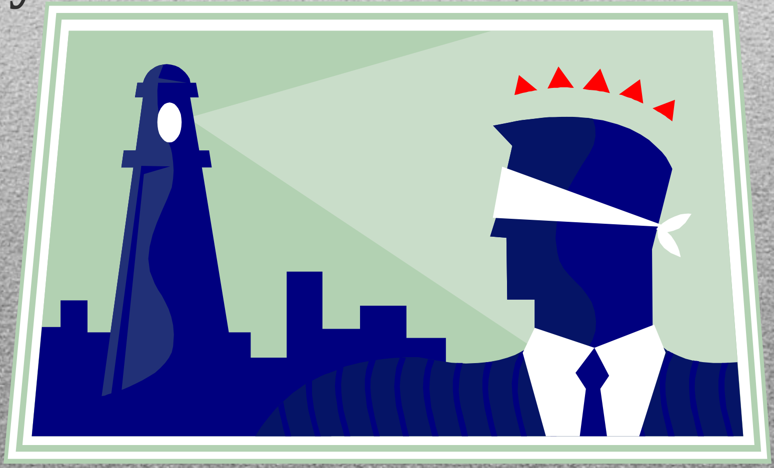
RDMA Migration and RDMA Fault Tolerance for QEMU

Michael R. Hines
IBM
mrhines@us.ibm.com

http://wiki.qemu.org/Features/RDMALiveMigration
http://wiki.qemu.org/Features/MicroCheckpointing

Migration design / policy Problems

- **Admins want to evacuate any size VMs**
  - Tens of GB / Hundreds of GB
  - Arbitrary storage configurations
  - Arbitrary processor configurations
- **Management software is still pretty blind:**
  - Is VM idle?
  - Busy?
  - Full of zeros?
  - Mission Critical?
  - System software wants our cake
  - and eat it too



- **Customers don’t (yet) want to re-architect**
  - Just make my “really big JVM” or my “honking DBMS” run
  - Don’t ask me any questions about the workload
  - But willing to tell you “high-level” workload characteristics
  - Co-workers keep telling me security is important,
  - probably don’t want any “extreme” visibility anyway
- **Customers do *want* high availability**
  - But they don’t really trust us (much)
    - They think we’re really good at running workloads
    - Not so sure we’re good at *moving* workloads
  - Also don’t want to re-architect
  - Don’t want to put all their eggs in one basket
    - Still very willing to spend money on mainframes
- “PaaS” not a panacea, but has the potential to be

High Availability design / policy problems:

State of the art

- **Migration:** 40 Gbps *ethernet* hardware already o
  market
  - Faster ones not far away
  - Can we do TCP @ 40 Gbps?
    - Sure, at 100% CPU – that's not good

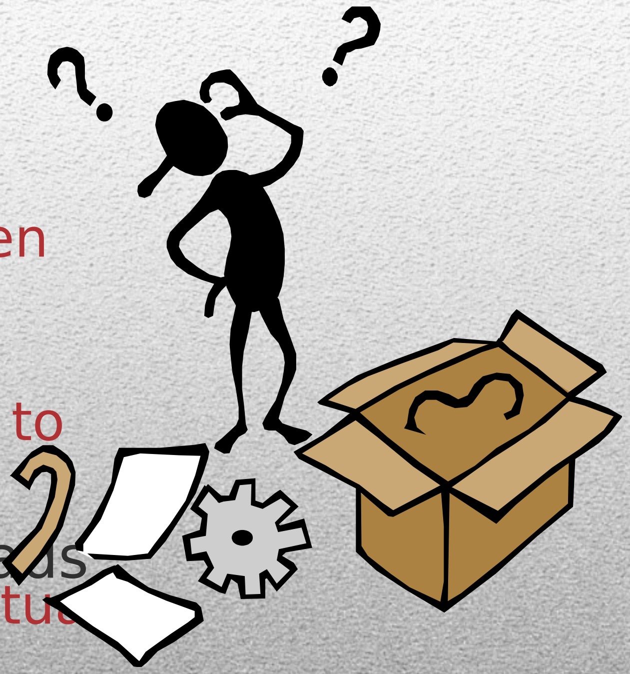
- **Fault Tolerance:**
  - A.K.A: Micro-Checkpointing / Continuous Replication
  - Multiple hypervisors are introducing it:
    - Remus on Xen
    - VMWare lockstep
    - Marathon / Stratus EverRun
  - Where is KVM in all this "HA" activity?

- Smaller active set of physical servers
- evacuate low utilization servers

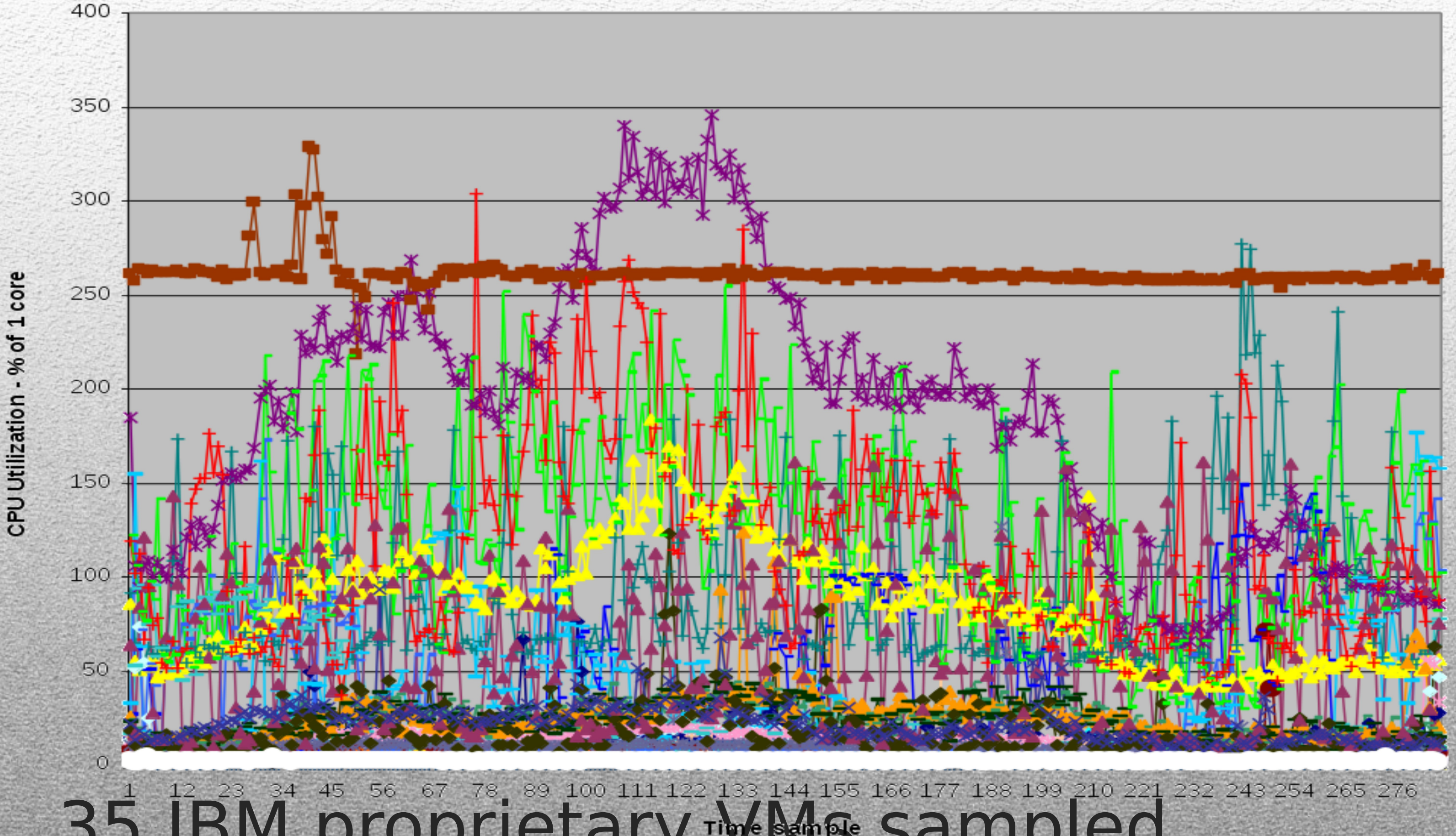
- Higher utilization for active servers
- lower headroom requires rapid rebalance when workload spikes

