Automated Performance Problem Mitigation in Multi-Service Apps
Elie Krevat, Greg Ganger (CMU)

OVERVIEW
- Many apps are composed of multiple shared services
  › Problems in one service can cause cascading delays
- Performance changes happen often in unexpected ways
  › Problem diagnosis across services is time-consuming
- Goal: Automated performance problem mitigation
  › Short-term fix before diagnosis
- Approach: Apply more or better resources where needed
  › Feedback loop informed by req flows + resource usage

RIGHT-SIZING FEEDBACK APPROACH
- Invoke workflow in response to problem (simple detector)
  › Service Level Objectives (SLOs) + free machines
- Exploit request flows for automation
  › Discover global flow and synchronicity
  › Find bottleneck services on critical path
- Predict improvement potential from recent history
  › Assess elasticity properties
- Assign resources efficiently (limit overprovisioning)
  › Informed by resource usage demands
- Observe and evaluate gains

EXAMPLE SCENARIO
- LOAD CHANGE: WEB INDEXER requests
  › RIGHT-SIZE: +1 to SEARCH
  › +1 again after feedback

SHARED SERVICE ARCHITECTURE
- Complex dependencies exist between services
  › Managed by separate teams

SELECTING DIFFERENT RESOURCES
- Manage costs and avoid overprovisioning
  › Two primary "quick fixes":
    › More machines: good for overloaded services
    › Better machines: good for bottleneck resources

EVALUATION PLAN
- Build system of mock services/workloads
- Instrument end-to-end request tracing w/ resource usage
- Inject synthetic performance problems
  › Service slowdown, overload, dependency change
- Compare right-sizing against baselines
  › Static vs. limited resource usage or request flow info
- Evaluation criteria:
  › Localization accuracy
  › Problem mitigation time