

# What is the runtime performance cost of a Docker container?

Asked 7 years, 3 months ago   Active 1 month ago   Viewed 185k times

608



218



I'd like to comprehensively understand the run-time performance cost of a Docker container. I've found references to [networking anecdotally being ~100µs slower](#).

I've also found references to the run-time cost being "negligible" and "close to zero" but I'd like to know more precisely what those costs are. Ideally I'd like to know what Docker is abstracting with a performance cost and things that are abstracted without a performance cost. Networking, CPU, memory, etc.

Furthermore, if there are abstraction costs, are there ways to get around the abstraction cost. For example, perhaps I can mount a disk directly vs. virtually in Docker.

linux

docker

performance

virtual-machine

containerd

Share Edit Follow

edited May 12 at 23:06



CrazyPylor

6,525 ● 2 ● 39 ● 63

asked Feb 19 '14 at 18:19



Luke Hoersten

6,975 ● 3 ● 18 ● 18

- 4 possible duplicate of [Is there a formula for calculating the overhead of a Docker container?](#) – Golo Roden Feb 19 '14 at 18:22
- 2 @GoloRoden that question is similar but not exactly the same. I'm looking for latency costs with reasons like "networking is being passed through an extra layer" whereas that question's accepted answer is more about measuring the costs of the container + app. – Luke Hoersten Feb 19 '14 at 18:29
- 3 Okay, that's right. I retracted my close vote. – Golo Roden Feb 19 '14 at 18:43
- 10 I'm glad you posted it though. That question didn't come up in my search. The measurement/metrics article is super useful: [blog.docker.io/2013/10/gathering-lxc-docker-containers-metrics](http://blog.docker.io/2013/10/gathering-lxc-docker-containers-metrics) – Luke Hoersten Feb 19 '14 at 22:15
- 2 This is a good session titled "Linux Containers - NextGen Virtualization for Cloud" telling performance metrics by comparing docker, KVM VM and bare metal: [youtube.com/watch?v=a4oOAVhNLjU](https://www.youtube.com/watch?v=a4oOAVhNLjU) – shawnzhu May 22 '14 at 3:49

## 3 Answers

Active

Oldest

Votes

▲  
539  
▼

An excellent 2014 IBM research paper "[An Updated Performance Comparison of Virtual Machines and Linux Containers](#)" by Felter et al. provides a comparison between bare metal, KVM, and Docker containers. The general result is: **Docker is nearly identical to native performance and faster than KVM in every category.**

✓

The exception to this is Docker's NAT—if you use port mapping (e.g., `docker run -p 8080:8080`), then you can expect a minor hit in latency, as shown below. However, you can now use the host network stack (e.g., `docker run --net=host`) when launching a Docker container, which will perform identically to the Native column (as shown in the Redis latency results lower down).

🔄

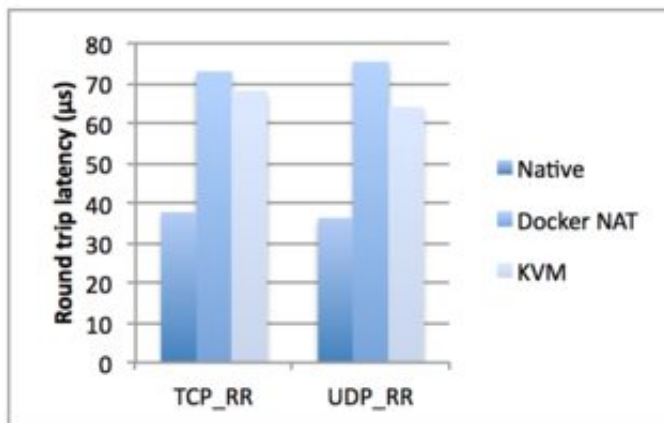
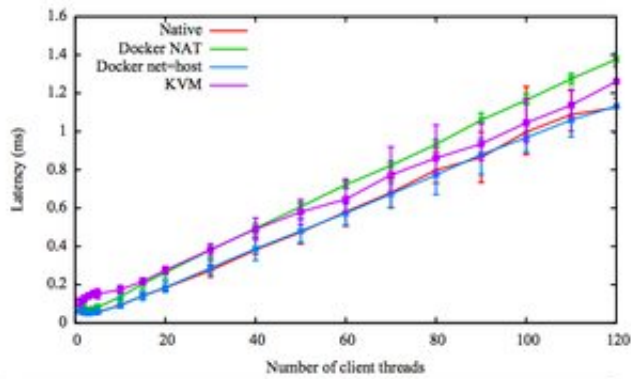


Fig. 3. Network round-trip latency (μs).

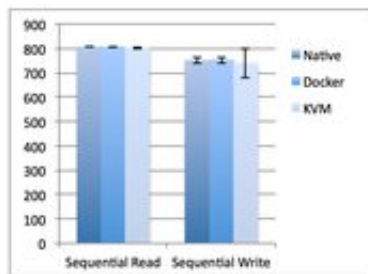
They also ran latency tests on a few specific services, such as Redis. You can see that above 20 client threads, highest latency overhead goes Docker NAT, then KVM, then a rough tie between Docker host/native.