- A decrease in energy use –
- rapidly adjust servers online to load variation to preserve SLA

- Customers buy/install servers for peak loads
  - fast VM migration allows dynamic size to actu



Unsolved Needs

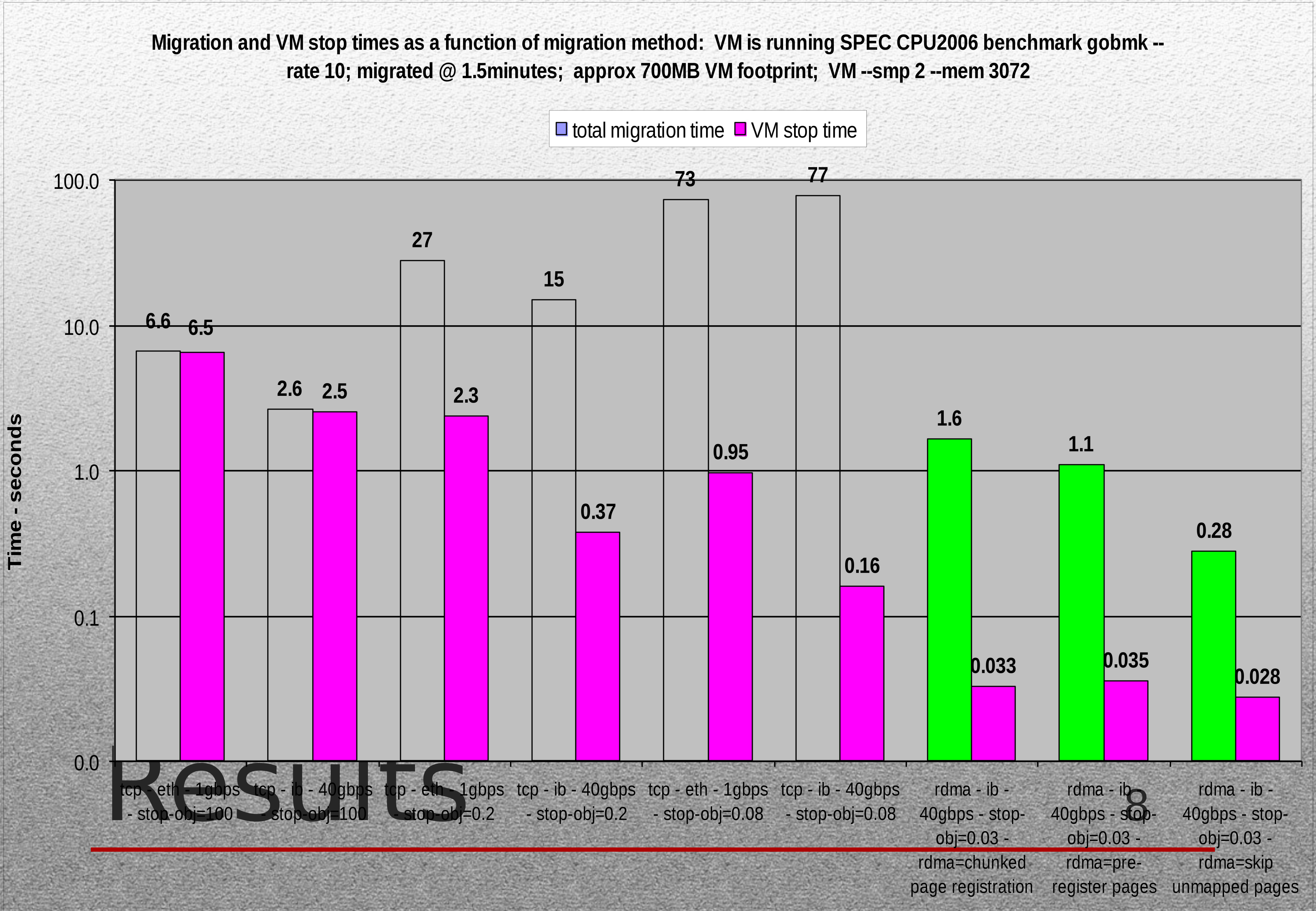


35 IBM proprietary VMs sampled every 5 minutes (WebSphere, DB2) over 24 hours

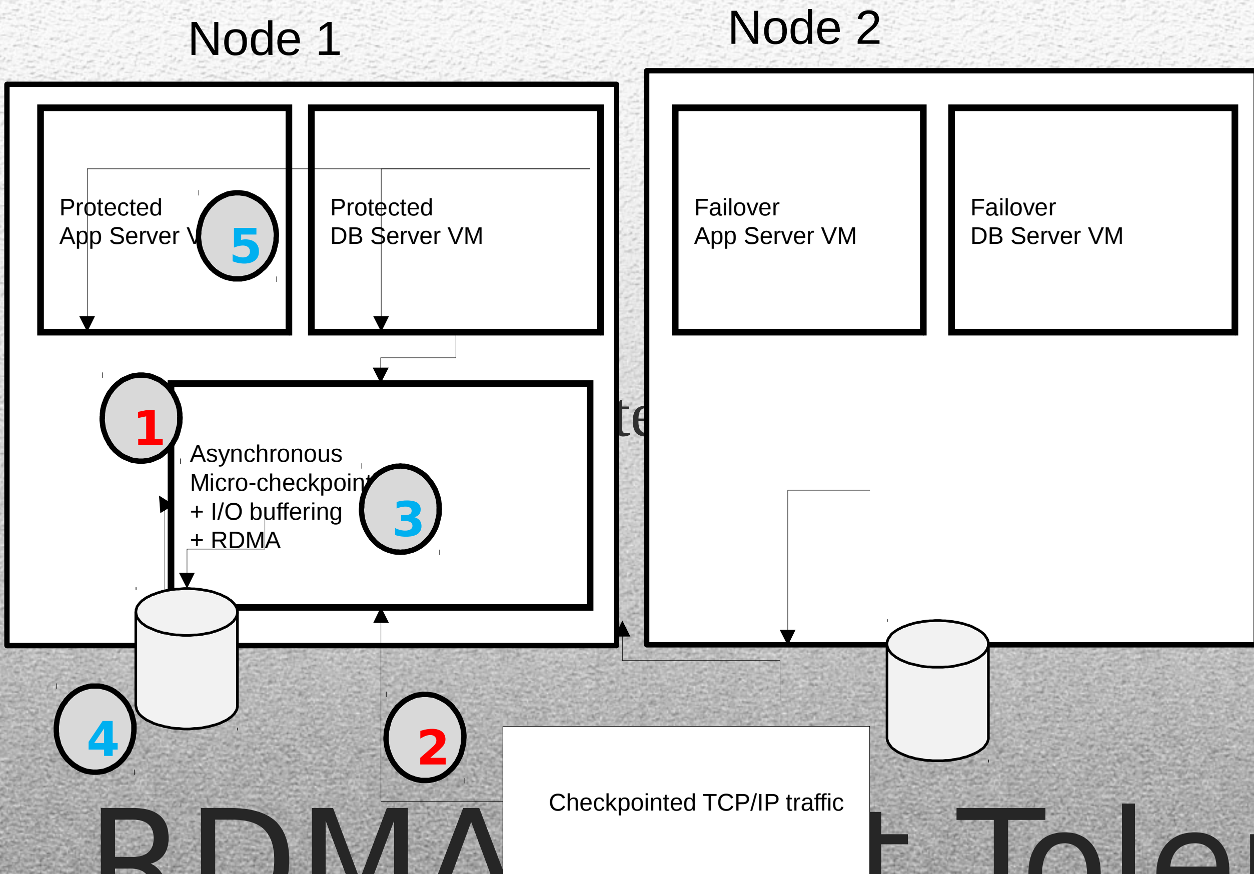
- **RDMA usage:**
  - Memory must be registered on both sides
  - Small RDMA transfers are kind of useless
    - IB programming is like IPv6:
      - Necessary evil, but proven technology
      - RDMA over Ethernet products from multiple Vendors
- **QEMU:**
  - Avoid changing the QEMUFile abstraction – many users
  - Avoid re-architecting savevm.c/arch_init.c
  - Avoid “mixing” TCP and RDMA protocol code –
    - They are really mutually exclusive programming paradigms
    - One gives you a bytestream and one does not

