SEDA – Staged Event-Driven Architecture

SEDA: An Architecture for well-conditioned scalable internet services. Matt Welsh, David Culler & Eric Brewer

Presented by Rahul Agarwal

Overview

- Motivation
- Key Points
- Thread vs. Event concurrency
- SEDA
- Experimental Evaluation
- *My* Evaluation
- Questions to Consider

Authors

- Main Author
  - Matt Welsh's PhD thesis at UC Berkeley
  - Now Assistant Professor in CS Dept at Harvard
  - Currently working on wireless sensor networks
  - Research interests in OS, Networks and Large Scale Distributed Systems
- Culler & Brewer - Advisors

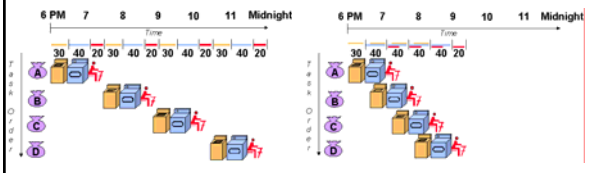
Motivation

- High demand for web content – for example concurrent millions of users for Yahoo
- Load Spikes – "Slashdot Effect"
  - Over provisioning support not feasible financially and events unannounced
- Web services getting more complex, requiring maximum server performance

Motivation – Well-conditioned Service

- A well-conditioned service behave like a simple pipeline
- As offered load increase throughput increases proportionally
- When saturated it **does not** degrade throughput – *graceful degradation*

Pipelining



- How can we improve throughput in a pipeline? Potential problems?

Key Points

- Application as a network of event-driven **stages** connected by **queues**
  - Dynamic Resource Controllers
  - Automatic load tuning
    - Thread pool sizing
    - Event Batching
    - Adaptive load shedding
- Support massive concurrency and prevent resources from being overcommitted

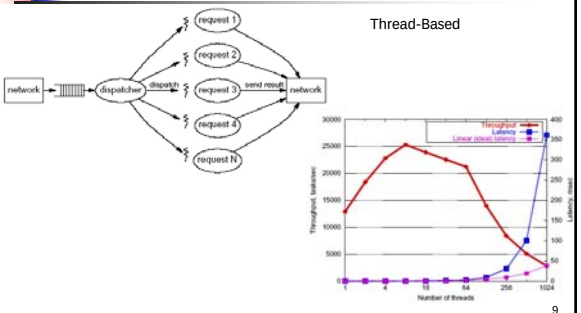
7

Thread-Based vs. Event-Based Concurrency

- Thread
  - Thread-per-request
  - OS switches and overlaps computation and I/O
  - Synchronization required
  - Performance Degradation
    - Eg: RPC, RMI, DCOM
  - Overcome by more control to OS
    - Eg: SPIN, Exokernel, Nemesis
  - Overcome by reuse of threads
    - Eg: Apache, IIS, Netscape... everyone!
- Event
  - One thread per CPU (controller)
  - Process events generated by apps and kernel
  - Each task implemented as a FSM with transitions triggered by events
  - Pipelining!
    - Eg: Flash, Zeus, JAWS
  - Harder to modularize and engineer

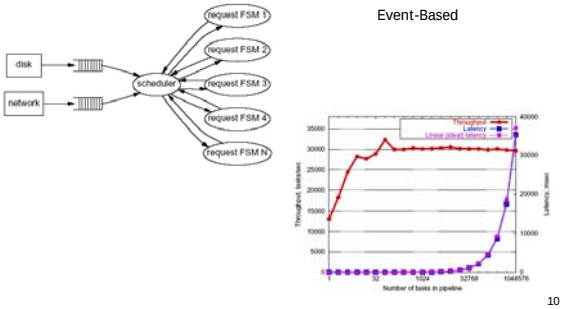
8

Thread-Based vs. Event-Based Concurrency (Contd.)



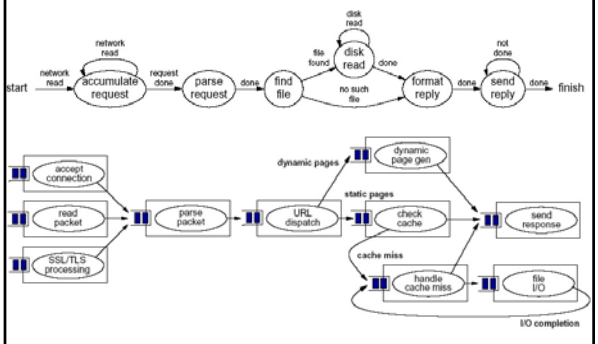
9

Thread-Based vs. Event-Based Concurrency (Contd.)



