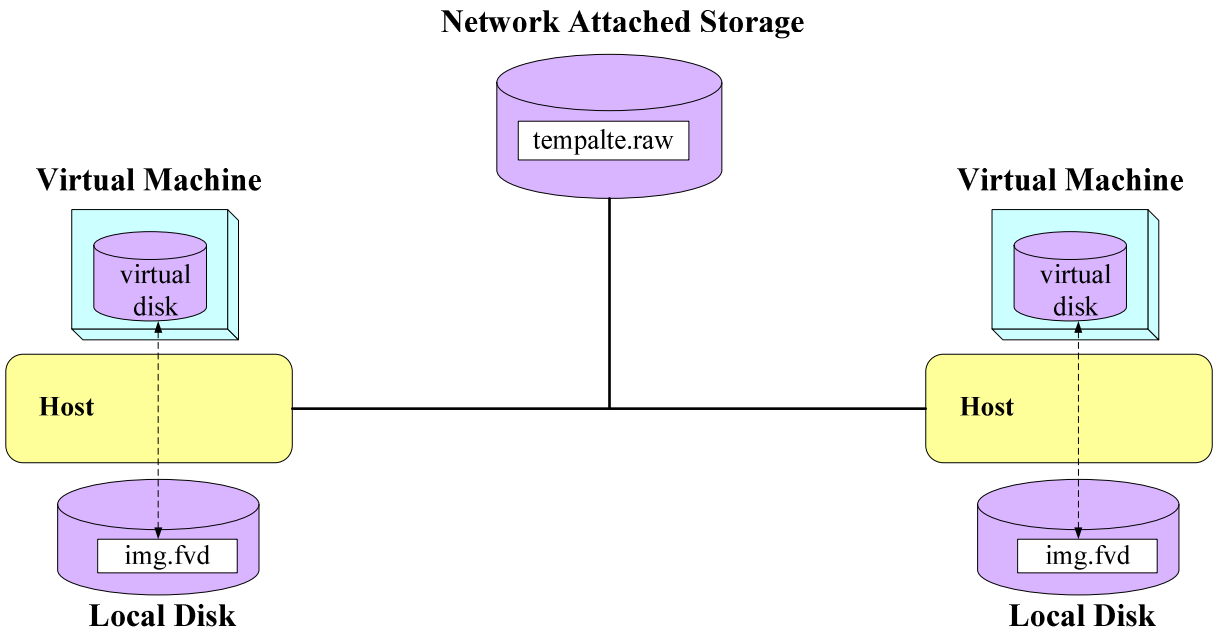
FVD: A High-Performance Virtual Machine Image Format for Cloud

Chunqiang (CQ) Tang

IBM T.J. Watson Research Center
ctang@us.ibm.com

June 2011

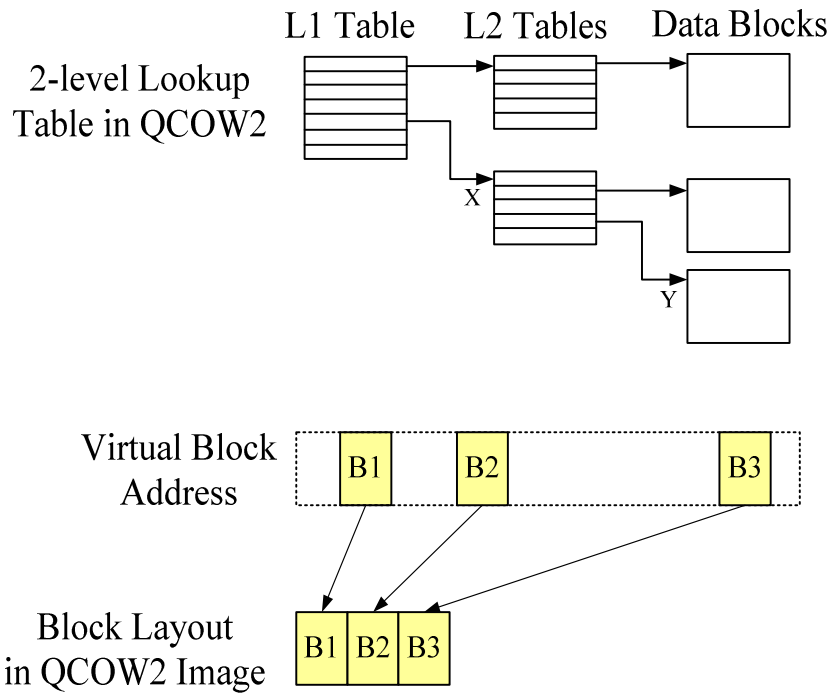
Virtual Disk Benefits from Copy-on-write, Copy-on-read, and Adaptive Prefetching



