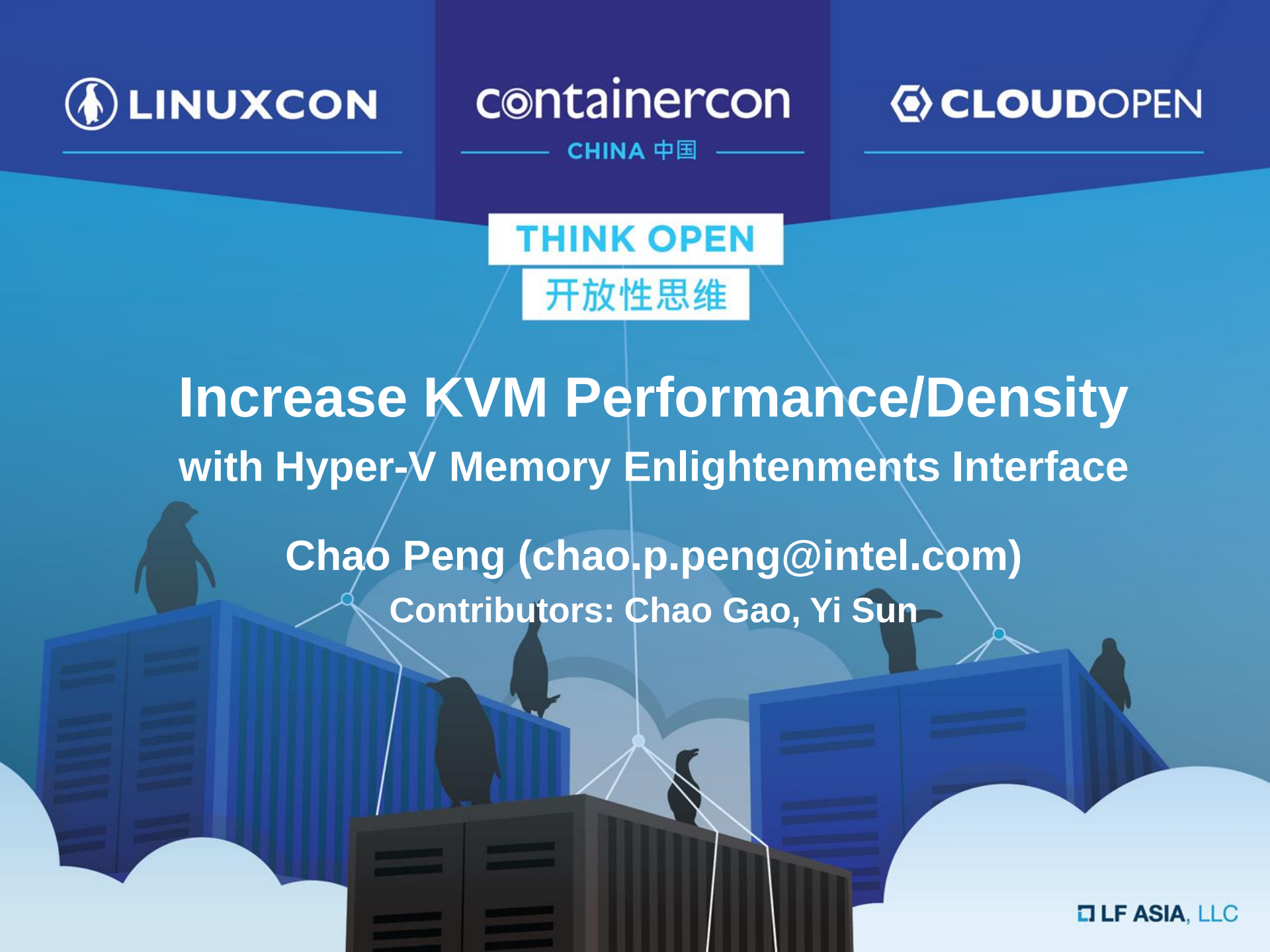


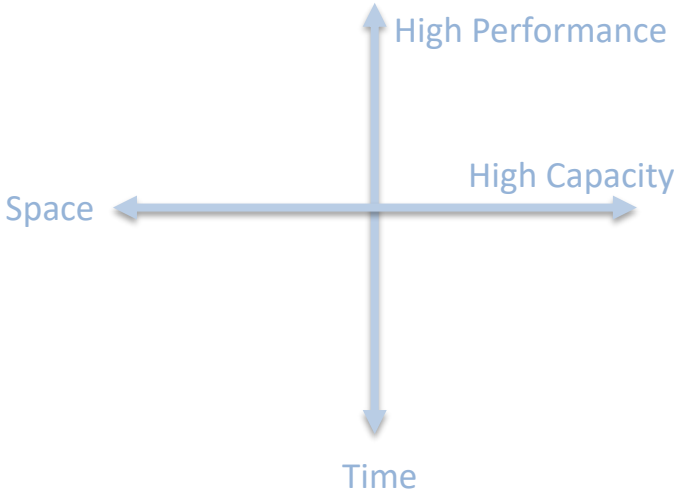
Increase KVM Performance/Density

with Hyper-V Memory Enlightenments Interface

Chao Peng (chao.p.peng@intel.com)

