Linux Kongress 2008

Scalable and Practical OpenSource Embedded Systems

History

1

Embedded Linux 1998-2008

Back Story and Scope

- Embedded Linux emerged in about 1998
- Widely deployed, well understood and mature
- noMMU uClinux and MMU Linux 2.0-2.6
- Everything from DVD player to Cell Phones

- No Unified "Distribution"
- Very Portable

Linux in Embedded Systems

- Functionality Fit
  - Networking.
- Very large footprint
  - compared to a Commercial RTOS
- Potential to do much more
  - Single function devices, Yes
  - Platform Capability

Commercial Systems

- Everyone builds their own
  - Usually provided by the Silicon Vendor
- Functionality is remarkably similar
  - Classes of device and a few major Apps
- Impossible to count, or even know where used
  - We think in 100s of Millions units
  - Largest Linux installed base by far
- Changed Embedded Systems Engineering

Common Design Practice

From Reference Design...

- Provided Cookie Cutter
- No product customization
- Vendor doesn't know!
- Fixes, GPL, Quality....
  - Forget it!

From OpenSource...

- Huge Barrier
- No consistent process
- Engineering commitment
-
- Mailing List...
  - “How do I make my project, I have 2 days so tell me what I need and I really need you guys to help me right now so send it to my email I don't read this mailing lis

2

Next Generation Platforms
Hardware and Software

Open Systems Objectives

- Freedom for the Engineer
  - Control of the Platform and Application
  - Code Freedom: To build what you need
  - Design Freedom: To build it the way you want
- Freedom for the User
  - Freedom of Use: Make product do what you need
  - Code Freedom: Change the product
- Freedom for the Community
  - To Benefit from Past Works

Design Examples

- Examples of Project Scope:
  - Phone
  - DVR
  - VoIP VPN Firewall Router
  - NAS SetTopBox
- Examples of Projects out of Scope
  - Web Server Appliance
  - ...

Software Stacks

- Well Understood Practice*
  - Upstream projects have this well covered
  - Standard POSIX like environment
  - Not necessary to go into here
- Standard but limited Application Packages
  - Functionality driven, not “Kitchen Sink”

`* Linux is what comes after C -Kenneth Albanowski`

Portability

- Minimum Requirements for Kernel Support
  - 32 or 64 Bit Address Space and Data types
  - Periodic Interrupt
  - GCC
- About 2Mbyte memory
- Only the memory requirement is unique
  - Most modern SoC has a capable CPU

CPU Families

- All Standard Families are Supported
  - ARM series, MIPS, SH, m68k/Coldfire
- Architecture Specific code
  - In Kernel
    - setup, entry, interrupt about 2950 Lines C + asm*
  - In libc
    - syscall interface and bit operations
    - setjmp/longjmp

* v850 uClinux implementation

Initial Targets

- Criteria: Minimum Requirements and Openess

- Simple Target: Plasma MIPS
  - VHDL model and C simulation
  - CPU, Memory and 24 bit hardware timer
  - ~1500 FPGA LUTs
  - ~600 lines of C code for simulator

•Just an Instruction Chewer

```
void cycle(State *s)
{
...
   opcode = mem_read(s, 4, s->pc);
...
   s->pc = s->pc_next;
   s->pc_next = s->pc_next + 4;
...
   switch(op)
   {
      case 0x03:/*JAL*/    r[31]=s->pc_next;
      case 0x02:/*J*/      s->pc_next=target;           break;
      case 0x04:/*BEQ*/    branch=r[rs]==r[rt];     break;
      case 0x05:/*BNE*/    branch=r[rs]!=r[rt];     break;
      case 0x06:/*BLEZ*/   branch=r[rs]<=0;         break;
      case 0x07:/*BGTZ*/   branch=r[rs]>0;          break;
      case 0x08:/*ADDI*/   r[rt]=r[rs]+(short)imm;  break;
      case 0x09:/*ADDIU*/  u[rt]=u[rs]+(short)imm;  break;
...
   }
}
```

Initial Targets

- Criteria: Minimum Requirements and Openess

- Modern Target: LEON2/3 SPARC
  - VHDL model and C simulation
  - CPU, Memory and full peripheral set
  - Full SoC with MMU and SMP in FPGA
  - Cycle accurate Simulator and CoSimulation

