consideramos que es "casi cero".

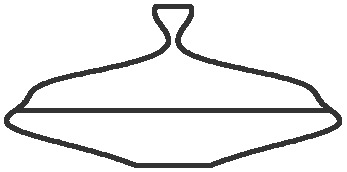
68.000 J

66,000 J

Energía

64.000 J

Linux Docker



no es elegible que los operadores decidan, sin embargo, más adelante describimos cómo hacer retroceder la diferencia en el consumo de energía.

5.2 WordPress

La distribución del consumo de energía para ejecutar una carga simulada en un servidor de WordPress en Linux y dentro de Docker se muestra en la Figura 4. Una guarida- sidad de poder se proporciona en la Figura 5. Usando la prueba Shapiro-Wilk, sólo la distribución del consumo de energía bajo Linux de metal desnudo se distribuía normalmente; por lo tanto, utilizamos pruebas no paramétricas para la comparación y el tamaño del efecto. Tanto la prueba de suma de rango Kruskal- Walis como la de Wilcoxon por pares dieron un valor p cercano a cero, lo que significa que las distribuciones son significativamentecantly diferentes. Para el tamaño del efecto, pusimos puted un delta de Cliff de 1.0, lo que implica distribuciones completamente no superpuestas. En otras obras, todas las muestras

Figura 2: Gráfica de violín del consumo de energía inactiva

5.1 Inactivo

La distribución del consumo de energía sin carga durante 10 minutos se da como una parcela de violín en la Figura 2. En la Figura 3 se proporciona una densidad de potencia. Al realizar la prueba Shapiro-Wilk, descubrimos que ni el Linux desnudo ni las distribuciones de Docker se distribuyen normalmente. Usando la prueba No paramétrica Kruskal-Wallis y la prueba de suma de rango Wilcoxon por pares, obtuvimos un valor p cercano a cero, lo que indica que las distribuciones son realmente diferentes. Usando el delta de Cliff, , tenemos un tamaño de efecto de 0.80, lo que indica- ing que los valores en la distribución de Docker son casi

80% de probabilidades de ser mayor que una observación en la distribución Linux de metal desnudo. Otra forma de pensar en esta diferencia, es que tres cuartas partes del tiempo, ning observamos que al ejecutarse en Linux sin carga sin carga utilizaría menos de 63.380 julios de en- ergy, mientras que si simplemente el demonio de Docker se ejecutaba (sin contenedores en ejecución), tres cuartas partesdel tiempo observaríamos que la máquina consume más de 63.380 julios de energía por no hacer nada durante diez minutos. Esta diferencia de energía no se puede atribuir al rendimiento,, ya que el tiempo se fija a 10 minutos en ambos casos.

Esta línea base establece que, dado que el demonio de Docker es un servicio inevitable que debe ejecutarse, independientemente de si los contenedores se están ejecutando o no, la ejecución de Docker conlleva una sobrecarga de energía. Si esta diferencia en el consumo de energía a lo largo del tiempo es neg-

