Triage: Performance Differentiation for Storage Systems Using Adaptive Control

Presentation of a paper by M. Karlsson, C. Karamanolis, and X. Zhu from the ACM Transactions on Storage 1(4), Nov 2005

Ken Salem

David R. Cheriton School of Computer Science
University of Waterloo

November 1, 2006

System Model and Goals

- storage clients sends block requests to a shared storage system

System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload

System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload
- each workload has a latency requirement

System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload
- each workload has a latency requirement
- each workload has a throughput allocation target

System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload
- each workload has a latency requirement
- each workload has a throughput allocation target
- goals
  1. : enforce latency requirements

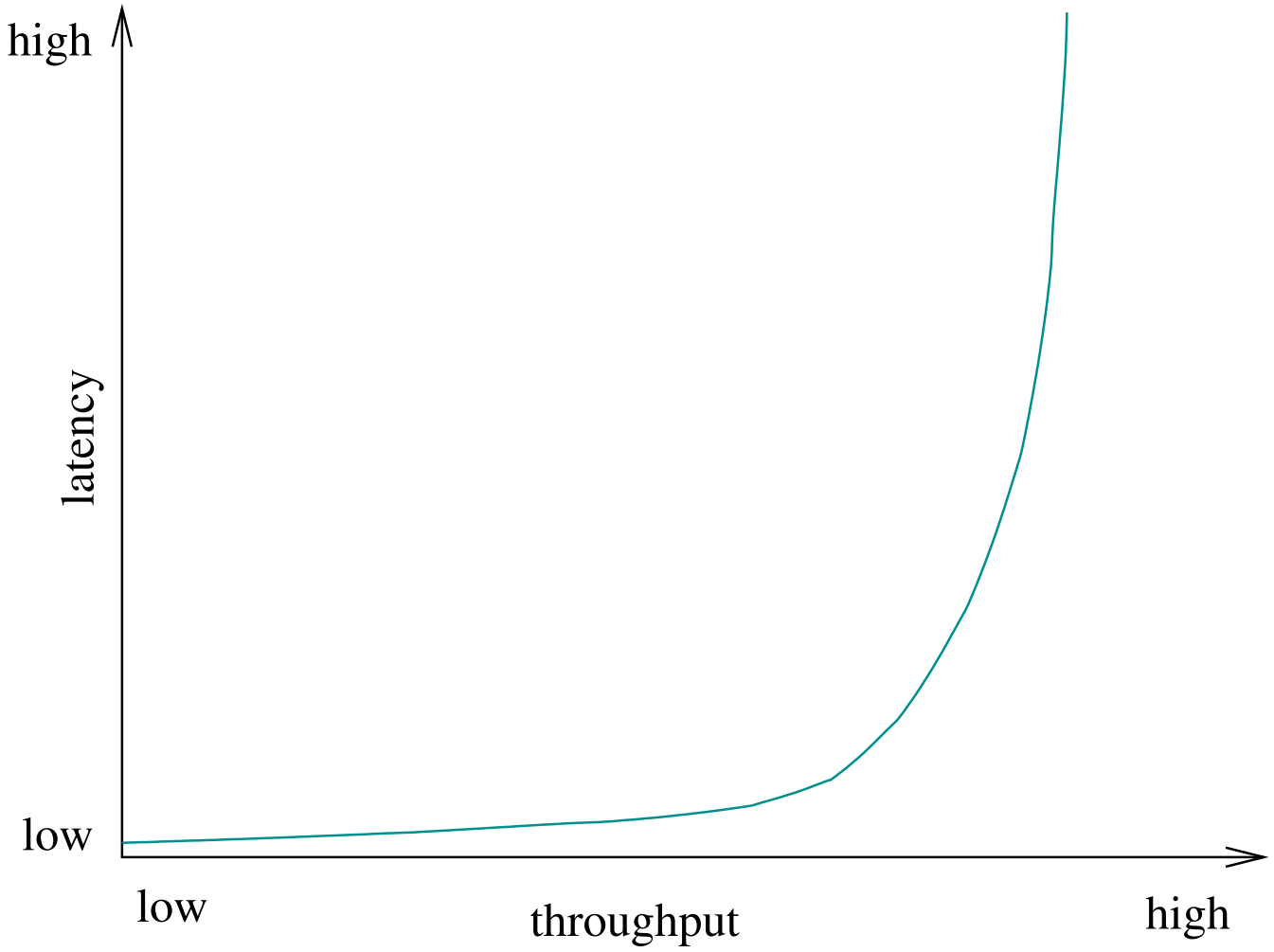
System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload
- each workload has a latency requirement
- each workload has a throughput allocation target
- goals
  1. : enforce latency requirements
  2. : maximize aggregate throughput (across all workloads)

