The Linux/ia64 Project

Khalid Aziz
Hewlett Packard Co.
khalid_aziz@hp.com
The Linux/ia64 Project

IA-64 Overview
Key features of IA-64

- 64-bit architecture co-designed by Intel and HP
- New EPIC (Explicitly Parallel Instruction-set Computing) paradigm
- First implementation: Itanium™
- Massive resources: 128 integer & 128 floating point registers
- Predication: Avoid costly branches
- Speculation (Control & Data): Hide memory latency
- Register Stack Engine
- Performance Monitoring
- IA-32 hardware emulation
The Linux/ia64 Project

Project Overview
Goals of the Linux/ia64 project

- Single IA-64 Linux port
- Optimized for IA-64
- Open source the code before or at Itanium™ processor launch
- Cooperative effort to deliver best possible code
  - follow the Linux development model
  - many players contributing technology and resources:
    - Caldera, CERN, HP, IBM, Intel, Linuxcare, NEC, Redhat/Cygnus, SGI, SuSe, TurboLinux and VA Linux Systems.
Project History

⇒ Project started at HP Labs in February 1998

⇒ Initial contributors:
  - HP Labs
    - Initial toolchain, kernel architecture and implementation, simulator, apps
    - Demonstrated first kernel booting to user mode on simulator in March 1999 (1st LinuxWorldExpo, San Jose)
  - CERN (birthplace of the WWW)
    - User-level libraries (libc/libm)

⇒ Spring ’99: HP co-founded the "Trillian" project
  - now renamed "IA-64 Linux project"
Milestones

Kernel
- Complete source code released 2/2/2000 (LWE)
- Merged into mainline by Linus as of 2.3.42
- Current work being done on 2.4.0 kernels
- Latest IA-64 patches:
  - http://www.kernel.org/pub/linux/ports/ia64

Development tools (GNU tools including gdb)
- Source code released 2/17/2000 (IDF)

User Libraries (GNU libc/libm)
- Source code released 2/2/2000 (LWE)

Linux/ia64 first OS available for this architecture
The Linux/ia64 Project

Kernel Design
Kernel Design Goals

▬ Pure 64-bit kernel for IA-64

▬ Minimize modifications to machine independent code:
  ▪ Added arch/ia64 and include/asm-ia64
  ▪ Follow development kernels: from v2.1.126 to v2.4.0-test7

▬ APIs compatible with Linux/x86 wherever possible:
  ▪ sig-num, ioctl-num, errno,...

▬ Optimize for 64-bit performance

▬ Follow standard wherever possible:
  ▪ IA-64 SW calling convention, DiG, EFI, UNIX ABI
LP64 Data Model

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Alignment</th>
<th>Type</th>
<th>Size</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1</td>
<td>1</td>
<td>float</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
<td>2</td>
<td>double</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
<td>4</td>
<td>long double</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>long int</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>long long int</td>
<td>8</td>
<td>8</td>
<td>void *</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

with current gcc size=8,align=8

Endianess

- Little-endian is native byte order
- Big-endian processes are possible
Kernel Register Usage

Fix some registers to point to heavily used variables
- takes advantage of large register file
- saves memory access latency
- uses gcc -ffixed-X option

Limit floating point usage in kernel
- minimize context switch time (don’t touch high partition)
- [f10-f15] and [f32-f127] are not accessible
- useful for some integer or bit operations (multiply, find highest bit set)

<table>
<thead>
<tr>
<th>Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>r13 (&quot;thread pointer&quot;)</td>
<td>current task pointer (current)</td>
</tr>
<tr>
<td>ar.k0</td>
<td>legacy I/O base</td>
</tr>
<tr>
<td>ar.k5</td>
<td>floating point high partition owner</td>
</tr>
<tr>
<td>ar.k6</td>
<td>physical address of current task structure</td>
</tr>
<tr>
<td>ar.k7</td>
<td>physical address of page table</td>
</tr>
</tbody>
</table>
Virtual Memory Subsystem

