Generic vm interface

The virtual machine "device" also accepts the ioctls KVM_SET_DEVICE_ATTR, KVM_GET_DEVICE_ATTR, and KVM_HAS_DEVICE_ATTR. The interface uses the same struct kvm_device_attr as other devices, but targets VM-wide settings and controls.

The groups and attributes per virtual machine, if any, are architecture specific.

1. GROUP: KVM_S390_VM_MEM_CTRL

Architectures: s390

1.1. ATTRIBUTE: KVM_S390_VM_MEM_ENABLE_CMMA

Parameters: none
Returns: -EBUSY if a vcpu is already defined, otherwise 0

Enables Collaborative Memory Management Assist (CMMA) for the virtual machine.

1.2. ATTRIBUTE: KVM_S390_VM_MEM_CLR_CMMA

Parameters: none
Returns: -EINVAL if CMMA was not enabled; 0 otherwise

Clear the CMMA status for all guest pages, so any pages the guest marked as unused are again used any may not be reclaimed by the host.

1.3. ATTRIBUTE KVM_S390_VM_MEM_LIMIT_SIZE

Parameters: in attr->addr the address for the new limit of guest memory
Returns: -EFAULT if the given address is not accessible; -EINVAL if the virtual machine is of type UCONTROL; -E2BIG if the given guest memory is to big for that machine; -EBUSY if a vcpu is already defined; -ENOMEM if not enough memory is available for a new shadow guest mapping; 0 otherwise.

Allows userspace to query the actual limit and set a new limit for the maximum guest memory size. The limit will be rounded up to 2048 MB, 4096 GB, 8192 TB respectively, as this limit is governed by the number of page table levels. In the case that there is no limit we will set the limit to KVM_S390_NO_MEM_LIMIT (U64_MAX).

2. GROUP: KVM_S390_VM_CPU_MODEL

Architectures: s390

2.1. ATTRIBUTE: KVM_S390_VM_CPU_MACHINE (r/o)

Allows user space to retrieve machine and kvm specific cpu related information:

```
struct kvm_s390_vm_cpu_machine {
      __u64 cpuid;           # CPUID of host
      __u32 ibc;             # IBC level range offered by host
      __u8  pad[4];
      __u64 fac_mask[256];   # set of cpu facilities enabled by KVM
      __u64 fac_list[256];   # set of cpu facilities offered by host
}
```

Parameters: address of buffer to store the machine related cpu data of type struct kvm_s390_vm_cpu_machine*
Returns: -EFAULT if the given address is not accessible from kernel space; -ENOMEM if not enough memory is available to process the ioctl; 0 in case of success.

2.2. ATTRIBUTE: KVM_S390_VM_CPU_PROCESSOR (r/w)

Allows user space to retrieve or request to change cpu related information for a vcpu:

```
struct kvm_s390_vm_cpu_processor {
      __u64 cpuid;           # CPUID currently (to be) used by this vcpu
      __u16 ibc;             # IBC level currently (to be) used by this vcpu
      __u8  pad[6];
      __u64 fac_list[256];   # set of cpu facilities currently (to be) used
                             # by this vcpu
}
```

KVM does not enforce or limit the cpu model data in any form. Take the information retrieved by means of

KVM_S390_VM_CPU_MACHINE as hint for reasonable configuration setups. Instruction interceptions triggered by additionally set facility bits that are not handled by KVM need to by imlemented in the VM driver code.

Parameters: address of buffer to store/set the processor related cpu data of type struct kvm_s390_vm_cpu_processor*.
Returns: -EBUSY in case 1 or more vcpus are already activated (only in write case); -EFAULT if the given address is not accessible from kernel space; -ENOMEM if not enough memory is available to process the ioctl; 0 in case of success.

2.3. ATTRIBUTE: KVM_S390_VM_CPU_MACHINE_FEAT (r/o)

Allows user space to retrieve available cpu features. A feature is available if provided by the hardware and supported by kvm. In theory, cpu features could even be completely emulated by kvm.

```
struct kvm_s390_vm_cpu_feat {
      __u64 feat[16]; # Bitmap (1 = feature available), MSB 0 bit numbering
};
```

Parameters: address of a buffer to load the feature list from.
Returns: -EFAULT if the given address is not accessible from kernel space; 0 in case of success.

2.4. ATTRIBUTE: KVM_S390_VM_CPU_PROCESSOR_FEAT (r/w)

Allows user space to retrieve or change enabled cpu features for all VCPUs of a VM. Features that are not available cannot be enabled.

See :ref:`KVM_S390_VM_CPU_MACHINE_FEAT` for a description of the parameter struct.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\virt\kvm\devices\[linux-master][Documentation][virt][kvm][devices]vm.rst, line 129); *backlink***
>
> Unknown interpreted text role "ref".

Parameters: address of a buffer to store/load the feature list from.
Returns: -EFAULT if the given address is not accessible from kernel space; -EINVAL if a cpu feature that is not available is to be enabled; -EBUSY if at least one VCPU has already been defined; 0 in case of success.

2.5. ATTRIBUTE: KVM_S390_VM_CPU_MACHINE_SUBFUNC (r/o)

Allows user space to retrieve available cpu subfunctions without any filtering done by a set IBC. These subfunctions are indicated to the guest VCPU via query or "test bit" subfunctions and used e.g. by cpacf functions, plo and ptff.

