The Nanvix Operating System

Pedro H. Penna

BSc. Computer Science
pedrohenriquepenna@gmail.com

October 2016

Agenda

Introduction

Context

- Early days of the Computer Era
  - Computers filled up entire rooms to perform simple calculations
  - Programs were written in machine code
  - Computer scientists had to deal with the bare hardware

Introduction

Context

- Early days of the Computer Era
  - Computers filled up entire rooms to perform simple calculations
  - Programs were written in machine code
  - Computer scientists had to deal with the bare hardware
- Information Era
  - Parallel architectures
  - Large-scale distributed systems
  - Modern programming languages

Introduction

Context

- Early days of the Computer Era
  - Computers filled up entire rooms to perform simple calculations
  - Programs were written in machine code
  - Computer scientists had to deal with the bare hardware
- Information Era
  - Parallel architectures
  - Large-scale distributed systems
  - Modern programming languages
- What has enabled such evolution?
  - Computer abstractions
  - Software stack hardware

Introduction

Motivation

- Roles of an operating system
  - Multiplex access to resources
  - Extend the functionalities of the underlying

Introduction

Motivation

- Roles of an operating system
  - Multiplex access to resources
  - Extend the functionalities of the underlying
- Expertise in operating systems
  - Design complex systems
  - Master powerful mechanisms and strategies
  - Deliver extra expertise in emerging topics

Introduction

Motivation

- Roles of an operating system
  - Multiplex access to resources
  - Extend the functionalities of the underlying
- Expertise in operating systems
  - Design complex systems
  - Master powerful mechanisms and strategies
  - Deliver extra expertise in emerging topics
- Problems when learning about operating systems design
  - Simplistic, unrealistic or complex operating systems
  - Lack in documentation and teaching material

Agenda

Agenda

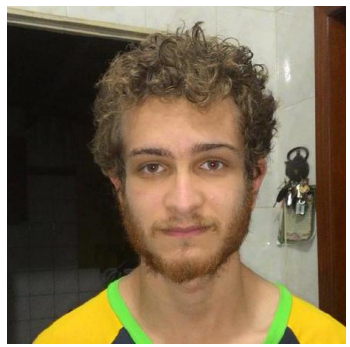
The Nanvix Operating System

Project Overview

- Created from scratch for educational purposes
- Designed to be small, simple, modern and fully featured
- Publicly available under the GPL v3 license at:

www.github.com/ppenna/nanvix

Pedro H. Penna
UFSC



Davidson Francis
PUC Minas

Subhra Sarkar
EchoStar Corp.

Figure: People involved in the Nanvix Project.

The Nanvix Operating System

Kernel Features

- POSIX compliant system call interface
- Unix System V architecture
- Non-preemptive
- Time-sharing
- Multiprogramming
- Interprocess communication
- Virtual memory with swapping
- Minix file system
- Uniform device interface

The Nanvix Operating System

User-Land Features

- Standard C Library
- Unix-Like utilities

```
Nanvix - A Free Educational Operating System

The programs included with Nanvix system are free software
under the GNU General Public License Version 3.

Nanvix comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Copyright(C) 2011-2016 Pedro H. Penna   <pedrohenriquepenna@gmail.com>
                       2015-2015 Davidson Francis <davidsondfgl@gmail.com>
                       2016-2016 Subhra S. Sarkar <rurtle.coder@gmail.com>

# echo "Hello World"
"Hello World"
# ps
------------------------------ Process Status ------------------------------
NAME                PID   UID       PRIORITY   NICE   UTIME   KTIME     STATUS
idle                0     0         40         20     0       4334      READY
init                1     0         40         20     1       0         WAITING
tsh                 2     0         40         20     0       1         WAITING
ps                  4     0         40         20     0       0         RUNNING

Last process: idle, pid: 0

#
```

Figure: Nanvix running.

Agenda

Baby Steps

Source Tree

- `bin`: binaries

Baby Steps

Source Tree

- `bin`: binaries
- `doc`: documentation

Baby Steps

Source Tree

- `bin`: binaries
- `doc`: documentation
- `include`
  - `include/dev`: device drivers headers
  - `include/fs`: file systems headers
  - `include/i386`: platform-specific headers
  - `include/nanvix`: kernel headers

Baby Steps

Source Tree

- `bin`: binaries
- `doc`: documentation
- `include`
  - `include/dev`: device drivers headers
  - `include/fs`: file systems headers
  - `include/i386`: platform-specific headers
  - `include/nanvix`: kernel headers
- `lib`: libraries

Baby Steps

Source Tree

- `bin`: binaries
- `doc`: documentation
- `include`
  - `include/dev`: device drivers headers
  - `include/fs`: file systems headers
  - `include/i386`: platform-specific headers
  - `include/nanvix`: kernel headers
- `lib`: libraries
- `src`
  - `src/kernel`: kernel sources
  - `src/lib`: libraries sources
  - `src/sbin`: superuser utilities sources
  - `src/ubin`: user utilities sources

