מפגש בנושא:

שילוב שיטות עבודה של קוד פתוח

אמתי בורשטיין, Gizra
נדב הראל, Cloudius Systems Ltd.
גלעד בן-יוסף, איזיצ'יפ

22.10.2013

מלון דניאל, הרצליה

סדר היום	
13:45 – 14:15	התכנסות ודברי פתיחה
14:15 – 15:00	**כיצד לגשת לפרוייקט סגור בגישה וכלים של קוד פתוח – אמתי בורשטיין**
15:00 – 16:00	**מדוע חברות יוצרות קוד פתוח - נדב הראל**
16:00 – 16:15	הפסקת קפה
16:15 – 17:15	**שיתוף פעולה עם קהילת הקוד הפתוח ליצירת יתרון תחרותי, חקר מקרה – גלעד בן-יוסף**

Why Do Companies Write Open-Source Software?

Nadav Har'El, Cloudius Systems, Ltd.

October 22, 2013

For-profit companies write open-source.

- They start new open-source projects.
- They open-source software they already wrote.
- They contribute to existing open-source projects.
- They cooperate with competitors on this software.
- **But why?**
  - Are they crazy?
  - Are they communists?
  - Or are they actually being **smart**?
  - Does this make **business sense**?

The first good answer

- June 2002, **Joel Spolsky.**
- Entrepreneur & Blogger.
- http://www.joelonsoftware.com/articles/StrategyLetterV.html
- http://hebrew.joelonsoftware.com/Articles/StrategyLetterV.html

"**Smart companies try to commoditize their products' complements.**"

Complement product

- A product that you usually buy **together** with another product.
  - Gas and cars.
  - Computer hardware and computer software.
  - Flight and hotel room.
  - DVD player and DVDs.

Complement product

- Demand for a product **increases** when the price of its complements **decreases**.
  - If hotels in certain area become cheaper, more people fly there. And vice versa.
  - As DVD players become cheaper more people buy DVDs.
  - If gas was cheaper, more people would buy cars.
- So demand for your product will increase if its complements are cheap.

Demand for your product will increase if its complements are cheap.

So a company's best interest is:

- For its complements to be come **commodities** (many competing manufacturers).
- For its complements to be as cheap as possible.
- When free is possible, it is cheapest :-)

When **software** is your complement

- If you have a product which needs additional software, and
- If this software is not your competitive advantage.
- Then you have a **software** complement.
- You want it to be open source:
  - It will be cheap/free for your product's users.
  - If no such open source projects exists yet, by starting one you may share the costs of developing in-house.

Google's Android

- Good example to free-software complement: Google Android.

Why did Google do Android?

Why open-source?

They didn't **have to** go open-source (could have used BSD instead of Linux, like Apple did).

Google's strategy

- ACM SIGIR, July 2008

  Keynote by **Kai-Fu Lee,** Google VP Engineering, and President, Google Greater China.

- Talked about increasing Google's use in China (and the rest of the world).

Google's strategy

- Google's main problem in China wasn't **market share**, it was **market size**.
- You need a computer to use Google's products. Only a minority of Chinese had them.
- In the developed world, more **new** users had smartphones (iPhones, Blackberry) than computers.
- But these were too expensive for most Chinese.
- Google would benefit from Internet-capable mobile phones becoming a commodity!

Google's strategy

- So Google:
  - Bought Android (2005) and **open sourced** in 2007.
  - Convinced hardware makers to take Android to create more smartphones (Open Handset Alliance, 2007).
  - October 2008 – first Android phone (HTC Dream)
- As of 2013 – **strategy was successful**:
  - 1 Billion Android devices activated – all can use Google.
  - 75% of smartphones, 90% in China.
  - New Android-specific revenues for Google (market, ads...)
    - Even if everyone switched to Bing, new review streams.

Example #2 - Netscape

- Netscape Navigator came out in 1994. **Not free** for businesses.
- Threatened to commoditize the OS. Microsoft retaliates with IE in 1995.
- Netscape cannot compete. Tries to save its **server products** and **websites** by open-sourcing Mozilla in 1998.
- AOL buys Netscape. Incoherent strategy. Crash.