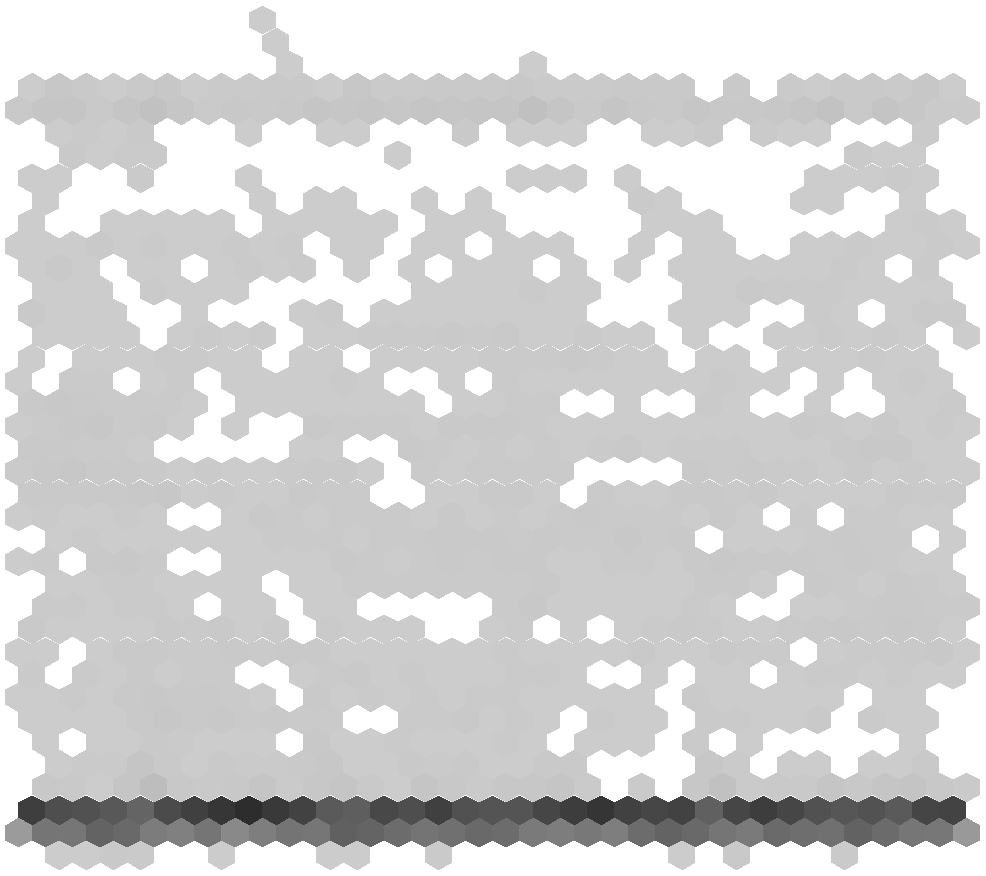
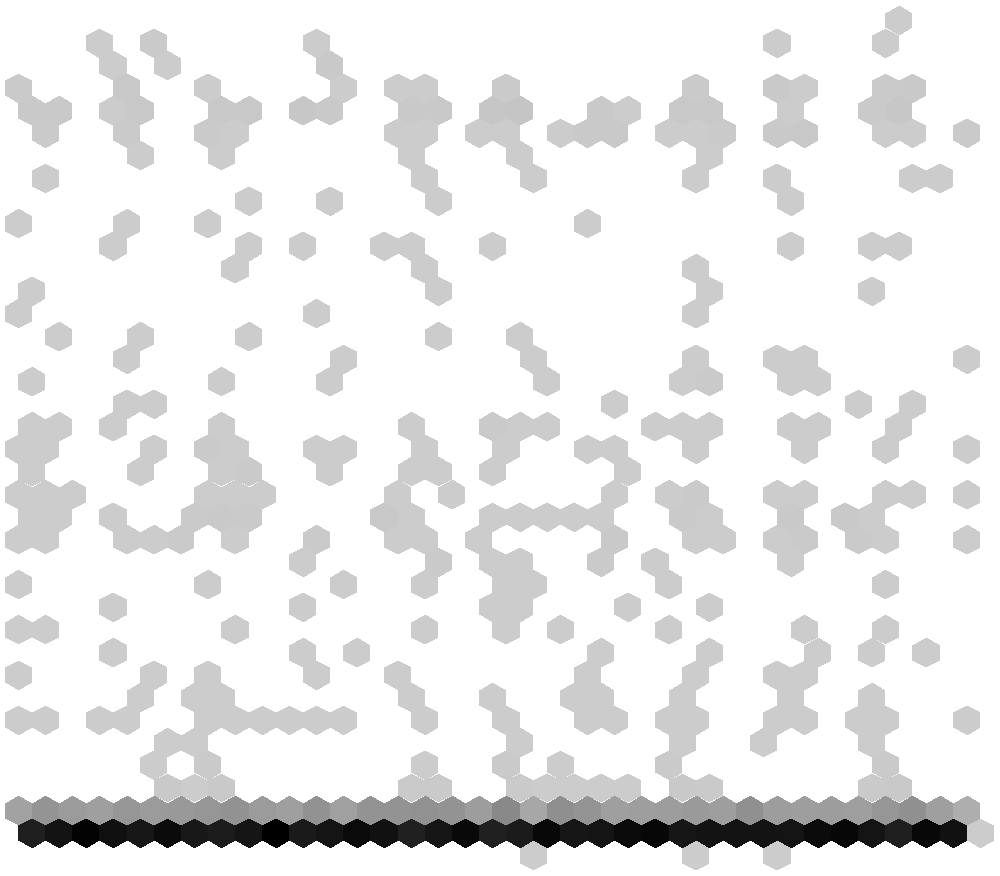
en las ejecuciones de prueba de Docker eran más altas que todas las muestras en Linux de metal desnudo. Por último, la correlación lineal de energía y tiempo de ejecución para Linux y Docker de metal desnudo fue de 0.8303 y 0.9885 respectivamente.

