SUSE Raspberry business use cases

A Case Study

Gabriel Cavalcante <gabriel@ssys.com.br>
Rodolpho Pivetta <rodolpho.pivetta@ssys.com.br>
SSYS
About SSYS

• Founded in 2014.

• Expertise in Linux Enterprise in Brazil.

• Experienced professionals certified in all SUSE portfolio (SEA, SCE e SCI).

• Projects with SUSE Manager, SUSE OpenStack Cloud, SUSE Storage, SUSE HA with SAP Application and HANA etc.

• A software development company.
CASE 1 - GPA
monitoring warehouse shipment status
About the Company – Largest Retail in Brazil

GPA

Pelo poder de escolher

2092
Stores

49
Warehouse Centers

23 States
Brazil

+ 140000
Employees
About the project

- **Objectives**
  - Open web page, inform to dock operators the shipment status and the time to the next shipment
  - Open pages in kiosk mode, be simple to operate. Avoid operating and misconfiguration errors.
  - Force configurations in a decentralized way (no master)
  - Cascated automatic update configuration, give a chance to avoid updates.
Raspberry Pi 3 Model B

- Broadcom system on chip (SoC) with ARM processor and GPU.
- Anatel Regulation (Brazil specific, blue board).

Components:
- 64bit 1.2GHz Quad-core ARM
- 1GB of RAM memory
- SD Card
- 4 USB ports
- FullHD HDMI output
- Audio Output
- GPO Interface
- 100Mbps Ethernet
- 2.4 and 5 GHz Wireless B/G/N
Support

• Anatel Blue Board (required for Brazil)
• SUSE support to deliver enterprise grade Linux for Raspberry Pi
• Security and compliance for enterprise
• SSYS brings customizations to meet business needs
Infrastructure

- Warehouse WIFI – same to scanning guns
- Low power requirements
- No mouse/keyboard
Development Details

- Use a SuSE Image with a masterless SaltStack:
  - States to remove all unused packages and services (games, interface features, etc)
  - States to configure all the services (network, ntp, etc)
  - States to tuning the grub, specially to prevent data corruption (caused by forced resets)
  - States to install and configure Google Chrome as application opening a pre-defined URL from Salt Pillar
Image creation process

- Get a fresh image
  - Download official OpenSuSE/SuSE image for raspberry

- Install salt-minion and apply states
  - Install salt-minion, configure as *masterless* and apply a series of states to prepare the image

- First Boot Process
  - Remove control files, fill all the free space with zeros and power off
Raspberry roadmap usage

- Initial usage in PoC (Proof of Concept) on OpenSUSE – 30 units

- Next Steps:
  - Scale to 300 units
  - Orchestration using Saltstack with grains and pillars to manage specific functionality (control URLs and more)
  - SUSE Linux Enterprise for new devices
  - SUSE Manager to orchestrate and consolidate management
Kiosks on Warehouse
CASE 2 - Digital Signage for Retail
About CASE2

- Big Retail Company in Brazil
- ~290 stores
- ~1M clients/day
- > 20,000 employees
Project Conception

• **How it was:**
  • They’re using an Open Source Digital Signage Solution – Xibo
  • Using Windows as clients (the only xibo digital signage client option)
  • ~120 TVs

• **Objectives:**
  • Reuse old point of sale computers (nettop)
  • Get a xibo client in SLED
  • Manage with SuSE Manager + Salstack
  • Scale to 600 TVs
Project Execution – Phase 1

- **Python Xibo Client was developed (using mplayer as video engine)**
  - Only with video and image capabilities
- **Kiwi was used to create a customized SLED Image**
  - SuSE Manager join on first boot
Project Planning – Phase 2

- Use Raspberry with SUSE as digital signage clients
- Use an external wireless antenna to boost raspberry signal capacity
- Use Salstack + SuSE Manager
- Use higher video resolutions (nettops had an old Atom chip, cannot decode 1080p videos without buffering).
- Port Python Xibo Client to use omxplayer
- Adapt Python Xibo Client to open Web Pages
Project Execution – The KIT

