KVM Forum 2011

Anthony Liguori – aliguori@us.ibm.com

IBM Linux Technology Center

Aug 2011
QAPI, QCFG, and Code Gen

- QAPI is a framework to move QEMU to the next level of feature, function, and robustness.

- To fully understand QAPI, we need to understand what's holding us back...
Good Bones

- We've gained a lot of weight over the years in the form of features.
- Features aren't necessarily bad for you, but we have a particular appetite for salty, deep fried features.
- We're growing so fast, and are so popular, that we simply don't have time to exercise and eat healthy.
It's catching up with us

- Native KVM Tools is the doctor calling. We're on the verge of developing software type II diabetes
- Developing in QEMU “just isn't fun”

- But why?
qemu -hda linux.img -snapshot -net tap -net nic -usbdevice tablet
1) -hda, -hdb, -hdc, ...
2) -drive
3) -blockdev
4) drive_add
5) drive_del
6) blockdev_add  1) bdrv_register
7) blockdev_del  2) bdrv_open
8) query-block  3) whitelisting
Char Layer

1) -chardev OPTS
2) -serial URI
3) -monitor URI
4) -parallel URI
5) query-chardev

1) qemu_chr_open
2) no dynamic registration
Display Layer

Everything is open coded :-(

[Diagram showing the relationships between SDLState, DisplaySurface, DisplayAllocator, DisplayChangeListener, and DisplaySurface]
Network Layer

1) query-networks
2) -net
3) -netdev
4) netdev_add
5) netdev_del

1) net_client_init

Diagram:

- VLANState
  - NICState
    - VLANClientState
  - TAPState
    - VLANClientState
Monitor Layer

Open coded?
The Fat

- Each subsystem has added its own infrastructure
- Everyone needs:
  - Type serialization
  - Inheritance
  - Polymorphism
  - Object properties
  - Object enumeration
  - Factory interfaces
  - Mechanism to build an object graph
QAPI: Type Serialization

- Decompose serialization into two parts:
  1) Marshalers – for a given C type, call a method in the object for each primitive member in type.
  2) Transport – given a marshaler that can visit each primitive member in a C type, provide the translation of primitive types to arbitrary representations

- Visitor – see qapi/qapi-visit-core.h
- QmpOutputVisitor – see qapi/qmp-output-visitor.h and qapi/qmp-input-visitor.h
QEMU Object Model

• Standard Object Model supporting:
  - Inheritance; single inheritance model + interfaces
  - Polymorphism; class based polymorphism (no monkey patching)
  - Object properties; common base class that implements properties in terms of Visitors
  - Object enumeration; standard enumeration interface
  - Factory interface; standard factory interface with delayed construction
    • Construction properties are just normal properties
Plugs and Sockets

• Two special property types
  – Plug; a reference to a sub-object composed within the object.
  – Socket; a strongly typed pointer to an object

• Together, Plugs and Sockets allow for a directed acyclic graph
  – Can be used to model relationships between layers and within layers (i.e. busses).
From Here

- QAPI is already merged
  - QMP is being converted to use it
- QOM patches are on the ML
- Begin conversion with smaller layers (chardev)
  - Initial patches posted
- Build a plan to convert the other layers including the Device Layer
  - Can we incrementally morph qdev into a QOM type system?
QEMU 2.0

• Given a common infrastructure, we would have the following:
  - All backends and devices were created and manipulated by a set of about 6 commands
  - All object creation and manipulation could be done through QMP
  - Command line arguments are just QMP invocations (mostly just calls to above 6 commands)
  - Device model and backends are fully introspectable
  - Tree is fully modular (and type can be removed with no code change)
QEMU 2.0

- Current QMP and Command Line interface is purely legacy
- We could either (1) deprecate it and remove it in 2.0 or (2) move it entirely to a separate tool potentially written in a HIL

- Significant simplification of QEMU
- There will always be command line options or monitor commands that don't go through QOM, but it should be the exception.
Questions

- Questions, comments, flames?