

Application Specific Linux (ASL)

Lamia Youseff, Rich Wolski, Chandra Krintz
Computer Science Department
University of California, Santa Barbara

Recent advances in high-performance processors and network technologies are making clusters of workstation computers cost-effective platforms that can support the next generation of scientific applications. Low per-unit cost, advances in computing and communication power, and the availability of Linux as a free, easy-to-use, and nearly standard operating system, make high-end computing with these systems accessible both to a very large user base. As part of this evolution, Linux has emerged as a nearly ubiquitous, open-source operating system with a wide-range of readily available programming support tools and specialized libraries. It is currently the system-of-choice in academic and production scientific computing settings.

A key limitation to the use of Linux for high-end cluster computing however is its potential performance impact on application execution. Linux, like other general-purpose operating systems (GPOS) continues to evolve to support an enormous range of user requirements, application domains, and devices. In contrast, scientific applications executing in clustered settings are frequently large, resource intensive, long-running, and use space-sharing to gain exclusive access to the machines they use through a batch system. They do not compete dynamically for processor and I/O resources like many other application domains. Therefore, Linux includes many features and built-in policies that do not promote the performance of high-end scientific applications. In particular, scientific applications typically do not require the extensive support for fair resource sharing (since they execute in production space-shared, and not time-shared, environments) or quick response time (since they may not be interactive).

In our Poster, we are presenting *Application Specific Linux (ASL)*, a customized Linux image that enhances the performance of scientific applications. The goal of our research is to investigate techniques that maintain the ease-of-use and cost benefits of Linux while enhancing the performance achievable by high-end scientific applications executing in large-scale cluster computing settings. To enable this, we are studying ways to automatically customize the Linux instance for an application based on its own needs. We are also exploiting the exclusive processor access that batch scheduling implemented to eliminate unneeded mechanisms and policies in the kernel. We use both runtime and compile-time approaches to customize the Linux instance used by each application. Each of which will allow the scientific programmer to use unmodified Linux as a local development and debugging environment; then to apply our techniques as an additional compilation step before initiating high-end cluster execution.

In order to enable Linux customization, our system consists of four logical customization phases. Static and dynamic application analysis is our system's first phase. The application as well as its potential coupling effect on the kernel is studied using Phases profiling techniques and kernel performance-annotated call graphs. The analysis phase would suggest a number of static customization enhancements for the kernel image, which would be produced at the development site. The second phase: static customization would include inlining the application code inside the kernel image to reduce the user-kernel space crossing overhead. This phase also involves inlining some kernel modules and creating execution shortcuts in the kernel for enhanced performance of common execution scenarios for the application. The customized Linux image (including the application) is shipped to the production environment.

At the production environment, ASL is deployed on a minimal virtual machine monitor (VMM) that is capable of protecting volatile hardware resources (e.g. BIOS code). This model allows us to introduce unsafe customizations, since the VMM is sandboxing the execution of the application. During its execution, ASL is a self evolving and dynamically adopting system that keeps modifying itself to meet the application's requirements at runtime for a better performance.

Much prior work in the area of application-specific operating systems (OSs) has thoroughly studied extensibility, specialization, and minimization of the OS in general. However, our research is novel in that it combines and extends these efforts into a system that automatically customizes the Linux operating system for a single application run and is specifically focused on the application domain of scientific computing using high-performance clusters. At the same time, our use of Linux ensures that the environment for scientific applications developers remains familiar and unified across the development and production platforms they use thereby promoting ease-of-use, programmer productivity, and efficient program management. Our Research end-goal is a software system that automatically enables high performance scientific computing on commodity systems through application-specific customization and dynamic adaptation of a low-cost, popular, and familiar Linux operating system.

APPLICATION SPECIFIC LINUX (ASL)

LAMIA YOUSSEFF

RICH WOLSKI

CHANDRA KRINTZ

Problem

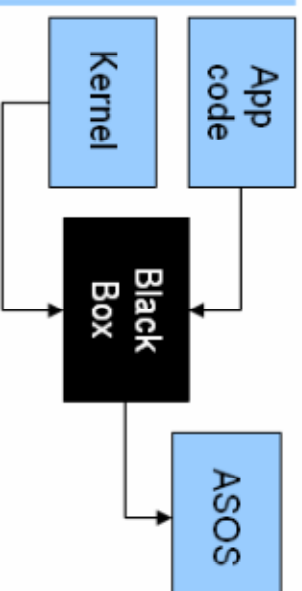
Specific Domains of Applications suffer from the general policies dictated by general-purpose OS (GPOS).

Ex: DB applications, scientific applications.

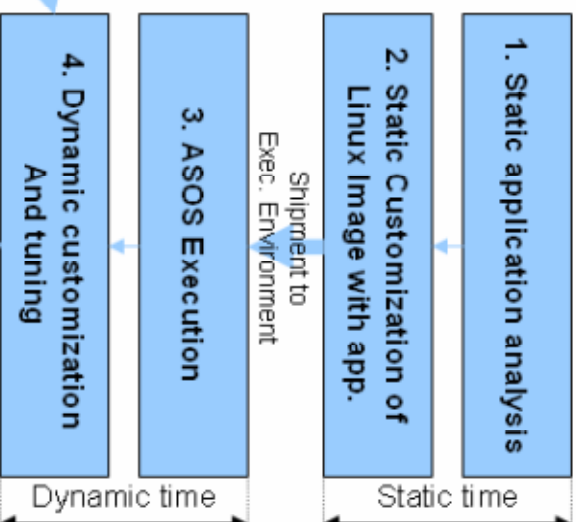
Project Objective

Investigating specialization techniques that automatically customize GPOS used by an application according to its application-specific requirements and time-varying behavior.

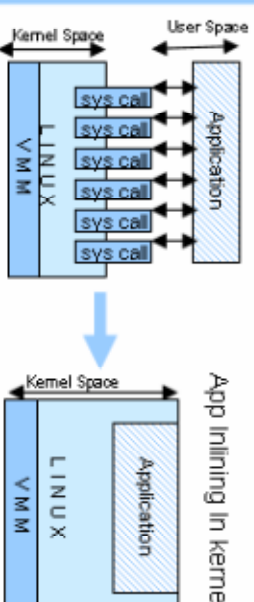
- Which GPOS? Linux Image
- Which Apps? Scientific Apps
- Metric of Success? App performance



Customization Phases



2. Static customization



3. ASOS Execution



4. Dynamic Customization

