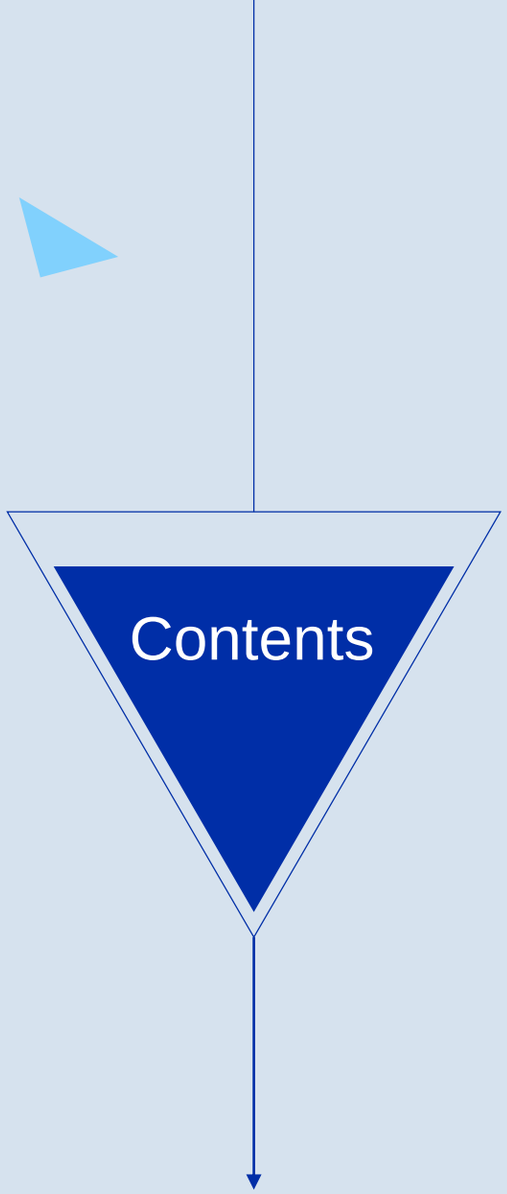
sysMaster: Redesigning process1 in Rust

Jingxiao Lu

Linux Architect, openEuler Community

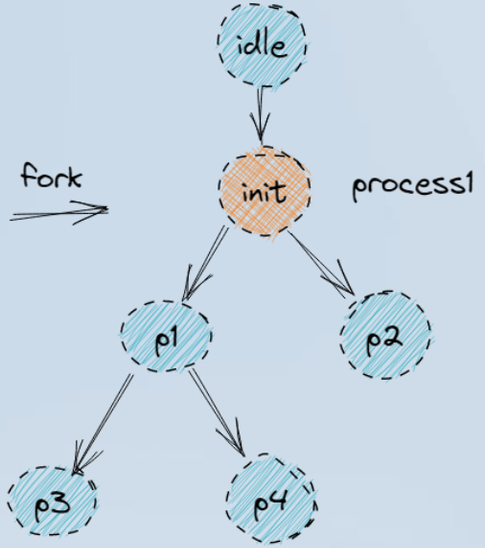
- innovative open source OS platform, cover all scenarios
- built on kernel innovations and a solid cloud infrastructure
- incubated and operated by the OpenAtom Foundation



What is Process1?

Process1 is:

- created through the fork syscall by idle.
- the first user-space process.
- the ancestor of all other user-space processes.



Primary roles:

- system startup
- zombie reaping

Very important:

- The stability of process1 determines the reliability of the OS.

Current State of Process1

- Different distributions have different process1. Up to now, there are 20+ different processe1 implementations.

[● ◂] systemd	upstart	SysVinit	openrc	launchd automation scripts
2010' • Red Hat, openSUSE • Parallel startup • On-demand loading • De facto standard	90' • Debian, ChromeOS • Asynchronous work • Service monitoring • Extensible event-driven model	80' • PCLinuxOS, Porteus • /etc/initab • Serial startup • Startup script	• Alpine, Gentoo • start-stop-daemon • supervise-daemon	• Mac OS X 10.4 • Daemons and Agents

Process1	Description	Startup Mgmt.	Process Recycling	Service Mgmt.	Parallel Startup	Device Mgmt.	Resource Control	Log Mgmt.
SysVinit	The init process tool used in earlier versions. It gradually fades out of the stage.	✓	✓					
upstart	Formerly used by Debian 7 and Ubuntu 14. Project is in maintaince mode only	✓	✓	✓	✓			
systemd	Faster startup speed. Compared with traditional SysVinit, systemd is a major innovation and has been used by most Linux distributions.	✓	✓	✓	✓	✓	✓	✓

The demands on Process 1 varies in different OSs

Scenario 1: Traditional Servers

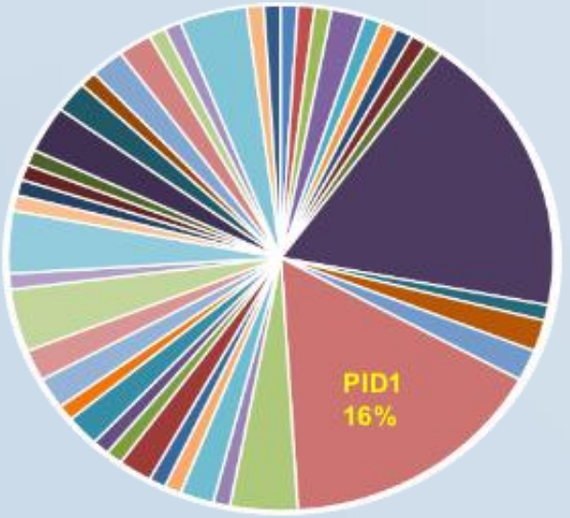
Scenarios: Servers in data center

Challenges:

