SEDA: An Architecture for Highly Concurrent Server Applications

Matt Welsh, Harvard University
Last updated 9 May 2006

Introduction

My Ph.D. thesis work at UC Berkeley focused on the development of a robust, high-performance platform for Internet services, called **SEDA**. The goal is to build a system capable of supporting massive concurrency (on the order of tens of thousands of simultaneous client connections) and avoid the pitfalls which arise with traditional thread and event-based approaches.

SEDA is an acronym for *staged event-driven architecture*, and decomposes a complex, event-driven application into a set of *stages* connected by *queues*. This design avoids the high overhead associated with thread-based concurrency models, and decouples event and thread scheduling from application logic. By performing admission control on each event queue, the service can be well-conditioned to load, preventing resources from being overcommitted when demand exceeds service capacity. SEDA employs dynamic control to automatically tune runtime parameters (such as the scheduling parameters of each stage), as well as to manage load, for example, by performing adaptive load shedding. Decomposing services into a set of stages also enables modularity and code reuse, as well as the development of debugging tools for complex event-driven applications.

February 19, 2007 - A Note on the status of SEDA

I continue to receive many requests for information about SEDA. I am no longer actively working on this project, so all of these web pages should be regarded as "archival".

It is also worth noting that a number of recent research papers have demonstrated that the SEDA prototype (in Java) performs poorly compared to threaded or event-based systems implemented in C. This would seem to contradict the findings in my work. (For more information I invite you to read recent papers by Vivek Pai's group at Princeton and the Capriccio work from UC Berkeley.)

While I do not discount these later results, it is important to keep a few things in mind when interpreting them. First, the SEDA implementation in Java was developed and tuned on a particular JVM implementation (IBM JDK 1.3), on a particular version of the Linux kernel (2.2), using the `/dev/poll` event dispatch mechanism. More recent studies have varied the environment substantially.

I have several theories about what could be causing this poor performance, although I have not had an opportunity to perform new measurements. I have noticed that performance of the SEDA networking layer is highly dependent on a number of parameters, such as the poll interval used by the various threads. This likely needs to be tuned or redesigned to support high bandwidth networks and more recent Linux and JVM implementations. Also, SEDA imposes a high context switch overhead in certain cases, depending on the number of threads and stages used, and the processing granularity within each stage.

Tim Brecht's group at Waterloo has undertaken a study of competing Web server architectures and has shown that a SEDA implementation in C++, appropriately tuned, performs comparably to alternatives, so I do not believe these performance issues are fundamental to the architecture.

The most *fundamental* aspect of the SEDA architecture is the programming model that supports stage-level backpressure and load management. Our goal was never to show that SEDA outperforms other server designs, but rather that acceptable performance can be achieved while providing a disciplined approach to overload management. Clearly more work is needed to show that this goal can be met in a range of application and server environments.

Please feel free to get in touch if you have new results or questions about the SEDA approach.

Our current prototype of a SEDA-based services platform is called **Sandstorm**. Sandstorm is implemented entirely in Java and uses the NBIO package to provide nonblocking I/O support. Support for the JDK 1.4 `java.nio` package is included as well. Despite using Java, we have achieved performance that rivals (and sometimes exceeds) that of C/C++. We have also implemented a SEDA-based asynchronous SSL and TLS protocol library, called **aTLS**. All of this software is available for download below.

We have built a number of applications to demonstrate the SEDA framework. **Haboob** is a a high-performance Web server including support for both static and dynamic pages. Other applications include a **Gnutella packet router** and **Arashi**, a Web-based email service similar to Yahoo! Mail.

The best place to start for more information is the SOSP'01 paper on SEDA and the corresponding talk slides. My Ph.D. thesis has much more information as well. If you have questions, comments, or are interested in collaborations, please feel free to contact me by e-mail (see my home page).

A number of open source and commercial systems are based on SEDA and NBIO. These include:

- LimeWire runs runs its server based Web crawler on NBIO.
- TerraLycos runs its chat servers on NBIO, supporting over 30,000 simultaneous users
- Rimfaxe Web Server
- Apache Excalibur Event Package
- SwiftMQ, a JMS Enterprise Messaging Server
- MULE Universal Message Objects, a distributed object broker
- OceanStore, a global, secure, peer-to-peer filesystem
- Various companies, both large and small, are building projects based on SEDA/NBIO.

Project News

July 12, 2002: Lots of updates. CVS, release, and mailing list hosting is now at http://seda.sourceforge.net. Now you can access the latest SEDA codebase via anonymous CVS, hopefully encouraging more collaborative development of the code.

The `seda-users` mailing list is back up - please subscribe.

All of the code has been consolidated into a single CVS tree under the package name **seda** (renamed from **mdw**). The Haboob Web server and aTLS code are also released and more completely documented. And a nice one-line performance patch to NBIO is included that increases network bandwidth by 30% or so!