A subfunction block is only valid if KVM_S390_VM_CPU_MACHINE contains the STFL(E) bit introducing the affected instruction. If the affected instruction indicates subfunctions via a "query subfunction", the response block is contained in the returned struct. If the affected instruction indicates subfunctions via a "test bit" mechanism, the subfunction codes are contained in the returned struct in MSB 0 bit numbering.

```
struct kvm_s390_vm_cpu_subfunc {
     u8 plo[32];           # always valid (ESA/390 feature)
     u8 ptff[16];          # valid with TOD-clock steering
     u8 kmac[16];          # valid with Message-Security-Assist
     u8 kmc[16];           # valid with Message-Security-Assist
     u8 km[16];            # valid with Message-Security-Assist
     u8 kimd[16];          # valid with Message-Security-Assist
     u8 klmd[16];          # valid with Message-Security-Assist
     u8 pckmo[16];         # valid with Message-Security-Assist-Extension 3
     u8 kmctr[16];         # valid with Message-Security-Assist-Extension 4
     u8 kmf[16];           # valid with Message-Security-Assist-Extension 4
     u8 kmo[16];           # valid with Message-Security-Assist-Extension 4
     u8 pcc[16];           # valid with Message-Security-Assist-Extension 4
     u8 ppno[16];          # valid with Message-Security-Assist-Extension 5
     u8 kma[16];           # valid with Message-Security-Assist-Extension 8
     u8 kdsa[16];          # valid with Message-Security-Assist-Extension 9
     u8 reserved[1792];    # reserved for future instructions
};
```

Parameters: address of a buffer to load the subfunction blocks from.
Returns: -EFAULT if the given address is not accessible from kernel space; 0 in case of success.

2.6. ATTRIBUTE: KVM_S390_VM_CPU_PROCESSOR_SUBFUNC (r/w)

Allows user space to retrieve or change cpu subfunctions to be indicated for all VCPUs of a VM. This attribute will only be available if kernel and hardware support are in place.

The kernel uses the configured subfunction blocks for indication to the guest. A subfunction block will only be used if the associated STFL(E) bit has not been disabled by user space (so the instruction to be queried is actually available for the guest).

As long as no data has been written, a read will fail. The IBC will be used to determine available subfunctions in this case, this will guarantee backward compatibility.

See :ref:`KVM_S390_VM_CPU_MACHINE_SUBFUNC` for a description of the parameter struct.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\virt\kvm\devices\[linux-master][Documentation][virt][kvm][devices]vm.rst`, line 195); *backlink*
>
> Unknown interpreted text role "ref".

Parameters: address of a buffer to store/load the subfunction blocks from.
Returns: -EFAULT if the given address is not accessible from kernel space; -EINVAL when reading, if there was no write yet; -EBUSY if at least one VCPU has already been defined; 0 in case of success.

3. GROUP: KVM_S390_VM_TOD

Architectures: s390

3.1. ATTRIBUTE: KVM_S390_VM_TOD_HIGH

Allows user space to set/get the TOD clock extension (u8) (superseded by KVM_S390_VM_TOD_EXT).

Parameters: address of a buffer in user space to store the data (u8) to
Returns: -EFAULT if the given address is not accessible from kernel space; -EINVAL if setting the TOD clock extension to != 0 is not supported

3.2. ATTRIBUTE: KVM_S390_VM_TOD_LOW

Allows user space to set/get bits 0-63 of the TOD clock register as defined in the POP (u64).

Parameters: address of a buffer in user space to store the data (u64) to
Returns: -EFAULT if the given address is not accessible from kernel space

3.3. ATTRIBUTE: KVM_S390_VM_TOD_EXT

Allows user space to set/get bits 0-63 of the TOD clock register as defined in the POP (u64). If the guest CPU model supports the TOD clock extension (u8), it also allows user space to get/set it. If the guest CPU model does not support it, it is stored as 0 and not allowed to be set to a value != 0.

Parameters: address of a buffer in user space to store the data (kvm_s390_vm_tod_clock) to
Returns: -EFAULT if the given address is not accessible from kernel space; -EINVAL if setting the TOD clock extension to != 0 is not supported

4. GROUP: KVM_S390_VM_CRYPTO

Architectures: s390

4.1. ATTRIBUTE: KVM_S390_VM_CRYPTO_ENABLE_AES_KW (w/o)

Allows user space to enable aes key wrapping, including generating a new wrapping key.

Parameters: none
Returns: 0

4.2. ATTRIBUTE: KVM_S390_VM_CRYPTO_ENABLE_DEA_KW (w/o)

Allows user space to enable dea key wrapping, including generating a new wrapping key.

Parameters: none
Returns: 0

4.3. ATTRIBUTE: KVM_S390_VM_CRYPTO_DISABLE_AES_KW (w/o)

Allows user space to disable aes key wrapping, clearing the wrapping key.

Parameters: none
Returns: 0

4.4. ATTRIBUTE: KVM_S390_VM_CRYPTO_DISABLE_DEA_KW (w/o)

Allows user space to disable dea key wrapping, clearing the wrapping key.

Parameters: none
Returns: 0

5. GROUP: KVM_S390_VM_MIGRATION

Architectures: s390

5.1. ATTRIBUTE: KVM_S390_VM_MIGRATION_STOP (w/o)

Allows userspace to stop migration mode, needed for PGSTE migration. Setting this attribute when migration mode is not active will have no effects.

Parameters: none
Returns: 0

5.2. ATTRIBUTE: KVM_S390_VM_MIGRATION_START (w/o)

Allows userspace to start migration mode, needed for PGSTE migration. Setting this attribute when migration mode is already active will have no effects.

Parameters: none
Returns: -ENOMEM if there is not enough free memory to start migration mode; -EINVAL if the state of the VM is invalid (e.g. no memory defined); 0 in case of success.

5.3. ATTRIBUTE: KVM_S390_VM_MIGRATION_STATUS (r/o)

Allows userspace to query the status of migration mode.

Parameters: address of a buffer in user space to store the data (u64) to; the data itself is either 0 if migration mode is disabled or 1 if it is enabled
Returns: -EFAULT if the given address is not accessible from kernel space; 0 in case of success.