Contributors: Chao Gao, Yi Sun

Expectations in Memory Virtualization



People want fast and large size memory

- Performance
  - People want near native memory performance in guest
  - Memory virtualization overhead should be small
- Density
  - Guest memory overcommit
    - Some users want cheap VMs, memory sharing is acceptable
  - Small memory footprint
    - VM based containers require small memory footprint VMs

Available Approaches for KVM/QEMU

- Performance
  - Hugepage(2M/1G) for nesting paging(e.g. Intel® EPT) mapping
    - Reduce pagefaults and TLB misses
  - '-mem-prealloc' guest memory
    - Reduce pagefaults
- Density
  - Host Swapping
    - Linux automatically swaps out guest pages to disk when host memory pressure is high
  - KSM(Kernel Samepage Merging)
    - De-duplicate guest memory pages to save memory
  - VirtIO balloon
    - Guest unused memory can be returned to host and used for other guests

Hyper-V Memory Enlightenments

- Hyper-V enlightenments in general
  - A para-virtualization approach to reduce virtualization overhead
  - Guest is aware of virtualization and guest change is required
  - Was initially designed by Microsoft on Hyper-V + Windows, recently expanded to Linux guest
  - Similar to KVM paravirt_ops in arch/x86/kernel/kvm*.c
- Hyper-V memory enlightenments
  - Memory zeroing
  - Memory access hints
  - Enlightened Page Fault Handler

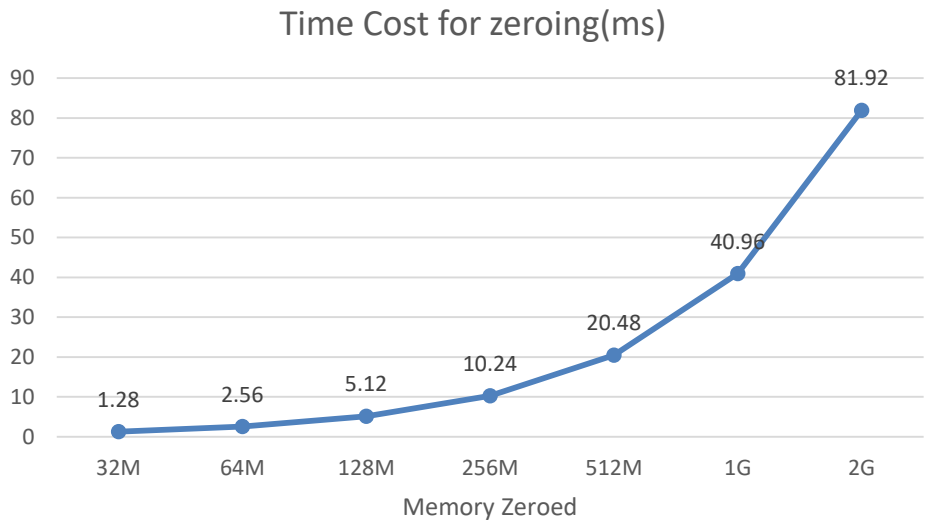
They are used to improve performance/density on Hyper-V + Windows

Hyper-V Enlightenments on KVM

- Motivations
  - Provide better Windows performance in a virtual machine under KVM
  - Can also benefit Linux guest when it's configured to
- KVM is Hyper-V compatible
  - All enabling code live in arch/x86/kvm/hyperv.c
    - Hyper-V hypercall page & assist page
    - Time reference count
    - VAPIC enlightenment
    - Guest Crash enlightenment
  - Patches still WIP in community to date
    - PV TLB flush enlightenment
    - Nested enlightenments: Enlightened VMCS/MSR bitmap enlightenment
  - Memory enlightenments: not enabled yet

Memory Zeroing Enlightenments

- Double-zeroing
  - VMM zeros all memory before giving it to guest
    - Prevents information disclosure
  - Operating systems(guest) zero memory again
    - Because memory content is non-deterministic



NOTE: the data was collected on specific hardware/software, your data may differ depending on configurations.

