Profile Guided Kernel Optimization for Individual Container Execution on Bare-Metal Container

Kuniyasu Suzaki, Hidetaka Koie, Ryousei Takano, National Institute of Advanced Industrial Science and Technology

Background

Container technologies becomes popular on super computers.

Some tools are proposed (e.g., Singularity, Shifter, CharlieCloud) and are used on real super computers (e.g., Shifter on "Piz Daint" No3 of Top500).

Unfortunately, container technologies do not allow changing the kernel.

Applications do not receive the benefit of an optimized kernel, especially Profile Guided Kernel Optimization (PGKO).

What is PGKO?

Profile Guided Optimization (PGO) is an optimization technology based on a profile of trial execution and reduces branch mispredictions, cache miss hits, and code size. PGO is supported on current compilers (e.g., Visual Studio, GCC, LLVM/Clang) and used on real applications (e.g., Google Chrome, Firefox).

PGO is not limited to a user space application and can be applied to the Linux kernel [3]. Profile guided kernel optimization (PGKO) creates a Linux kernel which is optimized for the behavior of an application.

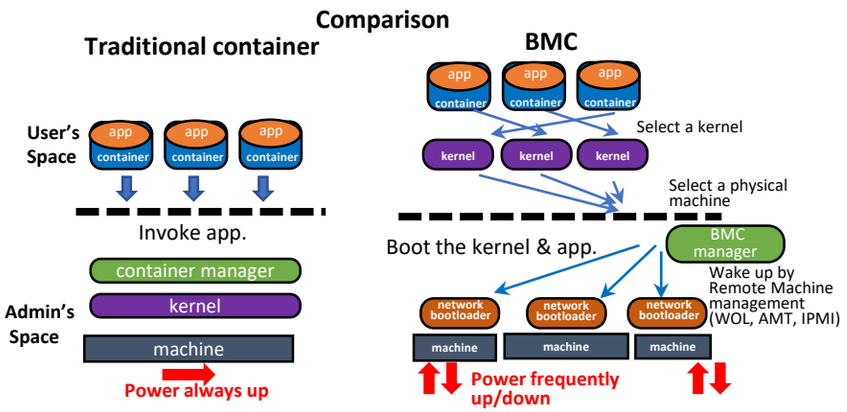
Approach

Bare-Metal Container (BMC) runs a container image on a remote machine with a suitable kernel. BMC is customized to create an optimized kernel by PGKO and apply it to the target application automatically.

What is BMC?

Bare-Metal Container (BMC) [2] is a technology to run a container image on a remote machine with a suitable Linux kernel, similar to an OS provisioning system (e.g., Ironic of OpenStack and Kadeploy3). BMC easily allows changes of the kernel and machine, which facilitates the comparison of the effects of different kernel optimizations on the machine. BMC utilizes remote machine management technologies (IMPI, Intel AMT, and WakeupOnLAN) to power on and off a remote machine. BMC also utilizes network bootloader "iPXE" to control Linux booting. iPXE has an original scripting language and allows the selection of a Linux kernel and initrd from HTTP/HTTPS servers. The container image is deployed on RamFS and is used as the root file system.

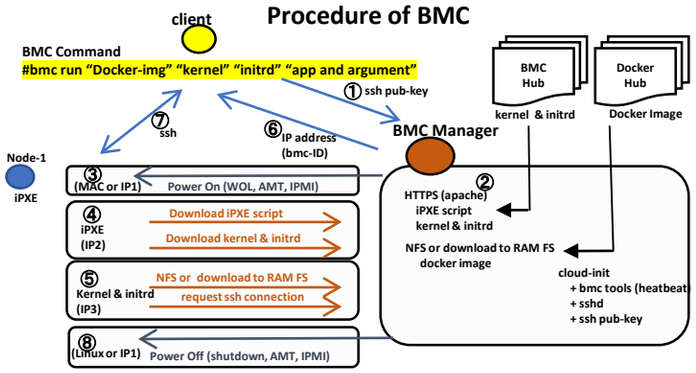
Comparison



Pros: Multi Tenant (high CPU utilization)
Cons: Kernel is not replaced.
Administrators must maintain the Linux distribution.

Pros: Apps select a kernel & hardware. Administrators don't need to install Linux on nodes.
Cons: Set-up overhead (Rebooting). But, kernel optimization surpasses the overhead on some applications.

Procedure of BMC



BMC's Customization for PGKO

BMC's procedure is managed by the shell script.

