Mingtian SHAO, Kai LU, Wenzhe ZHANG, 2022. Self-deployed execution environment for high performance computing. *Frontiers of Information Technology & Electronic Engineering*, 23(6):845-857. https://doi.org/10.1631/FITEE.2100016

Self-deployed execution environment for high performance computing

Key words: Execution environment; High performance computing; Light-weight; Isolation; Overlay

Corresponding author: Kai LU
E-mail: lukainudt@163.com
ORCID: https://orcid.org/0000-0002-6378-7002

Motivation

- Workload. Users need to develop and debug their program and configure the environment on the login node, but the job is ultimately running on the compute nodes.
- Privacy. Multiple users may share the same login node, so user files and processes are visible to other users.
- System environment. Different users are likely to have different requirements for the system environment.

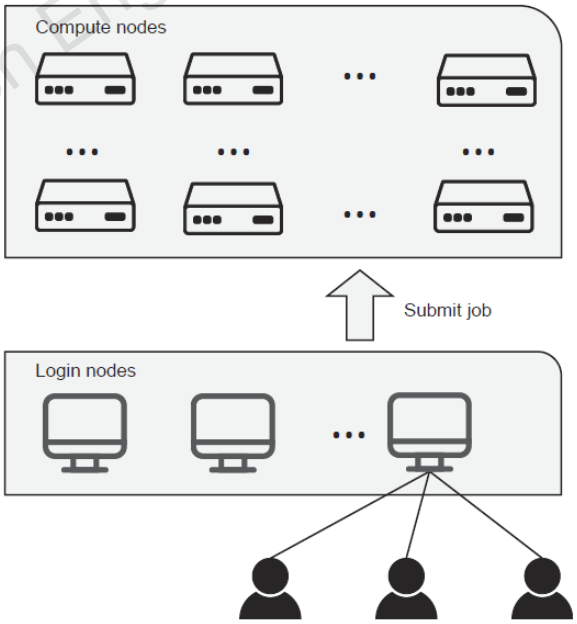

Fig. 2 Traditional job execution in HPC

Main idea

- Reduce the burden on users. Transparent to the user and completely automatic from the user's perspective. More efficient than manual user deployment.
- Protect user privacy. The development environments of different users are isolated from each other.
- Support user customization. The system administrator maintains and manages the underlying basic environment. The underlying base environment is shared among different users, while the user's customization has a higher priority.
- Be lighter than containers, with less performance and space overhead.

Method

Process isolation: Multiple users log in to the same login node and cannot see the specific process of other users. This allows the job to run in an isolated environment.