How to run the open-source project

- Two styles of open source projects:

Centralized: One person, group, or company, develops the code.

Accept bug reports and patches, but outside contributors always remain outside.

Company keeps control.

Distributed: Encourage other companies and individuals to join the discussions, decisions, and become equal contributors.

Company gives up control.

How to run the open-source project

- And a whole spectrum between the extremes.
- E.g., **Android:**
  - Based on Linux, on which **Google** has no control.
  - **Google** does control rest of Android's development,
  - ...but through “Open Handset Alliance”.
  - Other companies (e.g., **Samsung**) have Android for today's products, but rely on Google for its continuing development (beyond Linux).

Why **give up control**?

- Reduce your development costs.
  - Let other companies with the same complement help you.
  - They won't help you if they feel a conflict of interests.
- Faster development than a single company can ever do.
  - Linux grew faster than BSD because it encouraged many more people to contribute.

How to give up control?

- Open the source-code through a well-known impartial organization. E.g., **Apache Software Foundation**.
- Create a new organization. **Open Handset Alliance.**
- Use source-code repositories, domain names, etc., not associated with your company. E.g.,
  - **java.com**, not **java.sun.com**
  - Use **github.com** for the source code.

How to give up control? (cont.)

- Avoid in-house mailing lists, bug trackers, etc.,
- Have commit rights for non-employees.
- Do not require copyright assignment.
- Do not patent parts of the code – or offer royalty-free license.

Before giving up control...

- Is the code really your **complement** or your **product**?
  Netscape wasn't sure...
- Have you chosen the right license?
  - Afraid you'll want to turn this into a product:
    - Use BSD-like license, dual license, copyright-assignment.
    - Keep submarine patents...
  - Afraid this will turn into a competitor's product:
    - Use GPL license.

Example #3 – Red Hat

- Red Hat is a services company: You don't pay it for the software (OS, Virtualization, Middleware), you pay for:
  - Support
  - Integration
  - Packaging of well-QAed software
- More than $1 billion revenue in 2012.
- Red Hat services need good open-source software to exist. A **complement**? More like **raw materials**...

Corollary

Smart companies try to build on an open-source infrastructure.

- Complement – infrastructure the users have.
- Raw material – infrastructure the company has.

Anything which isn't the company's competitive advantage.

Red Hat

- So Red Hat works on dozens of open-source projects: Linux, KVM, WildFly (Jboss), Fedora, ...
- Uses all “give up control” techniques, even in projects it originated (e.g., Fedora, KVM), so:
  - Millions help Red Hat do QA (Fedora).
  - Companies like IBM participate in development.
  - **Reduced development cost** (compare: Unix).

The Unix Story

- In the end of the 20th century, many companies sold Unix-based workstations: Sun, DEC, SGI, HP, IBM, DG, AT&T, ...
- All started with 1980 7th Edition Unix or BSD.
- Each continued Unix development alone.
- Was never a competitive advantage: Nobody bought Sun just for SunOS.
- Today, such companies would adopt GNU/Linux.

Example #4 – Qumranet

- Small Israeli startup making a **remote desktop** platform.
- Needed to use a hypervisor, existing one (Xen) wasn't good enough, so wrote a new one: KVM.
- 2006: Decided to open-source KVM (into Linux):
  - Wasn't Qumranet intended product, but a complement:
    Qumranet's software needs a host with a good hypervisor.
  - Get more Linux developers to help with KVM.
    Qumranet was too small to continue on its own.
  - Get noticed.

Qumranet

A success story?

- Nice exit - bought by Red Hat in 2008, $107M.
- The “complement” KVM is much more popular than the original remote-desktop product.

“Contributing back”

Companies use existing open-source projects and contribute code back to the project. **Why?**

- The license requires them to (GPL).
- To avoid maintaining your patches.
- To show goodwill to other developers – and maybe get help or agreement from them later on.