- A new VM's virtual disk is created as a copy-on-write image based on a shared, read-only image template
- Copy-on-read and adaptive prefetching avoid repeatedly read unmodified data from network attached storage

Challenges in Achieving High Performance for a Virtual Disk

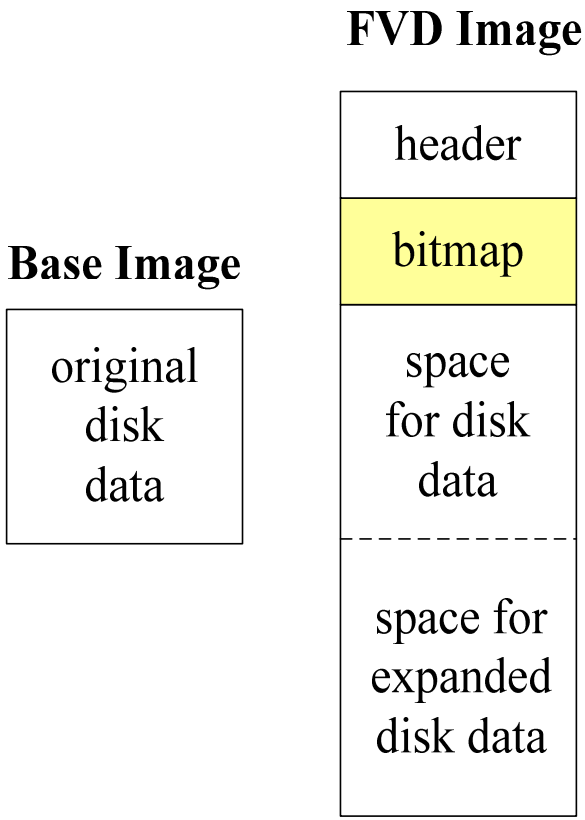
How QCOW2 works



Why a virtual disk is slower than a physical disk?

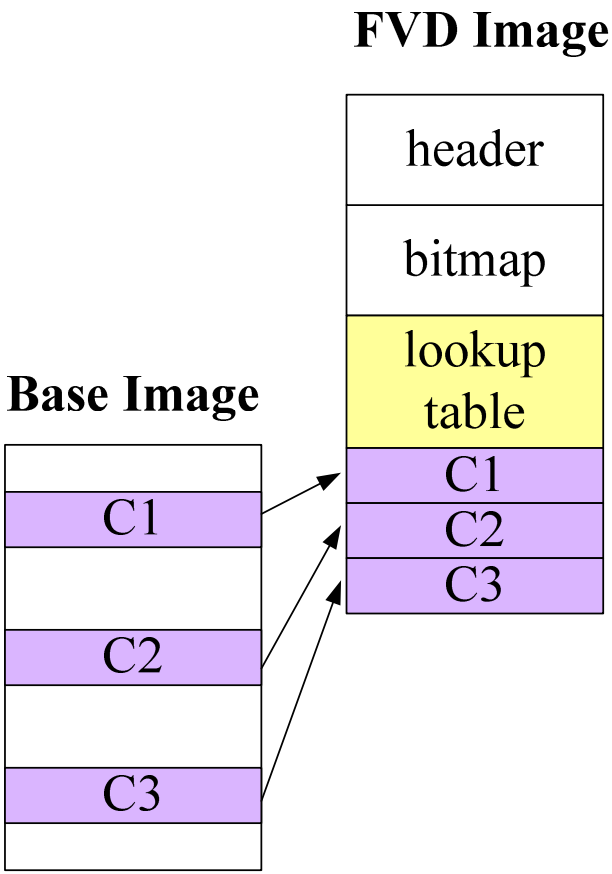
- Address translation destroies locality
- Overhead in reading metadata
- Overhead in writing metadata
- Overhead of a host file system
- Implementation inefficiency, e.g., blocking metadata access

