第二届 eBPF开发者大会

www.ebpftravel.com

基于eBPF的应用层负载均衡的优化实践与探索

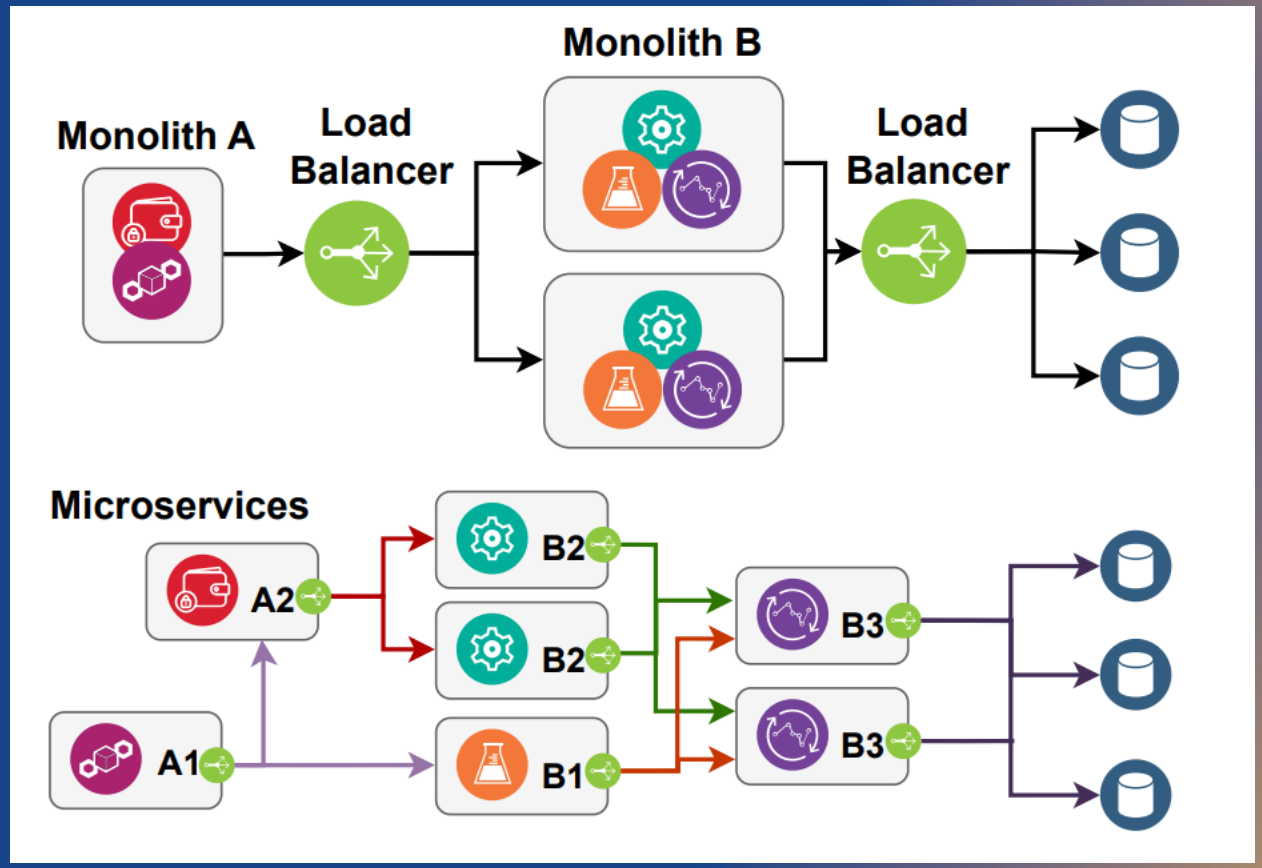
任玉鑫·华为

中国·西安

Background: micro-service

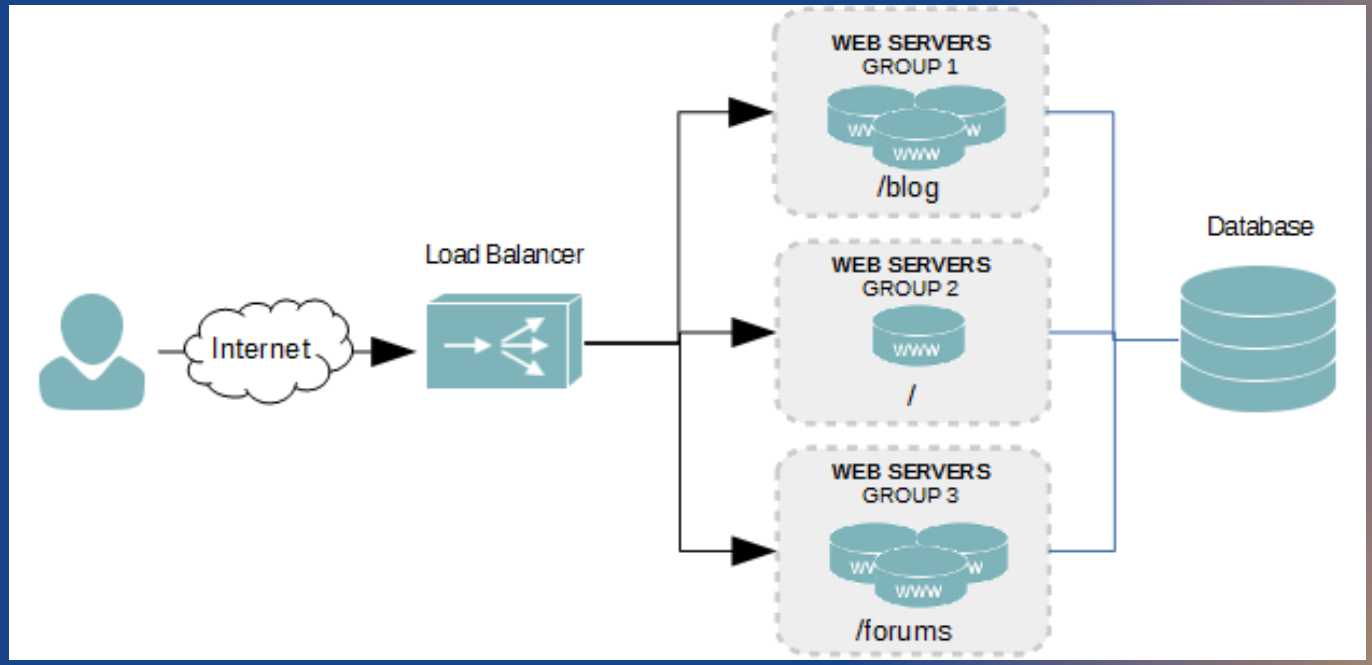
Characteristics

- Long service chain
