Yabusame update:
Postcopy Live migration for QEmu/KVM

* Isaku Yamahata, VALinux Systems Japan K.K. <yamahata@private.email.ne.jp>
Takahiro Hirofuchi, AIST <t.hirofuchi@aist.go.jp>

KVM Forum November 7th, 2012

Agenda

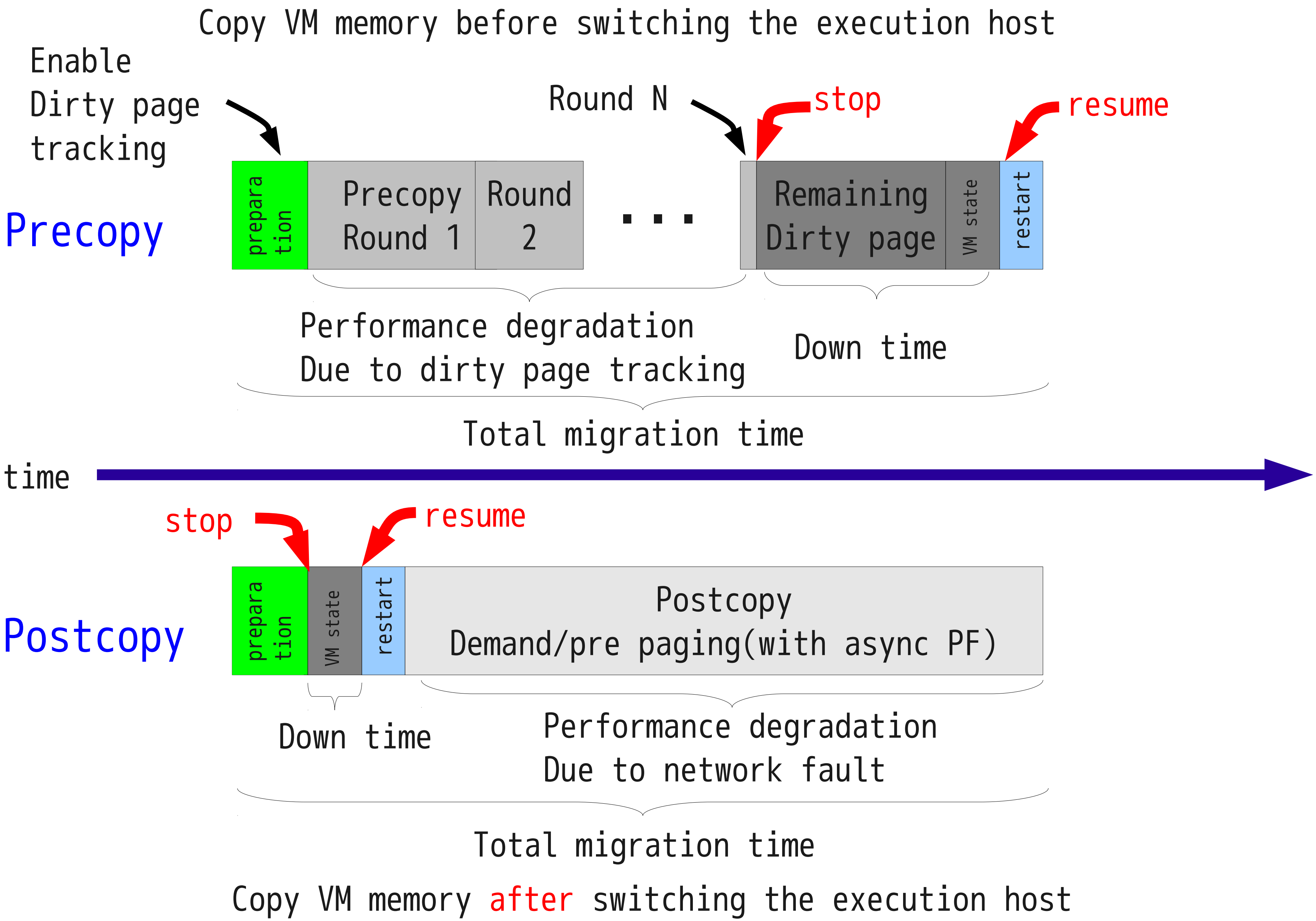
- Precopy vs Postcopy
- Update
- Evaluation
- Future work

From wikipedia

Yabusame is a joint project with Takahiro Hirofuchi, AIST and Satoshi Itoh, AIST.
This work is partly supported by JST/CREST ULP and KAKENHI (23700048).
The development of Yabusame was partly funded by METI (Minister of Economy, Trade and Industry) and supported by NTT Communications Corporation.

Precopy vs Postcopy

Precopy vs Postcopy



Copy VM memory before switching the execution host

Enable
Dirty page
tracking

Round N

stop

resume

Precopy

prepara
tion

Precopy
Round 1

Round
2

...

Remaining
Dirty page

VM state

restart

Performance degradation
Due to dirty page tracking

Down time

Total migration time

time

stop

resume

Postcopy

prepara
tion

VM state

restart

Postcopy
Demand/pre paging(with async PF)

Down time

Performance degradation
Due to network fault

Total migration time

Copy VM memory after switching the execution host

Characteristic comparison

- Precopy
  - Reliablility
    - Migration process can be aborted safely.
  - Total migration time and downtime depend on memory dirtying speed
    - Especially the number of dirty pages doesn't converge when dirtying speed > link speed
- Postcopy
  - network bandwidth friendly
    - Postcopy transfer a page only once
  - reliability
    - VM can be lost if network failure occurs during migration

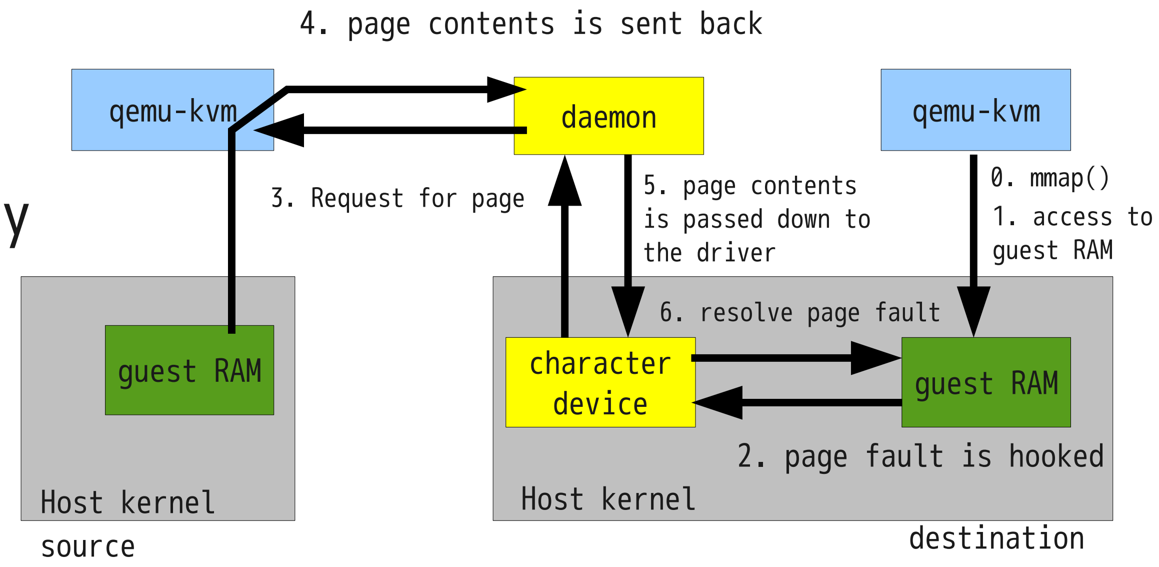
Postcopy is applicable for

- Planned maintenance
  - Predictable total migration time is important
- Dynamic consolidation
  - In cloud use case, usually resources are over-committed
  - If machine load becomes high, evacuate the VM to other machine promptly
    - Precopy optimization (= CPU cycle) may make things worse
- Wide area migration
  - Inter-Datacenter live-migration
    - L2 connectivity among datacenters with L2 over L3 has becoming common
    - VM migration over DCs as Disaster Recovery
- LAN case
  - Not all network bandwidth can be used for migration
  - network bandwidth might be reserved by QoS

