Porting Linux to IA-64

Le portage de Linux sur AI-64

Ottawa LinuX Symposium

July 1999

Stéphane Eranian
Hewlett-Packard Labs

HEWLETT
PACKARD

Outline

- What is IA-64 ?
- Goal of the project
- Development environment
- Kernel work
- User land
- Demo

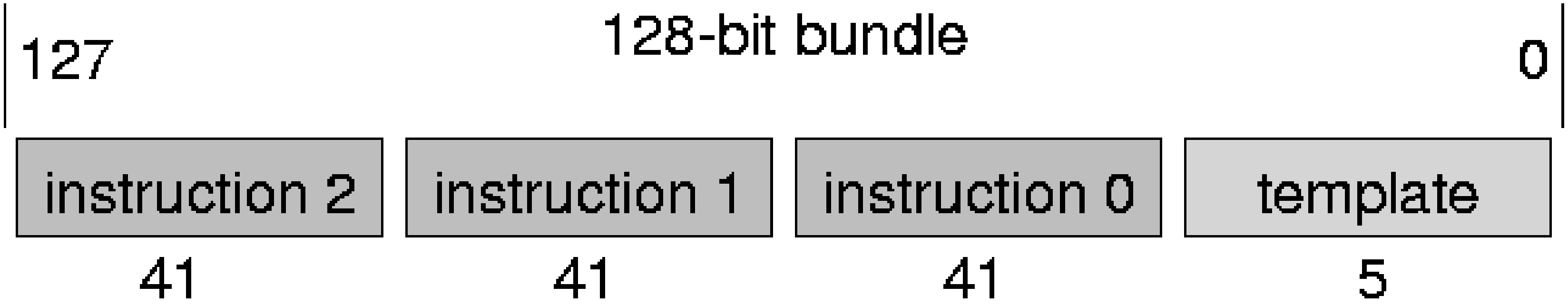
What is IA-64?

- ⇨ next-generation, high-performance architecture co-designed by Intel and HP
- ⇨ new EPIC paradigm: Explicitly Parallel Instruction-set Computing
- ⇨ first implementation: Merced
- ⇨ general availability: mid 2000

What is EPIC ?

⇨ Explicit parallelism

- bundles of 3 instructions
- template field encodes
  - type of execution units needed (M,I,B,F)
  - stop bit to express sequential dependency



128-bit bundle

127 … 0

instruction 2 | instruction 1 | instruction 0 | template

41 | 41 | 41 | 5

⇨ Massive resources

- 128 integer (64bits) & 128 floating point (82bits) registers
- lots of execution units

Predication

- To reduce branching
  - 64 predicate registers (1 bit each)
  - when predicate is false instruction is not executed

```
C code:
r2=r1==0?r4+r5:r3+r6+1;

IA-64 assembler:
     cmp.eq p1,p2=r0,r1;;
(p1) add r2=r4,r5
(p2) add r2=r3,r6,1
```

synchronization

Control Speculation

- ⇨ execution of a load before the branch that guards it
  - ❑ available for integer & floating point registers loads

- ⇨ Safety ensured with NaT (Not a Thing) bit
  - ❑ "65th" bit of integer registers
  - ❑ Specific "NatVal" used for floating point registers

