Panorama des systèmes de virtualisation :
des centres de calcul
aux systèmes embarqués temps-réel

François Armand

Novembre 2008

Francois.Armand@Virtuallogix.com
Agenda (1/2)

- Use cases
- Multi-core challenges
- Virtualization Taxonomy
  - System level versus Process level virtualization
  - Native versus Hosted Virtualization
  - Transparent virtualization versus para-virtualization
- Embedded & Real-Time Virtualization Requirements
- Hardware Evolution
Agenda (2/2)

Desktop, Data Center solutions

Survey of Embedded solutions:
- Hybrid OS’s, Co-running OS’s
- Asymmetric Multi-Processing
- Micro-kernel based solutions
- Embedded Real-Time Hypervisors

Miscellaneous
- Tools
- High-Availability
- Standardization
Use Case: Data Centers

- Consolidate workloads running on independent physical machines on a single server, while maintaining independence
- “Stop Machine sprawl” 😊
- Split (virtualize) physical servers into “smaller” Virtual Machines
- Allocate VM dynamically
Use Case: Workstation

- Similar to data centers
- Run more than a single environment at once on a single machine
Use Case: Mobile Handsets

- Run Linux applications on baseband processor
- Re-use existing modem software stack with its RTOS (no changes)
- Support of Linux at a minimal development cost
- Operating System independence for future evolutions
- See also NTT Docomo OSTI
Virtualization in High-Throughput Network Equipment

Example Use-Case: Blade Consolidation
Use case: Virtualization in Multimedia Devices

Reduction of BOM for high-volume low-end products
- No need for a General Purpose Processor
  • ~ 20 to 25 % BOM reduction
- Run Linux together with OS supporting Codecs on a single TI DSP
- Leverage Linux environment
- Reuse existing DSP software
Other Use Cases

- Instrumentation, Automation
  - Run a RTOS and a GPOS for Graphical Interface

- Points of Sales
  - Run the UI and the secure transaction environment on the same processor

- Mil / Aero
  - Run securely isolated / certified environments simultaneously

- More…
Agenda (1/2)

- Use cases
- Multi-core challenges
- Virtualization Taxonomy
  - System level versus Process level virtualization
  - Native versus Hosted Virtualization
  - Transparent virtualization versus para-virtualization
- Embedded & Real-Time Virtualization Requirements
- Hardware Evolution
Multi-Core Challenges

- Increased power no longer provided by speed increase, but through higher density and multi-core chips

- Many legacy RTOS have been designed based on uni-processor assumption
  - Scaling on multi-core requires time
  - Has been (is still) a pain for GPOS (scalability issues for OS and Applications)

- Legacy RT applications designed for uni-processor as well… Need to move to true multi-thread with true parallelism….
  - Even more difficult than for OS’s
Multi-Core Challenges

- Virtualization enables to run multiple instances of a [legacy real-time] software loads –designed for uni-processor- simultaneously on a multi-core chip

- Possibility to run in uni-processor Virtual Machines, assigned to a physical CPU

- For GPOS: run n-way SMP instances on a larger machine (ex: 4 4-way Linux on a 16 cores machine)

- No modifications of legacy code, No risk to break code

- Scalability provided by Virtualization layer
Agenda (1/2)

- Use cases
- Multi-core challenges
- Virtualization Taxonomy
  - System level, Process level virtualization, OS virtualization
  - Native versus Hosted Virtualization
  - Transparent virtualization versus para-virtualization
- Embedded & Real-Time Virtualization Requirements
- Hardware Evolution
Virtualization is getting interest!

[Source: Google Trends on November 21, 2008]
Virtualization? Oh, Virtualization!