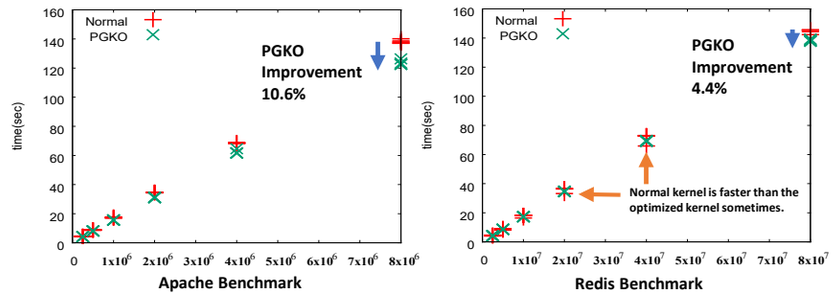
- First booting uses the profiling kernel of PGKO and takes the profile of a target application.
- The profile is used to re-compile the kernel, creating the optimized kernel.
- Second booting uses the optimized kernel and runs the application.

Evaluation

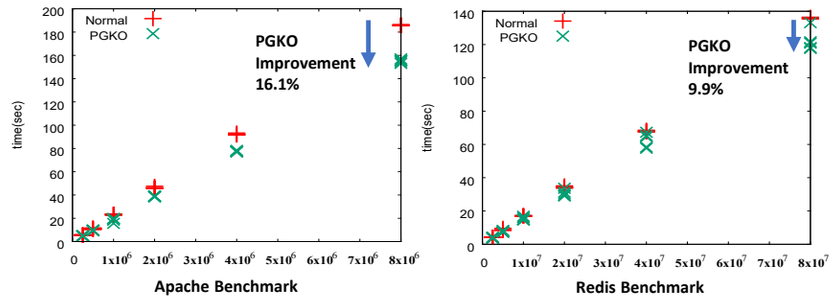
Kernels were optimized for big data workloads (Apache and Redis) on two machines; a server with Intel Broadwell Xeon E5-2630v4 10cores 2.20GHz and 64GB memory and a note PC (ThinkPad T430s) with Intel Ivy Bridge i7-3520M dual cores 2.90GHz and 16GB memory.

The profiles for KPGO were taken for benchmark size 1,000,000 on Apache and 50,000,000 on Redis. The optimized kernels were applied to different sizes. 5 trials for each benchmark size were measured. The optimized kernel was compiled on a separate machine and this overhead is omitted in order to assess the improvement only due to kernel replacement.

Intel Broadwell Xeon E5-2630v4 2.20GHz and 64GB memory



Intel Ivy Bridge i7-3520M 2.90GHz and 16GB memory



System call's improvement on Apache on Xeon. Measured by "strace -c". Benchmark size is 8,000,000.

System call	Improved time (sec)	Improved (%)	Calls	Total (sec)
fcntl	0.250522	2.63	16000114	9.259501
close	0.781239	9.16	8000087	7.749325
socket	0.387837	6.21	8000059	5.853391
epoll_ctl	0.753284	3.42	32000171	21.25565
connect	0.904899	6.47	16000115	13.07449
write	0.603767	4.88	16000054	11.7741
read	0.389067	3.52	16000027	10.66727
epoll_wait	-0.00056	-0.08	254755	0.73081
mmap	0.000012	2.84	75	0.00041
mprotect	0.000007	2	48	0.000343
open	-1.6E-05	-7.96	27	0.000217
fstat	0.000005	3.94	28	0.000122

System call's improvement on Redis on Xeon. Measured by "strace -c". Benchmark size is 80,000,000.

System call	Improved time (sec)	Improved (%)	Calls	Total (sec)
epoll_ctl	2.564906	3.94	100000218	62.52083
write	2.290927	8.9	20010657	23.46094
read	0.458358	3.37	20000042	13.15135
epoll_wait	-0.00527	-0.41	800007	1.290035
connect	0.000085	10.11	50	0.000756
fcntl	0.000016	3.11	100	0.000498
socket	0.000018	4.86	50	0.000352
setsockopt	0.000005	1.89	50	0.000259
close	0.000121	37.69	58	0.0002
mmap	0.000016	10.19	25	0.000141
mprotect	0.000005	5.68	12	0.000083
open	0.000017	23.61	8	0.000055

System calls issued many times are improved, but the effects are different.
The detail analysis is our future work.

Discussion

The overhead of compiling an optimized kernel requires some time making it unsuitable for single execution. However, it is useful when the target application is used repeatedly. We will analyze suitable usage in our future work. We also discovered that Intel Broadwell architecture improves branch prediction and may invalidate the software optimization. This may explain the discrepancy in the results when using Redis.

Acknowledgement: This poster is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References
1. BMC code : https://github.com/baremetalcontainer/bmc
2. K.Suzaki, H.Koie, and R.Takano, Bare-Metal Container: Direct Execution of a Container Image on a Remote Machine with an Optimized Kernel, High Performance Computing and Communications (HPCC) 2016.
3. P. Yuan, Y. Guo, and X. Chen, Experiences in profile-guided operating system kernel optimization, Asia-Pacific Workshop on System (APSys), 2014.