Concurrent Open Source Development Projects

Multiple Devices & Device-Mapper RAID in the Linux Kernel

Lars Marowsky-Brée

Architect Storage and High-Availability, SUSE

lmb@novell.com

Introduction and motivation

- I needed a topic for my master’s thesis (Information Systems Management at the University of Liverpool).
- I was the team lead and project manager in charge of the storage & HA & kernel.
- Device-Mapper RAID versus MD RAID somehow always annoyed me.

Methodology

- Literature & theory review
- E-Mail survey (expert interviews)
  - Psychological and sociological aspects hard to measure
- Analyse MD and Device-Mapper kernel- and user-space
  - History
  - Reasons given
  - Quantitative comparison of development artifacts
  - Convergence options
- Applicability to other projects

F/OSS & academia

- Three beneficial aspects:
  - Transparency
  - Communal reflexivity
  - Proximity to academic process (peer review et al)
- Three basic streams of research:
  - Motivations of an individual contributor
  - Governance, organization, and innovation process
  - Competitive dynamics

What had been studied so far?

- Contribution to F/OSS as opposed to building the same software in a proprietary context
- Some research into the dynamics of forks
  - “Right to fork” is actually one of the four freedoms in the GPL.
- But why do people choose one project over another to contribute to?
  - Why do they choose their own instead of contributing to an existing one?
  - What is the impact of these decisions?
  - Can they be mended, if needed?

Why people contribute to F/OSS

- Complex mesh, but some common themes:
- Achievement
  - “Scratch an itch”, career advancement, flow experiences, “homo ludens”, self-determination, personal needs
- Affiliation
  - Altruism (totally confuses researchers!), social integration, community identification, gift culture, helping behaviour
- Power
  - Reputation, maintainership

Possible impact of divergence

- Research into different directions
- N-version programming, heterogeneous ecosystem
- Broad coverage
- Arrive at best of breed
- Positive cross-pollination (if license allows)

- Community split & reduction
  - Developers, testers, documentation, users
  - Evaluation overhead
- Effort duplication

Reasons for divergence

- Technical arguments:
  - Significantly technical differences
  - Old code base is cruft
  - License disagreement
- Sociological arguments:
  - Disagreement with maintainer
  - Difficult community interaction
  - Governance model

Ease of satisfying motives (examples)

Motive	Contribute to existing project	Start new project
Achievement (+)	Can leverage existing infrastructure.	Does not need to bother with existing infrastructure Lack of modularity in existing one. Can satisfy just one quick issue.
Achievement (-)	Fear failure of a new project.	Fear that the contribution would not be accepted.
Power (+)	Gain more status on existing project.	Control new project.
Power (-)	Avoid “power struggle”.	Fears loss of control over contribution to existing project.
Affiliation (+)	Join and become a member of a large community.	Stronger, focused personal recognition within smaller community.
Affiliation (-)	Fear being cast out from community when starting a new project.	Avoid a controversial discussion in existing environment.

Estimating F/OSS development costs

- Nobody documents the time they spend on the projects.
- So estimates work by proxy and try to deduce the cost from the resulting code.
- COCOMO-II:
  - Complex parametric model
  - Calibrated with data from proprietary projects
  - Based on a snapshot, not incremental
  - Vastly diverse contributor community
  - Yet not entirely useless as a point of reference

Some COCOMO-II estimates

- Red Hat Linux 7.1: ~$1 billion (Wheeler, 2002)
- Linux kernel: ~$612 million (Wheeler, 2004)
- Linux distribution: ~$10.7 **b**illion (LF, 2008)
- Linux kernel: ~$1.3 **b**illion (LF, 2008)
- Linux kernel: ~$1.4 **b**illion (Garcia-Garcia & de Madgaleno, 2010)

Which costs are missing?

- Total cost of ownership:
  - Training
  - Choice between many alternatives
    - None of which may meet all requirements
    - Differences not always clearly documented
  - 3rd party documentation efforts
  - Incompatibilities when moving from one to another or exchanging data

Interviewees

- SCSI maintainer
- Maintainer of the dm RAID project
- Maintainer of the MD project
- Key Linux contact at a large storage vendor
- 3rd party contributor
- Issues:
  - Low response rate
  - Small sample
  - Difficulty due to e-mail medium

