Performance and consumption of workloads on bare metal / virtual guests / emulated guests

Christian Horn
IRCnet: globalc@#NetBSD
https://fluxcoil.net
Mastodon: https://chaos.social/@globalc

$ whoami

- Christian Horn
  (Better don’t) ask me about:
  - #Japanese (language | culture)
  - #Fedora
  - #BSD
  - #Cycling
  - #RSS-all-the-things
  - #Amiga
  - #HAMradio (JL1AYH)
  - #RetroHardware
  - #Demoscene
  - #SelfHost-as-much-as-possible

- What
  - Motivation:
    - I got a Apple MacBook M2. That’s aarch64, I’m emulating full x86-64 guests. It’s reasonably fast!
    - But: “qemu-system-x86-64 –cpu ?” lists dozens of cpu types!
      - → Performance differences?
    - Also: how fast are aarch64 guests emulated on x86-64?
    - Also: RISC-V is now first class citizen for Fedora. So are some emulated architectures performing better?
    - While we are at it: how much slower is emulation than virtualization?!

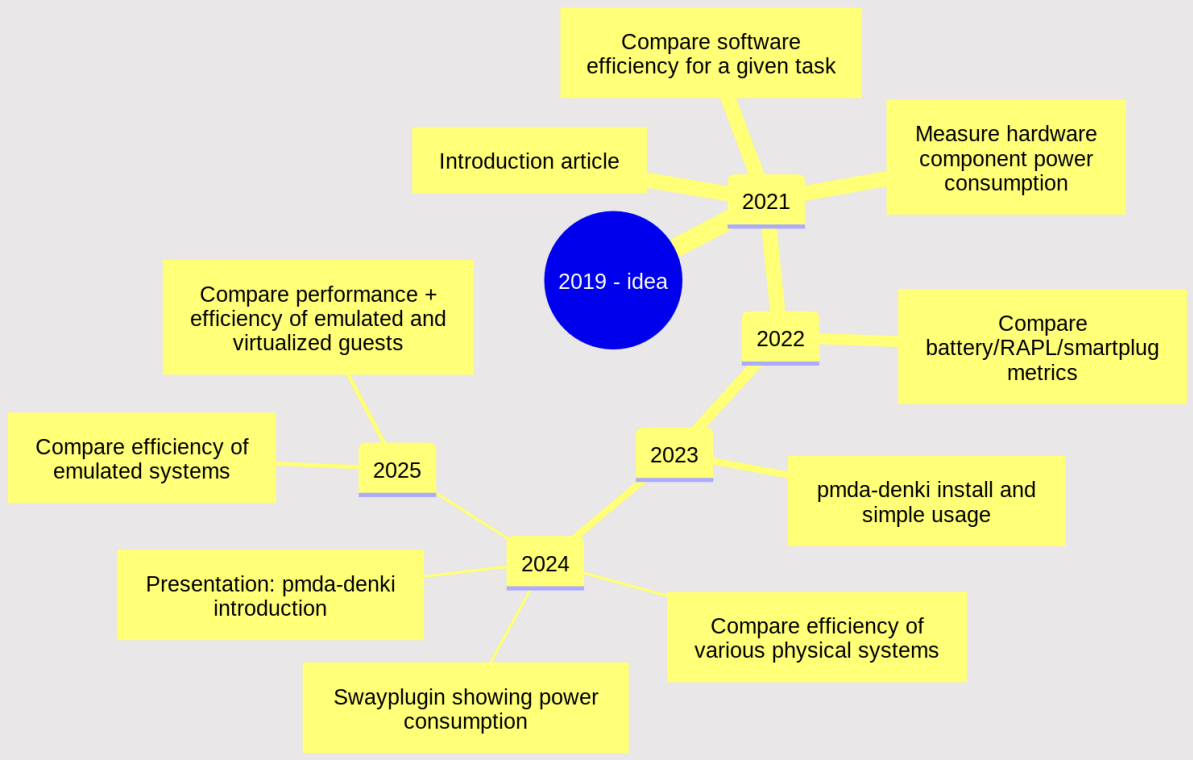
- Why?

  => IT history is growing, emulation requests are increasing.

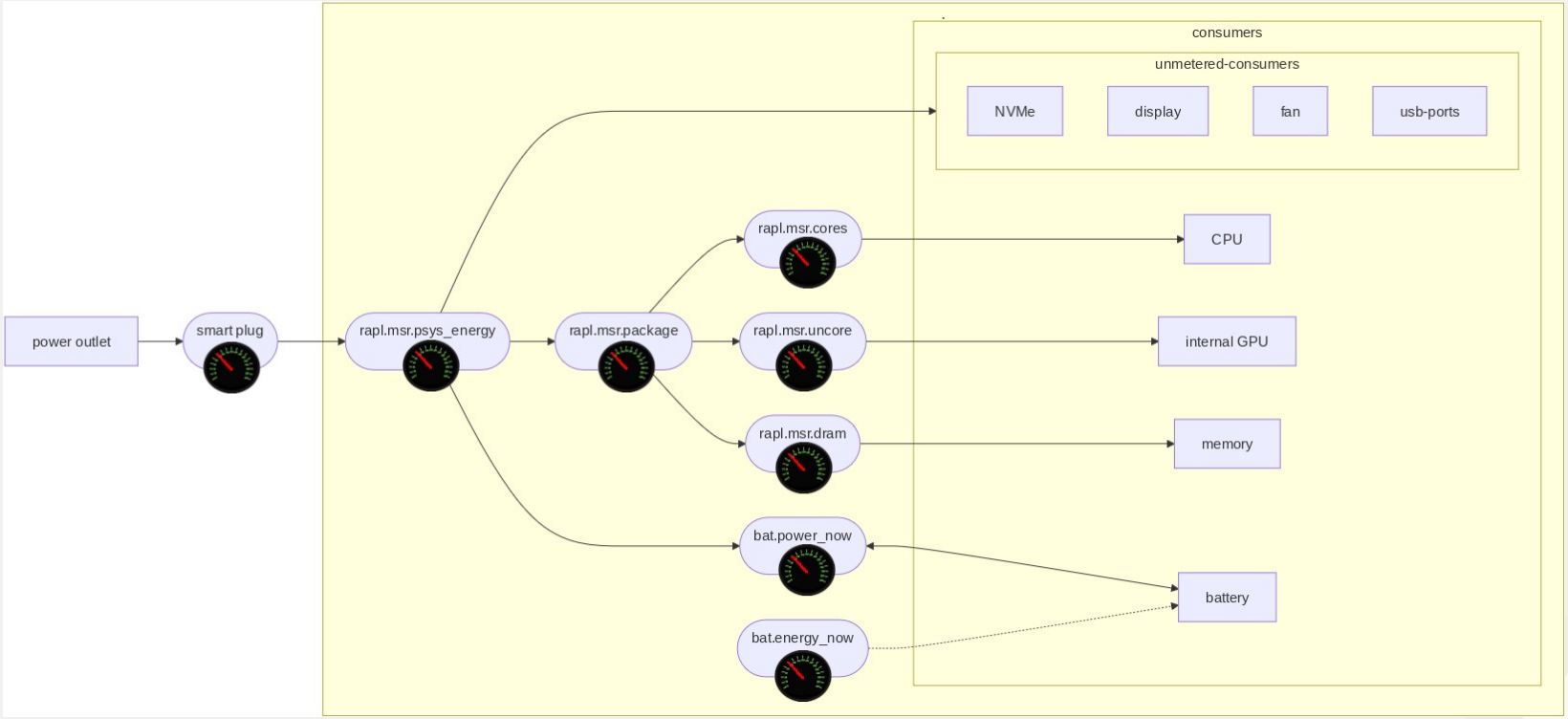
pmda-denki history

Stack:

- Ansible
- Python
- Performance Co-Pilot (PCP) w/ pmda-denki
- Bash



Sensors/metrics on an Intel system



What makes a good test workload?

