

Dynamic Resource Throttling for Well-Conditioned Internet Servers

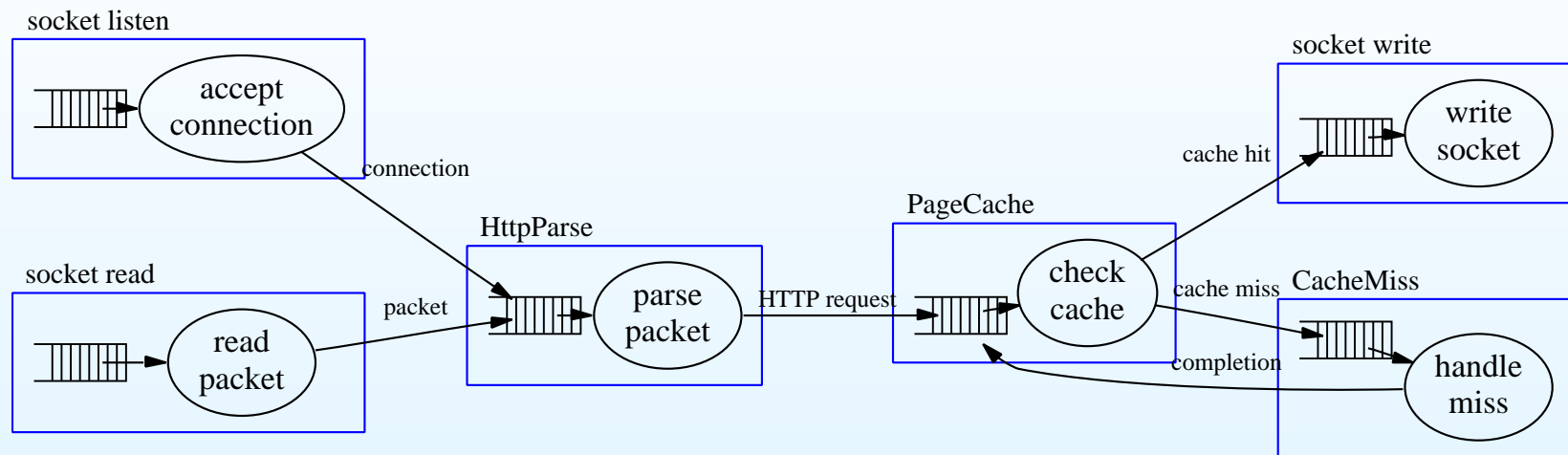
Matt Welsh and David Culler

UC Berkeley Computer Science Division

mdw@cs.berkeley.edu

USITS'01, San Francisco, March 27, 2001

SEDA: The Staged Event-Driven Architecture



Supports massive concurrency