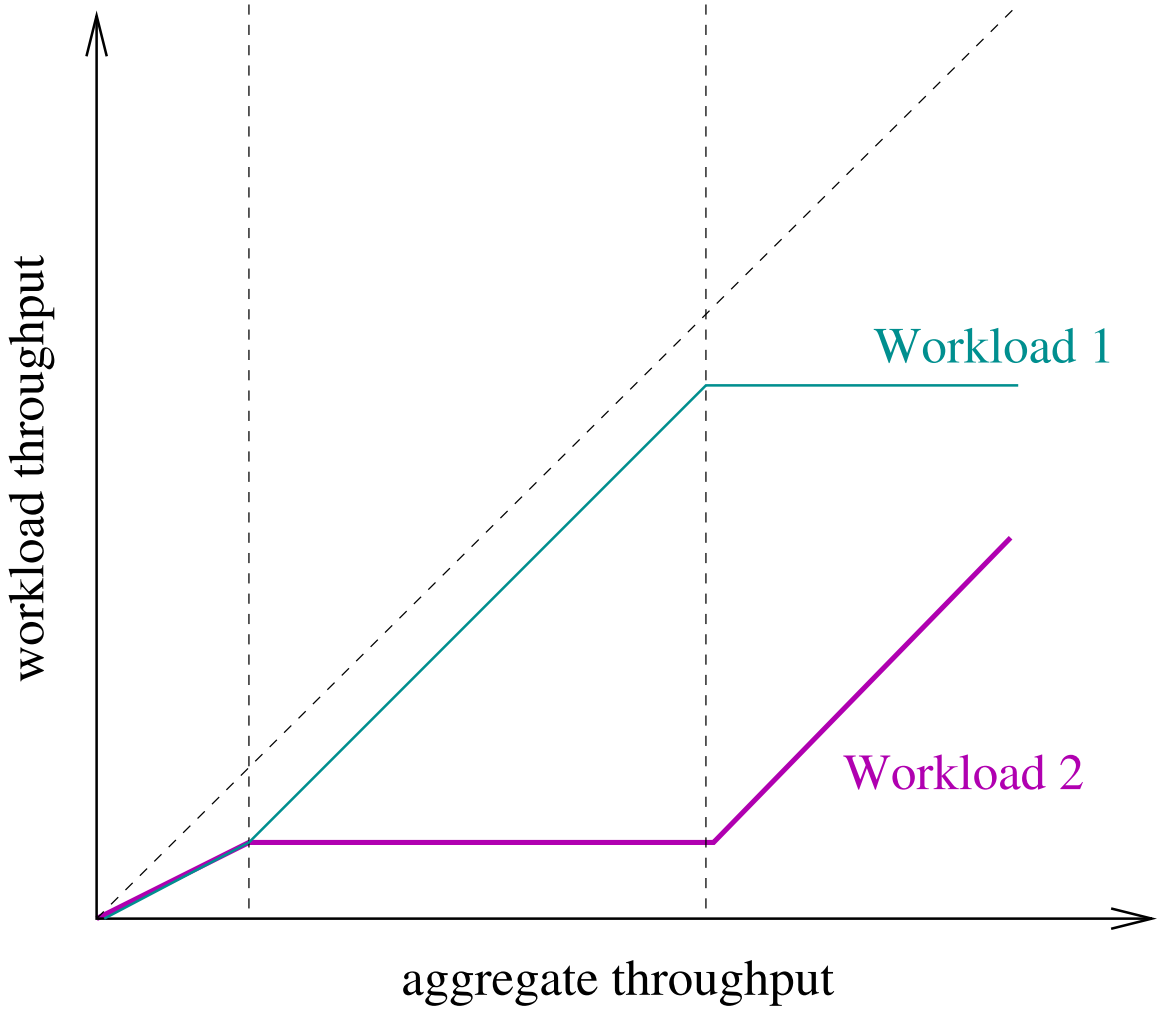
System Model and Goals

- storage clients sends block requests to a shared storage system
- each request is part of a workload
- each workload has a latency requirement
- each workload has a throughput allocation target
- goals
  1. : enforce latency requirements
  2. : maximize aggregate throughput (across all workloads)
  3. : allocate total throughput among workloads

