Proportional QEMU memory overhead allocation

Motivation

In addition to the memory allocated for the guest virtual machine, QEMU also requires memory for it's internal needs. RAM is a limited resource, and unlimited consumption of one of the tenants will negatively impact other guests, and the EVE-os itself. To avoid this impact we are limiting total amount of RAM which is allowed to be used by a guest (Guest RAM + QEMU overhead).

The problem is we do not have a reliable way to predict how much memory QEMU will require. If the limit is too low for guest's use case, process will eventually be killed by `out-of-memory-killer` (OOM killer).

To workaround this issue we have found experimentally that 600 MiB is enough for the most use cases. This value is hardcoded, and used for **any** guest virtual machine. While this does solve the issue of random kills of the guest VMs, this approach leads to overallocation for very small VMs which will going to use only a fraction of these 600 MiB. As the result customer's hardware is used inefficiently, and customer can launch fewer guest VMs.

Proposal Description

Proper solution would be to conduct a research on where does this memory go, and invest into efforts of reducing the consumption and/or making this consumption more predictable. However this takes time, and we need to address customer pain sooner.

This document proposes merely an adjustment to the existing workaround. Instead of the fixed overhead, we shall allocate overhead proportionally to the amount of RAM user requested for the guest VM. Unfortunately RAM is only an indirect indicator of how much memory QEMU would require, but it could be the limiting factor for the guest input-output activity, which as we suspect correlates with the overhead used by QEMU.

The implementation of the proposal can be found in [1], which is rather simple, and overhead is calculated as 30% of guest's RAM.

Another approach is proposed in [2] by Roman Shaposhnik: reserve a portion of all host's memory (e.g. 25%), and spread this reserve among existing guests proportionally to their memory. Whenever guest start or stop is requested, EVE shall recalculated limits for all existing VM's. While this approach does use memory more efficiently, it might be impossible to reclaim memory from an existing guest. For example the following situation would be a problem:

1. Allocate Guest A, served by process `qemu_a` `with` overhead limit 600MiB
2. Guest A experiences significant load, and `qemu_a` allocates 500 MiB of anonymous (non-reclaimable) memory for it's purposes
3. Request to allocate Guest B arrives, recalculated limit for `qemu_a` is 400MiB

In the described situation `cgroup (Control Groups [3])` interface would not allow to reduce the limit. So we left with 3 options:

1. Allocate less memory for the Guest B and hope it would not reach the limit
2. Report failure to the control plain (zed cloud/admam), and do not allocate Guest B
3. Kill and recreate the Guest A with lower limit (hard reset from the guest's point of view)

Arguably, each of these options would be rather unexpected behaviour.

References

1. Proposal implementation Pull Request
2. @Roman Shaposhnik's proposal
3. Linux Kernel cgroup documentation