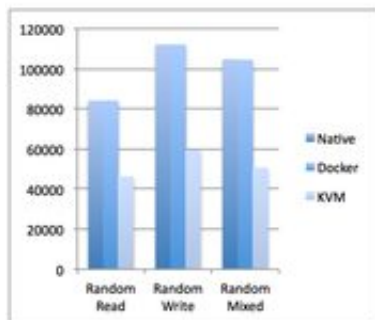
**Figure 14.** Average latency (in ms) of operations on different Redis deployments. Each data point is the arithmetic mean obtained from 10 runs.

Just because it's a really useful paper, here are some other figures. Please download it for full access.

Taking a look at Disk I/O:

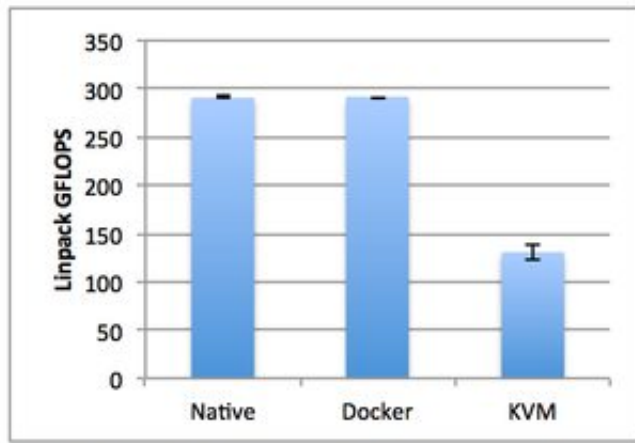


**Figure 10.** Sequential I/O throughput (MB/s).



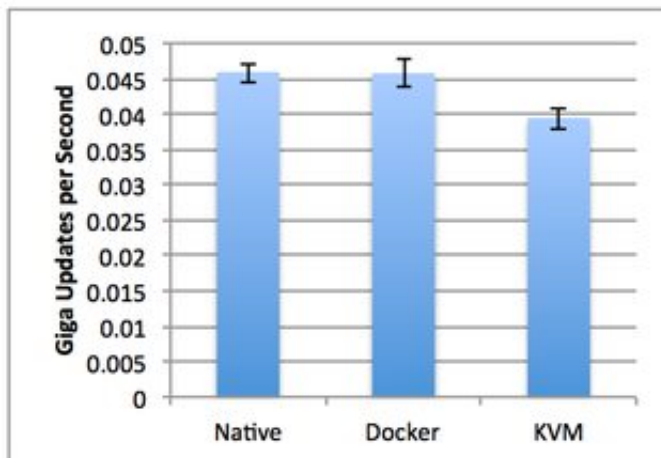
**Figure 11.** Random I/O throughput (IOPS).

Now looking at CPU overhead:



**Figure 1.** Linpack performance on two sockets (16 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.

Now some examples of memory (read the paper for details, memory can be extra tricky):



**Figure 4.** RandomAccess performance on a single socket (8 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.

Share Edit Follow

edited Mar 14 '20 at 2:49



ib.

25.4k ● 10 ● 72 ● 97

answered Oct 1 '14 at 20:26




Hamy

18.2k ● 14 ● 66 ● 94

- 25 As for the linpack numbers given in the paper... frankly, I find them hard to believe (not that I disbelieve that they're what linpack emitted, but that I disbelieve that the test was genuinely measuring nothing but floating-point performance as performed). The major overhead from KVM is in the userspace hardware emulation components (which only apply to *non-CPU* hardware); there's significant overhead around memory paging... but raw floating-point? I'd want to look at what was actually going on there -- perhaps excessive context switches. –

Charles Duffy Jan 8 '15 at 3:38 

- 
- 3 Correction for current Docker CLI syntax: `--net=host` (two dashes) and `-p 8080:8080` (lower case 'p') for NAT. – [bk0](#) Dec 17 '15 at 19:55 
- 
- 10 The cited IBM paper seems too focused on network IO. It never addresses context switches. We looked at LXC and had to quickly abandon it due to increased non-voluntary context switches resulting in degraded application processing. – [Eric](#) May 5 '17 at 3:09
- 
- 6 I'm also curious about filesystem operations -- directory lookups, for instance, are a place where I'd expect to see overhead; block-level reads, writes and seeks (which the given charts focus heavily on) *aren't*. – [Charles Duffy](#) Oct 14 '17 at 12:48
- 
- 48 I love charts with the same shade color. It's so easy to distinguish – [Viktor Joras](#) May 27 '19 at 10:51
- 

▲  
132  
▼

Docker isn't virtualization, as such -- instead, it's an abstraction on top of the kernel's support for different process namespaces, device namespaces, etc.; one namespace isn't inherently more expensive or inefficient than another, so what actually makes Docker have a performance impact is a matter of what's actually *in* those namespaces.