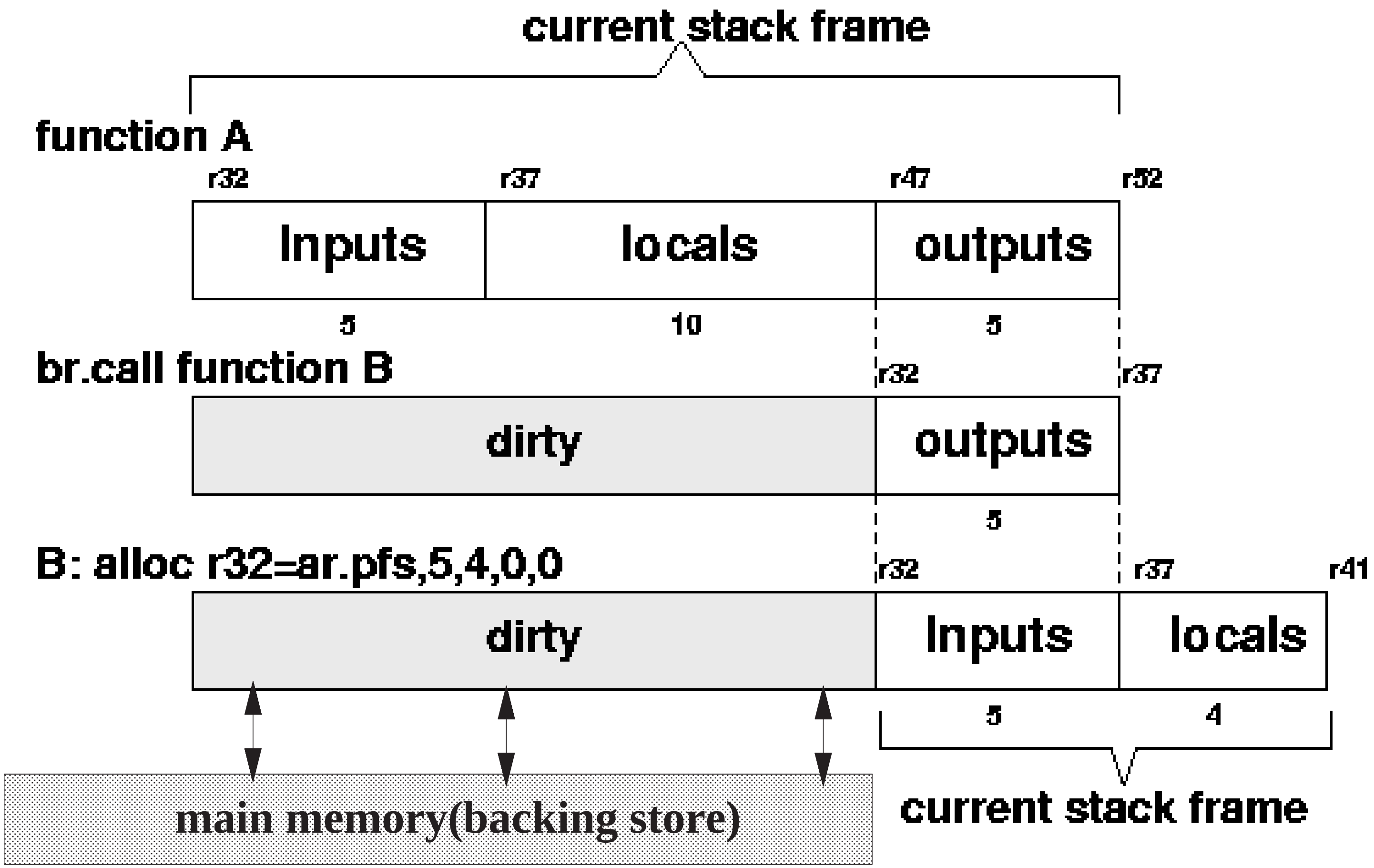
```
(p1) br.cond label  |0||      ld8.s r1=[r5]      |-2
     ld8 r1=[r5];;  |1||// do something else     |
     add r2=r1,r3   |3||(p1) br.cond label       | 0
                    | ||     chk.s r1, recovery  | 0
                    | ||     add r2=r1,r3        | 0
```

Data Speculation

- execution of a load before potentially conflicting stores (aliased address)
- also called advanced loads
- CPU internal table : ALAT (Adavanced Load Address Table)
- Specific check instructions to verify load target validity: chk.a, ld.c...

Register Stack Engine (RSE)

⇨ avoid spills/fills on procedure calls



Register Rotation

- easy loop unrolling
- no code expansion
- dynamic register renaming
  - integer (32-127), floating point (32-127) registers
  - predicate registers (16-63)
- software pipelining
  - loop prolog,epilog inside core loop body

Why Port Linux to IA-64 Now?