Virtual Networking?   Intel VT?
VMware?   Virtual Server?
Java, JVM?   Transitive QuickTransit?
AMD SVM?   Application Virtualization?
SIMICS?   Platespin?
TransMeta Crusoe?   Pascal Pcode?
Dynamo?   Virtual Solutions?
Virtual Reality?   VirtualPC?
FX!32?   Virtual Storage?
Different classes of virtualization

- Aggregation
  - Make N resources appear as 1 (clusters / grids)

- Partition / Replication:
  - Make a resource appear as N

- Translation (Emulation)
  - Make X appear as Y (*sometimes X is identical to Y*)
  - May be combined with “partition”

Mostly interested in “partition”
Virtual Machine Interfaces

- Systems built of: hardware, OS, applications
- 2 main interfaces: ISA (hardware), ABI (OS)

System level VM provide an ISA interface

Process level VM provide an ABI interface
Taxonomy (derived from E. Smith & Nair)

System Virtualization

- System VMs
  - (Same ISA)
    - Hardware Virtualization
      - Classic OS VM
        - Native, Type I
          - Paravirtualization
            - Paravirtualized Xen, VLX
            - HW Assisted Xen, VLX
          - Full/Native Virtualization
            - Dynamic Binary Translation
              - Vmware ESX
            - Transparent

  - (Possibly different ISA)
    - Hardware Emulation
      - Whole System Emulation
        - Bochs, QEMU
          - Hosted, Type II
            - VMware WS, KVM

VMs

Process Level (ABI)

- Process VMs
  - (Same ISA)
    - Multiprogrammed Systems
      - Dynamic Translators
        - (=OS)
          - ISA & OS Translator
            - FX!32
          - (#OS)
            - ISA & ABI Translator
              - WABI, WINE
  - (Possibly different ISA)
    - Multitask OS
      - OS Virtualization
        - Virtual Servers
          - Virtuozzo, Solaris Zones
        - Transparent
          - Dynamic Binary Translation
            - Vmware ESX
        - HW Assisted
          - Vmware ESX
          - High Level Language
            - Java
OS Virtualization or Virtual Servers

Examples:
- Solaris 10 Containers
- Linux-VServer, FreeBSD jails
- Virtuozzo / OpenVZ

Pro’s:
- Supports distribution heterogeneity
- Lightweight:
  - low memory and performance overhead
- Scales to many instances

Con’s:
- Single OS instance (common point of failure)
- Modified OS, intrusive: has to follow OS evolution
- Does not support OS heterogeneity (Hence no RTOS / GPOS combination)
Classic OS VMs

Enable to run multiple –independent- OS’s simultaneously on the same processor (guest os), each in its own “Virtual Machine”

Two main approaches:

- Native VM’s: Introduce a software layer between the hardware and the OS: Virtual Machine Monitor (VMM) or “Bare metal” Hypervisor

- Hosted VM’s: require a Host OS to start first
Classic OS VMs

Issues:

- Run multiple OS’s
- OS designed assuming it is the only software controlling the physical resources of the machine.
- Need to detect and resolve conflicts such as masking interrupts, initiating an I/O, or programming the MMU by providing the expected behavior within the Virtual Machine, not necessarily on the physical level.
- In brief: detect sensitive instructions and fake them.
  - User mode to trap such instruction upon execution
  - Modify OS ahead of time
  - Hardware support
Classic OS VM’s

Goal:

– Run the binary guest OS in user mode though it’s been written to run in supervisor mode.

Means:

– Transparent Virtualization: *(full or native)*
  
  • No modification of the OS image
  
  • Fully Virtualizable Processors (VT-x, AMD-V, IBM PPC)
  
  • Dynamic Binary Translation (VMware)

– Para-virtualization:
  
  • Modification of some of the OS HAL source files
  
  *(can be seen as a port to a new processor very similar to the real one).*
Fully Virtualizable Processors

Issue:

- Is there any instructions whose behavior differs or should differ in these 2 modes?

Sensitive Instructions:

- Trap to supervisor mode when executed from user mode: OK
  - Ex: cli, sti (Intel x86) trap when executed from user-mode

- No-op: not OK:
  - Ex: POPF (Intel x86): Interrupt-enable flag remains unaffected in user mode

- Get system / hardware status: not OK
  - Ex: Read CR3 (Intel x86) would return true physical info, instead of virtualized info.
Fully Virtualizable Processors
Hardware assisted virtualization

Existing Solutions

- Existing solutions utilize concepts such as binary patching or para-virtualization
- Complex, Source code...