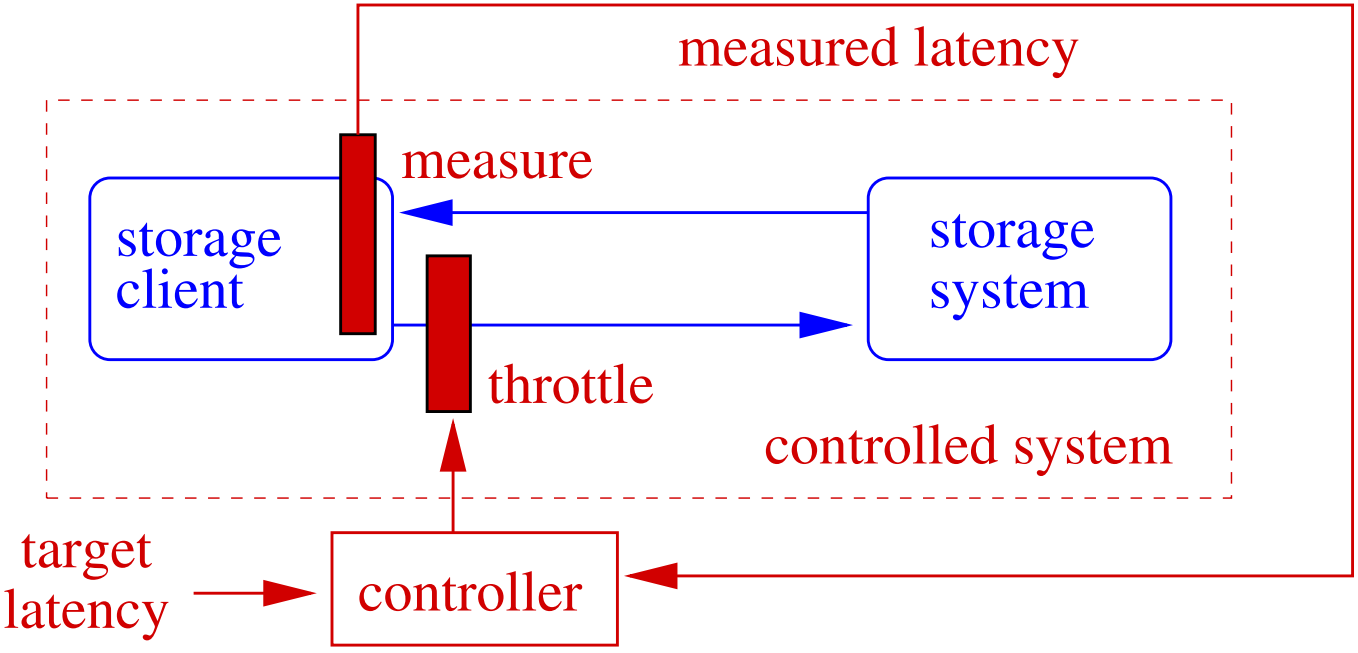
Throughput/Latency Tradeoff



Throughput Allocation Model



Approach #1: Non-adaptive Control



Non-adaptive Controller Design

- model the system to be controlled:

$$y(k) = \alpha y(k-1) + \beta u(k-1)$$

  Estimate α and β by fitting to observations under a calibration workload.
- build two models, one for calibration workload with lots of data locality, one for workload with little locality
- for each model, design a good (stable, accurate, short settling time, little overshoot) I controller.

Non-adaptive Controller Design

- model the system to be controlled:

