Automating VM Installation Testing

Anthony Liguori – aliguori@us.ibm.com
Open Virtualization
IBM Linux Technology Center

August 31st 2007
Overview

- State of the union
- kvm-test
- How it works
- How we can share
Guest Support Status

- http://kvm.qumranet.com/kvmwiki/Guest_Support
Could it be better?

- More up-to-date status
- Consistent definition of “fail”
- Ability to see why guest's fail
- Separation of installation/boot
  - boot testing is easy
- Fill out the matrix
Installation is important

- If installation fails, for new users, it doesn't matter if the VM works.
- Installation tends to use a different kernel (often with a very different config).
- Very I/O intensive (win2k-hack).
- Interesting things like hardware probing.
- Difficult to automate using conventional techniques.
  - autoyast/kickstart are not the same thing.
kvm-test

- Build a test harness that interacts with a guest via VNC to simulate a user input
- Provide a mechanism to “macro-ize” installation
- Each installation script can be arbitrarily complex to meet arbitrarily complex installation routines
  - Use python
- Save forensics for later debugging
  - ffmpeg integration
  - future: use the monitor to frequently snapshot
kvm-test: challenges

• Determining when input is needed
• Dealing with randomly changing screen eliminates
  - animations
• Relative input devices
Determining input

- Use screen shots of “input screens” as the basis for determining when input is needed
- Currently use PPM images
- Use QEMU's screen grab capable
- Still needs to deal with random eliminates
Random Eliminates

- Use a mask to hide random portions of the screen
- Still not perfect, must use a fuzzy algorithm
- *Switch to terminal*
Relative Mice

- Find the mouse using a linear search
  - not as slow as it would seem
- Once you know the x,y coordinate the mouse is at, and you know you want to move to x1,y1, you can generate an input event
- But, cursors are accelerated so you can't just send an (x1-x,y1-y) event
  - Send [(x1-x)/2, (y1-y)/2]
  - Converges with O(log n) complexity
  - Fudge at the end
Complexity is important

- Number of iterations per second is limited by the VNC refresh rate
  - theoretically 15 fps
- We sample cursor positions 5 times a second
- If we moved (1,1) at a time, and had to move across the screen of size (1024,768), it could take over two minutes.
- Using our algorithm, we converge in two seconds.
How do we share?

- Show http://build.samba.org
Thoughts on publishing

- I don't like writing web stuff
- We have a publishing mechanism already – kvmwiki
- We could:
  - push results to /<UserName>/kvm-test/<git id>/Name
  - push an index page to /<UserName>/kvm-test/Name
  - Multiple UserName's can be used for different machines
  - Can attach things like dmesg, /proc/cpuinfo, etc.
  - Should we attach videos of boot in event of failure?
- MoinMoin has an xml-rpc interface so publishing is easy
- Perhaps we can even have something that checks all of the test results and provides a single summary
  - Problem: any user can publish results
Where it's at

- Code to be posted on as a subproject of gtk-vnc after this conference
  - Will include harnesses for Fedora and Ubuntu
  - Can we distribute Windows harnesses?
- I'll add publishing support in the very near future
  - We (IBM) will start running nightly on at least an LS21 and HS21
Areas to hack

- More guest support
  - It's real easy to add new guests
  - Eccentric ones like Plan9 are particularly useful
- May be interesting to include boot tests too
- Monitor integration
  - frequent checkpoints
  - if failure's detected, perhaps a register dump
The End

- Questions?