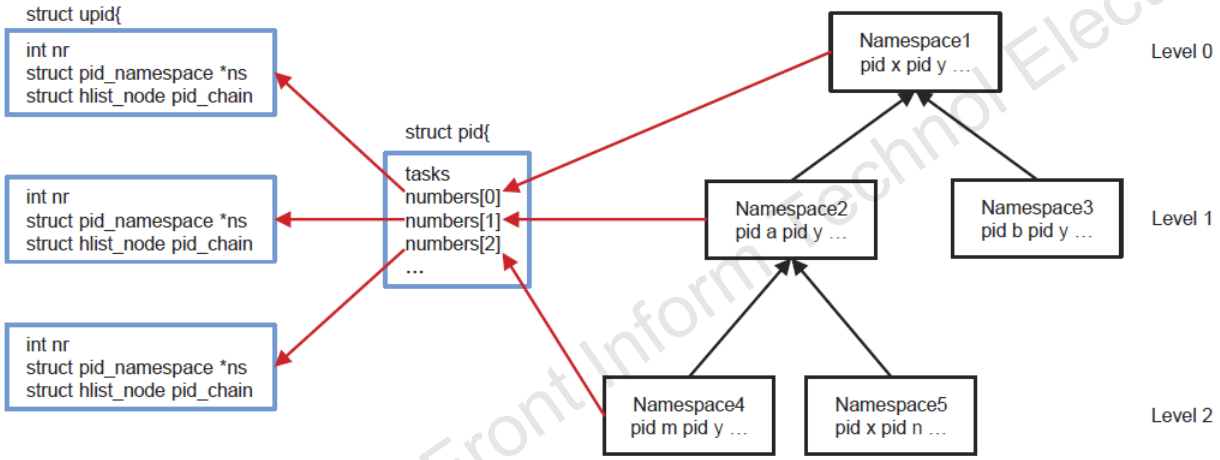

Fig. 3 Linux PID namespace

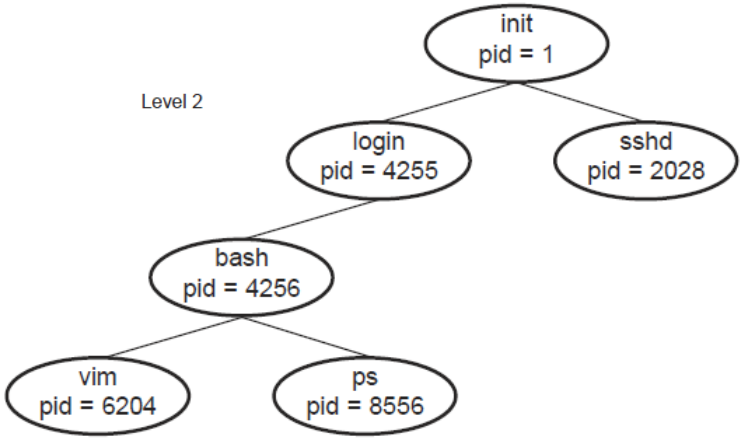

Fig. 4 The process tree of the new PID namespace

Method

File system: SDEE uses two file system layers, the upper and the lower layers. The lower layer is a standard system environment (the system's "/" directory) that is managed and maintained by the system administrator. An empty folder is then overlaid on the host "/" as the upper layer for the overlay file system.

