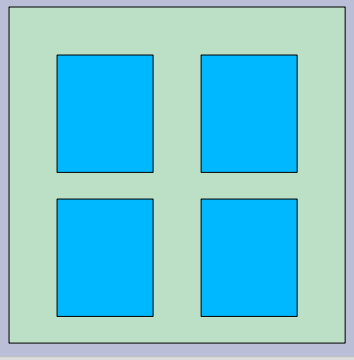
Live und in Farbe

Live Migration

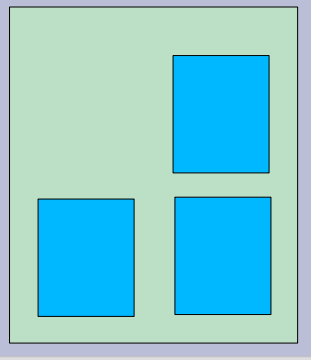
André Przywara
ap@amd64.org
CLT 2010

Agenda

- (Live) Migration explained (Why? Limits!)
- Xen and KVM usage
- Details
  - Memory synchronization
  - QEMU device state transfer
  - Host considerations (CPU features)
  - Cross Vendor Migration
- QEMU block device transfer
- Project Remus (Xen)
- Demo!



Guest Migration



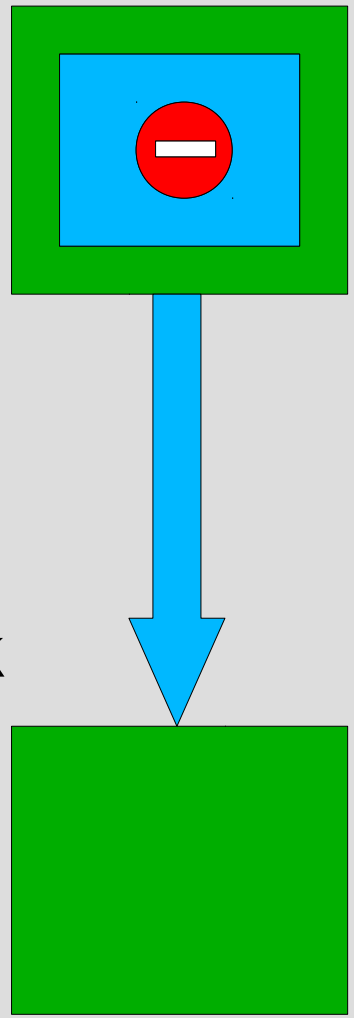
- move a *virtual machine* from one host to another
- offline:
  - power down the guest, copy files, restart
  - comparable to a reboot
- migration:
  - halt the guest, copy state, wake up again
  - minimal downtime
- live migration:
  - copy state in background, switch at one
  - (almost) no downtime at all

Reasons for migration

- Load balancing:
  - freeing loaded hosts in favor of idle ones
- Upgrade / update / planned downtime
  - migrate to a spare machine, rework the host, migrate back to the original one
- Roaming “eternal” desktop – Uptime, uptime!
  - desktop is running on a server, migrated to the respective client workstation
- Replacing older machines
- You name it!

How does it work?

- Host has full control over the guest
- Can read/write/protect memory
- Devices are (usually) also virtualized
- Host controls CPU usage
  - similar to OS vs. application
- Steps:
  - host de-schedules the guest
  - host copies memory content over network
  - host copies device state over network
  - old host signals new host to take over



Limits of migration

- disk images should be accessible
  - through a SAN, NAS, NFS
  - can also be copied / synced (DRBD)
- no downgrade of CPU features
  - maybe start with features disabled?
- No device pass-through
- Network connectivity must prevail
- Resources should match (memory, vCPUs)
- Matching software versions (devices!)

Xen / KVM usage

- Xen: via “xm” tool
  - **`$ xm migrate <domid> <newhost>`**
  - *xend* must be running on both sides
- KVM:
  - on target:

    `$ qemu -incoming tcp:0:<port>`
  - on source: via QEMU monitor

    `(qemu) migrate tcp:<host>:<port>`
  - Need to have the exact same guest parameters on the command line (management app!)

