CRITICAL TECHWORKS (CTW)

QEMU and KVM

Assignements:

First steps

1st task

On the cover use a dive image. You go from Application level tweaks to kernel level tweaks
Check the initial email, and see if everything os beeing done or not

1. Install QEMU – DONE!
2. Mount a W10 x64 inside QEMU – DONE!
3. Don't forget to remove the iso file from the VM details (i);
4. Benchmark its performance;
5. When building binary, W10_x86-64_KVM.sh, try to understand what the commands do;
6. What sort of configurations QEMU have, and what do they do to the machine;

0.1 Field notes:

- To check if a machine can run KVM, and the output gives us the number of processors that can run KVM:
  egrep -c '(vmx—svm)' /proc/cpuinfo
  - 'cpuinfo' is a file that has the description of all flags and things that the processor can run.
    we can view its content by running the following command:
    cat /proc/cpuinfo/
  - '*vmx*' and '*svm*' are the flags that need to be present in the cpu to run hardware emulation and virualization, allowing us to install kvm.
- dont forget to adduser to kvm and libvirt (this one to allow creating vm) groups;
- bridge-utils command when installing allows us to create a bridge connection network, so this way we will be able to utilize for example a router to get an IP address and access it from there. Router (....1) -¿ host PC (....2) — virtual machine (....3);

- Host pc -≤ virtual machine -≤ host based networking ≥ - different from bridge networking =¿ NAT to have a local ip address it implies two networks on the host PC.
- Some Qemu commands to add to our qemu-system-x86_64:
  - *'–enable kvm'* – allows virtualization machines (kernel linux faster);
  - *'-m 4G or 4096'* – for ram;
  - *'-smp 4'* – for allocating numbre of cores (cpu threads);
  - *'-name name'* – to give a name to the virtual machine, 'name';
  - *'-boot d'* – boot system from iso (disk);
  - *'-cdrom path'* – to select iso path;
  - *'qemu-img'* – to create virtual storage; We should add -hda disk location in the qemu-system command
    - create - f qcow2 'debianbuster.qcow2' ¡-this is the name, 30G ¡- This is the size
    - qemu -image info debianbuster.qcow2
- to exit full screen ctl+alt+f

CREATE A NEW VM:
qemu-system-x86_64 -cpu max -enable-kvm -smp cores=1,threads=20 -
cdrom /home/franciscosantos/Desktop/QEMU/Iso_Images/Windows/Win10_21H2_E
-drive file=/home/franciscosantos/Desktop/QEMU/Virtual_Disks/disk2.qcow2,cache=
-m 4G -vga virtio -display gtk,gl=on

GITHUB – Talk about this and the script and ofofofofofof

1 Investigar estes topicos

Cross compiling Usar a gpu?

Open source, é 5 estrelas, o problema é que está sempre a mudar!

2 KVM

https://www.youtube.com/watch?v=G21c-sbggk4
3:44 to install the program that checks if the machine can run KVM.
Kernel based Machines, for hardware acceleration. It gives the opportunitie to run coomands that affect directly the hardware.

Resource over-committing – wich allows you to allocate more virtualizes CPU's and memory than available resources on the host. The VM's the only use what they need, allowing other VM'S to use unused resources.

To disable KVM you just remove *'-enable kvm'* and on the *'-cpu'* we remove the argument and insert a new one that is a different cpu that the host.

KVM uses qemu as his device simulator, any device operation is simulated by user space QEMU program. When you write to 0xB8000, the graphic display is operated which involves guest's doing a CPU 'vmexit' from guest mode and returning to KVM module, who in turn sends device simulation requests to user space QEMU backend.

In contrast, QEMU wo KVM does all the jobs in unified process except for usual system calls, there's fewer CPU context switches. Meanwhile, your benchmark code is a simple loop which only requires code block translation for just one time. That cost nothing, compared to vmexit and kernel-user communication of every iteration in KVM case.

KVM provides near native performance
KVM can over-commit (if we have 20gb kvm can alocatte more) is it good or bad?
It is not supported by every processors linux-kvm.org/page/Processor_support

What is an hypervisor? https://www.youtube.com/watch?v=6k6gtzseq2A&list=PI

2.1 kvm tunning

look into it

3 Qemu

QEMU stands for Quick Emulator, and it is a program that creates a virtual machine and runs the code that we feed into it. It has a built in **bootloader** – where it copys a file to virtual memory and jumps to the start information that it gets out of the header of it –

qemu-system-x86_64 - -help

qemu-img create -f qcow2 nome.qcow2 tamanho em G
Para criar o disco virtual

Reasons to use QEMU?
Performance, near native performance by executing guest code directly on the host CPU. Basicly run sthe guest code on the real CPU.
Where does this helps us?
KVM – helps the QEMU performance expecially n the x64 CPU. We can get even closer to near native performance.

Qemu is an application that tries to emulate a particular architecture.

the -cpu option in QEMU?

Here accelerator refers to KVM. You can use other accelerators by using the -accel option. The -cpu help flag gives you a pretty good explanation.

-cpu max means emulate a cpu that has all the features supported by KVM (limited by the set of features supported by your physical machines, of course)

-cpu host means emulate a cpu that is the same as your host cpu (limited by the set of features supported by the accelerator)

There isn't a whole lot of difference between the two, unless the

accelerator you are using has very limited supported feature set.

It is recommended to just stick with -cpu host, which is also the default. If you have used QEMU on Linux, you have probably enjoyed the performance boost brought by KVM: the same VM runs a lot faster when you launch QEMU with the -accel kvm (or -enable-kvm) option, thanks to hardware-assisted virtualization. On Windows, you can achieve a similar speed-up with -accel hax (or -enable-hax), after completing a one-time setup process.

qemu-system-x86_64 -cpu host -enable-kvm -smp 4 -cdrom /home/franciscosantos/Desktop/QEMU/Iso_Images/Windows /Win10_21H2_English_x64.iso -boot menu=on -drive file=/home/ franciscosantos/Desktop/QEMU/Virtual_imgs/Image.img -m 2G -vga virtio -display gtk,gl=on
O GTK dá, o dld não. Ver se vale a pena investigar. Isto enables opengl. It continuos emulate de display. Emulates VGA com isto

Para posteriores usos remover o -cdrom pois já nao é necessario.

qemu-system-x86 -cpu help
...
x86 host KVM processor with all supported host features
x86 max Enables all features supported by the accelerator in the current host
So it is better to use max?
QEMU also integrates a Linux specific user mode em- ulator. It is a subset of the machine emulator which runs Linux processes for one target CPU on another CPU. It is **mainly used to test the result of cross compilers or to test the CPU emulator without having to start a complete virtual machine**.

www.winehq.org
What is Wine?
Wine (originally an acronym for "Wine Is Not an Emulator") is a compatibility layer capable of running Windows applications on several

POSIX-compliant operating systems, such as Linux, macOS, & BSD. Instead of simulating internal Windows logic like a virtual machine or emulator, Wine translates Windows API calls into POSIX calls on-the-fly, eliminating the performance and memory penalties of other methods and allowing you to cleanly integrate Windows applications into your desktop.
QEMU has two operating modes: Full system emulation. In this mode, QEMU emulates a full system (for example a PC), including one or several processors and various peripherals. It can be used to launch different Operating Systems without rebooting the PC or to debug system code.
User mode emulation. In this mode, QEMU can launch processes compiled for one CPU on another CPU. It can be used to launch the Wine Windows API emulator (http://www.winehq.org) or to ease cross-compilation and cross-debugging.

An "irqchip" is KVM's name for what is more usually called an "interrupt controller". This is a piece of hardware which takes lots of interrupt signals (from devices like a USB controller, disk controller, PCI cards, serial port, etc) and presents them to the CPU in a way that lets the CPU control which interrupts are enabled, be notified when a new interrupt arrives, dismiss a handled interrupt, and so on.
An emulated system (VM) needs an emulated interrupt controller, in the same way that real hardware has a real hardware interrupt controller. In a KVM VM, it is possible to have this emulated device be in userspace (ie QEMU) like all the other emulated devices. But because the interrupt controller is so closely involved with the handling of simulated interrupts, having to go back and forth between the kernel and userspace frequently as the guest manipulates the interrupt controller is bad for performance. So KVM provides an emulation of an interrupt controller in the kernel (the "in-kernel irqchip") which QEMU can use instead of providing its own version in userspace. (On at least some architectures the in-kernel irqchip is also able to use hardware assists for virtualization of interrupt handling which the userspace version cannot, which further improves VM performance.)
The default QEMU setting is to use the in-kernel irqchip, and this gives the best performance. So you don't need to do anything with this command line option unless you know you have a specific reason why the in-kernel irqchip will not work for you.

Em vez de usar -smp 4 -> usar cores=..., threads=..., (691 vs 654)

in the qemu "-smp" argument we can pass the number of: cores, threads and sockets. Does any of you know how do they correlate? I know the concepts os each on in singular, but mixed i am not certain of its functionality. The cores and threads i think i understand, but the socket part i don't know.
i am asking to answer the question: **How can i fine tune this relation to improve performance, and if it can be done to achieve that purpose?**
https://blogs.vmware.com/customer-experience-and-success/2021/06/sockets-cpus-cores-and-threads-the-history-of-x86-processor-terms.html
10 cores and 20 threads. I still can only execute 10 OS instructions per cycle. But I can queue up 20 OS instructions.

Disk perspective https://blogs.igalia.com/berto/2015/12/17/improving-disk-io-performance-in-qemu-2-5-with-the-qcow2-l2-cache/
L1 and L2 tables
What level cache is, and why we use it
Explain how to use it. Use the excel doc.

QCOW2 ARGUMENTS tests and results:

Conclusion, from cinebench and geekbench outputs, so far on the 3 arguments for the .qcow2 format:

preallocation – behaves similar to normal. But my image didn't change, so this behavior was expected;

lazy_refcounts – the best on both so far. (it benefits from the cache=writethrough argument that is passed to be default on all tests)

cluster_size – not optimized for the disk size, it under-performs versus all. I will try to understand how to fine tune this, make new tests and see if the result changes.

Note: The cache is not persistent, you have to specify its size every time you open a new disk image.

[*] Updated 10 March 2021: since QEMU 3.1 the default L2 cache size has been increased from 1MB to 32MB.

QCOW3

Improving performance is the key motivation for a QCOW3 image format.