Baby Steps

Building & Running Nanvix

```
$ cd ~
$ git clone https://github.com/ppenna/nanvix
$ cd nanvix
$ sudo bash tools/dev/setup-toolchain.sh
$ sudo bash tools/dev/setup-bochs.sh
$ sudo reboot now
$ cd ~/nanvix
$ make nanvix
$ sudo make image
$ sudo bash tools/run/run.sh
```

Agenda

The Nanvix Kernel

Overview

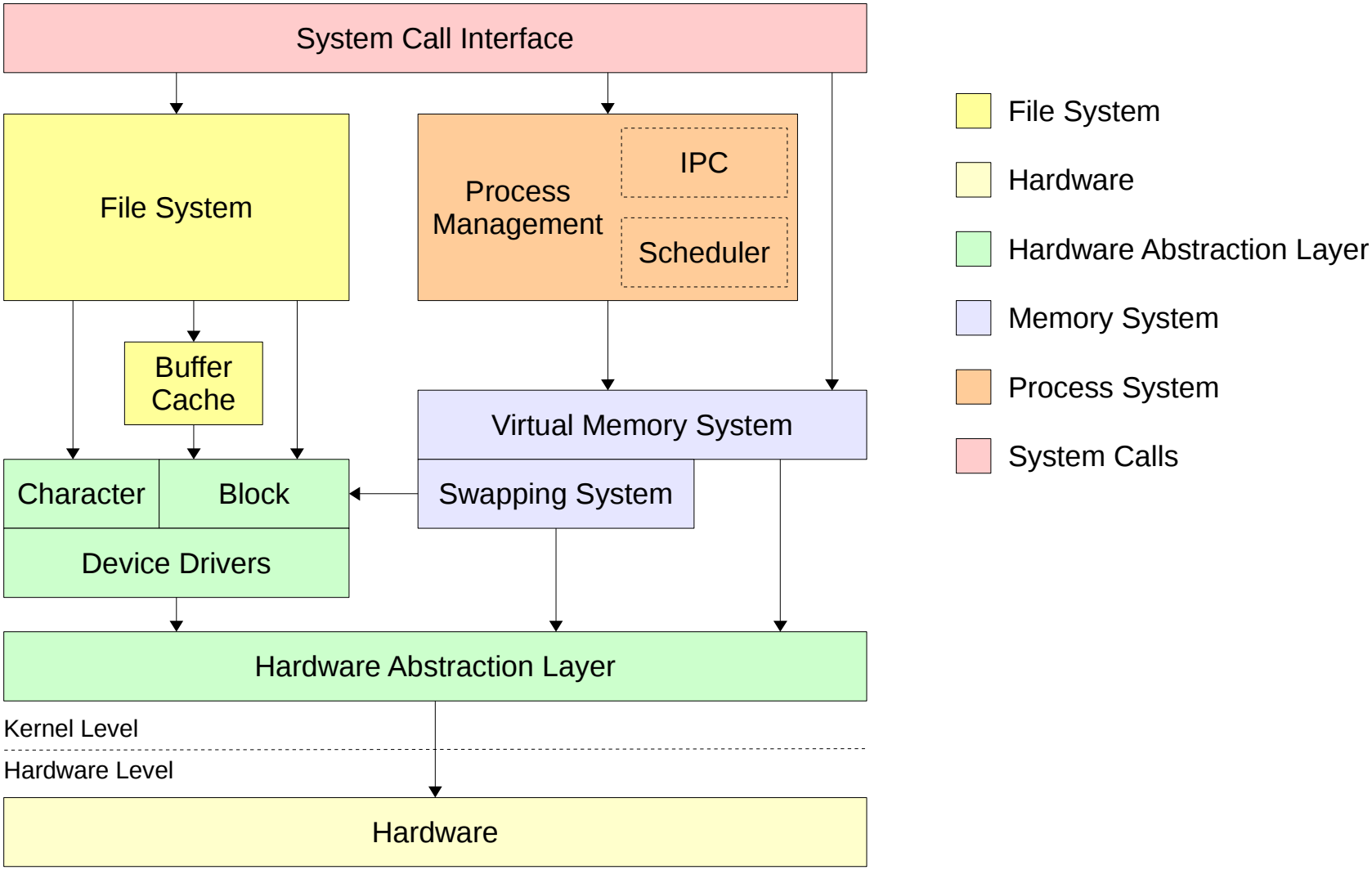


Table: Key system calls in Nanvix.

Category	**System Call(s)**	**Description**
File System	`open()`, `close()`	Opens/Closes a File Descriptor
	`read()`, `write()`	Reads/Writes to a File
	`link()`, `unlink()`	Creates/Removes a Link to a File
	`stat()`	Retrieves the Status of a File
	`chown()`	Changes the File Ownership
Process Management	`fork()`	Creates a Process
	`execve()`	Executes a Program
	`kill()`	Sends a Signal to a Process
	`pause()`, `exit()`	Suspends/Terminates the Process

Table: Key fields in a Process Control Block in Nanvix.