- Layer-7 load balancing
- Load balancer co-location



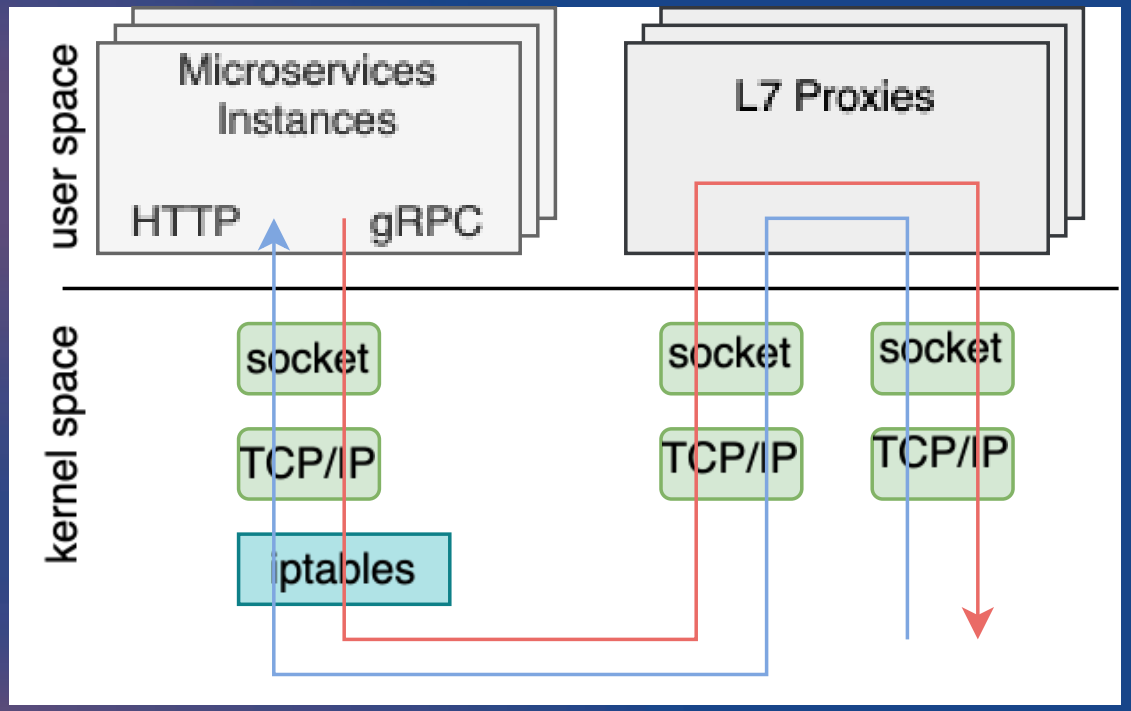
Background: L7 load balancing

- Nginx、HAProxy、Envoy
- Load distribution
- Application awareness
- High performance requirement