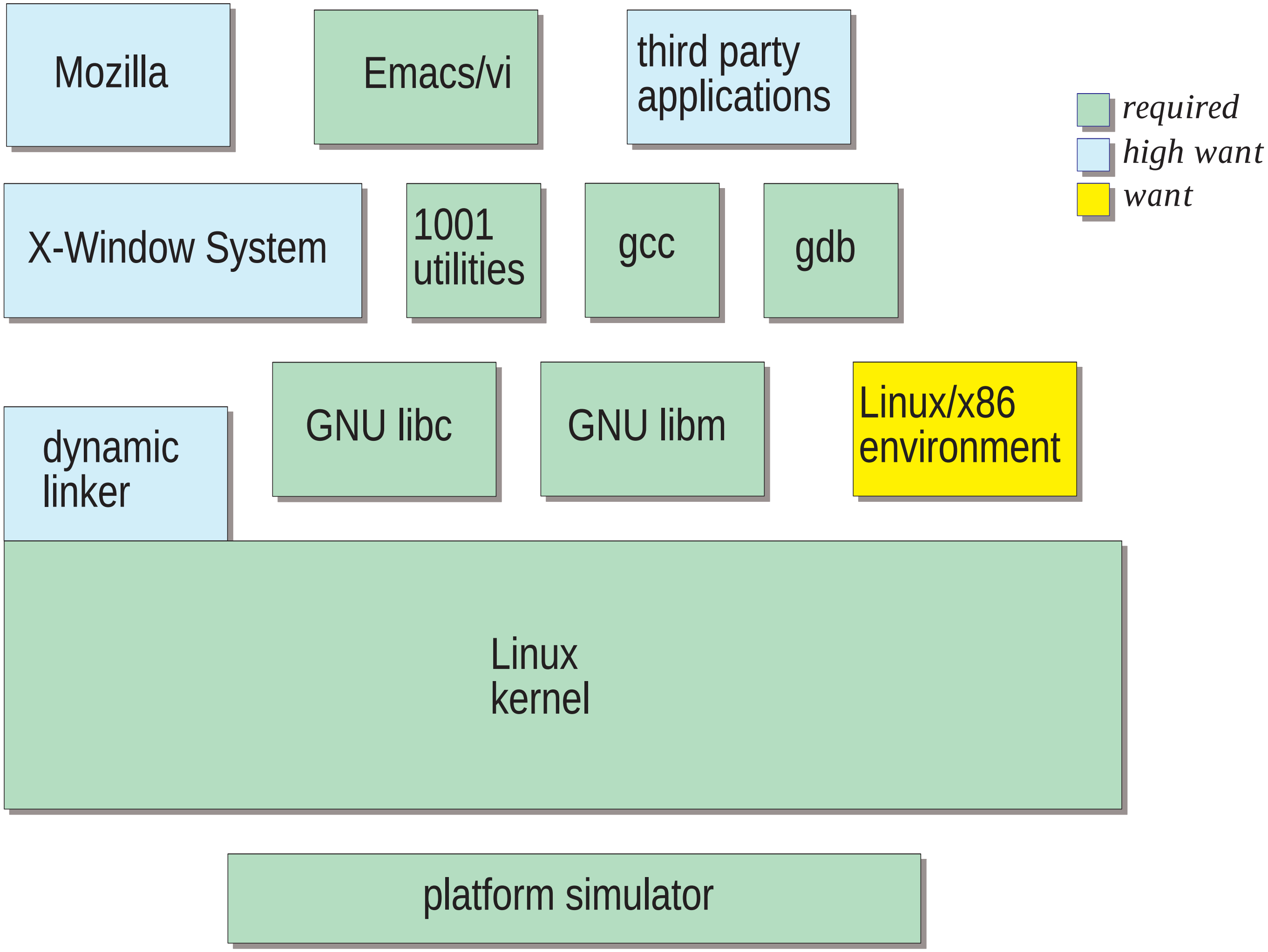
- ⇨ For Linux to be taken seriously, it must be ready for launch of first IA-64 platforms ("Merced" chip)
- ⇨ Developing IA-64 optimizing compiler, kernel, and applications takes time
- ⇨ GPL does allow "private modifications" as long as no distribution
- ⇨ Release to open source
  - ❑ when NDAs expire (general hardware availability)

Goal of Linux/ia64 Project

- Original goal (Feb ’98)
  - self-hosting Linux system ready when first IA-64 based machines become available
  - focus on functionality, not performance

- Revised goal (Feb ’99)
  - easy-to-install Linux distribution by launch of Merced
  - functionally complete
  - optimized for performance
    - compiler, kernel, libraries, and applications
  - SMP support
  - Linux/x86 binary compatibility

What’s involved?



Mozilla

Emacs/vi

third party applications

X-Window System

1001 utilities

gcc

gdb

dynamic linker

GNU libc

GNU libm

Linux/x86 environment

Linux kernel

platform simulator

🟩 *required*
🟦 *high want*
🟨 *want*

Who’s Involved?