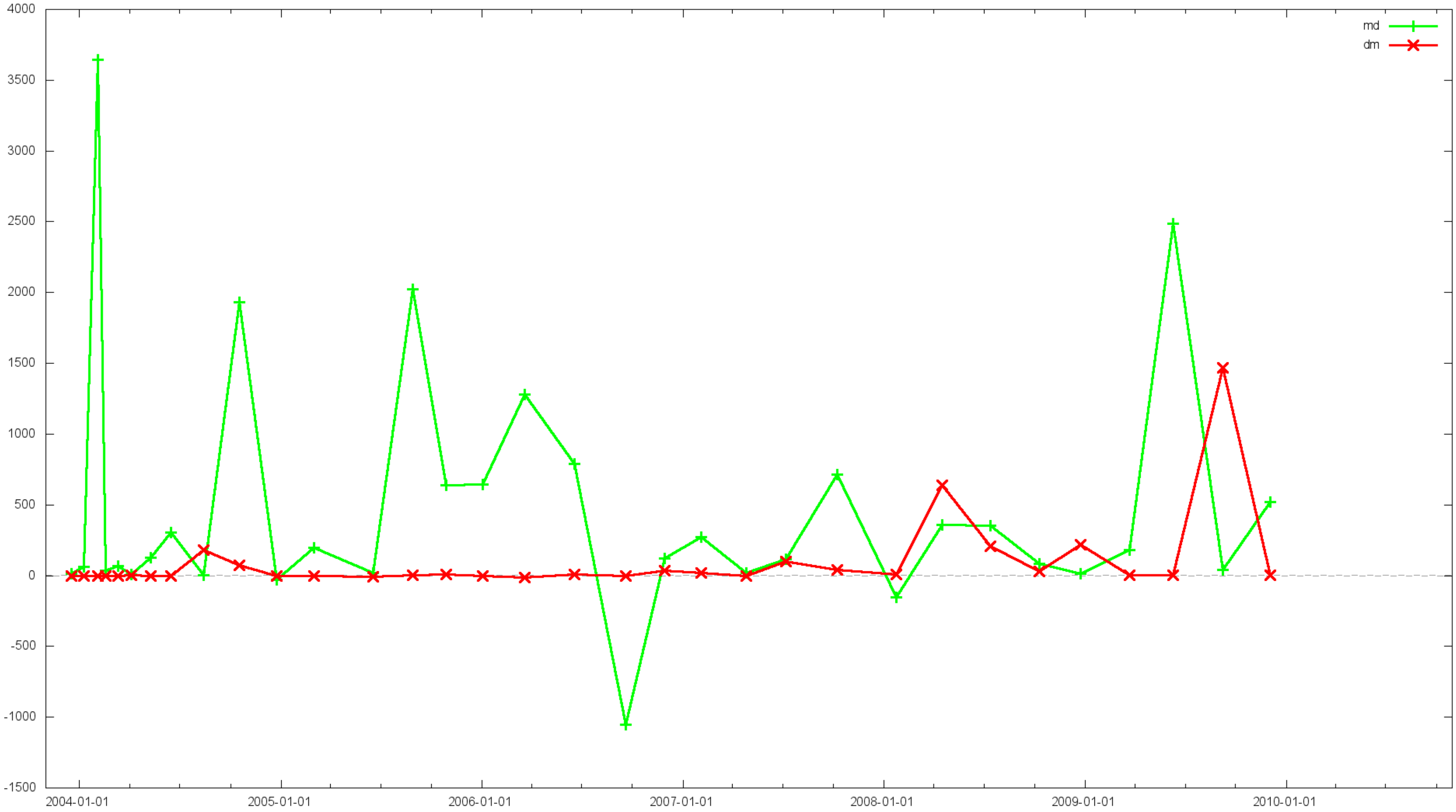
Reasons stated for dm/MD divergence

- MD not considered modular enough
- dmraid author wanted to provide cluster-aware RAID
- Handle meta-data formats outside the kernel
- MD assumed “well integrated and well tested”, but a somewhat dated framework, orthogonal to the “more modern” device-mapper
- MD had come out of a difficult organizational phase in 2.4
- MD and dm communities orthogonal, originally pursued different goals