Intel Virtualization Technology (VT-x)

- VT-enabled Intel CPUs integrate a new execution mode, enabled by an instruction set and control structure
- Allows for two new operating modes and a set of hardware-based triggers to switch between them

Diagram:
- Ring 0: Virtual Machine Monitor
- Ring 1: Binary Patching
- Ring 3: Application
- Unmodified OS
- Paravirtualized OS
- Root Mode
- Non Root Mode

Virtual Machine Monitor
Hardware
Dynamic Binary Translation

- Non Hardware assisted virtualization: (x86,…)
- Run unmodified binary guest in a less-privileged mode (ex: ring 3 instead of ring 0 on x86)
- Dynamic binary translation (e.g.: VMware on x86)
  - VMM dynamically “re-writes” privileged instructions which would be silently executed in user mode.
  - On demand, cached
  - Memory consumption impact
  - Timing determinism impact
- Transparent but complex solution
  - complexity resides in the VMM
Paravirtualization

- Modify the HAL of the guest OS
- For secure isolation purposes:
  - Rely on memory protection provided by hardware
  - Run guest OS in less privileged mode
    - Ex: ring 1 instead of ring 0 on x86
    - Ex: user mode instead of supervisor mode on ARM
- Better performance level than Dynamic Binary Translation, but intrusive.
  - Solution of choice when no hardware support (PPC, ARM) and OS modifications are possible
Agenda (1/2)

→ Use cases
→ Multi-core challenges
→ Virtualization Taxonomy
  – System level, Process level virtualization, OS virtualization
  – Native versus Hosted Virtualization
  – Transparent virtualization versus para-virtualization
→ Virtualization Requirements
→ Hardware Evolution
### Virtualization Requirements

<table>
<thead>
<tr>
<th></th>
<th>Data Center</th>
<th>Network Equipment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPOS support</td>
<td><strong>YES</strong></td>
<td><em>Some</em></td>
</tr>
<tr>
<td>RTOS support</td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>N x RTOS on multicores</td>
<td><strong>NO</strong></td>
<td><em>Some</em></td>
</tr>
<tr>
<td>GPOS &amp; RTOS on same core</td>
<td><strong>NO</strong></td>
<td><em>Some</em></td>
</tr>
<tr>
<td>Isolation / Performance Trade-off</td>
<td><strong>NO</strong></td>
<td><em>Some</em></td>
</tr>
<tr>
<td>Memory constraints</td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Performance</td>
<td><em>I/O Throughput</em></td>
<td><em>Timing, Latency, Determinism</em></td>
</tr>
<tr>
<td>Communication channels</td>
<td>Virtual Network</td>
<td>Depend on applications</td>
</tr>
<tr>
<td>Device Drivers</td>
<td>Virtual Network, Virtual Disks</td>
<td>Many, physical &amp; virtual</td>
</tr>
<tr>
<td>Native Device Driver</td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Dedicated Devices</td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Shared Devices</td>
<td><em>Some</em></td>
<td><em>Some</em></td>
</tr>
</tbody>
</table>
Some Virtualization Needs for Embedded & RT Systems

_ENABLE TO RUN SEVERAL RTOS AND APPLICATIONS DESIGNED FOR UNI-PROCESSOR UNCHANGED ON A MULTI-CORE PROCESSOR.

- Avoid modifications, debug, validation of OS and applications
- Shorten Time To Market

CONSOLIDATION OF (LEGACY + LINUX) SYSTEMS AND APPLICATIONS ALREADY TIGHTLY COOPERATING TO DELIVER A SERVICE.