- ⇨ HP Labs
  - ❑ toolchain, kernel architecture and implementation
- ⇨ CERN (European Laboratory for Particle Physics)
  - ❑ User-level libraries
- ⇨ High-ranking Linux kernel developer
  - ❑ kernel development and validation
- ⇨ Collaboration with Cygnus, Intel, SGI, VA Linux Systems (Project Trillian)

Virtual Team Picture

ToolChain

- ⇨ GNU C required : egcs-1.1.2
  - ❏ Functional back-end, no EPIC optimizations
- ⇨ Complete binutils (BFD, gas, ld)
  - ❏ gas-990404
- ⇨ Programming model : LP64
  - ❏ longs, pointers are 64bits, integers are 32bits
- ⇨ Binary format : ELF64/IA-64
- ⇨ Recompile when better compiler available

Simulator

- ⇨ HP’s instruction set architecture simulator
  - ❑ CPU only (no platform)
  - ❑ user/system modes
- ⇨ Ported to Linux/x86
  - ❑ easy system call emulation (user mode)
  - ❑ entire development hosted on Linux
- ⇨ I/O access via simulator (trap) and host OS
  - ❑ simulated disk using a file as a diskimage
  - ❑ simulated serial console using xterm
  - ❑ simulated ethernet using raw Ethernet frames

The Kernel

⇨ Approach

- ❑ Minimize modifications to machine independent code
- ❑ added arch/ia64 and include/asm-ia64
- ❑ Follow development kernel : from v2.1.126 to v2.3.9
- ❑ Incremental bring up of subsystems

⇨ Kernel Attributes

- ❑ Byte ordering : Little-endian
- ❑ Page size : >= 8KB
- ❑ Virtual address space: 43bits (8TB)

⇨ Special devices drivers for I/O access

- ❑ interrupt driven, trap into simulator
- ❑ simscsi (SCSI), simserial (console), simeth(network)

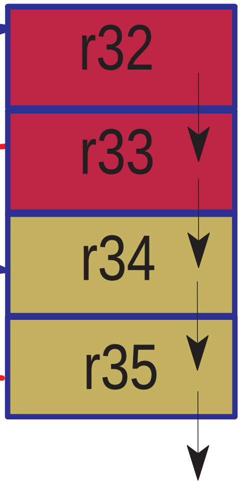
Kernel Status

- Completed
  - system initialization
  - process subsystem
  - virtual memory subsystem
  - signal subsystem
  - network subsystem
  - ptrace support
  - some optimizations
- To do
  - SMP support
  - Linux/x86 support
  - platform-dependent support
  - kernel modules
  - finish performance tuning

Code example: strlen_user()

```
      add r17=8,r16
      cmp.eq p6,p0=r0,r0
1:
      ld8.s r32=[r16],16
      ld8.s r34=[r17],16
      czx1.r r14=r33
      czx1.r r15=r35
      ;;
      cmp.eq     p6,p0=8,r14
      cmp.eq.and p6,p0=8,r15
(p6)  br.wtop.dptk.few 1b
```