- ▷ *Stages are event-driven and use nonblocking I/O*

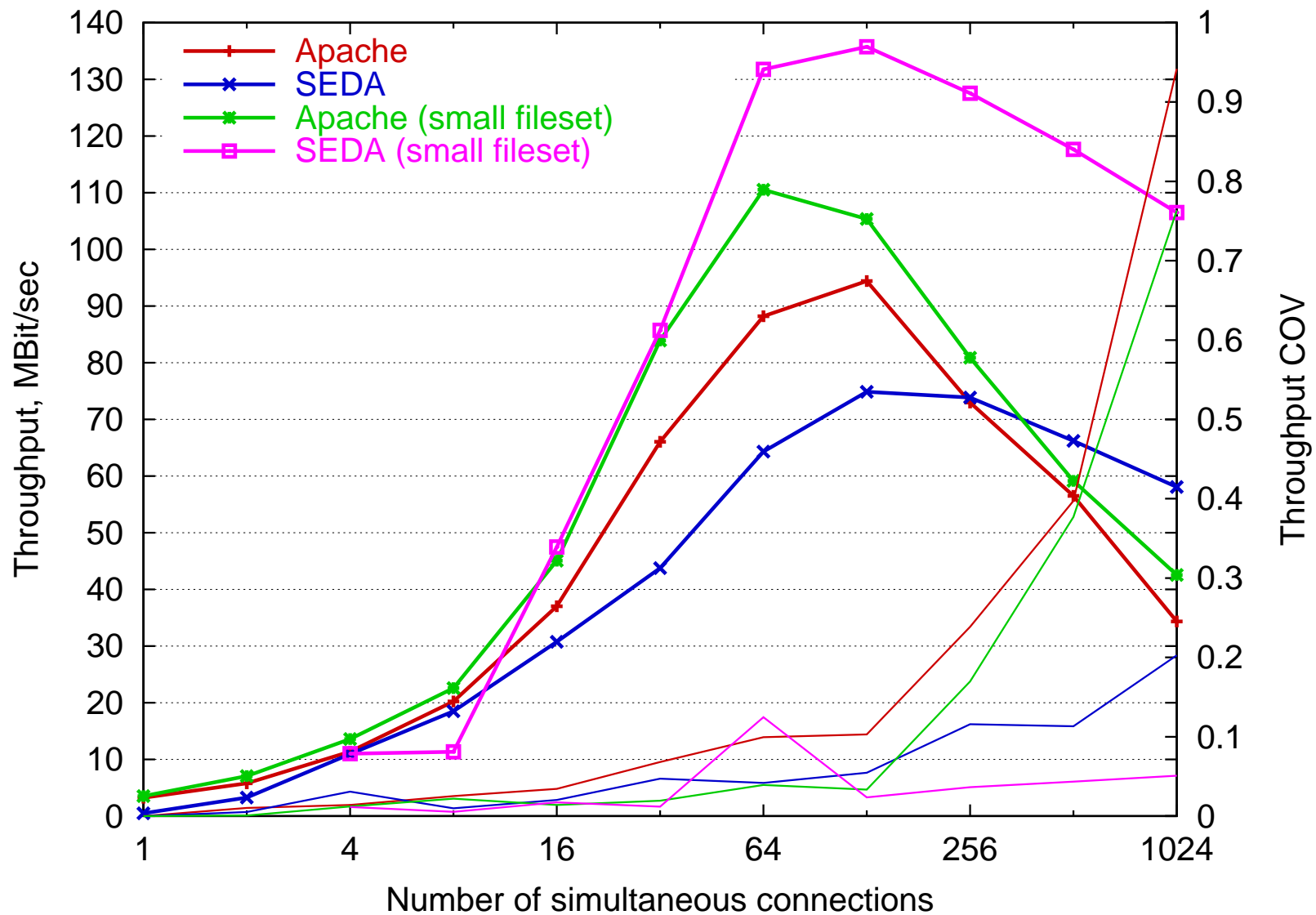
Enables load conditioning

- ▷ *Exposure of queues facilitates inspection of request stream*

Simplifies service construction

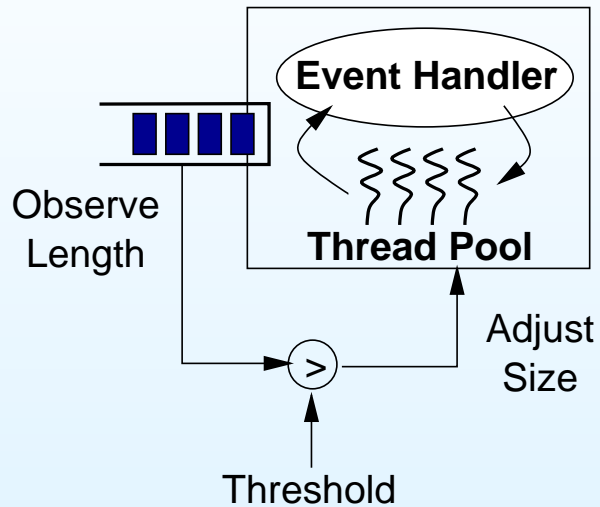
- ▷ *Stages embody robust, reusable service components*