L2-cache-size vs Cluster-size

https://www.ibm.com/cloud/blog/how-to-tune-qemu-l2-cache-size-and-qcow2-cluster-size

So L2-cache-size is fine tuned in QEMU and Cluster-size is fine tuned in QCOW2 image file.
QCOW2 format stores 2 headers with levels of reference: L1 – that saves l2 references; L2 – that saves data cluster references.
L1 – can is stored alway sin the hypervisor cache memory because it is small in size.
L2 – can be pretty large, so fine tune of the hypervisor L2 cache sise if needed for improoved performance.

What is an hypervisor – https://www.vmware.com/topics/glossary/content/hyperv
Is QEMU an hypervisor? https://sumit-ghosh.com/articles/virtualization-hypervisors-explaining-qemu-kvm-libvirt/
Why do we need to fine tune the hypervisor?
To access a data block in the virtual disk, without any caching, the hypervisor needs to make three IO operations to the QCOW2 image file. This is quite expensive. To improve performance, the hypervisor caches L1 and L2 tables in memory, significantly reducing actual IOs. Since the

L1 table is small, it can be easily cached in memory all the time. The L2 tables can be pretty large. Therefore, the L2 cache size needs to be tuned properly to obtain the best performance.

qemu command accepts a simple -D switch which can create a log file. So for example including -D ./log.txt will create "log.txt" in your working directory.

You can access more logging/debugging options via QEMU Monitor (e.g. qemu -monitor stdio).

see cpu flags:
qemu-system-x86_64 -cpu help — grep pdpe
in this case was the one for huge pages

3.1 Arguments

https://www.qemu.org/docs/master/system/invocation.html?highlight=smp

-smp [[cpus=]n][,maxcpus=maxcpus][,sockets=sockets][,dies=dies][,clusters=cluster

Simulate a SMP system with 'n' CPUs initially present on the machine type board. On boards supporting CPU hotplug, the optional 'maxcpus' parameter can be set to enable further CPUs to be added at runtime. When both parameters are omitted, the maximum number of CPUs will be calculated from the provided topology members and the initial CPU count will match the maximum number. When only one of them is given then the omitted one will be set to its counterpart's value. Both parameters may be specified, but the maximum number of CPUs must be equal to or greater than the initial CPU count. Product of the CPU topology hierarchy must be equal to the maximum number of CPUs. Both parameters are subject to an upper limit that is determined by the specific machine type chosen.

To control reporting of CPU topology information, values of the topology parameters can be specified. Machines may only support a subset of the parameters and different machines may have different subsets supported which vary depending on capacity of the corresponding CPU targets. So for a particular machine type board, an expected topology hierarchy can be defined through the supported sub-option. Unsupported parameters can also be provided in addition to the sub-option, but their values must be set as 1 in the purpose of correct parsing.

Either the initial CPU count, or at least one of the topology parameters must be specified. The specified parameters must be greater than zero, explicit configuration like "cpus=0" is not allowed. Values for any omitted parameters will be computed from those which are given.

For example, the following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 cores per socket, 2 threads per core) for a machine that only supports sockets/cores/threads. Some members of the option can be omitted but their values will be automatically computed:

-smp 8,sockets=2,cores=2,threads=2,maxcpus=8

The following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 dies per socket, 2 cores per die, 2 threads per core) for PC machines which support sockets/dies/cores/threads. Some members of the option can be omitted but their values will be automatically computed:

-smp 16,sockets=2,dies=2,cores=2,threads=2,maxcpus=16

The following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 clusters per socket, 2 cores per cluster, 2 threads per core) for ARM virt machines which support sockets/clusters /cores/threads. Some members of the option can be omitted but their values will be automatically computed:

-smp 16,sockets=2,clusters=2,cores=2,threads=2,maxcpus=16

Historically preference was given to the coarsest topology parameters when computing missing values (ie sockets preferred over cores, which were preferred over threads), however, this behaviour is considered liable to change. Prior to 6.2 the preference was sockets over cores over threads. Since 6.2 the preference is cores over sockets over threads.

For example, the following option defines a machine board with 2 sockets of 1 core before 6.2 and 1 socket of 2 cores after 6.2:

-smp 2

-numa node[,mem=size][,cpus=firstcpu[-lastcpu]][,nodeid=node][,initiator=initiator
-numa node[,memdev=id][,cpus=firstcpu[-lastcpu]][,nodeid=node][,initiator=initiator]
-numa dist,src=source,dst=destination,val=distance
-numa cpu,node-id=node[,socket-id=x][,core-id=y][,thread-id=z]
-numa hmat-lb,initiator=node,target=node,hierarchy=hierarchy,data-type=tpye[,latency=lat][,bandwidth=bw]
-numa hmat-cache,node-id=node,size=size,level=level[,associativity=str][,policy=str][

Define a NUMA node and assign RAM and VCPUs to it. Set the NUMA distance from a source node to a destination node. Set the ACPI Heterogeneous Memory Attributes for the given nodes.

Legacy VCPU assignment uses 'cpus' option where firstcpu and lastcpu are CPU indexes. Each 'cpus' option represent a contiguous range of CPU indexes (or a single VCPU if lastcpu is omitted). A non-contiguous set of VCPUs can be represented by providing multiple 'cpus' options. If 'cpus' is omitted on all nodes, VCPUs are automatically split between them.

For example, the following option assigns VCPUs 0, 1, 2 and 5 to a NUMA node:

-numa node,cpus=0-2,cpus=5

'cpu' option is a new alternative to 'cpus' option which uses 'socket-id—core-id—thread-id' properties to assign CPU objects to a node using topology layout properties of CPU. The set of properties is machine specific, and depends on used machine type/'smp' options. It could be queried with 'hotpluggable-cpus' monitor command. 'node-id' property specifies node to which CPU object will be assigned, it's required for node to be declared with 'node' option before it's used with 'cpu' option.

For example:

-M pc
-smp 1,sockets=2,maxcpus=2
-numa node,nodeid=0 -numa node,nodeid=1
-numa cpu,node-id=0,socket-id=0 -numa cpu,node-id=1,socket-id=1

Legacy 'mem' assigns a given RAM amount to a node (not supported for 5.1 and newer machine types). 'memdev' assigns RAM from a given memory backend device to a node. If 'mem' and 'memdev' are omitted in all nodes, RAM is split equally between them.

'mem' and 'memdev' are mutually exclusive. Furthermore, if one node uses 'memdev', all of them have to use it.

'initiator' is an additional option that points to an initiator NUMA node that has best performance (the lowest latency or largest bandwidth) to this NUMA node. Note that this option can be set only when the machine property 'hmat' is set to 'on'.

Following example creates a machine with 2 NUMA nodes, node 0 has CPU. node 1 has only memory, and its initiator is node 0. Note that because node 0 has CPU, by default the initiator of node 0 is itself and must be itself.

-machine hmat=on
-m 2G,slots=2,maxmem=4G
-object memory-backend-ram,size=1G,id=m0
-object memory-backend-ram,size=1G,id=m1
-numa node,nodeid=0,memdev=m0 -numa node,nodeid=1,memdev=m1,initiator=0
-smp 2,sockets=2,maxcpus=2
-numa cpu,node-id=0,socket-id=0
-numa cpu,node-id=0,socket-id=1

source and destination are NUMA node IDs. distance is the NUMA distance from source to destination. The distance from a node to itself is always 10. If any pair of nodes is given a distance, then all pairs must be given distances. Although, when distances are only given in one direction for each pair of nodes, then the distances in the opposite directions are assumed to be the same. If, however, an asymmetrical pair of distances is given for even one node pair, then all node pairs must be provided distance values for both directions, even when they are symmetrical. When a node is unreachable from another node, set the pair's distance to 255.

Note that the -numa option doesn't allocate any of the specified resources, it just assigns existing resources to NUMA nodes. This means that one still has to use the -m, -smp options to allocate RAM and VCPUs respectively.

Use 'hmat-lb' to set System Locality Latency and Bandwidth Information between initiator and target NUMA nodes in ACPI Heterogeneous Attribute Memory Table (HMAT). Initiator NUMA node can create memory requests, usually it has one or more processors. Target NUMA node contains addressable memory.

In 'hmat-lb' option, node are NUMA node IDs. hierarchy is the memory hierarchy of the target NUMA node: if hierarchy is 'memory', the structure represents the memory performance; if hierarchy is

'first-level—second-level—third-level', this structure represents aggregated performance of memory side caches for each domain. type of 'data-type' is type of data represented by this structure instance: if 'hierarchy' is 'memory', 'data-type' is 'access—read—write' latency or 'access—read—write' bandwidth of the target memory; if 'hierarchy' is 'first-level—second-level—third-level', 'data-type' is 'access—read—write' hit latency or 'access—read—write' hit bandwidth of the target memory side cache.

lat is latency value in nanoseconds. bw is bandwidth value, the possible value and units are NUM[M—G—T], mean that the bandwidth value are NUM byte per second (or MB/s, GB/s or TB/s depending on used suffix). Note that if latency or bandwidth value is 0, means the corresponding latency or bandwidth information is not provided.

In 'hmat-cache' option, node-id is the NUMA-id of the memory belongs. size is the size of memory side cache in bytes. level is the cache level described in this structure, note that the cache level 0 should not be used with 'hmat-cache' option. associativity is the cache associativity, the possible value is 'none/direct(direct-mapped)/complex(complex cache indexing)'. policy is the write policy. line is the cache Line size in bytes.