- Mini-keyboard
- Raspberry Pi 3 B+
- External USB Antenna
- HDMI Cable
- SD Card
- Power Adapter
Project Execution – Wireless Bonding

• The bonding module was used to keep connection always alive swapping through internal and external interfaces
  
• A third interface (wan0) was created keeping just one MAC Address and swapping between the interfaces (wifi1 and wifi2)
  
• The bonding configuration ensures that raspberry keep connected even if the external interface (antenna) is removed
  
• udev rules restart network services if a new adapter is detected
Project Execution – Wireless Bonding

wan0
- no wireless extensions.

wifi1
IEEE 802.11 ESSID:"OSHI"
Bit Rate=24 Mb/s Tx–Power=31 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Power Management:on
Link Quality=64/70 Signal level=-46 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:0 Missed beacon:0

wifi2
IEEE 802.11AC ESSID:"OSHI" Nickname:"<WIFI@REALTEK>"
Bit Rate:434 Mb/s Sensitivity:0/0
Retry:off RTS thr:off Fragment thr:off
Power Management:off
Link Quality=99/100 Signal level=100/100 Noise level=0/100
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:0 Missed beacon:0
Project Execution – Wireless Bonding

cat /proc/net/bonding/wan0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: wifi2 (primary_reselect always)
Currently Active Slave: wifi2
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 1000
Down Delay (ms): 200

Slave Interface: wifi1
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 1
Permanent HW addr: b8:27:eb:5b:cb:09
Slave queue ID: 0

Slave Interface: wifi2
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 74:da:38:4c:67:0b
Slave queue ID: 0
SUSE Linux for ARM

• Only 64 bits (aarch64) version
  • Feasible to open Web Pages and show images
  • Problem to play videos (no GPU acceleration)
• Upstream firmware available with all functions only in 32 bits (armhf)
• Unfortunately we used Raspbian to PoC video play
Main problems and limitations

- No omxplayer compilation for aarch64;
- No kernel support for V4L2 libs;
- No kernel API for openmax/mmal (required by omxplayer)

- Hope? Kernel 4.19?

**gentoo-on-rpi3-64bit**

Bootable 64-bit Gentoo image for the Raspberry Pi 3 Model B and B+, with Linux 4.19, OpenRC, Xfce4, VC4, camera & h/w codec support, profile 17.0, weekly-autobuild binhost
Development Details

- **Configuration panel and Status Dashboard developed with PyDialog**
  - Enabled with keyboard shortcuts
- **Saltstack orchestration and deployment**
  - States to remove unnecessary packages and services, configure network, ntp and other configuration files
- **First Boot config**
  - Remove control files, fill all SD free space with zeros and power off. In first initialization without control files a First-Start Settings panel appear, allowing tech make the basic configurations
- **Hack configuration on RealVNC**
  - ExperimentalRaspiCapture enables “direct capture mode” to allow to view video output layer directly from VNC
Troubleshooting

- pydialog create a status dashboard with debug information
- Simple to identify configs and network information
Players
Use Raspberry with Icinga2 to monitor env

- Use Raspberry as a IoT device and Icinga2 to monitor environment
- Use Arduino/raspberry sensors to get data
- Develop sensors plugins with python
- Examples:
  - Soil humidity sensor module
  - Flame sensor Module
  - Microphone sensor Module
  - Photo resistor sensor Module
  - Temperature sensor Module
  - Air humidity
  - etc
Improve Raspberry for digital signage

- **Measure and Collect audience reactions:**
  - Through image recognition
  - Through Bluetooth device scanning
Technical questions? With coffee please..

gabriel@ssys.com.br
rodolfo.pivetta@ssys.com.br
Unpublished Work of SUSE LLC. All Rights Reserved.

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. 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 SUSE LLC. in the United States and other countries. All third-party trademarks are the property of their respective owners.