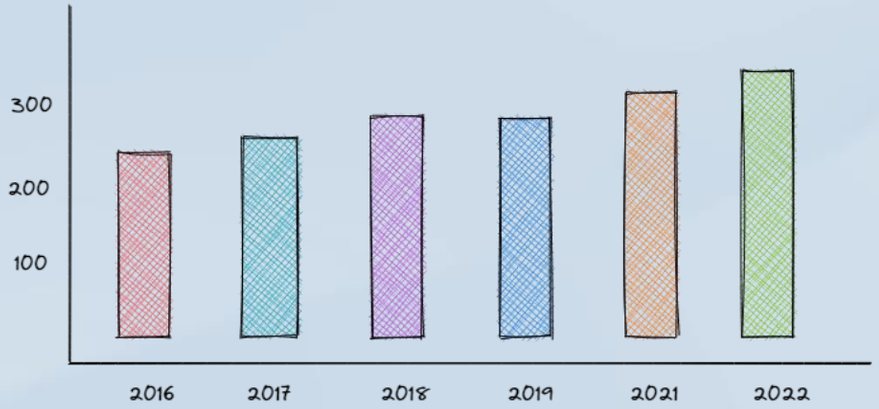
- Taking openEuler as an example，among the most difficult open source software issues in the past five years, PID1 issues account for 16%.
- According to the statistics of the openEuler community, the number of issues related to process1 does not decrease for a long time, and nearly 1/3 of the issues are memory issues.
- Without self-healing capability, its recovery relies on OS restart.

Objectives:

- Reliability of the minimum system and process1.



Classification of Difficult Problems



The PID1 problem persists

Scenario 2: Cloud-Native

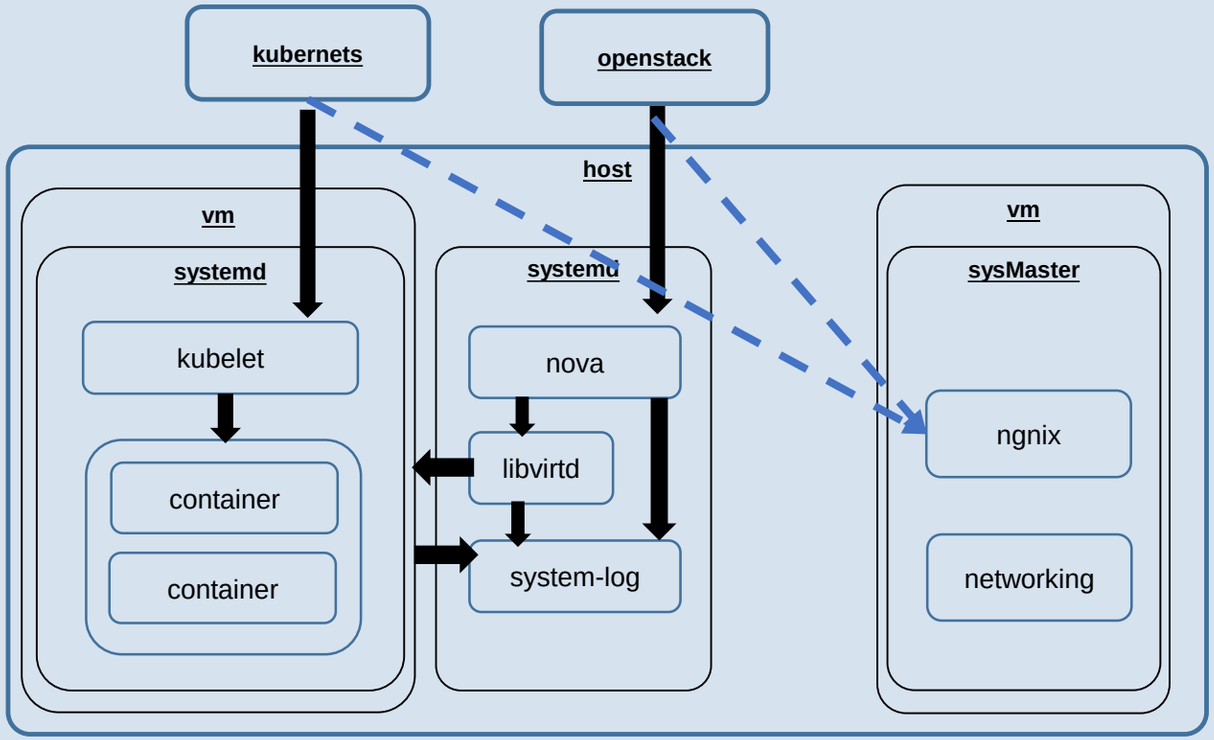
Scenarios: Cloud-Native

Challenges:

- Kubernetes and OpenStack cannot manage systemd units and system processes in hosts or VMs.
- HostOS maintenance is difficult and painful.

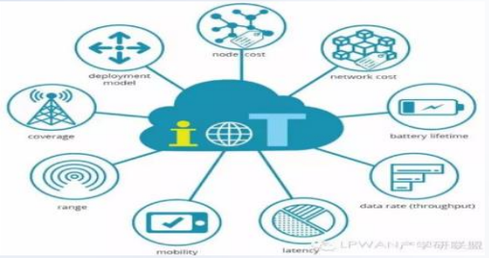
Objectives:

- **Easy to manage:** Provide a unit management agent and connect to distributed management frameworks (such as Kubernetes and OpenStack).
- **Easy on maintenance:** Make processes one of the resource types on the platform, improving experiences on host OS maintenance.