---



Docker's choices in terms of how it configures namespaces for its containers have costs, but those costs are all directly associated with benefits -- you can give them up, but in doing so you also give up the associated benefit:

- Layered filesystems are expensive -- exactly what the costs are vary with each one (and Docker supports multiple backends), and with your usage patterns (merging multiple large directories, or merging a very deep set of filesystems will be particularly expensive), but they're not free. On the other hand, a great deal of Docker's functionality -- being able to build guests off other guests in a copy-on-write manner, and getting the storage advantages implicit in same -- ride on paying this cost.
- DNAT gets expensive at scale -- but gives you the benefit of being able to configure your guest's networking independently of your host's and have a convenient interface for forwarding only the ports you want between them. You can replace this with a bridge to a physical interface, but again, lose the benefit.
- Being able to run each software stack with its dependencies installed in the most convenient manner -- independent of the host's distro, libc, and other library versions -- is a great benefit, but needing to load shared libraries more than once (when their versions differ) has the cost you'd expect.

And so forth. How much these costs actually impact you in your environment -- with your network access patterns, your memory constraints, etc -- is an item for which it's difficult to provide a generic answer.

Share Edit Follow

answered Sep 26 '14 at 21:18



Charles Duffy

237k ● 34 ● 308 ● 359

- 3 This is a good answer but I'm looking for more specific numbers and benchmarks. I'm familiar with the cost of cgroups but Docker is more than that as you've pointed out. Thanks a lot for the answer. – [Luke Hoersten](#) Apr 2 '15 at 21:55
- 9 Sure. My point is that any generalized benchmarks you find will be of very limited applicability to any specific application -- but that's not to say I disagree with folks trying to provide them, but merely that they should be taken with a heaping tablespoon of salt. – [Charles Duffy](#) Apr 2 '15 at 22:03 ✎
- 1 In that manner you could say that KVM "is not a virtualization it is simply an abstraction on top of x86 virtual technology calls". – [Vad](#) Aug 19 '16 at 9:10
- 12 @Vad, there's consensus agreement, going back decades (to IBM's early non-x86 hardware implementations!), that providing abstraction directly on the hardware layer is unambiguously virtualization. Consensus for terminology around kernel-level namespacing is considerably more fragmented -- we could each point to sources favoring our individual views -- but frankly, there are useful technical distinctions (around both security and performance characteristics) that moving to a single term would obscure, so I'm holding my position until and unless contrary industry consensus is reached. – [Charles Duffy](#) Oct 6 '16 at 17:02 ✎
- 1 @LukeHoersten, ...right, it's not the cgroups that have a significant cost, it's much more the contents of the network and filesystem namespaces. But *how much those costs are* depends almost entirely on how Docker is configured -- which specific backends you're using. Bridging is much, *much* cheaper than Docker's default NAT, for example; and the various filesystem backends' performance overhead also varies wildly (and in some cases, the amount of overhead depends on usage patterns; overlayfs variants can be much more expensive with big directories modified through multiple layers f/e). – [Charles Duffy](#) Jan 15 '19 at 17:00

▲ Here's some more [benchmarks](#) for `Docker based memcached server` versus `host native memcached server` using Twemperf benchmark tool <https://github.com/twitter/twemperf> with 5000 connections and 20k connection rate

26

▼ Connect time overhead for docker based memcached seems to agree with above whitepaper at roughly twice native speed.



## Twemperf Docker Memcached

```

Connection rate: 9817.9 conn/s
Connection time [ms]: avg 341.1 min 73.7 max 396.2 stddev 52.11
Connect time [ms]: avg 55.0 min 1.1 max 103.1 stddev 28.14
Request rate: 83942.7 req/s (0.0 ms/req)
Request size [B]: avg 129.0 min 129.0 max 129.0 stddev 0.00
Response rate: 83942.7 rsp/s (0.0 ms/rsp)
Response size [B]: avg 8.0 min 8.0 max 8.0 stddev 0.00
Response time [ms]: avg 28.6 min 1.2 max 65.0 stddev 0.01
Response time [ms]: p25 24.0 p50 27.0 p75 29.0
Response time [ms]: p95 58.0 p99 62.0 p999 65.0

```

## Twemperf Centmin Mod Memcached

```

Connection rate: 11419.3 conn/s
Connection time [ms]: avg 200.5 min 0.6 max 263.2 stddev 73.85
Connect time [ms]: avg 26.2 min 0.0 max 53.5 stddev 14.59
Request rate: 114192.6 req/s (0.0 ms/req)
Request size [B]: avg 129.0 min 129.0 max 129.0 stddev 0.00
Response rate: 114192.6 rsp/s (0.0 ms/rsp)
Response size [B]: avg 8.0 min 8.0 max 8.0 stddev 0.00
Response time [ms]: avg 17.4 min 0.0 max 28.8 stddev 0.01
Response time [ms]: p25 12.0 p50 20.0 p75 23.0
Response time [ms]: p95 28.0 p99 28.0 p999 29.0

