GSoC 2017 Project Proposal

Xen: share multiple ranges of memory area among VMs from the config file

1 Personal Information

1.1) Full Name: [MY NAME]

1.2) Email: [MY EMAIL]

1.3) Other contact information:

Github: https://github.com/blackskygg

Secondary Email: [MY SECONDARY EMAIL]

1.4) Other applications: No. I haven't applied for other GSoC projects other than this.

1.5) Previous experience: No. This is my first time participating GSoC.

1.6) Time commitment: I'm going to take a summer class at National University of Singapore from July 22 to August 22, so I can't guarantee a full-time participation. But I can guarantee that overall I can spend 30+ hours per week on this GSoC project.

1.7) Other programs: No. I haven't applied for other intern programs.

2 Preparation done so far

2.1) Hardware: After talking to the mentors, I'm sure that I have met the hardware requirements for this task (See: https://lists.xen.org/archives/html/xen-devel/2017-03/msg03037.html).

2.2) Ability to Build and Test: I've setup the dev environment and have built and tested the Xen hypervisor. (See: https://lists.xen.org/archives/html/xen-devel/2017-03/msg00626.html).

2.3) Make a Small Contribution: According to the mentors' suggestions, and thanks to their patience and guidance, I've taken XEN-41 (https://xenproject.atlassian.net/browse/XEN-41) as my initial task and get my first contribution merged to the staging branch (See: https://lists.xen.org/archives/html/xen-devel/2017-03/msg02708.html).

2.4) Other: I've roughly explored the code base of the Xen Hypervisor. And by making my initial small contribution, I've observed how others communicate and cooperate in the mailing list and what the workflow looks like. As for this specific task, I've read the related code, and have discussed the implementation plan with the mentors.

3 Proposed Project

3.1) Project Category: Xen hypervisor.

3.2) Proposal Title: share multiple ranges of memory area among VMs from the config file.

3.3) Link to original proposal: This proposal is extended from this proposal: https://wiki.xenproject.org/wiki/Outreach_Program_Projects#Share_a_page_in_memory_from_the_VM_config_file

4 Implementation Plan

4.1) Detailed Problem Description: Virtual machines use grant table hypercalls to setup a share page for inter-VMs communications. These hypercalls are used by all PV protocols today. However, very simple guests, such as baremetal applications, might not have the infrastructure to handle the grant table. This project is about setting up several shared memory areas for inter-VMs communications directly from the VM config file. So that the guest kernel doesn't have to have grant table support to be able to communicate with other guests.

4.2) Implementation Plan:

1. Introduce a new VM config option in xl:
   The shared areas is shared among several VMs, and every shared physical area is assigned to a set of VMs. Therefore, a “token” or “identifier” should be used here to uniquely identify a backing memory area.

   I would suggest using an unsigned integer to serve as the identifier. For example:
   In xl config file of vm1:
   static_shared_mem = [“addr_range1= ID1”, “addr_range2 = ID2”]
   In xl config file of vm2:
   static_shared_mem = [“addr_range3 = ID1”]
   In xl config file of vm3:
   static_shared_mem = [“addr_range4 = ID2”]

   In the example above. A memory area A1 will be shared between vm1 and vm2 -- vm1 can access this area using addr_range1 and vm2 using addr_range3. Likewise, a memory area A2 will be shared between vm1 and vm3 -- vm1 can access A2 using addr_range2 and vm3 using addr_range4.

   The shared memory area denoted by an identifier IDx will be allocated when it first appear, the memory pages will be taken from the first VM whose static_shared_mem list contains IDx. Take the above config files for example, if we instantiate vm1, vm2 and vm3, one after another, the memory areas denoted by ID1 and ID2 will both be allocated in and taken from vm1.
2. Store the mem-sharing information in xenstore:

The information should include the length and owner of the area. And it should also include information about where the backing memory areas are mapped in every VM that are using it. This information should be known to the xl command and all domains, so we utilize xenstore to keep this part of information. A current plan is to place the information under /local/shared_mem/ID. Still take the above config files for example:
If we instantiate vm1, vm2 and vm3, one after another, “xenstore ls -f” should output something like this:

After VM1 was instantiated, the output of “xenstore ls -f” will be something like this:

/local/shared_mem/***ID1***/owner = ***dom_id_of_vm1***
/local/shared_mem/***ID1***/size = ***sizeof_addr_range1***
/local/shared_mem/***ID1***/mappings/***dom_id_of_vm1*** = ***addr_range1***

/local/shared_mem/***ID2***/owner = ***dom_id_of_vm1***
/local/shared_mem/***ID2***/size = ***sizeof_addr_range1***
/local/shared_mem/***ID2***/mappings/***dom_id_of_vm1*** = ***addr_range2***

After VM2 was instantiated, the following **new** lines will be appear:

/local/shared_mem/***ID1***/mappings/***dom_id_of_vm2*** = ***addr_range3***

After VM2 was instantiated, the following **new** lines will appear:

/local/shared_mem/***ID2***/mappings/***dom_id_of_vm2*** = ***addr_range4***

When we encounter an id ***IDx*** during “xl create”:

- If it’s not under /local/shared_mem, create the corresponding entries (owner, size and mappings) in xenstore, and allocate the memory from the newly created domain.
- If it’s found under /local/shared_mem, map the pages to the newly created doamin, and add the current domain to /local/shared_mem/IDx/mappings.

3. Handle the newly added config option in tools/xl/xl_parse.c, and utilize toos/libxl to do the actual memory mapping
4. Add code to handle various errors: Invalid address, mismatched length of memory area etc.

4.3) Expected Outcomes/Goals: A new VM config option in xl will be introduced, allowing users to setup several shared memory areas for inter-VMs communications. This should work on both x86 and ARM.

4.4) Timeline:

- May 30 - June 10: Discuss with the mentors a refined designed of the newly added option and parameters and some high level implementation decisions.
- June 10 - June 20: Option and parameter parsing. Parsing stage error handling.
- June 20 - June 15: Real page allocation and sharing.
- July 16 - July 28: Runtime error handling. All features done.
- July 29 - August 29: Discuss and iterate. Comprehensive testing (on both x86 and ARM).

4.5) Maintenance: I would like to provide future maintenance (if required) and would like to become an active member of the Xen-devel community.

5 Experience and help needed

5.1) Describe experience:

- Have been working with the Xen hypervisor in several security related projects in the CGCL Lab of Huazhong University of Science and Technology.
- Have done a small code contribution to the Xen hypervisor prior to my application.

5.2) Other experience:

- Solid Linux and C programming skills:
    - Placed 1st in 'The Fourth Student RDMA Programming Competition' held by the HPC Advisory Council
- Limited Linux device driver development experience.
- Craze about hacking:
    - Placed 2nd in 'HACKxFDU 2016 Hackthon' held by Fudan University Committee of the Communist Youth League

5.3) Learning and support:

I'm still very new to the Xen hypervisor project, so I'm still not very familiar with the whole code base and the "Xen coding conventions". To be more specific, although getting the work done is not very difficult, but getting it done elegantly in the Xen way is hard. So I need guidance to get familiar with the structure of the code base and some Xen-specific coding skills.