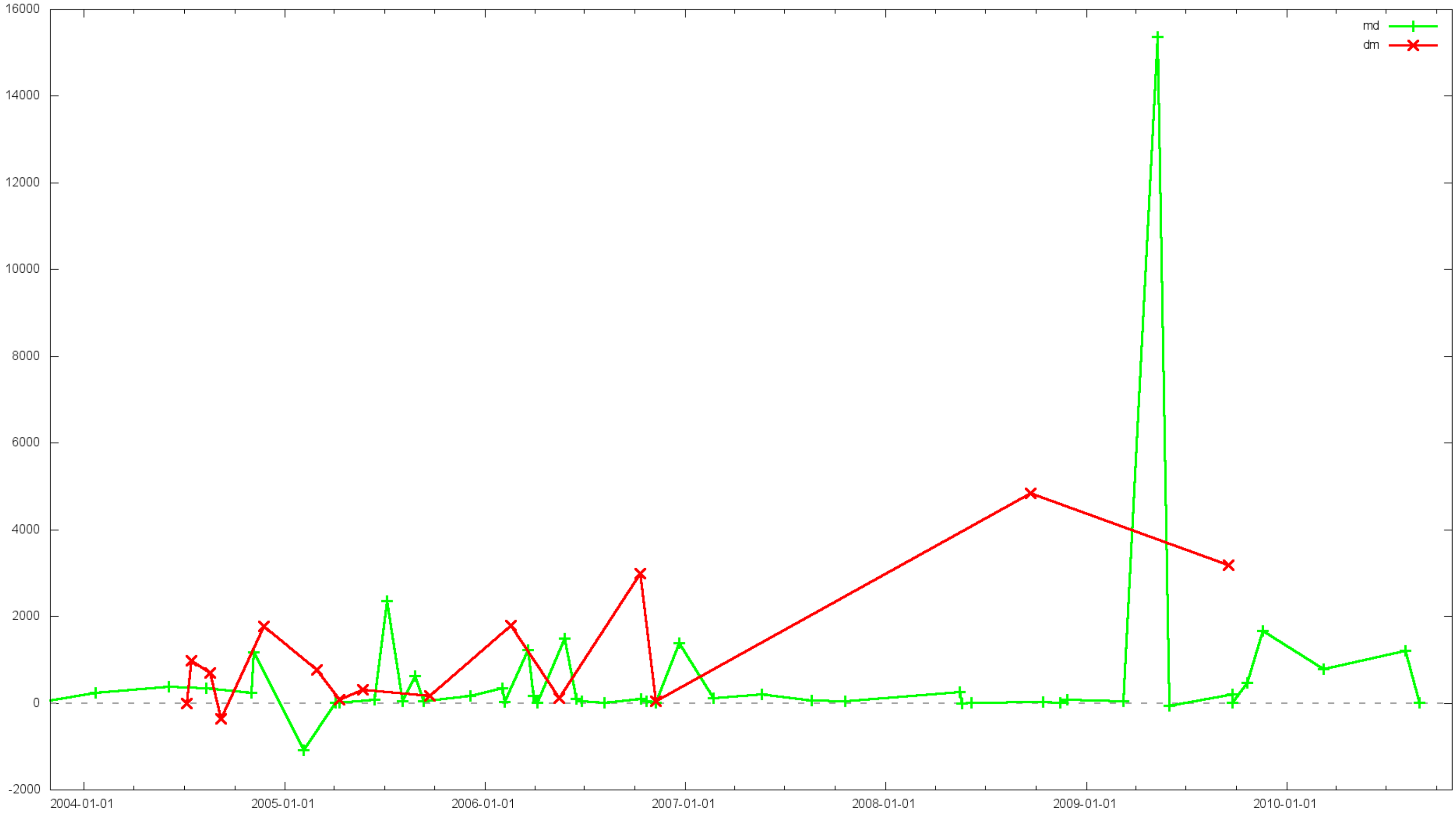
What constitutes dm versus MD RAID?

- Device-Mapper RAID handles most policy outside the kernel
  - Supporting libraries (device-mapper itself, LVM2) ignored, focus on RAID specific components
- MD is entirely RAID specific, both in user- and kernel-space
- Both benefit from shared kernel block infrastructure etc, which was not included
- 3rd party management or maintenance tools not included

DM versus md – kernel 2.6.0 – 2.6.32



DM versus md - user-space



COCOMO-II estimate for MD/dm

- Based on previous studies
- Parameters adjusted slightly, since more specific area examined
- Kernel-space:
  - embedded intermediate model with extra high complexity, high complexity, and very high timing requirements, but also high capabilities of the development group
- User-space:
  - Semidetached intermediate model with high reliability, high complexity, nominal timing requirements
- Higher than previous salary assumed (experience)

Estimated development cost (at 2.6.32 time)

Component	Physical lines of code	Person -Years	Schedule estimate	Average number of developers	Estimated cost to develop
MD kernel	16955	10.00	0.96	10.37	$2,735,633
mdadm user-space	27154	6.75	0.97	6.96	$1,845,966
MD total	**44109**				**$4.581.599**
Device-Mapper RAID kernel code	6366	3.09	0.66	4.66	$844.385
dmraid user-space	19508	4.66	0.85	5.47	$1.274.583
DM total	**25874**				**$2.118.968**

What happened?

- While dmraid did handle user-space meta-data first, and DDF in particular, mdadm gained this slightly later.
- Cluster-aware mirroring in dmraid became functional only in 2010.
- Initial state of dmraid was not very reliable and was unable to handle “partial recovery”
- Lack of interest in convergence from overarching maintainers and peers (until it was too late)
- No cross-pollination.

Feature state (approx. Sep 2010)

- Cluster-aware RAID1 in dmraid
- Both support user-space meta-data, but mdadm only supports DDF
- MD supports RAID6 and RAID-level migration, adding new RAID members, background scrubbing, auto-correction for (some) IO errors, off-loading of RAID6 computation to hardware
- MD user-base is larger, and the solution considered more mature; but for certain software-only RAID implementations, dmraid is still the only choice.