- **compute power:** I measure how often a file can be uncompressed
- **network throughput:** iperf2 and iperf3 are used
- **memory I/O:** for this, I write block files with zeros into a RAM disk
- **disk I/O:** let’s read and write blocks of data in the guest, with and without sync

Selecting the best CPU workload

Workload	Short runtime?	Includes various workloads?	Easy setup?
SPEC suite	---	+++	---
OpenSSL's ("openssl speed")	-	-	++
BSD make world	---	+	-
bzcat uncompression ("bzcat <file.bz2>")	++	-	++

Architecture remarks 1/2

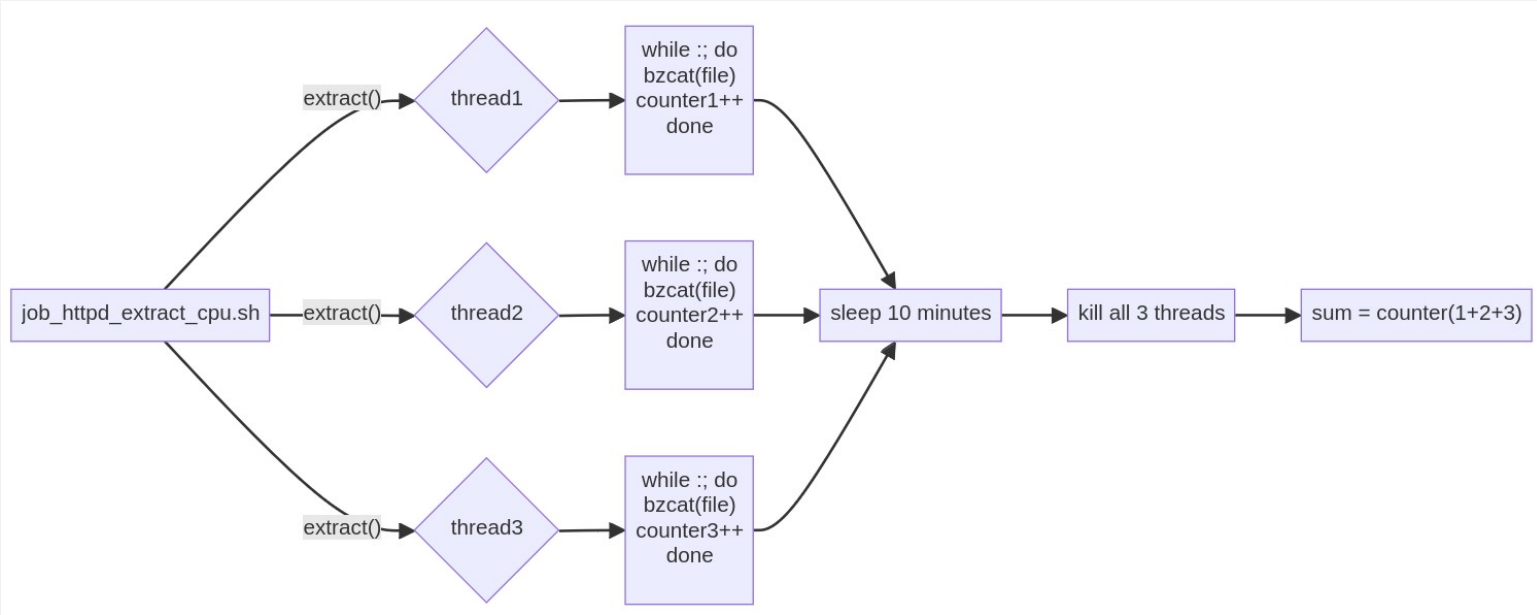
- **aarch64** is quite solid: emulating 16GB of memory is no issue. 32 cpus is also possible, but then the 8 cores of my hypervisor are 100% busy. I settled with emulating cpu cortex-a53, after cpu=cortex-a57 and cpu=max turned out slightly slower.

- The NetBSD **RISC-V** port is quite new, instead of doing an installation of the stable/released version 10.x I had to use the 10.99.12 development version. With that, I did not see signs of instability. My measure scripts use sudo, iperf3 and bash - for stable NetBSD on aarch64, sparc64 and so on I could fetch and install these as precompiled packages but here I compiled these myself, with pkgsrc.

