What is Performance Anomaly?

- Performance falls below expectation
  - What is the right expectation?
- Poor performance that cannot be explained by high-level design features and intuitions
- Often due to implementation deviations from high-level design
  - unintentional deviations that are obviously wrong (bugs)
  - intentional deviations that address practical issues that escaped high-level design

Performance Anomaly Identification

- How do you identify a performance anomaly?
  - Given two executions $A$ and $R$, if $A$ delivers much worse performance than $R$ (against "expectation"), then you call $A$ an anomaly in relation to $R$ the reference.

A concrete example:
- Anomaly – Linux 2.6.23
- Reference – Linux 2.6.10

Expectation

- Unsaid expectation:
  - A newer system version should not significantly degrade the performance.
- Learning the expectation:
  - Constructing the probability distribution on the performance change due to Linux version upgrade.
Reference-Driven Performance Expectation

- Help identify an anomaly by observing an "unexpected" performance degradation from a reference.
- Single-parameter relative model:
  - provide expected performance difference between two executions that only differ on one condition parameter.
  - preceding example - a relative model on OS version.
- A complex system has many exec. condition parameters:
  - configurable parameters and workload properties

Case Study: Linux I/O Subsystem

- System configurations:
  - File system prefetching depth
  - I/O scheduler: noop, deadline, anticipatory
- Workload parameters:
  - I/O concurrency
  - Access pattern: e.g., sequential/random
  - inter-I/O compute time

Anomaly Cause #1

- In Linux 2.6.10 and earlier:
  - anomaly: high-concurrency I/O sometimes delivers extremely low performance
  - disk is marked as "congested" when the device queue has 113 requests or more
  - when the disk queue is "congested", prefetching is cancelled
  - however, prefetching sometimes include synchronously requested data, which is resubmitted as single-page I/O

- In Linux 2.6.11 and above:
  - only cancel asynchronous prefetching when disk is "congested"
  - anomaly: medium-concurrency I/O sometimes delivers much worse performance than high-concurrency I/O

Anomaly Cause #2

- Anticipatory I/O scheduler

/* max time we may wait to anticipate a read (default around 6ms) */
define default_antic_expire ((HZ / 150) ? HZ / 150 : 1)

/* Kai Shen: the above is problematic given that HZ defaults to 250 for 2.6.23 kernel */
Anomaly Cause #3

- Anticipatory I/O scheduler
  - Sometimes anticipatory scheduler allows multiple outstanding device-level I/O requests
    - splitting of large I/O requests
    - asynchronous prefetching and synchronous I/O from one process
  - anticipation timer is started after the first disk request returns
    - incorrect because there are still outstanding requests