Initial Targets for Reference

- UserSpace will recompile anywhere
- Can make a Gate Array ASIC SoC cheaply
  - Even low volume FPGA based system
- Proves noMMU/MMU and Endian issues

- Porting to a “commercial” SoC is trivial

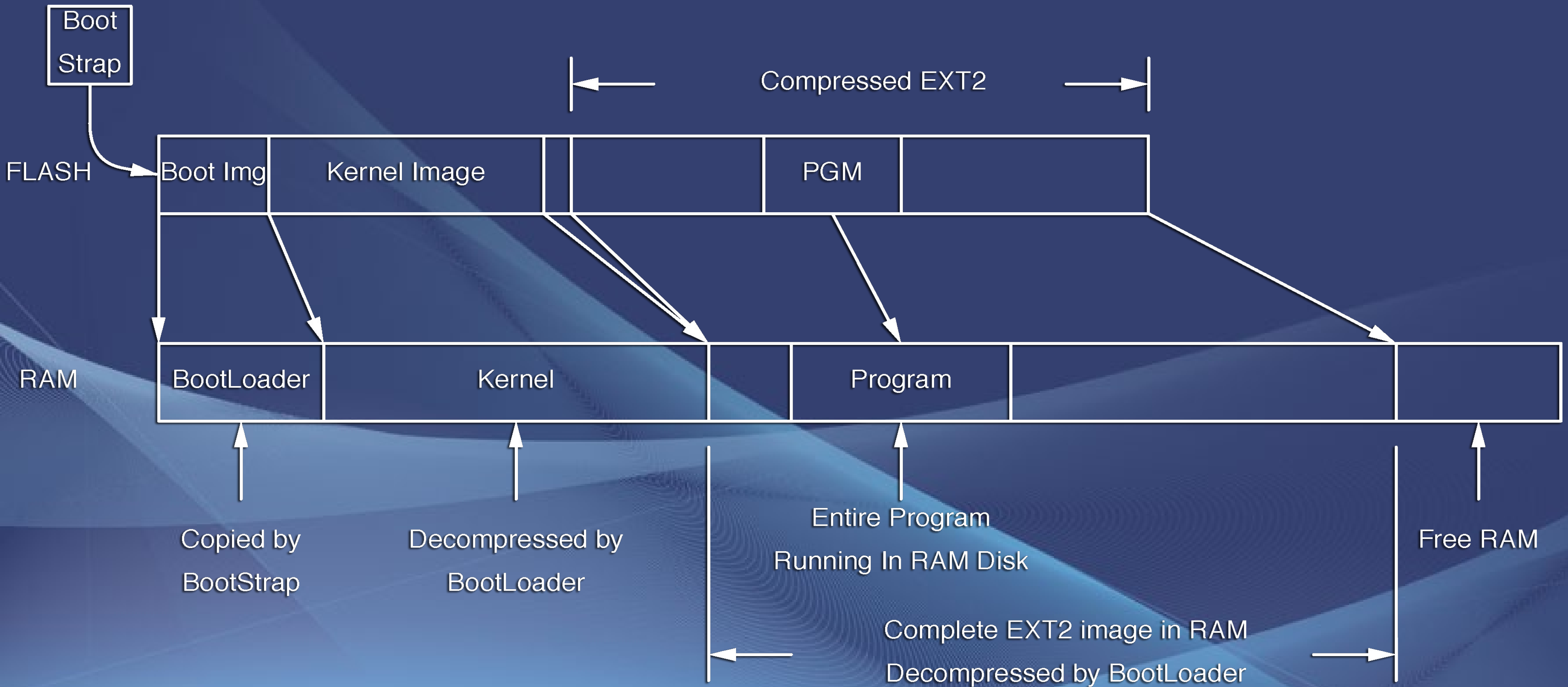
Hardware Examples

- Realtek MIPS: Similar to Plasma
  - MIPS R3000 style CPU with timer
  - Customer Peripheral set
  - Very successful commercial platform
- SH3: Similar to LEON SPARC
  - RISC instruction set with DSP extensions
  - Custom peripheral set
  - SoC platform