Updates on Yabusame implementation

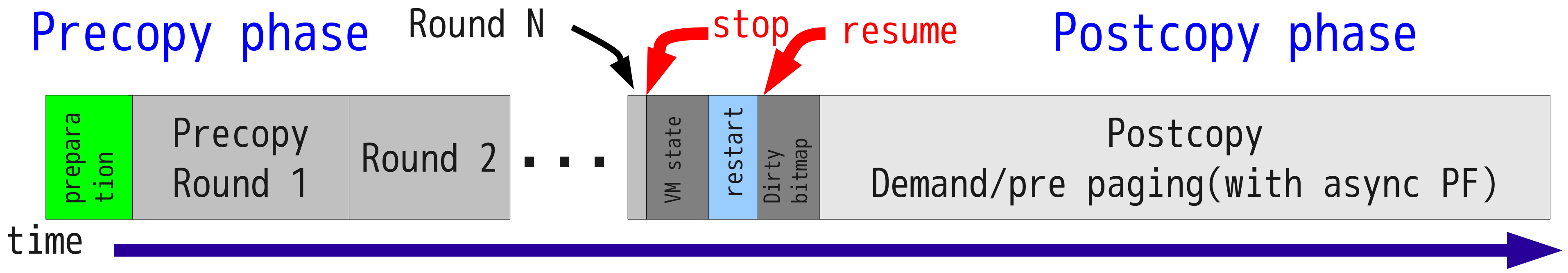
Updates

- Basic design is unchanged
  - Using a character device
- Precopy + postcopy optimization
- Auto detection of postcopy session
- Incoming side threading
- Reduced memory overhead



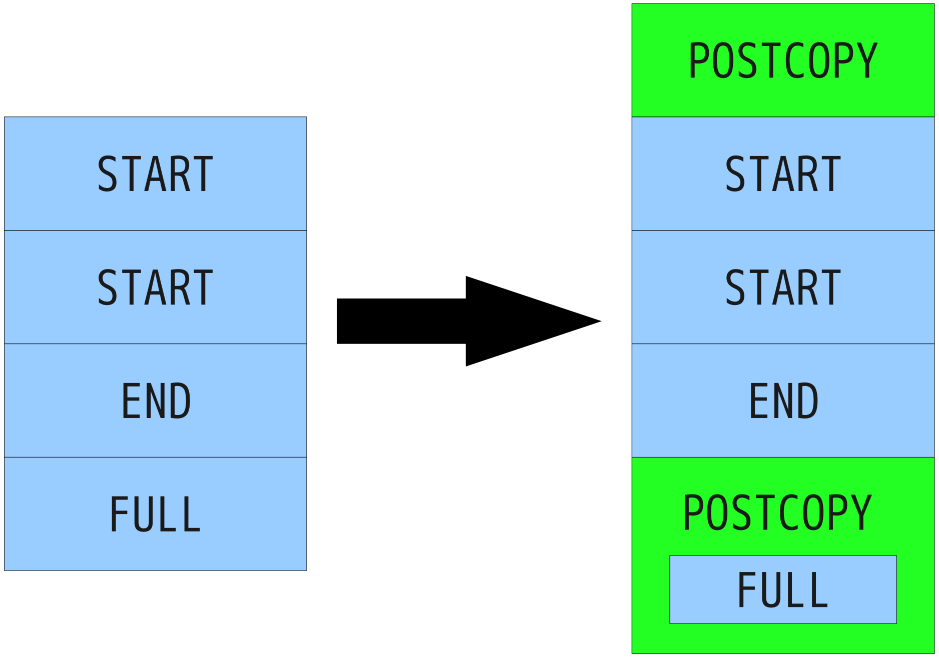
Precopy + postcopy

- migrate -p URI [<precopy count>]
  - Precopy count = 0 => disabling precopy
- Precopy the designated times (or meets downtime), then switches to postcopy mode
- Send dirty bitmaps after precopy
- Sending bitmap is tricky
  - qemu bitmap representation is unsigned long[] which is architecture dependent



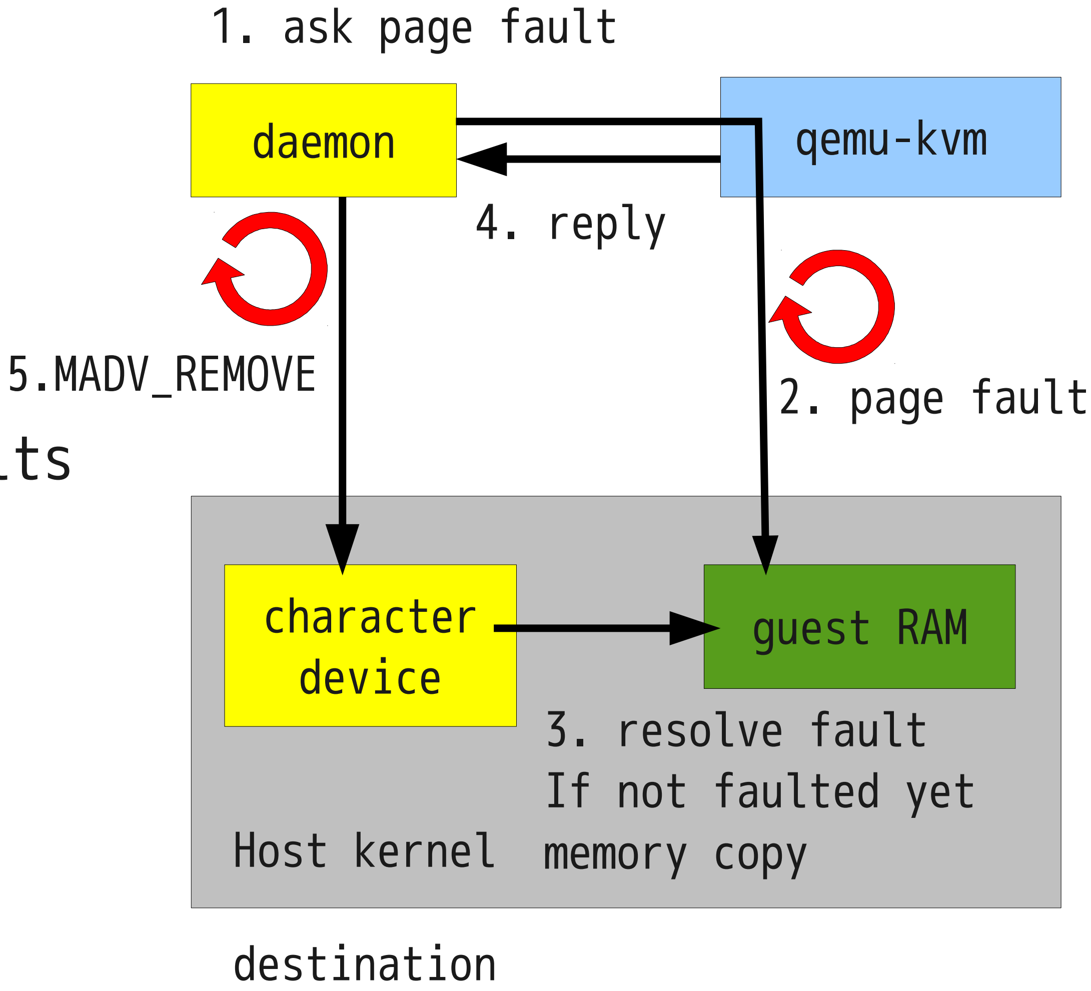
postcopy auto detection

- Incoming side auto-detects postcopy session
- New QEMU_VM_POSTCOPY section
- If incoming side doesn't know postcopy, it notices the new section as unknown and results in error.
- FULL section in POSTCOPY
  - Some device touches guest RAM at post_lost



