Project Design Document

Table of Contents

I. KVM performance testing

Host Hypervisor performance testing is depend on qa_testset_automation. So we need a framework to do performance testing for Guest of KVM.

II. Requirement:

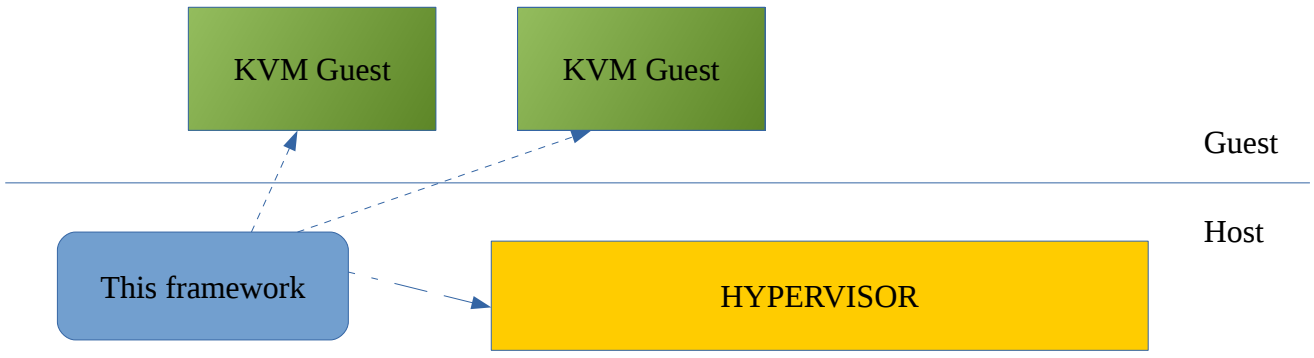
1. The tools should use qa_testset_automation as testcase runner.
2. This framework should be running in Host hypervisor.
  a).Control KVM Guest
  b).Control and configure Host OS
  c).support reboot and continue testing[1]

3. This framework should use libvirt/API as management interface.

4. This framework must be a full automation implement.

III. Detail:

KVM performance testing has more level than Hypervisor performance testing.
Due to testing need compare result between KVM Guest and KVM host. So we need collect data from different level.



Environment Preparation

Host/Guest environment configuration setup by this framework.

Installation (Guest)

autoYaST

Testing Deployment(Guest)

Guestfish

Testing Execution

IO testing

IO-target

Network testing

Multi-machines

IV. Framework High Level Design

USER INTERFACE

textmode is primarily for this framework.
One Shell script and some configuration files forms a complete framework as a prototype.

HOW TO CONTROL

Using some of environment variables to control framework is easy for shell session. And It should be more easy integrate into openQA.

HOW TO MANAGEMENT GUEST

libvirt as primarily interface to manage guest. Qemu as secondary, some of parameter might be unsupported in libvirt command line.

V. WORKFLOW OF PROTOTYPE FRAMEWORK

Explain the workflow of prototype framework. For now it has five steps:

1. check_files
2. prepare
3. start
4. do_test
5. wait_for_jobs_done

OVERVIEW

CHECK_FILES

The testing need some basic configuration files for correctly work. This step just check the files whether is exist.

PREPARE

The duty of this stage depends on the scenarios. It should include installation of dependency packages. Clean the dirty data from target directory. Setup a new environment of disk partition for testing, especially in IO-test. Configure SSH so that it could work in Non-interactive mode.

START

For now this function is core area, it helps to setup filesystem and mount target partition that be used as storing QCOW2 files, and then it create guest via a lots of configuration variables.

In prototype framework, there is a performance optimization for create guest. In the first time of create Guest, framework will have to start from scratch. But when it detect a same QCOW2 that is what it wants, it will create a snapshot that used ‘original’ QCOW2 as backing file, and import this snapshot into a Guest for testing.

After Installation of Guest OS, framework will copy some configuration files into Guest. These files will be used later.

DO_TEST

The testing inside Guest initiate from Host.

Prototype framework access Guest via SSH and virt Console.

First, it get IP of Guest via ‘virtsh console’, the sequence access steps via SSH session.

SSH session will finish the following steps:

- setup testing environment
- Update hostname of Guest
- Re-parted partition.
- Prepare list of testing run
- Prepare config of testing run
- Boot up qa_test_automation framework to finish performance testing

WAIT_FOR_JOBS_DONE

qa_testset_automation is fully automation framework for running a defined list of testcase. It is running in background so it wouldn’t hang-up console.

This stage will hang-up console until the testing is done in Guest.

When some one interrupt this stage that doesn’t impact whole testing in Guest.