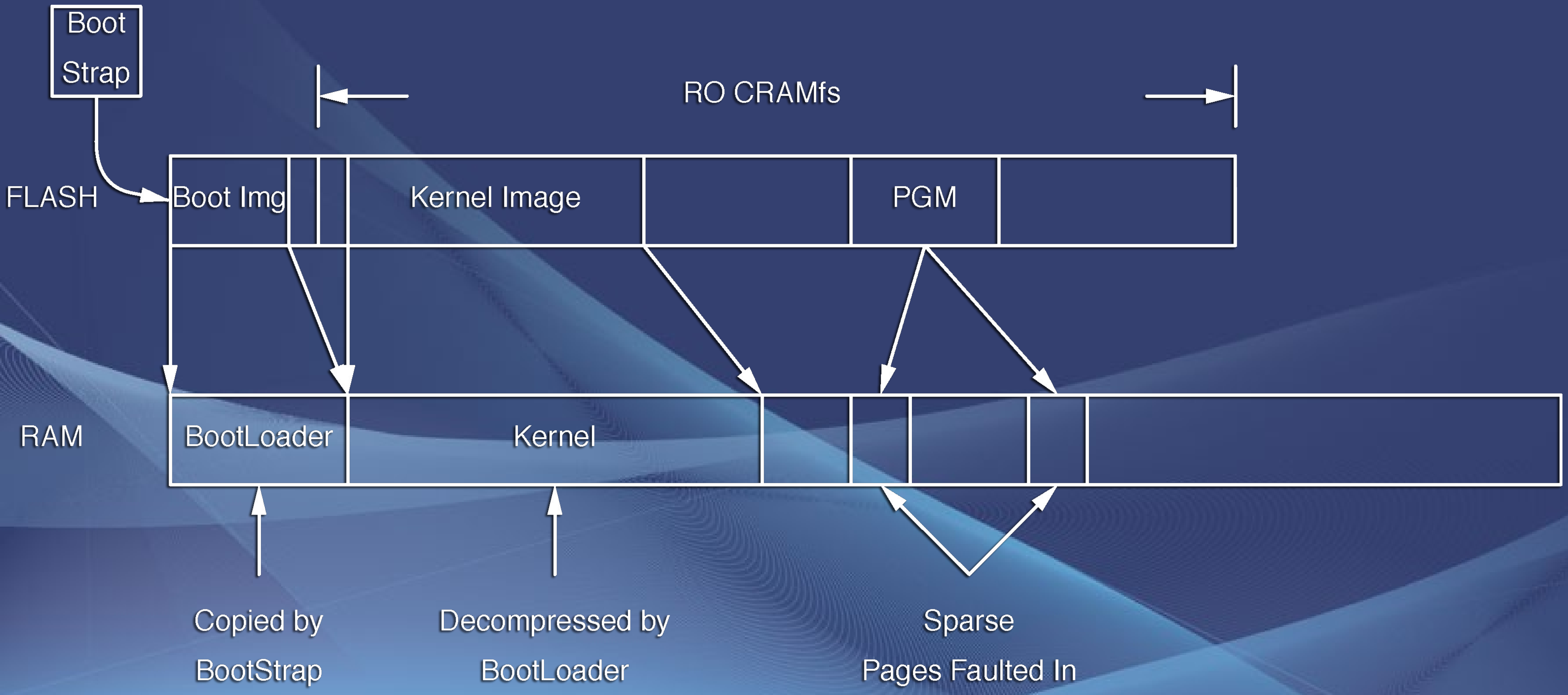
Technology Comparison

SH3+DSP	LEON SPARC SMP	Lexra MIPS	Plasma MIPS noMMU
233 MHz, cache, XY mem	50-400 MHz, 7 stage pipe	190 MHz, low complexity	36 MHz, Soft Core
Separate DSP Engine	1-4 SMP Cores + DSP inst	DSP in MIPS machine code	Single Core, no DSP
$8 SoC	$18 ASIC	$5 SoC	$12 FPGA
Wired. Dual Eth VoIP terminal router	Wired/Wireless. Dual Eth VoIP base-station	Wired/Wireless Base Band Consumer VoIP Router	Wired. Flexible App Specific Controller
4Mbyte NOR FLASH 8Mbyte RAM	2Mbyte Serial FLASH, SD Card, 64Mbyte RAM	2Mbyte NOR FLASH 16 Mbyte RAM	2-4Mbyte Serial FLASH 16-64 Mbyte RAM