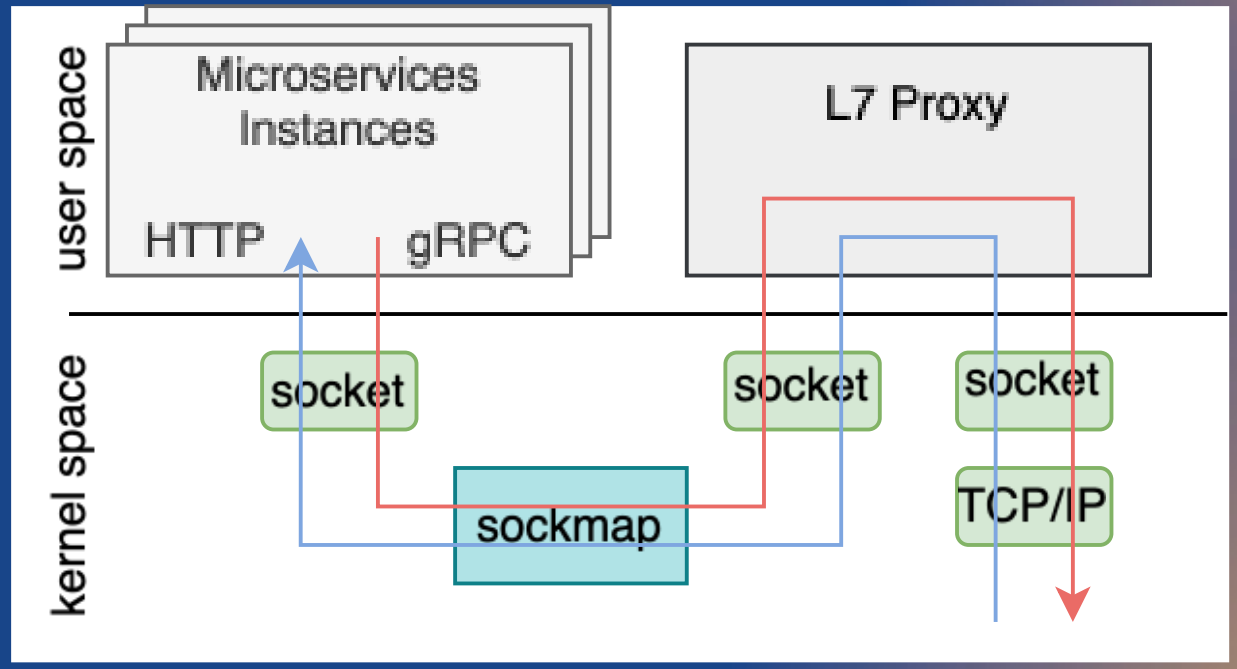


Background: current practice

Istio



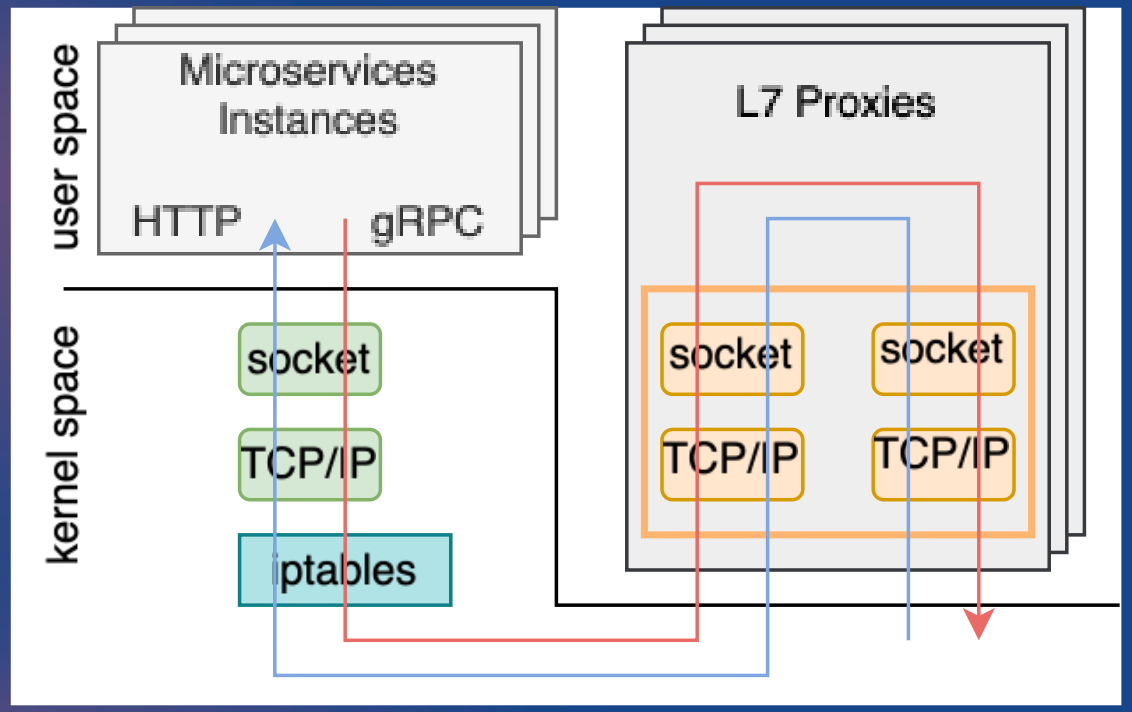
Cilium



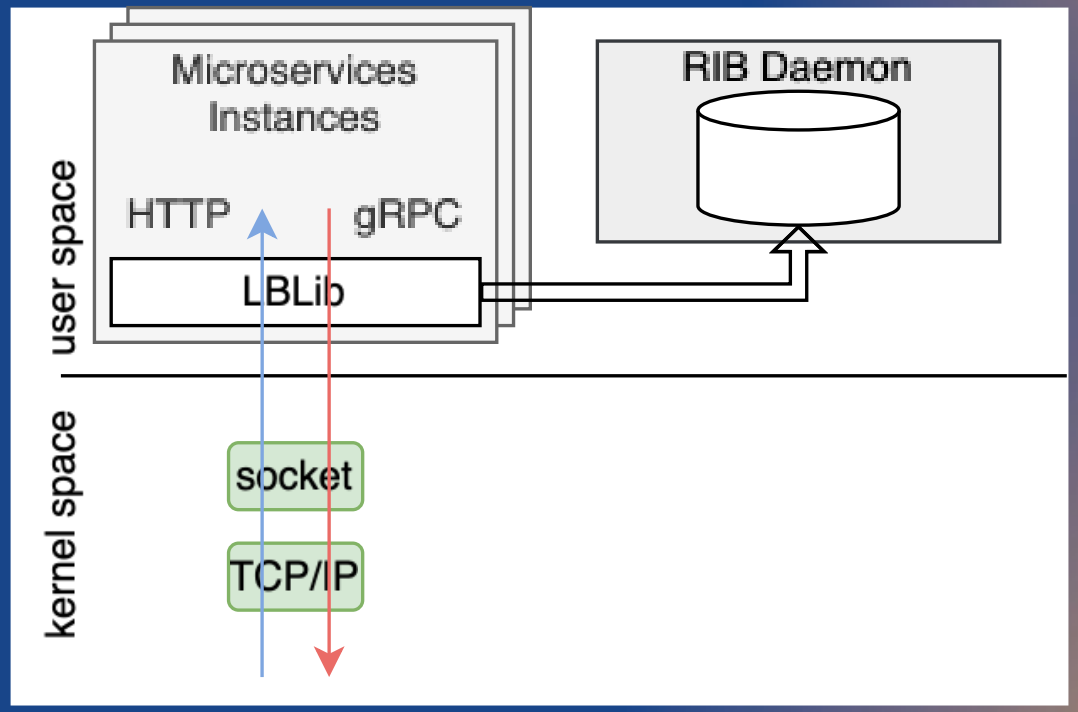
Proxy-based solution: more hops, redundant processing

Background: current practice

DPDK



Library



Security, Isolation, Compatibility issue

Problem summary

	Sidecar	Kernel Bypass	Library