RDMA Migration Challenges

Results



Migration and VM stop times as a function of migration method: VM is running SPEC CPU2006 benchmark gobmk --rate 10; migrated @ 1.5minutes; approx 700MB VM footprint; VM --smp 2 --mem 3072
total migration time
VM stop time
Time - seconds
100.0
10.0
1.0
0.1
0.0
6.6
6.5
2.6
2.5
27
2.3
15
0.37
73
0.95
77
0.16
1.6
0.033
1.1
0.035
0.28
0.028
tcp - eth - 1gbps - stop-obj=100
tcp - ib - 40gbps stop-obj=100
tcp - eth - 1gbps - stop-obj=0.2
tcp - ib - 40gbps - stop-obj=0.2
tcp - eth - 1gbps - stop-obj=0.08
tcp - ib - 40gbps - stop-obj=0.08
rdma - ib - 40gbps - stop-obj=0.03 - rdma=chunked page registration
rdma - ib - 40gbps - stop-obj=0.03 - rdma=pre-register pages
rdma - ib - 40gbps - stop-obj=0.03 - rdma=skip unmapped pages



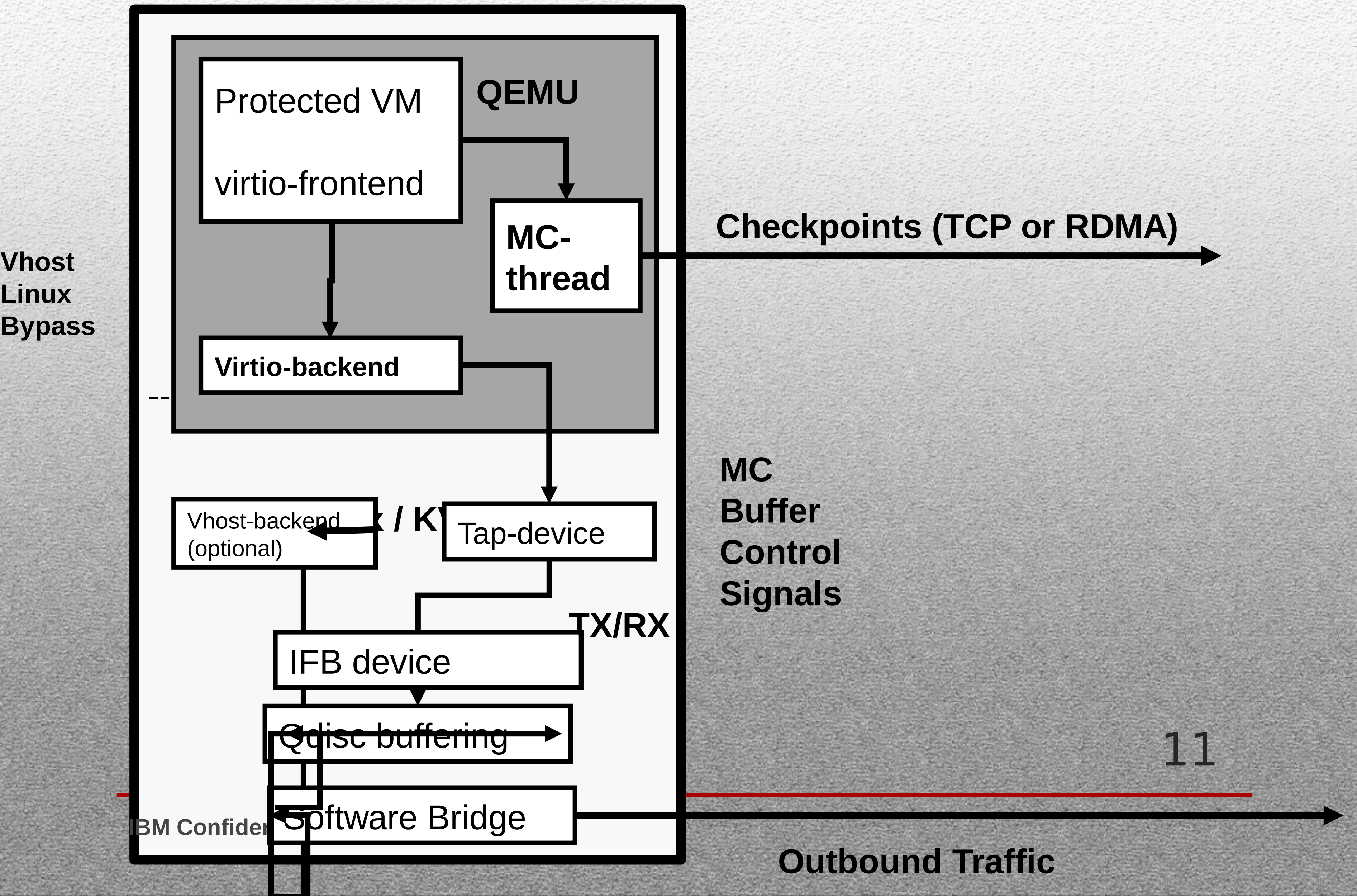
	Challenge	Status
#1	Memory / CPU	Done, ~10-20% penalty
#2	Network Buffering	Done, ~50% penalty
#3	Storage Buffering	In Progress
#4	RDMA Acceleration for I/O and Memory	In Plan
#5	Application Awareness	Not yet in plan