Threading in incoming side

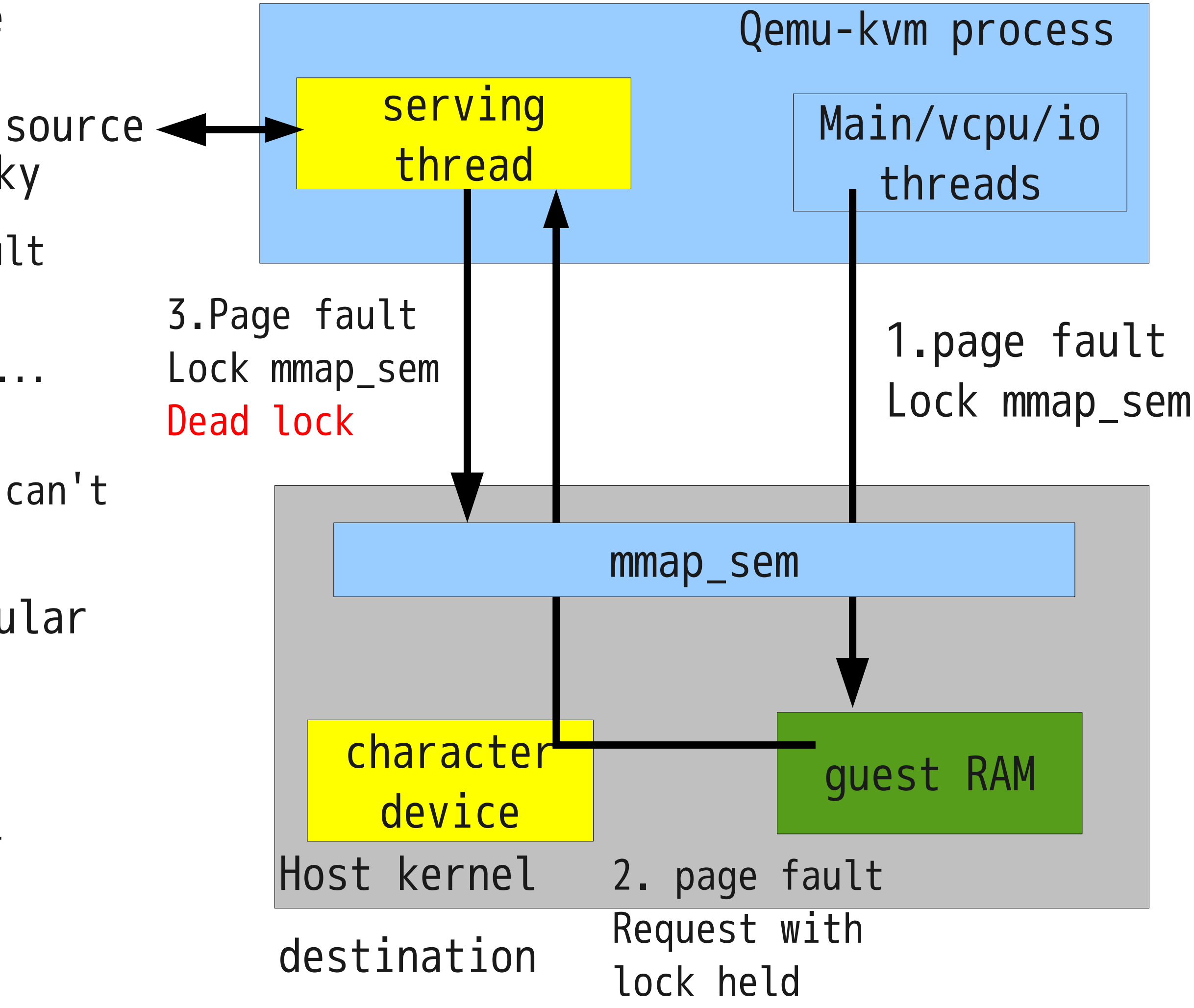
- Code simplification
  - thread vs select multiplex
- Reduce memory overhead
  - Dedicated threads
  - Make sure qemu-kvm faults on the page
  - frees already copy of served pages with dedicated thread



Findings

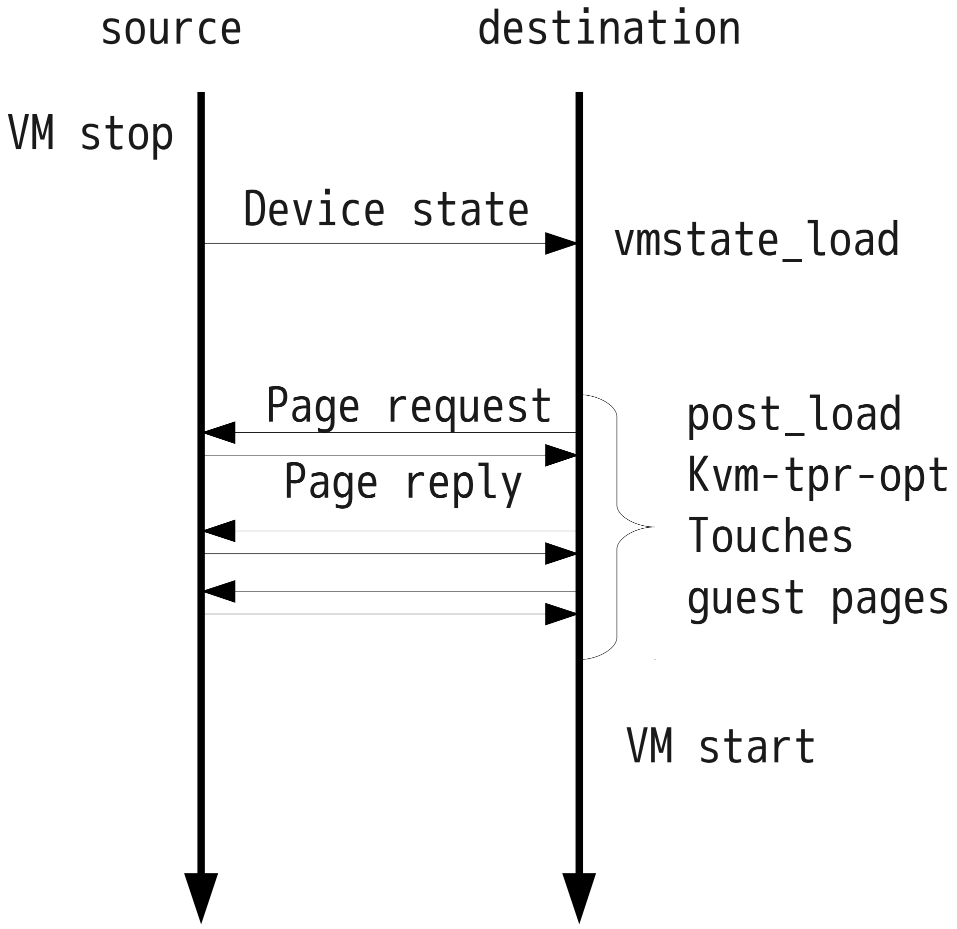
Serving page fault in qemu process is difficult