Memory Management - MIPS



Memory Management – SH3



Reusable Basic Platforms

- Open Hardware Platforms
  - 400k Gate FPGA board with 10Base-T
    - 2 Layer board, designed as a reusable module
    - Plasma SoC Prototype platfrom
    - Directly integrate into low volume projects/products
  - 1.6M Gate FPGA module with 2x 100Base-T
    - 4 Layer module
    - LEON2/3 SPARC SoC Prototype Platfrom
    - Module Form Factor for direct integration also
- Direct path to eASIC SoC implementation

2 Overall Objectives

- A Basic set of Platforms: HW and SW
  - Open and reusable as basis for future work
  - Serves as a benchmark and a starting point
  - Easily accessible and available
- A technical solution to Vendor Participation
  - An answer to "why should we, we can just take it"
  - A way for the community to benefit from the work
  - A technically compelling case

Reusable Code

3

Software Architecture

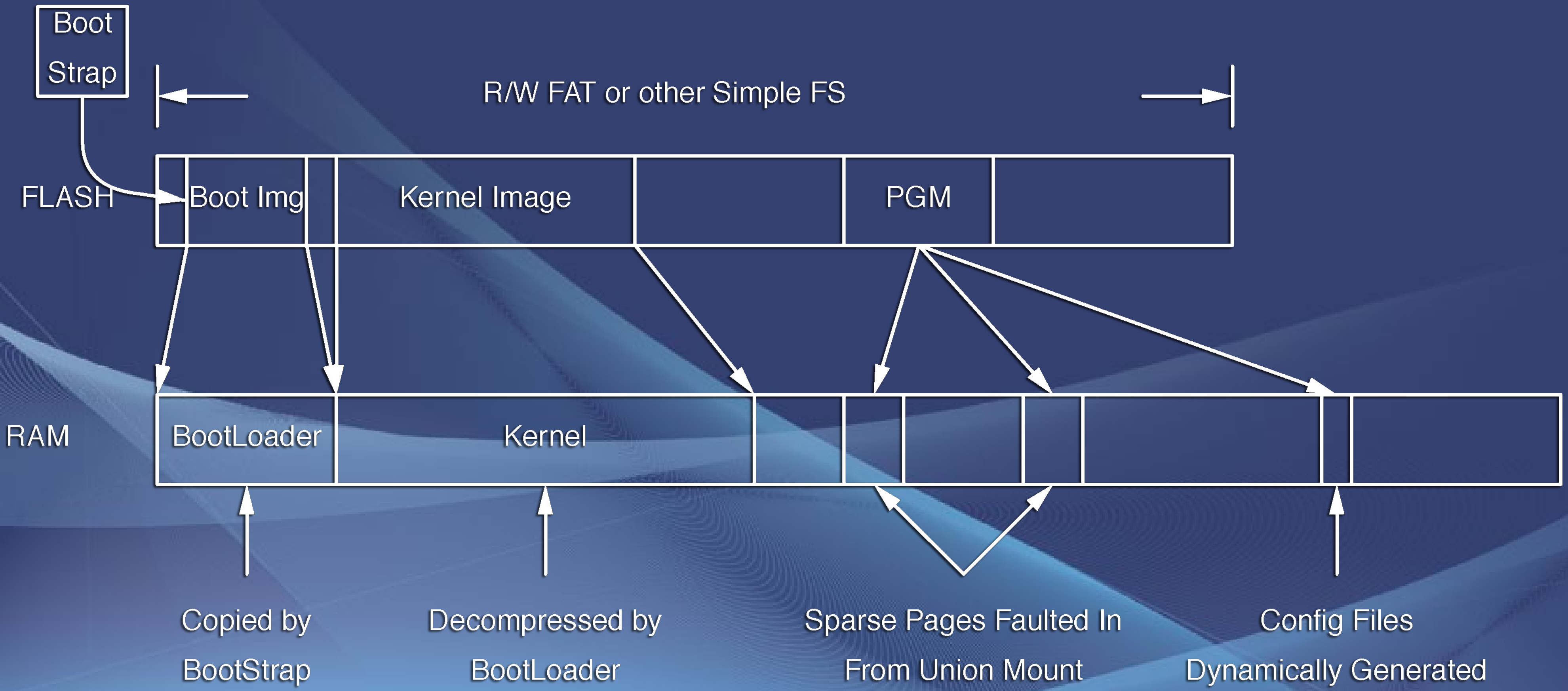
Unified Approach

- BaseOS Layer
  - Provide a standard Bundle of Functionality
    - Miniature POSIX like environement
    - Well Documented
    - Very portable and self contained
  - Packages for Berkeley Networking
    - Basic Networking
    - Routing and Storage layers
  - Management Framework
    - Configuration database and filesystem overlay

