montavista VISION 2007

EMBEDDED LINUX DEVELOPERS CONFERENCE

Appropriate Community Practices: Social and Technical Lessons

Deepak Saxena

Kernel Architect

dsaxena@mvista.com

Presentation Contents Licensed under the

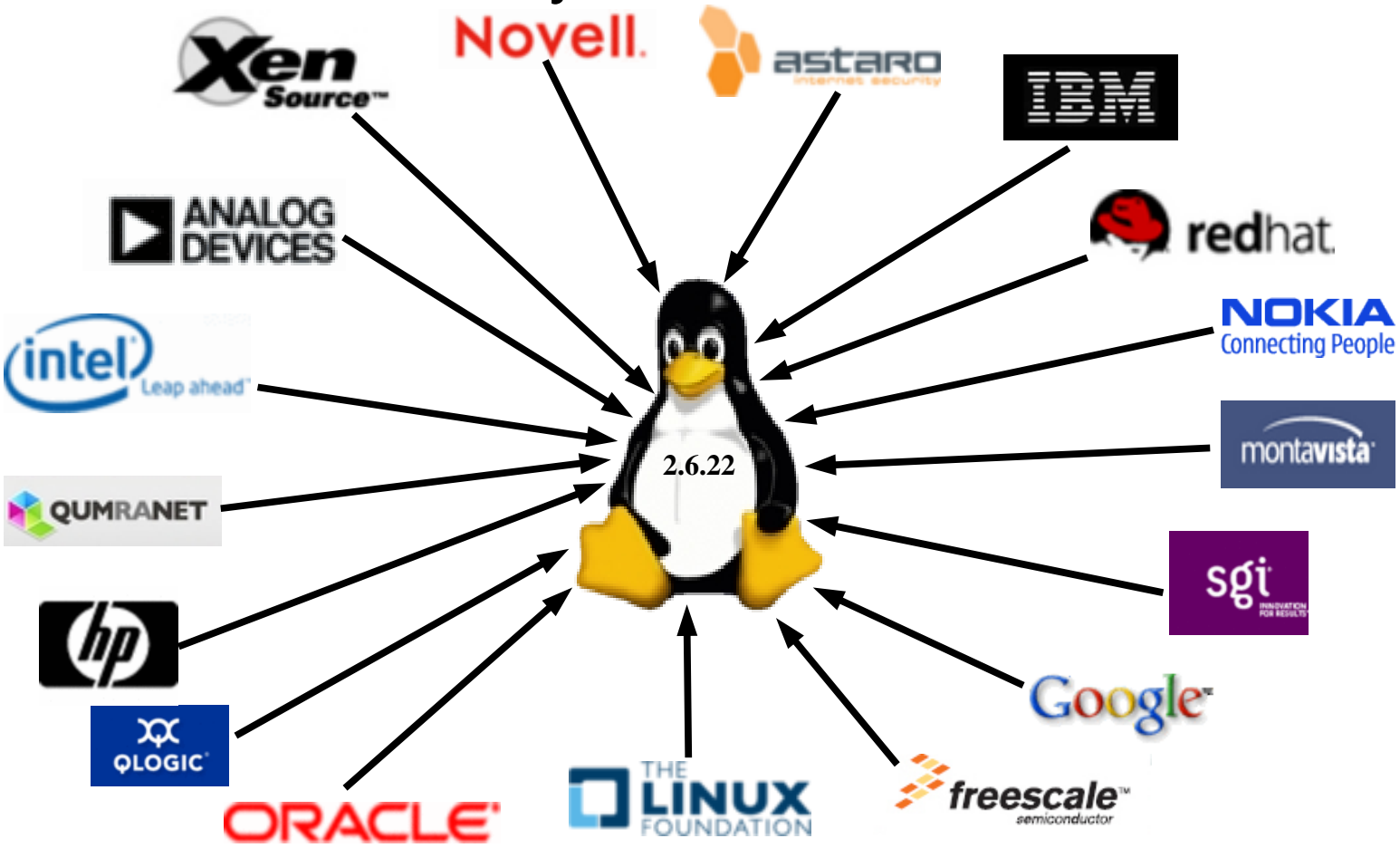
Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

montavista

Introduction

- Many new people getting involved with Linux development
- Same mistakes made over and over
  - Social
  - Technical
- Today's Goals:
  - Provide some pointers to get started working with community