Scenario 3: Embedded Devices

Scenarios: Embedded device



Challenges:

- Mature init projects, such as systemd, do not support embedded devices.
- Components are not independent of each other, which increases the complexity and instability.
- The size is large and the resource usage is high, causing resource waste.
- Reliability issues occur frequently.

Objectives:

- high reliability, low latency.

sysMaster: What & Goals

sysMaster:

1. Ultra-lightweight, high-reliable service management & process1 for cloud/edge/device scenarios
2. Introduces fault monitoring, self-recovery to improve OS stability and service availability

Goals:

- Initialization and service management of the Linux OS
  - Fault recovery within seconds, with no impact on services.
  - Lightweight and flexible resource overhead.
  - Systematically eliminating memory security issues.
- Simplify host OS maintenance in cloud scenarios
  - Provide a unit management agent and connect to distributed management frameworks (such as Kubernetes and OpenStack).
  - Make processes one of the resource types on the platform , enabling convenient host OS maintenance.

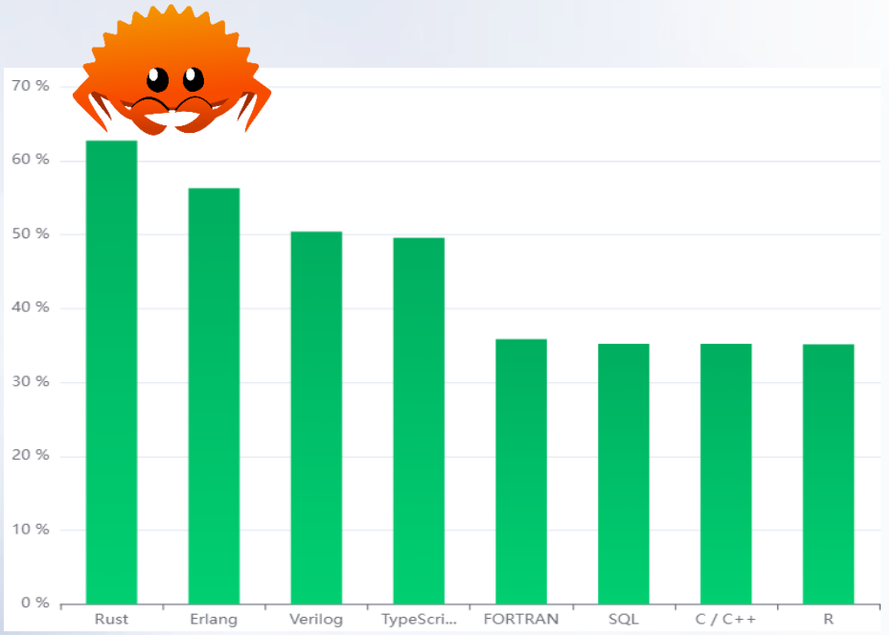
Rust Rocks

System-oriented Rust programming:

- **Security**: powerful security assurance
- **Performance**: close to C
- **Productivity**: higher than traditional system programming languages such as C
- Enriched underlying ecosystem

sysMaster features:

- **Security**: memory management issue solved
- **Performance**: no pursuit for ultimate performance of its own
- **Productivity**: no special requirements
- The required third-party libraries are mainly at the bottom layer.