Duplicated Processing	High	High	Low
System Calls	High	Low	High
Cross-process	High	High	Low
Isolation	High	High	Low
Compatibility	High	Low	Low

Approach comparison

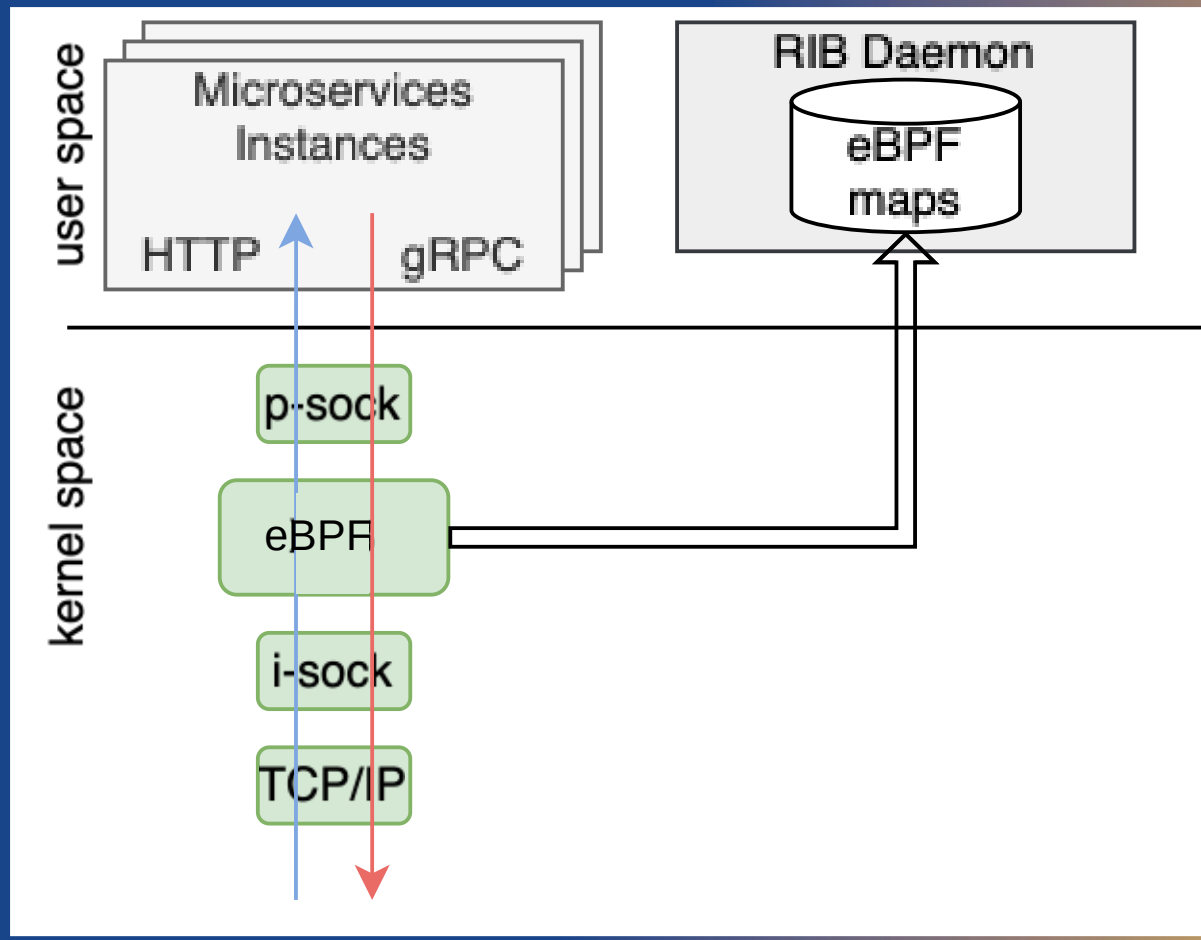
Components	Sidecar Proxy
Protocol parsing	4.5us(5.11%)
Load balancing	13us(14.78%)
Connection splicing	22us(25%)
Socket processing	3.83us(4.35%)
Kernel protocol	26.9us(30.62%)
Others	17.7us(20.12%)