```

Here's [benchmarks using memtier benchmark tool](#)

## memtier\_benchmark docker Memcached

4	Threads				
50	Connections per thread				
10000	Requests per thread				
Type	Ops/sec	Hits/sec	Misses/sec	Latency	KB/sec
-----	-----	-----	-----	-----	-----
Sets	16821.99	---	---	1.12600	2271.79
Gets	168035.07	159636.00	8399.07	1.12000	23884.00
Totals	184857.06	159636.00	8399.07	1.12100	26155.79

## memtier\_benchmark Centmin Mod Memcached

4	Threads				
50	Connections per thread				
10000	Requests per thread				
Type	Ops/sec	Hits/sec	Misses/sec	Latency	KB/sec
-----	-----	-----	-----	-----	-----
Sets	28468.13	---	---	0.62300	3844.59
Gets	284368.51	266547.14	17821.36	0.62200	39964.31
Totals	312836.64	266547.14	17821.36	0.62200	43808.90

[Share](#) [Edit](#) [Follow](#)

answered Apr 19 '15 at 20:31

**p4guru**

1,156 ● 2 ● 15 ● 22

- 
- 2 They compare two different builds of memcached, and also one of them in docker, other outside of docker, aren't they? – [san](#) Sep 26 '15 at 21:31
- 
- 5 Are these results with host networking or bridge networking in docker? – [akaHuman](#) Jan 11 '16 at 7:31
- 
- 21 With such big stddevs these measurements do not show any representable data **avg 200.5 min 0.6 max 263.2 stddev 73.85** – [Sergey Zhukov](#) Sep 24 '16 at 17:37 ✎
- 

**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.



[GET NEW REDDIT](#)  [MY SUBREDDITS](#) [HOME](#) [POPULAR](#) [ALL](#) [RANDOM](#) [FRIENDS](#) [USERS](#) | [ASKREI](#) [EDIT](#) [malik](#) (109) |    

  **docker**

COMMENTS

 This is an archived post. You won't be able to vote or comment.

 26 



### Docker performance on macos vs native linux

self.docker  
Submitted 2 years ago by lppier 

Hi, I will be using docker quite often in my new job, so basically this boils down to whether I want to get a macbook or a windows machine dual-booted to ubuntu.

For those doing docker on a day-to-day basis, does macos suffice for your needs?

Or would I better off with a pure linux setup?

I'm personally more used to macos as I have been an app developer prior. But now I'm doing machine learning and cloud development. (the NVIDIA GPU on a windows laptop if I get one would be a plus)

Thanks.

[44 comments](#) [share](#) [save](#) [hide](#) [give award](#) [report](#) [crosspost](#)

all 44 comments  
sorted by: best ▼

 **themightychris** • 17 points 2 years ago

Docker for Mac's access to the host filesystem is incredibly slow. It's a huge factor when you're trying to use a docker env for live development. Git within docker will take forever to check status on a host directory

Linux as a dev workstation is 1000% smoother and faster

[permalink](#) [embed](#) [save](#) [give award](#)

 **jacurtis** • 3 points 2 years ago

Can't emphasize this enough. Linux Docker is almost seamless compared to Mac Docker which feels like a lot of overhead on the system.

Linux Docker runs on almost no RAM and barely hits the CPU. Mac Docker uses quite a bit of resources in comparison. I found that Mac Docker uses at least 2gb of RAM and about 10% CPU

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)



**docker**  
134,631 readers [Join](#)

N/A

Submit Text

This post was submitted on 25 Apr 2019

**26 points** (100% upvoted)

shortlink: <https://redd.it/bh8rpf>

 **reddit premium**  
Get an ad-free experience with special benefits, and directly support Reddit.  
[Get Reddit Premium](#)

### Subreddit Info

 322 currently online

 Show my flair on this subreddit. It looks like:

**Docker** is an open-source project to easily create lightweight, portable, self-sufficient containers from any application. The same container that a developer builds and tests on a laptop can run at scale, in production, on VMs, bare metal, OpenStack clusters, public clouds and more.

MODERATORS

↑ [-] **themightychris** • 1 point 2 years ago

↓ Yeah you've got a virtual machine in the middle bringing a whole external level of abstraction to filesystem and network access. When I have to work on my Macbook I mostly just ssh into my Linux desktop over ZeroTier and run my containers there

If I was confident the trackpad would still work as well I'd rather run Linux on my Macbook. With my day spent in chrome and vscode Mac brings only negatives to the table

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **kylemhall** • 2 points 2 years ago

↓ This is painfully true. What I do as a work-around is to edit the files from my Mac and *not* edit them from within the container itself. The same applies to git commands which will take microseconds on my mac and tens of seconds in the container.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **sagespidy** • 1 point 2 years ago

↓ why don't you mount docker dir to host and make changes in dir itself ?

