# 基础设施层现状

目前，微服务架构通过将软件应用分解为微小且独立的服务，服务之间通过轻量级机制实现通信逐渐成为软件开发架构的范式。微服务架构的特点对运维治理也提出了新的要求。基础设施层作为微服务架构中最底层的部分，为微服务架构的实现提供了最基础的功能。如何利用基础设施层为服务治理提供更可靠的机制是目前工业界和学术界研究的热点。

工业界对于基础设施层的研究主要集中于工具的实现。Docker通过容器机制，捆绑了应用程序正常运行所需的所有文件和库，为软件的运行提供了隔离机制，却没有考虑到服务治理的实现，没有提供负载均衡、弹性伸缩、滚动更新和监控等功能[1]。Docker Swarm和Kubernetes在容器技术的基础上，提供了服务治理的功能，通过对CPU、内存、网络请求和QoS等数据的监控，对微服务实例进行伸缩和滚动更新[2]。Istio在Kubernetes的基础上，通过边车注入的模式，在不影响Kubernetes原有功能的基础上，使得微服务实例之间的通信更加透明，进一步地提供了微服务应用的性能[3]。

学术界对于基础设施层的研究主要集中于在特定领域对工业界存在的软件进行扩展。Alexiou团队基于Docker Swarm设置了一个资助代理系统Elixir，通过响应式监控内存、CPU等静态指标，并引入一个调度器来平衡节点之间的工作负载进而实现实时监控资源[4]。Bulej团队基于Kubernetes提出了一种自适应Kubernetes云控制器，用于调度时间敏感的应用程序，为实时系统提供服务，通过监控响应时间和吞吐量在共享云资源上调度服务以满足要求[5]。

[1] Uphill T, Arundel J, Khare N, et al. DevOps: Puppet, Docker, and Kubernetes[M]. Packt Publishing Ltd, 2017.

[2] Wei-guo Z, Xi-lin M, Jin-zhong Z. Research on Kubernetes’ Resource Scheduling Scheme[C]//Proceedings of the 8th International Conference on Communication and Network Security. 2018: 144-148.

[3] Larsson L, Tärneberg W, Klein C, et al. Impact of etcd deployment on Kubernetes, Istio, and application performance[J]. Software: Practice and Experience, 2020, 50(10): 1986-2007.

[4] Alexiou M S, Petrakis E G M. Elixir: An Agent for Supporting Elasticity in Docker Swarm[C]//AINA. 2020: 1114-1125.

[5] Bulej L, Bureš T, Hnětynka P, et al. Self-adaptive K8S Cloud Controller for Time-sensitive Applications[C]//2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2021: 166-169.