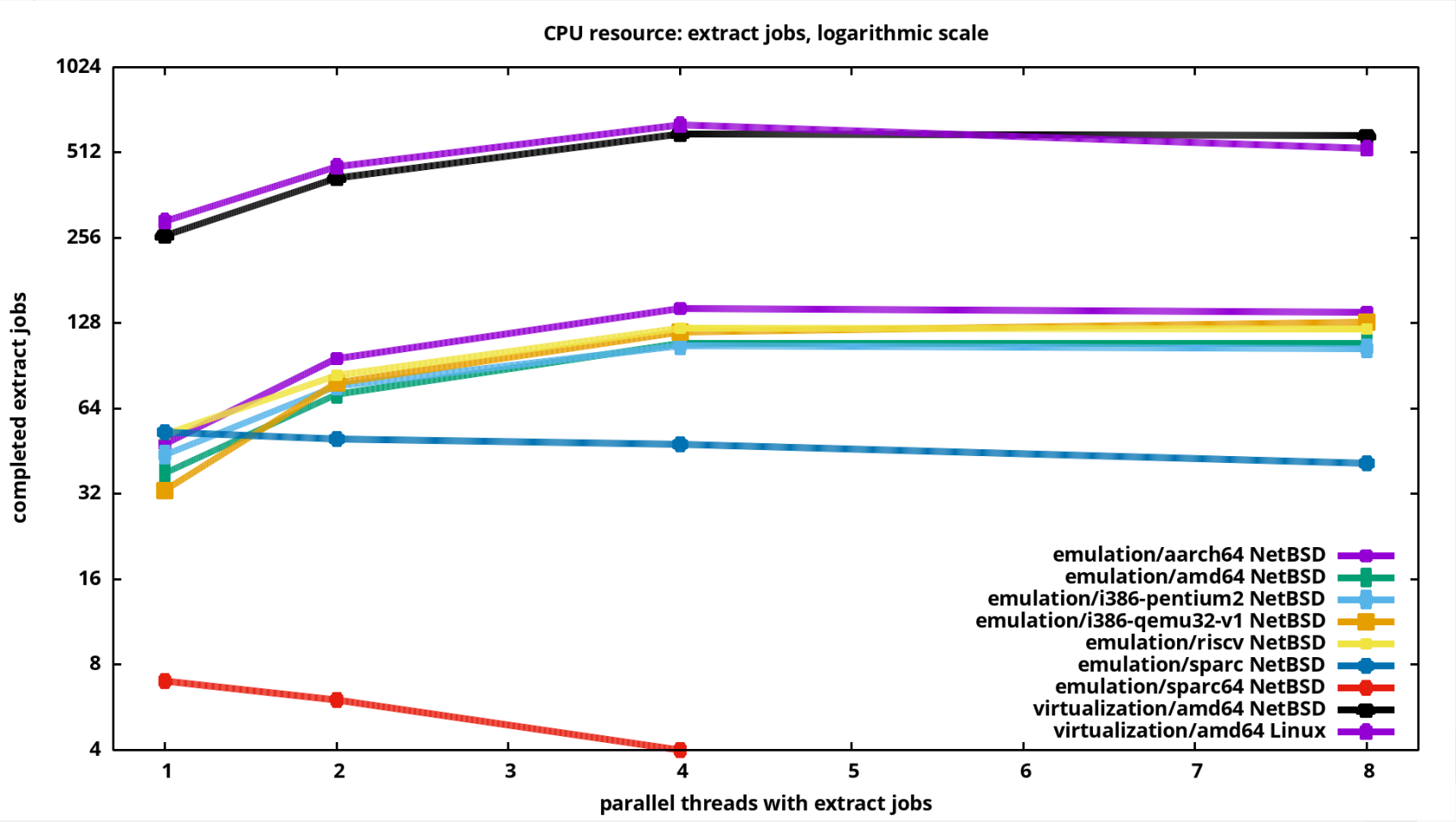
Architecture remarks 2/2

- **sparc:** Just got one cpu working
- **sparc64:** “issues” when NetBSD and FreeBSD are run
- **AMD64/x86_64** is of course matching my hypervisor for these tests, so besides emulating ("-cpu Icelake-server-noTSX") I did for reference also run NetBSD with KVM, so virtualized. I did also run it virtualized with KVM and Linux as guest, but then with a much more complex qemu config - not started by hand with “qemu-system-x86_64” but with a config suggested by libvirt.
- **i386** turned out also quite stable, I emulated and benchmarked 2 configs: “-cpu qemu32-v1” and “-cpu pentium2” to see how much they are different.