For example, the following options describe 2 NUMA nodes. Node 0 has 2 cpus and a ram, node 1 has only a ram. The processors in node 0 access memory in node 0 with access-latency 5 nanoseconds, access-bandwidth is 200 MB/s; The processors in NUMA node 0 access memory in NUMA node 1 with access-latency 10 nanoseconds, access-bandwidth is 100 MB/s. And for memory side cache information, NUMA node 0 and 1 both have 1 level memory cache, size is 10KB, policy is write-back, the cache Line size is 8 bytes:

```
  -machine hmat=on
-m 2G
-object memory-backend-ram,size=1G,id=m0
-object memory-backend-ram,size=1G,id=m1
-smp 2,sockets=2,maxcpus=2
```

```
-numa node,nodeid=0,memdev=m0
-numa node,nodeid=1,memdev=m1,initiator=0
-numa cpu,node-id=0,socket-id=0
-numa cpu,node-id=0,socket-id=1
-numa hmat-lb,initiator=0,target=0,hierarchy=memory,data-type=access-latency,latency=5
-numa hmat-lb,initiator=0,target=0,hierarchy=memory,data-type=access-bandwidth,bandwidth=200M
-numa hmat-lb,initiator=0,target=1,hierarchy=memory,data-type=access-latency,latency=10
-numa hmat-lb,initiator=0,target=1,hierarchy=memory,data-type=access-bandwidth,bandwidth=100M
-numa hmat-cache,node-id=0,size=10K,level=1,associativity=direct,policy=write-back,line=8
-numa hmat-cache,node-id=1,size=10K,level=1,associativity=direct,policy=write-back,line=8
```

4 Virtualization VS Emulation

Insert here stuff

5 Fine tunning

See linux kernel version: uname -r
Desativar cortana e essas merdas

QCOW file formats can have: (pagina 78 do qemu.doc)
cluster_size
Changes the qcow2 cluster size (must be between 512 and 2M).
Smaller cluster sizes can improve the image file size whereas larger
cluster sizes generally provide better performance.

A virtual disk, much like how operating systems treat physical disks, are split up into clusters; each cluster being a predefined size and holding a single unit of data. A cluster is the smallest amount of data that can be read or written to in a single operation. There is then an index lookup that's often kept in memory that knows what information is stored in each cluster and where that cluster is located.

preallocation
Preallocation mode (allowed values: off, metadata, falloc, full).
An image with preallocated metadata is initially larger but can improve performance when the image needs to grow. falloc and full preallocations are like the same options of raw format, but sets up metadata also.

lazy_refcounts
If this option is set to on, reference count updates are postponed with the goal of avoiding metadata I/O and improving perfor- mance. This is particularly interesting with cache=writethrough which doesn't batch metadata updates. The tradeoff is that after a host crash, the reference count tables must be rebuilt, i.e. on the next open an (automatic) qemu-img check -r all is required, which may take some time.

A socket is the physical socket where the physical CPU capsules are placed. A normal PC only have one socket.
Cores are the number of CPU-cores per CPU capsule. A modern standard

CPU for a standard PC usually have two or four cores.
And some CPUs can run more than one parallel thread per CPU-core. Intel (the most common CPU manufacturer for standard PCs) have either one or two threads per core depending on CPU model.
If you multiply the number of socket, cores and threads, i.e. 2*6*2, then you get the number of "CPUs": 24. These aren't real CPUs, but the number of possible parallel threads of execution your system can do.
Just the fact that you have 6 cores is a sign you have a high-end workstation or server computer. The fact that you have two sockets makes it a very high-end computer. Usually not even high-end workstations have that these days, only servers.

CACHE https://documentation.suse.com/sles/11-SP4/html/SLES-all/cha-qemu-cachemodes.html

By default, the cache=writeback mode is used. It will report data writes as completed as soon as the data is present in the host page cache. This is safe as long as your guest OS makes sure to correctly flush disk caches where needed. If your guest OS does not handle volatile disk write caches correctly and your host crashes or loses power, then the guest may experience data corruption. For such guests, you should consider using cache=writethrough. This means that the host page cache will be used to read and write data, but write notifi- cation will be sent to the guest only after QEMU has made sure to flush each write to the disk. Be aware that this has a major impact on performance. The host page cache can be avoided entirely with cache=none. This will at- tempt to do disk IO directly to the guest's memory. QEMU may still perform an internal copy of the data. Note that this is considered a writeback mode and the guest OS must handle the disk write cache correctly in order to avoid data corruption on host crashes. citado do qemu-doc.pdf

cpu isolation block app to access this cpu

savevm

savevm name

Save the virtual machine as the tag 'name'. Not all filesystems support this. raw does not, but qcow2 does.
loadvm

loadvm name

Load the virtual machine tagged 'name'. This can also be done on the command line: -loadvm name

With the info snapshots command, you can request a list of available machines.

Example: Create of current state of VM with a name ubuntu_fresh

qemu-img snapshot -c ubuntu_fresh ubuntu.qcow2

16.1 Accessing Monitor Console

To access the monitor console from QEMU, press Ctrl–Alt–2. To return back to QEMU from the monitor console, press Ctrl–Alt–1.
Test scenario of 4 options to improove qcow disk image performance
Install commands
franciscosantos@T14Lenovo: /Desktop/QEMU/Virtual_Disks qemu-img create -f qcow2 -o cluster_size=2M 'cluster_size_2M.qcow2' 40G
Formatting 'cluster_size_2M.qcow2', fmt=qcow2 size=42949672960 cluster_size=2097152 lazy_refcounts=off refcount_bits=16
franciscosantos@T14Lenovo: /Desktop/QEMU/Virtual_Disks ls cluster_size_2M.qcow2 disk2.qcow2 snapshot.qcow2 franciscosantos@T14Lenovo: /Desktop/QEMU/Virtual_Disks qemu-img create -f qcow2 -o preallocation=full 'preallocation_full.qcow2' 40G
Formatting 'preallocation_full.qcow2', fmt=qcow2 size=42949672960 cluster_size=65536 preallocation=full lazy_refcounts=off refcount_bits=16

franciscosantos@T14Lenovo: /Desktop/QEMU/Virtual_Disks qemu-img create -f qcow2 -o lazy_refcounts=on 'lazy_refcounts_on.qcow2' 40G
Formatting 'lazy_refcounts_on.qcow2', fmt=qcow2 size=42949672960 cluster_size=65536 lazy_refcounts=on refcount_bits=16

All arguments direct to a corrupt image ,Don't USE!
franciscosantos@T14Lenovo: /Desktop/QEMU/Virtual_Disks qemu-img create -f qcow2 -o cluster_size=2M,preallocation=full,lazy_refcounts=on '4_options_enabled.qcow2' 40G
Formatting '4_options_enabled.qcow2', fmt=qcow2 size=42949672960 cluster_size=2097152 preallocation=full lazy_refcounts=on refcount_bits=16

cluster size define-se na imagem
l2 cache size define-se no comando.
tb se pode definir outro parametro ao mesmo tempo.

...The table below shows the effectiveness of using the default L2 cache size. For example, if the cluster size is set to 2MiB, each L2 table of 2MiB can contain 262,144 references of data ... – Cada tabela L2 de 2Mb referencia 262K de data clusters de 2Mb cada um.
There are several possible default configurations for the L2 cache and all of them have drawbacks:

a) Have a small(ish) size, like now (1 MB per disk image).
+ Pros: low memory footprint, good enough for many common scenarios.
+ Cons: bad (or very bad) performance with images larger than 8GB and lots of I/O.

b) Have the maximum possible cache size (1MB per 8GB of disk image if using the default cluster size).
+ Pros: good performance in all cases, no need to worry about it.
+ Cons: it can be very wasteful of RAM. A 1TB disk image takes 128MB of RAM for the L2 cache alone. If there are more disk drives (or backing images) the problem gets worse. In most cases you're not going to peform

random I/O on the whole disk, so you don't need such a big cache.
c) Have a large cache size (like in b), and remove unused entries periodically (cache-clean-interval setting).
+ Pros: it provides the best of both worlds, you'll get good performance and the unused memory will be returned to the system.
+ Cons: you can still have peaks of RAM usage. We'd still need to decide what's the best length for the cache cleaning interval. The memory footprint of the VM becomes more volatile and difficult to control.
https://bugzilla.redhat.com/show_bug.cgi?id=1377735

For each 8gb of disk space we need 1Mb of l2-cache-size ram. So if we use default cluster size (64kB) we need to L2 tables

If we use CS of 128GB and 1Mb l2 cahce we can alocate 16GB of disk size.

Subcluster https://blogs.igalia.com/berto/2020/12/03/subcluster-allocation-for-qcow2-images/
Muito interessante isto
It needs qemu 5.2 or higher

previews without cluster size: 1sec ~ 25secs
after cluster size: 1sec ~

Process and interrupt priorities matter a lot when alocatting resources.

5.1 BEST SO FAR

Image creation with cluster size and lazy_refcounts on

img launch, l2 cache fine tuned, cache writethrough

5.2 BASH SCRIPS

Está feito um para criar as VM's.

echo $(nproc) – this says run echo command and turn nproc into an argument

5.3 BCCBPF Tools

They are used for high speed custom packet filtering

https://opensource.com/article/17/11/bccbpf-performance
https://github.com/iovisor/bcc

A tracer is down to 200nanoseconds per trace. so it is very fast

execsnoop

One of the benefits that Ftrace brings to Linux is the ability to see what is happening inside the kernel
The ftrace has a front end that is trace-cmd (better for human reading)
sudo apt-get install bpfcc-tools linux-headers-$(uname -r) to install
sudo apt-get install bcc

PAra dar run as tools:
sudo nome da tool-bpfcc

verificar o nome da tool na pasta usr sbin bpfcc.

fix python dependencies

echo "deb https://repo.iovisor.org/apt/bionic bionic main" — sudo tee /etc/apt/sources.list.d/iovisor.list

sudo apt-get install -y libbpfcc

sudo apt-get install bpfcc-tools linux-headers-$(uname -r)

cannot import name 'MutableMapping' from 'collections' (/us-

r/lib/python3.10/collections __init__.py) -¿ go to the script and write import MutableMapping and save

Install it from store is easier for now.

instalar ferramentas de analise como mpstat:
sudo apt install sysstat

https://www.youtube.com/watch?v=zUYCXBlUcYI
19:57 talks about how insecure this is.
there is eBPF and BPF.

HTOP as well, and it can be personalized.

5.4 Tracing

compilando um ficheiro c. Ver o output de objdump -d hello
it creates an elf – executable and linkeble format file – it is written in assembly . It is like an archive or rar to store executable data

Kernel is the system provider for appications, it play this via system calls.

we can see system calls with strace ./hello a lot of info.
Strace we can see the communication between the application and the kernel

ftrace, built and mainteined by steven rostedt, is best. The oficial linux tracing
How? first see if sudo mount -t tracefs nodev /sys/kernel/tracing
go to that directory cd /sys/....
ls and see everything that we can do with ftrace
it need to be su sudo su
cd /sys/kernel/tracing

ls

It is very heavy, and it works like by dinamicly modify the code that calls a tracer.
Patchin gruning kernel uses this.