ALLOWS HARDWARE CONSOLIDATION

- Lower Bill Of Material
- Lower Total Cost of Ownership
Some Virtualization Needs for Embedded & RT Systems

- Rather closed and static configurations
- Real-Time (hard / soft) important,
- I/O throughput may matter
- Usually, much more devices than just network and disks, and devices tight to applications
- Memory constraints, boot time constraints
- Underlying HW customized for applications dependence between apps and drivers, e.g. network interfaces in NI equipment.
Agenda (1/2)

➔ Use cases
➔ Multi-core challenges
➔ Virtualization Taxonomy
  – System level, Process level virtualization, OS virtualization
  – Native versus Hosted Virtualization
  – Transparent virtualization versus para-virtualization
➔ Embedded & Real-Time Virtualization Requirements
➔ Hardware Evolution
Hardware Evolution

**X86**
- Multi-core,
- Hardware assisted virtualization:
  - `vt-x`, `vt-i`, `svm`,
  - I/O’s: `vt-d`, `iommu`, (transparent virtualized DMA)
  - PCI: SRIOV, MRIOV, (VMDQ)

**Power Architecture**
- Roadmap to multi-core for embedded systems
- 3 rings: user, supervisor, hypervisor, not implemented in all processors

**ARM**
- Roadmap to multi-core
- No current support for virtualization
- However, TrustZone permits to isolate 2 environments

**SPARC**
- Niagara: multi-core / multi-threaded
- Hardware/Firmware support for para-virtualization
Agenda (2/2)

- Desktop, Data Center solutions

- Survey of Embedded solutions:
  - Hybrid OS’s, Co-running OS’s
  - Asymmetric Multi-Processing
  - Micro-kernel based solutions
  - Embedded Real-Time Hypervisors

- Miscellaneous
  - Tools
  - High-Availability
  - Standardization
Hosted Virtualization:

- Start a host OS first
- Dynamically extends the OS with kernel module and applications providing virtualization
- Virtualization relies on Host OS services
- Typically binary Guest OS supported (Transparent Virtualization)
- No isolation between Host OS and VMM

Examples:
- VMware WS, GSX, SVista, Parallels, VirtualBox
- KVM, User-Mode Linux
**KVM Architecture**

- Guest I/O are trapped by KVM and redirected to QEMU.

- VM’s are run and scheduled as Linux Host processes.
Native Virtualization

➔ Start a hypervisor / VMM on bare metal,

➔ Then, typically start a “control domain” (Xen) or a Service Console (VMware)

➔ Can then –dynamically- create VM’s

➔ Device Virtualization may be provided

  – By VMM (VMware ESX)

  – By other VM’s (Xen)
Bare Metal Hypervisors: XEN

Hardware Assisted Transparent Virtualization (VT)

Ring 3
- Control Plane Software
- User Software
- User Software
- User Software

Ring 0
- GuestOS XenoLinux
- GuestOS XenoLinux
- GuestOS XenoLinux
- GuestOS XenoLinux

Mode Root
- Domain0 control interface
- Virtual x86 CPU
- Virtual memory
- Virtual devices

H/W (phy mem, x86 CPU)

Para-Virtualization Pre-VT x86

Ring 3
- Control Plane Software
- User Software
- User Software
- User Software

Ring 1
- GuestOS XenoLinux
- GuestOS XenoLinux
- GuestOS XenoLinux
- GuestOS XenoLinux

Ring 0
- Domain0 control interface
- Virtual x86 CPU
- Virtual memory
- Virtual devices

H/W (phy mem, x86 CPU)
**XEN:**

- Mostly dedicated to Data Centers: x86, Itanium and IBM PPC
  - Paravirtualization or HW assisted virtualization eliminates the need for dynamic translation
  - Supports Linux, FreeBSD, Plan9
  - Supports Windows on Intel VT, AMD SVM.
  - Tools for data center management (migration, provisioning..)