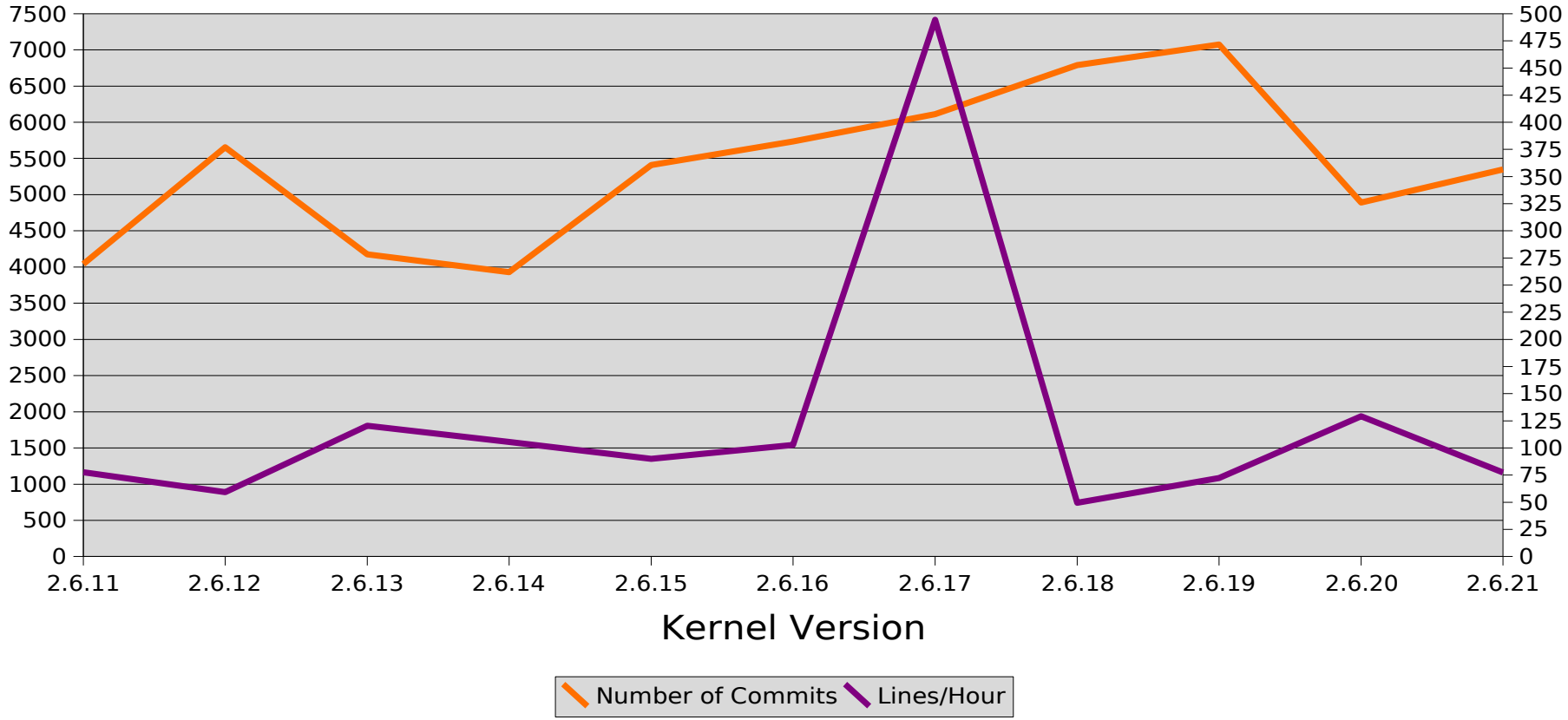
Who is this community?



