Acknowledgments

- Alex Williamson (HP)  
  - for being extremely helpful and listening to my rants
- Greg Edwards (SGI)  
  - for initial Xen Altix work, providing me with a base to start from
- The Xen/ia64 community  
  - for providing patches and assistance  
  - if trying to list everyone by name, someone will be forgotten
Agenda

• Xen's current system architecture
• What makes a NUMA system, such as the Altix, different from a regular PC?
• What has been addressed?
• What comes next?
• Wishes for the future
• Silly and not so silly questions
Xen's current system architecture

- Keep it simple
- Only direct support for most basic I/O:
  - VGA
  - Serial
  - Console controllers (such as Altix L1)
  - Rely on dom0 to do the heavy lifting
- Very basic scheduling support for virtual CPUs
- Simple memory allocator based on linear bitmap
- Everything sits on a flat memory bus ..... 
  - Xen/ia64 ... everything was a DIG box
Altix Characteristics

- Large number of nodes, CPUs (1024), I/O slots
- One Shub chip per node
  - Connects to CPUs, memory, I/O, and NUMAlink
- No regular PC style basic I/O:
  - VGA
  - Serial 16c550
- No flat system bus
  - System topology needed to access any PCI device, including VGA
- System console accessible via SAL call
- IPI and TLB flush handled via Shub register write
Altix Characteristics #2

• ITC (TSC on x86) not synchronized across nodes!
• Highly sparse memory layout, Shub addressing:
  
  - AS: Address Space, 0b11 == Cacheable (normal)
  - Actual physical memory map:
    - Node 0: 0x003000000000-0x004000000000
    - Node 1: 0x00b000000000-0x00c000000000
    - Node 2: 0x013000000000-0x014000000000
    - Node 3: 0x01b000000000-0x01c000000000
    - Node 4: .....
Issues addressed

• Machine vector support to have generic Xen kernel supporting multiple machine types
  – IPI, TLB, I/O register access
  – No DMA API support yet

• Xen kernel relocation support
  – Altix does not have physical memory at 0x4000000
  – Still a couple of places missing that needs to be chased down

• Altix topology support
  – Required for Shub chip register access: I/O, IPI, TLB flushing etc.

• Polling Altix console driver (output only)
Issues addressed #2

- Replaced a number of hard-coded DIG-isms with runtime checks
- Corrected handling of memory attributes
  - Regions of EFI mem-map have slightly different attributes depending on the system vendor
- Remap I/O port ranges from high physical address to lower metaphysical address
  - Altix I/O port range starting at 0x2000000000000000
  - Xen uses page tables to map all regions to metaphysical space, these only provide a limited address space.
- PCI device discovery via ACPI
  - Interrupts are not being delivered
Memory allocation

• Xen's bootmem allocator is based on linear bitmap mapping each page in the system
  – Removed assumption of memory starting from address 0x0. Lowest possible physical address on Altix is 0x3000000000
  – Net boot time reduction: 3 minutes!
  – Large amounts of memory still wasted covering holes

• Reduced excessive RAM scrubber output
  – Only print dots based on existing pages in system
Metaphysical Memory placement

- Metaphysical memory for dom0 must match expected physical memory address ranges
  - Required for NODE awareness (NUMA)
  - Similar implementation from Isaku Yamahata from VA Linux Japan, required to support PCI domains
- What about domU?
Current status

• Xen boots .....  
  – dom0 loads .....  
  – dom0 mounts /  
  – dom0 hangs  
    • dom0 trying to execute or map code in userland fails for some reason

• Still a lot of work to do before booting
  – Fix one item, a new one surfaces
  – Further Altix specific features needed for PCI support etc.
    • No special I/O node (TIO) support
    • Currently only worked on supporting Shub 1
  – DMA flush and other hardware workarounds to be ported
  – VGA support (maybe)
  – Figure out why PCI interrupts are not being delivered
Xen future work / wishes

- Scalable memory allocation / management (discontig)
- Scalable vCPU scheduler
- NUMA aware metaphysical placement for all dom0/U
  - Option to provide DIG style mem-map to domU would allow booting lesser operating systems on hardware these do not support today
  - Fake NUMA placement info provided to userland will cause highly unpredictable performance of userland applications.
  - Ditto for inefficient scheduling of tasks by domU
- Bind vCPU to node
  - Allow reliable scheduling of userland tasks by domU
- PCI/IOMMU/DMA support
  - Without IOMMU, no 32bit device support (including USB)
The IOMMU problem

• Without IOMMU support, no possibility for supporting 32 bit PCI devices on systems given no memory below 4GB

• Three options:
  – Guarantee all memory in dom0 has meta-phys == phys, and let dom0 do the IOMMU programming
  – Put the IOMMU code into Xen as well (a lot of extra code to add)
  – Do a static IOMMU mapping to be used for bounce buffers and swiotlb (will result in pathetic IO performance)
Reality check

• Is Xen's 'keep it super-ultra-simple' approach realistic?
  – Even consumer PCs are going multi-core and NUMA
  – The complexity of Linux was put in place for a reason
  – The IBM PC-XT is no longer on the market!

• Will Xen ever become a performance platform?
  – As opposed to simply a functionality platform

• The big mystery: Keeping Xen and dom0 separate?
Questions?