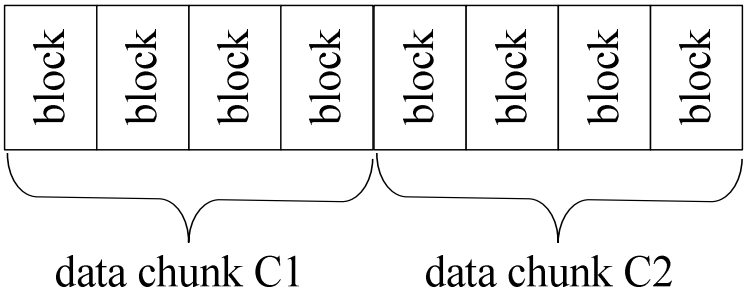
FVD Uses a Bitmap to Implement Copy-on-write, Copy-on-read, and Adaptive Prefetching



- No address translation and hence keeps data locality
- Small bitmap size allows easy caching (2MB for 1TB disk)
- Several techniques eliminate metadata writes in common cases
  - Fee write to expanded disk space
  - Free write to zero-filled blocks
  - Free copy-on-read and prefetching
  - Zero overhead once prefetching finishes
- **Benefit**: a CoW FVD image can be as efficient as a raw image
  - due to minimal metadata reads and writes, and no address translation

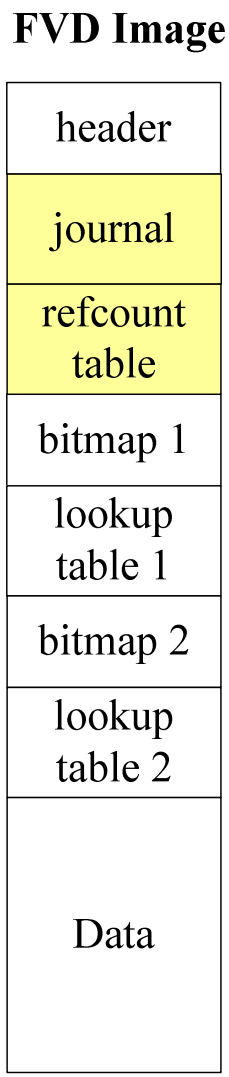
FVD Can Optionally Uses a Lookup Table to Support Compact Image





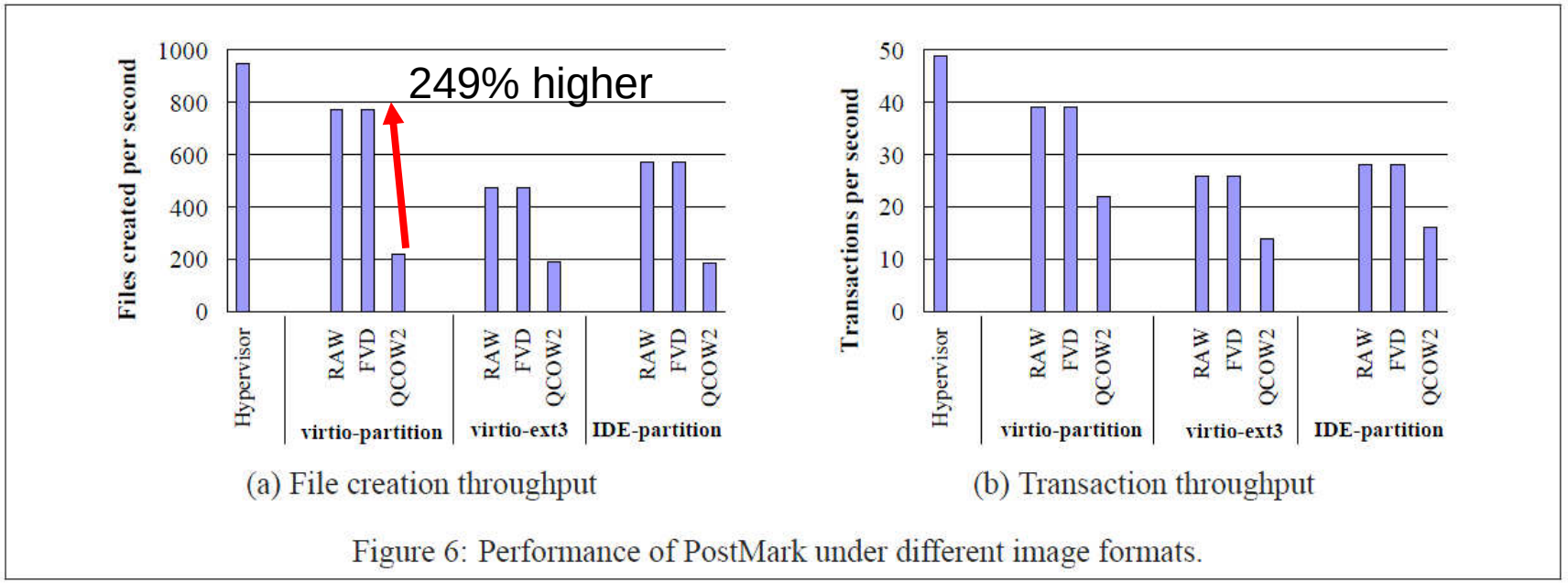
- A *chunk* consists of multiple *blocks*
- One entry of the lookup table maps the address of a chunk
- One bit in the bitmap indicates whether a block was written before
- **Benefit:** small metadata size
  - FVD 6MB vs. QCOW2 128MB for 1TB disk