RDMA Fault Tolerance Status

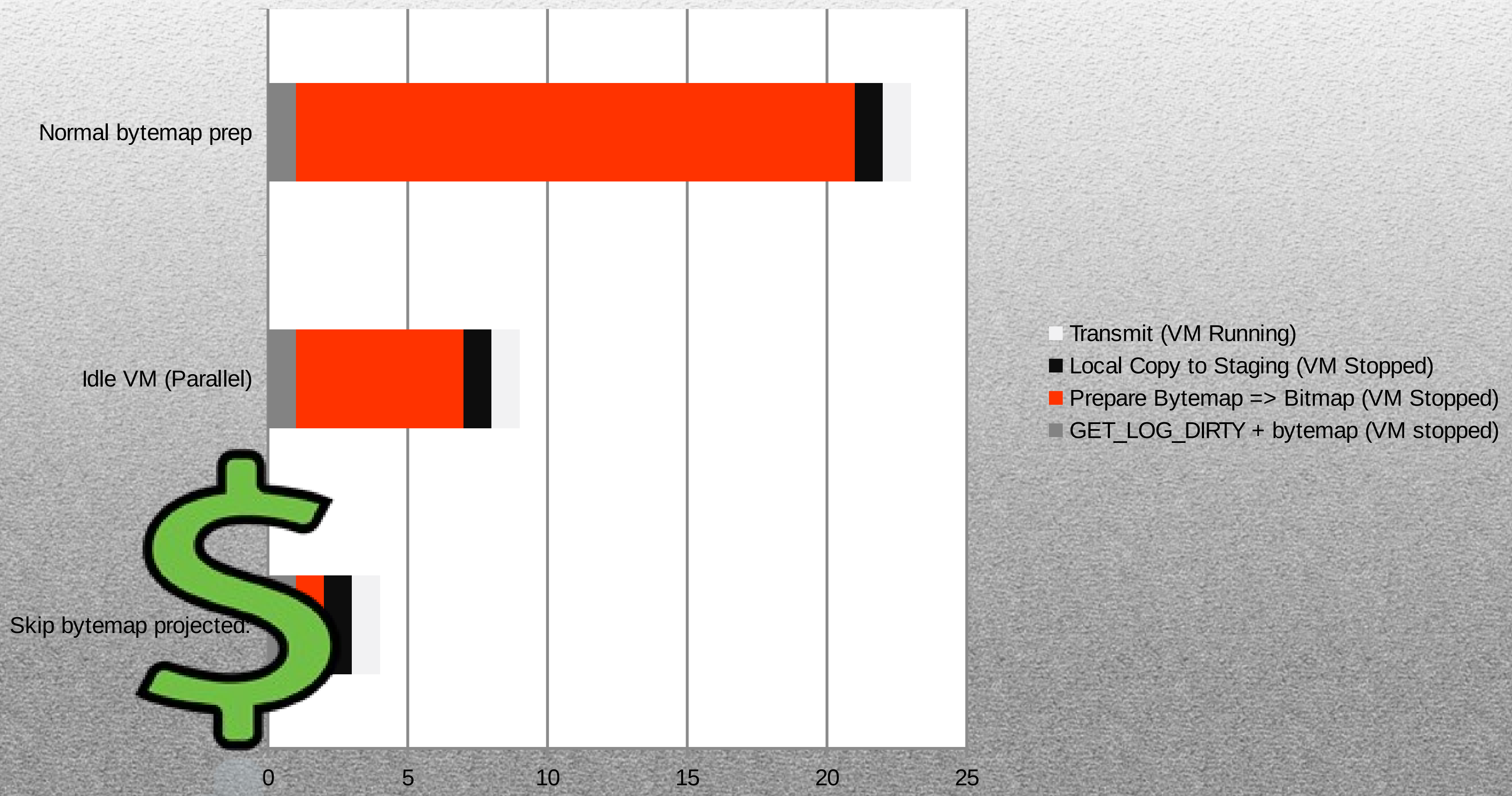
- **Sub-millisecond-FT requires consistent I/O**
  - Outbound network must be buffered
  - Outbound storage must be mirrored
- **Expensive Identification of dirty memory**
  - Multiple KVM log_dirty calls => QEMU bitmap
  - Multiple QEMU bitmaps => QEMU bytemap
    - Bytemap used by VGA? PCI? Others?
  - Bytemap => single bitmap
  - Stream through bitmap
- **60 milliseconds to identify dirty memory on a 16GB vm! Ahhhhh!**

RDMA Fault Tolerance Challenges

Network Barriers – how it works with Micro-Checkpointing (MC) ?



Micro-Checkpointing “barometer”: where do the costs come from?
Average checkpoint size < 5MB



Support for Over-commitment

- **Very "sensitive" topic =)**
  - Myth: Sub-second, sub-working-set over-commitment
  - Reality: Sub-day, sub-hour, working set
  - growth and contraction
- **RDMA Migration:**
  - Overcommit on both sides of connection?
  - Again: Does your management software know anything?
    - Lots of zero pages? VM is idle? Critical?
    - Why can't our management software have APIs that libvirt can share? Or maybe OpenStack can share?
    - Does policy engine know when to use RDMA?
- **RDMA Fault Tolerance:**
  - checkpoints arbitrary in size – double the VM in the worst case
  - Checkpoints also need to be compatible w/ RDMA

When *should* you use RDMA?

You (or your policy management) is dealing with a *known* memory bound workload that doesn’t converge with TCP

- **Migration Time = O(dirty rate)**
    - **Where Dirty rate > TCP bandwidth**
- You have sufficient available free memory on the destination host
- Your source host cannot tolerate stealing CPU cycles from VMs for TCP traffic

1. Is your VM idle?
   - **Migration Time = O(bulk transfer)**
   - Is your VM “young”? (Full of zeros?)
2. **- Migration Time = O(almost nothing)**
3. Your known-memory-bound application doesn’t converge AT ALL
   - **Migration Time = O(infinity)**
   - It is so heavy, you need “auto-converge” capability (new in 1.6)

When *not* to use RDMA Migration?

1. You have no idea what the workload is
   - But “someone” told you (i.e. paid) that it was important
2. Or you know what the workload is, but its memory content is mission-critical
3. I/O slow down is not a “big deal”
4. You must have high-availability
5. without re-architecting your workload

When should you use FT?

When you should *not* use FT?

1. Your workload can already survive restart
2. Your crashes are caused by *guest* bugs not host bugs or hardware failures
3. Your workload is stateless
4. Your workload cannot tolerate poor I/O
5. You’re customer is willing to tightly integrate with the “family” of HA tools:

    HACMP / Pacemaker / LinuxHA / Corosync / Zookeeper ….. The list goes on

- RDMA knows what a “RAMBlock” is
- New QEMUFileOps pointers:
  - save_page()

Allows for **source** ‘override’ of the transmission of a page

  - before_ram_iterate()

Allows for **source** to perform additional initialization steps *before* each migration iteration

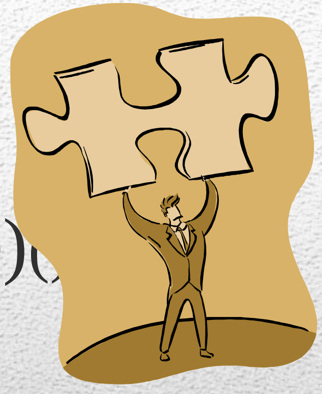
  - after_ram_iterate()

Similar **source** override, but at the end of each iteration

  - hook_ram_load()

Allows the **destination** override the *receipt* of memory during each iteration

(merged, 1) Technical Summary of QEMU Changes for RDMA



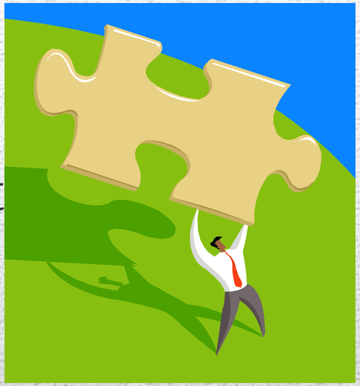
- QEMUFile operations:
  - Translate into IB Send/Recv messages
- qemu_savevm_state_(begin|iterate|complete)()
  - Invoke previous hooks (before_ram/after_ram)
  - Hooks instruct destination to handle memory registration on-demand
- ram_save_block()
  - Invokes save_page()
  - If RDMA not compiled or enabled, then we fallback to TCP

(merged, 2) Technical Summary of QEMU Changes for RDMA

- **Checkpoint size = O(2 * RAM), worst case**
  - Cannot allow QEMU to hog that much RAM
  - Cannot register that much with RDMA anyway
- **Solution:**
  - Splitup checkpoint into ‘slabs’ (~5MB each)
  - Grow and evict slabs as needed
  - Register slabs as they are needed