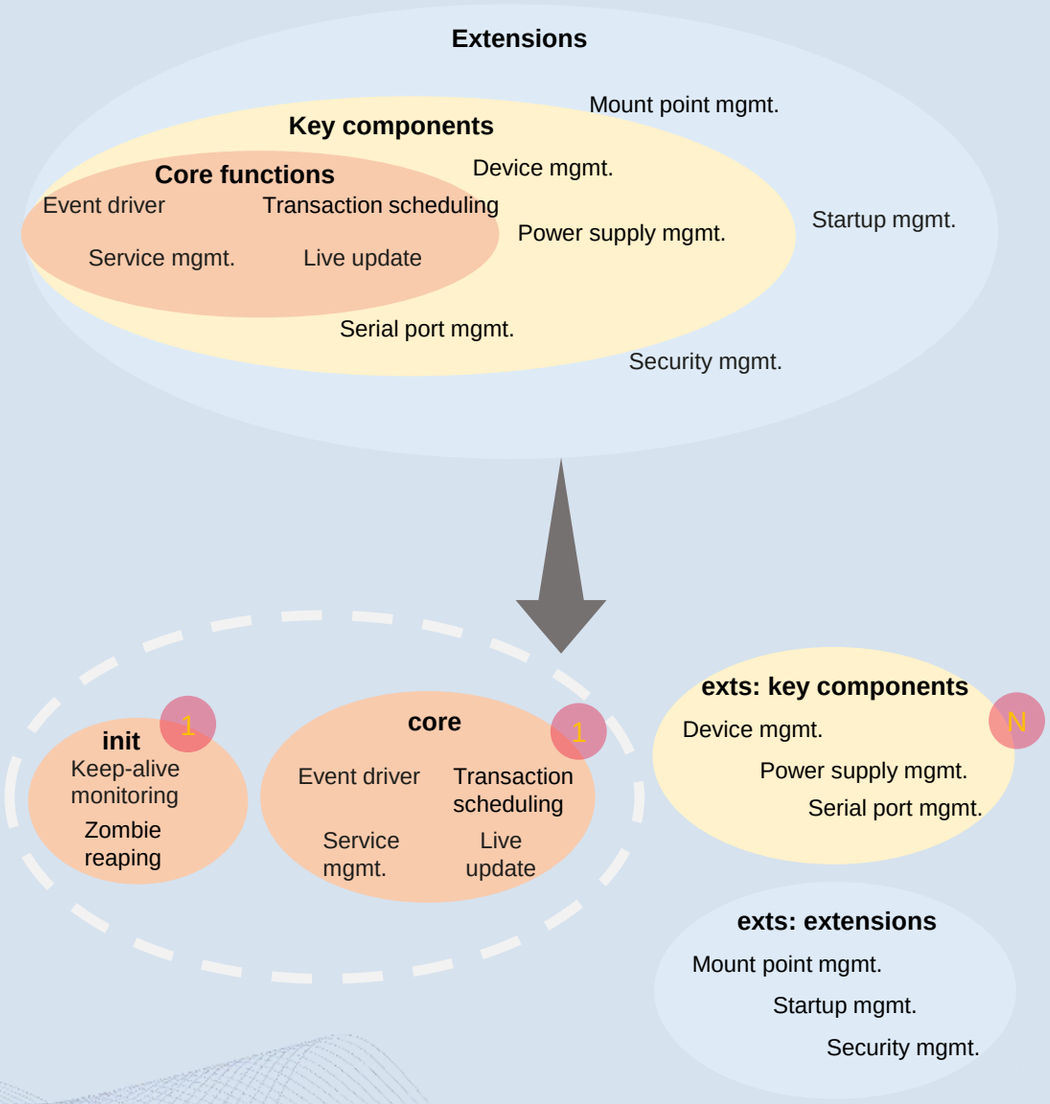


Growing fastest among programming languages in 2022

sysMaster: Design as 1+1+N

Design strategy:

Decoupled and layered to ***1+1+N***



1 init: the real process1

- Simplified functions, KLOC, ultimate reliability.
- Applicable to embedded and simplified systems.
- Implements the basic functions of traditional process1.

1 core: core functions for service management

- Introduces the reliability framework to enable quick self-recovery after a crash.
- Supports Live update and hot restart.

N exts: a collection of components to provide key system functions

- Decouples the functions of the originally coupled components.
- Supports the free combination of modules in different scenarios, like LEGO.

sysMaster: Design Goals and Architecture

Reliability:

- **Reliability:** fault detection + second-level recovery, process1 always online
- **Memory security:** zero memory issues

Easy-to-use:

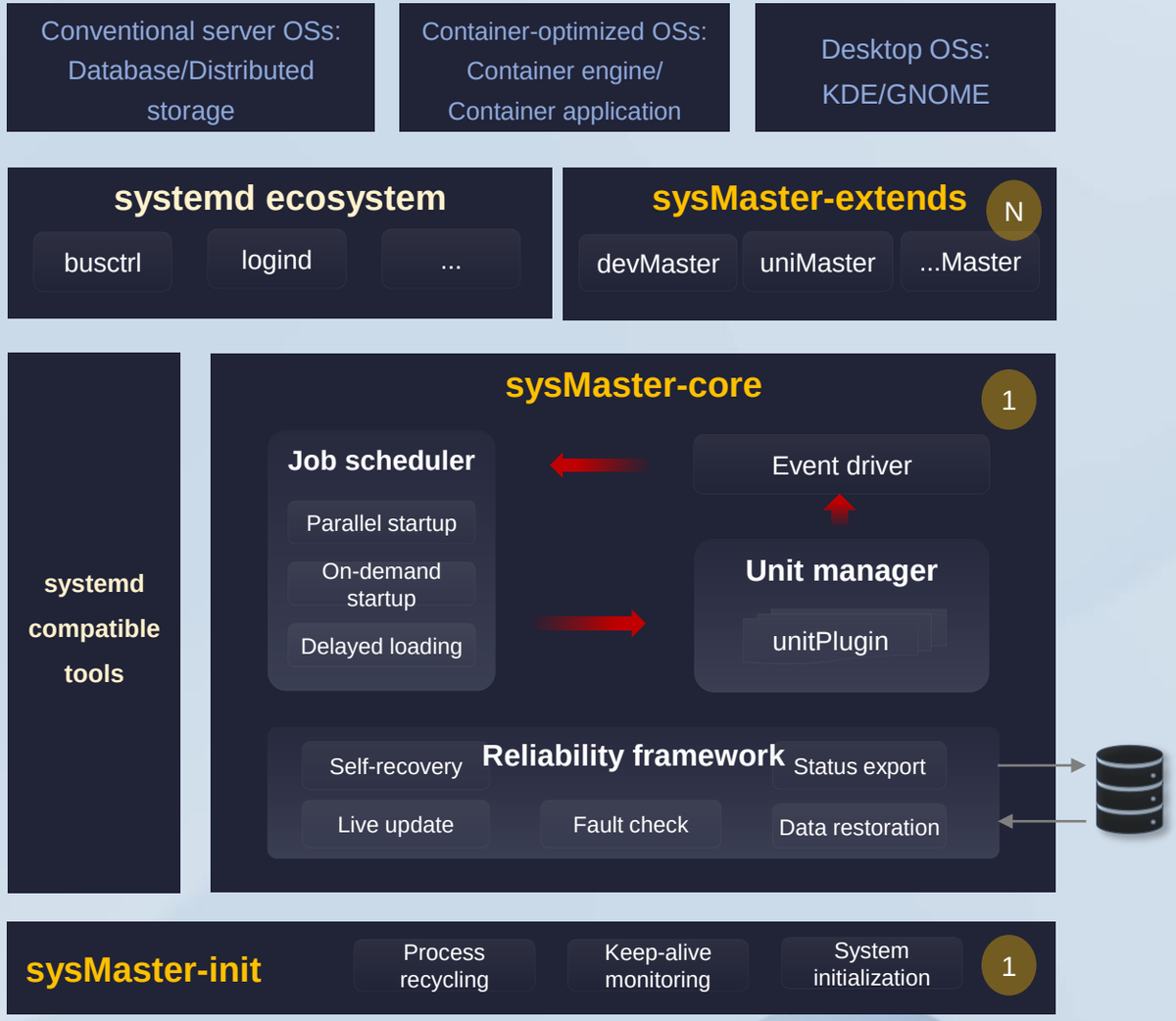
- **Easy O&M:** live update, on-demand tailoring, and flexible assembly
- **Ecosystem compatibility:** systemd conversion tool for ecosystem compatibility