- Serving thread in qemu process
  - Finer control would be possible
- Abandaned to reverted to fork
- Playing with mmap_sem is tricky
  - Some threads are already in fault handler with mmap_sem held.
    - Sometimes write lock is held: AIO...
    - Qemu does use AIO
  - Threads that serves page fault can't page-fault
- Even with multi process, circular dependency is possible
  - Same with FUSE
  - Essential solution is in-kernel



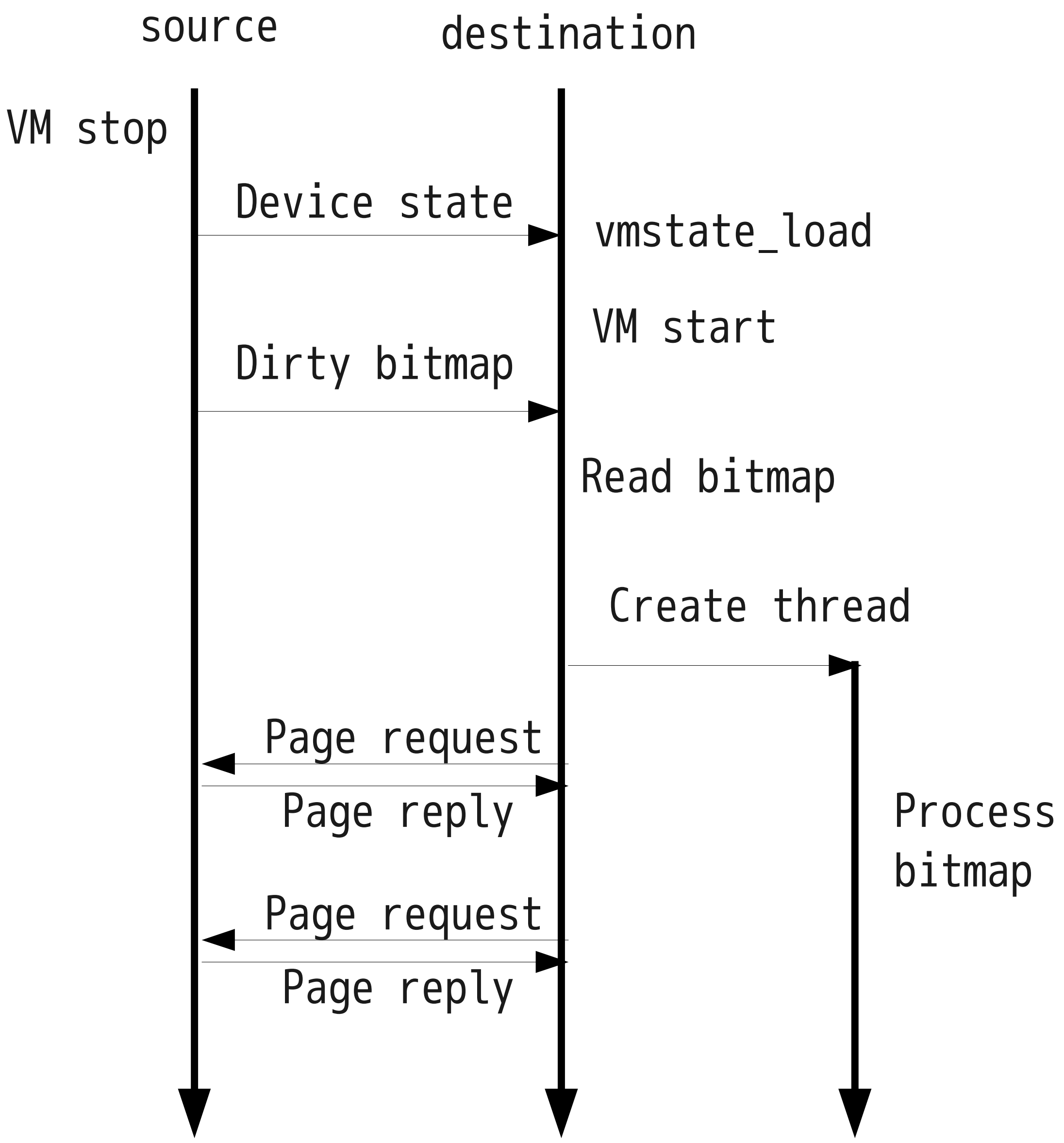
post_load

- Network fault right while device state load
  - Some post_load() touches guest RAM
    - Kvm-tpr-opt: patches guest RAM
    - Some devices start DMA emulation in qemu
  - Qemu main thread blocks before running vcpu thread
- pre+post optimization helps



pre+post dirty bitmap

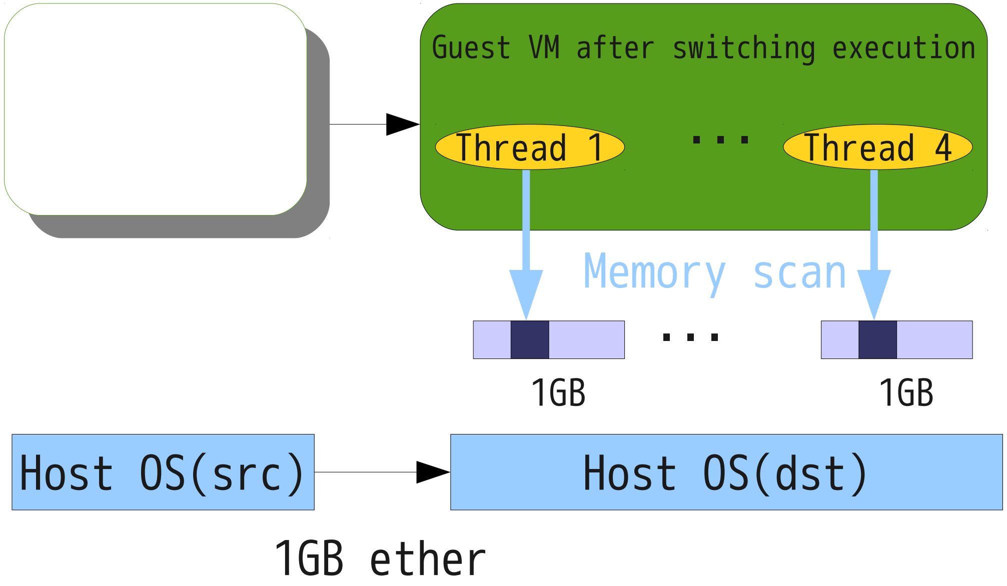
- Processing dirty bitmap causes long time
- Move it into another thread