Papers

- **Adaptive Overload Control for Busy Internet Servers**, Matt Welsh and David Culler. To appear in *Proceedings of the 4th USENIX Conference on Internet Technologies and Systems (USITS'03)*, March 2003. (PDF)
- ***An Architecture for Highly Concurrent, Well-Conditioned Internet Services***, Matt Welsh. Ph.D. Thesis, University of California, Berkeley, August 2002. (PDF)
- **Overload Management as a Fundamental Service Design Primitive**, Matt Welsh and David Culler. To appear in *Proceedings of the Tenth ACM SIGOPS European Workshop*, Saint-Emilion, France, September, 2002. (PDF)
- **SEDA: An Architecture for Well-Conditioned, Scalable Internet Services**, Matt Welsh, David Culler, and Eric Brewer. In *Proceedings of the Eighteenth Symposium on Operating Systems Principles (SOSP-18)*, Banff, Canada, October, 2001. (PDF)
- **Virtualization Considered Harmful: OS Design Directions for Well-Conditioned Services**, Matt Welsh and David Culler. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS VIII)*, Schloss Elmau, Germany, May, 2001. (PDF)
- **The Staged Event-Driven Architecture for Highly Concurrent Server Applications**, Matt Welsh. *Ph.D. Qualifying Examination Proposal*, November, 2000. (PDF)
- **A Design Framework for Highly Concurrent Systems**, Matt Welsh, Steven D. Gribble, Eric A. Brewer, and David Culler. UC Berkeley Technical Report UCB/CSD-00-1108, *Submitted for publication*, April, 2000. (PDF)

Talks

- Building Dependable Internet Services. Presented at the Tenth SIGOPS European Workshop, Saint-Emilion, France, September 22, 2002.
- SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. Presented at the Eighteenth Symposium on Operating Systems Principles (SOSP'01), Lake Louise, Canada, October 24, 2001.
- Containment vs. Control: Keeping Busy Internet Servers Well-Behaved. Work-in-progress presentation at the USENIX 2001 Annual Technical Conference, Boston, June 29, 2001.
- Virtualization Considered Harmful: OS Design Directions for Well-Conditioned Services. Presented at HotOS-VIII, Schloss Elmau, Germany, May 23, 2001.
- SEDA: Enabling Robust Performance for Busy Internet Servers. Presented at the Internet and Distributed Systems Seminar, Stanford University, April 18, 2001. Also presented at the Systems Seminar, University of Washington, April 27, 2001.
- Dynamic Resource Throttling for Well-Conditioned Internet Services. Work-in-Progress presentation at the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01), San Francisco, CA, March 27, 2001.
- The Staged Event-Driven Architecture for Highly Concurrent Servers. Ph.D. qualifying examination, UC Berkeley, December 13, 2000.
- Performance Aspects of The Staged Event-Driven Architecture. UC Berkeley Ninja Project Retreat, Lake Tahoe, CA, January 10, 2001.
- Designing Systems for High Concurrency. Presented at BEA WebLogic, San Francisco, CA, July 7, 2000. Also presented at the UC Berkeley Endeavour Retreat, Lake Tahoe, CA, June 19, 2000.
- Building Efficient, Scalable Systems in Java. Presented at IBM T.J. Watson Research Center, Hawthorne, NY, May 30, 2000.

Software Downloads

File downloads are hosted by SourceForge.net. Click here for the SEDA SourceForge Page. You may either download the SEDA software as a set of pre-packaged "official" releases (.tar.gz format, source code included), or use anonymous CVS to access the "live" tree. The CVS tree will be updated more frequently than the "official" releases, which are meant to represent stable, tested versions of the software. The CVS tree is the "live code" that is under constant development.

See the file `README` in each release for information on compilation and usage. All of the SEDA code is covered under an open-source license (see below).

Official releases

All files are available from this SourceForge page. Or, you may click on one of the links below:

Package	Latest version	Download
Latest SEDA "core" release: Includes the NBIO library and the Sandstorm runtime environment.	v3.0, July 12, 2002	seda-release-20020712.tar.gz
Latest NBIO-only release: Includes only the NBIO library.	v2.0, July 12, 2002	nbio-release-20020711.tar.gz
Haboob: A high-performance Web server built using Sandstorm. Requires the SEDA "core" release.	July 12, 2002	haboob-release-20020712.tar.gz
aTLS: An asynchronous TLS and SSL protocol library for Sandstorm. Requires the SEDA "core" release.	July 12, 2002	atls-release-20020712.tar.gz

Anonymous CVS access

Anonymous CVS access is available for those users who want to maintain a "live" source tree. To check out the SEDA tree using anonymous CVS, use the following commands:

```
cvs -d:pserver:anonymous@cvs.seda.sourceforge.net:/cvsroot/seda login
        ( Just press enter when asked for a password )

cvs -z3 -d:pserver:anonymous@cvs.seda.sourceforge.net:/cvsroot/seda co seda
```

Javadoc API documentation

You can browse the Javadoc documentation for SEDA.

Call for Developers

By transitioning the SEDA project to SourceForge.net, it is now possible to open up the development of the SEDA code a wider community. If you are an active user of the SEDA or NBIO code and would like to contribute, please join the seda-users mailing list. SourceForge makes it possible for all of the developers to share a single CVS tree, make code releases, and so forth. I encourage interested developers to join the team!

Performance Results

The best place to look for performance information about SEDA is the various papers about the system. Earlier (and somewhat outdated) performance results are discussed in the following web pages:

- I/O Core Benchmarks
- HTTP Server Load Benchmarks

Copyright License

If you have any questions, comments, or bug reports, don't hesitate to get in touch with me!