- Limited support for other processors
  - ARM in progress

- No support for embedded configurations
  - All VM’s created from Dom 0
  - Need to wait for complete initialization of Dom 0
Dynamic Binary Translation: VMware ESX

Ring 0

Ring 3
Bare-Metal Hypervisors: ESX

- Dedicated to Data Centers (x86 / Itanium only)
- Dynamic Binary Translation (OS unmodified in ring 3)
  - So far, even on Intel VT, AMD SVM
- “Big”:
  - Includes Virtual Memory Management, File system (VMFS), Network stack, drivers…
- Memory Management:
  - Physical Memory may be over committed (paging out guest OS’s)
  - Dynamic content based page sharing, Memory ballooning
- OS dynamically scheduled on processors
  - Some way to constrain scheduling (max, min, shares)
Live Migration of VM

Migrate a running VM from one physical server to another one without loss of service (minimal down-time): *Load Balancing, Preventive maintenance*
Virtual Appliance

- Deliver Application with its containing Virtual Machine
  - No OS version issue anymore
  - No compatibility of libraries, tools with other applications
  - Application already installed and partly configured
  - VMware has a large directory of Virtual Appliances for VMPlayer
  - Get rid of physical devices dependencies, deal only with virtual devices
New ways to look at applications

Virtual Appliances

Virtualized Applications

Pushed/pulled from server

V. M.
Appli.
OS

VMM
Server

UI/browser
OS
Computer

Provided by VMware
Promoted by VMware

Provided by Citrix

Virtual App.
Appli.
OS

Virtual App.
Appli.
OS

Virtual Logix Confidential
Evolution of Virtual Appliances

- Access to devices: virtualized
  - Either through dedicated VM(Xen), or by VMM(ESX)
Agenda (2/2)

- Desktop, Data Center solutions

- Survey of Embedded solutions:
  - Hybrid OS’s, Co-running OS’s
  - Asymmetric Multi-Processing
  - Micro-kernel based solutions
  - Embedded Real-Time Hypervisors

- Miscellaneous
  - Tools
  - High-Availability
  - Standardization
Hybrid OS: RTOS as a RichOS Plug-in

- RTOS installed as a module, similar to hosted virtualization

Examples:
- Linux: RTLinux, RTAI, ADEOS
- Windows: Kuka, Tenasys (x86 pre VT-x)

Pro’s:
- 2 worlds on single processor
- Integrated worlds

Con’s:
- Non legacy RTOS or specific one, no legacy sw reuse
- In practice, limited to 2 OS’s
- No independence / isolation: security, availability, (GPL?)
- No immediate way to dedicate resources
- Re-write RT drivers or use non RT ones
- Need to wait for Linux / Windows boot before getting RT applications
Dual Core AMP approach

Example VxWorks / Linux on MPC8461D
- Can be VxWorks/VxWorks or Linux/Linux as well

Pro’s:
- Simple
- Software stacks independence

Con’s:
- Hard to scale to other OS
- Start one OS first, then 2nd OS
- Weak / No Isolation
  - Depends on HW features
- Requires hardware dependent OS modifications
  - Somehow similar to paravirtualization
- Limited to 1 OS per core
- Does not solve device access, communication issues
Agenda (2/2)

- Desktop, Data Center solutions

- Survey of Embedded solutions:
  - Hybrid OS’s, Co-running OS’s
  - Asymmetric Multi-Processing
  - Micro-kernel based solutions
  - Embedded Real-Time Hypervisors (VLX)

- Miscellaneous
  - Tools
  - High-Availability
  - Standardization
**Microkernels** (e.g. L4, Sysgo)

- Microkernel provides:
  - Thread creation, scheduling,
  - Memory management
  - Independence from OS’s

- Pro’s:
  - May run more than one RichOS
  - Independent from OS vendors
  - Guest OS in user space (isolation)

- Con’s:
  - Microkernel API either not explicitly dedicated for RT, or not legacy
  - Need RTOS and RichOS adaptations
  - Performance and memory footprint overhead
Agenda (2/2)

- Desktop, Data Center solutions

- Survey of Embedded solutions:
  - Hybrid OS’s, Co-running OS’s
  - Asymmetric Multi-Processing
  - Micro-kernel based solutions
  - Embedded Real-Time Hypervisors (VLX)

- Miscellaneous
  - Tools
  - High-Availability
  - Standardization
VirtualLogix VLX
Real-Time Virtualization™ Software

*Combines differing operating system application environments into a single device*