To enable tracer:
echo function > current_tracer
cat trace to see

to disable tracer:
echo nop > current_tracer

NOTE: you have to always be root to do anything usefull.
we can use trace-cmd, it is easier to run (less instructions) and what it does is created by the same guy, it is a command line utility. this mount from us withou we have to

Instalar o make e sudo apt-get -y install asciidoc
then go sudo su
to start:
trace-cmd start -p function
trace-cmd show
to stop:
trace-cmd start -p nop

trace-cmd record -e syscalls -F ./hello (guarda isto num ficheiro)
trace-cmd report para ver o report
trace-cmd report — less para ver pagina a pagina

Example: AS we know trace-cmd is the front end for ftrace, it needs to be root to output any result. Don't forget!
trace-cmd -e syscalls -F ./hello
→ -e syscalls – we will record all system calls;

→ -F – we will follow the program;
→ ./hello – the program that is going to be followed and recorded;
So we can sum that trace-cmd records everthing that happens inside the kernel, regarding, in this case, system calls for ./hello.

More complex analysis:

trace-cmd record -p function_graph –max-graph-depth 1 -e syscalls -F ./hello
trace-cmd report -IS does not record HArd interrupts and Soft interrupts (see this)

What does the it mean?
see 24:34 https://www.youtube.com/watch?v=68osT1soAPM
It shows all functions that are called, and shows them on a "C" format. It show the enter and the exit of a function. Max depth of 1 only traces the first function that is going into the kernel and back, nothing else. It shows pages faults, a thing that is not shown in strace. (do page fault) – whe we execute a function it ddoes not load the elf file to memory. It just represents it in memory. 33:50 https://www.youtube.com/watch?v=JRyrhsx-L5Y
It only loads on demand, it is more efficient.
This means that it tryed to read memory that wasnt there.
we can do max depth 2 and this shows not only the first function but the functions that function calls.
we can look at a specific function by doing this:
trace-cmd record -p function_graph -g nome da funcao que queremos ver -F ./hello
to see the report, trace-cmd report
the -g means graph this function only. we can ignore functions inside our serch by doing this:
trace-cmd record -p function_graph -g nome da funcao que queremos ver -n nome da funcao a ignorar -F ./hello
trace-cmd report

we can -o interrupts (confirm what it does better):
trace-cmd record -p function_graph -g nome da funcao que queremos ver

-n nome da funcao a ignorar -o funcao interrupt \-F ./hello
trace-cmd report

VFS means virtual file system, wich handles all filesystem, all files pass in this layer.
ttwu means try to wake up

https://www.youtube.com/watch?v=68osT1soAPM
49' **Digg deeper:**
\-F it means only to trace this program (./hello) and nothing else. we can remove it to trace all.
we can trace just certain events with -e

Note that -f it stands for filter!
Note that -R stands for trigger!

man trace-cmd is a bless!

trace-cmd record -e sched_wakeup -e sched_switch -e sys_enter_write ./hello
with this we can really see the shcedular switching, it is not at the same time, but it is so fast that it seams.

but this e too much information! How can we visualize it better?
With KernelShark, it can read trace.dat files (this file is produced by trace-cmd record)
it gives visualization of what is happening
much easier to parse large amounts of data
sudo apt-get install -y kernelshark
open the app and load the .dat file created from the trace-cmd
It shows the thing that really worked up.

Now with visualization we can record all and see it.
trace-cmd record -e all ./hello
trace-cmd report

we can scroll wheel to zoom in
we can search based on columns!
we can rigth click on the entry to see more advanced stuff and filters.
we can drag an area on the plot to zoom.
Note: The reports are rewritten so be carefull if you want to save any.

Search for a systemcall process id:
trace-cmd record -e syscalls -P 1779 ./hello
(enable all systemcalls and see what that process is doing)
trace-cmd report

we can then run the:
trace-cmd record -p function_graph -g function ./hello
trace-cmd report -F '.:common_pid == 1779' (fazer so o report deste pid)

All timestamps are in seconds, with a resolution of 1/10 of a microsecond.
The data on the plots are either CPU specific or task specific.
If they are CPU specific, then the data holds the timeline of events that happened on a given CPU (which CPU is shown in the plot title area).
If the plot is task specific, then the timeline of events are for the given task regardless of what CPU it was on at the time. The task name is also shown in the plot title area.
The CPU plots change colors as different tasks run on the CPU, and the task plots change color depending on what CPU the task is running on. This makes it easy to see how much a task bounces around the CPUs. Zooming in on a task plot also shows some more characteristics of the task.
The hollow green bar that is shown in front of some events in the task plot represents when the task was woken up from a sleeping state to when it actually ran. The hollow red bar between some events shows that the task was preempted by another task even though that task was still runnable.

The columns of the list are:

- the position of the event in the full data set.
CPU - the CPU that the event occurred on.
Time Stamp - The timestamp of the event. This is in seconds with 1/10th microsecond resolution.
Task - The name of the process that was running when the event occurred. PID - The process ID of the task that was running when the event occurred. Latency - The latency is broken into 4 fields:
Interrupts disabled - 'd' if interrupts are disabled, otherwise '.' Need reschedule - 'N' if the kernel was notified that a schedule is needed, otherwise '.'
In IRQ - 'h' if in a hard IRQ (hardware triggered), 's' if in a soft IRQ (context where the kernel initiated a the IRQ handler) or if soft IRQs are disabled, 'H' if in a hard IRQ and soft IRQs are disabled or the hard IRQ triggered while processing a soft IRQ, otherwise '.'
Preemption counter - The index of the preemption counter. If it is other than zero, then the kernel will not preempt the running tasks, even if a schedule has been requested by some event. If the counter is zero, then '.' is shown.

Preemptive multitasking - Running several processes/threads on a single processor, creating the illusion that they run concurrently when actually each is allocated small multiplexed time slices to run in. A process is "preempted" when it is scheduled out of execution and waits for the next time slice to run in.

A preemptive kernel is one that can be interrupted in the middle of executing code - for instance in response for a system call - to do other things and run other threads, possibly those that are not in the kernel.

The main advantage of a preemptive kernel is that sys-calls do not block the entire system. If a sys-call takes a long time to finish then it doesn't mean the kernel can't do anything else in this time. The main disadvantage is that this introduces more complexity to the kernel code, having to handle more end-cases, perform more fine grained locking or use lock-less structures and algorithms.

Note: These may be different depending on the kernel the trace.dat file came from.
Event - The name of the event.
Info - The data output of a particular event.

The list search can find an event based on the contents in a row. Select a column, a match criteria and the content to match to find the next row that matches the search. The match criterion is one of the following:

contains - the row cell of the selected column contains the match data. This works with numbers as well.
full match - the row cell of the selected column matches exactly the match data.
does not have - the row cell of the selected column does not contain the match data.

NOTE: we can Export Filter: save the filter to a file, that can be imported at a later time
in filters in the "show..." we can just show what we want.

Filtering on events may not be enough. The ability to filter on the content of individual events may be needed. In order to accomplish this, the advanced event filtering is used. Selecting Advance Filtering from the Filter menu will pop up the advanced event filtering dialog.

see 49 minuts. very hard.

To disable all tracing, which will ensure that no overhead is left from using the function tracers or events, the reset command can be used. It will disable all of Ftrace and bring the system back to full performance.

trace-cmd reset

https://lwn.net/Articles/410200/

You can do extensive filtering on events and what CPUs you want to focus on:
trace-cmd report –cpu 0 -F 'sched_wakeup: success == 1'
version = 6
cpus=2
..
The –cpu 0 limits the output to only show the events that occurred on CPU 0. The -F option limits the output further to only show sched_wakeup events that have its success field equal to 1. For more information about the filtering, consult the trace-cmd-report(1) man page.

To track a specific cpu:
trace-cmd report –cpu 0

5.4.1 Tracing QEMU:

It is good to formulate questions xD
Enable tracing while a specific command is being executed:
trace-cmd record -p function ls -¿¿ Test this with qemu
we need to start the trace.cmd, launch this (see upper line) cmd and then run the ls. Lastly run the report, and stop the tracing by either set nop or reset.
https://lwn.net/Articles/425583/

1. as sudo su type: trace-cmd start -p function (no need with record)!
2. trace-cmd stat (to see if it is enabled)
3. trace-cmd record -p function ls (run it)
4. In other terminal window run your script
5. in the initial terminal generate the .dat file with: trace-cmd report
6. stop tracing by trace-cmd stop or reset.

NOTE: DOnt forget that the report saves in the current directory!

Don't forget to go to the filter tab to filter the information on the list, and go to plot tab to show more info in the plot area.

to trace just a function we use: trace-cmd record -p function_graph -g 'switch_fpu_return' ls

To search for a function inside the kernel that has a specifi word, we will use write as the word, on it we can run:
trace-cmd list -f write
We can then filter more by grepping it to syscals and see which syscall functions have write on it:
trace-cmd list -f write — grep syscals

We can combine that all in the record command like this:
trace-cmd record -p function_graph -g '*sys*write' -F ./hello
The -g stans for graph this function.
We can now trace any sys write functions that are called when running the hello program.
We can the create a trace-cmd report — less and see what the write syscall does and follow it through.
we can then open up the souce code related to any function and learn how a OS really works. That is the beauty of open source.

Something so simple as Hello Word, can be extremly complex at kernel level.

funcgraph_entry quando entra
funcgraph_exit quando sai
O tempo mostrado é o que demorou la dentro.

Start just enables tracing, record puts it into a file.

5.5 No HZ

Good for power managment. but worst for performance

5.6 TuneD from redhat? see this

man tuned or man tuned-profiles

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/
started-with-tuned_monitoring-and-managing-system-status-and-
performance

5.7 mmiotrace

Trace interactions between drivers and hardware
It puts the system in uni mode only one CPU.

https://www.youtube.com/watch?v=68osT1soAPM
9:27
nao foi o gajo que inventou mas talvez seja de interesse.

6 HOST preparation

https://theydiffer.com/difference-between-a-cpu-and-a-core/
One thing to consider is that a core is inside a CPU, and depending on its architectural design, a CPU may have multiple cores.

1. run check.sh to see if IOMMU and VT-D is enabled and suported in your chip and Ensuring that the groups are valid, and confirm that iommu is on: dmesg — grep -i -e DMAR -e IOMMU;
2.