Example #5 – Sun Microsystems

- Sun was a **hardware** company.
- Their best interest: make software a commodity.
- Opening Star Office (later Open Office) – **good.**
  - Though didn't really commoditize word-processor.
  - People still differentiate MS Office and Open Office.
- Opening Java – **business mistake!**

Example #5 – Sun Microsystems

- Sun was a **hardware** company.
- Opening Java – business mistake?
  - Java runs the same program on any computers.
  - Java commoditizes **hardware**!
- Sun hoped Java will increase the market for all servers, including Suns.
- It increased the market for all servers, but made it increasingly easier to switch to x86...

Example #6 – Cloudius Systems

- The company I work for.
- Small Israeli start-up.
- Developed **OSv** (http://osv.io/),
  - New operating system for VMs on the cloud.
  - Compatible with Linux, but smaller, faster, easier to configure.
- Open Sourced OSv.

Other reasons to write open source

- Small company wishing to create new software and get it accepted by customers.
  - “Don't be afraid the company will vanish, it's open source”
  - “It's free, try it!”
- How to profit?
  - Services or support? (you have the experts)
  - Proprietary product?
  - Dual license GPL/proprietary? (e.g, MySQL)
  - Your company gets bought? (Why?)

Other reasons to write open source

- Employee satisfaction
  - Programmers get world fame, not just company-wide fame.
  - Similar to publishing in the academic world.
- Peer review
  - Improve quality by having others outside the company review the code.
- Employee hiring
  - Attract open-source programmers.
    You know who to hire, because you've seen their code!

Legal issues to consider

Very little has been tested in a court...

- Copyright & copyright assignment
- Copyright infringement suits (SCO vs. IBM)
- Licenses
- Patents
  - What happens if you patent something you open-source?
  - What if a contributor patented his addition?
- Anti-trust laws. (careful when cooperating with competitors).
- Anti-dumping laws. (Fairsearch vs. Android)

Summary

- There are good reasons why your company should start an open-source project, or join an existing one.
- But there are also cases when this does not make business sense.
- So **do consider your business case**.
- Is this software your product?
  A complement to your main product?
  Infrastructure your product is built on?
  What will be your product a year from now?
  What can you lose by open-sourcing?
  What can you lose by **not** open-sourcing?

The Road Not Taken

A case study of using Open Source co-development to attain a commercial advantage (with a little help from Robert Frost

Gilad Ben-Yossef
Software Architect
EZchip Technologies

gilad@benyossef.com
@giladby

ILTAM USER'S ASSOCIATION

22 October 2013

Agenda

- **Introduction**
  - or who is EZchip and what does it do
- **The dilemma**
  - or how to integrate Open Source into your product
- **The road not taken**
  - or working in the upstream
- **Looking back**
  - or what has happened since

EZchip Overview

- Fabless semiconductor company, NASDAQ listed (EZCH)
- Leading provider of Network Processors (NPUs) to the Carrier Ethernet (CE) market, especially for edge routers
- EZchip is a strategic supplier to the top Carrier Ethernet vendors
- Recently announced the NPS, a revolutionary line of NPUs
  - For next generation L2-7 router line cards, data center and cloud
  - Enabling new scalable architectures for SDN & NFV
  - 256 C-programmable Task Optimized Processors, 4K virtual threads
  - Large set of hardware and algorithmic accelerations
  - Integrates EZchip's highly differentiated TM technology
- EZchip founded in 1999; 190 employees, 150 in R&D in Israel
- Global offices in Israel (HQ); San Jose, CA; Boston, MA; and China
- Strong financial model; $182M in cash, no debt

Enter the NPS

NPS-400 ® is a network processor that you program and manage like a general purpose processor.