What is happening since?

- In hindsight, most respondents stated that the goals should have been pursued in the MD framework
- dmraid community never really took off and did not reach critical mass
- dm and MD are converging:
  - Device-Mapper wrappers for MD personalities
  - dmraid essentially dormant

Other scenarios

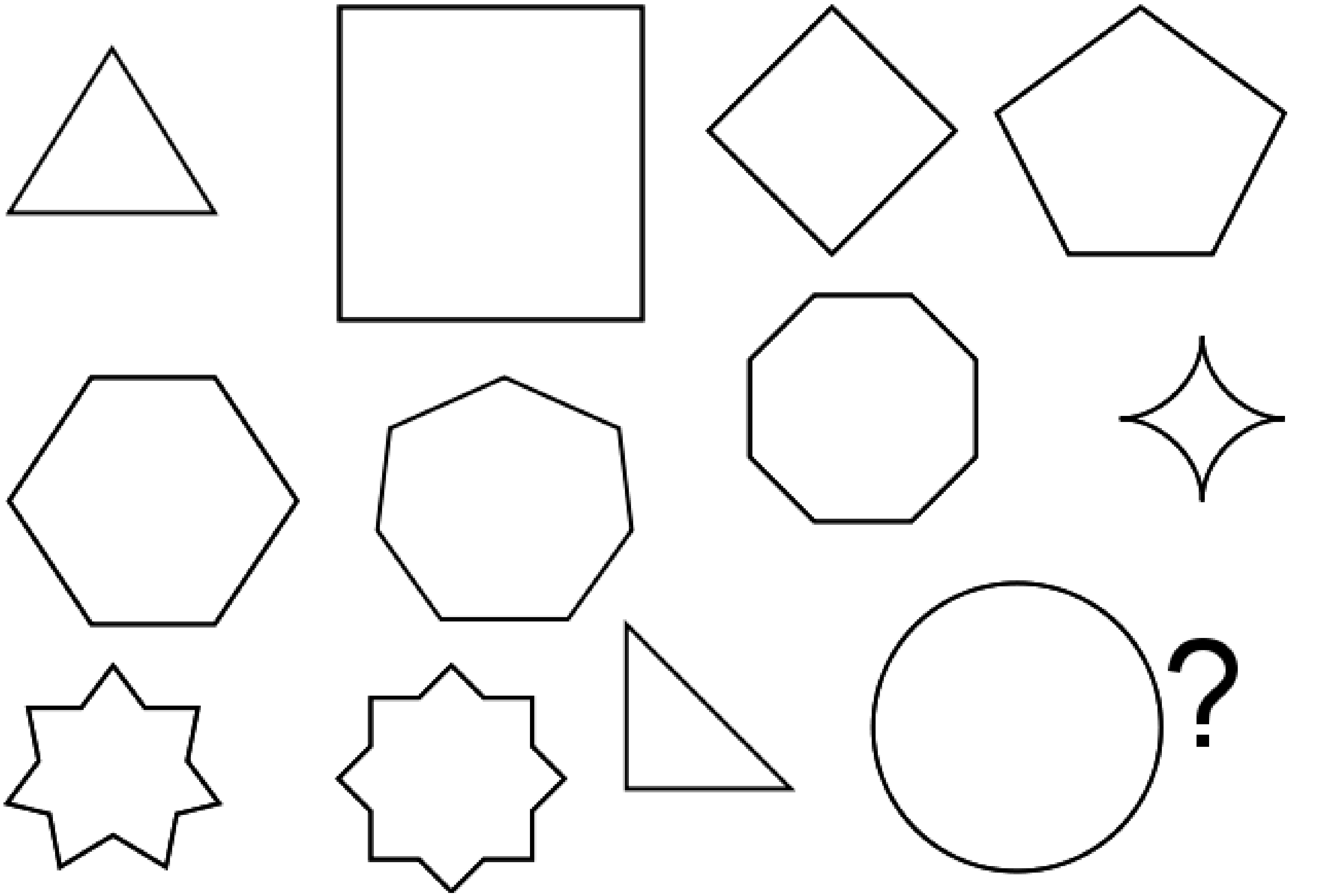
- We have how many desktop environments?
- How many file systems?
- Mercurial versus git versus bazaar?
- Office-software projects?
- Programming languages?
- Puts the ~$10 billion estimates into perspective

How could divergence be avoided?

- Modularity with the goal of extensibility
- Embracing new developers and users
- Open leadership and community style which empowers contributors
- Positive recognition of going the extra mile and getting the patch merged
- Community should discourage needless divergence, not just forks
- A lot of the cost is not borne by the developers, but users; this needs constructive feedback loops

Is it like this ...

… or more like this?



Questions & Answers

Thanks for your time!