5.3 PostgreSQL

Para esta prueba, tuvimos tres distri de consumo de energía - butiones: Linux de metal desnudo, con SSL desactivado; Linux de metal desnudo, , con SSL habilitado; y Docker, con SSL deshabilitado. No sólo estamos probando la diferencia ence entre Docker y bare-metal, sino que también estamos introduciendo la diferencia entre el cifrado de las conexiones nections en el bare-metal también. Las tres distribuciones de energía se muestran en la Figura 6. En la Figura 7 se proporciona una densidad de potencia. Usando la ing prueba Shapiro- Wilk, determinamos que las tres muestras se distribuyen normalmente, con el valor p más pequeño de 0,54 para la distribución de energía de Docker. Por lo tanto, usamos las pruebas t de Student emparejadas por pares para comparar cada distribución con las demás. La línea base (Linux con SSL deshabilitado) es significativamente diferente, tanto para Docker con SSL deshabilitado, como con Linux con SSL habilitado, con valores p cercanos a cero. Curiosamente, Do- cker con SSL deshabilitado no es significativamente diferente en comparación con Linux con SSL habilitado, con un valor p de 0.15. Esto implica que el equilibrio entre en- las conexiones de cifrado con SSL es similar al equilibrio entre el uso de Docker sin cifrado.

Para entender el tamaño del efecto, usamos d. Cohen's d de Cohen compara los medios de las dos muestras distribuidas normalmente, teniendo en cuenta su desviación estándar agrupada para determinar el desplazamiento [5]. Más grande

Linux Docker



150 W

140 W

130 W

Poder

120 W

110 W

100 W

0 s 200 s 400 s 600 s 0 s 200 s 400 s 600 s

Hora

Figura 3: Gráfica de densidad de mediciones de vataje en todas las pruebas inactivas se ejecuta con el tiempo.

340.000 J

330,000 J

Energía

320.000 J

310.000 J



Linux Docker

resultados indican una diferencia mayor en los medios. La comparación de PostgreSQL con SSL deshabilitado en Linux de metal desnudo frente a la misma configuración en Do- cker produce una d d d de Cohen muy grande de 1.55. Sin embargo, simplemente activar SSL en Linux sin metal, las pruebas contra Docker con SSL deshabilitado producen el tamaño de efecto más pequeño obtenido en este artículo: 0.31. Este corrobo- califica los hallazgos de Chowdhury et al. [4] que el simple uso de SSL/TLS tiene un efecto significativo en la con- sumpción de energía . La diferencia entre Linux de metal desnudo frente a habilitar SSL en la misma configuración también tiene un tamaño de efecto grande, con d de Cohen calculado para ser 1.32.

5.4 Redis

La distribución del consumo de energía para ejecutar redis-bench en Linux y dentro de Docker se muestra en la Figura 8. Se proporciona una densidad de potencia en la Fig- ure 10. Usando la prueba Shapiro-Wilk, ambas muestras se distribuyen normalmente. Comparamos las distribu- ciones usando la prueba t de un estudiante y obtuvimos valores p