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **themightychris** • 1 point 2 years ago

↓ That's what we're talking about, it's only host mounts that have the problem, unmounted volumes are fine

You can make changes in the dir itself on the host, but your tools running inside the container that need to scan through the working tree will be >10x slower.

Linux is the only host for Docker where you're not also running a VM in the middle and actually get to enjoy all the lightness and quickness of containers that make them better than VMs

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **j\_schmotzenberg** • 1 point 2 years ago

↓ I'm not sure this is true. I thought this would speed up performance but it didn't have any effect at all. Maybe you need to completely unmount all volumes from a container. I still had my SSH keys mounted in the container and the like.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **aledalgrande** • 1 point 1 year ago

↓ Yes you need zero volumes shared with the host. There is this utility called docker-sync that you can then use, but I never was able to make it work.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **crassus01** • 8 points 2 years ago

↓ I have been using Docker on Mac for a year or so, switched to Docker on Linux about two months ago.

MESSAGE THE MODS

[jimrhoskins](#)  
[ionrover2](#)

[about moderation team »](#)

[account activity](#)

This subreddit uses a customized version of the r/Naut theme.

Functionally, there is no difference. Startup is slower from cold on OSX because as other commenters pointed out, the underlying Linux VM needs to start up. Once it's running, it performs just fine (at least for my workloads; databases, web servers, ElasticSearch etc.)

When I used OSX I used to attribute the odd weird behaviour (Docker losing track of its network and needing restarting, that kind of thing) to the underlying VM later. Turns out that I have these same kind of small problems on Linux too.

YMMV but the differences are minor.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [\[-\] Carr0t](#) • -1 points 2 years ago

↓ Underlying Linux VM? I know Docker for Mac *used* to require you to run a virtual machine that it would connect to under the hood, but that hasn't been the case for a while now. Docker for Mac should be native now...

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] kylemhall](#) • 7 points 2 years ago

↓ Docker for Mac still uses a VM, it's just much more transparent now. Docker for Mac is a native MacOS X application that embeds a hypervisor (based on xhyve), a Linux distribution and filesystem and network sharing that is much more Mac native.

A lot of this only became possible in recent versions of OSX thanks to the Hypervisor.framework that has been bundled, and the hard work of mist64 who released xhyve (in turn based on bhyve in FreeBSD) that uses it.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] Carr0t](#) • 5 points 2 years ago

↓ I have learned a thing! I thought it was 'properly' native as I used to run a version where you had to have VMware Fusion or VirtualBox or whatever installed as well.

Thank you for correcting my misunderstanding :)

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] dragonmantank](#) • 1 point 2 years ago

↓ And keep in mind that this will almost always be the case. Containers execute their programs against the host kernel, so Linux binaries need a Linux kernel. Windows actually has two modes - Linux mode, which uses a VM like MacOS, and a Windows mode, which uses the native NT kernel. Linux binaries (99% of all containers) need a kernel, so they live in the VM environment under Hyper-V. If you have a Windows container, it can run without a VM.

You cannot, however, run a container on a host it was not built for. In theory someone could build a layer that allows the MacOS kernel to be used, but it would still be regulated to only running MacOS binaries. You would not be able to run a Linux container natively.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] GNUandLinuxBot • -7 points 2 years ago

↓ I'd just like to interject for a moment. What you're referring to as Linux, is in fact, GNU/Linux, or as I've recently taken to calling it, GNU plus Linux. Linux is not an operating system unto itself, but rather another free component of a fully functioning GNU system made useful by the GNU corelibs, shell utilities and vital system components comprising a full OS as defined by POSIX.

Many computer users run a modified version of the GNU system every day, without realizing it. Through a peculiar turn of events, the version of GNU which is widely used today is often called "Linux", and many of its users are not aware that it is basically the GNU system, developed by the GNU Project.

There really is a Linux, and these people are using it, but it is just a part of the system they use. Linux is the kernel: the program in the system that allocates the machine's resources to the other programs that you run. The kernel is an essential part of an operating system, but useless by itself; it can only function in the context of a complete operating system. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux. All the so-called "Linux" distributions are really distributions of GNU/Linux.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] Crotherz • 3 points 2 years ago

↓ Bad bot.


[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] Ariquitaun • 4 points 2 years ago\*

↓ Docker is native on linux, run via virtualisation elsewhere. Docker on mac is, and always will be, slow. File access from host mounted volumes is slow, CPU performance takes a significant hit. I/O bound and multithreaded workloads will be the worst affected. Forget about GPU access in docker on the mac without some serious fuckery involving essentially bypassing docker for mac and deploying your own VMWare solution.

I might be biased since I'm a long time desktop linux user, but I spent 7 months just last year working on a mac with docker and it was a painful and time-wasting experience.