Memory Zeroing Enlightenments

- Double-zeroing can be avoided
  - Host zeroing is needed anyway
    - Information disclosure is intolerable
  - Guest can skip zeroing for the first time access
    - Boot memory zeroing
    - Hot-add memory zeroing
- Benefits
  - Static mapped memory
    - Save CPU cycles for zeroing(e.g. booting faster)
  - Dynamic mapped memory
    - Save CPU cycles for zeroing(e.g. booting faster)
    - If guest lacks ‘zero page’ and zeroing results real allocation, then more benefits:
      - Reduce pagefaults when zeroing
      - Reduce memory allocated to guest, hence increase density

Memory Zoroing on KVM

- KVM enabling
  - Memory is already zeroed before mapped to guest
  - Expose memory zeroing capability to guest
  - Usually dynamic mapping is used, we can still tell guests we zeroed all the memory, although the zeroing happens at pagefault time
- Windows guest
  - Supported Windows editions benefit from:
    - Boot memory zeroing
    - Hot-add memory zeroing
- Linux guest
  - Boot memory: Certain boot memory zeroing can be avoided. Example: 64M zeroing in swiotlb_init()
  - Hot-add memory: Linux does not zero hot-added memory

Memory Access Hints

- Cold hint
  - Guest OS indicates the set of physical pages which can be unmapped and removed from the guest's working set
  - Host will trim unneeded pages to increase VM density
- Hot hint
  - Allows guest OS to indicate the set of physical pages needed for frequent or upcoming access
  - Host will opportunistically pre-fault these pages such that subsequent access should not fault

Memory access hints on KVM

- VirtIO balloon
  - Designed for VM memory over-committing
    - Same as increasing container density
  - Implemented two fundamental operations
    - Inflate: memory is taken from guest to host
    - Deflate: memory is taken from host to guest
  - Similar to Hyper-V memory access hints in several ways
    - Guest memory can be ‘free-ed’ to host
    - The free-ed memory can be re-allocated
    - Both work on page granularity
  - Restriction
    - A third-part monitoring program is required, to monitor both host/guest memory pressure, then adjust guest memory manually, or automatically
- Free Page Hinting
  - Developed by Nitesh Narayan Lal, still in upstreaming
  - Based on VirtIO balloon, but guest notify host on each arch_free_page()
  - Use MADV_FREE instead of MADV_WONTNEEDED which inflate uses

Memory access hints on KVM

- Cold hint
  - Free page hinting is quite close to Hyper-V cold hint
  - Linux guest can benefit from it once merged
  - Windows guest may be a problem, depending on the availability of ‘free’ hook
  - If free hook does not exist then some Hyper-V wrapper around free page hinting is needed
- Hot hint
  - No existing alternative in KVM
    - Hot hint was designed for performance improvement while deflate of VirtIO balloon more focuses on memory return from host to guest
    - Hot hint can pre-fault any memory while VirtIO deflate can only return memory inflated
  - KVM Enabling
    - KVM implementation is simple, only need to map requested pages
    - Windows guests(supported editions) require no code change at all
    - Linux guest support is something challenge: We need find the relevant code that can benefit from hot hint

Enlightened Page Fault Handler

- Hyper-V EPF(Enlightened Page Fault)
  - Normally, host software handles page fault synchronously by resolving the access fault and resuming the vCPU upon access fault completion
  - EPF allows the guest OS to reschedule threads on a vCPU which caused the page fault
- KVM APF(Asynchronous Page Fault)
  - APF is already enabled in KVM and Linux guest
  - KVM APF is almost identical to Hyper-V EPF
  - It's hard to implement APF in Window guest
    - Page fault handling are core code for the kernel
  - We can expose EPF interface to Windows, by reusing APF implementation in KVM

Summary

- Memory zeroing
  - Fast booting/instantiation
- Memory cold access hint
  - Increase density
- Memory hot access hint
  - Fast booting/instantiation
  - Reduce runtime memory virtualization overhead
- Asynchronous Page Fault
  - Improve runtime performance
- Practice for KVM as a Hyper-V compatible Hypervisor
  - We need clear interface/implementation separation when developing new features

Reference

- Guest Memory Overcommit - Page hinting, resizing & more
  https://www.linux-kvm.org/images/f/ff/2011-forum-memory-overcommit.pdf
- [RFC,QEMU] kvm: Support for guest page hinting
  https://patchwork.kernel.org/patch/10458411/
- [v21,0/5] Virtio-balloon Enhancement
  https://patchwork.ozlabs.org/cover/857395/
- KVM as a Microsoft-compatible hypervisor
  https://www.linux-kvm.org/images/0/0a/2012-forum-kvm_hyperv.pdf
- Hypervisor Top Level Functional Specification(TLFS)
  https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/tlfs


LINUXCON
containercon
CLOUDOPEN
CHINA 中国
THINK OPEN
开放性思维
Thanks!
LF ASIA, LLC