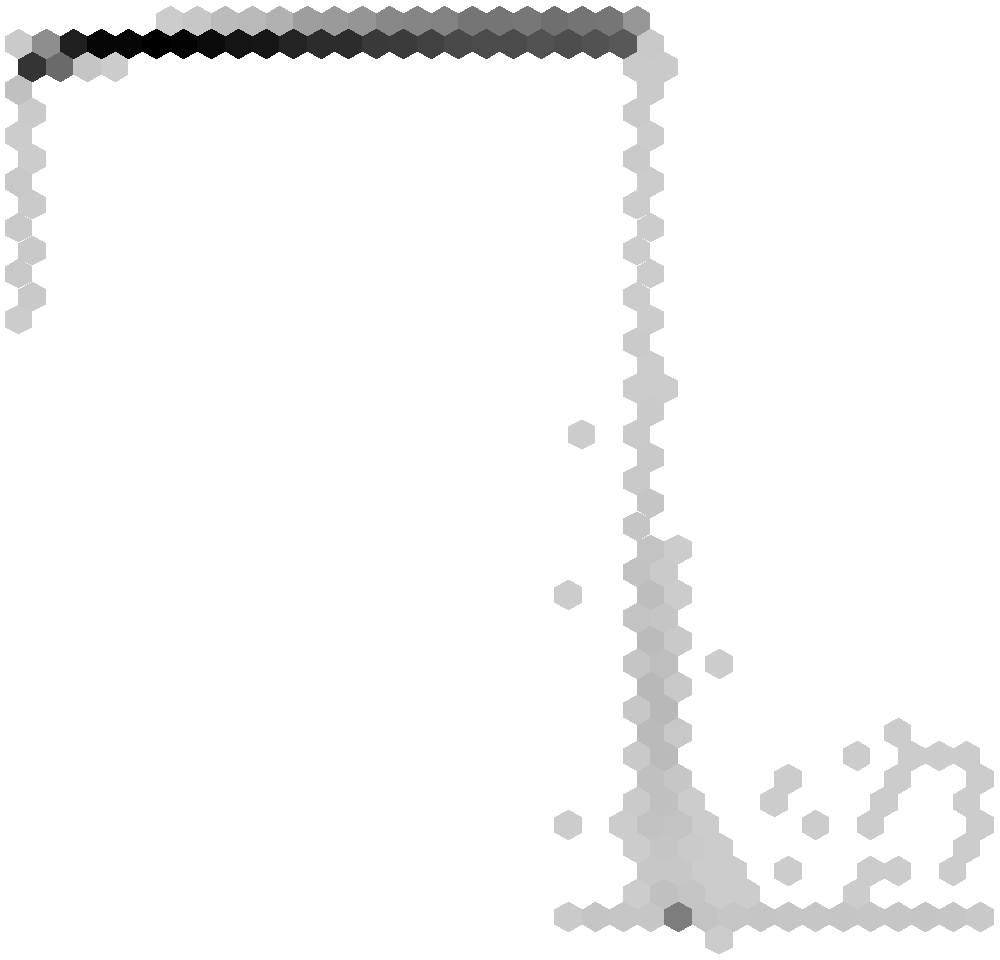
Figura 4: Gráfica de violín del consumo de energía en el

Experimento de WordPress

cerca de cero. Usando la d de Cohen, obtuvimos un enorme tamaño de fec de 11.31. Por lo tanto, este experimento muestra la mayor diferencia entre ejecutar en Docker en lugar de ejecutar en Linux sin sistema operativo.

La correlación lineal de la energía con el tiempo

Linux Docker



300 W

250 W

200 W

Poder

150 W

100 W

0 s 500 s 1000 s 15000 s 500 s 1000 s 1500 s

Hora

Figura 5: La gráfica de densidad de las mediciones de vataje en todas las carreras de WordPress a lo largo del tiempo.

30.000 J

29,600 J

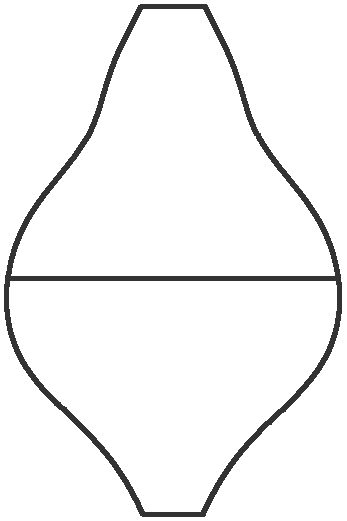
Energía

0.996 y 0.995 para Linux y Docker, respectivamente. Dada la muy alta correlación de energía con el tiempo, también comparamos la cantidad de tiempo que tomó completar cada prueba (Figura 9). Dado que el tiempo transcurrido es mayor en Docker, el consumo de energía será mayor, a menos que la potencia utilizada en Docker sea drásticamente menor, lo que no es el caso (figura 10).

6 Discusión

29.200 J

28.800 J



Linux Docker

ning En la figura 2 se muestra que el hecho de ejecutar el servicio Docker consume mucho más energía inactiva que sin Docker. El fondo estibado pro- cess explica la diferencia en el consumo de energía. Recuerde que Docker no es necesario para la ización de contenedores;; en su lugar, Docker proporciona una conveniente estructura en la nube para ejecutar aplicaciones en contenedores en Linux. Sin embargo, dockerd, el servidor Docker, escrito en el período del lenguaje de programación Go, se activa ódicamente para hacer el trabajo, incluso si se trata de contenedores activos cero de trabajo. Uso de perf top -p

Figura 6: Gráfica de violín del consumo de energía en el

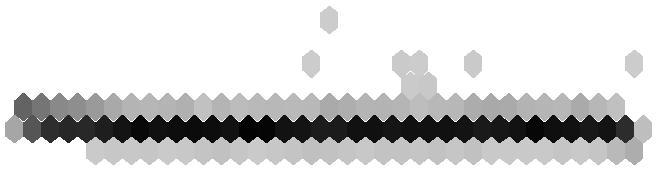
Prueba de PostgreSQL

$(pgrep dockerd) encontramos que el dockerd era

llamar periódicamente a funciones relacionadas con la programación

y recolección de elementos no utilizados en Go (por ejemplo, runtime.find- runnable, runtime.scanobject, runtime.heapBi- tsForObject, runtime.greyobject).