Robust Performance under Heavy Load



SEDA Stage Controllers

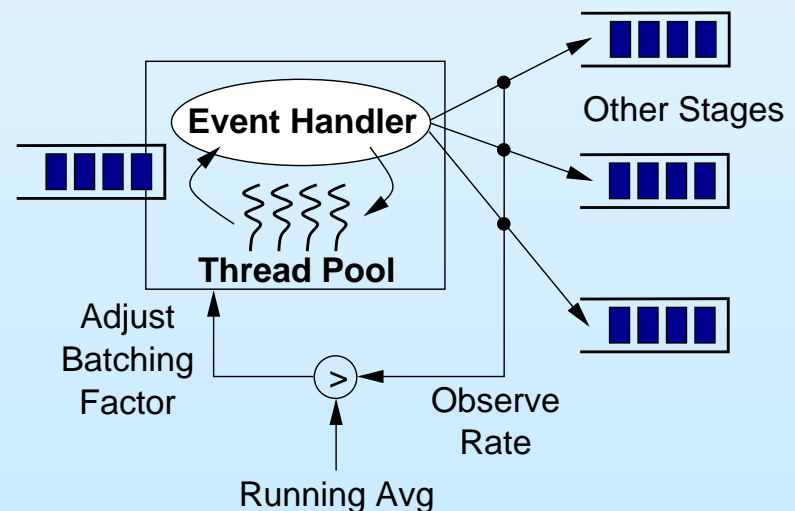
Thread pool controller



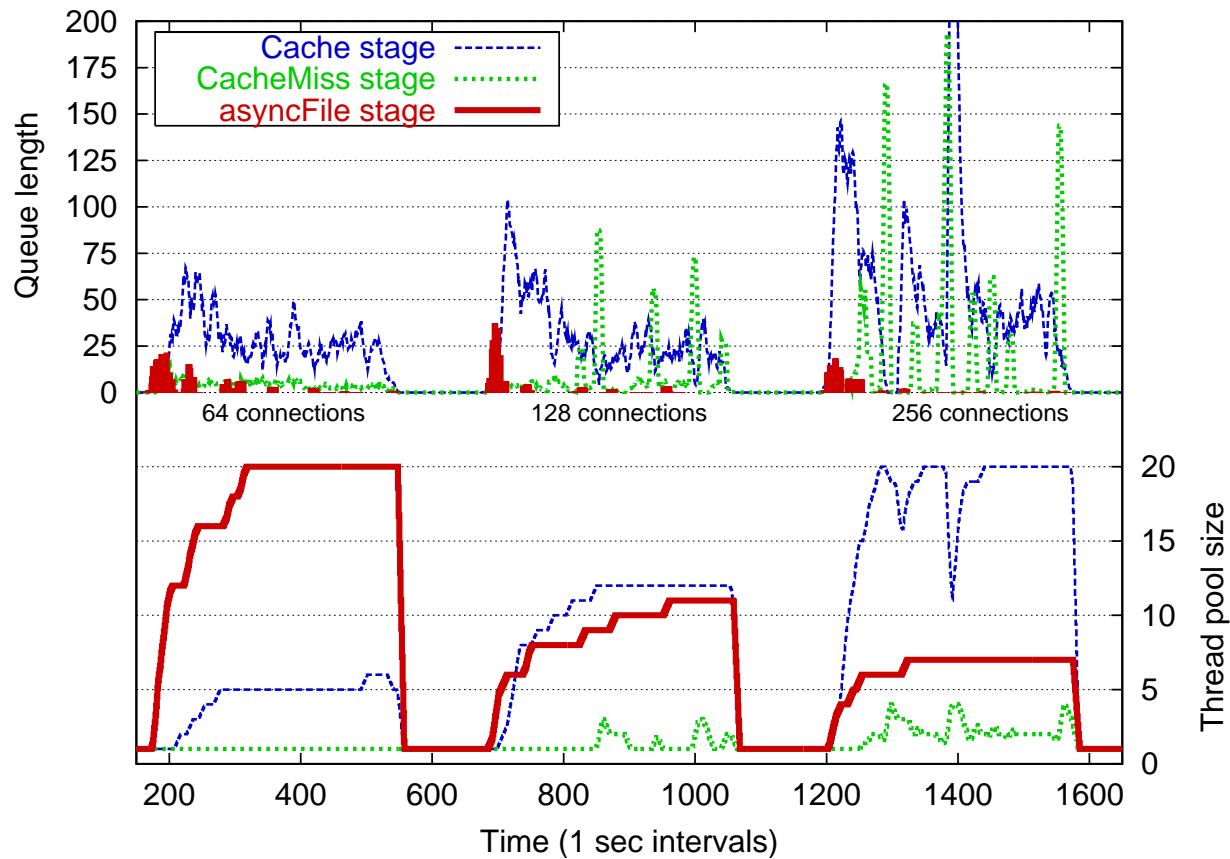
- Adjusts number of threads allocated to each stage
- Observes input queue length
- Adds threads if over threshold
- Idle threads removed from pool