- **N**etwork **P**rocessor for **S**mart Networks
- 400 stands for **400 Gbps**
  - 960 Gbps oversubscription
- **600 Mpps** at 64 byte frames

Yes, it runs Linux ®

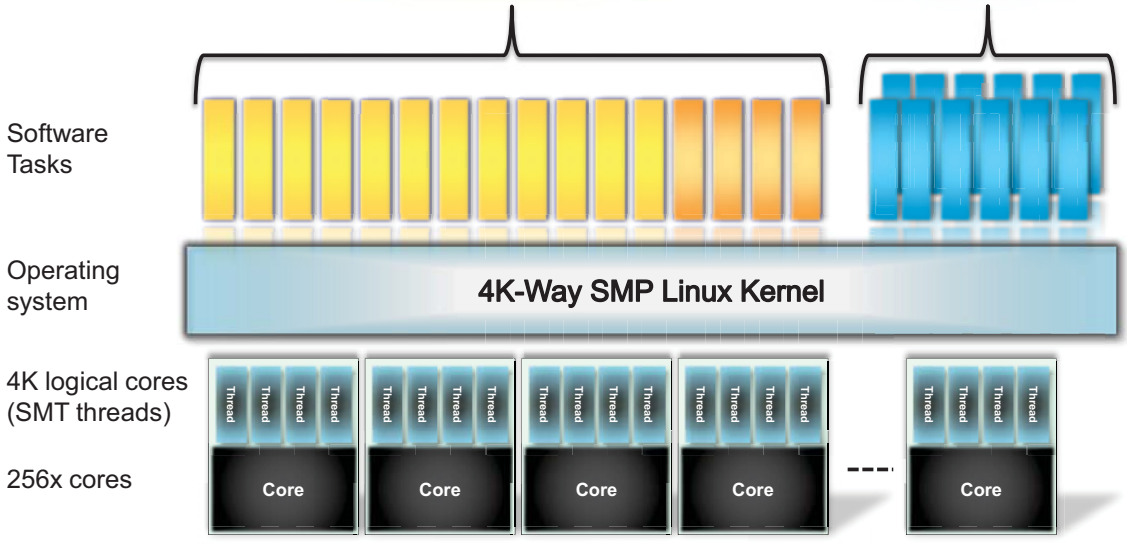
NPS-400 Main Features

- 400Gbps all layer C-programmable NPU
  - 600Mpps wire speed with up to 960Gbps oversubscription
- On chip Traffic Manager & VOQ
  - 1M queues, 5-level H-QoS
- 960Gbps of network I/O
  - Including 1GE, 10GE, 40GE, 100GE, 400G ILKN and PCIe Gen 3.0
- Integrated EFA (Ethernet Fabric Adaptor)
  - Enables a full line card on a chip
- 256xCTOP processors with 4K HW threads at 1GHz core speed
  - Native algorithmic instructions for efficient execution
  - 4K-way SMP Data Plane Linux, Run to complete architecture with no SW overhead
- Supports various inline services via dedicated HW accelerators
  - Security: IPSec/SSL encryption & decryption at 200Gbps
  - DPI: C-programmable RegEx stream content processing at 200Gbps
- On-chip TCAM with TCAM algorithmic extension to external DRAM
  - Scales to millions of ACL rules
- Based on commodity DDR3 & DDR4 memory providing up to 96GB
  - Virtually unlimited tables, states, counters at wire-speed performance