Lightweight:

- **Fewer resources:** memory usage reduced by 10%
- **Faster speed:** startup speed improved by 15%

Customizable:

- **Plug-in mechanism:** flexible extension of multiple service types
- ***1+1+N* architecture:** simplified init functions; non-core functions provided as components



sysMaster-init: Real Process1

Roles: process1 with simplified and reliable functions (already done)

- Zombie reaping
- Monitoring sysMaster-core

Solutions:

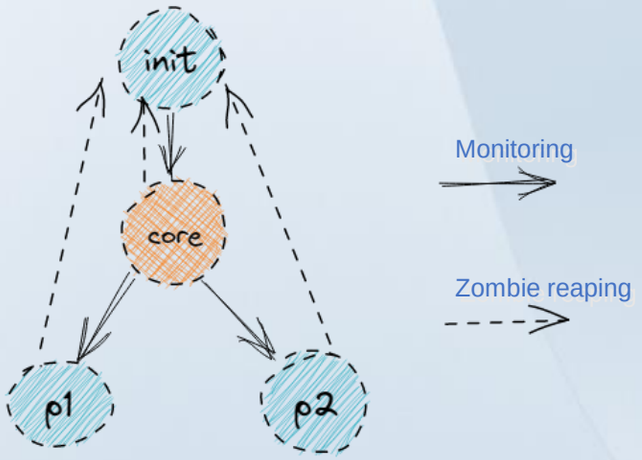
- Only essential functions of process1 are retained. All the other complex functions of traditional process1 are split to sysMaster-core and sysMaster-extends

Results:

- The LOC is fewer than 1K

Scenarios:

- Embedded edge device
- Simplified system



sysMaster-core: Service Management

Roles: Dependency manager, life cycle manager of service units (already done)

Solutions:

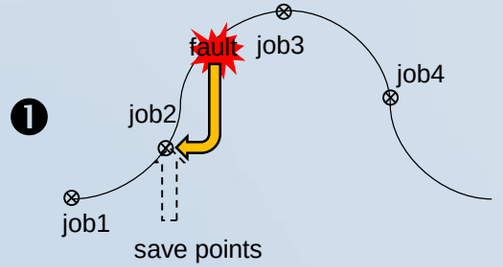
- Reliability framework is added to support hot restart and live update, to improve reliability.
- Plug-in mechanism, event-driven model, and job scheduling are used.

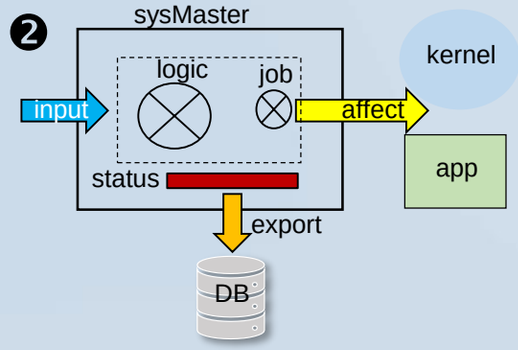
Key Technologies:

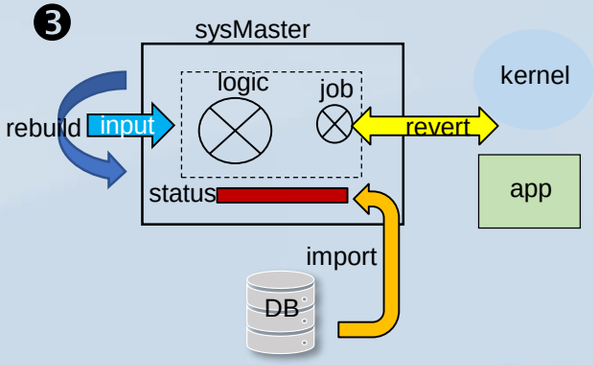
① Input rebuilding + Status export: Export inputs and status to databases.

② save points: Save restoration points.

③ Transaction rollback: Rollback operations on the OS.

