Study of the Scalability of Virtualization Platform

Oct, 2016

Wei Yang <richard.weiyang@huawei.com>

www.huawei.com

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Agenda

- **Scalability & Evaluation**
- **Issues & Proposals on Xen**
- **Summary**

What’s Scalability

- **Scalability** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth [wikipedia]

Source
https://en.wikipedia.org/wiki/Scalability

What’s Scalability -- VMs

- We care about below dimensions of scalability on a single server
  - Horizontal scaling: more VMs , more users
  - Vertical scaling: larger VM

How to evaluate - Test Matrix

- **Test Matrix on horizontal scale**

#VM	1	2	4	…	N
VCPU binding					
None binding					

Note: N stands for total vcpus == pcpus

How to evaluate - Benchmarks

- **Individual**
  - **CPU/Mem intensive – SPEC CPU**
  - **Storage Intensive – FIO**
  - **Network Intensive -- Iperf**
- **All-around**
  - **SPEC Virt**
  - **TPCC**
- **Your Core Scenario**

Agenda

- **Scalability & Evaluation**
- **Bottleneck & Proposals on Xen**
- **Summary**

Bottleneck 1: call lock

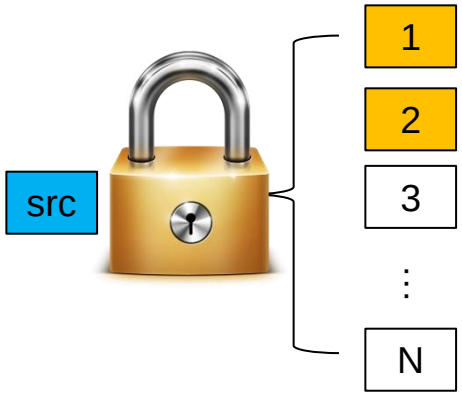
- **Call lock is global lock, used in on_selected_cpus to protect IPIs on targeted CPUs**

- **Bottleneck case**
  - **Frequent EPT entry changes invalidate EPT on related PCPUs**
    - **ept_sync_domain -> on_selected_cpus**
    - **heavy contention of call lock**

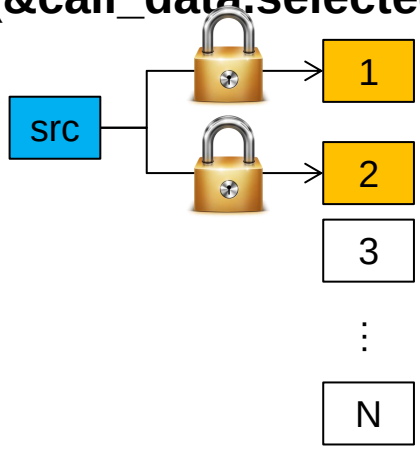
Proposal: scalable lock and finer-grain lock

- **Replace ticket lock with MCS lock for call lock**
- **Change global lock to per-cpu lock**
  - **Hold a per-cpu lock of each target CPU, instead of a global lock (call lock)**
  - **Then do IPIs:**

    smp_send_call_function_mask(&call_data.selected);



It locks all even though the IPI target CPUs are only CPU 1 and 2



It only locks IPI target CPUs (1 and 2), other CPUs are still available for IPI

Evaluation Environment and Method

- **Hardware**
  - **8 sockets, 96 cores, 192 threads**
  - **2TB RAM**
- **Host**
  - **Xen 4.7.0**
  - **Dom0: SLES 11SP3**
- **VM**
  - **Win 7 64bits**
  - **40GB RAM**
  - **64 vCPUs**
- **Test case**
  - Launch 8 VMs with POD in parallel, it will result in lots of EPT changes due to Windows OS will zero pages during booting, then scan and recycle zero pages when VM memory usage beyond reserved threshold.

Evaluation Results

- **With the optimization, the number of lock wait and each wait consuming are both reduced significantly**
- **Concurrent bootup time of VMs reduced significantly**

Bottleneck 2: vCPU load balance in credit scheduler

- **vCPU load balance**
  - **If the next highest priority local runable vCPU has already eaten through its credits, look on other PCPUs to see if we have more urgent work**
    - **Select non-idling CPUs**
    - **Try to acquire the schedule lock of a non-idling CPU**
    - **Then try to steal a task from this non-idling CPU**
    - **If stealing not succeed, go to next non-idling CPU**

- **The bottleneck analysis**
  - **In many core case, there will be lots of non-idling CPUs, it's a big waste if it often acquired the schedule lock but failed to steal a task.**

Proposal: Add a check before acquiring schedule lock

- **Add a check for each non-idling CPU before acquiring schedule lock**
  - **Add a bitmap of each PCPU, record what PCPUs its runq vCPUs can run (vCPU may be pinned on some PCPUs)**
  - **If this CPU (which wants to steal task from other CPUs) is not in the bitmap of a non-idling CPU, that means it cannot steal a task from this non-idling CPU. Save the cost of acquiring schedule lock.**

Evaluation Environment

- **Hardware**
  - **8 socket, 96 cores, 192 threads**
  - **2TB RAM**
- **Host**
  - **Xen 4.7.0**
  - **Dom0: SLES 11SP3**
- **VM**
  - **Win 7 64bits**
  - **40GB RAM**
  - **24 vCPUs**
- **Test case**
  - **1:1 pin vCPU to PCPU, and run NotMyFault in VM**

Evaluation Results: 1:1 pin VCPU to PCPU

- **With the optimization, the number of lock wait and each wait consuming are both reduced significantly**
- **While there is some regression when VCPU is not pinned to PCPU**

Summary

- The evaluation framework of the virtualization platform
- Some scalability bottlenecks and proposals
- Future work
  - Virtualization scalability benchmark
    - Scenarios
    - Measurement
  - Huge VM (vcpu > 128, memory > 1TB) support

Thank you

www.huawei.com

Backup Slide

Thank you

www.huawei.com

ticket spinlock is non-scalable

- **Xen uses ticket spinlock by default**
- **Ticket spinlock is fair, FIFO**
- **But, it's non-scalable for many core**
  - Spin on global shared variable
  - Expensive cache entries invalidation

Proposal: scalable lock

- **MCS lock is scalable**
  - Spin on local variable
  - Generate a constant number of cache misses per acquisition, avoid the performance collapse with many cores.

Current Status: Xen Max Limits

Xen 4.7

- **Host Limits (x86)**
  - Up to 4095 physical CPUs
  - Up to 16TB physical memory

- **HVM Guest Limits (x86)**
  - Up to 512 virtual CPUs
  - Up to 1TB virtual memory

Current Status: Scale to Thousands of VMs

- **3000 VMs were experimented successfully**
  - Improving Scalability of Xen: the 3,000 domains experiment (https://events.linuxfoundation.org/images/stories/slides/lfcs2013_liu.pdf)

Scalability was improved a lot in upstream

- **Grant table**
  - Split grant table lock into maptrack lock and grant table lock
  - per-vCPU maptrack free lists
  - Read-write lock
  - Per-active entry lock
- **Persistent grants for virtual block scalability**
  - Avoid grant operations and TLB flushes
- **Event channel**
  - Extend event channel limit: up to 131072
  - per-event channel lock for sending events
- **Scheduler**
  - node-affinity for vCPU load balance
- **P2M**
  - Use unlocked p2m lookups in hvmemul_rep_movs
  - defer the invalidation until the p2m lock is released