Category	**Field**	**Description**
Context Switch	`kstack`	Kernel Stack
File System	`pwd`	Current Working Directory
	`ofiles`	Opened Files
Memory Management	`pregs`	Memory Regions
	`pgdir`	Page Directory
Process Management	`state`	Current State
	`counter`	Remaining Quantum
	`pid`	Process ID
	`nice`	User-Level Priority

The Nanvix Kernel

The Process Management Subsystem

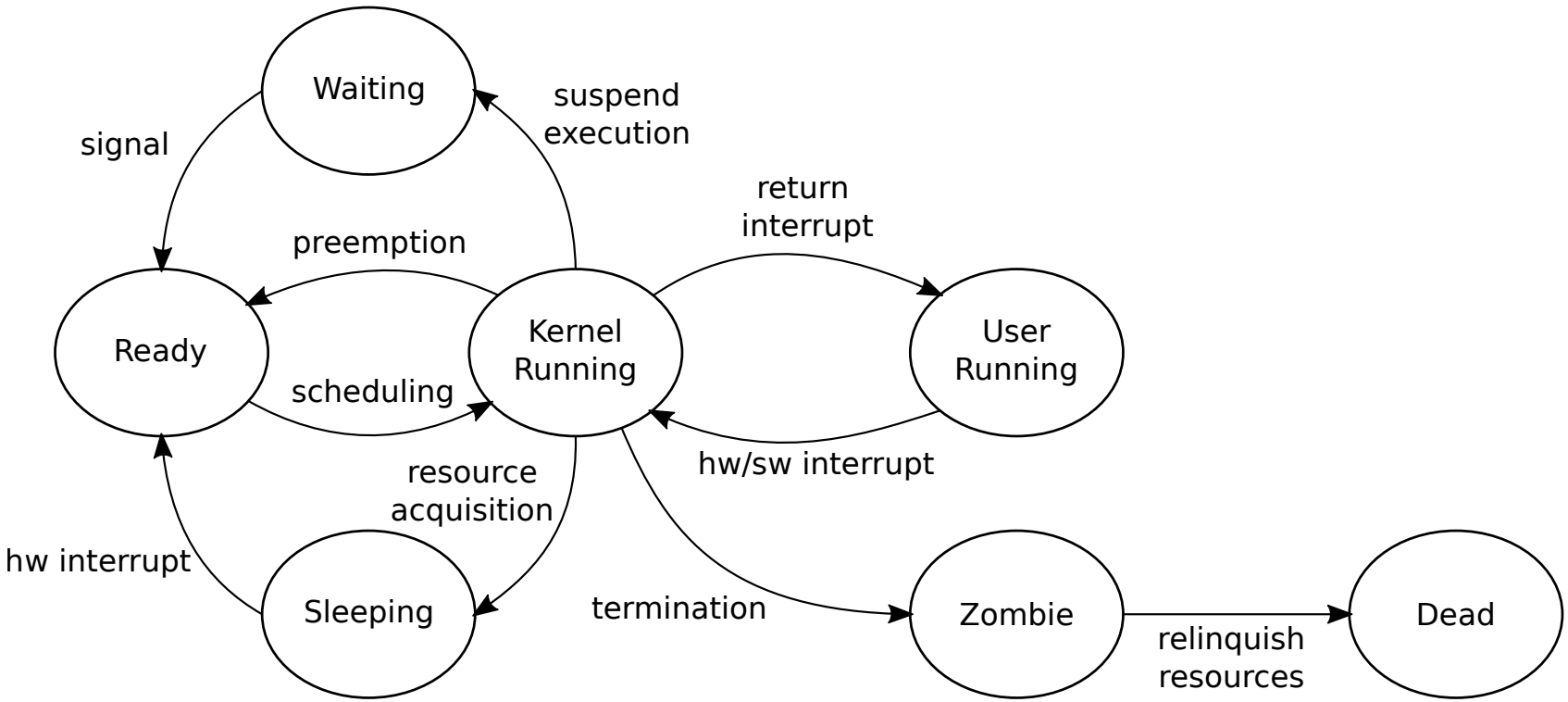


Figure: Figure: Life cycle of a process in Nanvix.

The Nanvix Kernel

The Process Management Subsystem

```
PUBLIC void yield(void) {
  struct process *p, *next = IDLE;

  if (curr_proc->state == PROC_RUNNING)
    sched(curr_proc);

  for (p = FIRST_PROC; p <= LAST_PROC; p++) {
    if (p->state != PROC_READY)
      continue;
    if (p->counter > next->counter) {
      next->counter++; next = p;
      continue;
    }

    p->counter++;
  }

  switch_to(next);
}
```