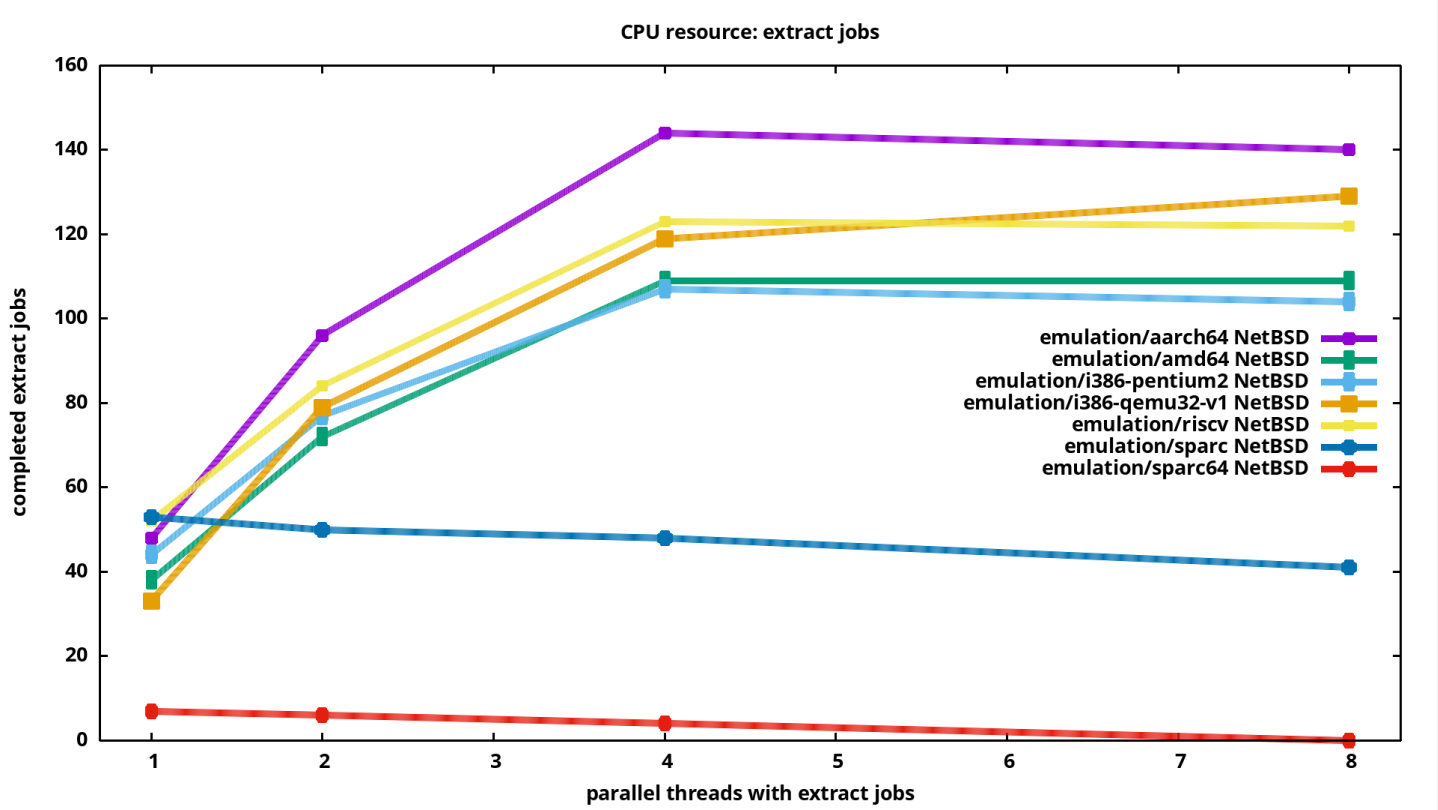
CPU resource: Job Loops



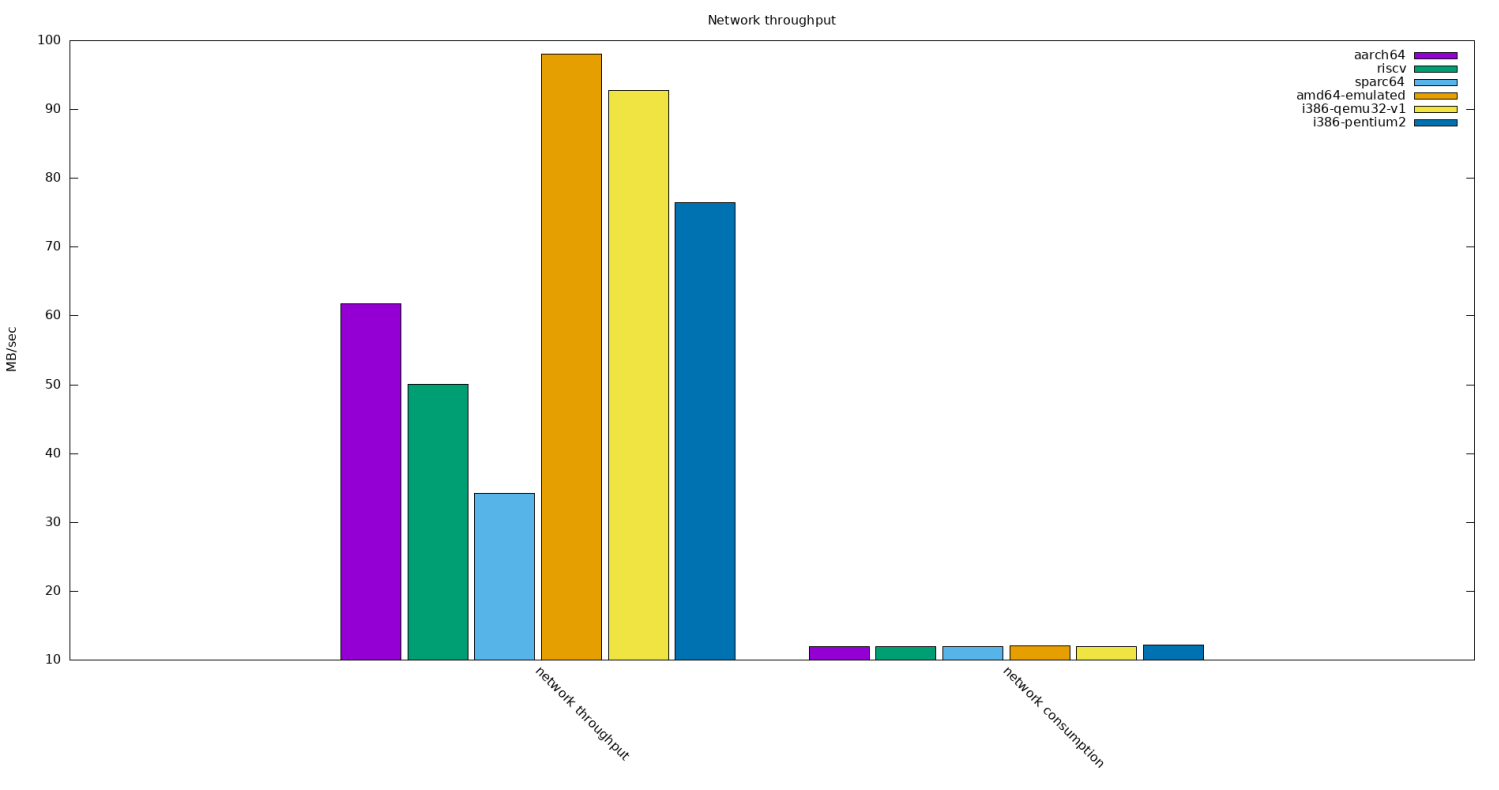
- Let’s start multiple loops, each constantly extracting data
- After 10 minutes count the completed extract operations


CPU resource: extract jobs, logarithmic scale
completed extract jobs
1024
512
256
128
64
32
16
8
4
1
2
3
4
5
6
7
8
parallel threads with extract jobs
emulation/aarch64 NetBSD
emulation/amd64 NetBSD
emulation/i386-pentium2 NetBSD
emulation/i386-qemu32-v1 NetBSD
emulation/riscv NetBSD
emulation/sparc NetBSD
emulation/sparc64 NetBSD
virtualization/amd64 NetBSD
virtualization/amd64 Linux

