Title: Using an LWK in a Multi-Kernel Environment to Improve Containers

Authors: Balazs Gerofi*, Yutaka Ishikawa*, Rolf Riesen+, Robert W. Wisniewski+
*RIKEN, +Intel

Over the last two decades there has been a trend toward requirements that are leading to an increasingly complex software stack on supercomputers. Supercomputers no longer sit in the corner of a data center, disconnected, running batch simulations. Instead, they have started to become an integral element of the data center and active participants in application workflows. While this trend has been going on for a while, in the last couple years it has significantly accelerated. Uncertainty quantification and multi-scale simulations that started this accelerated trend, led the way to more sophisticated analytics and the need for big data. More recently, to meet the needs of these applications and emerging ones, such as bio-informatics, machine learning, and others, ideas like extending the software stack to support containers, virtualizing the network, providing quality of service, and unifying services with cloud environments are being contemplated.

This new class of applications represents an important and upcoming role our supercomputers will need to play. Today though, few production cycles are going to the new model. Even in the future, applications will be a mix of classical simulation, analytics, and others. Thus, there is a requirement that the execution model for this new class of applications be able to simultaneously handle the requirements of the new work loads and provide the classical HPC execution model. McKernel and mOS are two multi-kernel operation systems that run both a lightweight kernel (LWK) and the Linux kernel on the same node. From a high level, the goal is to be able to service the system requests requiring high performance or scalability using the LWK, while leveraging the Linux kernel to provide compatibility to not only the Linux/POSIX APIs, but the full Linux environment. A positive and intended consequence of the clear separation of responsibilities is the ability to highly isolate the application running on the LWK from any system or application behavior on the cores executing Linux. Another feature of this architecture is a nimble LWK. Because we have the flexibility to specifically tailor the kernel to the needs of a class of applications, we can provide a highly specialized kernel. For example, if a class of applications do not need a preemptive process/thread scheduler because the number of processes/threads is smaller than the CPU cores, the kernel turns off interval timer, one of the OS noise sources.

Another way being explored to address the needs of, for example, running analytics and a simulation simultaneously, has been to use containers. Containers use Linux cgroups to allocate and dedicate resources to a given application with the goal of isolating it from other applications. However, the underlying Linux infrastructure is still shared. With the multi-kernel approach a more thoroughly isolated execution environment can be established. Both the McKernel and mOS architecture allow for multiple instances of an LWK to be run, providing isolation for a set of applications. The nimbleness characteristic of McKernel and mOS, as derived from their small LWK properties, opens the possibility of having those execution environments tailored to the needs of the application running on that LWK. Results shown in Figures 1-5 below, indicate that an LWK can provide tighter isolation, while retaining the benefit of a tailored environment. Figures 1 and 2 show the standard low noise versus Linux result but using cgroups. Figures 3-5 demonstrate that using cgroups and isolcpus can help provide the desired performance, but an LWK (in this case McKernel) can do a better job of providing the desired characteristics.

The work we are undertaking in collaboration between the McKernel and mOS teams has three intended contributions. First, we can run an application packaged in a container on McKernel or mOS. This enables application designers to package all the needed components in a container whether that container is meant to be deployed on a system using Linux cgroups or one of our LWKs. Second, the multi-kernel approach provides better isolation for applications running on the LWK. It provides a tighter guarantee on resources and reduces the interference between applications running on other kernels outside of the LWK. Third, by combining the first two, multi-kernels provide better performance, isolation, and QOS than cgroups. In this presentation, I will briefly describe the architecture of McKernel and mOS, describe how that leads to the above contributions and show some early results from McKernel demonstrating the isolation properties.

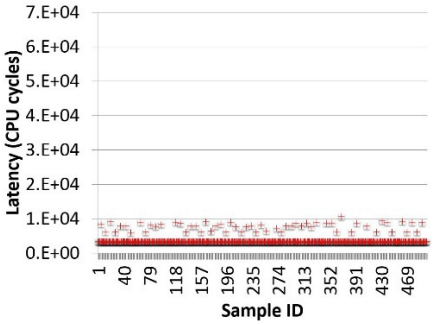

Fig 1: Linux with cgroups

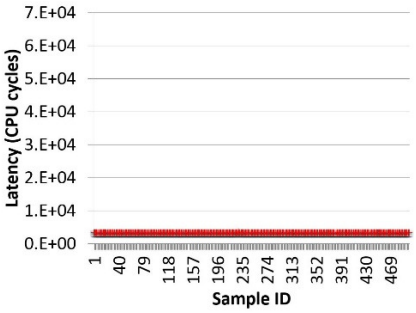

Fig 2: McKernel

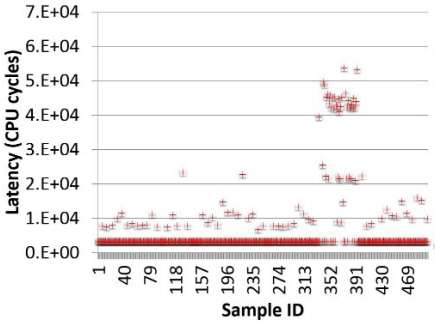

Fig 3: Linux with cgroups and Hadoop

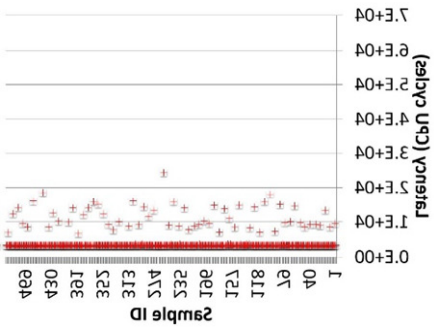

Fig 4: Linux, cgroups, Hadoop and isolcpus

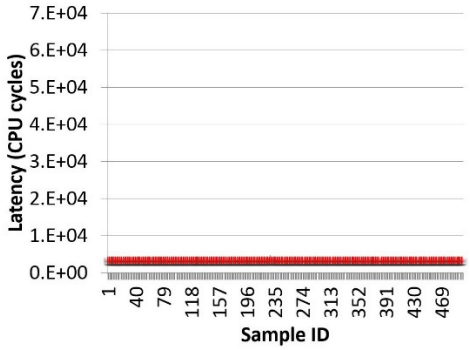

Fig 5: Hadoop with McKernel