6.1 Multi Threading

https://johnysswlab.com/crash-course-introduction-to-parallelism-multithreading/
Multi threading is possible on a single CPU machine, just like multiple programs running at once on your PC. The CPU time is just shared between all the processes and threads. Obviously there is no gain if you split up a single calculation into several threads on a single CPU system, but on the other hand there can be other requirements than pure calculation power wich lead to a multi threaded program design. Simple example: a GUI program, updating/repainting the GUI in one thread and performing whatever action the user requests in another thread.

Real world examples of parallelism:

- parallelizing calculations of physical equations on GPU's to speed up tremendously;
- anything refarding network communications usually requires multiple threads to make sense. E.g. send/receive packets on one thread, run whatever thing the program does in a different threads;
- a server type application that can answer to multiple requests at once (think of a webserver – no one wnats it to stall for 100 users while it sends a page to another user);

- anything regarding hardware, e.g. kernel drivers working with input events can't stall the whole system while processing an event;
- CI scripts working on large code bases (e.g. doing clang-format) can be speed up on multiple core CPU's easily.

https://www.javatpoint.com/multithreading-models-in-operating-system
Multi-threading predates multi.core CPU's, the 'classical' approach multithread teaching is starting with a single-core and showing all possible execution orders for N processes competing for compute time.
Multiple threads (and processes) exist in single-core computers, anything before 2004 was single-core, and they ran a lot of programs with a lot of threads. You need to picture a thread (or process) as a unit requesting compute time to the operating system, it does not matter the amount of threads/processes and cores, the logic is the same: the operative system distributes the available computer power to the demanding threads and processes.

So what are some real world examples where we can use parallelism in projects ?
almost any computer hungry program where time is a constraint has moved to parallelism. Video games is the easiest example for any person, you have tons of individual small entities computing a lot of things (sound, light, bounces, interactions, logic).
Simulations have been using parallelism since inception.

what is hyper threads. answer this
When should we use Hyper thread? When we can share the cpu cahce, other than that it is not worth it. So basically we need to take advantege of NUMA. In our host by default we have this because it is only one NUMA node.

6.2 CPU PINNING–Affinity

Why do we want cpu pinning? https://www.youtube.com/watch?v=pDWCgmvcGPY
The main reason is the performance, the latency and performance & la-

tency.
CPU cachee is the main reason. (see the video)

https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF#Prerequisites

if we want to have a really good performance this is it.
We allocate a CPU to work on a specific task, it prevents unpredicted process to disturb the normal behaviour.
System managment threads will still run on this CPU. And it is ok, because some are for the linux kernel, and are needed.

See cpu topology:
lscpu -e

As such, the local CPU cache benefits (L1/L2/L3) are lost each time the host scheduler reschedules the virtual CPU thread on a different physical CPU. This can noticeably harm performance on the guest. CPU pinning aims to resolve this by limiting which physical CPUs the virtual CPUs are allowed to run on. The ideal setup is a one to one mapping such that the virtual CPU cores match physical CPU cores while taking hyperthreading/SMT into account.

Hyper-threading/SMT is simply a very efficient way of running two threads on one CPU core at any given time.
You will want to take into consideration that the CPU pinning you choose will greatly depend on what you do with your host while your virtual machine is running.

In this writing, we have discussed the “taskset” command tool and its options to set the CPU affinity of a given process. The “taskset” command is a Linux tool that helps to assign the CPU core to a process to execute and run only on the designated CPU cores. We have also checked how to set CPU affinity for the process in a running state.

If l3 s equal just need to pair it based on the threads, if L3 is splitted we need to group by L3 group
do man taskset to have more straw
taskset antes do comando qemu resolve esta merda
it is big endian for high byte transfered first
https://www.allaboutcircuits.com/uploads/articles/techart_endian_3.JPG
It has inverser priority 1000000 is the 8th cpu, while 0000001 is the first
Convert that to hex

ver o process id do qemu
ps -a — grep qemu

lstopo –of console –no-io –no-caches

-smp [cpus=]n[,maxcpus=cpus][,cores=cores][,threads=threads][,dies=dies][,sockets
set the number of CPUs to 'n' [default=1] maxcpus= maximum number of total cpus, including offline CPUs for hotplug, etc cores= number of CPU cores on one socket (for PC, it's on one die) threads= number of threads on one CPU core dies= number of CPU dies on one socket (for PC only) sockets= number of discrete sockets in the system

usar o system monitor para apanhar o uso do CPU

htop PID para filtrar logo

Dont forget that we need to pin a different CPU for each machine that we want to run simultaneous.

try this?
for i in 0 1 2 3; do vcpu="$vcpu -vcpu vcpunum=$i,affinity=$i" done

Problem?
The smp when used to for 1 process, 4 cores, 8 threads, gives worst performance than 1 processa 1 core. Why??

Configure set of CPUS dedicated for CPU pinning:

ls to /etc/systemd/system.conf
find CPUAffinity= cpus to pin for the host to boot

Via kernel we say tht we want to remove a particular cpu from scheduler completly:
N implies the use of the last cpu by default.
isolcpus= [KNL,SMP,ISOL] Isolate a given set of CPUs from disturbance. [Deprecated - use cpusets instead] Format: [flag-list,]¡cpu-list¿

Specify one or more CPUs to isolate from disturbances specified in the flag list (default: domain):

nohz
Disable the tick when a single task runs.

A residual 1Hz tick is offloaded to workqueues, which you need to affine to housekeeping through the global workqueue's affinity configured via the /sys/devices/virtual/workqueue/cpumask sysfs file, or by using the 'domain' flag described below.

NOTE: by default the global workqueue runs on all CPUs, so to protect individual CPUs the 'cpumask' file has to be configured manually after bootup.

domain
Isolate from the general SMP balancing and scheduling algorithms. Note that performing domain isolation this way is irreversible: it's not possible to bring back a CPU to the domains once isolated through isolcpus. It's strongly advised to use cpusets instead to disable scheduler load balancing through the "cpuset.sched_load_balance" file. It offers a much more flexible interface where CPUs can move in and out of an isolated set anytime.

You can move a process onto or off an "isolated" CPU via the CPU

affinity syscalls or cpuset.
¡cpu number¿ begins at 0 and the maximum value is "number of CPUs in system - 1".

managed_irq

Isolate from being targeted by managed interrupts which have an interrupt mask containing isolated CPUs. The affinity of managed interrupts is handled by the kernel and cannot be changed via the /proc/irq/* interfaces.

This isolation is best effort and only effective if the automatically assigned interrupt mask of a device queue contains isolated and housekeeping CPUs. If housekeeping CPUs are online then such interrupts are directed to the housekeeping CPU so that IO submitted on the housekeeping CPU cannot disturb the isolated CPU.

If a queue's affinity mask contains only isolated CPUs then this parameter has no effect on the interrupt routing decision, though interrupts are only delivered when tasks running on those isolated CPUs submit IO. IO submitted on housekeeping CPUs has no influence on those queues.

The format of ¡cpu-list¿ is described above.

We can edit sudo nano /etc/systemd/system.conf to use all other cpus and not the ones we want but it doees not isolate.
But for this we just use taskset directly

check interrupts command:
cat /proc/interrupts
Without an IO-APIC, interrupts from hardware will be delivered only to the CPU which boots the operating system (usually CPU#0).

CPU pinning will not magically increase performance, but it will increase responsiveness. By pinning a CPU and moving existing tasks to other CPUs that do not handle VM cores you ensure that execution of your VM will not be preempted by a task running on host system. This is a very important factor for keeping DPC latency down, and that saves you from audio crackling. VMs essentially are realtime applications and you want VM to be scheduled right here and right now. Gaming VMs even more so. One more thing - it is critical to pin VM CPUs to right host logical CPUs on hyperthreaded hosts. For example on my system same physical core runs hyperthreaded cores 0 and 6, therefore i pin VM cores 0 and 1 to host cores 0 and 6 (qemu hyperthreaded cores go in pairs 0-1, 2-3 and so on). This ensures better utilization of cache as well.

6.2.1 NUMA

it is tied together to cpu pinning
Non Uniform Memory Access
See if it is worth explore.

https://www.youtube.com/watch?v=w3yT8zJe0Uw
13'33"

See RT-Tests. They are anotehr set of tools.

Basic Definition

A node refers to the physical box, i.e. cpu sockets with north/south switches connecting memory systems and extension cards, e.g. disks, nics, and accelerators

A cpu socket is the connector to these systems and the cpu cores, you plug in chips with multiple cpu cores. This creates a split in the cache memory space, hence the need for NUMA aware code.

A cpu core is an independent computing with its own computing pipeline, logical units, and memory controller. Each cpu core will be able to service a number of cpu threads, each having an independent instruction stream but sharing the cores memory controller and other logical units.

CPUs (chips) and each chip may have multiple cores. Each core can execute one (or sometimes multiple) stream of instructions.

numastat -c qemu (run this with the VM running)

Generally, optimal performance on NUMA systems is achieved by limiting guest size to the amount of resources on a single NUMA node. Avoid unnecessarily splitting resources across NUMA nodes.
Use the numastat tool to view per-NUMA-node memory statistics for processes and the operating system.

NUMA vCPU Pinning
vCPU pinning provides similar advantages to task pinning on bare metal systems. Since vCPUs run as user-space tasks on the host operating system, pinning increases cache efficiency. One example of this is an environment where all vCPU threads are running on the same physical socket, therefore sharing a L3 cache domain.
Combining vCPU pinning with numatune can avoid NUMA misses. The performance impacts of NUMA misses are significant, generally starting at a 10% performance hit or higher. vCPU pinning and numatune should be configured together.
If the virtual machine is performing storage or network I/O tasks, it can be beneficial to pin all vCPUs and memory to the same physical socket that is physically connected to the I/O adapter.

The lstopo tool can be used to visualize NUMA topology. It can also help verify that vCPUs are binding to cores on the same physical socket. Refer to the following Knowledgebase article for more information on lstopo: https://access.redhat.com/solutions/62879

Pinning causes increased complexity when there are many more vCPUs than physical cores.

The easiest and most significant way to improve performance so far is to use CPU pinning in conjunction with a host-passthrough CPU mode.

NUMA node(s): we only have 1 available
lscpu
20minutos do video https://www.youtube.com/watch?v=pDWCgmvcGPY

A single thread can not be split between multiple cores:
A program needs to be written to have more than one thread (one per core), or there needs to be more than one program. If not then you won't use the cores.