- RTOS & RichOS instances run in their own OS partition
- Real-time performance guarantees
- Critical and legacy tasks co-exist with RichOS applications & native device drivers
- HW resources dedicated or shared between several OS partitions with QoS guarantees
- OS partitions securely isolated from each other communicate thru secure channels

Available Today: Intel® IA, ARM, Power, TI DSP architectures
Linux, commercial and customer internal RTOSs
**VLX NI – High Level Features**

- **Run mix and match combinations of RTOS, Open OS and their applications**
- **Runs unmodified OS**
- **Strict isolation between guest OSs**
- **One core can be dedicated to one OS**
- **Scalable across many cores**
- **Multiple OS can run on one core.**

**VirtualLogix VLX Virtual Machine Monitor**

- **RTOS #1**
- **Linux (OS #2)**
- **RTOS #3**
- **Operating System #M**

**Intel VT Enabled Platform**

- **CPU Core #1**
- **CPU Core #2**
- **CPU Core #3**
- **CPU Core N**

**Devices**

- **Ethernet MAC**
- **Hard Disk Drive**
- **Ethernet MAC**
- **Device N**

**Guests use real device drivers**

**Guests share Devices**
Access to Devices

- Dedicated devices managed by native drivers in guest OS’s
  - Multiple VM’s can have dedicated devices
  - No bottleneck

- Virtual devices exported by one VM to others
  - Virtual Block Devices
  - Virtual Ethernet,…
VirtualLogix VLX Virtual Machine Monitor

- **Intel VT Enabled Platform**
- **Advanced VM schedulers**
  - Real-time guarantees
  - CPU resource reservation
  - Allow flexible multi-OS designs
- **Inter-OS communication framework**
  - Communication, data sharing and coordination between guest OS
  - Zero-copy shared memory based
  - High level services
- **Lightweight VMM**
  - Performance optimized
  - Partitions CPUs, memory and devices to isolate one guest OS from another
  - Virtualizes core platform resources (PIC, Timer, RTC, UART)
  - No Host-OS or Domain-0 OS required
- **The VLX Virtual MMU supports multiple modes**
  - Provides strict “sandboxing” of guest OS
  - Allow performance optimization with respect to number of system tasks, size, and period

**All CPU cores are available for “real” work**
Agenda (2/2)

Survey of some solutions:
- Hybrid OS’s, Co-running OS’s
- Asymmetric Multi-Processing
- Desktop, Data Center solutions
- Micro-kernel based solutions
- Embedded Real-Time Hypervisors

Miscellaneous
- Tools
- High-Availability
- Standardization
Agenda (2/2)

- Desktop, Data Center solutions

- Survey of Embedded solutions:
  - Hybrid OS’s, Co-running OS’s
  - Asymmetric Multi-Processing
  - Micro-kernel based solutions
  - Embedded Real-Time Hypervisors (VLX)

- Miscellaneous
  - Tools
  - High-Availability
  - Standardization
Performance Monitoring Tools

- **Host Graphical Tools**
  - Eclipse 3.1.2
  - Build/Config
  - Monitoring
    - CPU usage
    - Context switch

- **Nucleus PLUS**
  - Cyclic benchmark program

- **Linux**
  - Qtopia:
    - MP3 Player
    - Image viewer
  - Administration
Virtualization and HA

Traditional HA

- 1 to1 dependency of Appli / HA / OS / HW
- 1 application per processing blade
- Redundancy & HA managed at blade level

New HA

- Virtualization introduces new resource entity: VMs
- HA dependency chain is modified

- VMs mngnt by HA enhance platform availability
VirtualLogix vHA

- Support HPI, AIS subset
- Support Virtual Machines
Standardization

- Mostly focused on IT / Data Centers
  - DMTF has published OVF 1.0
  - Open Virtual Machine Format: XML description of a VM content to recreate it on a different hypervisor

- Scope Alliance has published a set of requirements

- Service Availability Forum latest specifications cover virtualization

- Multi-core Association has a “Hypervisor” WG

- Various interfaces in Linux:
  - Para-virtops,
  - virtio
MERCI!