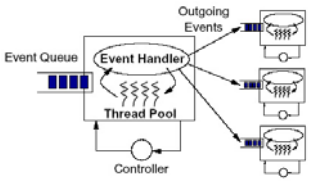
10

SEDA



11

SEDA - Stage



- Each stage has its own thread pool
- Controller adjusts resources
  - Controller may set "admission control policy"
    - Threshold, rate-control, load shedding
  - Adjust thread pool size
  - Adjust number of events processed

12

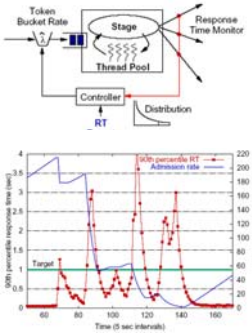
Overload Management

- Resource Containment
  - Static method
  - Usually set by admin
- Admission Control
  - Parameter based
  - Static or dynamically adjusted
- Control-theoretic approach
- Service Degradation
  - Deliver lower fidelity Service

13

SEDA Admission Control Policy

- Using 90th percentile of set response time as performance metric
- Adaptive admission control
  - Uses user classes (IPs, HTTP header info)
  - Uses token bucket traffic shaper
  - Possible to use drop-tail, FIFO, RED or other algorithms



14

Event-driven programming

- Benefits ☺
  - Applications can map cleanly into modules
  - Each stage self-contained
  - Typically little/no data sharing
- Challenges
  - Determining stages
  - Stages can block
  - Managing continuations between stages
  - Tracking events

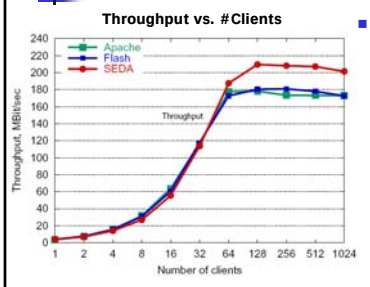
15

Software Contribution

- Sandstorm: SEDA Framework
- NBIO: Non-blocking I/O implementation
- Haboob: Implementation of a web-server
- aTLS: library for SSL and TLS support
  - All Java implementations, NBIO uses JNI
  - Last updated July 2002

16

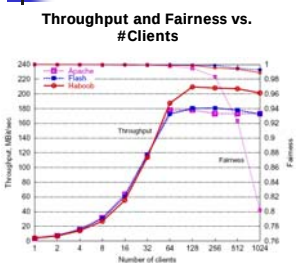
Experimental Evaluation



- Haboob web-server
  - Static and dynamic file load – SpecWEB99 benchmark
  - 1 to 1024 clients
  - Total files size 3.31Gb
  - Memory Cache of 200Mb

17

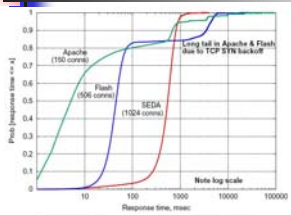
Evaluation (Contd.)



- Jain's Fairness Index
  - Equality of services to all clients
  - Suppose server can support 100 requests
  - Totally fair if it processes 10 requests of each
  - Unfair if say it processes 20 requests each for 5 users

18

Evaluation (Contd.)

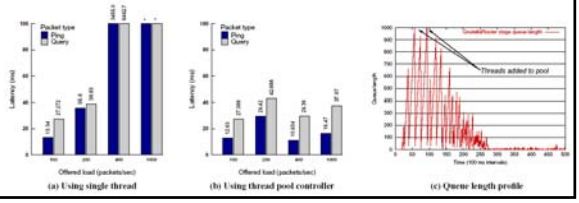


Cumulative response time distribution for 1024 clients

	SEDA	Flash	Apache
Mean RT	547 ms	665 ms	475 ms
Max RT	3.8 sec	37 sec	1.7 minutes

Evaluation (Contd.)

- Gnutella packet router
  - Non-traditional internet service, routing P2P packets
  - Ping: Discover other nodes
  - Query: Search for files being served



Summary

- Notion of stages
- Explicit event queues
- Dynamic resource controllers
- + Improved performance
- + Well-conditioned services
- + Scalability

My Evaluation

- Increased performance at higher loads which was the motivation
- Marginal increase in throughput but significantly more fair for higher loads
- Throughput does not degrade when resources exhausted
- Response times of other servers better at lower loads

My Evaluation

- In context of duality of OS structures (Lauer & Needham)
  - SEDA is message-oriented
  - Message and Procedure oriented can be directly mapped, however independent of the application under consideration can performance indeed be similar?
  - We will see how Capriccio does this – but no simple mapping!

Questions to consider

- SEDA is so good but the whole world (Apache, IIS, BEA, IBM, Netscape…) still uses thread-based servers?
- In real world scenarios how often are there load spikes, should goal be to increase average case performance instead?
- Is throughput or fairness a better metric?
- Being faster “despite” being in Java a bias or poor choice of language?

References

- Welsh, M., Culler, D., Brewer, E. (2001). SEDA: An architecture for well-conditioned, scalable internet services. *Proceeding of 18th SOSP,* Banff, Canada, October 2001
- SEDA Project http://sourceforge.net/projects/seda
- Welsh, M. (2002). *An architecture for well-conditioned, scalable internet services.* PhD Thesis, University of California, Berkeley
- Welsh, M. (2001) SEDA: An architecture for well-conditioned, scalable internet services. Presentation at 18th SOSP, Lake Louise, Canada, October 24, 2001
- Welsh, M., Culler, D., Adaptive overload control for busy internet servers. Unpublished.
- Pipelining: http://cse.stanford.edu/class/sophomore-college/projects-00/risc/pipelining/
- Lauer, H. C. & Needham, R. M. (1978). On the duality of operating systems structures. *In proceedings 2nd International Symposium on Operating Systems*, IRIA, October 1978, reprinted in *Operating Systems Review, 13*, 2 April 1979, pp 3-19.
- Behren, R. et al. (2003). Capricco: Scalable Threads for Internet Services. *Proc SOSP*, New York, October 2003

25