The Nanvix Kernel

The Memory Management Subsystem

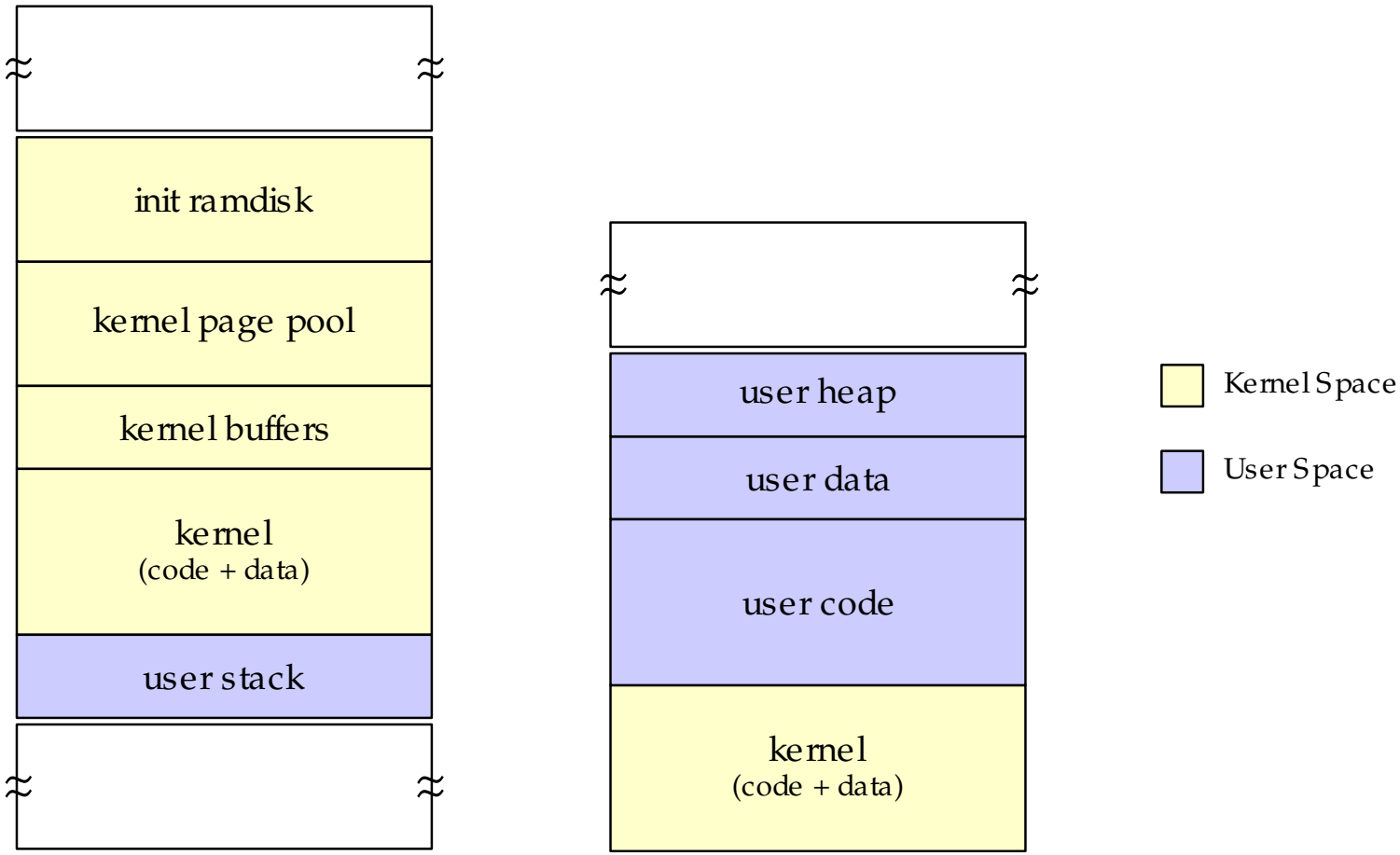


Figure: Memory layout in Nanvix.