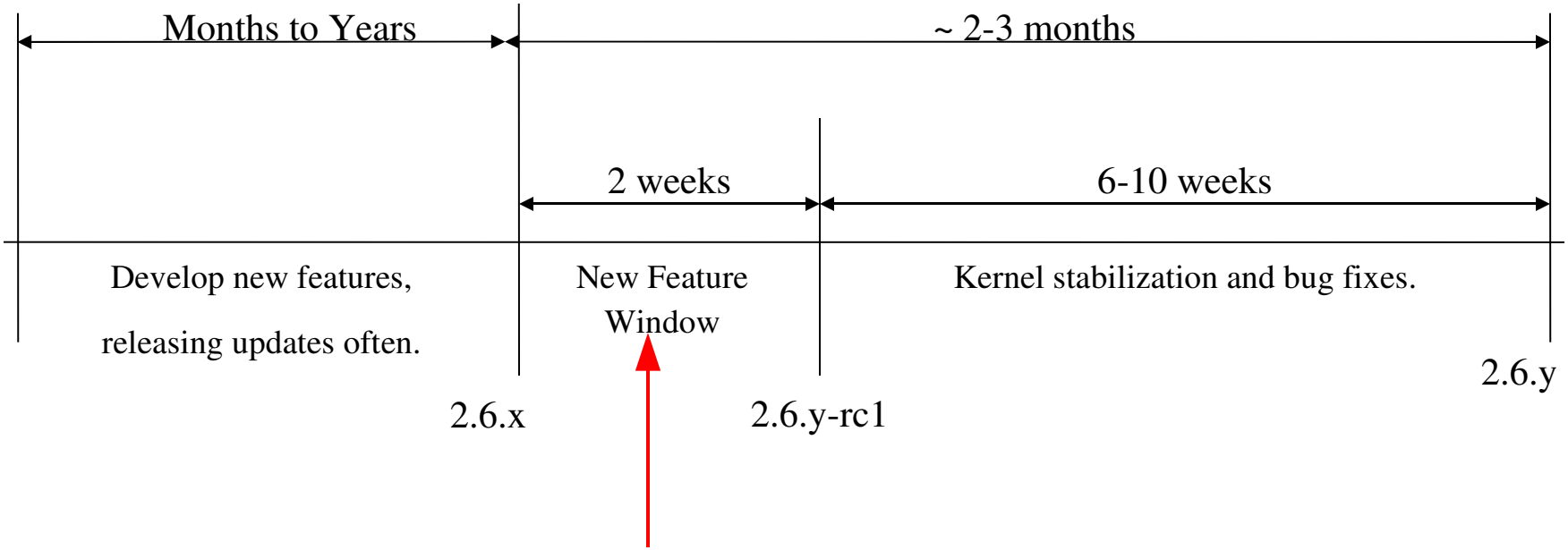
Logos Copyright © and Trademark (TM) their respective owners

Q: Why merge your code into kernel.org?

A: B/C you have to keep up with following:



Release Early, Release Often



Don't wait until this point to submit your code the first time and expect it to be accepted into the kernel.

Please Do Your Homework

- Many problems already have practical, known solutions in Linux
  - Driver APIs for specific functionality
  - User/Kernel interfaces and system-level tools
  - It may just not be obvious at first.
    - It may not be documented in one single place
    - It may just be tribal knowledge
- Google is your friend here
- Read the code
- Ask the experts
- **Release Early, Release Often to get design input**

Don't Abstract Unnecessarily

- HW vendors like to abstract code to make it work across Oses
- Developers often add wrapper macros around existing functions
- Too much abstraction makes code harder to maintain
  - Code is harder to read/debug
    - Abstraction layer might have bugs
  - Kernel API changes are hidden
  - Difficult for someone new to get involved
- **Release Early, Release Often to get feedback**

Don't Believe the Marketing Hype, Your HW is Not Unique

- Your HW may seem unique at first, however...
  - Others are close by or will soon follow
    - They will need same kernel support as you do
  - Don't blindly add hooks to kernel to support these features
    - We don't want multiple implementations of same idea
  - Need to add abstraction hooks at proper points in kernel
    - Work with external developers to develop these
    - Work with other HW vendors
- **Release Early, Release Often to get feedback**

The community is an extension of your team
Your team is an extension of the community

- Treat input from non-employees with same respect as employees
  - They are providing another valuable point of view
- Let your team interact with community members
  - Make community work part of their job description
  - Share data when needed
  - Don't create complicated processes for releasing code
  - Send your developers to key community conferences
    - Ottawa Linux Symposium, LinuxConf Australia, etc
    - Provides a chance to interact with community members
    - Builds relationships that help with code review process

Don't Use Intermediaries

- Who cannot answer all the questions b/c did not author code
- Who must go back and forth with the actual developer
- Who release patch to fix a customer issue but...
  - Don't explain how problem actually found or the test case
    - Other developers can't reproduce
    - Original assumptions may be wrong
    - You will probably just get ignored

Read The Following...

- Documentation/CodingStyle
  - Expected coding style for Linux
- Documentation/SubmitChecklist
  - Pointers of things to doublecheck before submitting code
- Documentation/SubmitingDrivers
  - High level guidelines and pointers for new driver authors
- kernelnewbies.org
  - Lots of advice for people new to Linux kernel world
- lwn.net
  - Tracks kernel API changes, interesting new developments, etc

Don't Be Arrogant...

"This is all part of what responsible release management is about. I was the junior whiz kid in professional release management teams before starting <unnamed company>. I listened to my elders and learned from them. My standards for professional conduct in this arena are higher than yours as a result of that. You are a bunch of young kids who lack professional experience in release management."

Summary

- Understand that you may not understand everything
  - Your code code may have ramifications you can't see
- Do your homework
  - Read documentation
  - Ask informed questions
- Participate openly and respectfully
  - Be part of the community, don't treat it as a hindrance
- Release Early, Release Often