Code Storage and Execution

- Storage as a set of components
- Digitally Signed
- Decompressed on-the-fly
- Sparse Pages in RAM, page cache backed
  - On uClinux, executables loaded at runtime
- Pluggable Application Layer

Memory Management

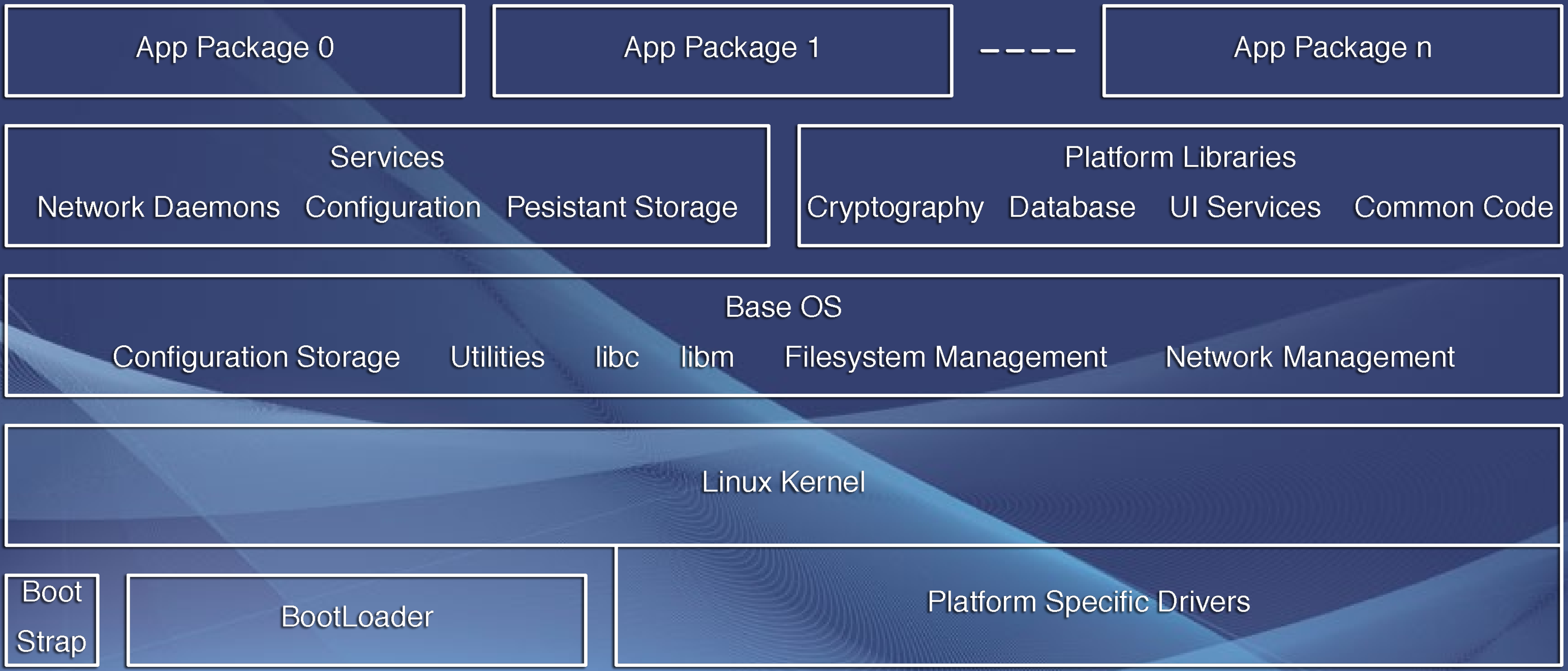

Boot Strap
R/W FAT or other Simple FS
FLASH
Boot Img
Kernel Image
PGM
RAM
BootLoader
Kernel
Copied by BootStrap
Decompressed by BootLoader
Sparse Pages Faulted In From Union Mount
Config Files Dynamically Generated

Embedded as a Platform

- User is in control of his/her device
- Loads whatever they want
  - Mangement/Configuration automatically integrates
- “Officially Supported” 3rd party Applications

- Basic Functionality of the OS guaranteed
  - BaseOS provides the standard Platfrom