The Nanvix Kernel

The Memory Management Subsystem

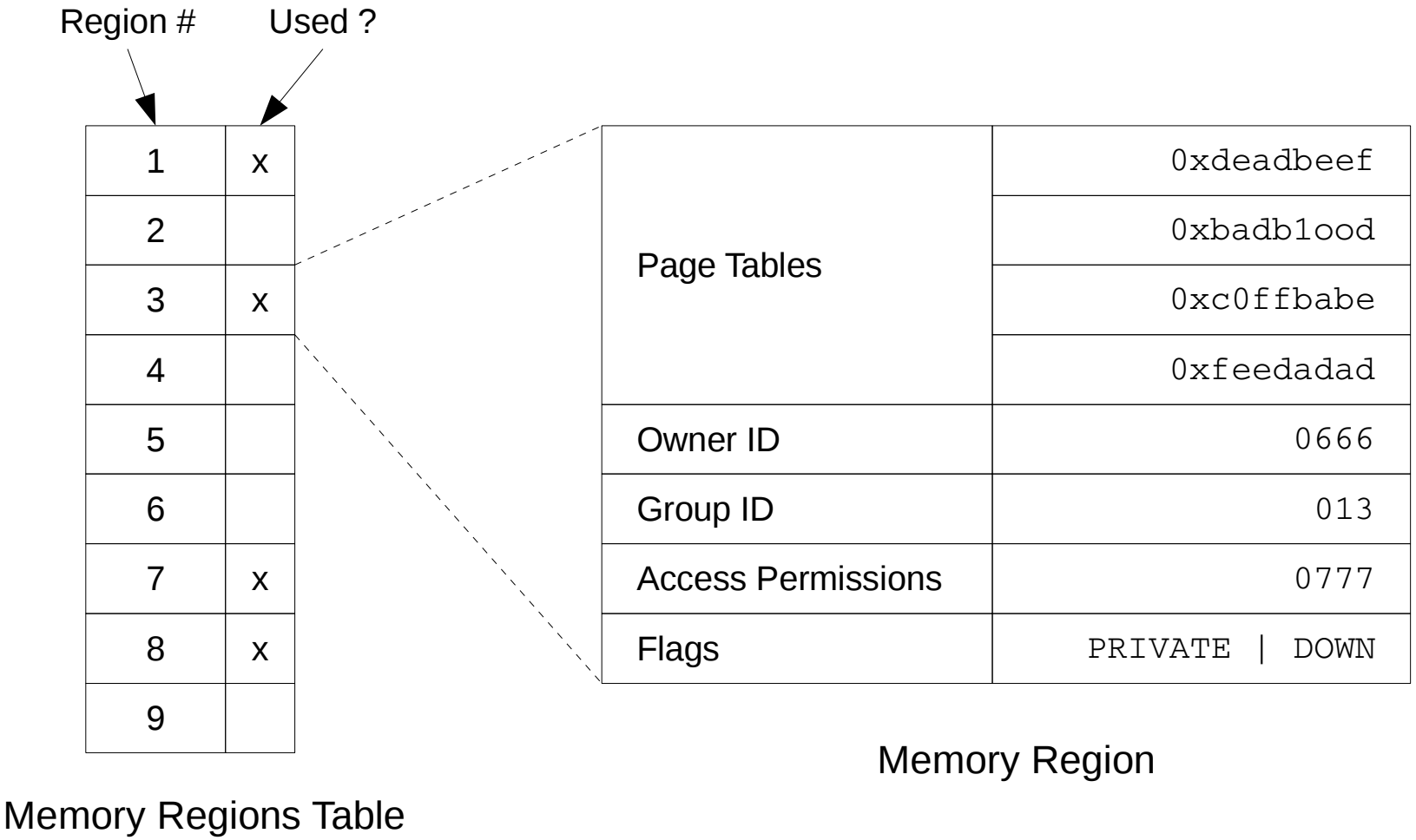


Figure: Memory regions in Nanvix.

