openSUSE® on ARM / AARCH64

Alexander Graf
openSUSE ARM Team
agraf@suse.com

Dirk Müller
openSUSE ARM Team
dmueller@suse.com

Vladimir Botka
Product Manager
vbotka@suse.com
openSUSE® Runs on ...

... your laptop

... your desktop

... your server
Is There More?
(open)SUSE® Runs on ...

155,656 x86_64 Cores with 300 TB of RAM
SUSE® Runs on ...

9728 ia64 Cores, 30 TB RAM
openSUSE® Also Runs on ...

2880 Power7 (ppc64) cores
SUSE® Runs on Mainframe

IBM zSeries, z9
Nothing More?
What About openSUSE® on This?

CuBOX—ARMv7 800MHz, 1GB RAM
Or This?
Or openSUSE® on This Little Fruit?

35 US$, 700 MHz armv6, 256 MB RAM

(We're talking about the one on the left side)
openSUSE® on This?
Build your own supercomputer out of Raspberry Pi boards

Summary: Who says you need a few million bucks to build a supercomputer? Joshua Kiepert put together a Linux-powered Beowulf cluster with Raspberry Pi computers for less than $2,000.
What is ARM?
What is ARM?

• Most popular CPU architecture:
  - More than 30,000,000,000 CPUs are ARM based

• “Low power leadership”

• Optimized for “System on a Chip”
Faster is Better?

• High CPU power is not needed everywhere:
  - Static web serving/CDN, caching
  - Batch analytics / “Big data”
  - Cloud, dynamic web content serving (to some extend)

• ARM designs can be efficiently combined with performant I/O or Network I/O on one “SoC”
ARM-based Machines

- Tablets
- Tiny laptops
- Smartphones
- Netbooks
- Cloud nodes and Low-Energy Servers
ARM's “Cortex – A“ Series

- ARMv8 (A57/53)
- ARMv7 (A15/7)
- ARMv7 (A8/A9)
ARM v5/6/7/8

- VFPv2
- Jazelle
- ARMv5
- Thumb-2
- TrustedZone
- SIMD
- NEON Adv SIMD
- VFPv3/v4
- ARMv6
- ARMv7-A/R
- A32+T32 ISAs
- Key feature ARMv7-A compatibility
- AArch32
- Scalar FP (SD and DP)
- Adv SIMD (SP Float)
- AArch64
- Scalar FP (SD and DP)
- Adv SIMD (SP & DP Float)
- ARMv7-A
- ARMv8-A
- CRYPTO

Cryptographic processes:
- Key feature ARMv7-A compatibility

- CRYPTO
openSUSE® and ARM
openSUSE® on ARM Team

Virtual team of technical experts from SUSE

Strong collaboration with openSUSE community and technology providers

Started in Q3/2011
openSUSE® ARM From the Beginning

First ARM platform enabled

First release for openSUSE 12.2 on ARM Highbank

December

2012

openSUSE ARM (port)

2013

openSUSE 12.2 ARMv7 tech preview
openSUSE® on ARM Today

- **openSUSE 12.3 ARM release**
  - April 10

2013

- **openSUSE AArch64 (port)**
  - March 5

2014

- **openSUSE 13.1 ARMv7 and ARMv8**

QEMU: open source processor emulator
openSUSE® on ARM Enabled Platforms
Building openSUSE® for ARM
Building openSUSE® in OBS

gcc
YaST2
Apache
openssl

open build service

DVD
Building in Open Build Service (OBS)

• Automatic Rebuilds
  - One small fix automatically rebuilds all packages and images that contain the code

• Rebuilds are reproducible
Reproducible Builds

WWW

Sand Box
Building a Single Package

Package sources are built in a clean environment based on a build description (.spec)
Building a Package on x86

All builds are done in a sand box:
- No network access
- No physical hardware access
- Minimal privileges

On x86 and PPC: we use KVM and Xen

On ARMv7:
- Initially we had no physical hardware that supported virtualization.
“Quick EMUlator”

• QEMU is an emulator that relies on Just in Time (JIT) dynamic binary translation to achieve good performance

• Supports full hardware emulation (optionally)

• Supports CPU emulation for many CPUs including ARM
Building a Package Emulated

- Very similar setup to building on native hardware

- Emulation is secure enough

**But:** Hardware emulation is quite slow
Build Time Comparison

- 5500+ packages, if only one machine used for building
Idea: Avoid Hardware Emulation

- Sand box is virtualized
- Host kernel can be reused
- Only the ARM build environment is running in an emulator

**But:** Everything in the build environment is run in the emulator
openSUSE® Build Time

- 5500+ packages, if only one machine used for building
Caveats with (CPU) Emulation

• For good emulation, all interfaces between host kernel and ARM target binaries need to be emulated
  - All needed syscalls
  - All needed ioctls
  - All relevant /proc/* files
  - “Special” communication paths like netlink

• QEMU is already quite good at that
  - We made patches to make it even better
Emulation Always Has Risks

- Misbehavior due to emulation bugs

- Testsuites are mandatory
Isn't there a better way?
Idea: CPU Emulation + Acceleration

• Many build binaries can be replaced with a host binary:
  - xz, gzip, bzip2, tar
  - msgfmt
  - grep, sed

• Components which support cross building can also be replaced:
  - “Cross” compiler
  - “Cross” linker
  - “Cross” rpmbuild
Build Time Comparison: “Big” Package

Build is mostly CPU bound:

