Problem Statement

- Large tech companies run apps with 100’s of microservices
- Kubernetes and Docker Compose help with application deployment but not with performance
- Current approach is to overprovision resources to microservices
- How can we quickly determine ideal resource configuration whilst limiting machine costs?

Two Approaches

**Gradient Descent**
- Throttles resources to measure service dependency on resource type; then iteratively transfers resources from less impacted to more impacted services
- Estimates gradient through performance drop after resource throttling
- Explores input space exhaustively, providing eventual significant performance gains

**Bayesian Optimization**
- Random search followed by Iterative Gaussian regression over best candidates
- Performance dependent on duration of random search as well as various parameters of randomness, such as min-neighbor finding
- Provides quick improvements, offering performance gains that compare to those made by exhaustive searches

Applications

1. **Bootstrapped Gradient Descent**, using configurations from Bayesian Random Search
2. **Improve Bayesian Optimization performance** by tuning random search length and min-neighbor finding randomness
3. **Dual-Optimization over the space of resources and replicas** to find a replica configuration that limits a user’s cost function while maximizing performance

Results and Comparisons

1. **Spearmint (Bayesian Optimization) vs Bootstrapped (BO + GD)**
2. **Steady State Performance After Varied Random Search Duration (BO)**