Multi-threading QEMU
or Ingo might be right.. sort of

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

Aug 2010
Ideal KVM Architecture

Design
• One thread per-VCPU
• Device models run concurrent in VCPU thread
• Long running operations run in additional device thread

Goals
• Maximize CPU affinity
• Minimize PIO/MMIO latency
QEMU/KVM Architecture

**Design**
- One thread per-VCPU
- One I/O thread
- All threads run in lock step
  protected by qemu_mutex

**Goals**
- Avoid rewriting QEMU
- Find a TCG-compat design
TCG Considerations

• Tiny Code Generator (TCG) is the emulator part of QEMU
• Cannot preserve atomicity of instructions
  – Due to design issues
  – Due to architectural issues (PPC vs. x86)
• QEMU's single thread design is ideal for TCG

• Device models are unlikely to ever run in parallel with TCG emulation
Evolving QEMU

- I/O thread gets our foot in the door
- Reduce granularity of locking
  - Push qemu_mutex out of kvm-all.c, exec.c, apic.c, ...
- Add locking to common infrastructure
  - Push qemu_mutex out of vl.c, async.c, block/*
- Start adding device specific threads

Easy, right?
## Two different worlds

<table>
<thead>
<tr>
<th>KVM</th>
<th>TCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Functionality over quality</td>
</tr>
<tr>
<td>Scalability</td>
<td>Performance doesn't matter</td>
</tr>
<tr>
<td>Reliability</td>
<td>All-in-one tool</td>
</tr>
<tr>
<td>External tooling</td>
<td></td>
</tr>
</tbody>
</table>

- We want to continue to share code
- Supporting multiple use-cases and architectures makes our code better
- We struggle to accommodate both worlds
Time for a change

- QEMU is bloated with lots of useful features
- We struggle to scale in every possible way

- VNC server
- virtual disk formats
- network interconnects
- generic transports
- multi-architecture device mode
- multi-architecture CPU emulation
- ....
libqemu-*.so

- Much of qemu would be better suited as libraries maintained as separate projects
- Let KVM develop a stand alone userspace that fits it's architecture model
- Multiple libraries to accommodate different architectures
  - With different emphasis on quality/features
- Continue to share code when it make sense

- Open QEMU code base to a wider audience
libqemu-block.so

- qemu-img is very popular outside of QEMU
- Many tools have developed over the years with a few making it into QEMU (qemu-nbd, qemu-io)
- These tools should be separate projects to allow other communities to contribute

- Patches on the list
libqemu-dm-pc.so

- Fork internal device models
  - Improve interfaces
  - Extensive unit tests
- As device models improve, we can replace the internal device models
- Experiment with radical to difficult problems
  - Migration
  - Versioning

- Some prototypes will be available soon
Other considerations

- Major concerns of split
  - KVM community ignores TCG; this is the point
  - QEMU has historically avoided dependencies

- Does fit general direction of QEMU
  - Single executable with pluggable CPU translation

- Split will not be successful without major changes
  - Test driven development
  - Rely more on external code
Questions

- Questions?