- Qemu
- Cortex-A9
- Accel. Qemu
- x86
Build Time: “Small” Package

Build time is mostly I/O bound

- Qemu
- Accel. Qemu
- Cortex-A9
- x86
Build Time Comparison

- QEMU (user mode)
- QEMU (accelerated)
- Cortex A9
- x86

• 5500+ packages, if only one machine used for building
Building openSUSE for

- In January 2013, there was no AArch64 hardware generally available


Early Access to ARMv8 with Virtual Platforms

**Foundation Platform**

- Models ARMv8 architecture
- Linux application development for ARMv8
- Fully compatible with Linaro v8 images
- Quick and easy setup and out-of-the-box Linux host
- Totally FREE
ARMv8 Foundation Model

- Full system emulator
- Accurate CPU emulation
- Very good for finding software compatibility issues and general testing without access to hardware
openSUSE Build Time on FM

- 5500+ packages, if only one machine used for building
Idea: Use Cross Build

Build a package on one “host” architecture for a different “target” architecture
Cross Build Pros / Cons

- Easy to do for a few packages
- Several projects support cross build “out of the box”
- Others need extra patching and tweaking
- Some packages are really hard to cross-build

- Difficult to maintain:
  - Configuration checks, test suites
  - “Generator” tools during build
One Issue with Bootstrapping

- openSUSE depends OpenJDK's JVM for building many central packages
  - Documentation Tools needed for various projects
  - Java bindings part of various central libraries
Why Crossbuild is Hard

- Building everything needed for OpenJDK “cross” is very time-consuming
- OpenJDK one needs a working JVM for building
  - Bootstrapping OpenJDK natively is impossible!
Isn't there a better way?
(Re-) Use QEMU

- openSUSE ARM team had made good experience with QEMU for ARMv7

- User Mode emulation is good enough for building all packages

- Much faster than full emulation
So... Does It Build?

YES!
Does It Run?

**NO!**

*(not yet)*
Missing Pieces

• Booting

• Deployment
Booting on x86(_64..)

Firmware Bootloader + Grub 2

OS openSUSE™
Booting on ARM (32bit)
Booting

- Firmware is part of OS, not of hardware

- Sometimes hardware specific kernel

- Operating system with customizations
Booting on ARMv8 (AArch64)

OS openSUSE™
Plans for Linux Kernel

• Migration to “device tree” support
  – Single Kernel Binary can handle many different devices by reading a standardized machine description provided by early stage bootloader

• Kernel-default, kernel-lpae (ARMv7)

• Kernel-default (ARMv8)
Missing Pieces:

- Booting
- Deployment
openSUSE® ARM, and Kiwi
Deployment Challenges

• Most ARM hardware does not have a CD drive

• Single install media is currently not possible
  - Special bootloader for each SoC needed
  - Kernel is also often still device specific

• Extended KIWI with extra targets for ARM
  - “Generic” Chroot target
  - SoC specific u-boot based Appliances
Does It Run?

YES!

http://www.flickr.com/photos/alorenzi/6277701171
Raspberry Pi
Samsung “Chromebook”
BeagleBoard.org
Current Test Hardware

Pandaboard.org
Exynos 5 boards
ARMv8 Foundation Model

[ OK ] Started LSB: Configure network interfaces and set up routing.
[ OK ] Reached target Network.
[ OK ] Reached target Host and Network Name Lookups.
  Starting LSB: Start the sshd daemon...
  Starting System Logging Service...
  Starting /etc/init.d/boot.local Compatibility...
[ OK ] Started /etc/init.d/boot.local Compatibility.
[ OK ] Started System Logging Service.
  Starting System Kernel Logging Service...
[ OK ] Reached target Syslog.
  Starting D-Bus System Message Bus...
  Starting Command Scheduler...
[ OK ] Started Command Scheduler.
  Starting LSB: Network time protocol daemon (ntpd)...
[ OK ] Started System Kernel Logging Service.
[ OK ] Started Login Service.
[ OK ] Started LSB: Network time protocol daemon (ntpd).
[ OK ] Started LSB: Start the sshd daemon.
[ OK ] Reached target Multi-User.
[ OK ] Reached target Graphical Interface.

Welcome to openSUSE 13.1 - Kernel 3.11.6 (ttyAMA0).

linux login: root
Password: linux
Have a lot of fun...
linux:~ #
Anything Else?

• We're working on some other devices as well

You can help!

- Test our machine images
- Provide us test hardware
- Help us with missing pieces for your individual device!
openSUSE 13.1

- ARMv6, ARMv7 and AArch64 is available
- Ready-to-use images are available for a few boards
- More will be added over the coming weeks
Future
Deployment Improvements

• Modularize the openSUSE images
  – Offer one unified image part
  – Offer multiple device-specific bootloader/kernel images parts
  – Offer installation patterns via “image addons”

• Deploy over network / other methods
  – Better mass deployments (Cloud nodes?)

• YaST2 on first boot

• YaST2 installer support
openSUSE® and ARM

- (open)SUSE® is prepared for future devices and future ARM architectures
  - Device specific work is quite small
  - Reusable and easily adaptable

- Qemu-Acceleration layer provides
  - Performance advantages of host hardware for build
  - Can be reused for other architectures
    - Even without native hardware :-)

---

[openSUSE® logo] [ARM® logo]
SUSE® Linux Enterprise for ARM

- SUSE will invest into openSUSE on ARM

- On the Enterprise/product side we see AArch64bit as the potential breakthrough and carefully continue to watch the market and its dynamics

- SUSE works with partners on AArch64 and is open for opportunities
Thank you

http://en.opensuse.org/Portal:ARM

opensuse-arm@opensuse.org
SUSE®
We adapt. You succeed.
Unpublished Work of SUSE. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.