Developer experience, in general, is superior in Linux due to native package management and whatnot. All the IDEs are available, all the libraries are there, any hardware is supported and pretty much anything you can do with a computer is possible. There's no walled garden, there's nothing the system won't allow you to do. The majority of devs I work with that use macs either come from Windows, or only have ever used mac, so that's what they benchmark developer experience against. They have no idea of what they're missing out on.

[permalink](#) [embed](#) [save](#) [give award](#) [\[-\] lppier\[S\]](#) • 1 point 2 years ago


Actually I was working on Centos purely during my previous job. And prior to that I was on mac platform, and I use macos at home. There are still advantages to the macos - easy access to word, for example. In enterprise, when documents come in, they are in word format, and the web word is still not up to par yet.

That said, I guess I would need a NVIDIA GPU eventually for training models..

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#) [\[-\] chafey](#) • 6 points 2 years ago

Get a windows machine and dual boot into ubuntu:

- 1) If you are doing ML, you probably want a nVidia GPU
- 2) Docker only supports GPU access on Linux
- 3) Modern Apple hardware only supports AMD GPUs


[permalink](#) [embed](#) [save](#) [give award](#) [\[-\] villelaitila](#) • 3 points 2 years ago

I run source code analysis processes, git repository data mining and related ML processes (in development environment). Docker for Mac performance is quite poor for this sort of IO intensive operations.


I have been using both Beta and Stable and haven't noticed big differences. But there was one big issue recently that made my laptop useless for most of the day. Docker image building consumed the last free 30 GB of my disk space and caused the filesystem go into readonly state. It was difficult to predict what would happen in the next boot because files could not be deleted to free up some space. Luckily it booted perfectly and I was able to fix the situation. Certainly Docker should have some logic to avoid this since OSX's journaling filesystem does not behave well when filled up totally.

Luckily our architecture at Softagram supports running most of the processes modularly in any Unix environments. That makes it easy to switch to host side if debugging processes with high performance requirements. It is also common to use GCE located Linux servers for similar tasks instead of using Docker for Mac.

It might make sense to switch to Linux laptop also because of the poor connectivity support in the recent Mac laptops.

[permalink](#) [embed](#) [save](#) [give award](#) [\[-\] lppier\[S\]](#) • 1 point 2 years ago

This is good info, thanks! Does something like Dell XPS 15 suffice for your use case?

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#) [\[-\] Arikuitaun](#) • 1 point 2 years ago

I'm on an XPS15 and couldn't be happier. I would recommend you go for the 1080p option though as one thing

linux sucks at is hidpi with external monitors that aren't.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **villelaitila** • 1 point 2 years ago

↓ Dont know that laptop. Good to know. Will check that if Mac dies...

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] [deleted] • 6 points 2 years ago

↓ I've got a 2017 MacBook Pro and the performance there is good enough to work with it.

However, wouldn't it be better to host docker machines on a server within your new company or even the cloud?

[permalink](#) [embed](#) [save](#)

↑ [-] **alc6379** • 2 points 2 years ago

↓ If you're doing development, you'd want it local, but if you're actually running an ML project, you're right-- the production workload should actually go on a server.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **overstitch** • 5 points 2 years ago

↓ Docker for Mac like other Docker solutions just runs a VM with Linux. I've never benchmarked it, but I've heard numerous people complain about how bloated and slow Docker for Mac. It does have some nice perks of one-Touch Kubernetes-but otherwise you get the pain of macOS accessory limitations (have to use special docks or actual Thunderbolt displays for multi-monitor when in the office. An Ubuntu compatible PC laptop is probably the cheaper, more capable and reliable option.

And if possible, don't dual boot unless you really do need Windows, it forces you to live the tooling and get a better level of familiarity with where your applications live. That all being said, if you need Windows containers, you have no choice but Windows and Hyper-V.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **Fenzik** • 2 points 2 years ago

↓ | now I'm doing machine learning **and cloud development**

Does your laptop's performance really matter if you're doing cloud development? Just use whatever you're comfortable with for writing code and then do the heavy lifting in the cloud.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **lppier[S]** • 1 point 2 years ago

↓ I'm thinking for some nlp training code, I would need to do it with an offline system in order not to incur expensive GPU server cloud charges?

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **Fenzik** • 2 points 2 years ago

↓ Depends on your needs and budget I guess. Cloud GPUs

aren't *that* expensive for big companies. But if you want to train locally on a GPU, then the MacBook is a no go

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **rennykoshy** • 2 points 2 years ago

↓ Docker on MacOS -- not the best experience. I've had more success running a VirtualBox VM running CentOS in headless mode and using that for docker testing /dev on my mac.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **lppier[S]** • 1 point 2 years ago

↓ This was something I was curious about - say I run Ubuntu in parallels, it should circumvent some of these performance issues you guys are talking about, no?

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **set2uk** • 2 points 2 years ago

↓ My older MacBook Pro runs awful with docker installed.