User Space Architecture



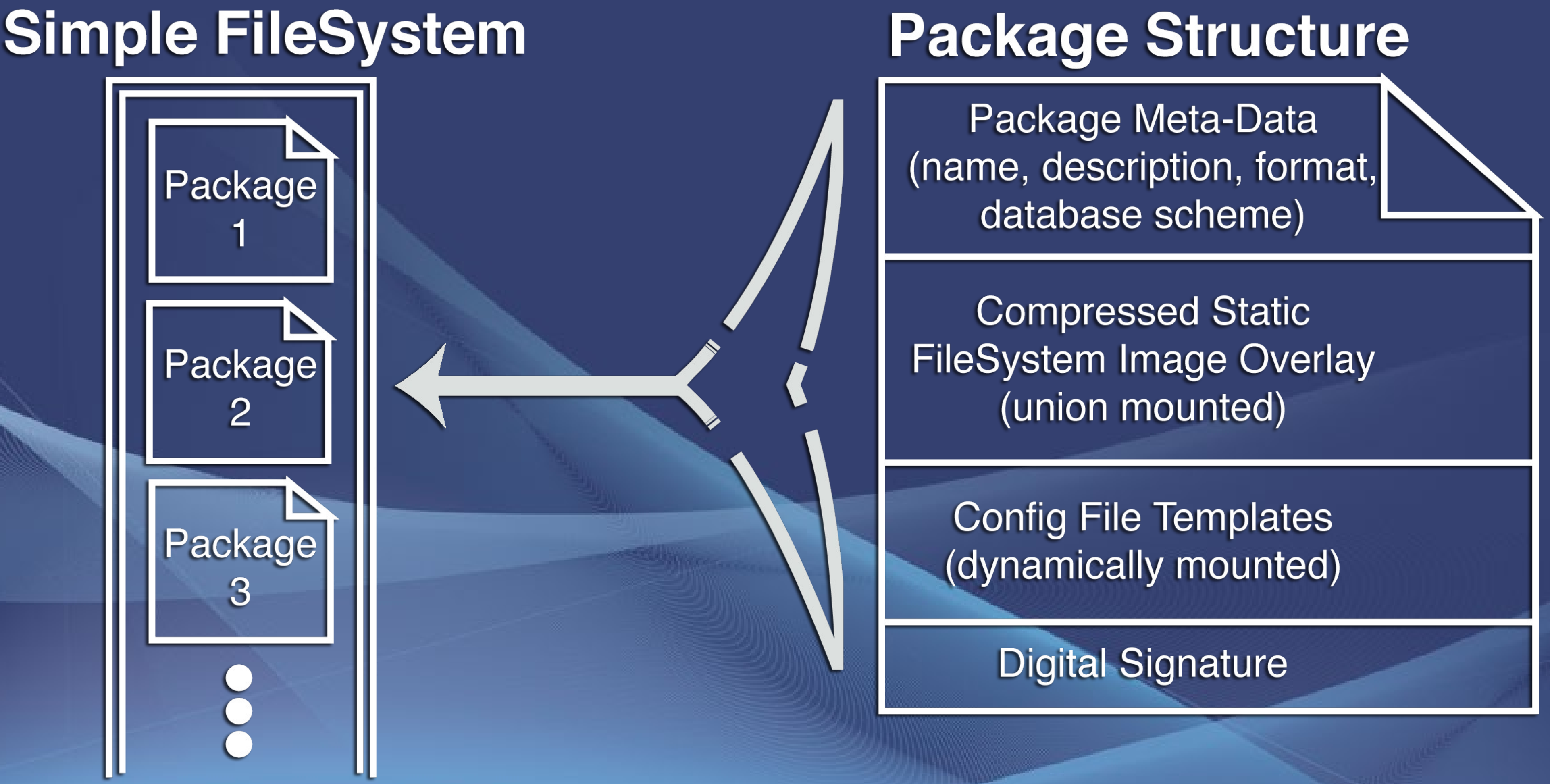
Blocks, Not Bricks

- Single Filesystem is dangerous
  - Update with incompatible package -> Brick
  - Install malware -> Brick
  - User Error -> Brick
- Storage of Atomic Components
  - Bootloader support for flaging broken Blocks
    - Fail-to-boot blame and recovery
  - Read-Only Component parts

