FusedOS: General-Purpose and Specialized OS Personalities Side by Side

Marius Hillenbrand
Karlsruhe Institute of Technology

Yoonho Park Bryan Rosenburg Kyung D. Ryu
IBM Research

Frank Bellosa
Karlsruhe Institute of Technology

1 Introduction

We currently see a change in what are considered the most demanding computational problems. So far, notions such as *supercomputing* have been firmly associated with high-performance computing (HPC) and problems from science and engineering. However, the statistical analysis of huge amounts of real-world data (*Big Data*) is increasingly perceived as a new great challenge for computing while HPC users move to new programming languages and frameworks at the same time. As a result, there is a strong incentive to facilitate the large-scale distributed systems used in HPC for new workloads – both for manufacturers of such systems and for users looking for scalable systems for their *Big Data* tasks [2].

On the hardware side, multicore counts continue to increase but heterogeneous technology is increasingly used to address power-efficiency and density challenge.

In our talk, we will first describe the design principles of FusedOS and present the architecture and the prototype of our first incarnation of a *fused OS* on IBM Blue Gene/Q. Then, we will discuss some lessons learnt and the directions that will drive our project in the future.

2 Principles

In our FusedOS project we address both heterogeneity of cores and the need for a richer and familiar operating environment on large-scale systems. We base our research on two principles: We aim to (1) combine two different OS personalities, while (2) presuming heterogeneous cores.

2.1 Fusing OS Personalities

Our decision to fuse two OS personalities is inspired by the situation in high-performance computing (HPC): There, developers either aim to streamline and customize an existing OS, or build a custom and minimal OS from scratch. The downsides of both approaches are well-known. In extending an OS, the existing code base limits how fundamental principles can be changed. These limits can be overcome by writing an OS from scratch, though at the cost of an enormous engineering effort.

In FusedOS, we propose an alternative option and instead put OS personalities side by side. Whereas traditionally virtualization has been used to run several OS's on one system, we embrace space partitioning as the prior means to that purpose. Some cores run the general-purpose FWK, others are serviced by a specialized and streamlined OS. Thereby, we support systems without virtualization support and can avoid nesting another level of virtualization when running inside a virtual machine.

Former proposals for specialized OS's have typically aimed at replacing a general-purpose OS as far as possible (recent examples are Libra and Unikernel [1, 5]). In contrast, we maintain both OS personalities and allow applications to choose the one best matching its demands. Instead of isolating the two domains, processes can interact between OS personalities (as far as the semantics of the respective system call interfaces permit).

2.2 Heterogeneity

In FusedOS, we presume heterogeneous cores. We differentiate between streamlined and low-complexity application cores, optimized to execute application code power-efficiently, and general-purpose system cores. The application cores run highly-parallelized parts of a program whereas the general-purpose cores provide high single-thread performance for serial parts.

Whereas our model fits both common variants of contemporary heterogeneous systems, we envision application cores to be inbetween these cases. In contrast to GPUs, we assume a more standard programming model. In contrast to Xeon Phi and former proposals for heterogeneity, we assume that application cores do not support the separation between system and user mode. Thus, they are controlled remotely by the system cores. The system cores will have features found in high-performance, general-purpose processors such as strong integer performance and out-of-order execution.

3 FusedOS for IBM Blue Gene/Q

Our first instantiation of the FusedOS principles is a prototype running on IBM® Blue Gene®/Q. We combine Linux

with the Compute Node Kernel (CNK) [3], a specialized light-weight kernel used by Blue Gene/Q. Thereby, we aim to support typical HPC workloads as well as jobs that require more POSIX functionality than CNK provides, without investing the effort of re-implementing these features.

Although Blue Gene/Q has homogeneous cores, we simulate heterogeneity by assigning a set of cores to act as application cores.

3.1 Architecture

FusedOS for Blue Gene/Q consists of four components: (1) Linux running on the system cores, (2) the Compute Library (CL) which encapsulates CNK's functionality in a Linux user process, (3) a Linux kernel module that provides a control interface for the application cores to CL and a small system-level monitor on the application cores.