- Page sizes:
  - kernel configuration option
  - supported sizes: 4, 8, 16 or 64KB
  - use getpagesize() to get current setting

- Why multiple sizes?
  - 4KB allows perfect Linux/ia-32 emulation
  - >4KB
    - good for Linux/ia-32
    - better for IA-64 native binaries (TLB pressure)
    - bigger sizes yield bigger virtual address space
Virtual address space breakdown

- Linux uses the 8 regions
  - headroom for future growth

<table>
<thead>
<tr>
<th>Region</th>
<th>Current usage</th>
<th>Page size</th>
<th>Scope</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>region 0</td>
<td>cached</td>
<td>large (256MB)</td>
<td>global</td>
<td>identity</td>
</tr>
<tr>
<td>region 1</td>
<td>uncached</td>
<td>large (256MB)</td>
<td>global</td>
<td>identity</td>
</tr>
<tr>
<td>region 2</td>
<td>vmalloc</td>
<td>kconfig (8KB)</td>
<td>global</td>
<td>page-table</td>
</tr>
<tr>
<td>region 3</td>
<td>stack segment</td>
<td>kconfig (8KB)</td>
<td>process</td>
<td>page-table</td>
</tr>
<tr>
<td>region 4</td>
<td>data segment</td>
<td>kconfig (8KB)</td>
<td>process</td>
<td>page-table</td>
</tr>
<tr>
<td>region 5</td>
<td>text segment</td>
<td>kconfig (8KB)</td>
<td>process</td>
<td>page-table</td>
</tr>
<tr>
<td>region 6</td>
<td>shared memory</td>
<td>kconfig (8KB)</td>
<td>process</td>
<td>page-table</td>
</tr>
<tr>
<td>region 7</td>
<td>IA-32 emulation</td>
<td>kconfig (8KB)</td>
<td>process</td>
<td>page-table</td>
</tr>
</tbody>
</table>

16384 PB (1PB = 1024TB)
Regions 0-4 mapped by single 3-level page table
Each region gets 1/8th of L1 page table

virtual address

3 21 7 10 10 13

63 61 60 40 39 33 32 23 22 13 12 0

= sign extension of bit 39

with 8KB pages: $2^{43}$ bytes address space
Anatomy of a user region

Sign extension (bit [40-60]) creates a software-imposed hole in the middle

2^61 bytes

inaccessible address space (access causes SIGSEGV)

2^39 bytes

0xffffffffffffffffffffff

0xffffffff8000000000

0x00000007ffffffff

0x0000000000000000
Region 5 has its own page table
- full 43 bits address space (with 8KB pages)

Mapped kernel region

- Region 5 has its own page table
- full 43 bits address space (with 8KB pages)

virtual address

```
63 _61 60 _43 42 _33 32 23 22 _13 12 _0
```

= sign extension of bit 42

page frame

SWAPPER_PG_ADDR

L1 PT (pgd)

L2 PT (pmd)

L3 PT (pte)

PTE
dword

= sign extension of bit 42

= sign extension of bit 42
VHPT usage

⇒ We use the short mode
  ▪ perfect with 3-level page table: L3 has VHPT format
  ▪ good for densely populated address space:
    ▪ with 8KB pages, one VHPT fault good for an area of 8MB (1024*8KB)

⇒ Where to place the table in each region?
  ▪ architecture requires at least 50 bit implemented
  ▪ placed in inaccessible area above unimplemented address space
Anatomy of user region with VHPT

⇒ bit [51-60] sign-extension of bit 50

- $2^{50}$ bytes
- Virtually mapped linear page table (no user access)
- $2^{61}$ bytes
- Unimplemented address space
- Inaccessible address space
- $2^{39}$ bytes
- Note: With 64KB pages, IMPL_VA_MSB must be bigger than 50 to avoid overlap with user space.
What?
- execute unmodified Linux/ia-32 user level binaries

Why?
- provide transition path for binary-only applications
  - Netscape Navigator, Acrobat, Applix, WordPerfect....