Overhead breakdown

Design: architecture

Goals

- Near-zero unnecessary overhead
- Service isolation and security
- Operational compatibility



Design： benefits

- ✓ Intercept complete message contents
- ✓ Fast message process
- ✓ Flexible message rewriting
- ✓ Modular extension

Design: challenges

- Challenge 1: insufficient and inflexible connection management in the kernel.

- Challenge 2: complex application layer states maintenance in the kernel.

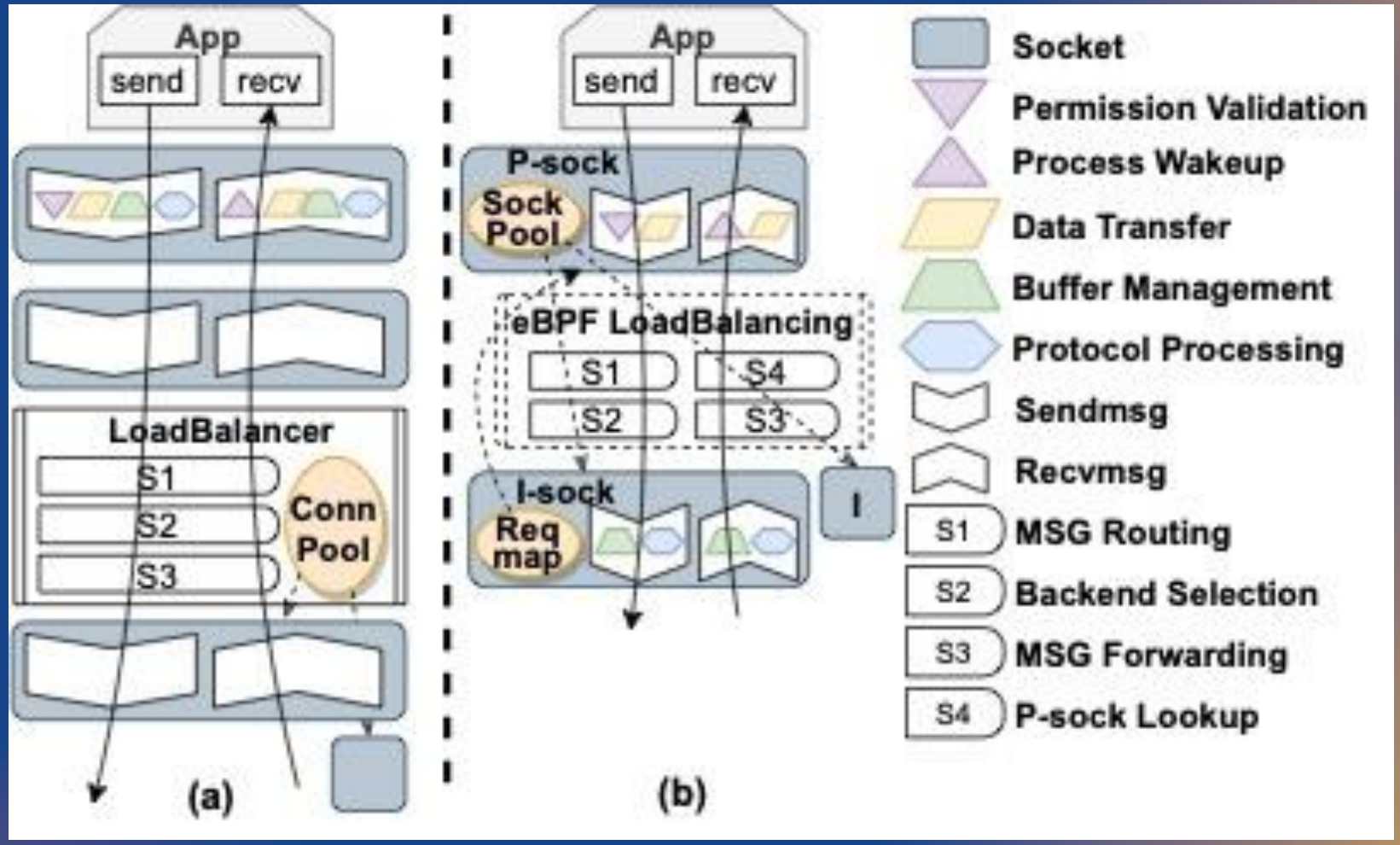