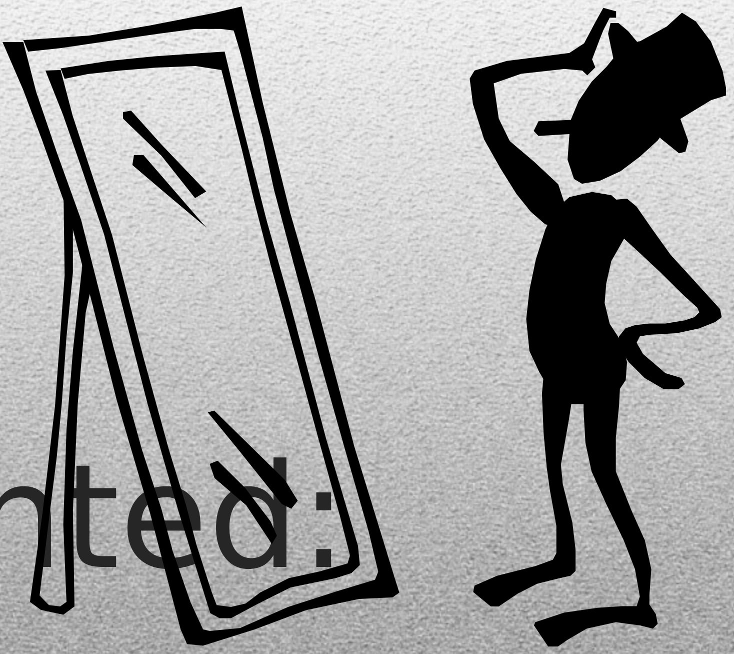
(Implemented) FT Technical Summary (2)

- **Network ‘output commit’ problem:**
  - Xen community has released their network buffering solut into netlink community:
    - Output Packets leave VM
    - End up in tap device
    - Tap device dumps packets into IFB device
    - Turns ‘inbound’ packets into ‘outbound’ packets
    - Qdisc ‘plug’ inserted into IFB device
    - Checkpoints are coordinated with ‘plug’ and ‘unplug’ operations



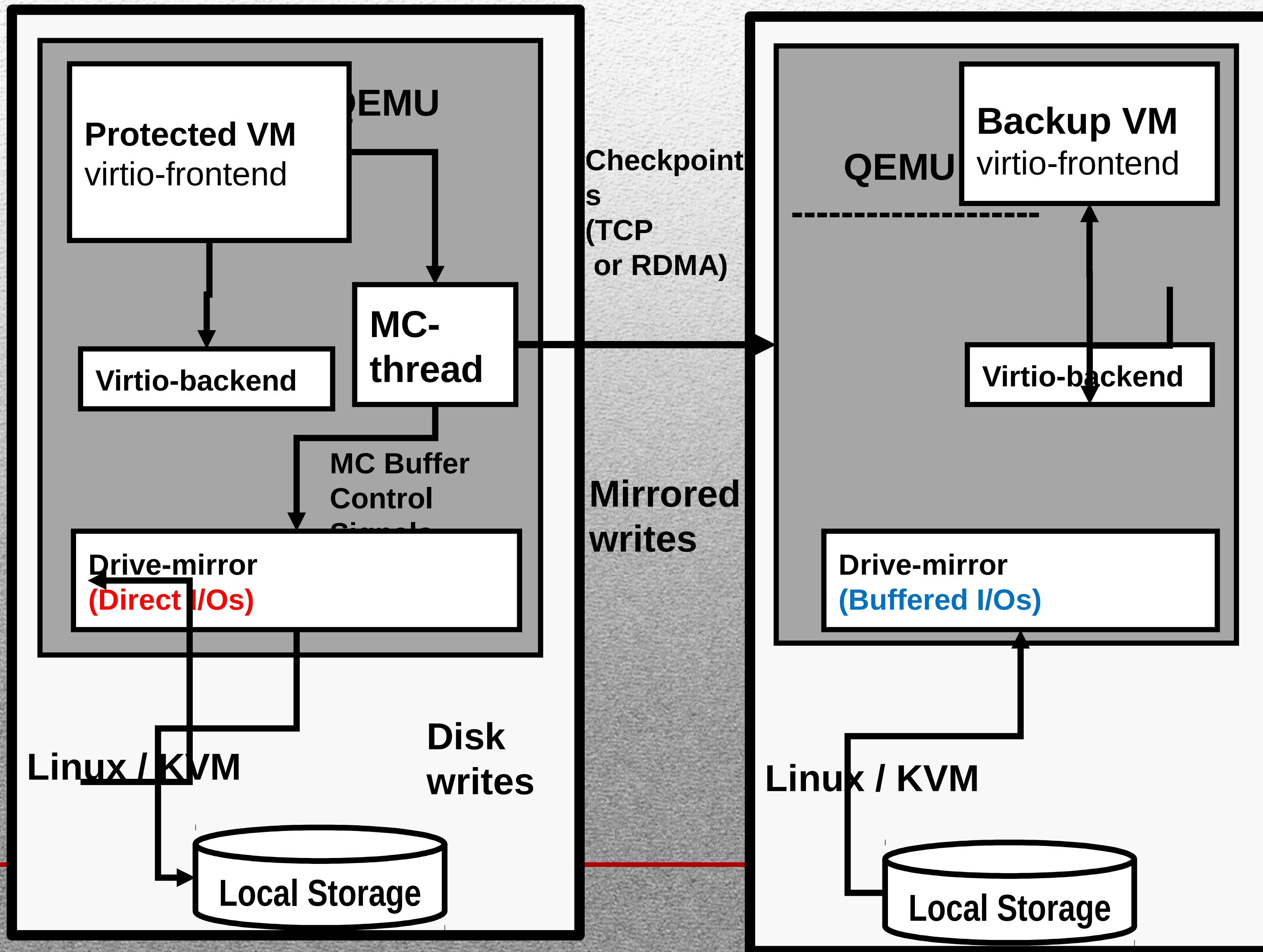
(Implemented) FT Technical Summary (2)

- **Plan is to active QEMU ‘drive-mirror’**
  - ‘drive-mirror’ would also need to be RDMA-capable
  - Local I/O blocks go to disk unfettered
  -
- **Mirrored blocks:**
- Held until checkpoint complete,
  - then released
  - This was chosen over shared storage
    - For performance reasons
    - And ordering constraints