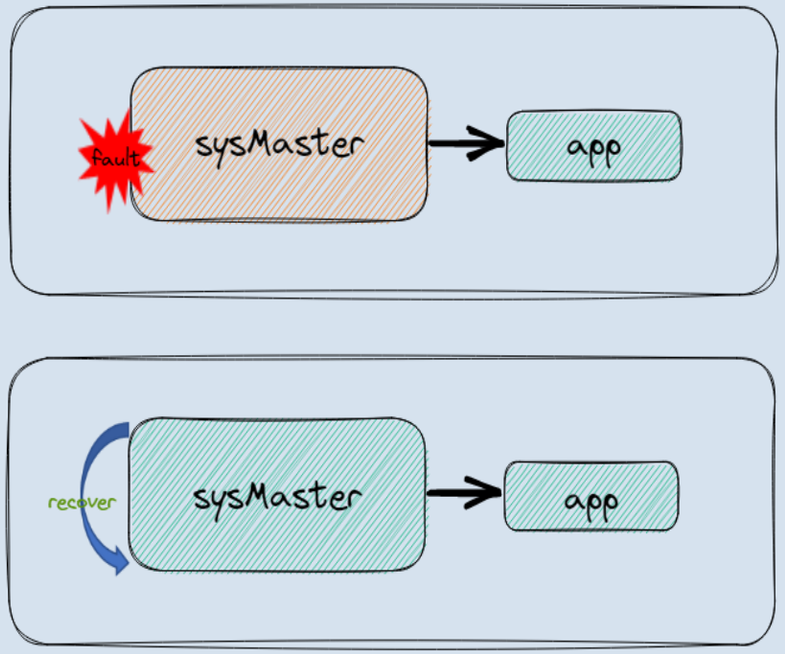






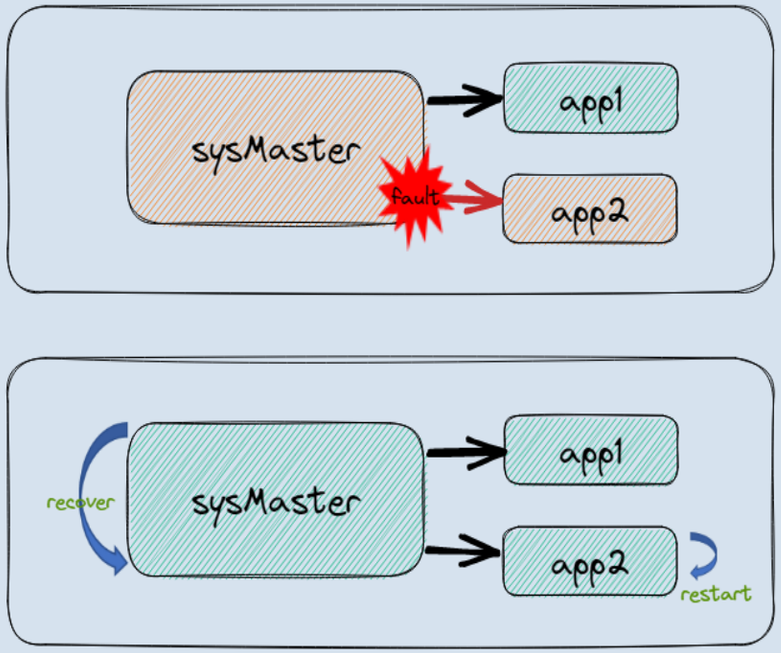
sysMaster-core: Self-recovery

Case 1



sysMaster breaks down during inactivity, and self-recovers.

Case 2



sysMaster breaks down during app operations, and self-recovers

During this period, the OS and services are running properly.

sysMaster-extends: provides N system functions

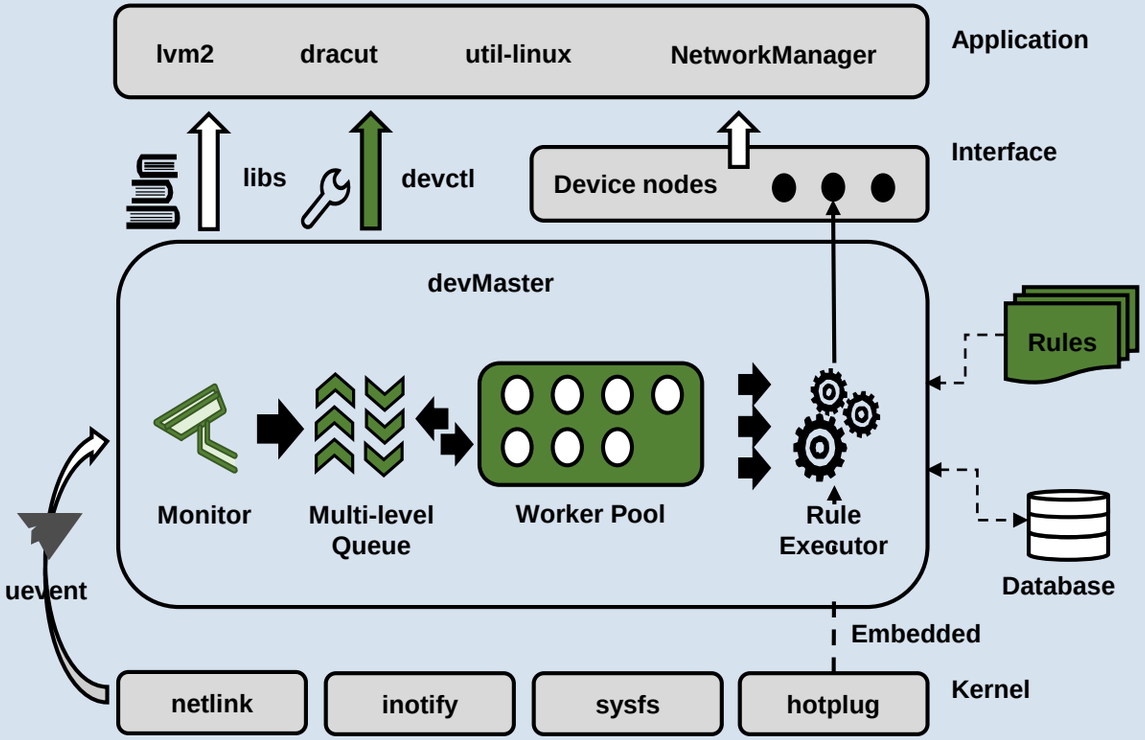
Roles: a collection of components that provide key system functions