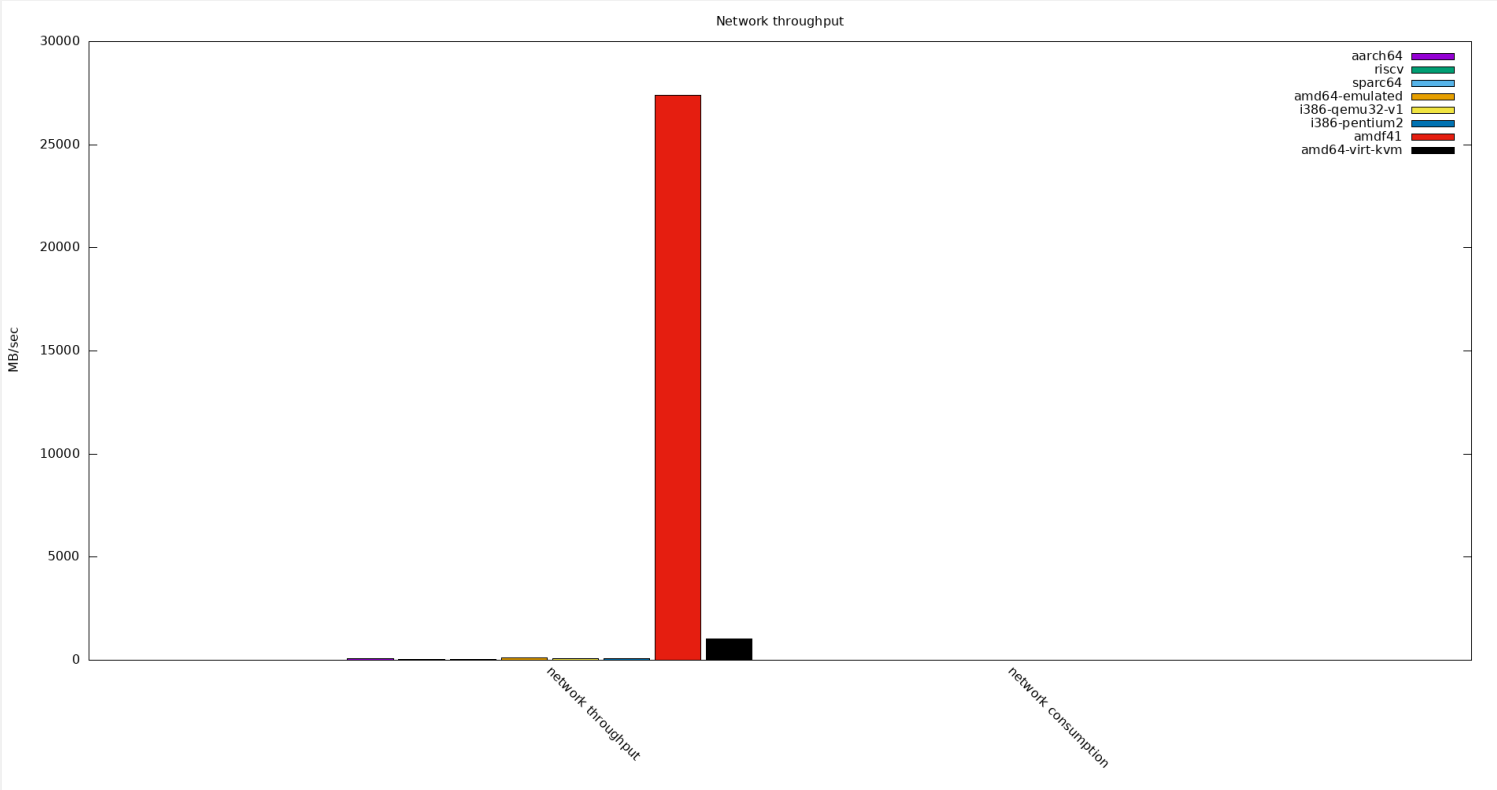
CPU: emulated guests only



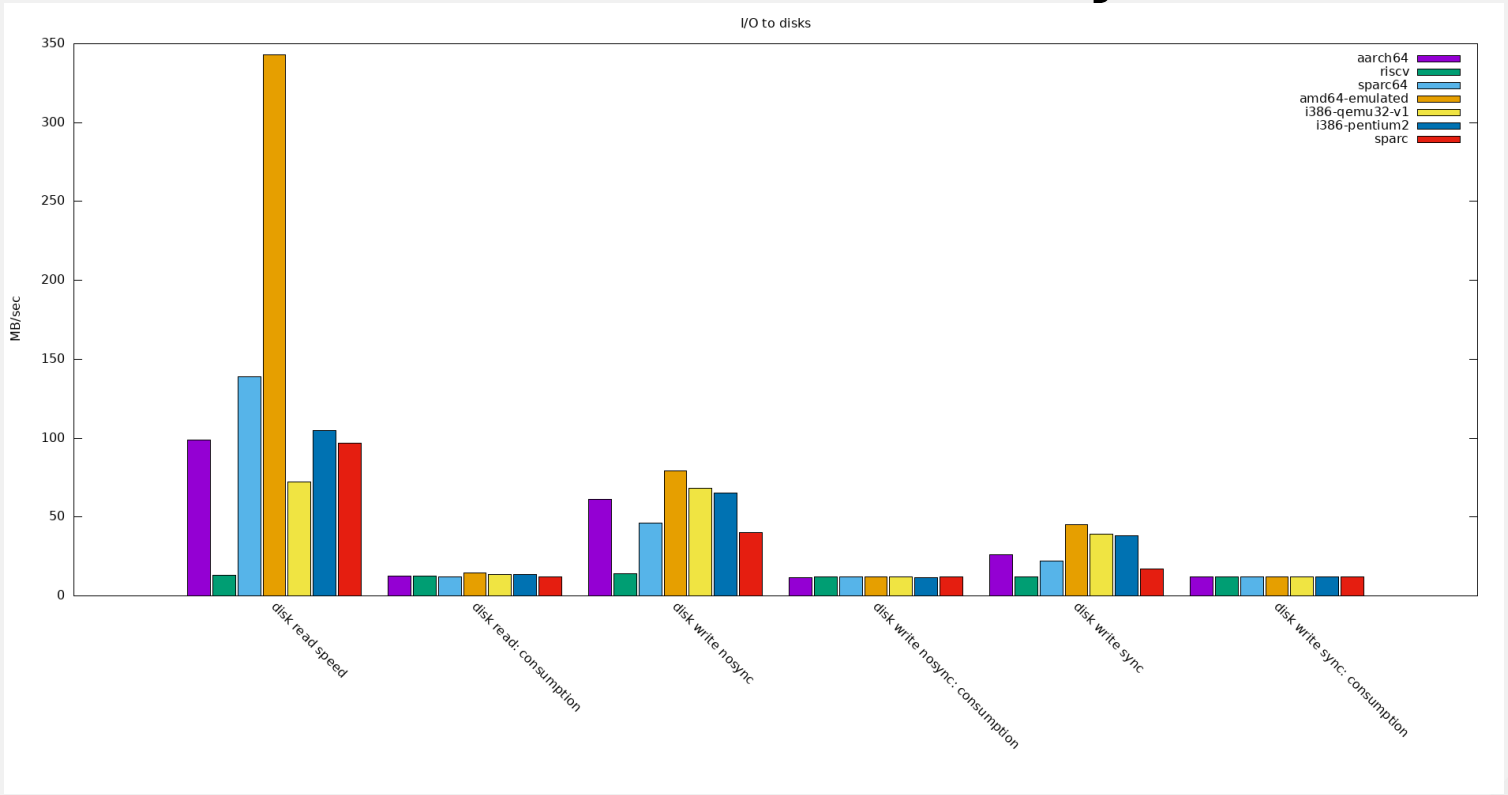
Network: emulated guests only



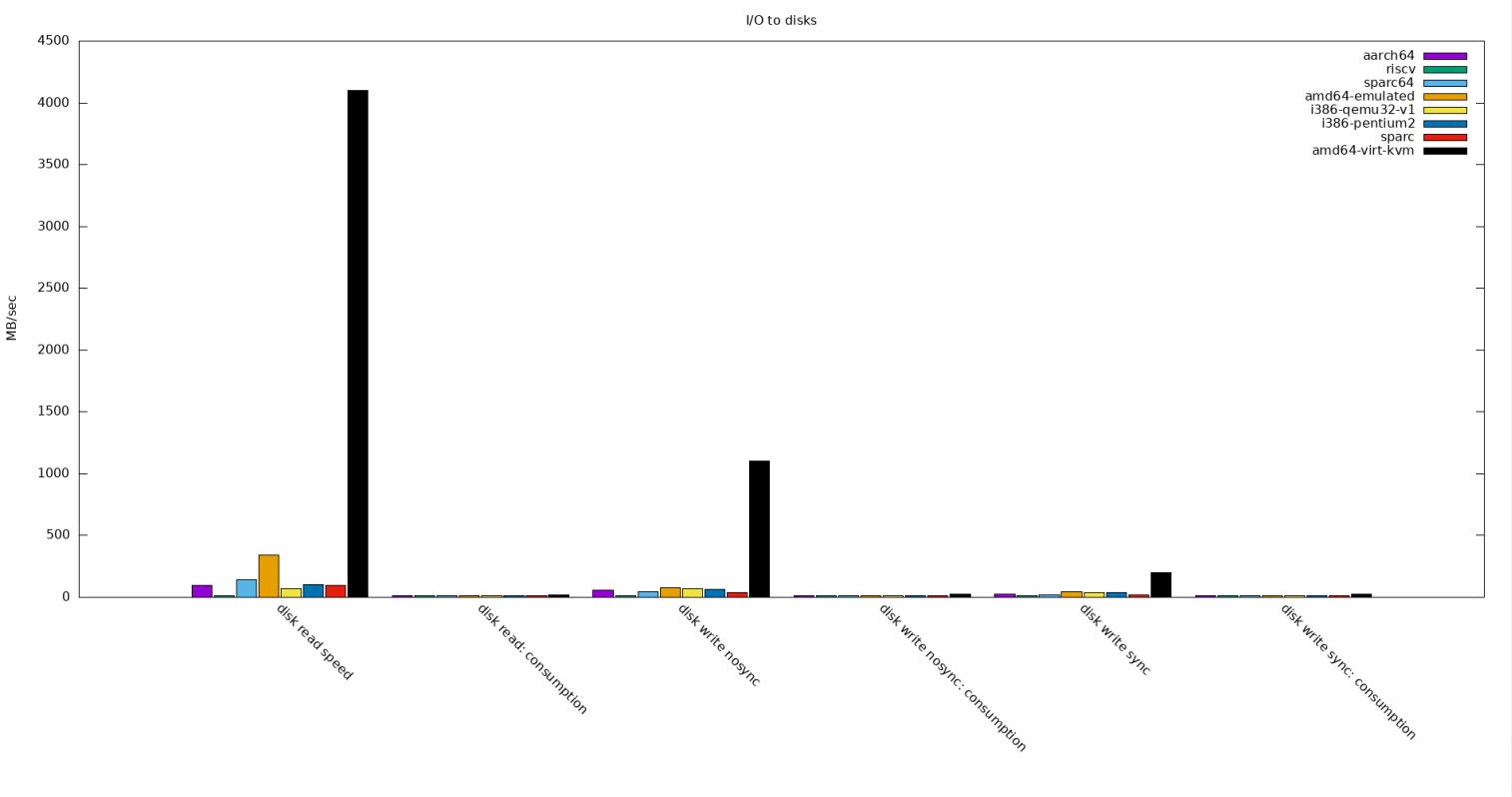
Network: all


Network throughput
MB/sec
30000
25000
20000
15000
10000
5000
0
aarch64
riscv
sparc64
amd64-emulated
i386-qemu32-v1
i386-pentium2
amdf41
amd64-virt-kvm
network throughput
network consumption