Evaluation

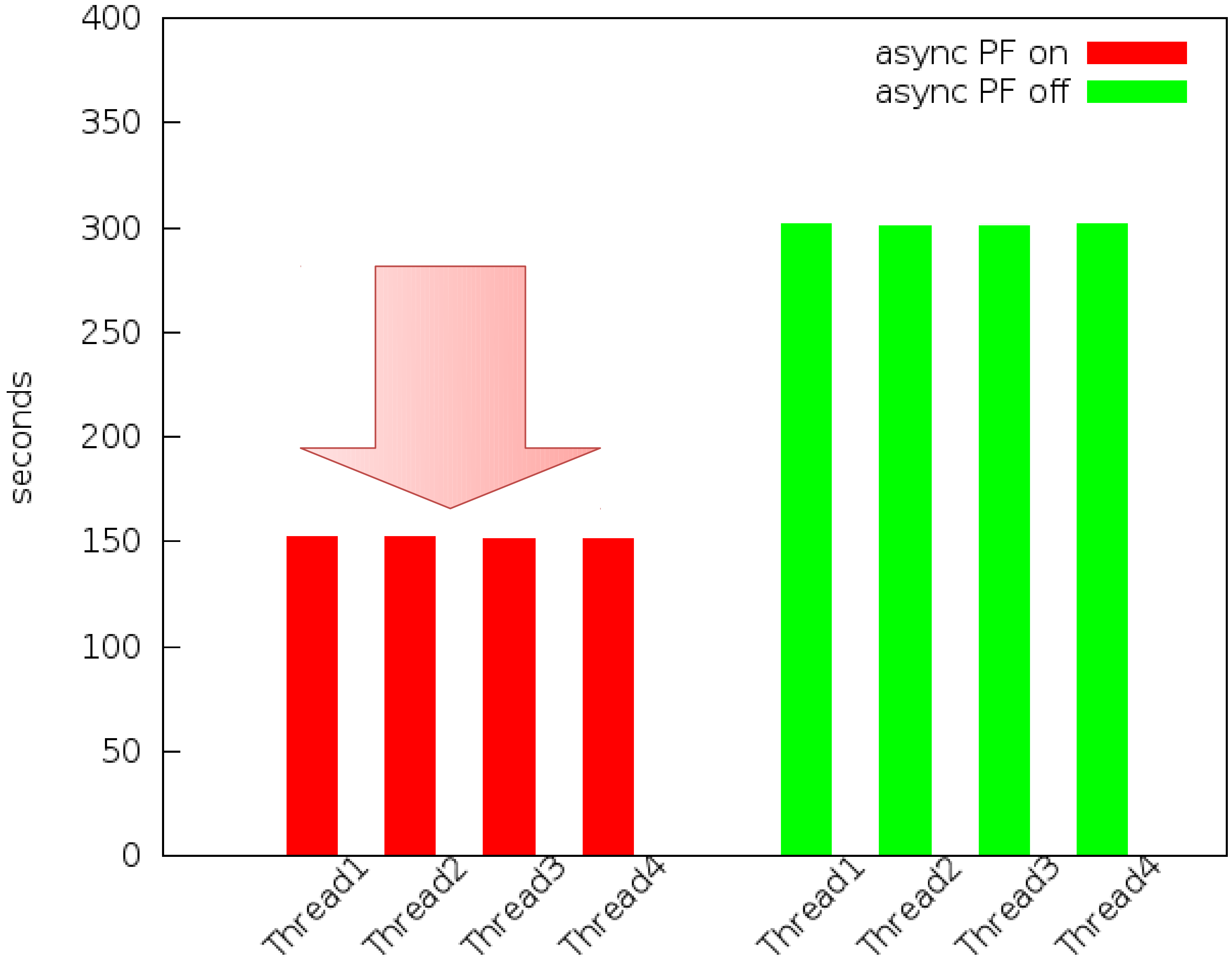
Memory scanning with postcopy

- 6GB memory Guest RAM
- 4thread
- Per-thread
  - 1GB
  - Each thread accessing all pages
  - Time from the first page access to the last page access
- Start each thread right after starting post-copy migration
- Background transfer is disabled

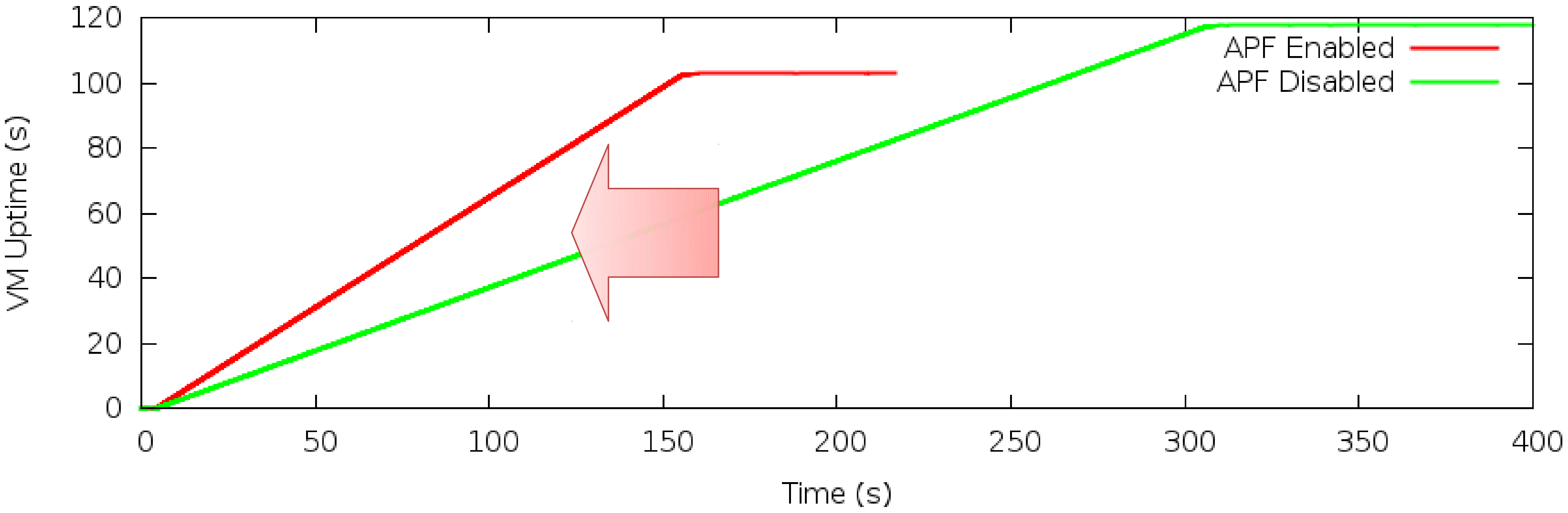


This evaliation is done with old implementation

Memory scan time(real)



Total CPU time allocated to guest VM



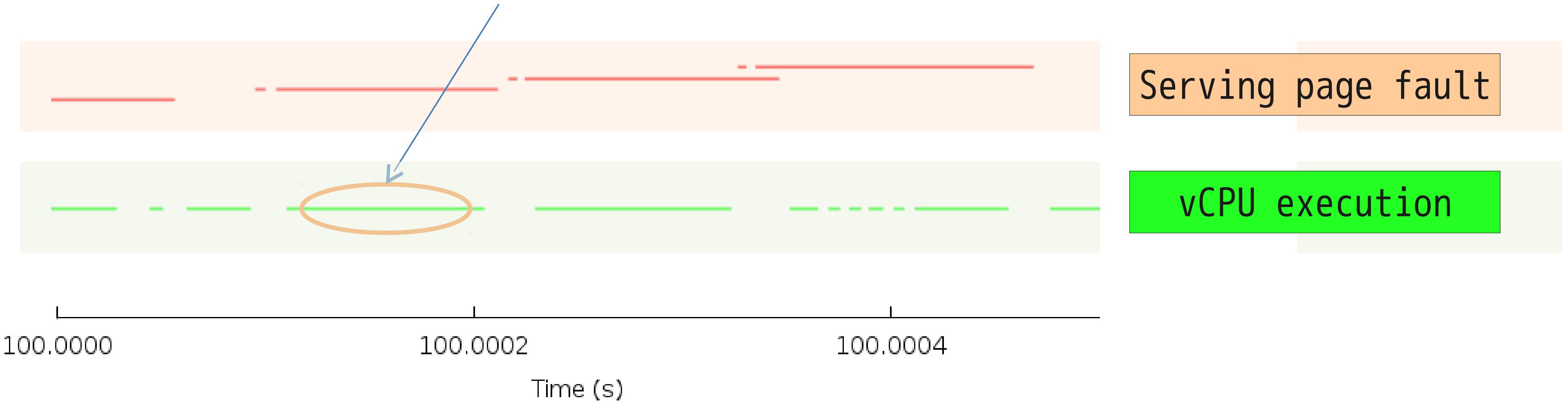
VCPU execution efficiency is improved cpu-time/real-time