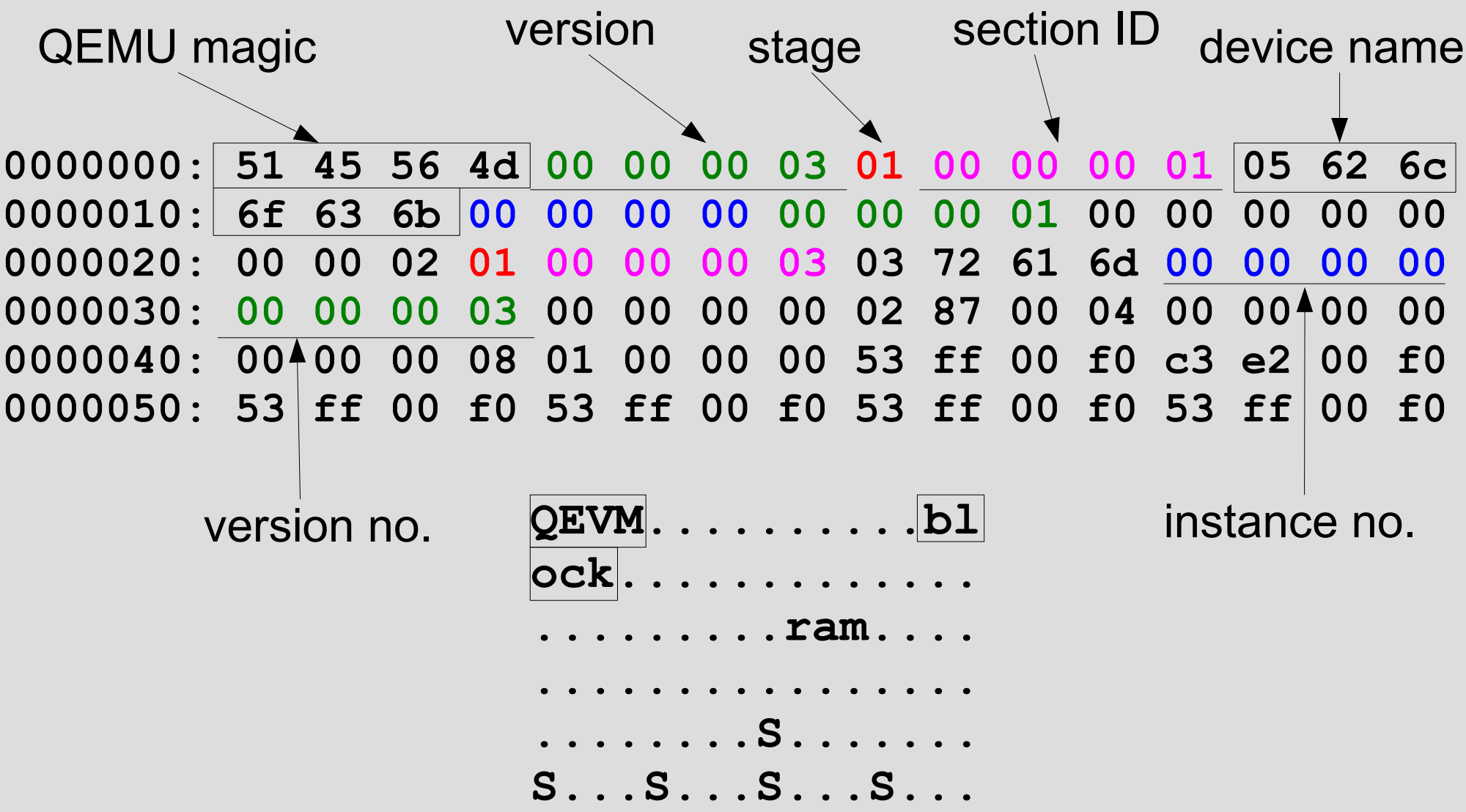
Memory synchronization

- Problem: transferring RAM image takes time
- e.g.: 1GB @ 40 MB/s = 25 sec
  - too long for live migration
- solution:
  - start copying (in background)
  - write protect already copied pages
  - on page fault: allow r/w again, mark page as *dirty*
  - repeat: copying dirty pages until
    - no more left
    - number of tries exhausted: halt guest and copy rest

