Low Bandwidth VM Migration Via Opportunistic Replay

Context of the paper

- Migration of VMs used in mobile computing
- Non-live migration
  - Suspend VM at the source host
  - Transfer state to destinatoin
  - Resume VM at destination

Problem

- VMs are large
  - Often tens of GB in size
- Mobile devices have low bandwidth
- Implies, excessive transfer time
- Goal: Reduce transfer time by reducing amount of information transferred.

Solution

- Opportunistic replay
  - Optimize migration between frequently visited hosts
- Key observation: Replay does not have to be perfect in order to be useful

Opportunistic replay

- Log user interactions with a VM, such as keystrokes, mouse clicks etc.
- Ship the log to the destination
- Replay the log on an identically configured VM which has the same initial state.

The catch?

- User interactions are not the only factors that affect the VM state.
  - Hardware interrupts, network packet processing, background system tasks, etc. all effect the final VM state.
- So replay will produce state that is close, but not identical to the desired final state.
- Solution: After replay, find out and ship only the differences in state between the source and the target.
- Worst-case: difference is large and we ship large amount of state.
- But correctness is never lost.

Challenge 1: Incomplete log capture

- Capturing all external stimuli requires
  - External interrupts
  - Data transfers
  - User input
- Can result in a very large log
  - Observed up to 1.2GB per day
- Also capturing all external events not possible in closed-source VMMs such as VMWare
- Solution: Just capture user-interactions
  - Most windowing systems provide interface to interpose logging code.
  - Will produce short, but incomplete, logs

Challenge 2: Non-deterministic Events

- Network access to websites with dynamic content
  - Replay of user click may return different content at the target.
  - E.g. Stock quotes, Parasitic content such as ads.
  - In most cases, users don't care about identical content.
- How about interrupts?
  - Ok, as long as it doesn't impact correctness or quality of user experience.

Challenge 3: Exactly once side effect

- Some events should not be replayed.
  - E.g. Stock purchase, online bank payments, sending emails.
- Solution 1: Block outbound network messages during replay.
- Solution 2: Detect exactly-once actions and skip them during replay. How?

Prototype implementation

- Xen 3.1
- Custom block driver that exports a virtual disk to VM.
  - Keeps track of dirty disk blocks and dirtying order
- rsync to synchronize (?) VM memory images.
- Xnee: X11 tool for record-replay.
- Doesn't address exactly-once problem.

Prototype (contd.)

- Initially both source and destination have identical copies of suspended VM.
- Replay speed may be faster than capture speed
  - Suppress the user think time.
- Modified disk chunks shipped to destination in the background while replay is in progress.
  - In reverse order (?)
- Differences in final state at source and destination computed
  - SHA-1 Hash
  - Applied on replayed image at target