The Nanvix Kernel

The Memory Management Subsystem

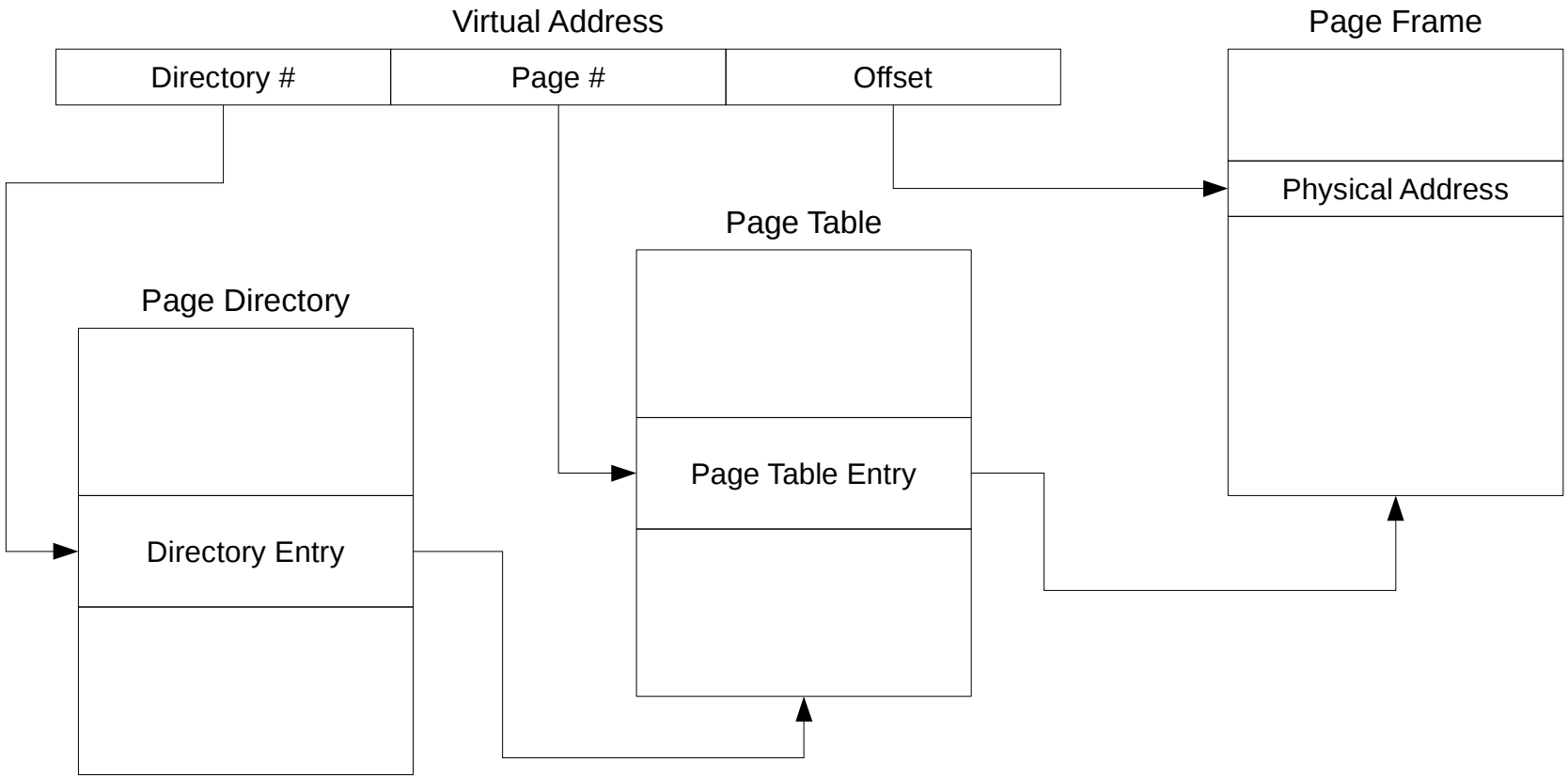


Figure: Paging scheme used in Nanvix.

The Nanvix Kernel

The Memory Management Subsystem

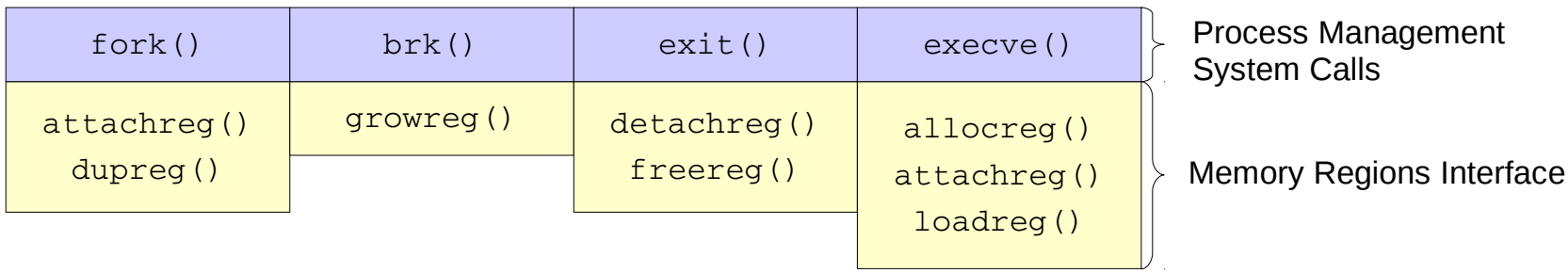


Figure: Memory regions interface in Nanvix.

The Nanvix Kernel

The File System

- Hierarchical file system
- Inodes hold metainformation about files

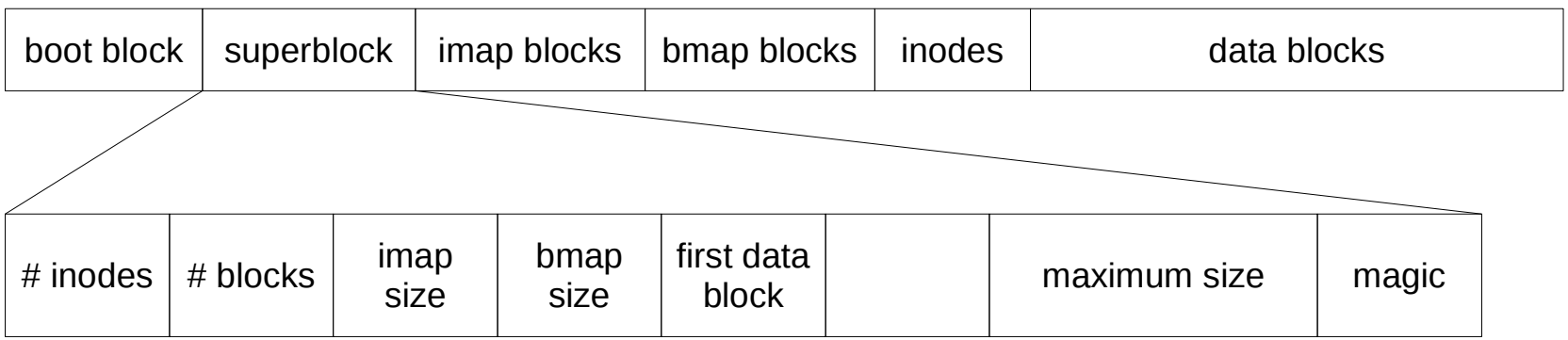


Figure: Nanvix file system layout.