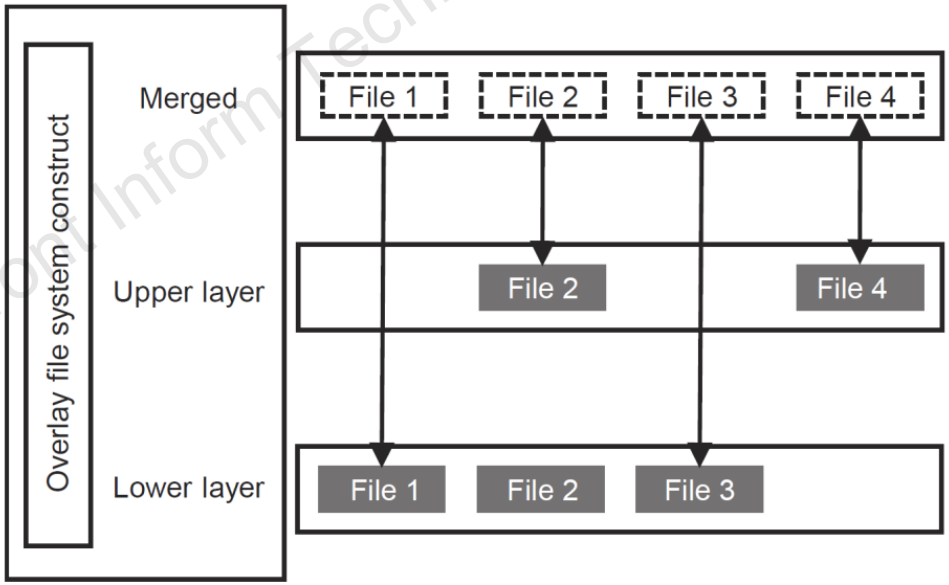

Fig. 5 The features of the overlay file system

Method

Deployment of the environment and job execution process

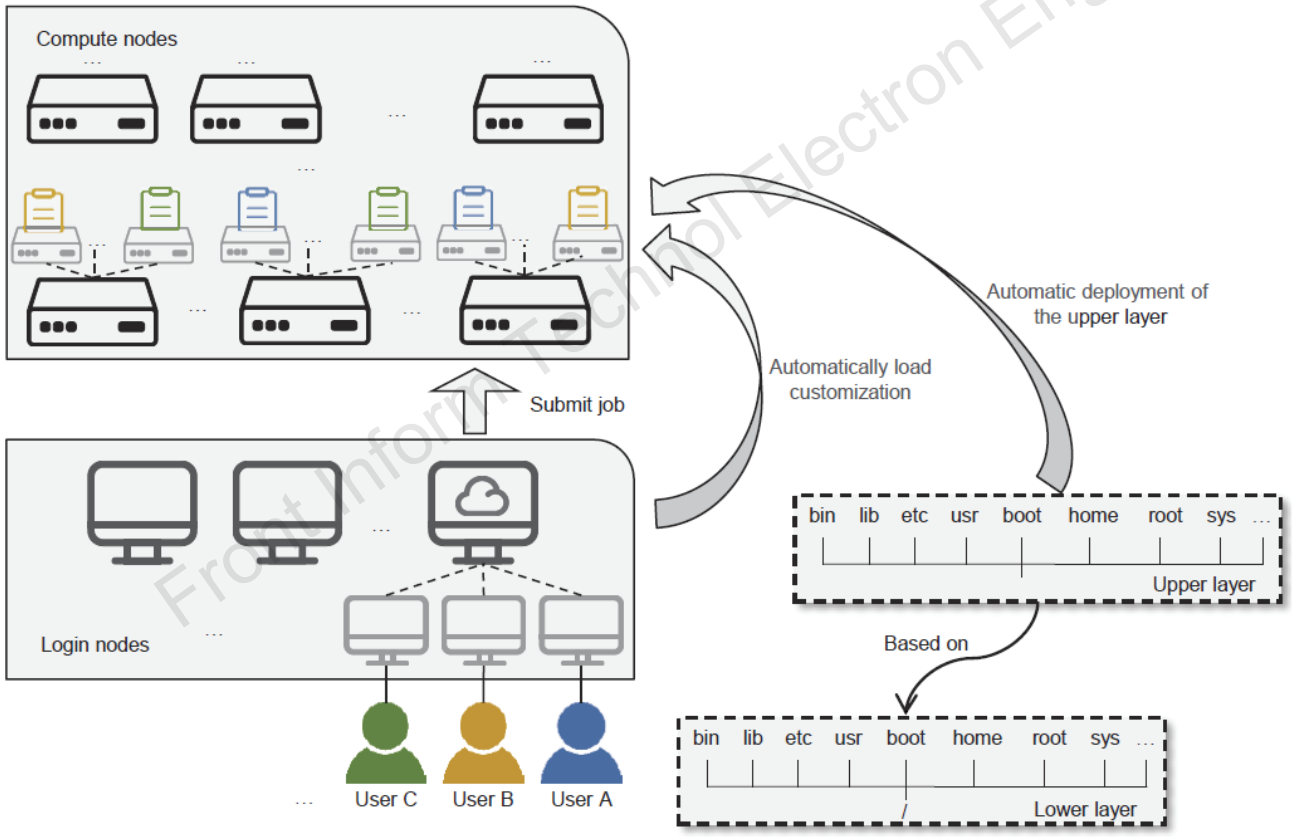

Fig. 6 The self-deployed execution environment (SDEE) framework

Major results

1. Overall performance (Unixbench)

Table 1 The comparison of Unixbench results (running one parallel copy of tests)

Benchmark	Score				Baseline	Unit	Overhead		
	Bare-metal	SDEE	Docker	Singularity			SDEE	Docker	Singularity
Dhrystone 2 using register variables	54 424 915.52	54 461 290.80	54 230 590.20	54 127 957.47	116 700	lps	−0.07%	0.36%	0.55%
Double-precision whetstone	3663.90	3657.17	3653.23	3647.17	55	MWIPS	0.18%	0.29%	0.46%
File copy 1024 bufsize 2000 maxblocks	1 625 214.80	1 621 733.70	1 601 870.70	1 618 400.37	3960	KBps	0.21%	1.44%	0.42%
File copy 256 bufsize 500 maxblocks	462 600.68	459 027.73	456 043.33	459 694.40	1655	KBps	0.77%	1.42%	0.63%
File copy 4096 bufsize 8000 maxblocks	3 605 343.52	3 591 148.63	3 593 464.83	3 557 815.30	5800	KBps	0.39%	0.33%	1.32%
Pipe throughput	3 064 019.30	3 046 674.73	3 033 976.87	3 053 341.40	12 440	lps	0.57%	0.98%	0.35%