Linux Linux w/SSL Docker



160 W

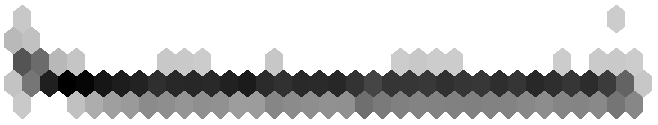
140 W

Poder

120 W

100 W

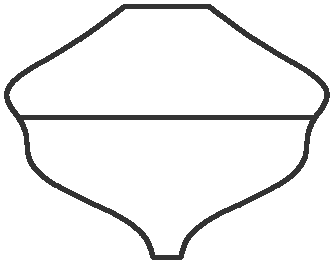
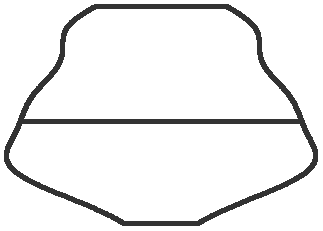
0 s 50 s 100 s 150 s 200 s 0 s 50 s 100 s 150 s 200 s 0 s 50 s 100 s 150 s 200 s



Hora

Figura 7: Gráfica de densidad de mediciones de vataje en todas las carreras de pgbench con el tiempo.

116.000 J



112.000 J

650 s

108,000 J

Energía

625 s

104.000 J

600 s

Linux Docker

Hora

Linux Docker

Figura 8: Gráfica de violín de consumo de energía que ejecuta el punto de referencia de Redis.

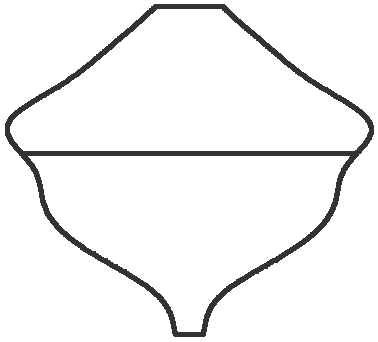
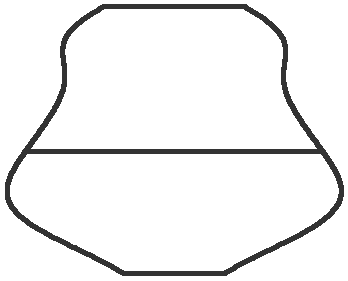


Figura 9: Gráfica de violín del tiempo transcurrido ejecutando el punto de referencia Re- dis. El tiempo transcurrido puede explicar la diferencia de energía (Figura 8)

Linux Docker



180 W

160 W

Poder

140 W

120 W

0 s 200 s 400 s 600 s 0 s 200 s 400 s 600 s

Hora

Figura 10: La gráfica de densidad de las mediciones de vataje en todas las carreras de Referencia de Redis a lo largo del tiempo.

Una posible estrategia de implementación de servicios consiste en crear:se hanquedo en las redes virtuales en las que cada microservicio se encuentra en su propio contenedor. Solo se requerirán servicios públicos-facing (de los cuales debería haber pocos) para utilizar cualquier tipo de cifrado por conexión, según lo proporcionado por SSL/TLS. Nuestros resultados muestran que, mientras que PostgreSQL en Docker utiliza más energía en comparación con la misma configuración en Linux, el efecto no es muy grande en comparación con la ejecución de PostgreSQL en Linux con el cifrado activado. En ese caso, ejecutar Post- greSQL dentro de contenedores, con comunicación entre contenedores sin cifrar puede ser realmente una opción más eficiente en- ergy.

Usando strace -c, medimos el tiempo invertido en las llamadas del sistema que ejecutan la aplicación redis-bench. Encontramos que tanto en Linux de metal desnudo como en Do- cker,, el servidor de Redis estaba llamando principalmente a write() (alrededor del 82% de todas las llamadas al sistema). Un punto de referencia de 32–39 segundos indujo alrededor de 1,7 millones de llamadas de sys- tem tem write(). La diferencia notable es que el servidor de Redis que se ejecuta dentro de un contenedor Docker pasó más del doble de tiempo haciendo escrituras (93,94 milisegundos) en lugar de ejecutar el servidor en Linux sin metal (44,08 milisegundos gastados en write()). Esto explica una pequeña parte del tiempo de ejecución más largo en Docker (y, por lo tanto, un mayor consumo de energía), aunque no se acerca a explicar la gran brecha en el tiempo de ejecución.