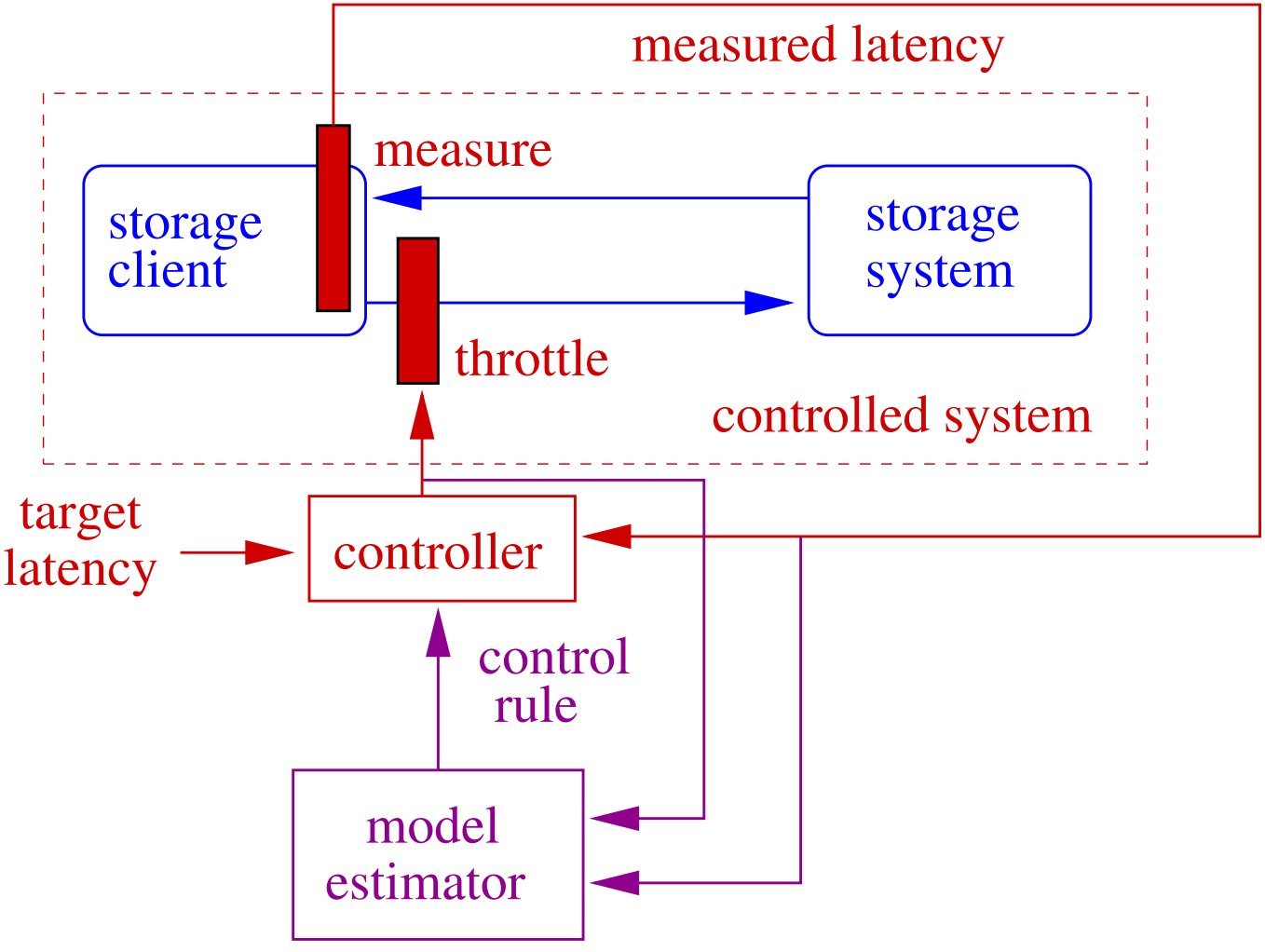
$$y(k) = \alpha y(k-1) + \beta u(k-1)$$

  Estimate α and β by fitting to observations under a calibration workload.
- build two models, one for calibration workload with lots of data locality, one for workload with little locality
- for each model, design a good (stable, accurate, short settling time, little overshoot) I controller.

Problem

No controller works well for both system models.

