CS240B Homework Two Paper Critique

Michael Lee

February 16, 2003

1 Introduction

This is a short critique of the paper *SEDA: An Architecture for Well-Conditioned, Scalable Internet Services* by Matt Welsh et. al., from UC Berkeley. The paper describes different paradigms for supporting concurrency in server software, presents the SEDA architecture, and discusses implementations of servers that use SEDA.

2 Concurrency Paradigms

2.1 Process-based

Process-based servers are a simple way of supporting concurrency. They merely require a platform with system calls that can support (instead of special libraries). Switching between processes is very expensive since each has its own context (including virtual address space, file handles, network connections, etc).

2.2 Thread-based

Thread-based servers use multiple threads within a single process. Since threads share one context, switching between threads is much cheaper. This solution, like the process-based solution, is generally easy to program, but the SEDA paper shows that threads still have considerable overhead in terms of scheduling and lock contention. In addition, threaded applications let the Operating System handle resource allocation issues, which is suboptimal for situations when efficiency is important.

2.3 Event-Driven

Event-driven systems attempt to provide a solution to the scalability issues inherent in process- and thread-based systems. Such systems control scheduling and event handling themselves and so they aren't as susceptible to bad scheduling decisions made by the OS. They have been shown to scale better than other solutions, but are difficult to program because many decisions that would otherwise be handled by the operating system must be dealt with by the programmer. They also depend on support for non-blocking I/O routines.

2.4 SEDA

SEDA builds upon existing Event-Driven theory to provide a framework of stages, each of which is a thread-based unit. One clear feature of this system is that event-driven systems are now easier to develop, since SEDA provides a foundation to build upon, and has features to make debugging and performance analysis easier. Thusfar it appears to only have been implemented in Java, which potentially reduces the applications it can be used for. It also requires support for non-blocking I/O routines (the current implementation uses a bounded thread pool to simulate non-blocking file I/O).

3 Other Features

One important feature of SEDA is the queues which allow stages to communicate and allow the application to interface with the external environment. By monitoring the queues, the Gnutella packet router was able to drop saturated client connections. In general, queue status can enable the application to more effectively adapt its operation to different load levels, by dropping connections, giving error messages, or providing stable but degraded service.

4 Paper Analysis

The paper presents two examples of applications developed using SEDA: the Haboob web server and a Gnutella packet router. It provides concrete benchmark results in situations that closely mimic real-life scenarios and shows that the SEDA-based solutions outperform other software.

5 Conclusion

SEDA represents a novel approach to managing concurrency in applications where scalability and efficiency are of utmost importance. As the system matures, perhaps it will define a movement away from traditional thread-based systems.