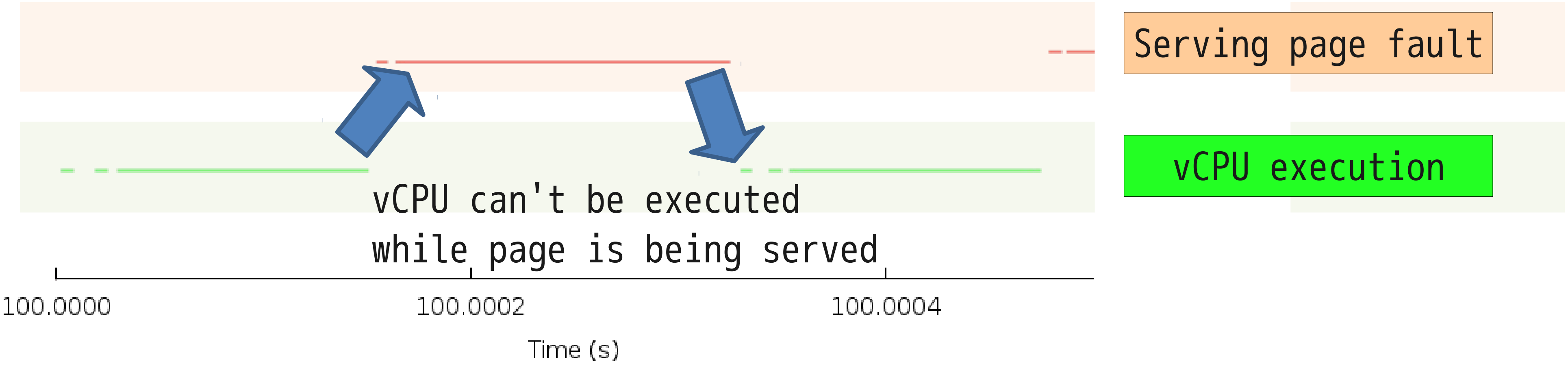
APF enabled: 0.7

APF disabled: 0.39

Analyze with SystemTap

vCPU is executing during page is being served

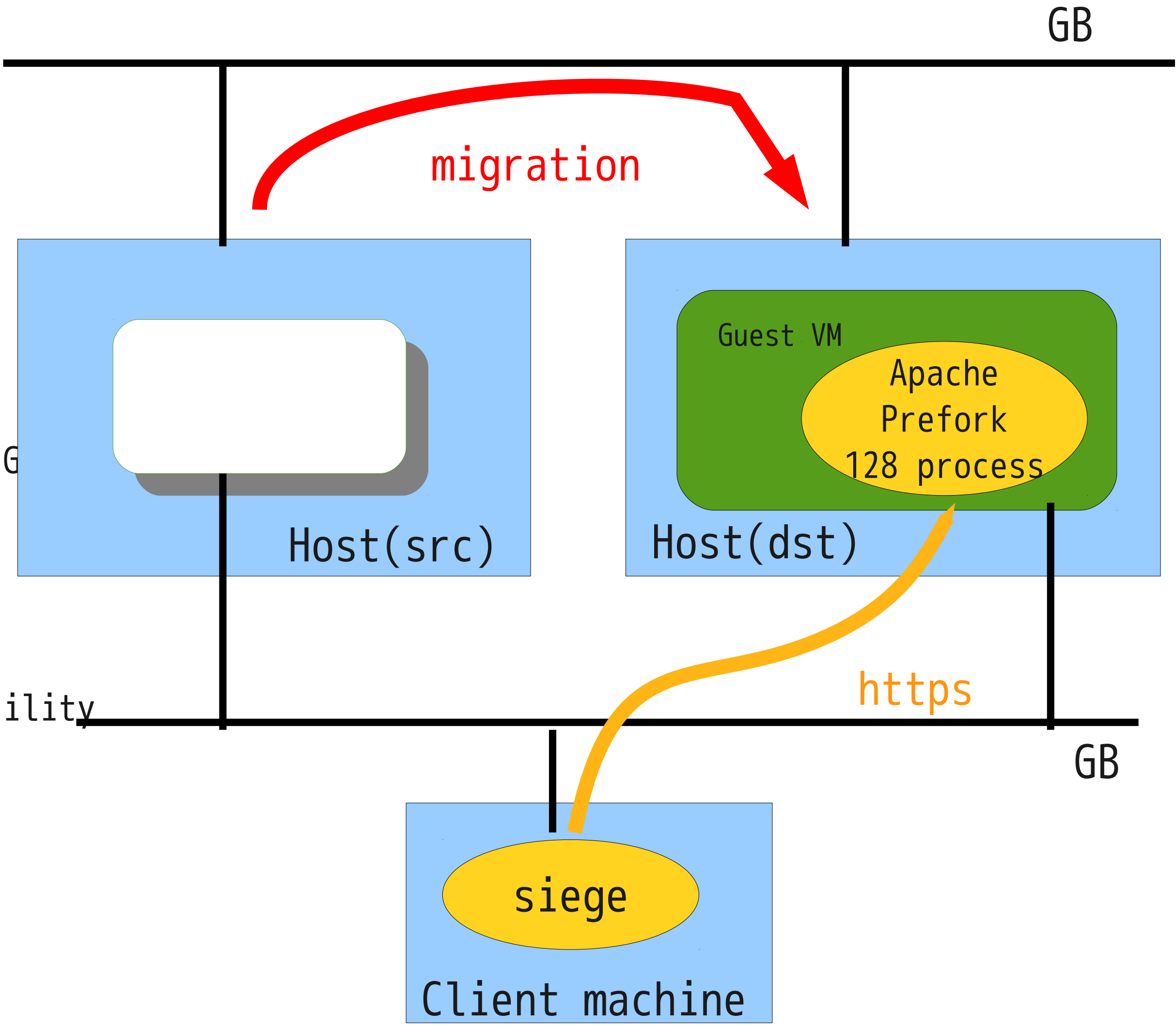




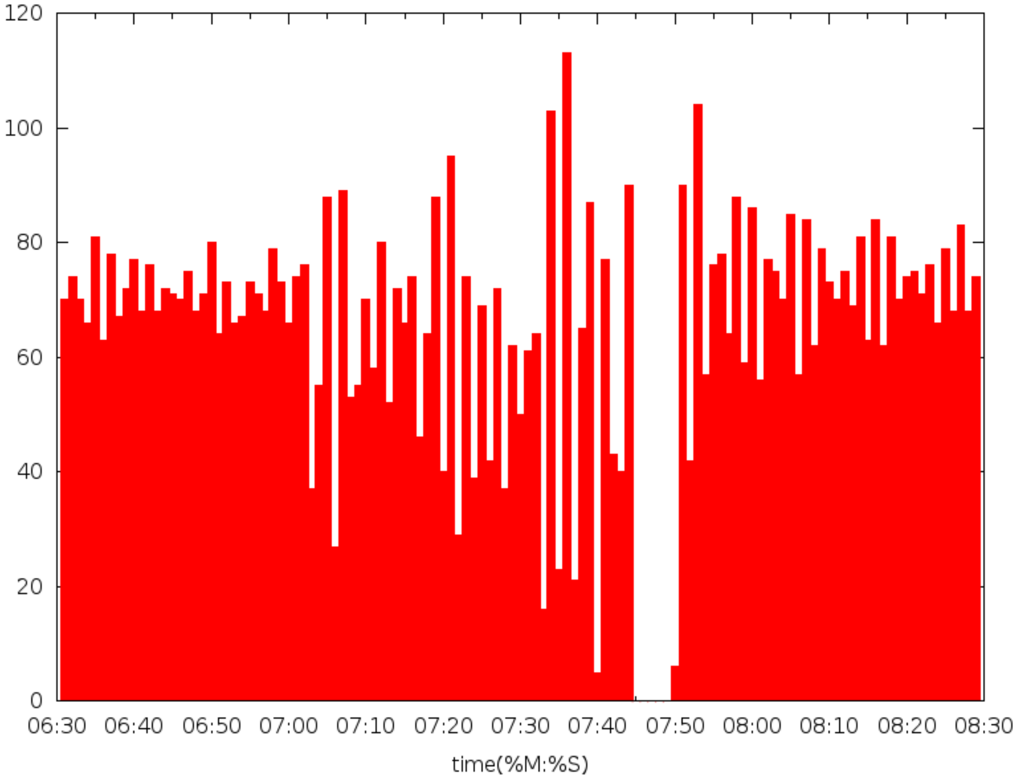
vCPU can't be executed
while page is being served