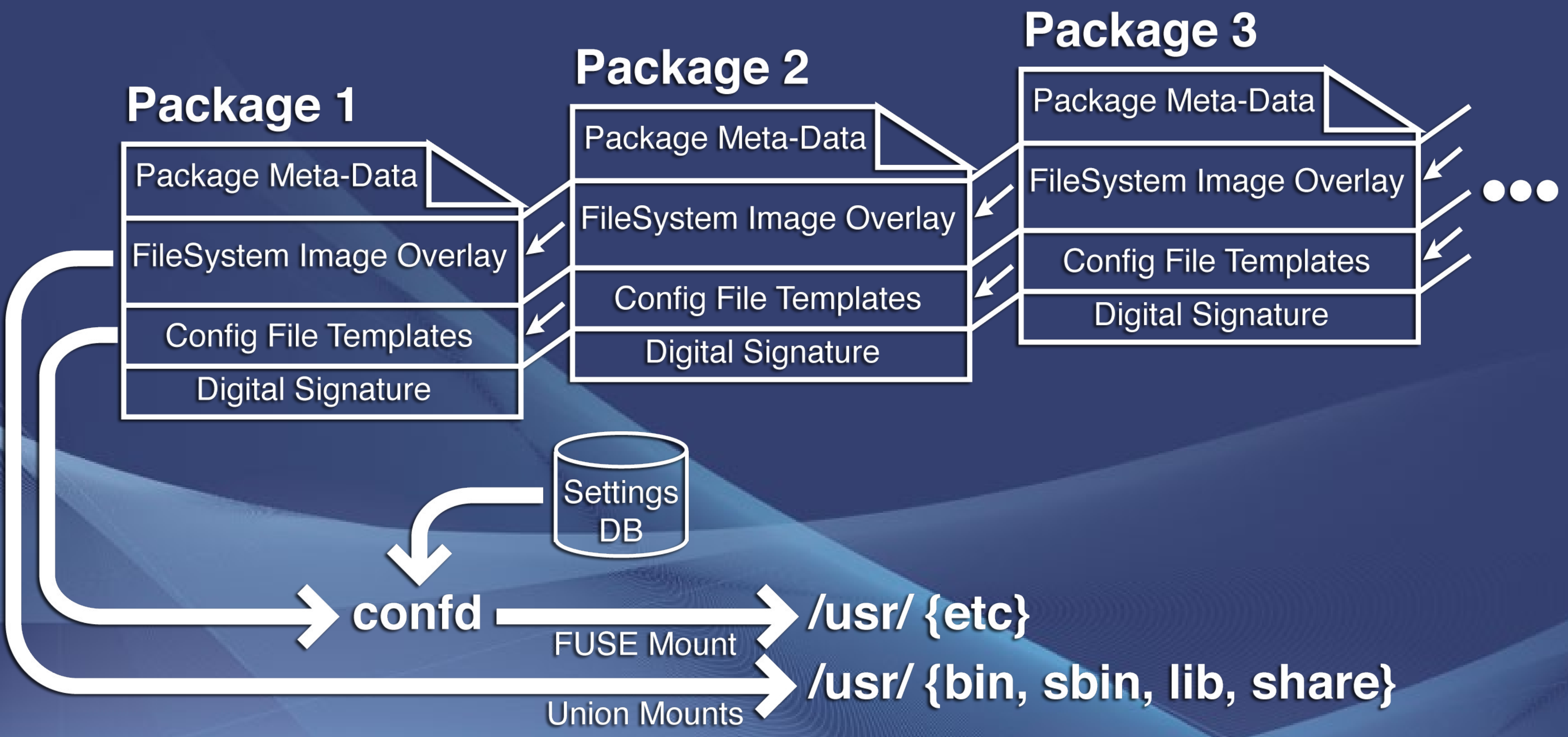
The Package Concept

- Basic Unit of Storage
  - Concept: PalmOS Even Apps are db records
    - Functionality is containerized
    - Safety through Digital Signature
- Package is Self Contained
  - Concept: Mac OS X Bundles or OpenOffice Docs
    - Self identifying and atomic
- Dynamic Integration
  - Concept: Registry or Management Information Base
    - Standard MIB2 Layout. eg. RFC1213 for TCP/IP

Packages in FLASH


Simple FileSystem
Package 1
Package 2
Package 3
Package Structure
Package Meta-Data (name, description, format, database scheme)
Compressed Static FileSystem Image Overlay (union mounted)
Config File Templates (dynamically mounted)
Digital Signature