Writing programs to use more cores is not trivial, and not all problems can be parallelised (written to run on more than one core). If a problem contains 20% essentially sequential code, then with an infinite number of processors, it will be no faster than 20% of original execution time (500% speed increase). Then there are the overheads (communication between threads).
If you don't have any application for the cores, then you are better off selling it, and getting a cheaper machine.
Each core will have a ton of parallelism, to deal with a single thread, but this is not visible. Now we are struggling to make a single core any faster, as we add cores. This works well at first.

Unix systems (such as Gnu/Linux e.g. Ubuntu), do a good job of using extra cores, up to around 2→4. Microsoft's Windows, gets improvements when you have a core for the virus scanner, and one for the defragmenter, and one for everything else.
After that it will only make a difference if you have applications designed for multi-core.

The numactl command can launch commands with static NUMA memory and execution thread alignment: numactl –hardware for example.

With qemu running you can check numastat -c qemu
we can see the alignement, the 5 pages that are there are from shared libraries of the system.

without numa scheduler (when there is more than 1 numa node) the processes will not be local to one node, they will be all over. It is not the best in perfomance.

Numa and shared memory don't work very well together.

Numa affinity can output from 15 to 20% https://www.youtube.com/watch?v=RLNMlsa173o 40:35 ´

6.2.2 Hyper threading enabled

enable enlightenments qemu
https://bbs.archlinux.org/viewtopic.php?id=233713
https://blog.wikichoon.com/2014/07/enabling-hyper-v-enlightenments-with-kvm.html

os novos parametros alocados ao cpu...
They enable Hyper-V enlightenments and those serious decrease CPU bottleneck.

6.2.3 IRQ AFFINITY

https://cs.uwaterloo.ca/ brecht/servers/apic/SMP-affinity.txt

The only reason to look at this value is that SMP affinity will only work for IO-APIC enabled device drivers.

6.3 GPU ISOLATION

Warning: Once you reboot after this procedure, whatever GPU you have configured will no longer be usable on the host until you reverse the manipulation. Make sure the GPU you intend to use on the host is properly configured before doing this - your motherboard should be set to display using the host GPU.

Most use cases for PCI passthroughs relate to performance-intensive domains such as video games and GPU-accelerated tasks. While a PCI passthrough on its own is a step towards reaching native performance, there are still a few ajustments on the host and guest to get the most out of your virtual machine.

1. all commands must be sudo;
2. go to /etc/default;
3. open grub (with nano or vim);
4. on the DEFAULT line add isocpus=cpu number or range. In a range you can use N to specify the last;
5. save the document;
6. update-grub;
7. reboot host;
8. stress test
9. isntall stress if you dont have it, then run;
10. stress -c 8 in one terminal and htop in the otehr and seee that all cpus except 7 are beeing stressed;
11. taskset -c "isolated cpu" stress -c 1 (1 is it is only 1) and check.

grub line to edit like this. preempt voluntary helps with ram allocation and implies fast boot .
GRUB CMDLINE LINUX DEFAULT="quiet splash intel iommu=on

preempt=voluntary isolcpus=7”

examples
isolcpus=1,2,10-20,100-2000:2/25

https://www.kernel.org/doc/html/latest/admin-guide/kernel-parameters.html?highlight=isolcpus

sudo apt-get install cpuset
cset – python app, see man page
we can create a group of isolated cpus withour reboot
sudo cset shield –cpu=$vCPU_PINNED –kthread=on

we can reset them via
sudo cset shield -r

with shield is simple, with proc is more complex
https://manpages.debian.org/unstable/cpuset/cset-shield.1.en.html

we can check the list:
cset set -l

You can exec new processes in specific cpusets and move tasks around existing cpusets. See the cset-proc(1).
We can create persistent cpusets.

nohz_full= cpu number to completly isolate the cpu from the kernel

To run process in isolated cpus we do
sudo chrt -r 1 taskset.., we set the priority to 1. See chrt man for more info

Melhor para o cenario servdor.

Linux control groups. I think it is the solution

To check the affinity:
ps -a — grep qemu
169570 pts/1 00:00:08 qemu-system-x86
taskset -c -p 169570
pid 169570's current affinity list: 3,7

6.4 Memory tunning and Latency

What is it? https://www.youtube.com/watch?v=Tkra8g0gXAU
The time from when an event is suppose to happen to the time it actually does happen.

trace-cmd record says we want to record it to a file.
19:39
trace-cm list -e will list all available events.

The reports we can let them run for a while or time it or just apend it to ls

31' for hystogram latency.

http://52.40.97.209/posts/qemu-optimization/post.php#memory

"

Preallocated Memory Preallocating memory is one of the easiest ways to improve VM performance. This setting does exactly what it says, it dedicates and area of memory that is the entire size of the guest VM's memory so that new blocks of memory do not have to be allocated by the host as the guest requests them. This does however leave all memory preallocated to the VM unusable by the host until the guest is halted.

Preallocated memory can be enabled with one QEMU argument. -mem-prealloc

we just need to add this -mem-prealloc to qemu

From nikhil test: [Yesterday 5:27 PM] Nikhil Santhosh
so I tried it out.. there is no improvement in performance

Nikhil Santhosh
but.. I saw that it improved the performance of the VM in the CPU isolation method

Nikhil Santhosh
it bought the simulation to 22 secs
like 1

Nikhil Santhosh
from 30 secs

together with all the other changes so far, I've configured HugePages (8 is our host max) dynamically with mem prealloc, then changing qemu processes priority after boot to be 99 (the highest in kernel).

This is all done via running one script. Since it is changing kernel stuff it needs to be runned as sudo su.

I feel that my qemu is more responsive and fast. but you have to try in your side.

6.5 HugePages

Hugepages for better RAM performance
https://linuxconfig.org/how-to-enable-hugepages-on-linux

QEMU will use 2MiB sized transparent huge pages automatically without any explicit configuration in QEMU or Libvirt, subject to some important caveats.

The described procedures have some drawbacks but will not necessarily net you a great performance advantage. According to Red Hat benchmarks, you should not expect more than a 2% performance gain from this over #Transparent huge pages.

Huge pages are a mechanism to manage memory in larger "chunks" than the usual 4KiB (in Linux, no idea what Windows does tbh). This is more efficient than managing many small chunks of memory, and it's easier for the operating system to keep track of said memory, which can improve performance.

The performance improvement is less bookkeeping of memory.
Sort of like how a warehouse only handles cases of product, not individually wrapped loose single products. Lots of overhead unless they are in hugeboxes..

https://www.educba.com/linux-hugepages/

The ¡hugepages/¿ element will cause guest RAM to be allocated from the default huge pages size pool, and will cause immediate allocation of those huge pages at QEMU startup. Huge pages are always locked into host RAM.

The ¡locked/¿ element is used to control other QEMU memory allocations that are separate from guest RAM. It ensures all non-guest RAM pages are also locked into host RAM.

If you are not trying to overcommit host RAM, then using huge pages on the host side is a very good idea for maximizing performance as it improves page table hit rates. This is true regardless of whether the guest

OS in turn uses huge pages. This performance benefit will apply to all workloads and all guest OS.

The KSM feature for when you are trying to overcommit host RAM (ie run many guests whose total RAM exceeds available host RAM). It tries to share RAM pages between guests to avoid risk of swapping during overcommit. KSM has a quite significant penalty in CPU consumption though, so its a tradeoff as to whether it is useful for a particular workload and/or guest OS.

Note that -mem-path /dev/hugepages will fall back to allocating normal RAM if it doesn't have enough hugepages, triggering (b) to fail. Using -mem-prealloc enforces the allocation using hugepages

Care should be taken with the size of hugepage used; postcopy with 2MB hugepages works well, however 1GB hugepages are likely to be problematic since it takes 1 second to transfer a 1GB hugepage across a 10Gbps link, and until the full page is transferred the destination thread is blocked.

unable to map backing store for guest RAM: Cannot allocate memory –¿ guest is requesting more memory than available.

https://www.youtube.com/watch?v=jTJ_X3fJ1Ik
Huge pages need continuous memory space to work.
The number og HP tells us the number of pages that we need to map between virtual memory and physical memory.
It is completly dependent on the processors version and architecture.
Why do we use it? Less page table lookups in theory imply more "performance". So the number of Translation Lookaside Buffer (TLB) will be smaller, because we can fit more of this pages in the TLB cache.
TLB is like a cache in the core of the cpu that stores some addresses in a table. This table is searched first, and when it has what we need it is a *hit* and only spends 1 memory access. But when what we need is not there, it is a *miss* and we spend 2 memory access to achieve the result.
TLB has a limit of a few hundred pages, so how can we improve on it

(how can we decrease the number of misses)?
Increasing TLB size by itself is to expensive in terms of resources, so the best slution is to increase the page size, so we will have less pages to map. A regular page sin linux has 4Kb, a HugePage (HP) starts from 2Mb, 1Gb, and more in some special cases.

What is transparent huge pages? Is the case where the application doesn't have knowledge of what a HP is, but the OS does, so it will transparently allocate HP to where it seems fit. Mind that it needs to be a continuos block of memory.
https://www.youtube.com/watch?v=RLNMlsa173o 41:35 –¿ excellent explanation.

CPU flag of 1gb hugepage support and guest OS support/enabling are not enough to get 1 gb hugepages working in virtualized environment.
The idea of huge pages both on PMD (2MB or 4 MB before PAE and x86_64) and on PUD level (1 GB) is to create mapping from aligned virtual region of huge size into some huge region of physical memory (As I understand, it should be aligned too). With additional virtualization level of hypervisor there are now three (or four) memory levels: virtual memory of app in guest OS, some memory which is considered as physical by guest OS (it is the memory managed by virtualization solution: ESXi, Xen, KVM, ...), and the real physical memory. It's reasonable to assume that hugepage idea should have the same size of huge region in all three levels to be useful (generate less TLB miss, use less Page Table structures to describe a lot of memory - grep "Need bigger than 4KB pages" in the DickSites's "Datacenter Computers: modern challenges in CPU design", Google, Feb2015) – pdf DataCenter.pdf.
So, to use huge page of some level inside Guest OS, you should already have same sized huge pages in physical memory (in your Host OS) and in your virtualization solution. You can't effectively use huge page inside Guest when they are not available for you host OS and Virtualization software. (Some like qemu or bochs may emulate them but this will be from slow to very slow.) And when you want both 2 MB and 1 GB huge pages: Your CPU, Host OS, Virtual System and Guest OS all should support them (and host system should have enough aligned continuous physical memory to allocate 1 GB page, you probably can't split this page

over several Sockets in NUMA).
Don't know about ESXi, but there as some links for RedHat and some(?) linux virtualization solutions (with libvirtd). In "Virtualization Tuning and Optimization Guide" manual hugepages are listed for Host OS: 8.3.3.3 Enabling 1 GB huge pages for guests at boot or runtime :

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/virtualization_tuning_optimization_guide-memory-huge_pages-1gb-runtime

Procedure 8.2. Allocating 1 GB huge pages at boot time

To allocate different sizes of huge pages at boot, use the following command, specifying the number of huge pages. This example allocates 4 1 GB huge pages and 1024 2 MB huge pages:
'default_hugepagesz=1G hugepagesz=1G hugepages=4 hugepagesz=2M hugepages=1024' Change this command line to specify a different number of huge pages to be allocated at boot.