Batching controller

- Adjusts number of events processed by event handler
- Observes output event rate
- Attempts to find smallest batching factor with stable throughput

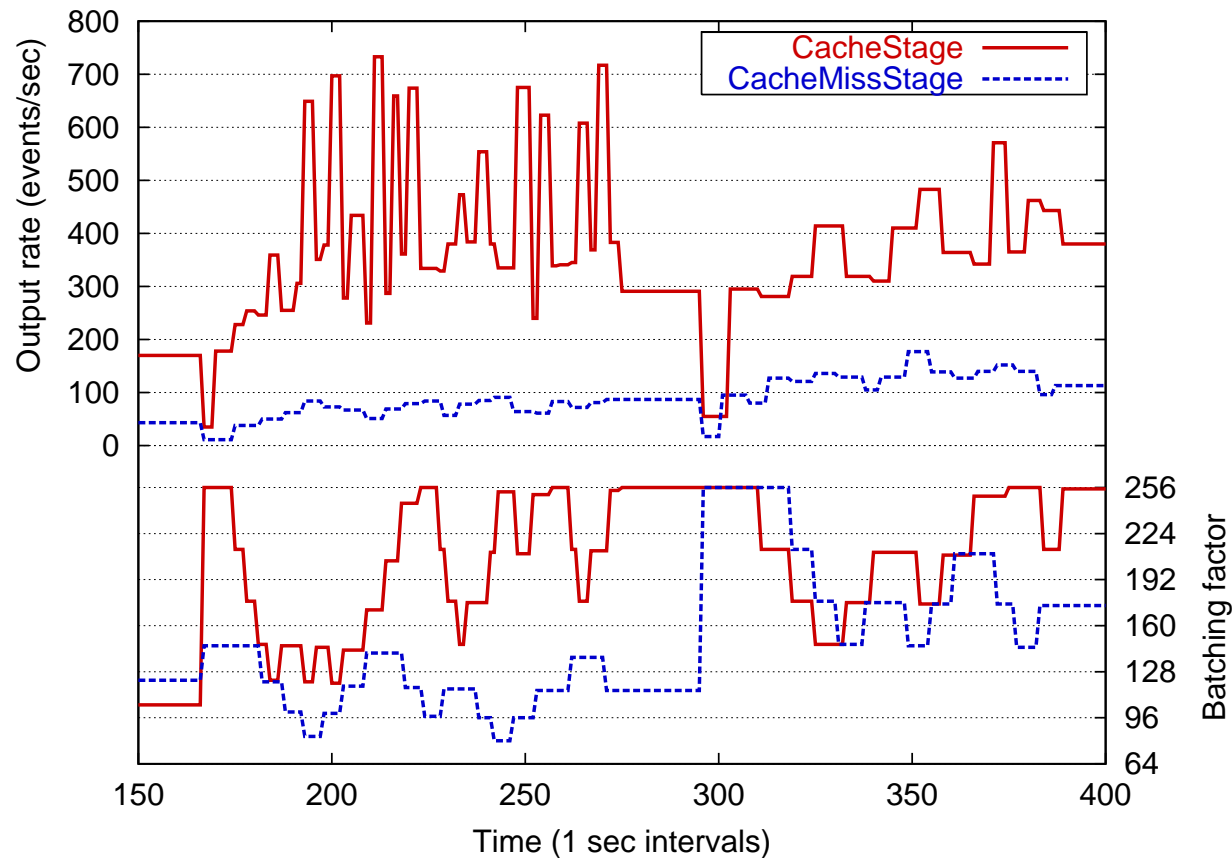


Thread Pool Controller in Action



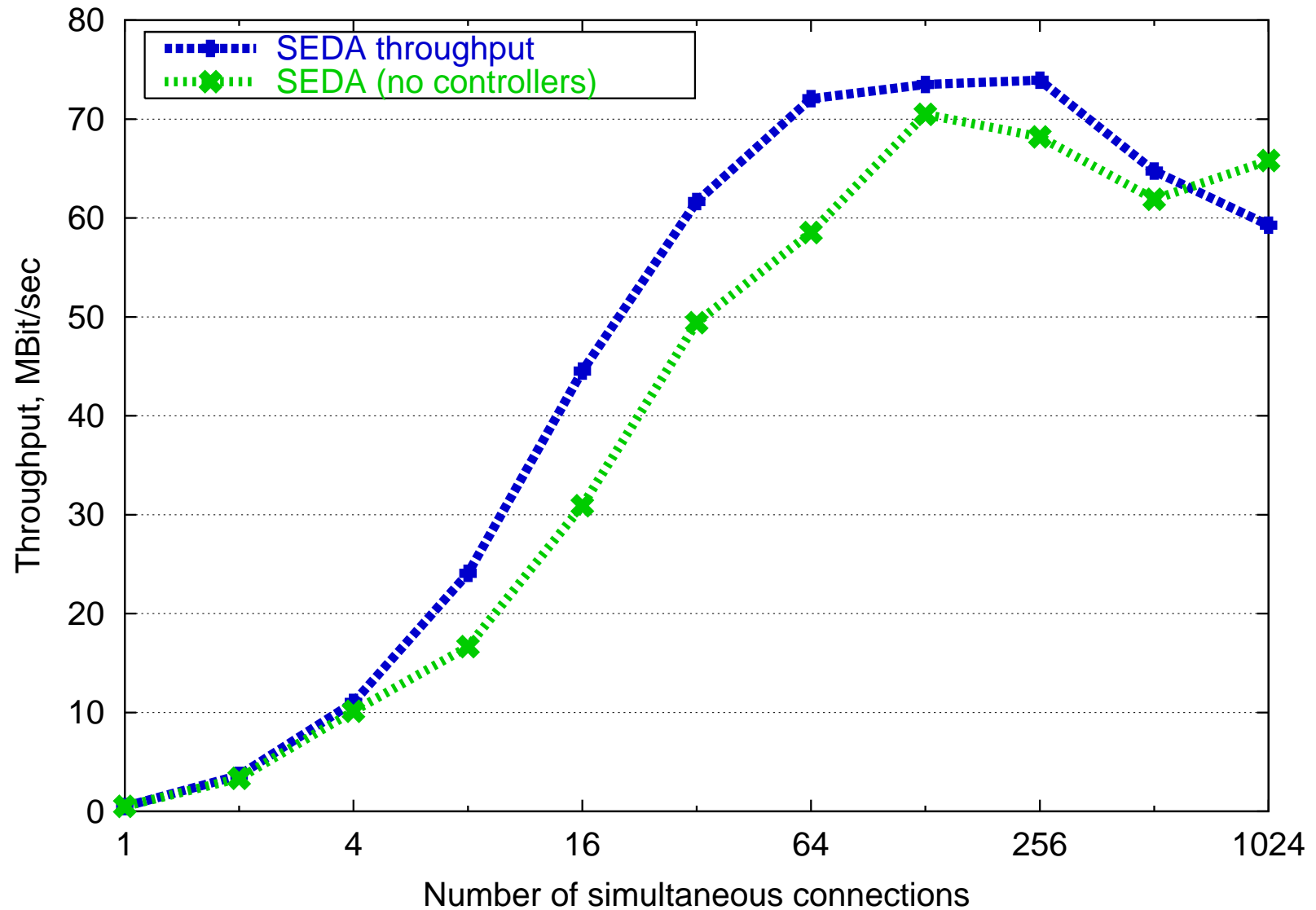
- Fewer threads needed for file I/O over time
 - ▷ *Filesystem buffer cache warming up*