VI. Document for Prototype

Virtualization testing coverage multi level, so the most important thing is how to make the framework understand multi-level status so that framework could do right thing according to environment.

Environment Variables

Testing configuration variables

This kind of variables is working for the whole testing plan. For example, the name of testing, the plan id of testing etc.

```
  test_name="io-test"
  testing_run_id="19870828"
```

Host configuration variables

Some variables for setup host environment.

```
  imagepool_path=/kvm
  mempool_path=${imagepool_path}/tmpfs
  testpool_path=${imagepool_path}/${test_name}
  test_fs=xfs
  host_kernel_command_line=""
```

Guest configuration variables

Some of variables for setup a guest.

```
  vm_hostname=vm-io-test-on-`hostname`-nospec-all
  vm_domain_name=${vm_hostname}
  vm_ram_size=8192
  vm_cpu=host-passthrough
  vm_cpus=8
  vm_cpuset=
  vm_per_cpu_thread=$(lscpu | grep "Thread" | awk '{print $4}')
  vm_cpu_cores=$((${vm_cpus} / ${vm_per_cpu_thread}))
  vm_cpu_socket=1
  vm_serial_log=${testpool_path}/${vm_domain_name}.serial.log
```

Guest installation variables

```
  install_media_url="http://mirror.suse.asia/dist/install/SLP/SLE-15-SP1-Installer-Beta2/x86_64/DVD1/"
  sha1no=`echo ${install_media_url} | sha1sum | awk '{print $1}'`
  vm_install_kernel_parameter="console=ttyS0,115200n8 install=${install_media_url}
autoyast=http://dashboard.qa2.suse.asia/index2/jnwang/sles-15-sp1-kvm-guest-         autoyast.xml"
```

Environment configuration files

```
list file
config file
parted-vm.sh
guestfish.sh
install_automation_sles15_sp1_vm.sh
config.ssh
get-ip-from-alive-guest.expect
```

VII. User manual

In above chapter, framework has come config files, they are need you to modify for adapting your target.

Testing list file

This file is used as a runlist store which testcase will be run by qa_testset_automation in Guest. In prototype framework it is named ‘io-test-list’.

```
#!/bin/bash
SQ_ABUILD_PARTITION=/dev/vdb1
function add_kernel_command() {
set -x
        sed -ie "/^GRUB_CMDLINE_LINUX=/s/\".*\"/\"\"/g" /etc/default/grub
        sed -ie "/^GRUB_CMDLINE_LINUX=/s/\"$/ $*\"/g" /etc/default/grub
        sed -ie "/^_QASET_KERNEL_TAG/d" /root/qaset/config
        kernel_tag=$(echo $(for var in "$@"; do echo $var; done | sort -rn) | tr " " "_")
        echo "_QASET_KERNEL_TAG=\"-${kernel_tag}\"" >> /root/qaset/config
        grub2-mkconfig -o /boot/grub2/grub.cfg
        sync
        echo "$(date '+%Y-%m-%d-%H-%M-%S')" > ${SQ_TEST_CONTROL_FILE_SYSTEM_DIRTY}
}

#    workload_database_postgre_smallMem_pgbench_read_xfs
#    workload_database_postgre_smallMem_pgbench_read_xfs
#    workload_database_postgre_smallMem_pgbench_read_xfs
#    workload_http_apache2_siege
SQ_TEST_RUN_LIST=(
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_http_apache2_siege
    _func:add_kernel_command:nopti
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_database_postgre_smallMem_pgbench_read_xfs
    workload_http_apache2_siege
)
```

There is a new feature in qa_testset_automation.

```
_func:<function-name>:<function-parameter1>[:<function-parameter2>]
```

This feature allows a customize routine be called in run list. This function could help you finish environment setup and destroy.

Testing config file

This file is used as a configuration file store some variables that be used by qa_testset_automation.

In prototype framework, it would include these lines:

```
   _QASET_RUNID=${testing_run_id}
   _QASET_HYPERVISOR_KERNEL=$(uname -r)${host_kernel_command_line}
```

```
_QASET_HYPERVISOR_TYPE=kvm
_QASET_HYPERVISOR_RELEASE=$(/usr/share/qa/tools/product.pl -p)
_QASET_HYPERVISOR_BUILD=$(/usr/share/qa/tools/product.pl -r)
```

Testsuite deploy script

This script will be deployed into Guest via Guestfish script. It could help to install require packages.

Guest parted file

This file works for Guest's partition jobs. Usually, don't need modify it. It will parted *vdb* into one partition.

Guestfish script

Guest fish tool could copy files into Guest's disk without Guest boot-up. It is very powerful for testing prepare.