Note The next two steps must also be completed the first time you allocate 1 GB huge pages at boot time.

Mount the 2 MB and 1 GB huge pages on the host:

```
mkdir /dev/hugepages1G
mount -t hugetlbfs -o pagesize=1G none /dev/hugepages1G
mkdir /dev/hugepages2M
mount -t hugetlbfs -o pagesize=2M none /dev/hugepages2M
```

Restart libvirtd to enable the use of 1 GB huge pages on guests:

```
service restart libvirtd
```

1 GB huge pages are now available for guests.

For Ubuntu and KVM: "KVM - Using Hugepages"

By increasing the page size, you reduce the page table and reduce the pressure on the TLB cache. ... vm.nr_hugepages = 256 ... Reboot the system (note: this is about physical reboot of host machine and host OS) ... Set up Libvirt to use Huge Pages KVM_HUGEPAGES=1 ... Setting up a guest to use Huge Pages

For Fedora and KVM (old manual about 2MB pages): https://fedoraproject.org/wiki/Features/KVM_Huge_Page_Backed_Memory

ESXi 5 had support of 2MB pages, which should be manually enabled: How to Modify Large Memory Page Settings on ESXi

For "VMware's ESX server" of unknown version, from March 2015 paper: BQ Pham, "Using TLB Speculation to Overcome Page Splintering in Virtual Machines", Rutgers University Technical Report DCS-TR-713, March 2015:

Lack of hypervisor support for large pages: Finally, hypervisor vendors can take a few production cycles before fully adopting large pages. For example, VMware's ESX server currently has no support for 1GB large pages in the hypervisor, even though guests on x86-64 systems can use them.

Newer paper, no direct conclusion about 1GB pages: https://rucore.libraries.rutgers.edu/rutgers-lib/49279/PDF/1/

We find that large pages are conflicted with lightweight memory management across a range of hypervisors (e.g., ESX, KVM) across architectures (e.g., ARM, x86-64) and container-based technologies.

Old pdf from VMWare: "Large Page Performance. ESX Server 3.5 and ESX Server 3i v3.5". https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper — only 2MB huge pages are listed as supported

VMware ESX Server 3.5 and VMware ESX Server 3i v3.5 introduce 2MB large page support to the virtualized environment. In earlier versions of ESX Server, guest operating system large pages were emulated using small pages. This meant that, even if the guest operating system was using large pages, it did not get the performance benefit of reducing TLB misses. The enhanced large page support in ESX Server 3.5 and ESX Server 3i v3.5 enables 32-bit virtual machines in PAE mode and 64-bit virtual machines to make use of large pages.

grep -i huge /proc/meminfo

HugePages Total is the size of the pool of huge pages and is configured using /proc/sys/vm/nr_hugepages
HugePages Free is the number of huge pages in the pool that are not yet allocated.
HugePages Rsvd is short for "reserved," and is the number of huge pages for which a commitment to allocate from the pool has been made, but no allocation has yet been made. Reserved huge pages guarantee that an application will be able to allocate a huge page from the pool of huge pages at fault time.
HugePages Surp is short for "surplus," and is the number of huge pages in the pool above the value in /proc/sys/vm/nr_hugepages. The maximum number of surplus huge pages is controlled by /proc/sys/vm/nr_overcommit_hugepages.
Hugepagesize is the default hugepage size (in Kb).

Note: if hugepages in grub are not set to 1gb at the moment i dont have a solution to change them from the default 2MB
add to grub: transparent_hugepage=never hugepagesz=1G

hugepages=100 default_hugepagesz=1G
The hugepages=100 is the number of hugepages we want. It is the better way.
AnonHugePages is reference to THP.

or never wich is best?
CPU flag of 1gb hugepage support and guest OS support/enabling are not enough to get 1 gb hugepages working in virtualized environment.
How bad can 1GB pages be

we cannot have both 2Mb and 1Gb enable at the same time. We will have one or the other. We can only do that during boot time unfurtunatly.

Tests:

1. 2MB vs 1GB with number of pages set–¿ 32seconds vs 27seconds (vLab time)
2. Default Vs -mem-prealloc just –¿ 35s vs 1'09"
3. THP ON or OFF –¿ ON (30) is better than off, but in a case where we have an huge amount of sapce it may be the opossite.

4. server –¿ 25s with huge pages 27 without
5. 18seconds with 1GB pages on nikihil laptop. IMPORTANT we need to close all the chrome tabs to have the best performance.

Conclusions:

One of the benefits of using hugepages is that it pins the memory so the swapping doesnt touches it.In databases it is a must

See hugepages.txt that i have done

[10:29 AM] Nikhil Santhosh
Francisco Manuel Santoswe just need to verify if it is worth rebooting or if it is similar to the othersafe to conclude it is worth rebooting like 1

[10:29 AM] Francisco Manuel Santos
ok. if it is worth rebooting, it is worth isolate and pin in grub as well

6.6 Linux Process Scheduler Tuning

Sched fifo .sh e está.

Meter palha.

6.7 CPU governor

watch lscpu

6.8 Cgroups

cgroups is the foundation for containers. They are the buildingblocks from the kernel where containers are build with.
They allow us to subset a system.
https://www.youtube.com/watch?v=RLNMlsa173o 49'
Container_group
here we have a complete control of numa.

6.9 GPU PASSTHROUGH

Conclude that at this moment in time we dont need to dig deep into this.
Yup thats it
we need to test it

Placa gráfica
- Intel Ultra HD
- Opcional: NVIDIA® GeForce® Mx330 com 2 GB GDDR5

sudo lshw -C vídeo

output:
00:02.0 VGA compatible controller [0300]: Intel Corporation UHD Graphics [8086:9b41] (rev 02)

2d:00.0 3D controller [0302]: NVIDIA Corporation GP108M [GeForce MX330] [10de:1d16] (rev a1)

To get device detailed info:
sudo lspci -v -s 2d:00.0

to see hardware info:
sudo apt install hardinfo

for what i am reading about GPU pass-through we must have at least 2 GPU's (one internal and one external, or both external) for this to work. Because, and i am now quoting "Attention! After the upcoming steps, the guest GPU will be ignored by the host OS. You have to have a second GPU for the host OS now!".
this has been mentioned in multiple sources. and it requires reboot to be enabled at least once we are reaching a point of 2 possible scenarios:
1.all set up with the settings we can do without reboot;
2.Set up a machine with reboot and "better" options, so the kernel knows

by default what it has to do
Both have pros and cons

7 Guest tune

Install OS as enterprise for better regitry access
Disable indexing windows 10
disable windows defender
disable cortana

8 KernelShark Before and After CPU pinning

In linux every process has a unique PID (process identification value). that's how we can track it. We can check the process using ps in the terminal.
Threads are the smallest unit of processing that can be performed in an operating system. In the modern days multiple threads can be done in a single process.
each thread is considered a single porcess in the kernel.

9 Questions that need a response?

1. What is a good way to benchmark a machine?
2. Compare the same app or math computation with and withouth KVM enabled.
3. is time and memory a good benchmark?
4. running for the second time is way more fast. why?
5. why use W10? (thats not may goal)
6. **How to avoid loading boo everytime QEMU runs?**
7. **How to increase the performance when compiling QEMU**
8. try to increase clock on a CPU for highr time
9. gpu passthroug?
10. looking glass????
11. does adding sudo to the qemu run helps? Nope
12. What is a cluster vs l2 cache?
13. Cluster size and Clean time?
14. **How it works?**
15. **Why it works?**

Note. Multiplciar floats, its a good test? Comando htop para visualizar alguns parametros

10 Benchmark tools

The best benchmark is against the most basic qemu run. laun vs launch tuned

Cinebench -> install it on the virtual machine to test effect of arguments.

https://en.wikipedia.org/wiki/Benchmark_(computing)
Who Should Use Cinebench? ... hardware manufacturers can utilize the feedback in optimizing their latest products. Any computer owner can evaluate their individual system. Unlike abstract benchmarks, which only test specific functions of CPUs, Cinebench offers a real-world benchmark that incorporates a user's common tasks within Cinema 4D to measure a system's performance.

Cinebench R23 now supports Apple's M1-powered computing systems
Cinebench is now based on the latest Release 23 code using updated compilers, and has a minimum runtime activated by default (previously hidden in preferences) Cinebench R23 provides improved benchmark accuracy for current and next generation CPUs to test if a machine runs stable on a high CPU load, if the cooling solution of a desktop or notebook is sufficient for longer running tasks to deliver the full potential of the CPU, and if a machine is able to handle demanding real-life 3D tasks.
Users now have the option to directly test the single core performance without manually enabling the "Advanced benchmark" option. The "Advanced benchmark" allows users to set arbitrary minimum runtimes to stress test the hardware for even longer periods of time.
Because of the code and compiler changes, Cinebench score values are readjusted to a new range so they should not be compared to scores from previous versions of Cinebench.
Cinebench R23 does not test GPU performance.
Cinebench R23 will not launch on unsupported processors. On systems lacking sufficient RAM to load the test scene, a warning will be displayed and the CPU benchmark will not be executed.
Background tasks can significantly influence measurement and create

diverse results. It's always a good idea to shut down all running programs and disable any virus checking or disk indexing but it's impossible to eliminate all background processes. Modern operating systems perform various background tasks that cannot or should not be disabled, even though they could have a minor influence on the results.
Test results can vary slightly because it's impossible to disable every background task of the operating system. These tasks are one factor that may have a slight influence on measurements. Also, modern computers and graphics cards dynamically adjust clock speeds based on environmental conditions like power and temperature. For instance, processors will reduce clock speed when running too hot to allow for cooling and prevent damage. With many modern processors, the reverse is also true. They are able to overclock themselves when the temperature is low enough.
Therefore, a system freshly started in a relatively cool environment will typically run faster than the same system that has been performing benchmarks for several hours in a heated office.