Design： eBPF Interposition

implements backend selection in eBPF to execute in the kernel

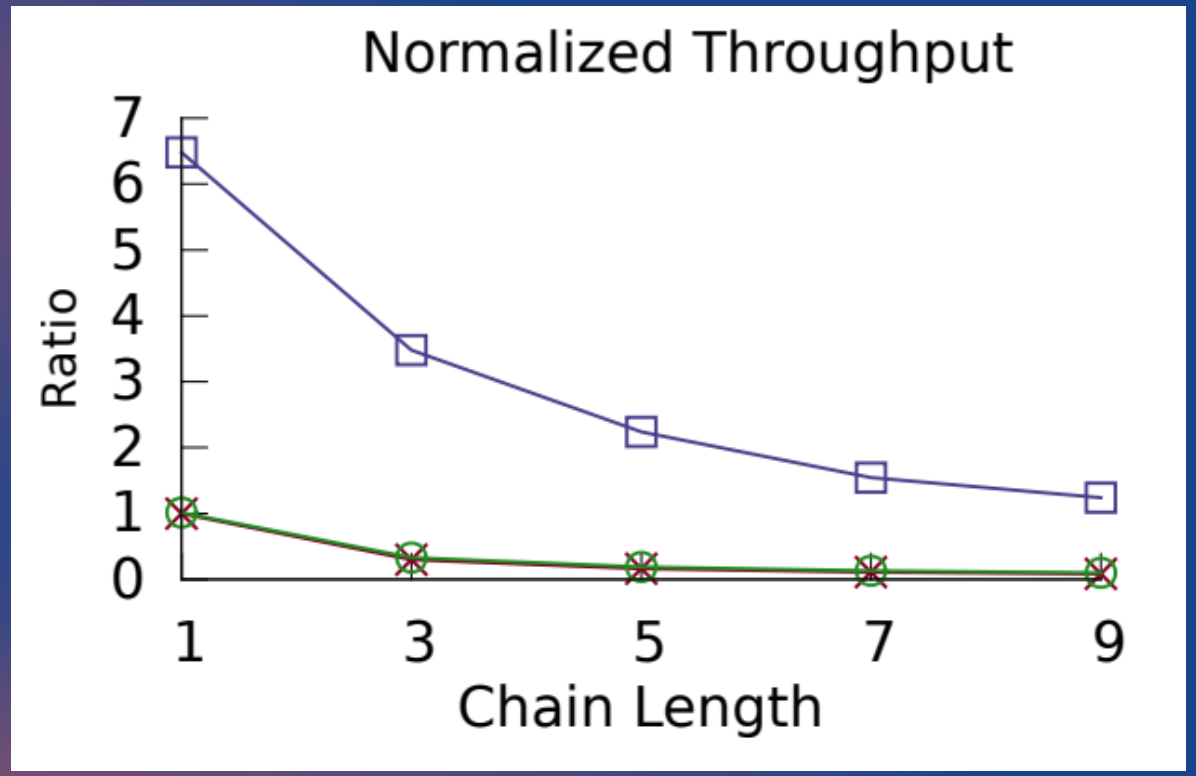
- **Packet parsing：** extract packet contents
- **Service location**: combine message contents with IP
- **Routing**: match requests with routing rules sequentially, and the last matched rule resolves the destination service
- **Instance selection**: conventional loading balancing algorithms, such as round-robin, random, and the least request

