openQA Project - coordination #64746

[saga][epic] Scale up: Efficient handling of large storage to be able to run current tests efficiently but keep big archives of old results

2020-03-24 10:24 - mkittler

Status:	Resolved	**Start date:**	2020-03-18
Priority:	Normal	**Due date:**	
Assignee:	okurz	**% Done:**	100%
Category:	Feature requests	**Estimated time:**	0.00 hour
Target version:	Ready		
Difficulty:			

Description

ideas

- OSD specific: cron job which checks if df is showing low space available for /results, then call find … delete videos older than <…>
- DONE: ~~Default to NOVIDEO=1 for all scenarios that set a MAX_JOB_TIME higher than default 2h~~ -> gh#os-autoinst/openQA#3112 commit ab74357
- Improve video compression codecs (a 20MB video.ogv can be compressed easily to 14MB video.ogv.xz, that can be improved)
- Setup additional storage server for old assets and results
  - Use overlayfs to make archived assets/results appear alongside the regular assets/results
  - Move old assets/results at some point from the regular web UI host to the storage server, e.g. automatically via a Minion job
    - Mark jobs with archived results as such

Subtasks:

action # 64574: Keep track of disk usage of results by job groups	**Resolved**
action # 67342: Support using a "generic" video encoder like ffmpeg instead of only rel...	**Resolved**
action # 75256: Try out AV1 video codec as potential new default	**Resolved**
action # 76987: re-encode some videos from existing results to save space	**Resolved**
coordination # 80546: [epic] Scale up: Enable to store more results	**Resolved**
action # 69577: Handle installation of the new "Storage Server"	**Resolved**
action # 77890: [easy] Extend OSD storage space for "results" to make bug investigation...	**Resolved**
action # 88546: Make use of the new "Storage Server", e.g. complete OSD backup	**Resolved**
action # 90629: administration of the new "Storage Server"	**Resolved**
action # 91347: [spike][timeboxed:18h] Support for archived jobs	**Resolved**
action # 91779: Add monitoring for storage.qa.suse.de	**Resolved**
action # 91782: Add support for archived jobs	**Resolved**
action # 91785: UI representation for archived jobs	**Resolved**
action # 92788: Use openQA archiving feature on osd size:S	**Resolved**
coordination # 96974: [epic] Improve/reconsider thresholds for skipping cleanup	**Resolved**
action # 98922: Run asset cleanup concurrently to results based on config	**Resolved**
action # 103954: Run asset cleanup concurrently to results based on config on o3 as well	**Resolved**
action # 103953: Use openQA archiving feature on o3 size:S	**Resolved**

Related issues:

Related to openQA Infrastructure - action #30595: [ppc64le] Deploy bcache on ...	**New**	**2018-01-22**
Related to openQA Project - coordination #34357: [epic] Improve openQA perfor...	**New**	**2017-08-25**
Related to openQA Infrastructure - action #58805: [infra]Severe storage perfo...	**Resolved**	**2019-10-29**
Related to openQA Infrastructure - action #66709: Storage server for OSD and ...	**Resolved**	**2020-05-12**
Copied to openQA Project - coordination #92323: [saga][epic] Scale up: Fine-g...	**New**	**2020-05-20**
Copied to openQA Project - coordination #110833: [saga][epic] Scale up: openQ...	**New**	**2022-05-09**

History

#1 - 2020-03-24 11:00 - cdywan

mkittler wrote:

> ## ideas
>
> use innovative storage solutions

What real technology are you referring to here?

#2 - 2020-03-27 20:56 - okurz

cdywan wrote:

> What real technology are you referring to here?

Something better than single fixed, unflexible volumes. E.g. dm-cache, lvmcache, bcache, SUSE enterprise storage. Maybe also splitting stored results in openQA by "recent, active" and "old, archived" and then put both categories in different folders which can be mounted from different storage locations, e.g. fast, expensive for "recent, active" and slow, cheap, big for "old, archived"

EDIT: 2020-04-08: What I had been reading and thinking:

- http://strugglers.net/~andy/blog/2017/07/19/bcache-and-lvmcache/ is a very nice blog post comparing HDD, SDD, bcache, lvmcache with git tests and fio. Overall result: bcache can reach results very near to SSD, lvmcache is good but stays below bcache but lvmcache is much more flexible and probably easier to migrate to and from
- ZFS seems to provide good support for using fast SSD for caching but that is not an easy option for us due to no support in our OS for now
- mitiao already worked on bcache for our tmpfs workers in #30595 but I think overall there was great misunderstanding and confusion and obviously no resolution. Still, the mentioned hints can be evaluated as well. In this specific case I wonder though how much RAM+HDD is comparable to SSD+HDD, maybe we need special solutions for both cases, not a common, too generic one for both
- Discussed with nsinger: tmpfs should only use as much RAM as is actually used within the tmpfs so maybe for our tmpfs workers we should just allocate bigger tmpfs (as big as RAM) and let it use what is there? Or configure "dirty_ratio" and "dirty_background_ratio" to effectively allow caching 100% but start writing to persistent storage in the background as soon as possible to avoid I/O spikes? Like "dirty_ratio=100%", "dirty_background_ratio=0%"? But that applies to all devices. It is also possible to configure parameters per device though. nsinger also suggested https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/delay.html for local experiments or a qcow image on tmpfs that we map into a VM as "fast storage" and a qcow image on HDD/SDD as "slow storage" -> #30595
- ramfs has no size limit and is not using swap, risky.

EDIT: 2020-04-20: Some more:

- https://unix.stackexchange.com/questions/334415/dirty-ratio-per-device describes that we can set a dirty ratio per device which we could use for tmpfs workers. The low-level details are in https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-class-bdi -> #30595
- https://unix.stackexchange.com/questions/237030/how-safe-is-it-to-increase-tmpfs-to-more-than-physical-memory describes how tmpfs can be configured with even bigger size than complete RAM size to use swap. For tmpfs workers, maybe just configure a much bigger tmpfs and supply a big swap -> #30595
- https://superuser.com/questions/1060468/confused-whether-to-switch-from-ext2-to-ext4-or-not/1060818#1060818 recommends ext4 over ext2 due to mentioned features, e.g. "like extents, pre-allocation, delayed allocation and multiblock allocators which all contribute to reduce fragmentation and therefore extend your SSD life."
- https://www.thomas-krenn.com/de/wiki/SSD_Performance_optimieren describes how gpt+ext4 can perform better. We should ensure we use GPT at least but if we do not use a partition table at all maybe that also has an impact?
- https://unix.stackexchange.com/questions/155784/advantages-disadvantages-of-increasing-commit-in-fstab describes Advantages/disadvantages of increasing "commit" in fstab but we can experiment with higher commit values and see for performance impact
- https://heiko-sieger.info/tuning-vm-disk-performance/ describes that qemu-img create -f raw -o preallocation=full vmdisk.img 100G can bring best performance for VMs, i.e. "raw" with "preallocation=full". Also recommend to use the virtio driver within qemu VMs with "iothread", e.g. -object iothread,id=io1 -device virtio-blk-pci,drive=disk0,iothread=io1 -drive if=none,id=disk0,cache=none,format=raw,aio=threads,file=/path/to/vmdisk.img, "note the aio=threads options, preferred option when storing the VM image file on an ext4 file system. With other file systems, aio=native should perform better. You can experiment with that."
- http://mail-archives.apache.org/mod_mbox/cloudstack-users/201708.mbox/raw/%3CF7FFAF25-5228-47F1-9DD2-7A828E071520@gmail.com%3E/ recommends sparse qcow files to prevent multiple writes (actual write and image size extend), using "writeback" caching mode, use "fat" allocation with qcow which takes more time initially but helps further down, recommends XFS for datastore over ext?
- http://www.linux-kvm.org/page/Tuning_KVM recommends qemu parameters -cpu host (equivalent to libvirt selection "host-passthrough") and also if=virtio for storage. A command qemu command like in openQA tests looks like /usr/bin/qemu-system-x86_64 -only-migratable … -cpu qemu64 … -smp 1 -enable-kvm … -S -device virtio-scsi-pci,id=scsi0 -blockdev driver=file,node-name=hd0-file,filename=/var/lib/openqa/pool/1/raid/hd0,cache.no-flush=on -blockdev driver=qcow2,node-name=hd0,file=hd0-file,cache.no-flush=on -device virtio-blk,id=hd0-device,drive=hd0,serial=hd0 -blockdev driver=file,node-name=cd0-overlay0-file,filename=/var/lib/openqa/pool/1/raid/cd0-overlay0,cache.no-flush=on -blockdev driver=qcow2,node-name=cd0-overlay0,file=cd0-overlay0-file,cache.no-flush=on -device scsi-cd,id=cd0-device,drive=cd0-overlay0,serial=cd0. As we never migrate machines to other hosts just boot the qcow images on other machines "host-passthrough" might not be a problem.
- https://qemu.weilnetz.de/doc/qemu-doc.html#qemu_005fimg_005finvocation describes how one can run benchmarks against qemu drive files using qemu-img bench which can help us to find out performance of pool filesystems maybe more easily than running complete openQA test runs
- https://blog.frehi.be/2011/05/27/linux-performance-improvements/ mentions KSM to save memory on workers running the same OS in multiple VMs; Also mentions "deadline" scheduler for hosts mainly running VMs