Not implemented: FT storage mirroring

Drive-mirror + MC

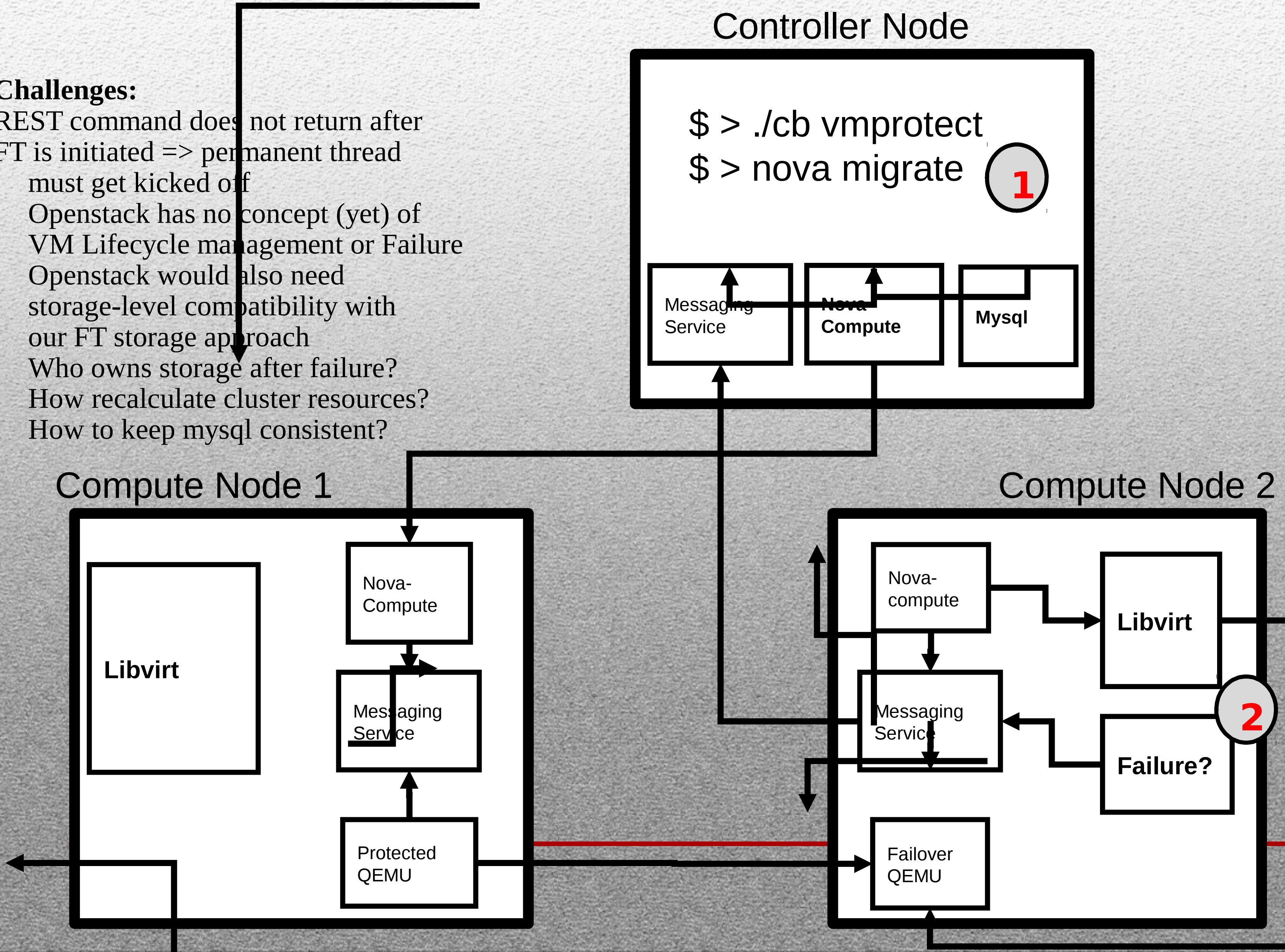


Thanks!