7 Amenazas a la validez

La validez de la construcción En general, el uso de marcos de referencia no modela necesariamente el uso real de las aplicaciones. Esto es especialmente cierto cuando no ha habido ninguna investigación sobre lo que sería un uso típico realista de estas aplicaciones, como fue el caso aquí. El trabajo futuro debe comenzar descubriendo lo que es representativo del uso típico para cada uno de los casos de prueba (un perfil), o un punto de referencia utilizando datos y acciones del mundo real, si es posible.

Docker tiene una serie de opciones de configuración que seleccionan las redes y el sistema de archivos. Probablemente, cualquier administrador que implemente Docker en producción ajustaría esta configuración ampliamente. Como tal, nuestro uso de valores predeterminados "listos para usar" (desplegando directamente desde la imagen de Docker Hub mediante un comando como docker run postgres:latest) no es representativo de los verdaderos de- ployments con Docker.

Cada una de las aplicaciones estudiadas sólo servía a un solo host. Cada herramienta de benchmarking proporcionó sup- port para simular múltiples clientes, y estas fea- tures se utilizaron en todas las pruebas. La calidad de los clientes multi-ple simulados de un solo cliente cuando com- pared a los usuarios del mundo real es desconocida y por lo tanto puede no enfatizar de manera realista las aplicaciones. Además, los servidores sólo usaban un solo gigabit Ethernet

conexiones, donde las implementaciones reales pueden ver varias conexiones de red que comparten la carga de solicitudes.

Validez interna Uno puede poner en duda la precisión y fiabilidad de las mediciones de potencia obtenidas del Watts Up? pro. Otra amenaza a la validez es que dejamos servicios como OpenSSHy OpenVPN ejecutándose en el System-Under-Test, cuyo uso de energía también se incluye en todo el sistema mea- surements. Por lo tanto, los números exactos pueden no ser en- dicative de cargas reales, pero, después de tomar varias muestras de energía, las comparaciones pueden dar una idea de la dif- ferences. SSH y VPN sólo se utilizaron para configur- ing las máquinas antes de que se ejecutaran las pruebas; ninguno del tráfico en ninguna de las pruebas utilizó SSH o utilizó la misma interfaz de red que la VPN.

Validez externa Las aplicaciones seleccionadas como casos de prueba no se aplican necesariamente a otras aplicaciones- ciones,, incluso de tipo similar. Las generalizaciones son difíciles de extraer de un conjunto tan pequeño de aplicaciones. Incluso diferentes versiones de la misma aplicación tienen perfiles de energía diferentes [3, 22], especialmente cuando la carga realiza diferentes llamadas al sistema operativo. Los par- vinculaciones externos deben tener en cuenta los recursos requeridos por su aplicación para evaluar mejor las consecuencias del uso de Docker.

Por último, el System-Under-Test que usamos solo representa una única configuración de máquina. Tener varias plataformas de prueba que difieren en rendimiento y arquitectura permitiría hallazgos más generalizados..

8 Conclusión

En este artículo, comparamos el consumo de energía de varias cargas de trabajo que se ejecutan dentro de contenedores administrados por Docker y en Linux "bare-metal". Después de casi

2 días y 20 horas de tiempo total recopilando mediciones de potencia, encontramos que, en todos los casos, las cargas de trabajo que se ejecutan en Docker tienen una sobrecarga de energía mensurable. Simplemente running dockerd inactivo induce una diferencia de 2 vatios en la potencia media, y por lo tanto un aumento en la energía con el tiempo. Sin embargo, el aumento en el consumo de energía puede atribuirse principalmente al rendimiento en tiempo de ejecución. En el caso de Redis y WordPress, el aumento de la energía se puede atribuir al aumento en el tiempo de ejecución, por lo que la disminución en el rendimiento explica el aumento en el consumo total de energía.

Los equipos de operaciones deben decidir cuál es más im- portante: sostenibilidad y consumo de energía y

rendimiento entiempo de ejecución de la reducción del uso de recursos mediante el empleo de Linux sin sistema operativo, o el aislamiento de procesos y la capacidad de mantenimiento de las aplicaciones en contenedores de Docker. Ahorrar energía y calor es importante para algunos escenarios,sinembargo, el costo humano de mantenimiento puede superar con creces el tiempo de ejecución, la energía y los costos de calefacción de la menor ineficiencia de Docker.

