Minding the Gap: R&D in the Linux Kernel

Muli Ben-Yehuda
IBM Haifa Research Lab
muli@il.ibm.com

Eric Van Hensbergen
IBM Austin Research Lab
ericvanhensbergen@us.ibm.com

Marc Fiuczynski
Princeton University
mef@cs.princeton.edu

"Hello everybody out there using minix—I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)"
—Linus Torvalds announces the Linux kernel on comp.os.minix, August 25th, 1991.

"Be thankful you are not my student. You would not get a high grade for such a design :-)"
—Prof. Andrew S. Tanenbaum responds.

1. INTRODUCTION

The Linux kernel, since its inception in 1991, has captured the interest of many thousands of developers and millions of users. It recently celebrated its 16th anniversary, includes many millions of lines of code, and is used in production systems around the world. It is also advancing at an increasingly rapid pace, undergoing many changes every single day. Indeed the kernel's importance to many large corporations has sparked a high level of contribution by those companies [3] [4], including the employment of many core kernel developers. Recently Linus Torvalds published statistics relating to contributions to the kernel over the past three years: 96,885 patches attributed to 4068 distinct authors have been accepted [5].

Two of your co-editors claimed in an SIGOPS OSR editorial [1] that mature open source systems are a perfect vehicle to explore new ideas. The gist of our argument was that open source projects tend to evolve toward a culture promoting inclusion. However, we noted that many times open source developers are so focused on their code that they neglect publication of their designs, implementations, and evaluations outside of their project's ecosystem—i.e., the code is only published on the Linux kernel mailing list (LKML).

On the flip side, Linux seems to have become the de-facto standard for (academic) systems software research. For example, four-fifth of the papers published in the 21st Symposium on Operating Systems Principles conference involved experimentation with or evaluation on the Linux kernel! Unfortunately only 1% of the contributions to the Linux kernel over the past three years can be attributed to the (academic) research community [2].

Clearly there is a gap between the kernel community and the research community. A primary motivation behind this special topics OSR issue was to help bridge this gap by encouraging kernel developers to publish recent additions to the Linux kernel as well as to provide a forum for experience papers which describe the introduction and integration of research into the mainstream Linux kernel. We have included representative papers of each flavor, as well as papers which discuss promising new areas of Linux research and development. We think it is important for the research community and the kernel community to cross pollinate more. We hope this issue will be the first of many venues where the will be able to do so.

2. THE PAPERS

For this OSR special issue, we welcomed technical papers covering the latest advances that have been or will soon be merged into the Linux kernel, as well as new idea papers discussing promising experimental work. We encouraged papers from both the Linux kernel community and the research community. Quoting from the original call for papers, the OSR issue aims to:

- Expose members of the Linux kernel community to exploratory research work that is going on which might influence Linux's evolution, and
- Expose members of the systems research community to the latest happenings in a mature, production kernel that is widely used and advancing rapidly.

The content of this special issue was determined by a peer reviewed selection process followed by shepherding, revision, and further review. Authors initially submitted papers for consideration in response to a general call. Each paper was peer reviewed by at least three reviewers and most papers were reviewed by four or more reviewers. An editorial committee meeting was held during which committee members vociferously argued the relative merits of each paper. The meeting's outcome was that half the papers were rejected, the remaining papers were accepted pending shepherding or invited for revision and re-submission under the the guidance of one or more members of the editorial board.

To assist the reader, the editors have grouped the accepted papers according to the area of the kernel they address. These groups, together with a brief summary of each paper's content, follow.

2.1 Core Technologies

There is often a symbiotic relationship as a technology is introduced into an open source project such as Linux. *Paul McKenney* provides an excellent discussion of this relationship while describing the introduction and evolution of Read-Copy Update (RCU) within the Linux kernel, as well as the introduction of Linux into RCU.

Bahmann and *Froitzheim* discuss the intricacies of providing efficient and reliable notification mechanisms between the kernel and user space. They describe an extension of the kernel Futex (fast mutex) synchronization primitives to provide a unified event notification mechanism. The unified notification mechanism allows moving many thread activation policy decisions into user-space, with benefits for multi-threaded reactive applications.

Ashwin Ganti gives a detailed description of his port of the Plan 9 authentication and capability mechanisms to the Linux kernel. He discusses userspace extensions of the pluggable authentication manager (PAM) to integrate these mechanisms with existing applications and discusses the resulting improvement to system security.

2.2 Resource Management

Over the past couple of years there has been much coding activity surrounding the Linux scheduler including the introduction of the Completely Fair Scheduler (CFS). *Wong*, *Tan*, *Kumari*, and *Wey* expose a weakness in CFS's current allocation scheme whereby an application can receive extra cycles by spawning more threads. They describe an algorithm which attempts to rectify the problem by taking into account which process spawned the threads.