Data Plane Linux®

Data Plane
Single Linux process per logical core
Process fully monopolizes CPU

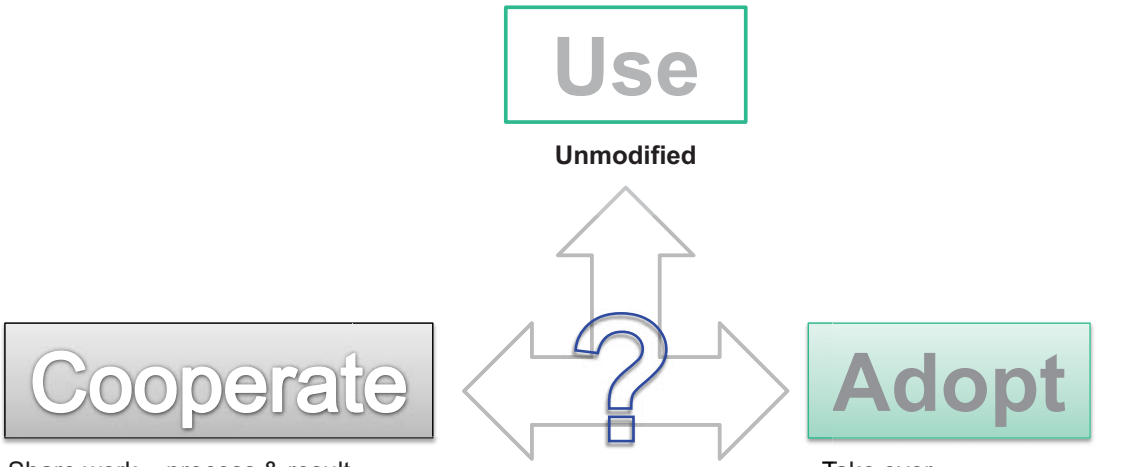
Control Plane
Many processes
Software schedulable



The Dilemma

"Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;"

Use, Adopt or Cooperate?



Share work – process & result
Enjoy external contribution
Assume stewardship
Future proof your work
Work as part of the community

Take over
Assume full responsibility
Maintain secrecy, full IP
Subject to license
You are on you own

Pros and Cons

Pros

- No first mover advantage
  - Previous existing implementation of concept
- License limitation
  - Core kernel is GPL v2
- Not a core skill set
  - EZchip is a silicon vendor, not a software house
- Pre-existing low key community efforts
- Ongoing maintenance is an issue
- Does not expose core IP

Cons

- Secrecy
  - Product was not announced yet
- Reception of community unclear
  - Multiple attempts to introduce similar features failed in the past
- Schedule constraints
  - Open Source software ships when it's ready, we have a schedule

A bit of history: real time support

- Many out of tree contenders
  - RT-Linux
  - I-Pipe / Adeos nano kernel
  - Monte Vista Linux
  - …
- Last man standing
  - PREEMPT-RT
- History shows that niche specific features that are developed as out of tree extensions reach a dead end
  - Especially when commercial vendors view their out of tree status as commercial advantage
- **Merge in or die out**