Solutions:

- Different system functions by different masters, LEGO combination for different scanrios

Key Technologies:

- **devMaster**: a new device manager (doing)
- **busMaster**: a Rust implementation of the D-Bus protocol (todo)
- **uniMaster**: provides a unified agent (todo) for interconnecting with the CloudNative platform.
- More **Masters** are coming…



devMaster

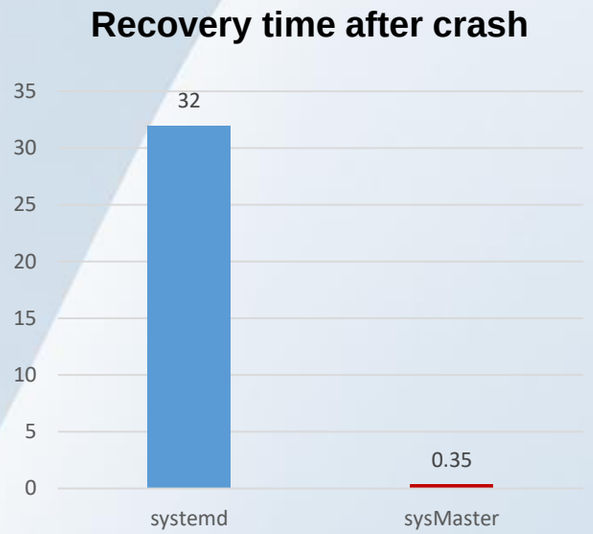
Demos

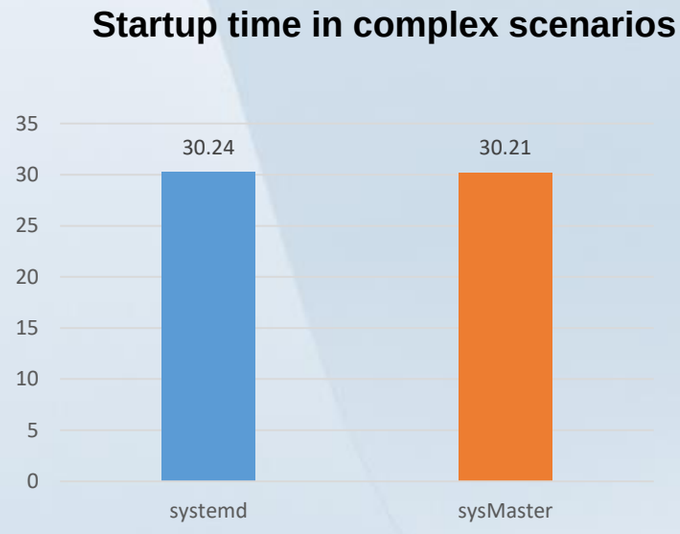
Demo Scenarios:

1. An IoT device that uses a microkernel
2. A container that uses the openEuler iSulad container engine.

Demo Content:

- **Reliability**: Demonstrates the availability and performance of process1 after it breaks down.
- **Performance:** Demonstrates the management service startup performance of process1 in complex scenarios (300+ service units).





openEuler summit 2022 live demo

Future: Keeping the Ball Rolling

2022

Version 0.1

Basic framework

support systemd containers and IOT devices

- Service management
- On-demand and parallel startup
- On-demand loading
- Fault monitoring and recovery
- 1+1+*N* architecture

2023

Version 0.5

Basic functions

support for VMs/GuestOS

- Live update and hot loading
- Device management (devMaster)
- Unit plugin framework
- Event-drive module
- switch_root
- OS startup
- Power management