init p6 to true



parallel compares

User Level

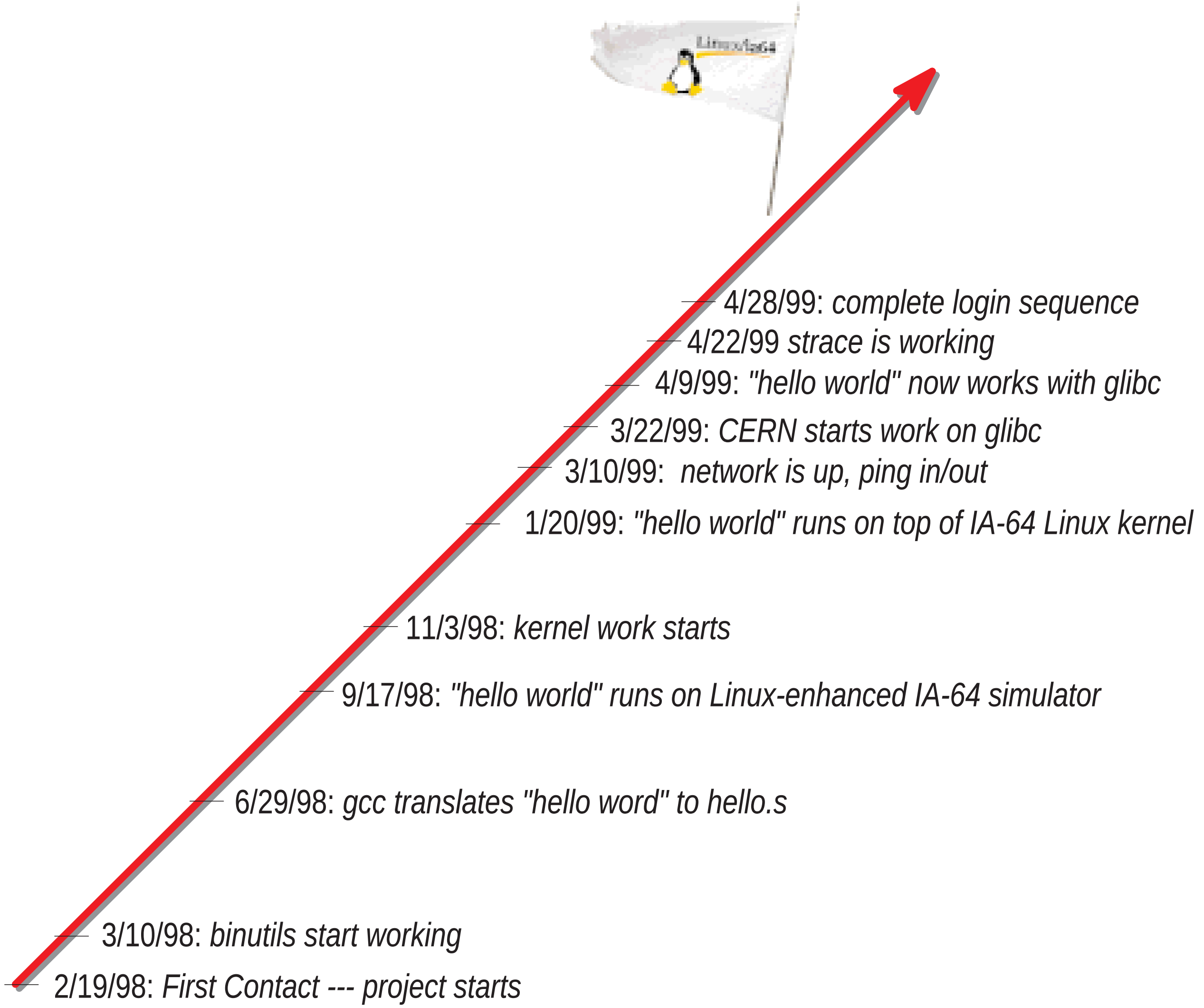
- ⇨ CERN port of GNU libc v2.1 (libc,libm)
  - ❑ generic port first (optimizations later)
  - ❑ statically linked

- ⇨ Basic packages "available":
  - ❑ ported packages from standard distribution RPMs
  - ❑ complete login sequence: init, mingetty, login
  - ❑ shells: pdksh, bash, tcsh, ash
  - ❑ editor: vim, vile, emacs (not complete)
  - ❑ utilities: fileutils, sh-utils, text-utils, netkit-base...

User Level Todo List

- ⇨ debug and optimize libc/libm
  - ❑ EPIC optimizations to performance critical routines
  - ❑ dynamic loader
- ⇨ start porting higher-level applications and libraries
  - ❑ X-Window: XFree86, GNOME/KDE
  - ❑ Web browser: Mozilla
  - ❑ debugger: gdb
  - ❑ Languages like Java, Fortran, Tcl/Tk, Perl
  - ❑ others...
- ⇨ Help by making sure software is 64-bit clean
  - ❑ no abusive casts (long & pointer 64bits, int 32bits)
  - ❑ careful with hardcoded data structure sizes

Timeline of Linux/ia64 evolution



4/28/99: *complete login sequence*

4/22/99 *strace is working*

4/9/99: *"hello world" now works with glibc*

3/22/99: *CERN starts work on glibc*

3/10/99: *network is up, ping in/out*

1/20/99: *"hello world" runs on top of IA-64 Linux kernel*

11/3/98: *kernel work starts*

9/17/98: *"hello world" runs on Linux-enhanced IA-64 simulator*

6/29/98: *gcc translates "hello word" to hello.s*

3/10/98: *binutils start working*

2/19/98: *First Contact --- project starts*

Attributes of Linux/ia64

Host platform:	Linux/x86 & HP-UX
Cross compiler:	egcs-2.91.66 (egcs-1.1.2 release)
(Dis-)Assembler, linker, ELF:	binutils based (gas-990404)
Simulator:	kernel & Linux user level simulator
Linux kernel:	Linux v2.3.9 (and tracking...)
Programming model:	LP64
Byteorder:	little endian
Object file format:	ELF64/IA-64
Calling convention:	99.9% standard
Page size:	>= 8KB
Virtual address space size:	43 bits (with 8KB pages)