Approach #2: Adaptive Control



Adaptive vs. Non-Adaptive Control

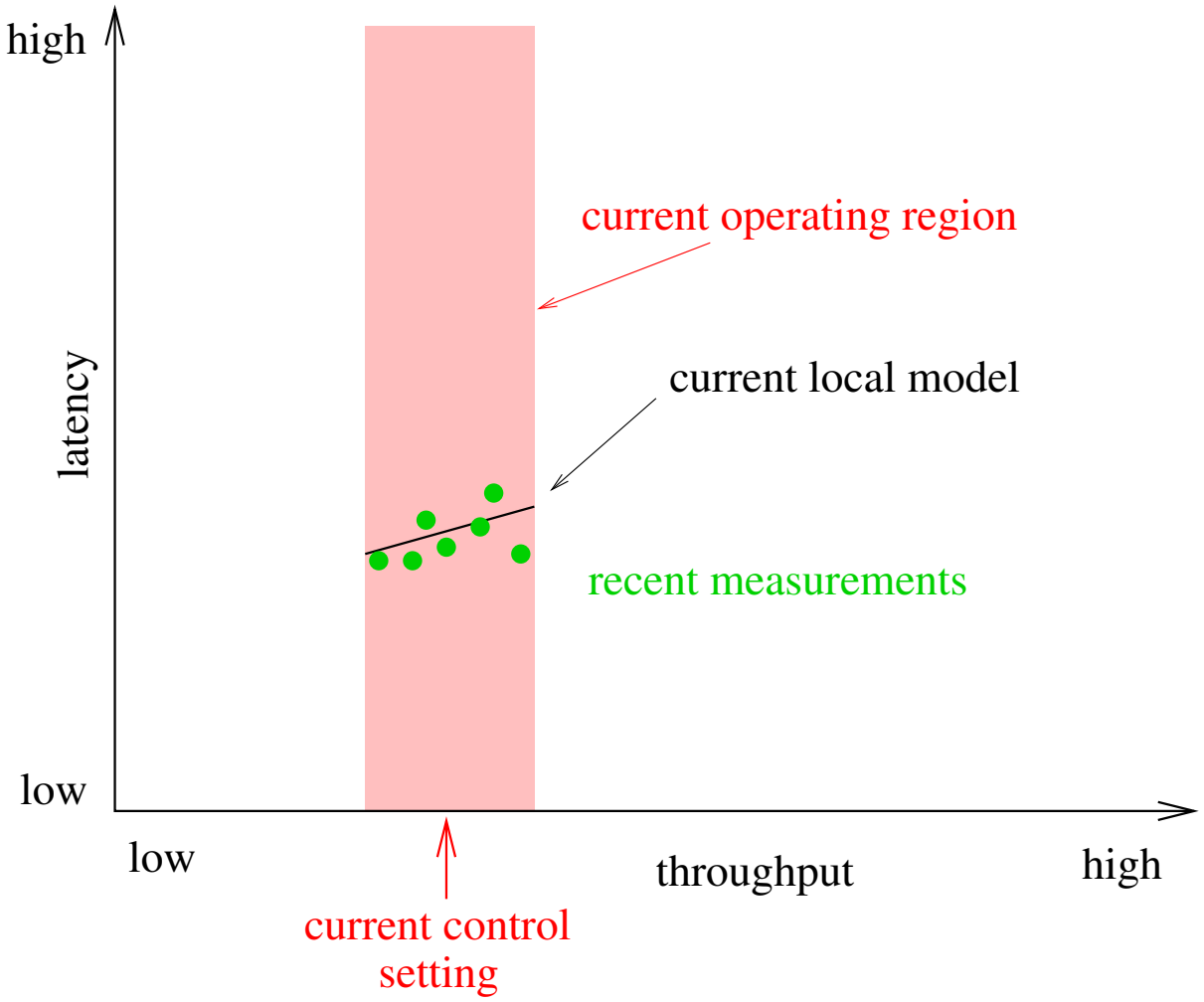
Non-Adaptive:

- offline: build a global system model
- offline: design a controller for that model
- on-line: use the controller - model is not used

Adaptive:

- on-line: continuously update model of system in the current operating region
- on-line: control rule uses current model

Adaptive Models



Observations

- Adaptive controllers harder to reason about than non-adaptive controllers. (Stability almost proved, but no analysis of other properties.)

Observations

- Adaptive controllers harder to reason about than non-adaptive controllers. (Stability almost proved, but no analysis of other properties.)
- Control rule has parameters and special cases: “age out” old models instead of replacing them, check actuator setting for boundary conditions, check for model divergence.

Observations

- Adaptive controllers harder to reason about than non-adaptive controllers. (Stability almost proved, but no analysis of other properties.)
- Control rule has parameters and special cases: “age out” old models instead of replacing them, check actuator setting for boundary conditions, check for model divergence.
- This approach is very general. What kind of system would such a controller not work for?

Observations

- Adaptive controllers harder to reason about than non-adaptive controllers. (Stability almost proved, but no analysis of other properties.)
- Control rule has parameters and special cases: “age out” old models instead of replacing them, check actuator setting for boundary conditions, check for model divergence.
- This approach is very general. What kind of system would such a controller not work for?
- Triage actually uses a distributed controller implementation. One controller per workload. Each controller recommends an aggregate throughput based on latency target and observed latencies for its own workload. System uses the minimum of the throughput recommendations.