Table 2 The comparison of Unixbench results (running four parallel copies of tests)

Benchmark	Score				Baseline	Unit	Overhead		
	Bare-metal	SDEE	Docker	Singularity			SDEE	Docker	Singularity
Dhrystone 2 using register variables	204 047 022.52	203 873 046.10	203 715 198.60	203 373 046.10	116 700	lps	0.09%	0.16%	0.33%
Double-precision whetstone	14 462.35	14 405.60	14 404.10	14 430.60	55	MWIPS	0.39%	0.40%	0.22%
File copy 1024 bufsize 2000 maxblocks	1 828 673.10	1 809 848.30	1 803 758.97	1 812 348.30	3960	KBps	1.03%	1.36%	0.89%
File copy 256 bufsize 500 maxblocks	491 701.37	487 144.18	484 199.23	484 394.18	1655	KBps	0.93%	1.53%	1.49%
File copy 4096 bufsize 8000 maxblocks	5 218 754.62	5 196 358.93	5 172 533.63	5 186 358.93	5800	KBps	0.43%	0.89%	0.62%
Pipe throughput	11 514 943.03	11 526 694.20	11 378 097.50	11 501 694.20	12 440	lps	−0.10%	1.19%	0.12%

Major results

2. CPU performance (NPB)

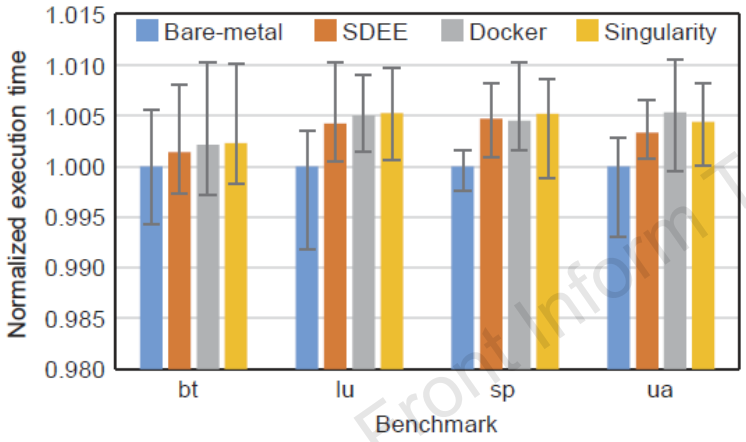

Fig. 7 Overhead of the NPB benchmark (serial)

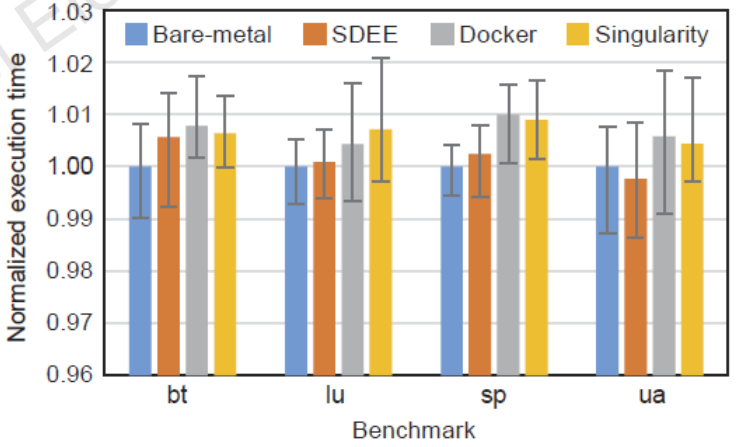

Fig. 8 Overhead of the NPB benchmark (MPI on 16 processes)