QEMU device state transfer

- QEMU devices used for Xen and KVM
- each device has a VMStateDescription
  - describes the data that holds the complete state
- variables will be dumped to the stream
- contains version information (backward compatible)
- QEMU will iterate through all devices
  - sends the device name and instance number
  - executes a pre_save callback function
  - dumps the device' state to the stream (TCP)

QEMU device state dump

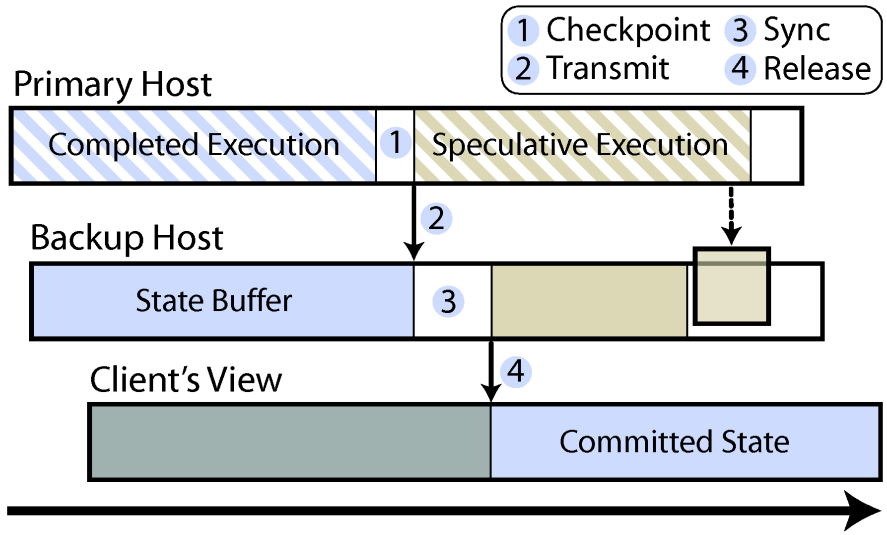


QEMU block device transfer

- Recent QEMUs can transfer the block device
- No need for a shared storage
- **`(qemu) migrate -b tcp:<host>:<port>`**
- Can also migrate overlay only (-i)
- Uses same approach like RAM transfer
- Works like this:
  - Transfer data in chunks of 1 MB
  - Each chunk is preceded by a 64bit address
  - Allows gaps
  - Each chunk has the block device name in it

Project Remus (Xen)

- High availability using migration
- “Continuously” migrating the guest
- Avoids slowdown by snapshotting
- Only commits results when transmitted
- Snapshot frequency about every 25ms
- Running machine in the past
- part of Xen 4.0



Host considerations

- Applications and libraries rely on a consistent set of CPU features (like SSEx)
- CPU instruction set may change at migration
- no downgrade! (loss of a feature)
- upgrade can be hidden (CPUID masking)
- least common denominator in a migration pool dictates the feature set of all guests
- KVM: use -cpu kvm64
- migration pool should be well defined before starting the guest