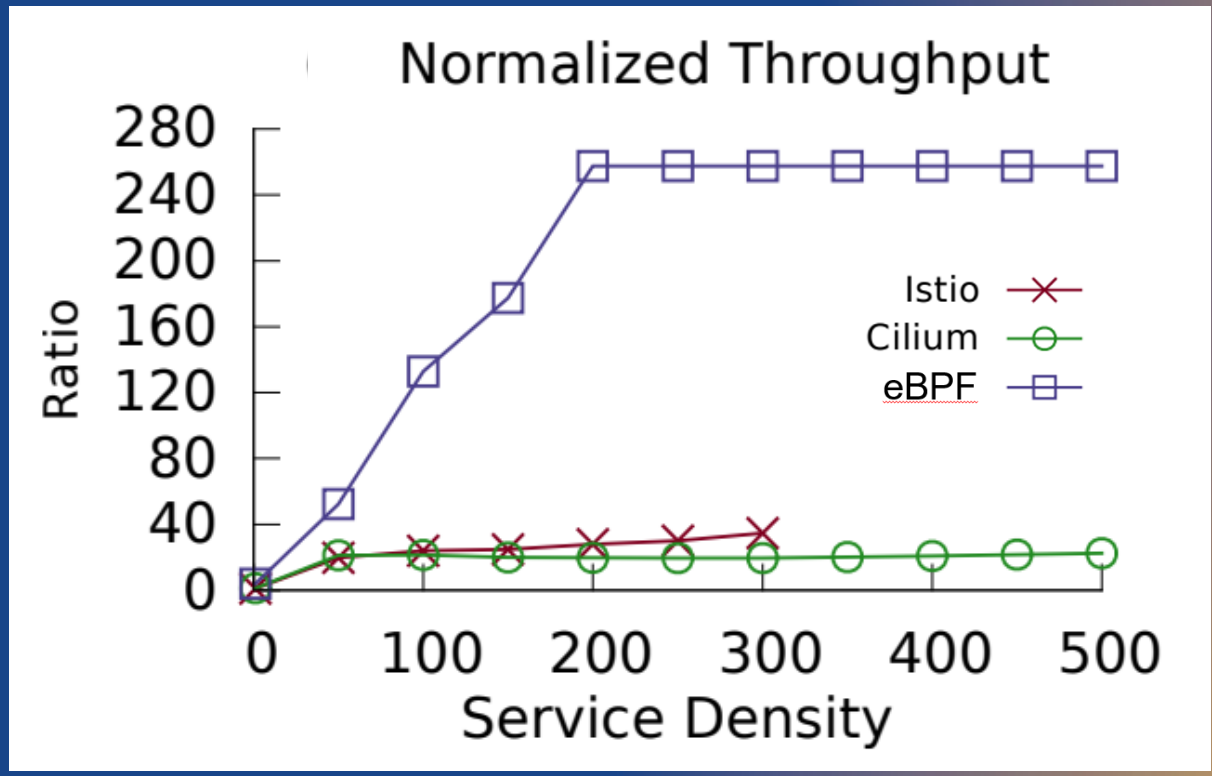
Design： socket redirection

- New socket types
- Connection pool
- Request map



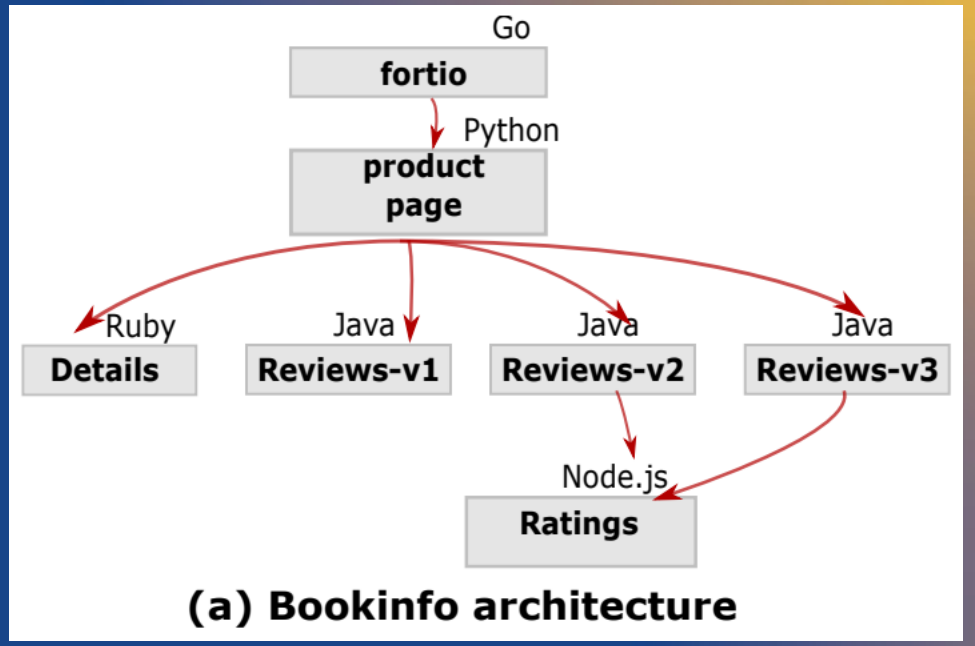
Evaluation： Scalability



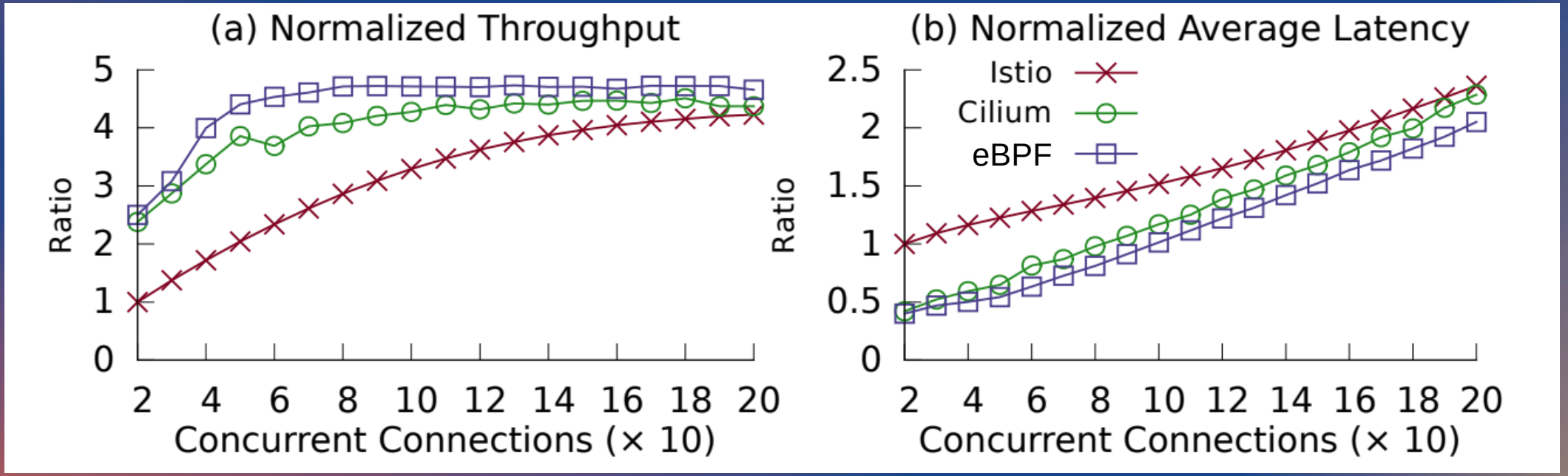


Evaluation： Bookinfo

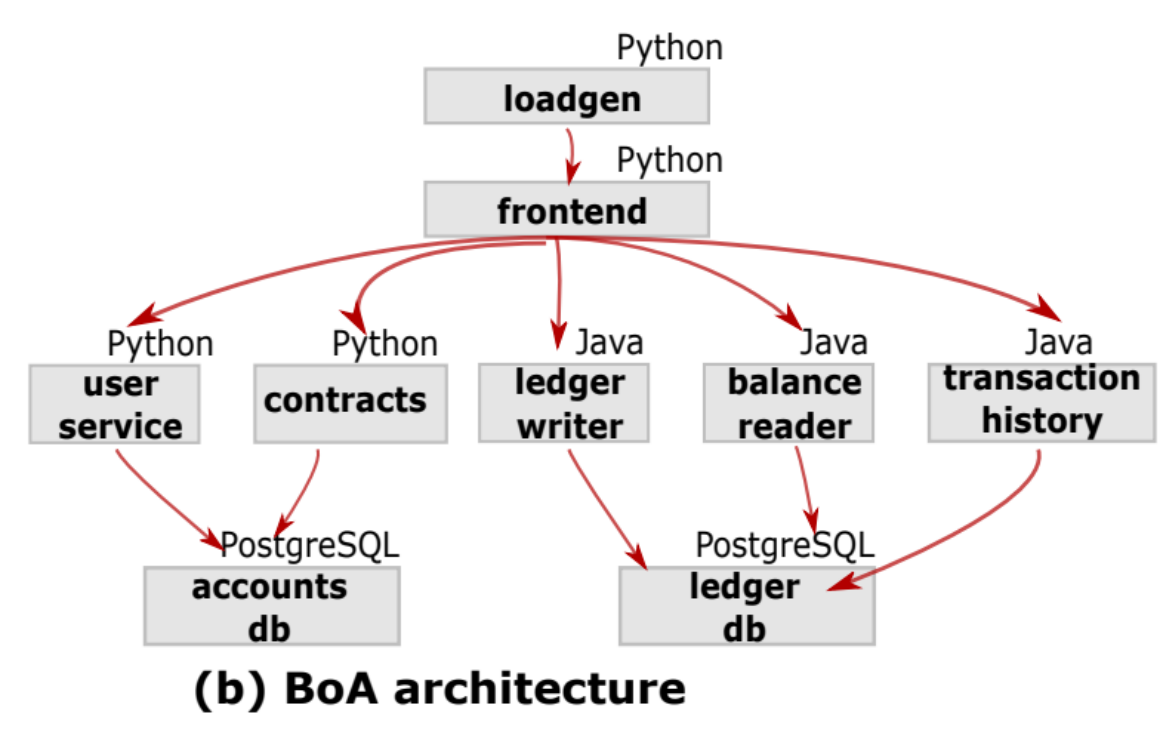
Istio： 43.2% higher throughput**, 33.2% lower** latency
Cilium： 10.2% higher throughput**, 13.3% lower** latency



(a) Bookinfo architecture



Evaluation: Google BoA



(b) BoA architecture

All data are normalized to Istio 5 users result

system	users	throughput	latency	tail atency
Istio	5	1.0	1.0	1.0
	500	13.11	7.59	4.79
Cilium	5	0.96	1.02	0.98
	500	13.27	7.52	3.94
eBPF	5	1.26	0.79	0.84
	500	14.47	6.73	3.66

(c) BoA performance

Future work

- More protocols support
- More L7 functionalities
- Coordinating with user-space customized features
- Online update
- Reliability

Open source

- Kmesh：an efficient in-kernel service mesh framework

https://github.com/kmesh-net/kmesh

Will integrate into kmesh!

平滑兼容	高性能	低开销
• 应用无感的流量治理 • 自动对接Istio等软件	• 网格转发时延60%↓ • 服务启动性能40%↑	• 网格底座开销70%↓

安全隔离	全栈可视化	开放生态
• ebpf虚机安全 • cgroup级编排隔离	• 端到端指标采集* • 主流观测平台对接*	• 支持xDS协议标准

Kmesh技术交流群

KMESH

OpenEuler