Composing Microservices and Serverless for Load Resilience

Microservices architecture has become widely popular for modern application development due to its ability to break down applications into smaller, independent services, making development, deployment, and maintenance more manageable. However, a major challenge faced by microservices is efficiently scaling compute resources to handle fluctuating and unexpected spikes in traffic. Typically deployed as containers within virtual machines (VMs), microservices struggle to scale resources efficiently. At present, companies often allocate more resources than necessary to their microservice systems in anticipation of unexpected increases in demand, resulting in excess costs. However, these resources typically remain unused during periods of low demand.

Currently, two distinct strategies are employed to address microservices scalability: proactive and reactive scaling. Proactive scaling involves preemptively allocating resources based on anticipated demand, while reactive scaling adjusts resources dynamically in response to real-time changes in demand or performance metrics. While proactive scaling attempts to manage regular load fluctuations based on forecasts, it often fails to address sudden increases in traffic due to their unpredictable nature. In contrast, reactive scaling can accommodate unforeseen traffic surges but is hindered by the time it takes for scaling events to occur in microservice frameworks, typically requiring several seconds to complete, or even longer if new virtual machines need to be initiated.

Recognizing the challenges, serverless computing emerges as a promising solution due to its elasticity and ultra-fast startup times. With serverless computing, users only pay for the actual resources used, and cloud providers manage resource allocation, provisioning, and scaling on-demand. By leveraging the above insight, we propose Hydra, a hybrid architecture that combines VM-based microservices with serverless computing. Under normal load, Hydra operates online applications as VM-based microservices, as commonly done in current deployments. During load spikes, Hydra seamlessly incorporates serverless components to handle excess load while launching new microservice instances in the background.

Our evaluation demonstrates that Hydra significantly reduces peak tail latency by 62.4% compared to Kubernetes auto-scaling mechanisms, with only a minimal 2.3% increase in cost. This underscores Hydra's effectiveness in achieving load resilience cost-efficiently within modern online service architectures.