Boost.Asio

A cross-platform C++ library for network and low-level I/O programming

Dustin Spicuzza
November 18, 2009

Today's Talk

- What is Boost.Asio?
- Technical Overview
- Why it doesn't suck
- Sometimes it sucks

What is Boost.Asio?

- C++ library for doing I/O
  - Sockets
  - Files
  - Serial Ports
  - Timers
- Provides asynchronous and synchronous operations

What is it used for?

- Primarily known as a networking library
- Various types of asynchronous I/O
- Highly concurrent clients/servers
- Event-driven programs

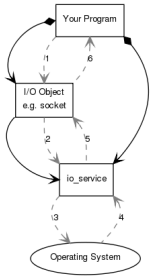
How I've used Asio

- SMITE
  - Event-driven data analysis
  - Reading from libpcap @ 200Mbps
  - Blocking and non-blocking network communications
- WebDMA (http://code.google.com/p/webdma/)
  - Used for instrumentation and tuning of our team's robot via a browser
  - Heavily optimized version of the sample HTTP server that comes with Asio
  - vxWorks on PPC, Linux, Windows XP

Asio Design Goals

- Portability
- Scalability
- Efficiency
- Model concepts from established APIs, such as BSD sockets
- Ease of use
- Basis for further abstraction

Traditional (synchronous) I/O in ASIO



- Create io_service
- Create I/O object
- Ask object to do operation
- Result is returned after blocking operation is completed

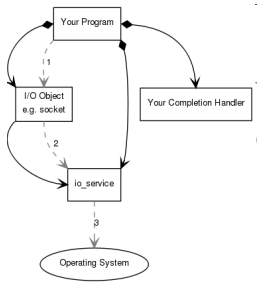
Async I/O Review

- Why asynchronous I/O?
  - Program can typically queue lots of async I/O operations at once
  - Instead of waiting for a blocking I/O operation to complete, can do other work
  - Particularly suited for event-driven programming
  - Typically doesn't require explicit threads or locking

Async I/O Review

- A generic async I/O access will typically look something like this
  - Setup some initial state (socket, etc)
  - Tell the OS to do some I/O
  - Do some other things
  - OS signals program that the I/O is complete

Async I/O in Asio



- Create io_service
- Create I/O object
- Ask object to initiate operation
- Do stuff
- Completion handler gets called when operation complete

Async I/O in Asio

- Consistent asynchronous interface across various types of I/O

```
{
   io_service io;
   some_io_object object(io, ...);

   async_op( object, buffer, boost::bind( &some_fn, .. ));
   io.run();
}

void some_fn( boost::error_code &ec)
{
   if (!ec)
      do_something();
}
```

Async I/O in Asio

- Free functions vs object methods
  - IO object methods not guaranteed to read/write number of bytes you request
  - Free functions will, or throw an error

```
some_io_object object(io, ...);

// free function
async_op( object, buffer, boost::bind( &some_fn, .. ));

// object method
object.async_op( buffer, boost::bind( &some_fn, .. ));
```

Error Handling

- Asynchronous operations call the completion handler with an error_code object
- Blocking operations support exceptions or returning error codes
  - Will throw unless you pass in an error code for it to set
- Exceptions bubble up to io_service.run()

Things Worth Noting

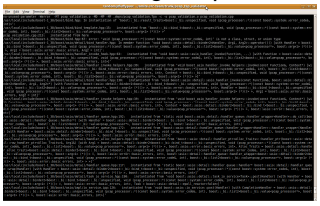
- Read/write operations support single buffer or scatter/gather
- Timing out I/O requires some extra work
- Asynchronous operations not dispatched when using a blocking call
- Iostream-style interface for sockets is also supported

Why it doesn't suck

- Cross-platform
- Uses optimal I/O features on a platform
- Peer reviewed & open source
- Documentation, tutorials, and *useful* example code available
- Submitted for inclusion in TR2

Sometimes, it sucks

- Slow compile times for large projects
  - Header-only implementation
- Library is inline using templates



Required Knowledge

- If you're going to use Boost.Asio, you'll probably want to be familiar with:
  - C++ & templates
  - boost::shared_ptr
  - boost::bind

Useful References

- Boost.Asio documentation
  - http://www.boost.org/doc/libs/1_40_0/doc/html/boost_asio.html
- Asio mailing lists
  - http://sourceforge.net/mail/?group_id=122478
- Asio home page
  - http://www.think-async.org/
- Thinking asynchronously in C++ blog
  - http://blog.think-async.org/

Questions?