Journal and Snapshot in FVD



- Journal allows efficient metadata updates
  - batching, sequential writes, concurrent writes
  - No journal cleaning overhead
- The refcount table supports efficientinternal snapshots
  - Creating/deleting a snapshot amounts to incrementing/decrementing reference counts
  - More efficient thant QCOW2 snapshot
    - The refcount table is never updated during normal execution of VM

Experimental Result



(a) File creation throughput

(b) Transaction throughput

Figure 6: Performance of PostMark under different image formats.

- FVD is implemented in KVM/QEMU 0.12.30
- The throughput of FVD is 249% higher than that of QCOW2 when using the PostMark benchmark to create files

Copy-on-read Helps Reduce Network Traffic

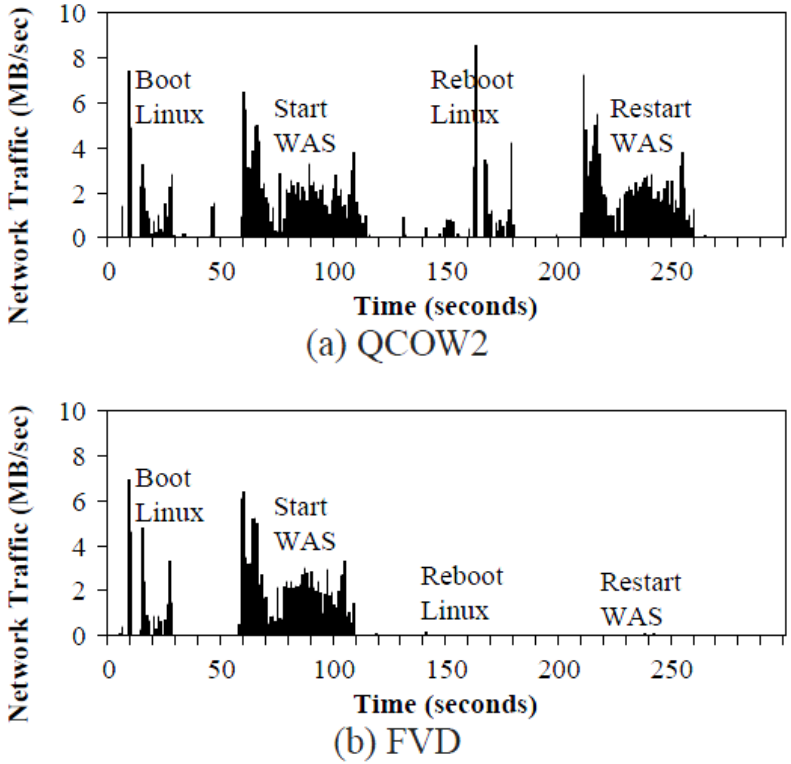


Figure 13: FVD's copy-on-read feature helps reduce network traffic and I/O load on the NAS server.

Summary of FVD

- FVD on-disk metadata
  - **bitmap** implements copy-on-write, copy-on-read, and adaptive prefetching
  - **lookup table** optionally implements compact image (i.e., address translation)
  - **journal** allows efficientmetadata updates
  - **refcount table** implements efficient internal snapshot
- Other Features of FVD
  - Storage thin provisioning without a host file system
  - Encryption
  - Fully asynchronous implementation
  - Autoated testing with deterministic replay for debugging
- Source code available at https://sites.google.com/site/tangchq/qemu-fvd
- Longer version of the paper available at https://sites.google.com/site/tangchq/publications