EDIT: 2020-05-01: More:

- https://wiki.archlinux.org/index.php/Ext4#Improving_performance also has a nice list for ext4 filesystems, e.g. what we use on o3 workers , e.g. now using noatime,data=writeback,commit=1200 on aarch64.o.o , did not try "barrier=0" yet.

EDIT: 2020-05-13: coolo did some research years ago, see https://github.com/os-autoinst/os-autoinst/pull/664

EDIT: 2020-06-13: in combination zram might help us as well

#3 - 2020-04-10 06:44 - okurz

- Status changed from New to In Progress

- Assignee set to okurz

#4 - 2020-04-20 04:32 - okurz

- Subject changed from [epic] Handle large storage efficiently to be able to run current tests but keep big archives of old results to [saga][epic] Handle large storage efficiently to be able to run current tests efficiently but keep big archives of old results

#5 - 2020-04-21 07:50 - okurz

- Related to action #30595: [ppc64le] Deploy bcache on tmpfs workers added

#6 - 2020-04-21 07:51 - okurz

- Related to coordination #34357: [epic] Improve openQA performance added

#7 - 2020-04-23 11:17 - okurz

- Related to action #58805: [infra]Severe storage performance issue on openqa.suse.de workers added

#8 - 2020-04-28 06:48 - cdywan

- Target version set to Current Sprint

Should this be on Feedback? Should it be in the Current Sprint? I'm not clear on the current status.
It looks like a research ticket which needs to be refined further but you set it to In Progress.

#9 - 2020-04-28 06:50 - cdywan

- Target version deleted (Current Sprint)

#10 - 2020-04-28 12:39 - okurz

cdywan wrote:

> Should this be on Feedback? Should it be in the Current Sprint? I'm not clear on the current status.
> It looks like a research ticket which needs to be refined further but you set it to In Progress.

Yes, I set it "In Progress" because I have many articles in my reading backlog. Not "Feedback" because I am not waiting for anything, just doing more than just this task. Please see that I updated my previous comments with pretty recent updates. I was asked to add less comments in tickets to prevent "too many mail notifications" hence I took that approach instead.

#11 - 2020-05-14 11:51 - okurz

- Related to action #66709: Storage server for OSD and monitoring added

#12 - 2020-05-26 09:01 - okurz

- Description updated

#13 - 2020-07-03 20:49 - okurz

- Description updated

- Status changed from In Progress to Blocked

- Target version set to Ready

#14 - 2020-09-08 14:25 - mkittler

- Description updated

#15 - 2020-10-12 13:31 - szarate

- Tracker changed from action to coordination

- Status changed from Blocked to New

#16 - 2020-10-12 13:44 - szarate

See for the reason of tracker change: http://mailman.suse.de/mailman/private/qa-sle/2020-October/002722.html

#17 - 2020-10-13 11:55 - okurz

- Status changed from New to Blocked

#18 - 2020-11-21 23:05 - okurz

- Subject changed from [saga][epic] Handle large storage efficiently to be able to run current tests efficiently but keep big archives of old results to [saga][epic] Scale up: Handle large storage efficiently to be able to run current tests efficiently but keep big archives of old results

#19 - 2021-04-28 21:20 - okurz

- Subject changed from [saga][epic] Scale up: Handle large storage efficiently to be able to run current tests efficiently but keep big archives of old results to [saga][epic] Scale up: Efficient handling of large storage to be able to run current tests efficiently but keep big archives of old results

#20 - 2021-05-07 15:35 - okurz

- Copied to coordination #92323: [saga][epic] Scale up: Fine-grained control over use and removal of results, assets, test data added

#21 - 2021-05-07 15:40 - okurz

- Description updated

split out some "future" ideas into the future saga #92323

#22 - 2021-12-23 10:08 - okurz

- Status changed from Blocked to Resolved

With the archiving feature and multiple others completed and enabled archiving on both o3+osd we are much more flexible and should be good for the future

#23 - 2022-05-10 09:40 - okurz

- Copied to coordination #110833: [saga][epic] Scale up: openQA can handle a schedule of 100k jobs with 1k worker instances added