Dynamic Runtime Filesystem



DB Backed File Generation

- XML Syntax like JSP, stored tokenized

```
for tag looping test
<ctl:for var=i start="10" stop="13">
<ctl:for var=j start="10" stop="${$i}">
i has value <ctl:out value="${$i}"/>
j is <ctl:out value="${$j}"/></ctl:for></ctl:for>
------
EL arithmetic test
(7+3)*5 = <ctl:out value="${ ( 7 + 3 ) * 5}"/>
-7+3*5 = <ctl:out value="${ -7 + 3 * 5}"/>
------
Set and if test
<set var="val" value="true"/><out value="${$val}"/>
<if test="${$val}"> Taken! </if>
------
MIB Namespace
<mib:get var=”val” oid=”SysUpTime”/>
System Running for <out value=”${$val}”/>hrs.
```

Specification and Implementation

- OpenSouce and Portable
  - Bootloader support
  - Dynamic FileSystem Implemetation
  - Packaging Utilities
  - Cryptographic Utilities
- Specifications
  - Documentation
  - Test Suites
- Example Build Kit

Projects and Participation

4

Vendors, Engineers and Community

Community Building

- A Community Framework
  - Not just Project Components, also Process
- Vendor a part of the process, not just feed off it
- Upstream Synchronization
  - No more years-old known bugs in products
- OpenSourceEmbedded

3 Parties, Views and Objectives

OSE.org – Generating Community Effect

- Public Project hosting (web front-ends) necessary for community involvement.
- for the various projects that have gone public with code and documentation.
- for Vendors showcasing their OpenSource based products
- provides the technical and end-user documentation

3 Parties, Views and Objectives

OSE.net – Software Engineers Collaborating

- Like a sourceforge, with project management tools
- Place to host the code, but with various legal frameworks for public release
- Provides access to NDA material to engineers
- provides the distribution mechanism for the GPL distribution services of the .com entity

3 Parties, Views and Objectives

OSE.com – Getting the Vendors to Contribute

- Corporate entity legally able to sign NDA's and other contracts
- Standard engagement method with services and point of contact for Silicon Vendors, ODM's, and OEM's.
- Standard tree of code as a starting point, and to promote “platform” unification
- Provides GPL distribution and License Compliance
- Provides Project Management tools and services
- Access to Engineers, or lets Vendor's engineers work with the structured web tools

The uClinux Experience

- A “kit” approach works for vendors
- Community Building requires a code base
  - Easily accessible, non threatening
  - A clear set of goals that mesh with engineer's needs
- Engineers Contribute
  - Vendor organizations don't (generally)
- Sponsorship is a bust
  - The project needs to be Vendor Agnostic
    - Not even a preferred Silicon Vendor

OpenSourceEmbedded

- New Home for 3 parties

- OpenSourceEmbedded.org
  - Users
- OpenSourceEmbedded.net
  - Community
- OpenSourceEmbedded.com
  - Vendors

Process Flow

- Vendors contract with the OSE.com entity
- OSE.com entity works with Engineers
- Engineers create code & projects on OSE.net
- OSE.net publishes

- Vendors have a page at Vendor.OSE.com
- Customers get code and updates at OSE.org

GPL and Vendors

- Vendors have no incentive to participate
  - So we have to give them one.
- Vendors are only interested in shipping products
  - OpenSourceEmbedded gives them scalability
  - OpenSourceEmbedded Provides compliance
- Enforcement is a secondary option
  - Use enforcement to bring vendors into the process

Getting Participation

- From Engineers
  - Status, Cool Code, Cool Projects and Jobs
- From Vendors
  - Scalable design process, High Quality Code
  - Silicon Vendors just want to sell chips
  - Need to Solicit individual ODMs
  - GPL Compliance Process can bring Vendors in
- From End Users
  - App Store!!!!

Upstream Projects

- Pull code directly from the O.S.E Repositories
  - Complete Coherent BaseOS and Basic Apps
  - Embedded Specific trees hosted at OSE.org
- Become involved in the development process

  "...and I hear my application is used in some routers"
- No more Years Old bugs and Regressions
  - Tracking of upstream by package maintainers

- GPL Compliance by design

Thank You

Questions and Discussion

http://dionne.ca/lk2008