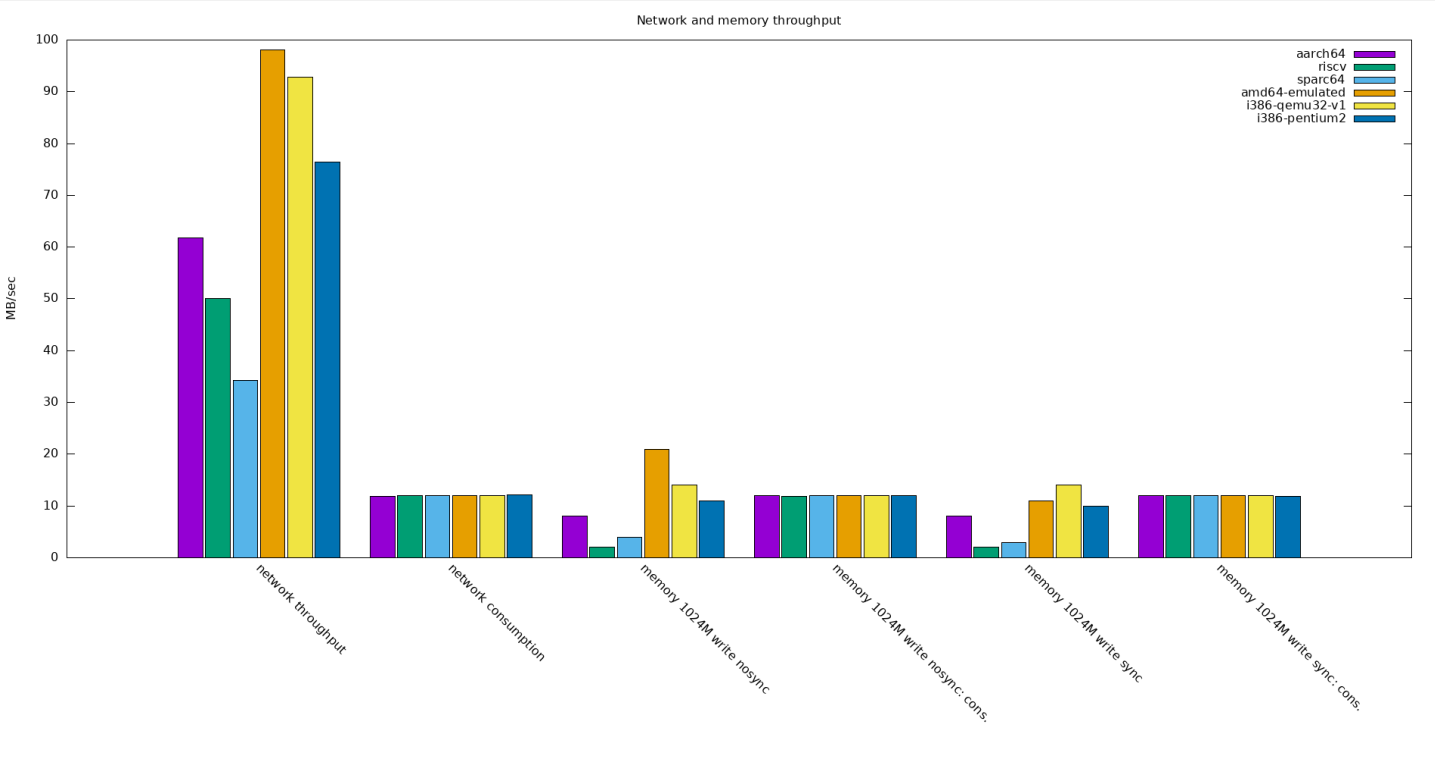
Disk I/O: emulated only



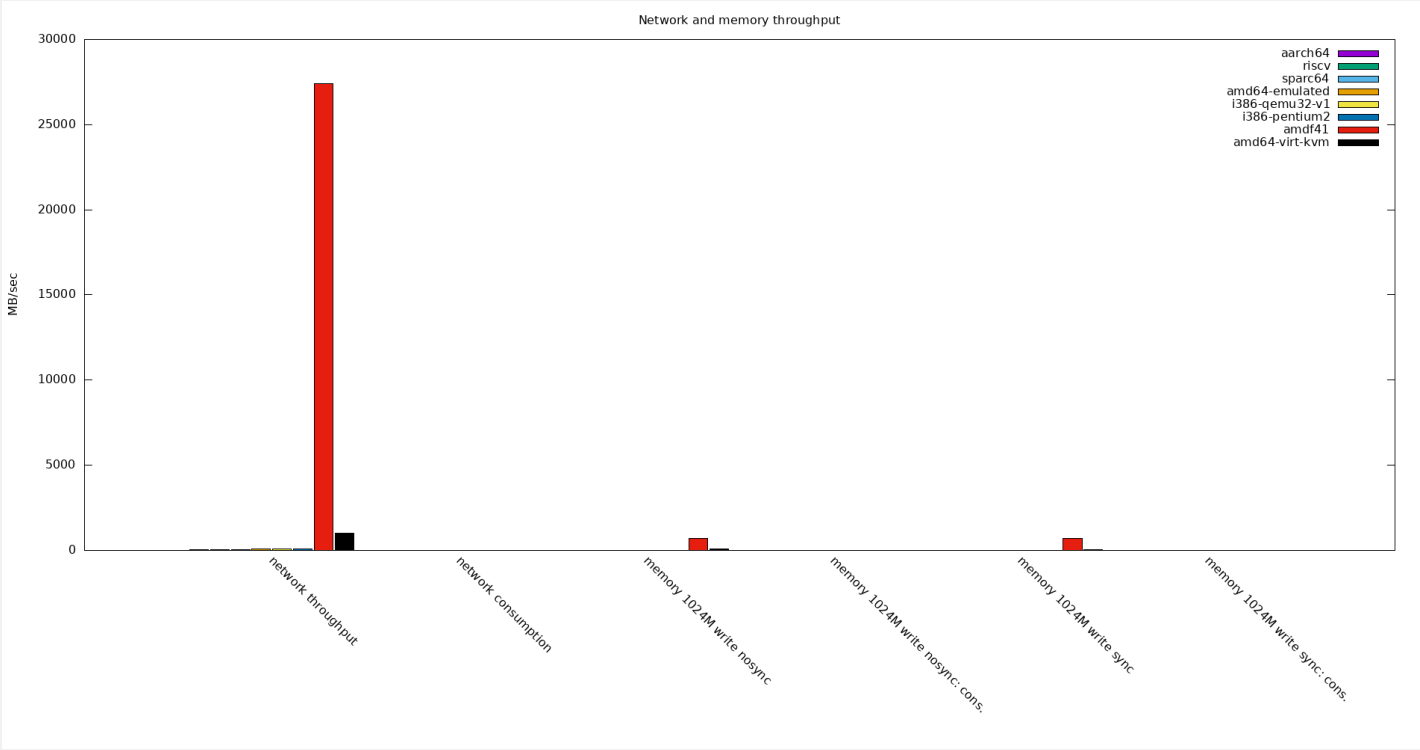
Disk I/O: all


I/O to disks
MB/sec
4500
4000
3500
3000
2500
2000
1500
1000
500
0
aarch64
riscv
sparc64
amd64-emulated
i386-qemu32-v1
i386-pentium2
sparc
amd64-virt-kvm
disk read speed
disk read: consumption
disk write nosync
disk write nosync: consumption
disk write sync
disk write sync: consumption