Cinebench is a CPU-based rendering benchmark which tells you how fast your CPU can render a scene using all CPU threads. The score can be compared to other CPUs to determine the relative performance between them. The numbers won't tell you anything in regards to video production or gaming as those workloads are different.

Cinebench without vga see excel file lazy_refcounts was the best

Geekbench
Takes too much time. The bse single core score is 815
But...

Uset -rtc!

qemu-system-x86_64 -cpu max -enable-kvm -smp cores=1,threads=20 -name 'cluster' -rtc base=localtime,clock=host -drive file=/home/franciscosantos/Desktop/QEMU/Virtual_Disks/disk2.qcow2,cache=write
-m 4G

TO use on the guest:
Stress teste: use the sme url but with stress, see link tabs www.cpux.net/cpu-benchmark.online

11 Create snapshot

This is for internal snapshot
https://www.youtube.com/watch?v=EaTPEy2Tw_c
At 6:47 we can see how to enter via terminal to the VM.

1st boot the VM then create snapshot in VMManager
virsh snapshot-create-as –domain nome da VM –name ”nome entre aspas do snap”
Example:
virsh snapshot-create-as –domain dummy1 –name teste
click enter
”Domain snapshot ....” should print in the terminal

See if it created:
virsh snapshot-list dummy1
Name Creation Time State
————————————————-
teste 2022-03-18 14:49:23 +0000 running

virsh shutdown dummy1
Domain dummy1 is being shutdown

virsh list –all
Id Name State
————————————-
- dummy1 shut off

Revert to the snapshot:
virsh snapshot-revert –domain dummy1 –snapshotname teste –running
Click on open to enter VM
State == running it means that the VM was running when the snapshot was created.
virsh start dummy1 par aexecutar? Testar isto

IF BIOS all ok
IF UEFI we have to change the xml file. See bookmark https://www.youtube.com/watch?v=1SDvth66i-4

NOT YET TESTED!
This is for external snapshots
https://www.youtube.com/watch?v=1SDvth66i-4 8 minuts Virt-manager doesnt support external snapshots, so we do it from the terminal.
first turn the VM off
then virtual machine – details – overview – xml tab
(enable xml editing)
fazer cenas
remove metadata associated to external machine
delete external snapshot file itself
Snapshot is then restored
start the virutal machine.

Qemu monitor 1st STOP 2nd savevm name run the load via terminal

BETTER YET snapshot repository:
1 – Create external .qcow2 file to store the snapshot:
qemu-img create -f qcow2 -b disk2.qcow2 snapshot.qcow2
2 – Run qemu with this new file as the virtual disk:
qemu-system-x86_64 -name 'dummy3' -cpu max -enable-kvm -smp cores=1,threads=20 -drive file=/home/franciscosantos/Desktop/QEMU/Virtual_Disk
-m 4G

3 – We put the machine in the state that we want and enter the qemu monitor to save the snapshot:
stop
savevm name
4 – To load this snapshot we run:
qemu-system-x86_64 -name 'dummy3' -cpu max -enable-kvm -smp cores=1,threads=20 -drive file=/home/franciscosantos/
Desktop/QEMU/Virtual_Disks/snapshot.qcow2,cache=writethrough -m 4G -loadvm snap1
Note: The new file will be smaller than the original, and all the changes will be on this one. Be carefull with this for future changes.

List snapshots inside qcow2 image:
qemu-img snapshot -l image name

delete snapshot:
emu-img snapshot -d snap_dummy4 disk2.qcow2

Save changes to the original disk image
https://techpiezo.com/linux/use-and-implementation-of-backing-file-and-snapshot-in-qemu-kvm/

Trying to benchmark guest software under QEMU emulation is at best extremely difficult. QEMU's emulation does not have performance characteristics that are anything like a real hardware CPU's: some operations that are fast on hardware, like floating point, are very slow on QEMU; we don't model caches and you won't see anything like the performance curves you would see as data sets reach cache line or L1/L2/etc cache size limits; and so on.

Important factors in performance on a modern CPU include (at least):

raw instruction counts executed
TLB misses
branch predictor misses
cache misses

QEMU doesn't track any of the last three and only makes a vague attempt at the first one if you use the -icount option. (In particular, without -icount the RDTSC value we provide to the guest under emulation is more-or-less just the host CPU RDTSC value, so times measured with it will include all sorts of QEMU overhead including time spent translating guest code.)

Assuming you're on an x86 host, you could try the -enable-kvm option to run this under a KVM virtual machine. Then at least you'll be looking at the real performance of a hardware CPU, though you will still see some noise from the overhead as other host processes contend for CPU with the VM.

Five sections, in the Libvirt virtual machine configuration, are crucial in order to optimize the virtual machines performance:
CPU pinning
CPU model information
Hyper-V enlightments
Clock settings
Hugepages
https://mathiashueber.com/performance-tweaks-gaming-on-virtual-machines/
List CPU info on the terminal: lscpu

Inside the kernel thr priorities are reversed, so the higher the priority the lowest priority it has. So 98 priority in kernel is a pryority 2, and a 0 priority is a 100 so its first.

circular dependencies?

Kernel space vs User space

We cannot push something into the kernel, it must be pulled in - Steven Rosted
https://www.youtube.com/watch?v=4UY7hQjEW34

mutex is a lock (wait) and release mechanism.
Semaphores are signalling mechanisms that signal to processes the state of the Critical section in OS and grant access to the critical section accordingly.

Mutex works in user-space and Semaphore for kernel
Mutex is for Threads, while Semaphores are for processes.
A thread may acquire more than one mutex

When I am having a big heated discussion at work, I use a rubber chicken which I keep in my desk for just such occasions. The person holding the chicken is the only person who is allowed to talk. If you don't hold the chicken you cannot speak. You can only indicate that you want the chicken and wait until you get it before you speak. Once you have finished speaking, you can hand the chicken back to the moderator who will hand it to the next person to speak. This ensures that people do not speak over each other, and also have their own space to talk.

Replace Chicken with Mutex and person with thread and you basically have the concept of a mutex.

Of course, there is no such thing as a rubber mutex. Only rubber chicken. My cats once had a rubber mouse, but they ate it.

Of course, before you use the rubber chicken, you need to ask yourself whether you actually need 5 people in one room and would it not just be easier with one person in the room on their own doing all the work. Actually, this is just extending the analogy, but you get the idea.

Interrupts are threads, and they can be traced by their priority.

Latency always happens.

Both processes and threads are independent sequences of execution. The typical difference is that threads (of the same process) run in a shared memory space, while processes run in separate memory spaces.

I'm not sure what "hardware" vs "software" threads you might be referring to. Threads are an operating environment feature, rather than a CPU feature (though the CPU typically has operations that make threads efficient).

Erlang uses the term "process" because it does not expose a shared-memory multiprogramming model. Calling them "threads" would imply that they have shared memory.

In kernel we dont do recursive calls because it can blow up a stack with ease.

Choose kernel image when boting qemu

13 Cross compiling

canadian cross compiling
Cross compiling is ver hard.
https://landley.net/writing/
53mints is where he talks about this.

In sum if we can avoid cross compiling we should try!

Native compiling under emulation is so much easier. (see if native compiling is the same as normal compiling. i think yes).

the guy Rob Landley made mkroot.

see muscl-cross-make

the difference between normal compiling and cross compiling is....

13.0.1 Canadian Cross Compiling

14 FLUTTER APP

do it with the team

15 Conclusion

But i think, regarding the time that i have left to present this (+/- 2months), that for the report itseflt i will conclude with the HugePages subjects (since it was the one that impaacted) and introduce the Real Time Kernel Patches as the road to continue exploring (and i will do that after the report is done). I will mention as well that one of th goals is to have a user interface, and some other scripts beyond bash to auto figure this out
yes i agree with you... we covered some major points here (atleast ones which I was more curious of)... this is enough for us to conclude that in our case the limitation is the CPU frequency because of the performance of the Vlab tool.
and one final outcome can be a small tool to help the developers to configure a system for optimal perfromance of any VM they are trying to run in it.. it can even be a small script .. some simple UI will make more of an impact visually... but if you can contribute any simple tool to CTW through this internship effort then it would be an awesome way to launch your career here

Notably the use of hugepages has helped many users, but its usefulness in the context of GPU pass-through is questionable and the use of core isolation which prevents the hosts from running tasks on core passed to your VM.

but we can conclude that the performance is correlated to the memory managment of the virtual machine.

Best results so far:
GRUB CMDLINE LINUX DEFAULT="quiet splash intel iommu=on preempt=voluntary isolcpus=3,7 hugepagesz=1G hugepages=20 default_hugepagesz=1G transparent_hugepage=never"
update grub
launch qemu with taskset -c ...

16 Uteis

How Do I Make A Script Executable?
Create a script by going to the directory where it will be created.
Your file should be made as simple as. sh.
In order to create the script, you will have to work with an editor.
Use command chmod +x *fileName¿ to create the script executable.
You can script by using "/*fileName" instead of [.]"file".

qemu-img info nomeimg.qcow2
qemu-img map nomeimg.qcow2

We can link a qemu imge to a qemu machine so we don't have to pass it always.

-p option shows the percentual progress of the task

find folders:
sudo find . -name QEMU -¿ procurar merdas

nproc no terminal dá-nos o numero de processadores que o host tem.

Find inside files:
grep -r 'switch_fpu_return*'

to cal a function inside a script
. ./cset.sh && list_cset

look pid filter by chromium and grab the first column
ps -a — grep qemu — awk ' print $1 ' — xargs -I taskset -c -p

To parse a line
lscpu -p — sed -n '5 p'

Array:
echo ”$cpu[0]:0:1”

Clear variables
exec env –ignore-environment /bin/bash

source in bash need to be locally done, so if we need inside a function we put it inside function.
vmstat -s
man vmstat for statistics

long-command & sleep 300; do-this-after-five-minutes

grep -r hello /sys/ 2¿ /dev/null
to send output to null

Jsut shows the current path on terminal
export PS1=’\w:’