2024~

Version 1.0

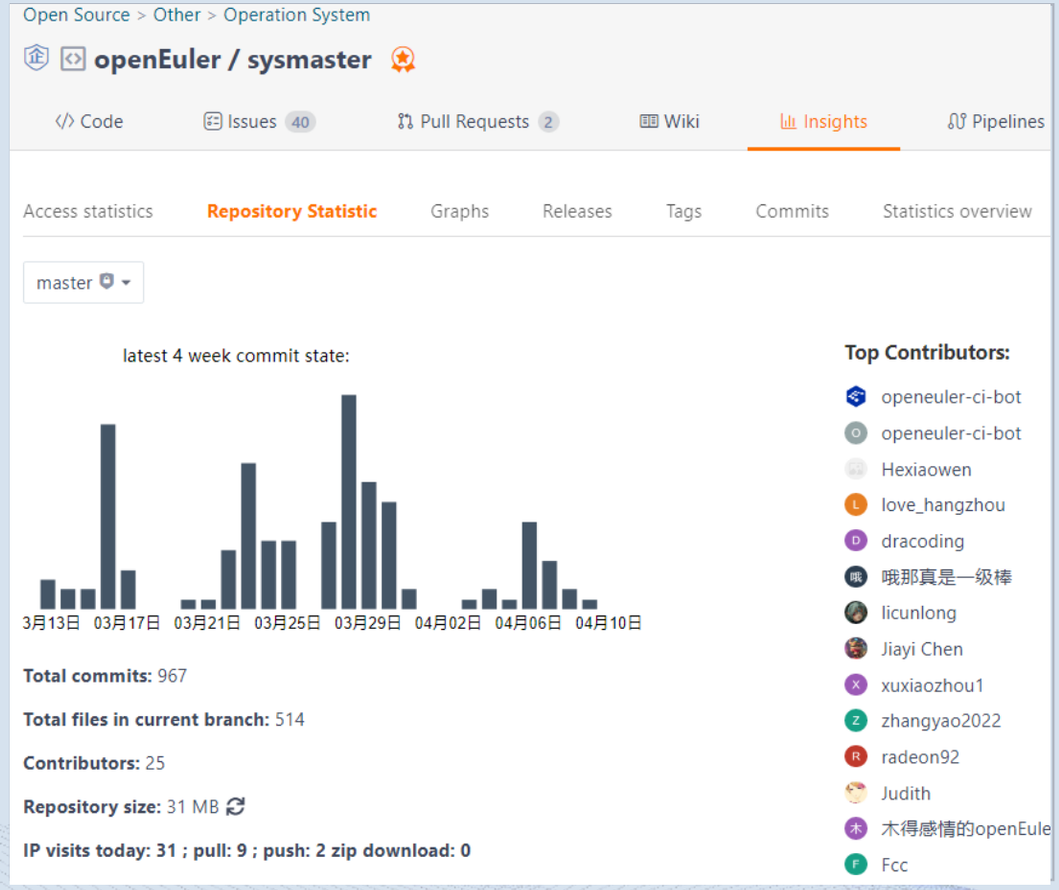
Full functions

support for HostOS and servers

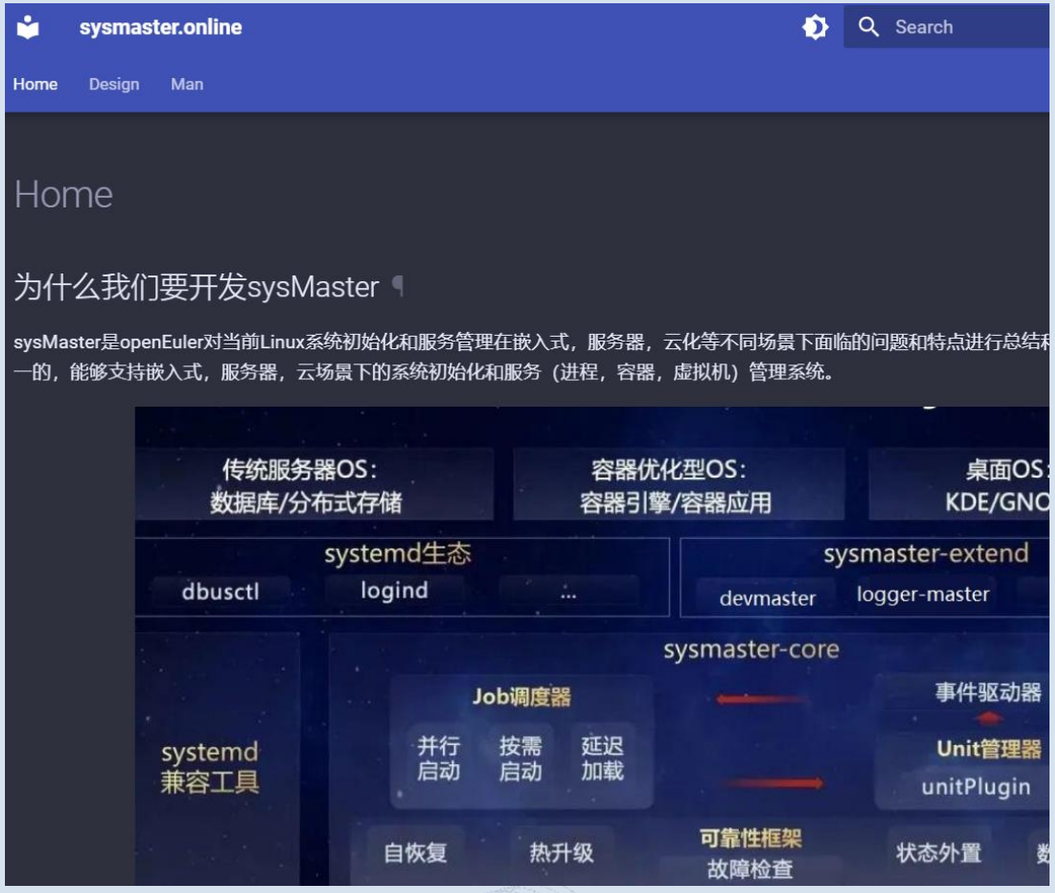
- Ecosystem compatibility tools
- uniMaster (agent)
- busMaster
- cgroup
- Disk imaging tool

Buzzing in Community

gitee.com/openeuler/sysmaster.git



sysmaster.online



How to Engage

@openEuler
https://twitter.com/openEuler

r/openEuler
https://new.reddit.com/r/openEuler/

openEuler
https://www.youtube.com/@openeuler/

openEuler
https://www.linkedin.com/company/86315548/

Official website

LinkedIn newsletter

Join SIGs

Download

openEuler

Inclusive Architecture | Rapid Evolution | Reliable Supply Chain