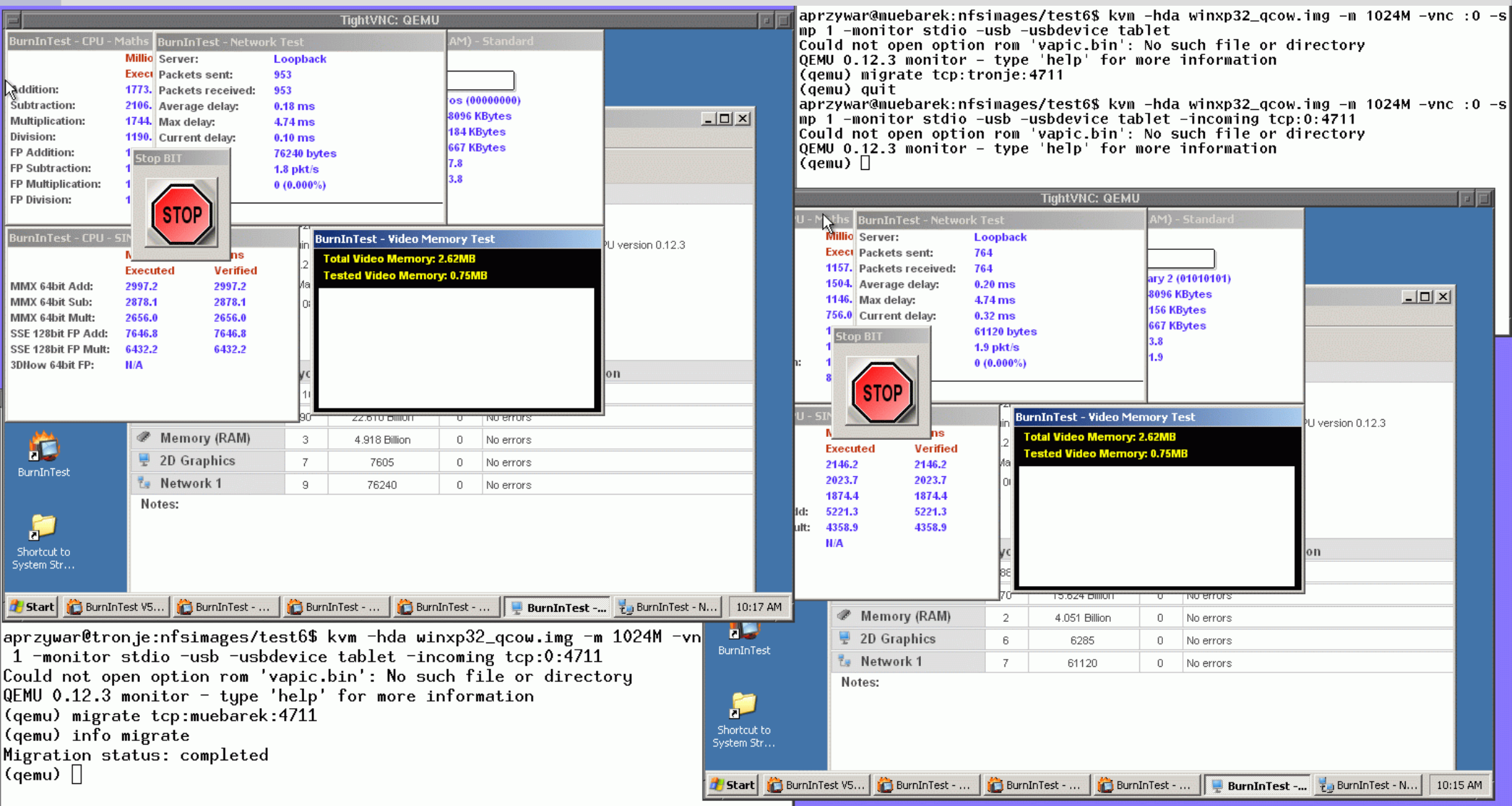
Cross Vendor Migration

- Migrating from an Intel box to an AMD box (and vice versa ;-)
- allows for bigger migration pools
- avoids vendor lock in
- maps mostly to different CPU generations
- but subtle differences:
  - x87 FPU rounding on some instructions (e.g. for fsin, deprecated)
  - sysenter/syscall support in compat mode (emulation upstream)
  - slightly different guest state checks (fixed)
  - Model specific registers (MSRs) (fixed)
- Both Xen and KVM support it now!

Demo! Live! In Color!

- Using KVM (qemu-kvm 0.12.3, kernel 2.6.33)
- Migration between servers, using VNC
- Windows XP 32 guest with running Passmark stress test application
- Between Intel C2Q and AMD Opteron

Demo screenshot



References

- Project Remus: http://dsg.cs.ubc.ca/remus/
- Cross Vendor Migration: http://developer.amd.com/assets/CrossVendorMigration.pdf
- QEMU live migration: http://kvm.et.redhat.com/wiki/images/5/5a/KvmForum2007$Kvm_Live_Migration_Forum_2007.pdf