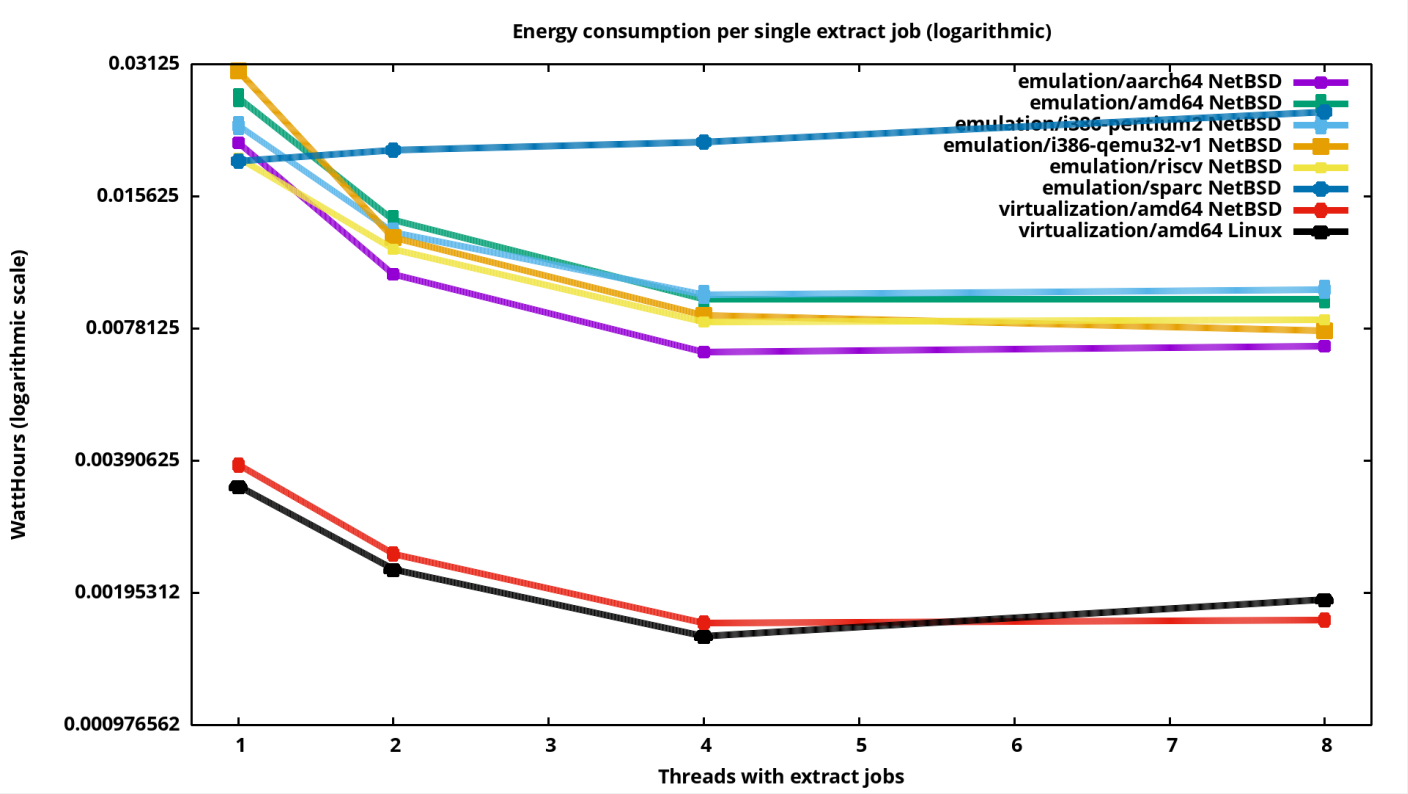
memory I/O: emulated only



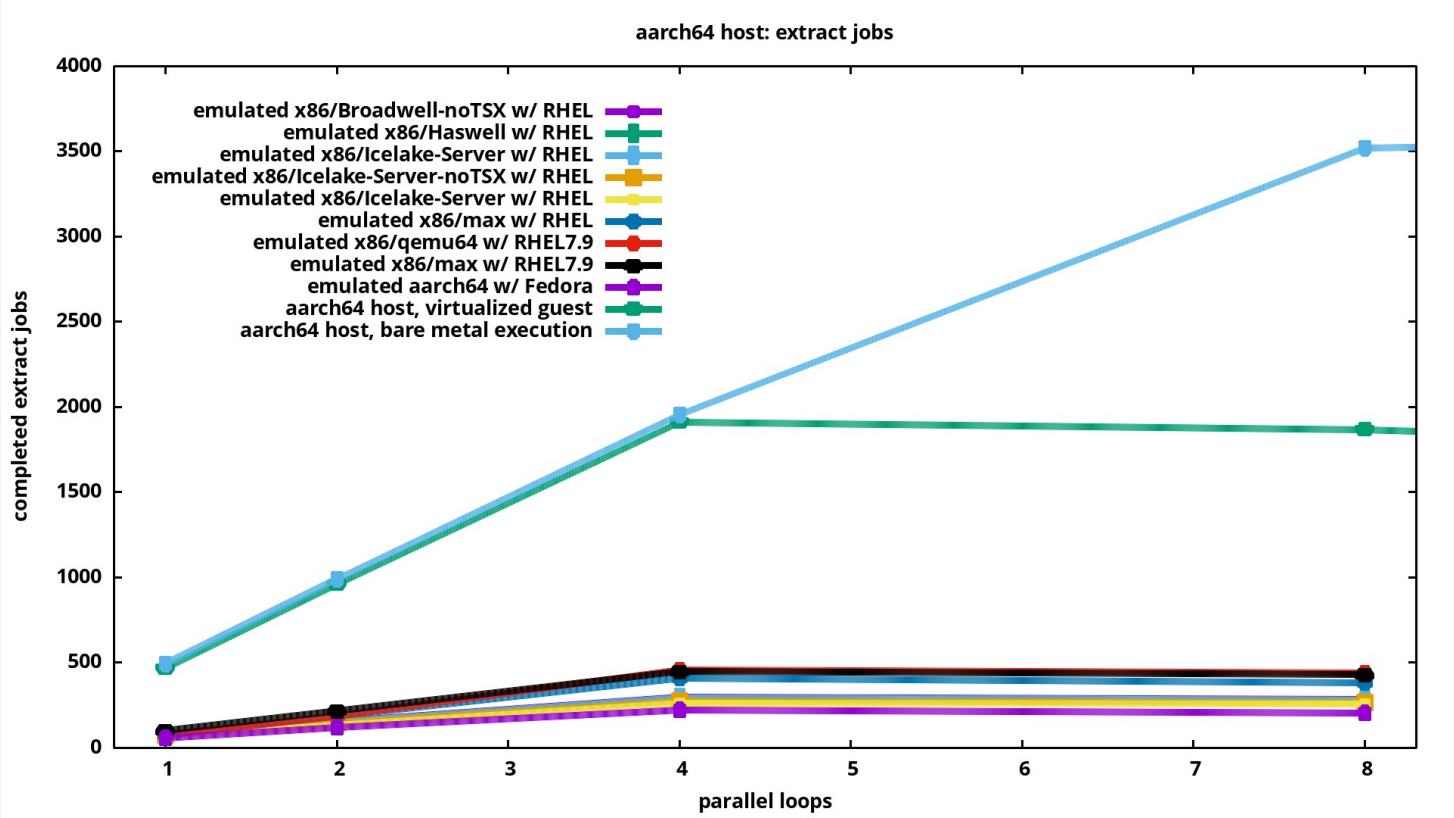
memory I/O: all



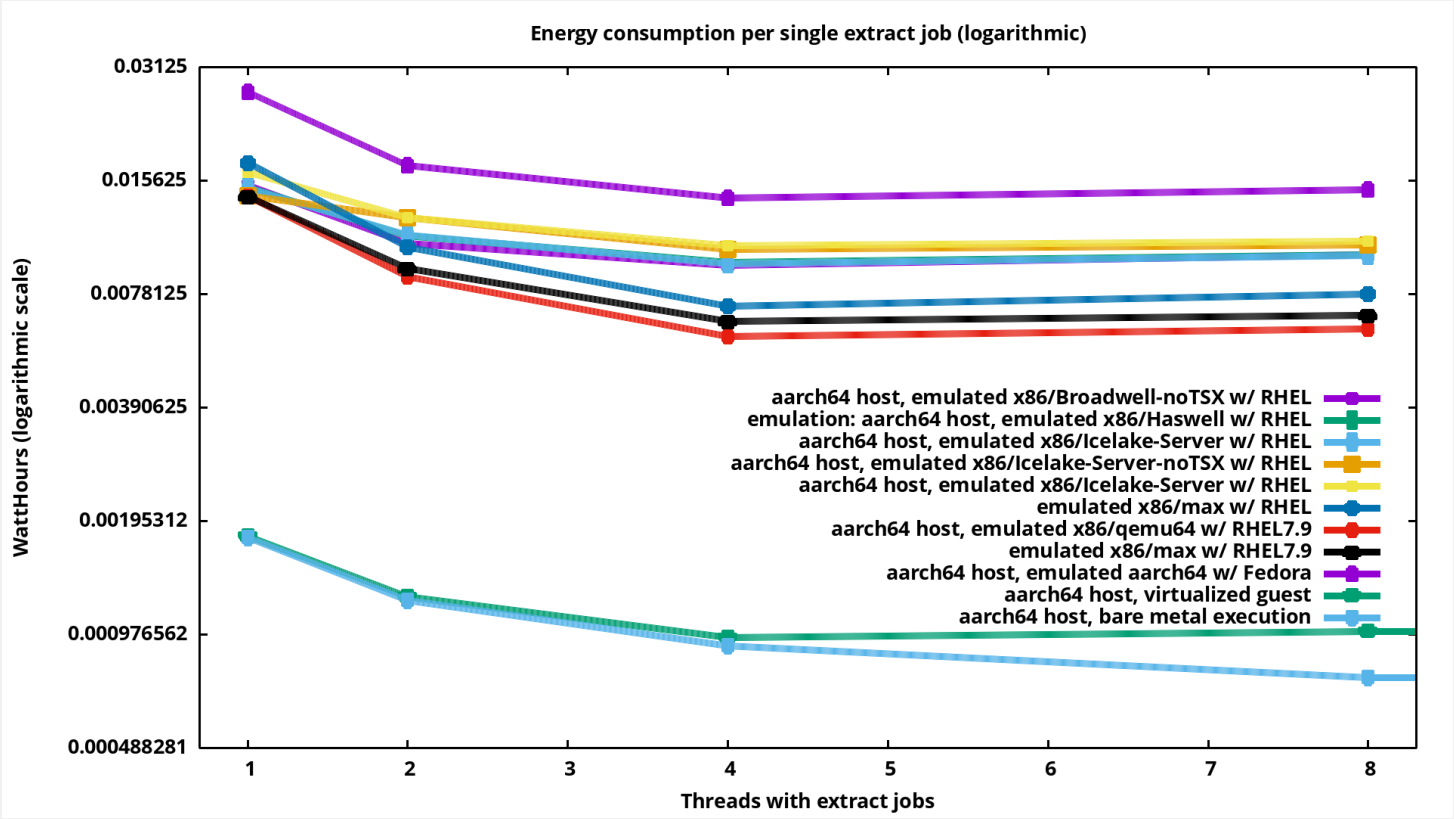
Energy efficiency: CPU workload



Comparing emulated chipsets



Bare/Virt/Emulated efficiency



Links

- This whole thing could have been an emai^Wblog article!
  - Indeed. See Link1 and Link2.
  - Read the second link (or ask me) to understand why bzcat on RHEL7@emulated-x86 is faster than on RHEL9@emulated-x86.
- Links to various investigations rooted in pmda-denki
- More on pmda-denki on the blog
- The code: https://github.com/christianhorn/smallhelpers

Thanks!

ありがとうございます

Danke!

Спасибо

Christian Horn
IRCnet: globalc@#NetBSD
https://fluxcoil.net
Mastodon: https://chaos.social/@globalc

Hallucinations & visions 1/2

- Worth implementing
  - AMD MSR metrics
  - TPMI metrics, Intel's successor to RAPL MSR
  - A pmda which takes overall system consumption and based on how many cycles a process consumed, compute how much energy it consumed. Won't be perfect, but works reasonably well, I have example code running.
- Stabilize code to assign overall consumption to processes:
  - "this KVM guest causes 80% of the cycles on CPUs, so it's consuming 80W (of the overall 100W system consumption)
  - If we inform the KVM guest how much it's consuming, we can attribute power consumption of whole data centers to processes!
  - pcp-htop as of PCP 6.1 can report anything in top-style, so it could provide a nice overview of the top-power-consuming processes

Hallucinations & visions 2/2

- Apple has apparently directly ~20 power sensors directly on the AppleSilicon hardware. As of 2023-09-11, these are not available, but will probably in the near future appear on Linux as sensor values. These will likely appear in pmda-lmsensors without further efforts required.
  Apple silicon power sensors
- Research how Android handles power management
- CPU flaws like Spectre/Meltdown, AMD Zenbleed, Intels Downfall get fixed in the Linux kernel and microcode. When running a workload on these, how much energy is the workload causing with and without the mitigation? pmda-denki can help to measure that.
- [github.com/xianek/Happy6502] is a great graphical representation. How about modeling a laptop or server (or whole data center), and with colours in the style of an infrared camera illustrate how much power the components consume?
- Together with details of "how the co2 imprint of the used power source is", one could express consumption in "co2". Could work via derived metrics. If the power is from coal or fuel, the consumed power could also be expressed in gramm-of-ignite or liter-of-fuel.