Siege benchmark with Apache 1GB case

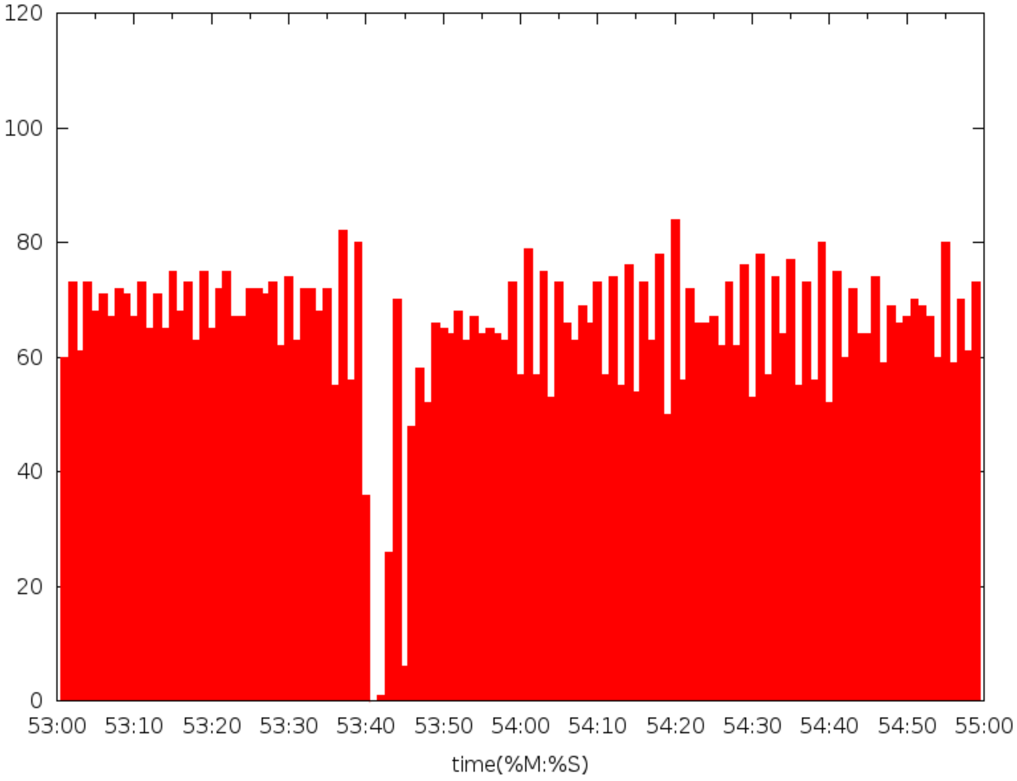
- Host
  - Core2 quad CPU Q9400 4core 2.66GHz
  - Memory 16GB
- Qemu/kvm
  - Memory about 6G(-m 6000)
  - Virtio, kernel_irqchip
- Apache
  - Prefork 128 process fixed
  - Data: 200K * 100000000files = about 2G
  - Warm up by siege before migration
    - So all data are cached on memory
- Siege
  - http load testing and benchmarking utility
  - http://www.joedog.org/siege-home/
  - 128 parallel (-c 128)
  - Random URL to 10000 URLs



Precopy



Postcopy



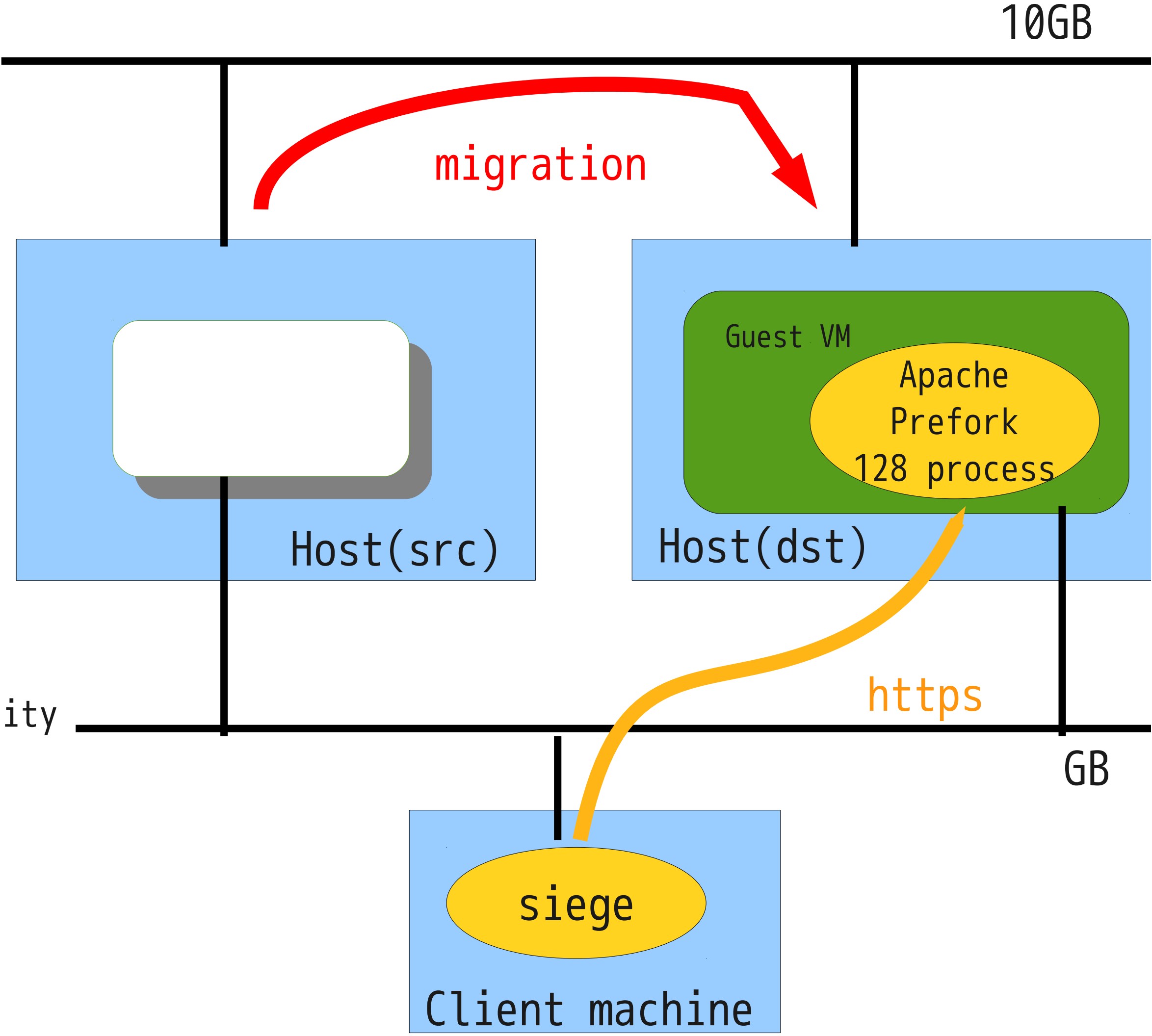
Precopy
Migrate set_speed=125M
(without XBZRLE)