How?
- Kernel level:
  - 2 ELF loaders: ELF32 and ELF64
  - system call emulations: LP32/LP64 differences
  - IA-32 context setup
- User level:
  - 2 dynamic loaders: ld-linux.so.2 (IA-32) & ld-linux-ia64.so.1 (IA-64)
  - slightly modified dynamic loader: architecture-aware enhancements
IA-32 system call emulation

- Processing varies depending on calls:
  - limit amount of work by using same constant values
  - nothing to do: getpid()
  - systematic 32/64 bit conversions for parameters:
    - example: gettimeofday()/settimeofday(). timeval structs differ

- Kernel has 2 separate system call tables
  - when int 0x80 encountered, trap in kernel, switch to IA-64 mode
  - dispatched via IVT to IA-32 system call table

- IA-32 binaries are placed in region 0 (lower AS)
Port & develop Linux/ia64 apps on Linux/ia32
Based on HP Labs Native User Environment (NUE)

- Create illusion of target dev. environment:
  - cross dev. tools, headers files, libs : real name and location
  - chrooted environment: no conflict with host system
  - no tweaking of Makefiles required
  - RPM rebuild easy: `% rpm --rebuild mingetty-0.9.4.src.rpm`

- Execution of produced binaries possible:
  - includes the HP IA-64 instruction set simulator (ski)
  - transparent execution of IA-64 binaries at shell prompt
  - mix & match IA-64/IA-32 binaries
  - debugging with ski
IA-64 Software Development Kit

⇒ Kernel development & execution possible
  • very good for low level code debugging
  • applications testing
  • OS research activities

⇒ Other Open Source OS development possible
  • like FreeBSD or NetBSD

⇒ Kit available for free since 06/13/2000
  • web site: http://www.software.hp.com/ia64linux
  • Mailing list: lia64-sim-request@linux.hpl.hp.com
Ski & NUE

vi main.c

Ski

IA-64 instruction set simulator
Debugger
IA-64 Linux syscall emulator

NUE

gcc

gas

ld

libc.so
libX11.so

libm.so

Standard Linux user environment (Red Hat & TurboLinux based)

x86 Linux kernel

x86 code
IA-64 code
cross-tool (x86 code, but manipulates IA-64 code)
Kernel Simulation Environment

Linux/ia64 kernel

- Terminal subsystem
- SCSI subsystem
- Network subsystem

HP IA-64 instruction set simulator

normal Linux/x86 kernel packet processing

packets for sim IP or bcast

packets for host IP addr or bcast

Ethernet

simserial
simscsi
simeth

bash
vi ls.c
strace
ps auxw

xterm: sim. terminal
loading vmlinux...
starting kernel
Linux version 2.3.0
Calibrating delay loop...

伦理/ia64 host

simserial simscsi simeth

simserial simscsi simeth

/btmp/sda
/btmp/sdb

nic

Copyright (c) 2000 Hewlett-Packard Co.
The Linux/ia64 Project

Conclusion
Conclusion

- Linux/ia64 is real!
  - First OS publicly available for IA-64
  - Initial distributions from Caldera, Redhat, SuSE & TurboLinux available

- HP is committed to Linux
  - Major technology contributions

- IA-64 SDK allows Linux/ia64 user & kernel development on any x86 Linux PC

- Next steps:
  - Debugging
  - Tuning, performance optimizations
Resources on the WEB

▶ IA-64 Architecture
  - http://www.hp.com/go/ia64
  - http://developer.intel.com/design/ia-64

▶ HP & Linux
  - http://www.hp.com/go/linux/

▶ IA-64 Linux project (formerly Trillian)
  - http://www.linuxia64.org/