Conclusions

- Linux/ia64 will be ready for Merced-launch in mid-2000

- NDAs make open-source development harder, but not impossible as the successful collaboration on Linux/ia64 demonstrates

- You can help
  - 64-bit clean software
  - test suites

Acknowledgments

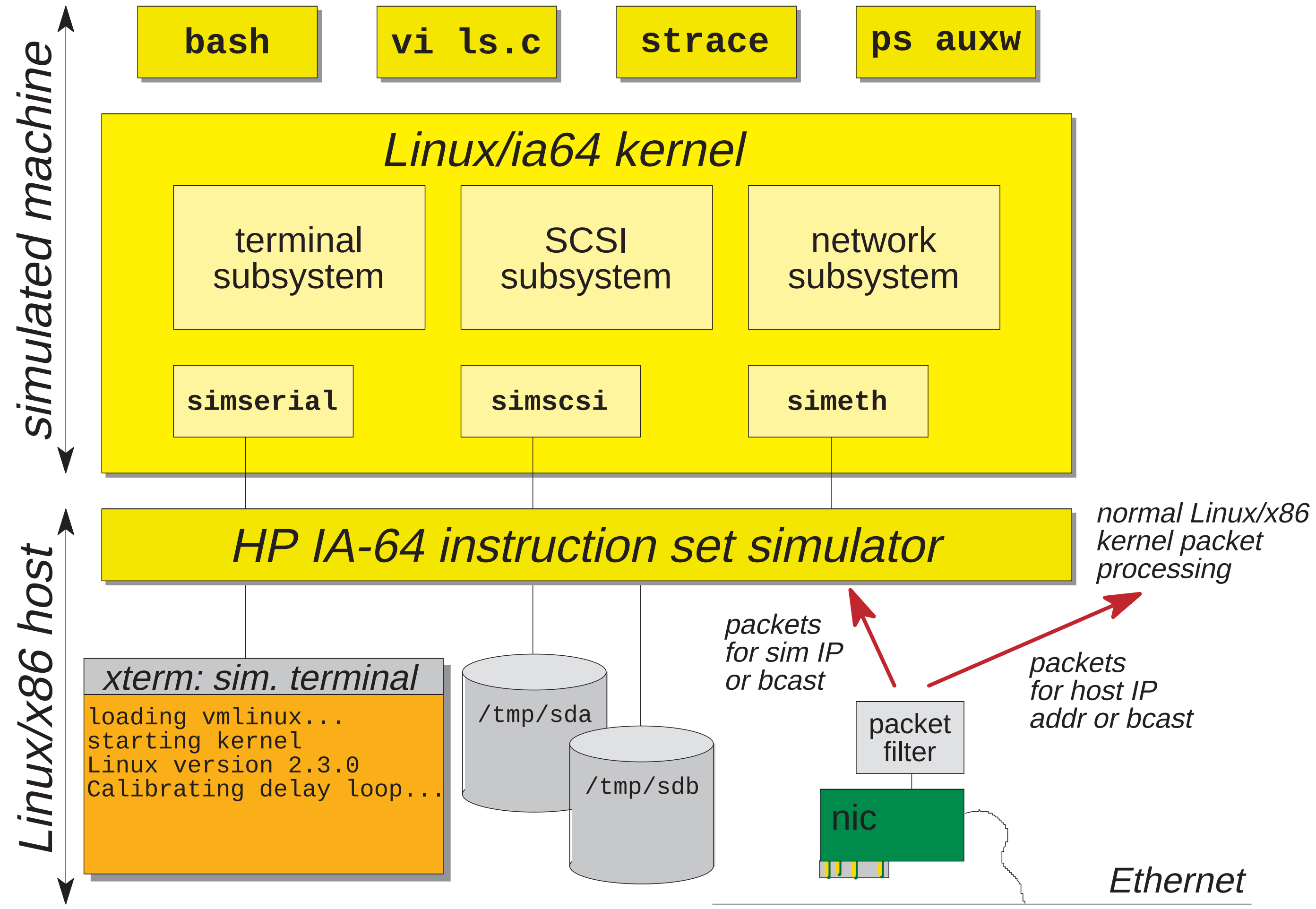
Thanks to

- Fred Luiz, HP Labs
- Jean Gascon, HP Labs
- Michel Benard, HP External Research

Resources

http://www.hp.com/go/linux/
http://www.hp.com/go/ia64/

Kernel simulation environment



```
loading vmlinux...
starting kernel
Linux version 2.3.0
Calibrating delay loop...
```