Postcopy w/o background transfer
Prefault forward=100

```
migrate -p -n tcp:<IP address>:4444 0 100 0
```

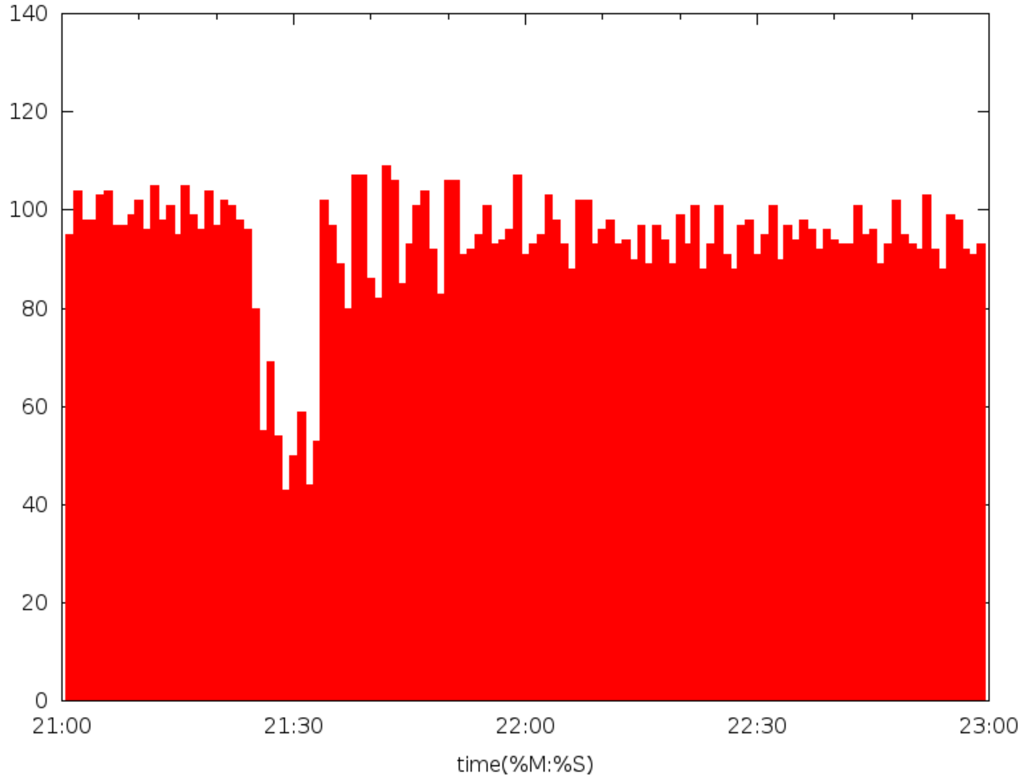
Siege benchmark with Apache 10GB case

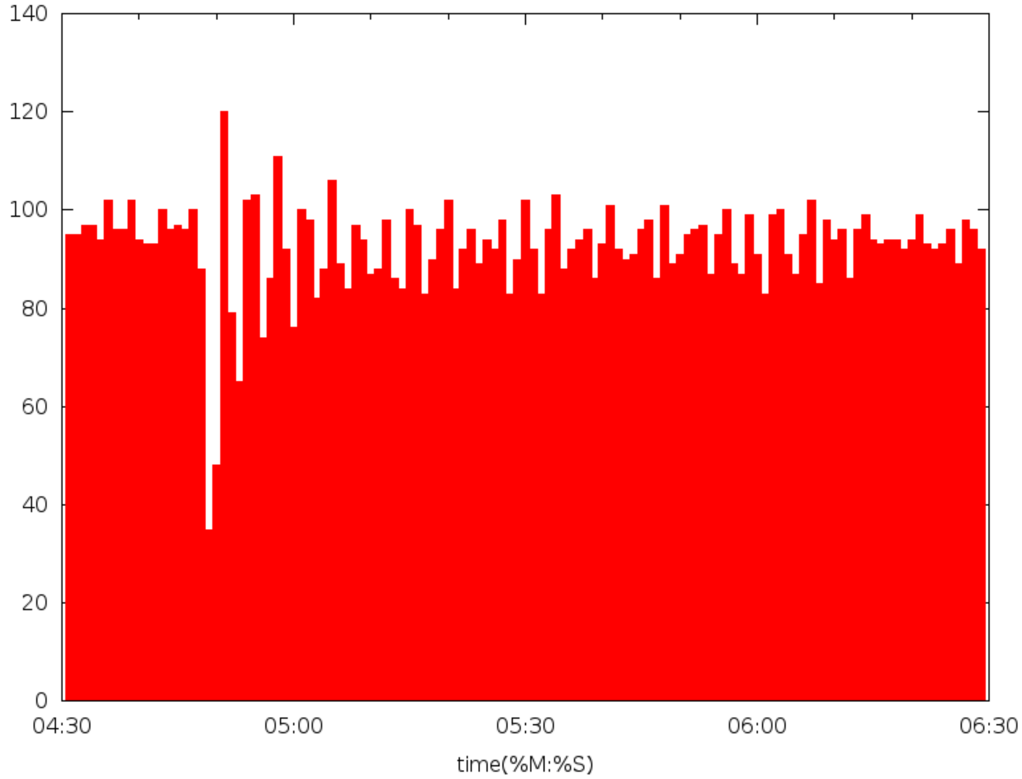
- Host
  - Xeon quad CPU E5620 2.40GHz * 2
  - Memory 24GB
- Qemu/kvm
  - Memory about 6G(-m 6000)
  - Virtio
- Apache
  - Prefork 128 process fixed
  - Data: 200K * 100000000files = about 2GB
  - Warm up by siege before migration
    - So all data are cached on memory
- Siege
  - http load testing and benchmarking utility
  - http://www.joedog.org/siege-home/
  - 128 parallel (-c 128)
  - Random URL to 10000 URLs



Precopy

Postcopy





```
Precopy
Migrate set_speed=1250M
(without XBZRLE)
```

```
Postcopy w/o background transfer
Prefault forward=400
migrate -p -n tcp:<IP address>:4444 0 400 0
```

Future work

- Upstream merge
  - Benchmark: Others are already working on it.(Benoit Hudzia and Vinod, Chegu)
  - Integration with RDMA approach. Find clean design
  - Investigate for fuse version of umem device and evaluation
    - See if it's possible and its performance is acceptable
- downtime work
  - Fetch latency sensitive page first
    - post_load page
    - Pv device page

Thank you

- Questions?
- Resources
  - Project page
    - http://grivon.apgrid.org/quick-kvm-migration
    - http://sites.google.com/site/grivonhome/quick-kvm-migration
  - Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension: proof-of-concept, ad-hoc prototype. not a new design
    - http://grivon.googlecode.com/svn/pub/docs/ccgrid2010-hirofuchi-paper.pdf
    - http://grivon.googlecode.com/svn/pub/docs/ccgrid2010-hirofuchi-talk.pdf
  - Reactive consolidation of virtual machines enabled by postcopy live migration: advantage for VM consolidation
    - http://portal.acm.org/citation.cfm?id=1996125
    - http://www.emn.fr/x-info/ascola/lib/exe/fetch.php?media=internet:vtdc-postcopy.pdf
  - Qemu Wiki
    - http://wiki.qemu.org/Features/PostCopyLiveMigration
  - Demo video
    - http://www.youtube.com/watch?v=lo2JJ2KWrlA
  - Github repo
    - git://github.com/yamahata/qemu.git qemu-postcopy-nov-03-2012
    - git://github.com/yamahata/linux-umem.git linux-umem-oct-29-2012