My advice...get any old P.O.S. server off ebay, install docker on top of Linux and connect to it on your LAN/WAN. That way you can have the shiny Mac and still have the performance.

I got a HP DL360 with HDDd for less than a £100 (warning: this server produces huge fan noise when it's hot)

My server runs samba so I can access source code directories on the LAN, which then bind mount during docker compose up.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **bateszi** • 2 points 2 years ago

↓ One thing that's made a big difference for me on MacOS is [docker-sync](#). Developing a large web application on my 2018 MBP, using Docker became slow enough that I needed an alternative and docker-sync has really helped, it just involves having the docker-sync process running before the container.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **benabbottnz** • 2 points 2 years ago

↓ I have a Mac Mini at work and use Docker. It only has 8GB RAM, and 2GB goes to Docker whether it uses it or not due to VM.

It works, but like, only just.

I'm considering installing Ubuntu on it so it doesn't chew up so much of my precious RAM.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **ashleyw** • 2 points 2 years ago

↓ Has anyone found a good solution for syncing third-party dependencies like `node_modules` back to the Mac's filesystem? Everything is fine until I have to `yarn install` or `yarn add` within the container/VM, and it takes 12+ minutes instead of the 60

seconds it takes if I don't sync the dependencies back. And I really need to do so to get a good Typescript experience in VSCode.

(The sister issue to syncing the files themselves is reliable inotify filesystem event proxying, so dev servers restart correctly when a relevant file changes.)

Things I've tried:

- Docker for Mac w/ delegated volumes (sloooooow...)
- docker-sync (often inconsistent)
- Vagrant/VirtualBox + NFS mounts (w/ [inotify proxy hacks..](#))
- Vagrant/VirtualBox + [reverse NFS mount](#) + rsync to copy code files (much like docker-sync, can become inconsistent)
- Rsyncing my monorepo's `package.json` and `yarn.lock` files recursively to a non-synced folder (or a temporary RAM disk!), installing dependencies there, and rsyncing `node_modules` back (quicker to install, way quicker, but ends up being slower by the time it's all synced back)
- Experimental filesystems/filesystem caches like [overlayfs](#), and [mccache](#) (fun to play with, very interesting, but ultimately you still must flush the contents to the synced drive at some point, which is still slow. Although one benefit this has over all the above is that you gain an intermediary step where your containers have the dependencies installed and ready to use super quick, it's just that they won't yet show up on the host/VSCode until you manually flush the files..which is still slow. Better than nothing I suppose!)

Looking to test out when I have time:

- [Mutagen](#) - Looks very interesting, aims to solve all the problems with "Development-focused design"
- [code-server](#) -- VSCode web IDE, i.e. running VSCode within the Linux VM and never needing to sync anything back to the host. Solves all the problems, at the expense of an awkward development process and concerns around losing uncommitted/unpushed code if the VM/container dies. Worth it..?

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [\[-\] Ariquitaun](#) • 2 points 2 years ago

↓ Short of switching to linux for dev, have you tried changing your volume binds so that you're in fact not mounting `node_modules` but only your code & config? You'd need to `yarn add` once (on your host) and `yarn install` (within docker).

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] ashleyw](#) • 2 points 2 years ago

↓ Yeah that's something I've tried in the past. Having to install dependencies twice is a little frustrating though. And there's always that occasion where the easiest way to debug a package is to modify its source, so ideally I'd like to keep that workflow.

Another option I tried was to `yarn install` on macOS and set flags to target Linux when compiling the binaries, which in theory I think should be possible, but is more trouble than it's worth when it comes to linking shared libraries.



I'm surprised this isn't a bigger issue in the Node world. I would imagine most devs use macOS or Windows, and an increasing number of them are using Docker or some kind of Linux environment in development, so the massive overhead of using effectively network shares becomes a big bottleneck in development.

Think my next port of call will be to trial a code-server workflow. Using a web IDE is a little alien but probably a small price to make my workflow more sane. (I'll be able to install a new 2KB dependency without it being a 15 minute ordeal!)

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] Arikuitaun](#) • 1 point 2 years ago

↓ This is an issue as well in the php world, for the exact same reasons, although dependency trees and the sheer number of small files is much lower than on a typical node project. I've felt the pain in both and had to hack my set up in the way I suggested, and un unhack it when I've had to debug libs. Realistically, mac is not really the right tool for this job.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] stixxie](#) • 2 points 1 year ago

↓ Have you tried:

- Using anonymous volumes for these kind of folders
- This vagrant/virtualbox/nfs solution:

<https://github.com/nijens/docker-host>

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [\[-\] ashleyw](#) • 1 point 1 year ago

↓ Yep, I tried both of those. Anonymous folders didn't enable the bi-directional syncing required and while NFS gave a slight improvement in performance it wasn't really worth the hassle.

Thankfully shortly after writing that comment discussing code-server as a potential option, the VSCode team released [Remote-SSH](#), which solved all my problems!