The Nanvix Kernel

The File System

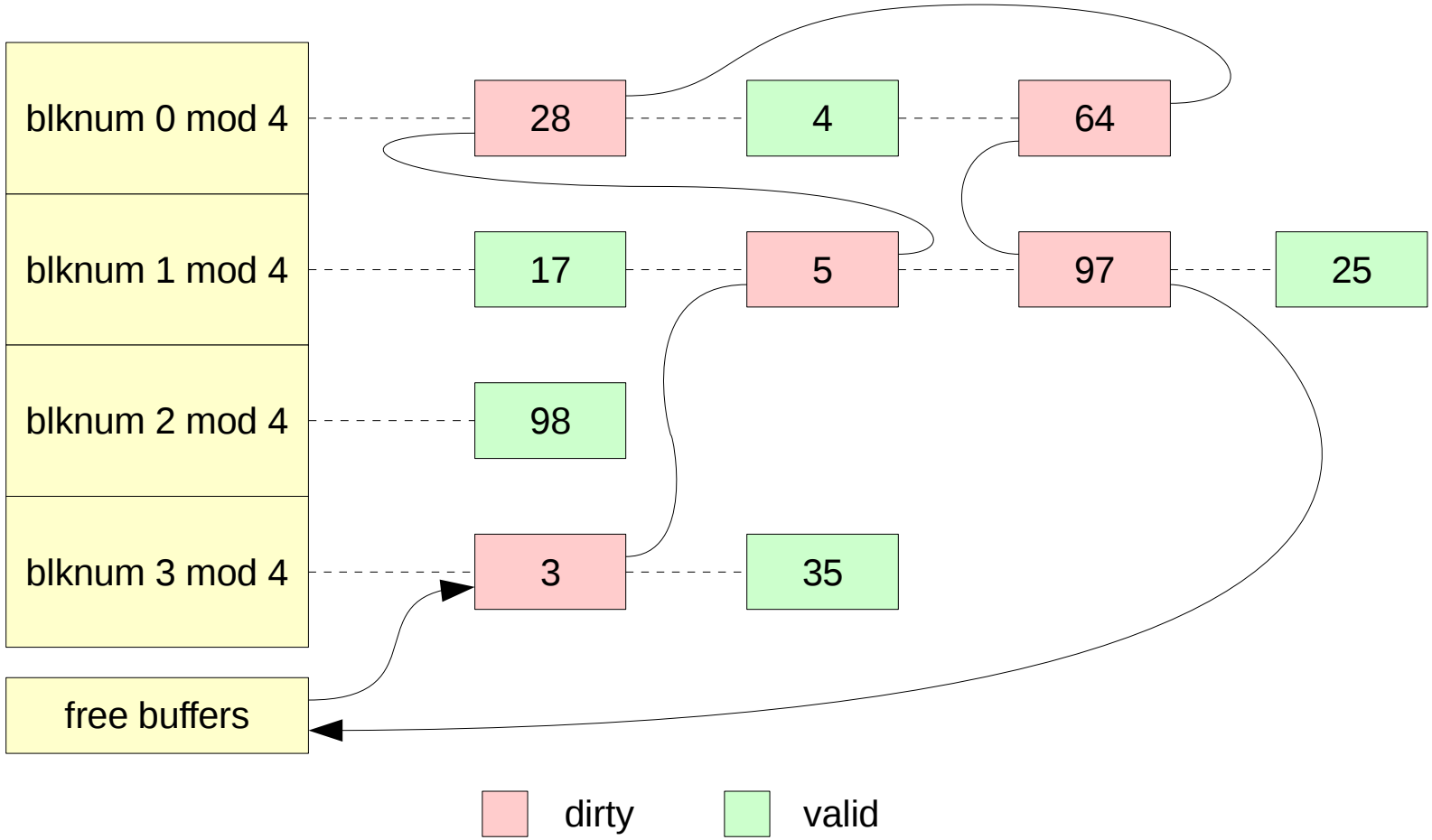


Figure: The buffer cache subsystem.

Agenda

Perspectives

High Performance Computing

- Manycore architectures
- Heterogeneous architectures
- Reconfigurable platforms

