Virtualizing Mac OS X
Leopard in a cage
About me

- Alexander Graf
- Working for SUSE Linux Products GmbH
  - Research on KVM / Qemu
  - SUSE Studio
Goal

• Make Mac OS X boot in KVM
Why

- Improve emulation accuracy
- Proof that it can be done
- Enable users to use Linux, whilst keeping Mac applications
The Challenge

- OS X bundled with Hardware
- Only supports Apple Hardware
- Is dongled with Apple Hardware
- Boots differently
What is an Intel Mac

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mac</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Core(2)Duo</td>
</tr>
<tr>
<td>ISA Bridge</td>
<td>ICH-7 LPC</td>
</tr>
<tr>
<td>HPET</td>
<td>yes</td>
</tr>
<tr>
<td>IDE</td>
<td>ICH-7</td>
</tr>
<tr>
<td>Additional</td>
<td>AppleSMC</td>
</tr>
<tr>
<td>Firmware</td>
<td>EFI</td>
</tr>
<tr>
<td>ACPI</td>
<td>Full-Blown</td>
</tr>
<tr>
<td>Feature</td>
<td>Qemu</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>CPU</td>
<td>Non-existing AMD64</td>
</tr>
<tr>
<td>ISA Bridge</td>
<td>PIIX3</td>
</tr>
<tr>
<td>HPET</td>
<td>no</td>
</tr>
<tr>
<td>IDE</td>
<td>PIIX3</td>
</tr>
<tr>
<td>Additional</td>
<td>-</td>
</tr>
<tr>
<td>Firmware</td>
<td>BIOS</td>
</tr>
<tr>
<td>ACPI</td>
<td>Rudimentary</td>
</tr>
</tbody>
</table>
# Qemu vs. Mac

<table>
<thead>
<tr>
<th></th>
<th>Mac</th>
<th>Qemu</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Core(2)Duo</td>
<td>Non-existing AMD64</td>
</tr>
<tr>
<td><strong>ISA Bridge</strong></td>
<td>ICH-7 LPC</td>
<td>PIIX3</td>
</tr>
<tr>
<td><strong>HPET</strong></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>IDE</strong></td>
<td>ICH-7</td>
<td>PIIX3</td>
</tr>
<tr>
<td><strong>Additional</strong></td>
<td>AppleSMC</td>
<td>-</td>
</tr>
<tr>
<td><strong>Firmware</strong></td>
<td>EFI</td>
<td>BIOS</td>
</tr>
<tr>
<td><strong>ACPI</strong></td>
<td>Full-Blown</td>
<td>Rudimentary</td>
</tr>
</tbody>
</table>
How

- Emulate devices that Mac OS X supports
- Provide a way to boot Mac OS X
- Pass through the dongle key
CPU

- Checks for GenuineIntel and certain CPU Families
- Requires
  - SSE2 for 32-bit
  - SSE3 for PPC emulation
  - SSSE3 for 64-bit
#define CPUID_VID_INTEL  "GenuineIntel"
#define CPUID_VID_AMD   "AuthenticAMD"

void
cpuid_set_info(void)
{
    bzero((void *)&cpuid_cpu_info, sizeof(cpuid_cpu_info));

    cpuid_set_generic_info(&cpuid_cpu_info);

    /* verify we are running on a supported CPU */
    if ((strncmp(CPUID_VID_INTEL, cpuid_cpu_info.cpuid_vendor,
                 min(strlen(CPUID_STRING_UNKNOWN) + 1,
                     sizeof(cpuid_cpu_info.cpuid_vendor))) ||
         (cpuid_cpu_info.cpuid_family != 6) ||
         (cpuid_cpu_info.cpuid_model < 13))
        panic("Unsupported CPU");

    cpuid_cpu_info.cpuid_cpu_type = CPU_TYPE_X86;
    cpuid_cpu_info.cpuid_cpu_subtype = CPU_SUBTYPE_X86_ARCH1;

    cpuid_set_cache_info(&cpuid_cpu_info);

    cpuid_cpu_info.cpuid_model_string = ""; /* deprecated */
}
ICH7

- Accesses PCI config space registers for LPC unconditionally
- Does not detect older IDE-controllers
- Accesses HPET unconditionally
pmCPU.h

#define cfgAadr 0xCF8
#define cfgDat 0xCFC
#define lpcCfg (0x80000000 | (0 << 16) | (31 << 11) | (0 << 8))

hpct.c

/*
 * Map the RCBA area.
 */
static void
map_rcbaArea(void)
{
    /*
     * Get RCBA area physical address and map it
     */
    outl(cfgAadr, lpcCfg | (0xF0 & 0xFC));
    rcbaAreap = inl(cfgDat | (0xF0 & 0x03));
    rcbaArea = io_map_spec(rcbaAreap & -4096, PAGE_SIZE * 4, VM_WIMG_I0);
    kprintf("RCBA: vaddr = %08X, paddr = %08X\n", rcbaArea, rcbaAreap);
}

/*
 * Is the HPET memory already enabled?
 * If not, set address and enable.
 */
xmod = (uint32_t *)(rcbaArea + 0x3404); /* Point to the HPTC */
uint32_t hptc = *xmod; /* Get HPET config */
DBG("current RCBA.HPTC: %08X\n", *xmod);
if(!hptc & hptcAE)) {
    DBG("HPET memory is not enabled, 
    enabling and assigning to 0xFED00000 (hope that's ok)\n");
    *xmod = (hptc & ~3) | hptcAE;
}
ICH7

hpeth.h

#define hpetAddr 0xFED00000

hpetc.c

/*
 * Get physical address of HPET and map it.
 */

hpetArea = hpetAddr | ((hptc & 3) << 12);
hpetArea = io_map_spec(hpetArea & 0x4096, PAGE_SIZE * 4, VM_WIMG_I0);
kprintf("HPET: vaddr = %08X, paddr = %08X\n", hpetArea, hpetArea);
EFI

• EFI Implementation for Qemu exists
• Not up-to-date
• No support for HFS+
• BIOS bootloader for Mac OS X exists
• Convenient
• Patched version by David Elliot to run new kernels
AppleSMC

- System Management Chip for
  - Fan Control
  - Backlight Control
  - Dongle key storage
- Easy to emulate
- Key must be given by user
What works

- Mac OS X
- Rosetta
- 64-Bit
- Network
- USB
What does not work

• Graphic glitches
• Sound
• In-kernel APIC
• About This Mac
• Keynote
License Issues

A. Single Use. This License allows you to install, use and run one (1) copy of the Apple Software on a single Apple-labeled computer at a time. You agree not to install, use or run the Apple Software on any non-Apple-labeled computer, or to enable others to do so. This License does not allow the Apple Software to exist on more than one computer at a time, and you may not make the Apple Software available over a network where it could be used by multiple computers at the same time.
Where to get it

- http://alex.csgraf.de/qemu/osxpatches.tar.bz2
Questions?