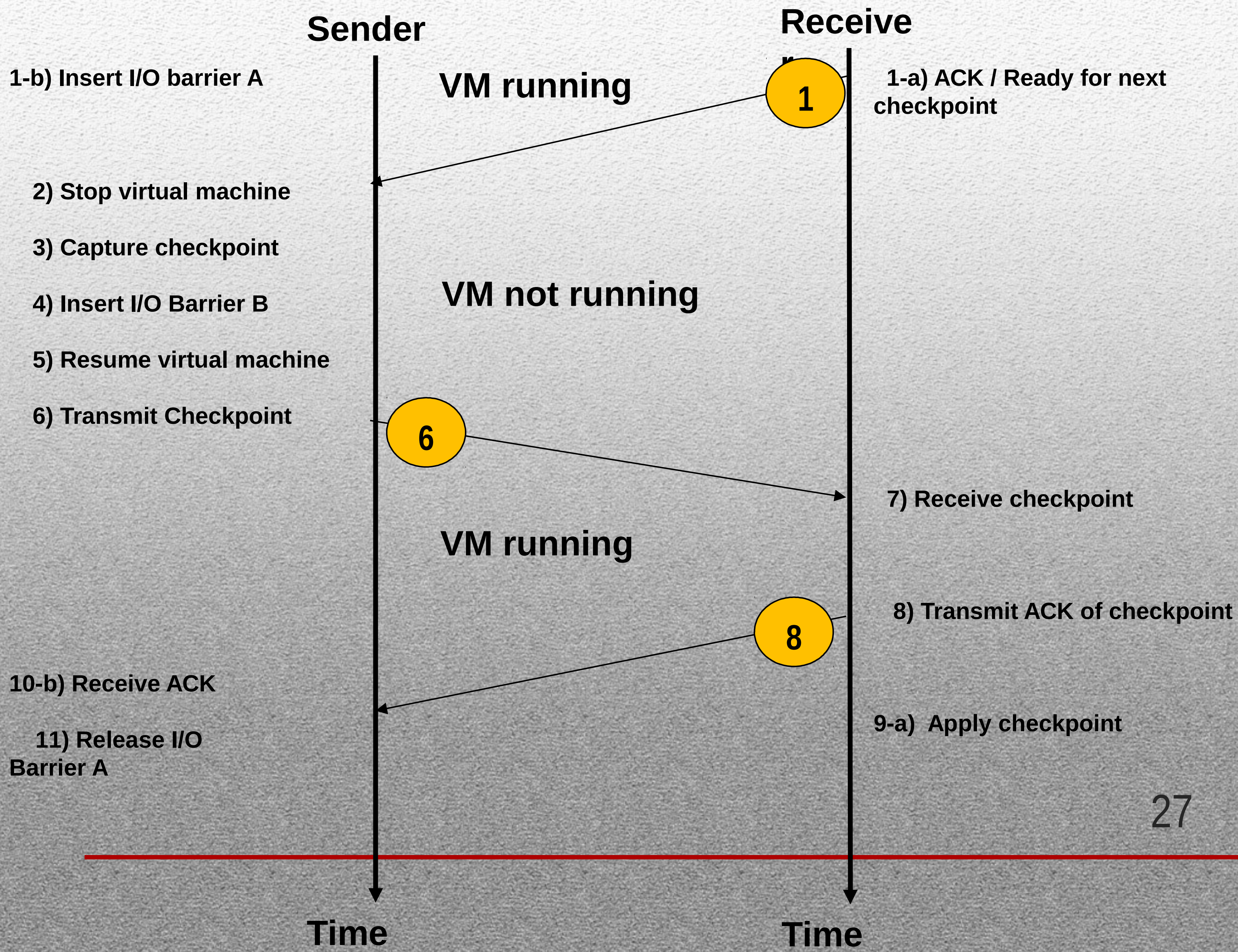
Backup Slides

OpenStack Cloud: Possible Management Workflow

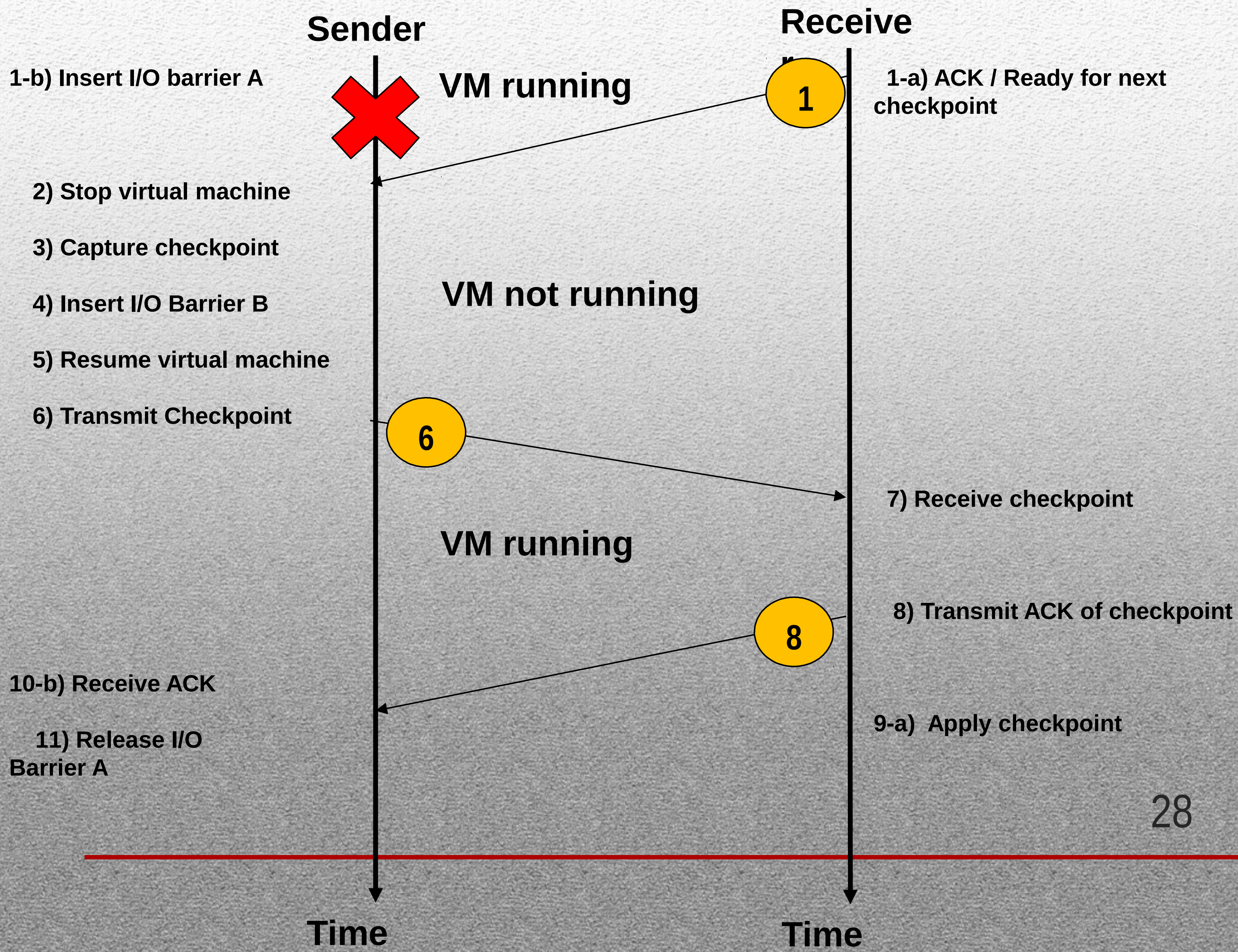
- **Challenges:**
- REST command does not return after
- FT is initiated => permanent thread
  - must get kicked off
  - Openstack has no concept (yet) of
  - VM Lifecycle management or Failure
  - Openstack would also need
  - storage-level compatibility with
  - our FT storage approach
  - Who owns storage after failure?
  - How recalculate cluster resources?
  - How to keep mysql consistent?



Sequence of Events:

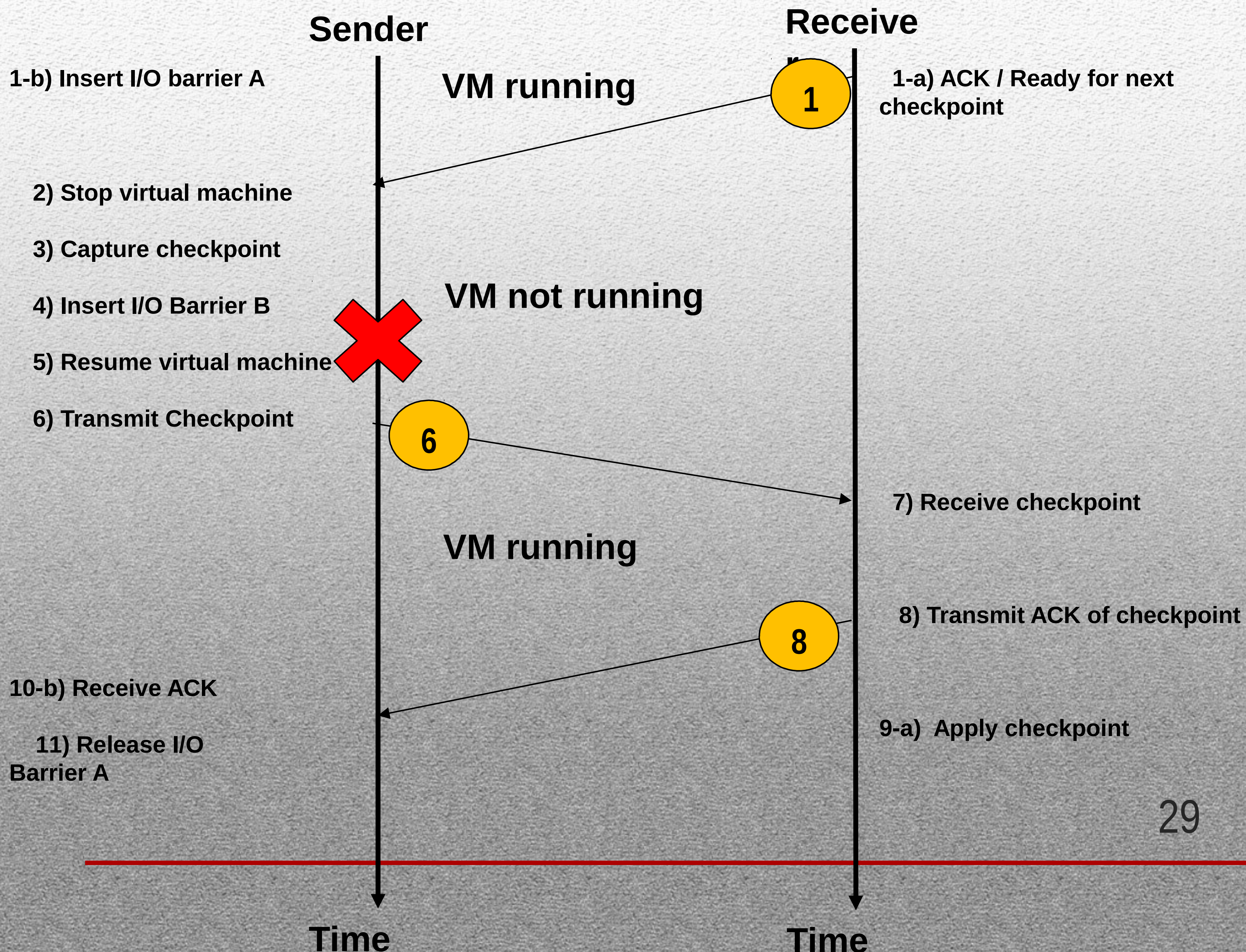


checkpoint

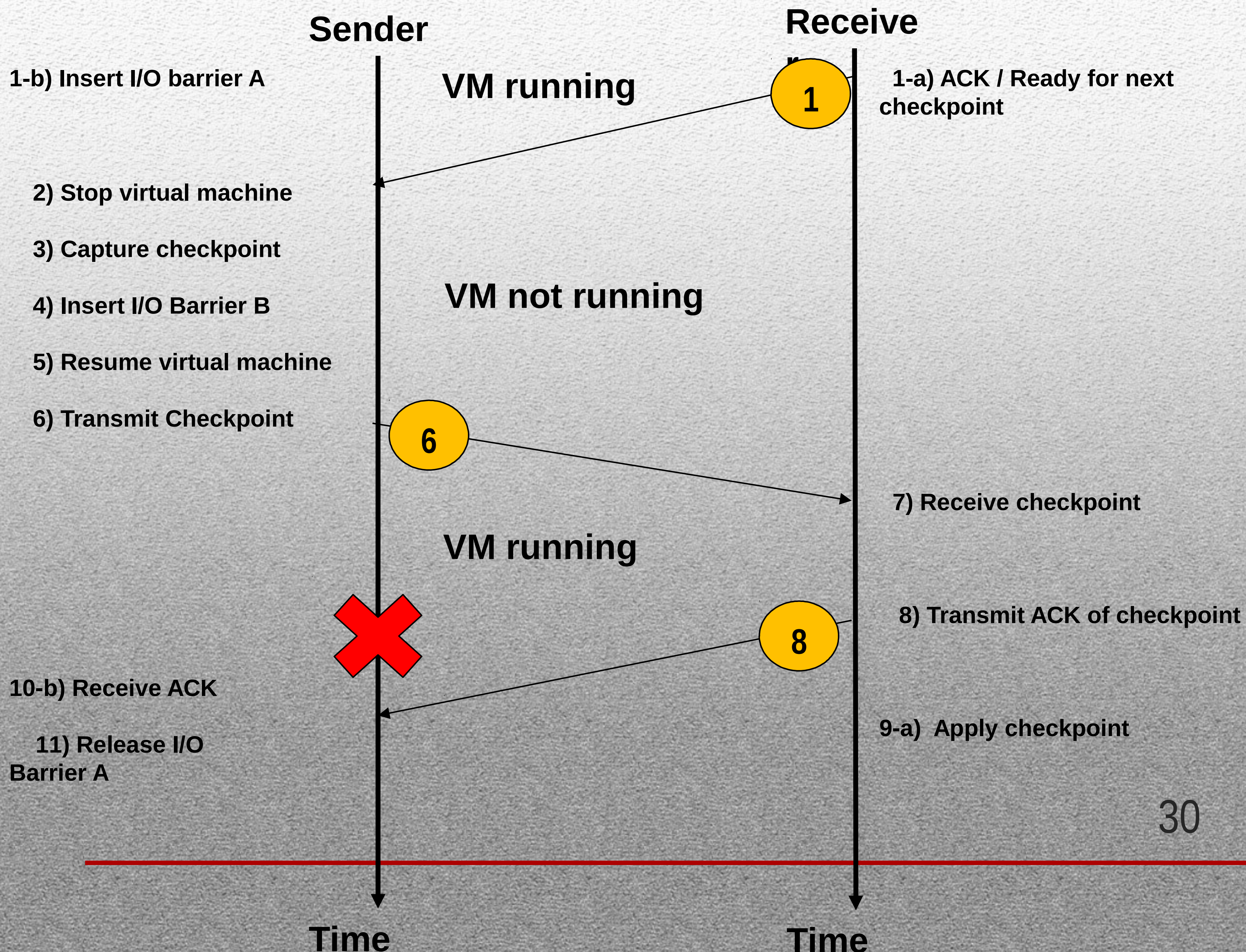


Sender
Receive
r
1-b) Insert I/O barrier A
VM running
1
1-a) ACK / Ready for next checkpoint
2) Stop virtual machine
3) Capture checkpoint
4) Insert I/O Barrier B
VM not running
5) Resume virtual machine
6) Transmit Checkpoint
6
7) Receive checkpoint
VM running
8
8) Transmit ACK of checkpoint
10-b) Receive ACK
9-a) Apply checkpoint
11) Release I/O Barrier A
Time
Time

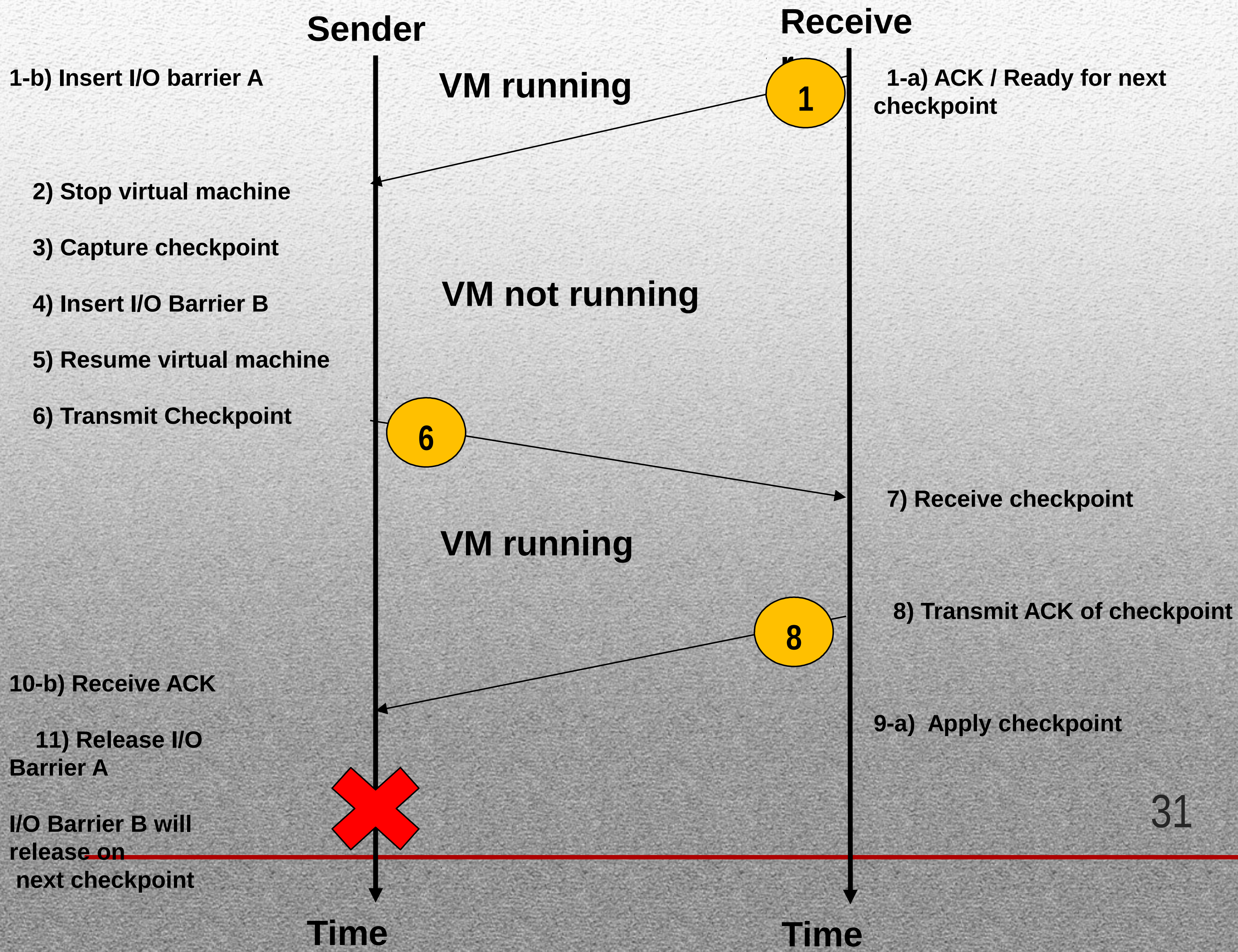
revert checkpoint



revert checkpoint



checkpoint



Sender
Receiver
1-b) Insert I/O barrier A
VM running
1
1-a) ACK / Ready for next checkpoint
2) Stop virtual machine
3) Capture checkpoint
4) Insert I/O Barrier B
VM not running
5) Resume virtual machine
6) Transmit Checkpoint
6
7) Receive checkpoint
VM running
8
8) Transmit ACK of checkpoint
10-b) Receive ACK
9-a) Apply checkpoint
11) Release I/O Barrier A
I/O Barrier B will release on next checkpoint
Time
Time