We modified Linux to partition cores and memory and to export the application cores' control interface to CL. In order to minimize the changes to Linux, we added only minimal hooks and placed the bulk of our code in two loadable file-system kernel modules. Using these pseudo filesystems, CL can allocate application cores and control the execution of CNK applications on application cores via ioctl and mmap system calls. CL itself runs on a system core as a regular application in Linux. It provides OS services for CNK applications, many of which it delegates to Linux, taking the role of a *proxy process*. The supervisor-state monitor on application cores controls address translation and process execution on behalf of CL. When a thread on an application core makes a system call or encounters an exception, the monitor saves the thread's context and then passes control to CL.

3.2 Prototype Status

Our prototype supports running Linux applications on system cores as well as unmodified CNK applications on application cores. For each CNK process, there is one CL process running in Linux. As CNK's system call interface resembles Linux's, processes can interact between the two domains. For example, a Linux process could spawn many processes running in parallel on the application cores and can coordinate them, at run-time, via IPC.

We have shown that single-node performance of HPC applications on FusedOS is competitive with CNK and much better than on Linux [6]. In ongoing research, we have added support for running MPI applications with inter-node communication via the Blue Gene/Q torus network. We will include early performance results on our poster.

4 Research Goals and Future Directions

The FusedOS project started with a clear focus on HPC. We began exploring how we could bring a general-purpose OS into what had strictly been the domain of customized systems before. We fused the existing special OS with a general-purpose OS to maintain performance characteristics. At the same time, we embraced the premise of future HPC systems being heterogeneous and, thus, cores having different roles.

System call latency in our prototype turned out to be very high, as processes on application cores get their system calls serviced on a remote core. That is very different from other application-domain-driven library OS's (such as [1, 5]), which turn system calls into function calls and do the costly interaction with a remote general-purpose OS only for services they do not implement themselves.

Still, the performance of HPC applications turned out to be surprisingly good. We observed that, after an initial setup, such applications hardly do any time-critical system calls. Instead, they rely on the message-passing interface (MPI) libraries with user-level network access that they are linked against [4]. Thus, these MPI stacks have actually evolved into library OS's. Instead of on exokernels or hypervisors, they are based on the API of a general-purpose OS.

In future work, we will look for similarly evolved de facto-library OS's in domains besides HPC, as well, and whether our FusedOS approach can be generalized for these scenarios. At the same time, we will follow the development of the hardware of heterogeneous systems and evolve our heterogeneity model accordingly.

References

[1] G. Ammons, J. Appavoo, M. Butrico, D. Da Silva, D. Grove, K. Kawachiya, O. Krieger, B. Rosenburg, E. Van Hensbergen, and R. W. Wisniewski. Libra: a library operating system for a jvm in a virtualized execution environment. In *VEE '07*, pages 44–54, New York, NY, USA, 2007. ACM.

[2] J. Appavoo, A. Waterland, D. D. Silva, V. Uhlig, B. S. Rosenburg, E. V. Hensbergen, J. Stoess, R. W. Wisniewski, and U. Steinberg. Providing a cloud network infrastructure on a supercomputer. In S. Hariri and K. Keahey, editors, *HPDC*, pages 385–394. ACM, 2010.

[3] M. Giampapa, T. Gooding, T. Inglett, and R. W. Wisniewski. Experiences with a lightweight supercomputer kernel: Lessons learned from Blue Gene's CNK. In *ACM/IEEE International Conference for High Performance Computing (SC10)*, New Orleans, LA, November 2010.

[4] S. Kumar, A. R. Mamidala, D. Faraj, B. E. Smith, M. Blocksome, B. Cernohous, D. Miller, J. Parker, J. Ratterman, P. Heidelberger, D. Chen, and B. D. Steinmacher-Burow. Pami: A parallel active message interface for the blue gene/q supercomputer. In *IPDPS*, pages 763–773. IEEE Computer Society, 2012.

[5] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft. Unikernels: Library operating systems for the cloud. In *ASPLOS '13*, 2013.

[6] Y. Park, E. V. Hensbergen, M. Hillenbrand, T. Inglett, B. S. Rosenburg, K. D. Ryu, and R. W. Wisniewski. Fusedos: Fusing lwk performance with fwk functionality in a heterogeneous environment. In *SBAC-PAD*, pages 211–218. IEEE, 2012.