Referencias

[1] Arijs, P. Estadísticas de uso de Docker: Mayor adopción por parte de las empresas y para el uso de la producción. http://www.coscale.com/blog/docker-usage- estadísticas-aumento-adopción-por-empresas-

[parasu uso,](http://www.coscale.com/blog/docker-usage-statistics-increased-adoption-by-enterprises-and-for-production-use)  [julio](http://www.coscale.com/blog/docker-usage-statistics-increased-adoption-by-enterprises-and-for-production-use) de 2016. (Consultado el 12/21/2016).

[2] Canonical Ltd. Linux ContenedoresLinux. [https:](https://linuxcontainers.org/)

[linuxcontainers.org/de](https://linuxcontainers.org/) febrero de 2017. (Ac-

cesado el 02/06/2017).

[3] Chowdhury, S. A., y Hindle,, A. GreenO- racle: Estimando el consumo de energía del software con la corpora de medición de energía. En Proceed- ings of the 13th International Conference on Mining Software Repositories (Nueva York, NY, EE.UU., 2016), MSR '16, ACM, pp. 49–60.

[4] Chowdhury, S. A., Sapra, V., y Hindle,A. Eficiencia energética del lado del cliente de HTTP/2 para desarrolladores de aplicaciones web y móviles. En 2016 IEEE

23a Conferencia Internacional sobre Software Anal- ysis,, Evolución y Reingeniería (SANER) (marzo de 2016), vol. 1, pp. 529–540.

[5] Cohen, J. Análisis estadístico de poder para las ciencias ser- haviourales. havioural ladera. NJ: Lawrence Earl- baum Associates (1988).

[6] CoreOS, Inc. rkt, un motor de contenedor basado en estándares orientado a la seguridad. [https://](https://coreos.com/rkt)  [coreos.com/rkt,](https://coreos.com/rkt) febrero de 2017. (Accedido en

02/06/2017).

[7] Información general de Docker Compose. [https:](https://docs.docker.com/compose/overview/)

[docs.docker.com/compose/overview/,](https://docs.docker.com/compose/overview/)

septiembre de 2016. Consultado: 2016-09-02.

[8] Docker Hub. https://hub.docker.com/

[explorar/,](https://hub.docker.com/explore/) septiembre de 2016. Consultado: 2016-09-

02.

[9] Docker Inc. ¿Qué es docker? [https://](https://www.docker.com/what-docker#/VM)

[www.docker.com/what-docker#/VM ,,](https://www.docker.com/what-docker#/VM) Noviembre

2016. (Consultado el 12/21/2016).

[10] Ellis, C. S. El caso de la gestión de energía de alto nivel. En Temas calientes en Sistemas Operativos tems, 1999. Actas de la Séptima Tienda de Trabajo en (1999), IEEE, pp. 162–167.

[11] Felter, W., Ferreira, A., Rajamony, R., y Rubio, J. Un rendimiento actualizado de máquinas virtuales y contenedores Linux. En Análisis de Rendimiento de Sistemas y Software (ISPASS), 2015 IEEE International Sym- posium On (2015), IEEE, pp. 171–172.

[12] Ferranti, M. Encuesta: 96% de aumento en el uso de la producción de con- tainer durante el año pasado . clus- terhq. https://clusterhq.com/2016/06/16/  [encuesta de contenedores/](https://clusterhq.com/2016/06/16/container-survey/), junio[de](https://clusterhq.com/2016/06/16/container-survey/) 2016. (Accedido en

01/30/2017).

[13] Ferrara, J. Contenido de ejemplo de WP.

https://wordpress.org/plugins/wp- [example-content/,](https://wordpress.org/plugins/wp-example-content/) septiembre de 2016.

[14] Gupta, A., Zimmermann, T., Bird, C., Na- gappan, N., Bhat, T., y Emran, S. Des-estoy femiendo patrones de energía en el desarrollo de software. Microsoft Research Microsoft Corpora- tion One Microsoft Way Redmond, WA 98052 (2011).

[15] Hindle, A. Minería verde : Una metodología de relacionarse así queel cambio ftware al consumo de energía. IEEE, pp. 78–87.

[16] Linux Kernel Organization, Inc. Kvm. <http://www.linux-kvm.org/page/Main_Page>de noviembre [de](http://www.linux-kvm.org/page/Main_Page) 2016. (Consultado el 02/06/2017).