Processor resources aren't the only system resource in demand, and *Craciunas*, *Kirsch*, and *Roeck* focus on a mechanism for controlling system throughput and responsiveness by accounting for and throttling system calls associated with I/O activity. Their approach attempts to solve the same problem as resource containers while minimizing the intrusiveness of code changes.

2.3 I/O

Linux has no shortage in the number of I/O mechanisms and interfaces. *Bruijn* and *Bos* present a consolidation based on a virtual file systems (VFS) extensions to the Unix pipeline model. They optimize performance by limiting the number of copies, context switches, and cache misses. Their implementation is structured in such a way as to adapt to many different types of underlying hardware.

Ha, *Rhee*, and *Xu* describe the CUBIC extension to the TCP stack, which builds upon the approach of BIC-TCP with a focus on improving bandwidth fairness. The paper describes the protocol extension in detail, as well as the particular details of the Linux implementation and evaluates fairness, efficiency, and stability.

The Linux file system layer includes numerous sub-systems aimed at optimizing performance for a variety of workloads that sometimes perform in sub-optimal and unpredictable ways given the different characteristics of workloads, underlying file systems, storage technologies, and storage networks. *Wu*, *Xi*, and *Xu* present a design for a new readahead framework which seeks to limit complexity and provide a platform for exploring more effective algorithms. They use their platform to implement and evaluate readahead optimizations for NFS, and describe future work which could explore optimal techniques for varying access patterns and back-end storage technologies.

Boutcher and *Chandra* present a mechanism for communicating file system liveness information across the Linux file systems and storage stack in order to improve utilization, security, migration and caching. They describe a "purge" operation which can be used by file systems to pass data to lower levels with minimal changes to the system API, thereby letting the lower FS levels discard data that is no longer needed.

2.4 Virtualization and Containers

The introduction of inexpensive multi-core processors has sparked a resurgence in the research and development activities surrounding virtualization. The mainline Linux kernel currently supports 8 different infrastructures for virtualization. In order to contain the multitude of I/O virtualization methods, *Rusty Russell* describes his virtio framework for unifying paravirtualized I/O, which has recently been merged into the mainline kernel and has been adopted by KVM and is used by his lguest hypervisor implementation.

Hallyn, *Bhattiprolu*, *Biederman* and *Lezcano* provide an overview of the various changes being made across the Linux kernel in order to provide better isolation through the introduction of private namespaces for various system services. Their changes also provide a higher level of abstraction for many system services which facilitates checkpoint, restart, and migration of user applications.

3. ACKNOWLEDGEMENTS

We would like to extend our thanks to the many authors who submitted papers, including the ones we were unable to include in this issue, for the time and effort they put into their submissions.

We would also like to thank the members of our Editorial Committee who did a very thorough job of evaluating the submissions and who also shared shepherding responsibilities for the accepted papers. Listed in alphabetical order, the Editorial Committee is:

- Patrick Bridges (University of New Mexico)
- Angela Demke Brown (University of Toronto)
- Hubertus Franke (IBM Research)
- Oren Laadan (Columbia University)
- Paul McKenney (IBM Linux Technology Center)
- Chris Mason (Oracle)
- Ron Minnich (Sandia National Laboratory)
- Stephen C. Tweedie (Red Hat)
- Chris Wright (Red Hat)

- Pete Wyckoff (Ohio Supercomputer Center)

The editorial committee was ably assisted by the following external reviewers:

- Sapan Bhatia (Princeton University)
- Eli Brosh (Columbia University)
- Fernando Laudares Camargos (Universite de Sherbrooke)
- Steven Muir (Vanu)
- Ram Pai (IBM Linux Technology Center)
- David Pressoto (Google)
- Edi Shmueli (IBM Haifa Research Lab)
- Ben-Ami Yassour (IBM Haifa Research Lab)

We would also like to thank the Operating Systems Review editor, Jeanna Matthews, and ACM SIGOPS for supporting a special topics issue on the Linux Kernel.

REFERENCES

[1] M. Ben-Yehuda and E. Van Hensbergen. Open source as a foundation for systems research. *SIGOPS Oper. Syst. Rev.*, 42(1):2–4, January 2008.

[2] J. Corbet. 2.6.24 - some statistics. Available: http://lwn.net/Articles/264440/ [Viewed May 30, 2008].

[3] G. Kroah-Hartman. Linux kernel development: how fast is it going, who is doing it, what they are doing, and who is sponsoring it. In *OLS '06: The 2006 Ottawa Linux Symposium*, pages 239–244, July 2006.

[4] G. Kroah-Hartman, J. Corbet, and A. McPherson. Linux kernel development: how fast is it going, who is doing it, what they are doing, and who is sponsoring it. Available: https://www.linuxfoundation.org/publications/linuxkerneldevelopment.php [Viewed May 30, 2008].

[5] L. Torvalds. Linux 2.6.26-rc3. Available: http://lkml.org/lkml/2008/5/18/319 [Viewed May 30, 2008].