No matter the approach, syncing a folder between a OS X host and a Linux VM introduced way too much overhead. Remote-SSH bypasses the issue by allowing VSCode to install a headless server inside the VM. In the editor's file explorer I can see and manipulate all the files (incl. node\_modules) but they're not actually synced to the host at all. It runs full project search, Intellisense and third-party extensions via the VM's headless server too so you can't even tell that you're developing on a "remote" machine. It's the modern equivalent of SSHing into a machine and using VIM in your terminal.

- `yarn install` down from 12+ minutes to ~1 minute.
- No more flakey inotify events
- Massively improved the performance of application

boot times, bash scripts running in the VM, operations in VSCode and... basically everything which touches the disk.

Only downside is since the files are "stuck" in the VM they can no longer be backed up by Time Machine. Also if you manage to corrupt your VM it's difficult to get your files back (although this encourages frequent git pushes!)

But other than that, I think this is the only feasible solution to the problem if you want native Linux and Docker performance, but with the niceties of OS X.

[permalink](#) [embed](#) [save](#) [parent](#) [give award](#)

↑ [-] **markshust** • 2 points 2 years ago

↓ I see this time and time again. The performance is pretty much driven from filesystem IO.

I wrote a whole blog post about this. It's written in the view of Magento, which has an extremely large filesystem, so performance issues are exasperated. That said, after making tweaks to filesystem mounts I'm achieving 95% native speed with zero issues.

I do have bash helpers scripts which aid in the transfer of files to and from containers. Unfortunately this is needed until a new methodology for host bind mounts is invented, and this isn't specific to Docker but to VM's in general. Right now osxfuse and the delegated flag is the best we get (and it's really not that bad).

See my post for more details

<https://markshust.com/2018/12/30/docker-mac-filesystem-volume-mount-approach-performance/>

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **ratnose** • 4 points 2 years ago

↓ Since Docker is a Linux invention and is virtualized on all other platforms I would guess if it's got best performance on Linux. :)

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [-] **DeusOtiosus** • 2 points 2 years ago

↓ No matter the laptop, you'll never get great GPU performance.

I'm running a 2017 MBP and docker is pretty great. Since it's all development, I don't worry about performance, but I also don't find it lacking either. I don't have any GPU acceleration because there's simply no NVidia GPU. But that's fine, because I have a desktop with a proper 2080ti that does have a proper GPU running jupyter notebook. Turns it into a total non issue. I did the math, and the 2080ti, while expensive, does far more work than an aws GPU. If I ran it full tilt for 2 days straight, it does the same work that renting aws space for about the same purchase cost as the GPU. Or put another way, if I am simply training my models for 2 days on that GPU, it will pay for itself.

At the end of the day, I want the machine that I'm physically typing

into to be zero hassle. I can't be spending hours and hours customizing things or fixing things like I would with a Linux GUI. When I sit down, I need it working. If the whole thing shifts the bed, or I have hardware failure, I need to be able to bring it somewhere to get it fixed. While that's exceptionally rare, I know I'll get amazing support from Apple. Even if the whole thing is destroyed, it takes me less than 2 hours to restore from time machine and I'm back to exactly where I left off; no fussing around. I'm also no slouch when it comes to Linux. I've administered large Linux infrastructure, and even have code (15 years ago) in X.org, mplayer, and some other major OSS projects. So when I say, I see Linux on the desktop as purely experimental, I'm not some anti-Linux jagoff.

My GPU rigs, well those I can fuss with. If they're busted, then I can spend some time with them because they're not my primary entry. And you'll get far better price for performance on a desktop GPU than you'll ever get from a laptop. Leaving a long ML run churning on a laptop gets super toasty, and chains you to your desk. Instead, having the ML running on a dedicated machine means you can take the laptop and go elsewhere while it's running, or do something else.

So for my money, get a MBP as it lets you write apps; you will miss that if you've written apps before. Then get a cheaper desktop and throw in a bonkers desktop grade GPU.

[permalink](#) [embed](#) [save](#) [give award](#)

↑ [\[-\]](#) **progzos** • 2 points 2 years ago

↓ Get a real PC and single boot your favorite GNU+Linux OS! :p

[permalink](#) [embed](#) [save](#) [give award](#)

## about

[blog](#)  
[about](#)  
[advertising](#)  
[careers](#)

## help

[site rules](#)  
[Reddit help center](#)  
[reddiquette](#)  
[mod guidelines](#)  
[contact us](#)

## apps & tools

[Reddit for iPhone](#)  
[Reddit for Android](#)  
[mobile website](#)

## <3

[reddit premium](#)  
[reddit coins](#)  
[redditgifts](#)

Use of this site constitutes acceptance of our [User Agreement](#) and [Privacy Policy](#). © 2021 reddit inc. All rights reserved.  
REDDIT and the ALIEN Logo are registered trademarks of reddit inc.