[17] McCullough, J. C. , Agarwal, Y., Chan- drashekar, J., Kuppuswamy, S., Snoeren,A. C. , y Gupta, R. K. Evaluar la eficacia de la potencia basada en modelos characteriza- ción. En USENIX Conf Técnico Anual (2011), vol. 20.

[18] Morabito, R. Power Consumo de energía de Virtu- Tecnologías de alización: una Investigación Empírica - tion. arXiv preimpresión arXiv:1511. 01232 (2015).

[19] Acerca de—PostgreSQL. [https://www.](https://www.postgresql.org/about/) [postgresql.org/about/,,](https://www.postgresql.org/about/) 2016. Accedido:

2016-09-09.

[20] El desarrollo global de PostgreSQL

Grupo. pgbench(1), PostgreSQL 9.5.4 ed., 2016.

[21] Redis. [http://redis.io/,](http://redis.io/) septiembre de 2016. Cesado cessed: 2016-09-02.

[22] Romansky,, S., y Hindle,, A. Sobre la mejora de la minería verde para el software de reconocimiento de energía analy- hermana. En Actas de la 24a Conferencia Internacional Anual on Ingeniería de Ciencias de la Computación y Software (Riverton, NJ, EE.UU., 2014), CAS- CON '14, IBM Corp., pp. 234–245.

[23] Shea, R., Wang, H., y Liu, J. Consumo de energía de máquinas virtuales con transacciones de red: Medición y mejoras. En IEEE INFOCOM 2014 - IEEE Confer- ence on Computer Communications (apr 2014), pp. 1051–1059.

[24] TPC. TPC-B. [http://www.tpc.org/tpcb/,](http://www.tpc.org/tpcb/)

1990. (Consultado el 01/25/2017).

[25] Tsung. [http://tsung.erlang-projects.org/,](http://tsung.erlang-projects.org/)

2016. Consultado: 2016-09-07.

[26] van Kessel, J., Taal,, A., y Grosso, P.

Eficiencia energética de la virtualización basada en hipervisor frente a la virtualización basada en contenedores. Uni- versidad de Amsterdam (2016).

[27] Vasiá, N., Barisits, M., Salzgeber, V., y Kostic, D. Hacer que las aplicaciones de clúster sean conscientes de la energía. En Procedimientos de la 1a tienda de trabajo en Control Automatizado para Datacenters y Nubes (Nueva York, NY, EE. UU., 2009), ACDC '09, ACM, pp. 37–42.

[28] W3Techs.com. Uso de sistemas de envejecimiento del contenido para sitios web. [https:](https://w3techs.com/technologies/overview/content_management/all/)

w3techs.com/technologies/overview/ [content\_management/all/,](https://w3techs.com/technologies/overview/content_management/all/) septiembre de 2016. Consultado: 2016-09-02.

[29] ¿Watts Up? medidores de carga de enchufe. [https:](https://www.wattsupmeters.com/secure/products.php?pn=0&wai=322&more=4)

www.wattsupmeters.com/secure/ [products.php?pn?0&wai-322&more-4,](https://www.wattsupmeters.com/secure/products.php?pn=0&wai=322&more=4) 2016. Consultado: 2016-09-09.

[30] Watts Up Pro. http://www.vernier.com/  [productos/sensores/wuwu-pro/,](http://www.vernier.com/products/sensors/wu-pro/) 2016. (Consultado el 12/21/2016).

[31] Acerca de—WordPress. https://wordpress.org/

[sobre/,](https://wordpress.org/about/) 2016. Consultado: 2016-09-05.

[32] Instalación de WordPress: Codex de WordPress. https://codex.wordpress.org/Installing\_  [WordPress,](https://codex.wordpress.org/Installing_WordPress) 2016. (Consultado el 12/22/2016).

[33] Xu, C., Zhao, Z., Wang, H., Shea, R., y Liu, J. Eficiencia energética de las ma- barbillas virtuales de la nube: Desde el patrón de tráfico y CPU Affin- Ity Perspectives. IEEE Systems Journal PP, 99 (2015), 1–11.

[34] Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P. , Mao, Z. M. , y Yang, L. Estimación precisa de la energía en línea y generación de modelos de energía basada en el comportamiento de la batería au- tomática para teléfonos inteligentes. En Actas de la octava conexión ference internacional IEEE/ACM/IFIP sobre el codesign de hardware/software y la síntesis sys- tem (2010), ACM, pp. 105–114.