Batching Controller in Action

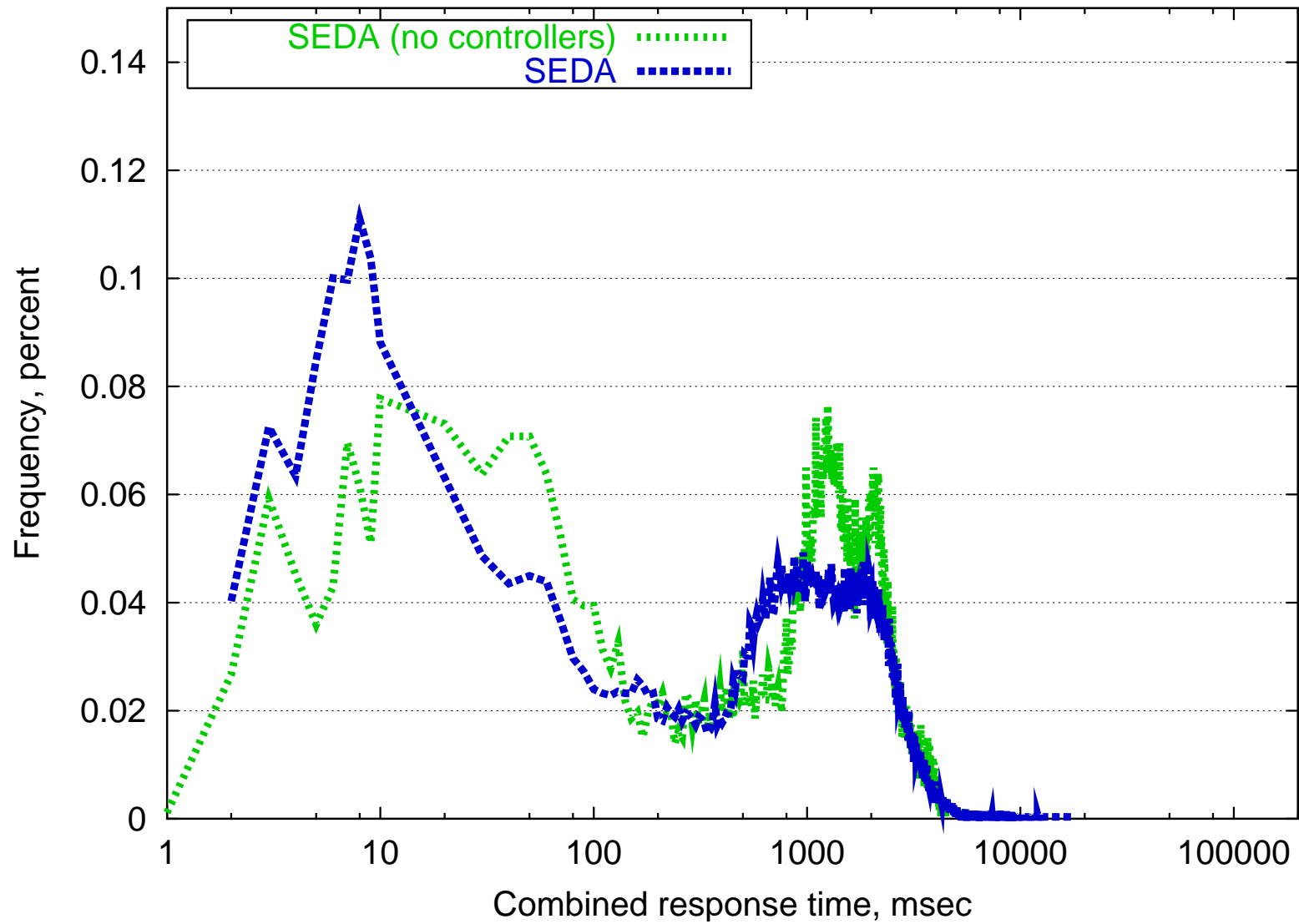


- At light load, maximum batching → high throughput, low inherent response time
 - *Respond to sudden drop in load by resetting batching factor to max*

Throughput benefit of controllers



Response time benefit of controllers



New Way of Thinking about Software

Support for massive concurrency requires new design techniques

- SEDA introduces service design as a *network of stages*
- Design for robustness and adaptivity, rather than best case
- Expose request streams to applications for load conditioning

Resource throttling to keep stages within operating regime

- Adapt behavior at runtime to deal with changing load
- Controllers shield service developers from much of this complexity

Implications for OS and language design

- SEDA model opens up new questions in service design space
- Bring body of work on control systems to bear on service design
- Many interesting controller algorithms possible

For more information:

<http://www.cs.berkeley.edu/~mdw/proj/seda/>