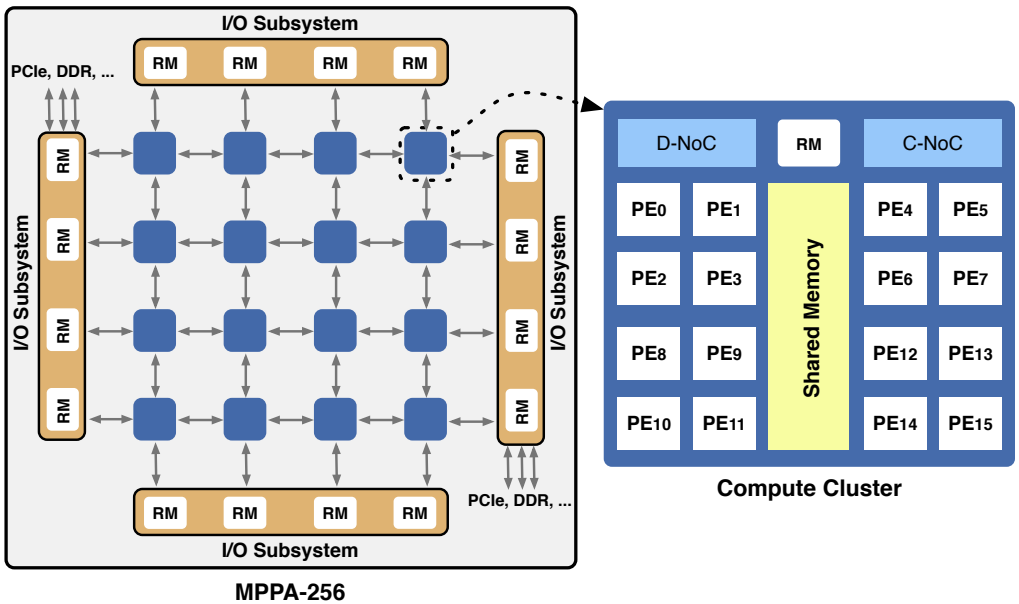


Figure: Kalray's MPPA-256 manycore processor

Perspectives

High Performance Computing

- Manycore architectures
- Heterogeneous architectures
- Reconfigurable platforms

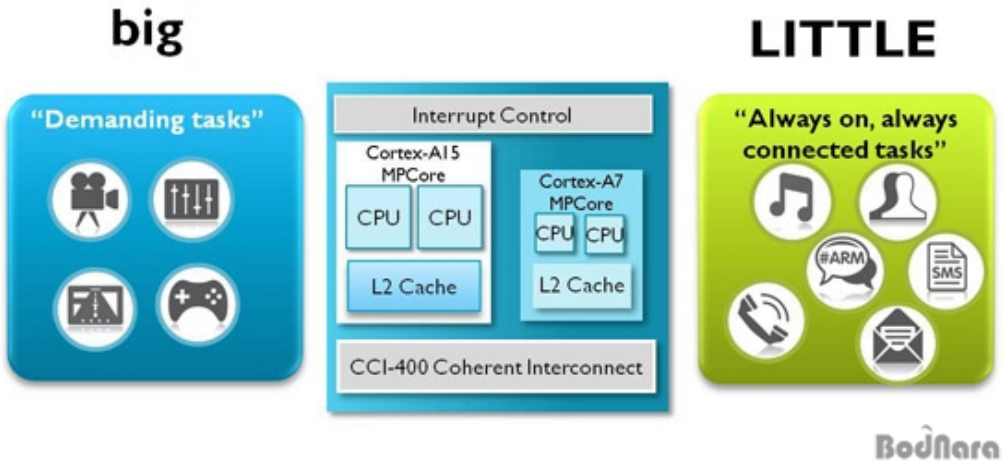


Figure: Kalray's MPPA-256 manycore processor

Perspectives

High Performance Computing

- Manycore architectures
- Heterogeneous architectures
- Reconfigurable platforms

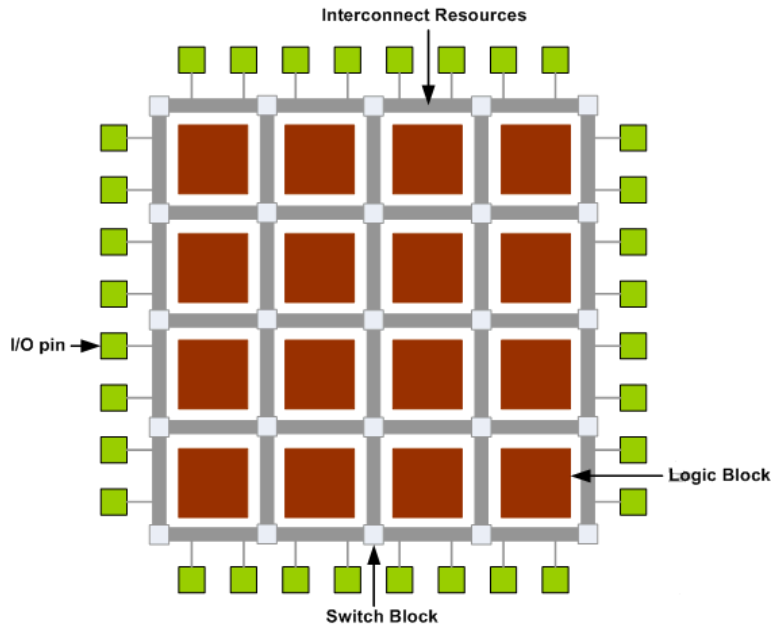


Figure: Kalray's MPPA-256 manycore processor

Perspectives

Manycore Architectures

- Enhance the memory management subsystem
  - Mini-regions

Perspectives

Manycore Architectures

- ~~Enhance the memory management subsystem~~
  - ~~Mini-regions~~
- Port user-level software
  - Newlib C Library
  - Assembler, Compiler & Linker

Perspectives

Manycore Architectures

- ~~Enhance the memory management subsystem~~
  - ~~Mini-regions~~
- Port user-level software (ongoing)
  - Newlib C Library
  - Assembler, Compiler & Linker
- Add multithreading support
  - Kernel threads

Perspectives

Manycore Architectures

- ~~Enhance the memory management subsystem~~
  - ~~Mini-regions~~
- Port user-level software (ongoing)
  - Newlib C Library
  - Assembler, Compiler & Linker
- Add multithreading support
  - Kernel threads
- Add message passing support
  - Light-weight MPI implementation