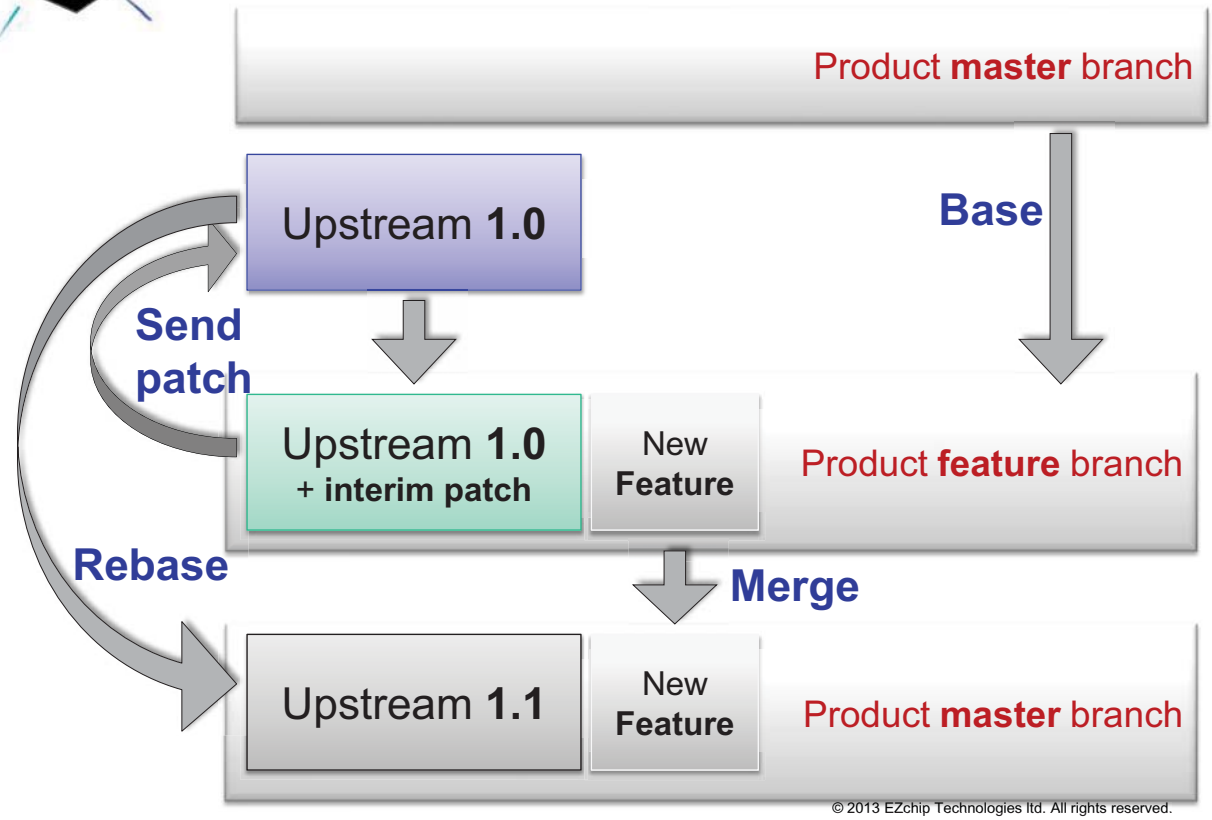
The Road Not Taken

"Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same"

Upstream workflow example

- Vadim, **feature owner** (13/10/2013): "*And what about rpc-json? Can we change its code for set port value as 0 and save it value for our purposes? It is GNU code*."
- Eyal, **manager** (14/10/2013): "*It is ok to change if it is something we have no other alternative for, but we need to do it through the upstream (i.e. within the original open source project). Noam – you should lead the activity vs. the open source community…, Vadim – in the meantime you can do a temporary local solution so we can proceed."*
- Noam, **OSS owner** (18/10/2013): "*We are upstream. Vadim please get update.*"

Working in upstream workflow



Advices for the road

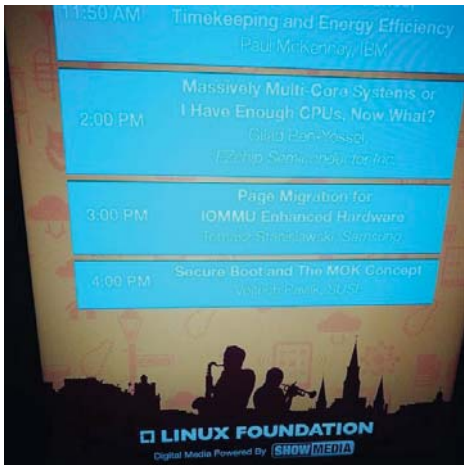
- Do your research
  - Who tried this in the past and why did they failed? What can you do differently? Who might be interested?
- Start early and small
  - Do not work alone for 6 months and than drop the 254 lines patch unto the mailing list
- Communicate well
  - Be ready to defend your changes from the communal standpoint: why the community as a whole needs the change, not why you need it
- Do the work
  - Send patches, respond to criticism, supply measurements, act diligently on bug reports
  - Attain karma – previous good community service does wonders to good will

Looking Back

"And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back."

Two years later…

- Engaged top tier developers across many companies
  - IBM, RedHat, Intel ..
- Received substantial feedback, assistance and testing
- One quasi-competitor with previous implementation shared his code with us
- With 2 part time developers EZchip got the work worth of a cadre of top notch field experts
- Community took ownership of feature, protecting it from future changes, lowering maintenance costs by order of magnitude
- Customer interaction made it clear that upstream integration is a major boon



The End

"I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference."

Thank you for listening and sincere apologies to Robert Frost.