Graduate Operating Systems

Fall 2019

Paper “SEDA”

• “Slashdot effect”; peak load
• “Well-conditioned service”
  – Throughput: saturate with load
  – Response time: increase linearly with load
  – Graceful degradation
Thread-Based Concurrency

- Easy to program; high concurrency
- Overheads
- Throughput degradation (bounded thread pools)
- Latency

![Thread-Based Concurrency Diagram]

Event-Driven Concurrency

- Small number of threads (typically one per CPU); non-blocking I/O
- Robust to load
- Latency
- Scheduling decisions; load dropping

![Event-Driven Concurrency Diagram]
SEDA: Staged Event Driven Architecture

Resource Controllers

- Thread pool controller
  - Adjust number of threads executing
Resource Controllers

- **Batching controller**
  - Adjust number of events processed by each iteration of the event handler

Queues

- **Queues are finite**
  - Enqueuing may fail
  - Block on full queue -> backpressure
  - Drop rejected events -> load shedding

- **Queues introduce explicit execution boundaries**
  - Threads may only execute within a single stage
  - Performance isolation, modularity, independent load management

- **Explicit event delivery support inspection**
  - Trace flow of events through application
  - Monitor queue lengths to detect bottleneck
Asynchronous I/O

**Summary & Discussion**

- **SEDA: Staged, Event-Driven Architecture**
  - Applications consist of connected stages each serviced by one or more threads
  - Dynamic resource controllers examine and react to high load conditions and control thread usage

- Measurement and control vs. reservation
  - Mechanisms for detecting overload
  - Policies to deal with overload

- **SEDA ease of programming**
  - Reduced need for synchronization & race conditions
  - Separate stages for different components of application/server