Major results

3. CPU performance (SysBench)

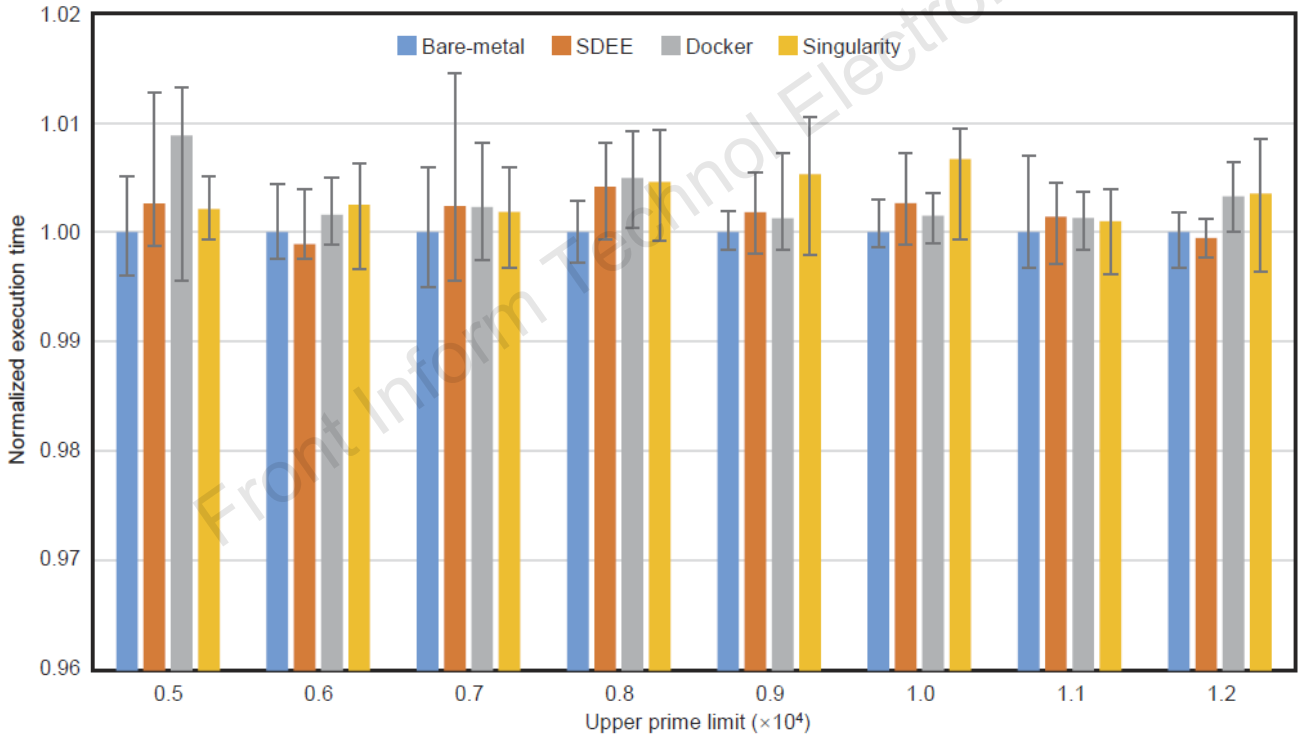

Fig. 9 Overhead of running the SysBench CPU test

Major results

4. File operation (Fio)

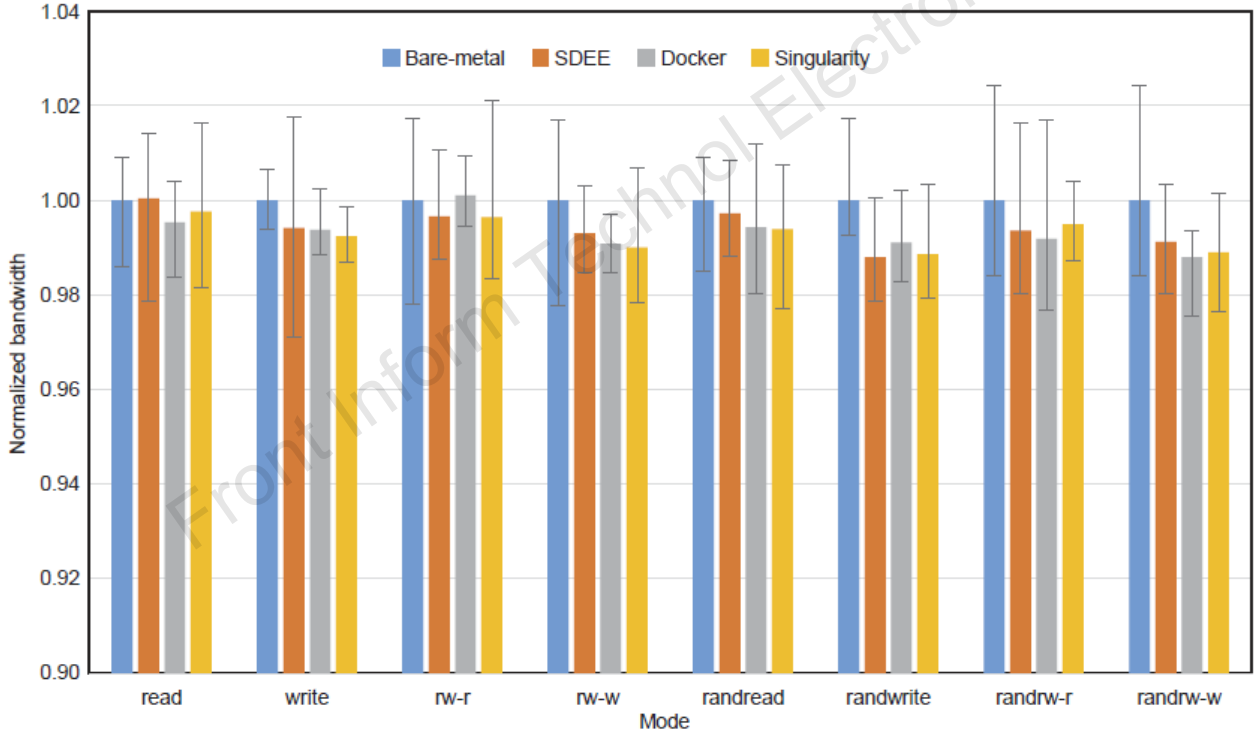

Fig. 10 Fio test overhead

Major results

5. Startup and deployment time

Table 3 Startup time of 100 SDEE, Docker, and Singularity environments

Environment	SDEE	Docker	Singularity
Total time (s)	2.848	16.930	12.489

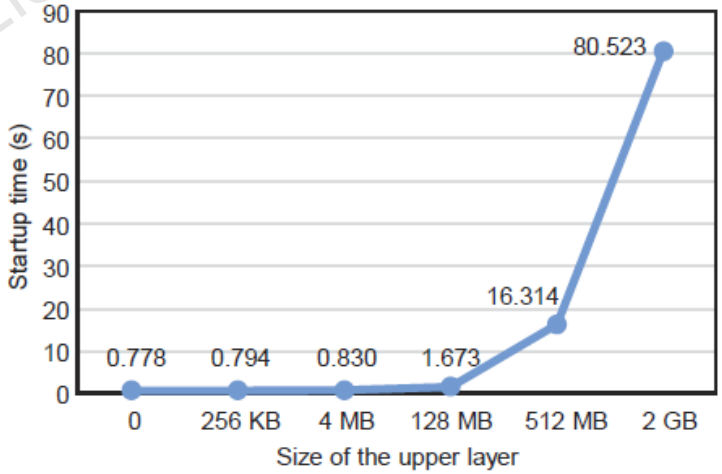

Fig. 11 SDEE deployment time

Conclusions

- SDEE implements self-deployment of the job's execution environment for high performance computing, and it reduces the user's burden in this process.
- SDEE supports flexible user environment customization, which not only protects user privacy but also ensures system security.
- Experiments show that the overhead introduced by SDEE is negligible. The SDEE overhead is lower than both the Docker overhead and Singularity overhead.