Building High Performance Scalable TCP/IP Servers with Apache MINA

Originally presented at ApacheCon Europe 2006 in Dublin

Latest slides and code samples at http://people.apache.org/~proyal

Presented by Peter Royal, <proyal@apache.org>

Goals of this presentation

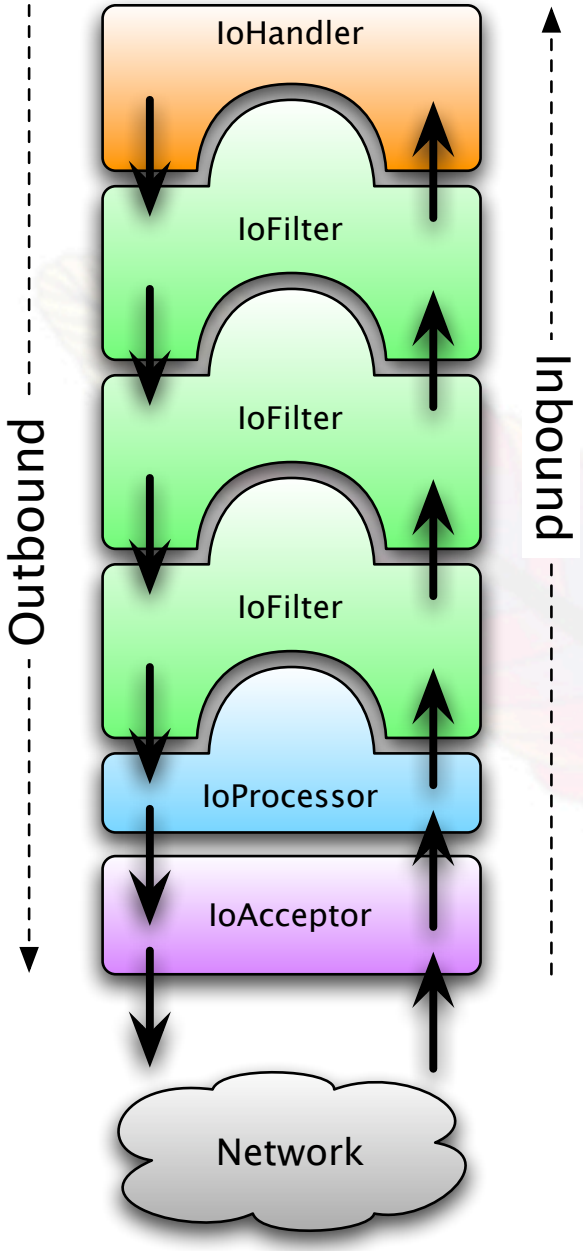
- Introduction to MINA
- Demonstration of what it can do
- Converting blocking-IO code to MINA
- Hopefully inspire you to use it :)

What is MINA

- **M**ultipurpose **I**nfrastructure for **N**etworked **A**pplications
- A framework (the F word!) for building networked clients and servers based on non-blocking IO
- http://directory.apache.org/subprojects/mina/

Brief history of MINA

- Started out as Netty2 from Trustin Lee
- Joined the Directory Project as the SEDA-based directory needed an asynchronous I/O layer.



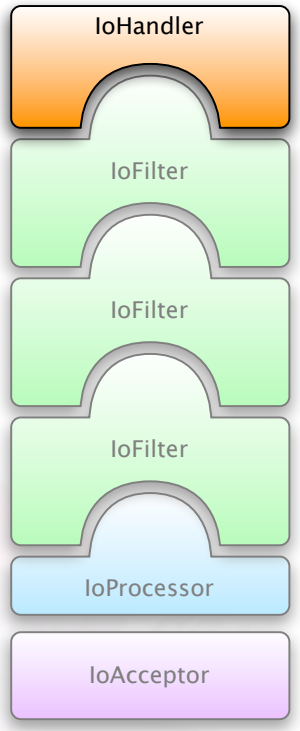
Architectural overview

IoSession

- Holder of state for a connection (either client-side or server-side)
- Passed along with every event
- Important Methods
  - write
  - close
  - get/setAttribute

IoHandler

- Akin to a Servlet
- Endpoint of a filter chain
- Important Methods
  - sessionOpened
  - messageReceived
  - sessionClosed

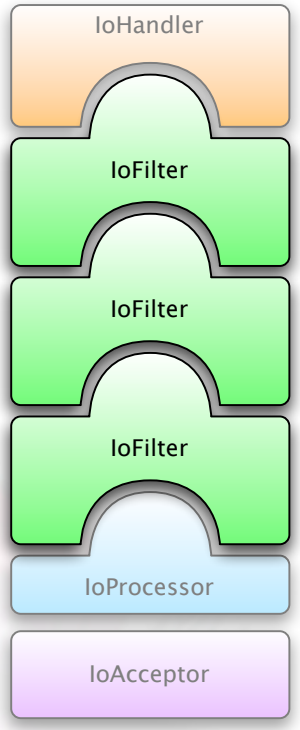


IoFilterChain

- Chain of IoFilter's for each IoSession
- Can setup template chains per IoConnector/IoAcceptor
- Dynamic addition/removal

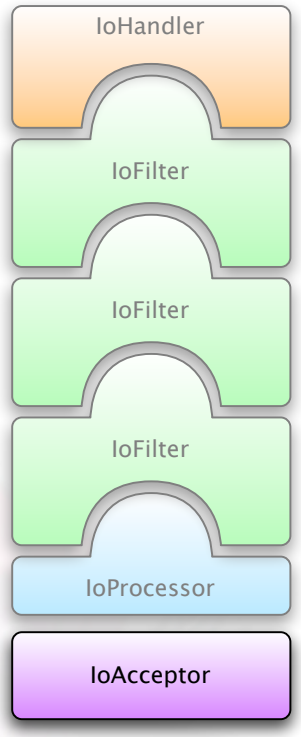
IoFilters

- Akin to a ServletFilter
- View/Hack/Slash the event stream
- Important Methods
  - sessionOpened
  - messageReceived
  - filterWrite
  - sessionClosed



IoAcceptor

- Server-side entry point.
- Accepts incoming connections and fires events to an IoHandler
- Important Methods
  - bind

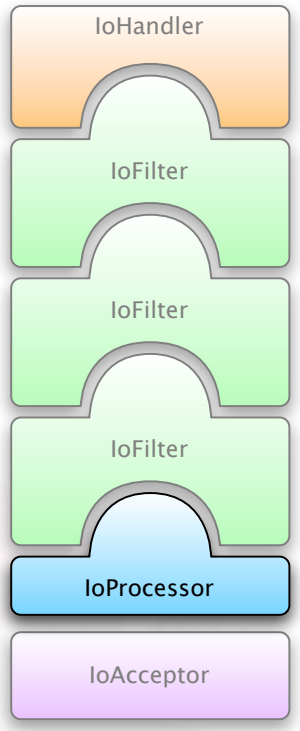


IoConnector

- Client-side entry point
- Initiate connections to a remote service, and fires events to an IoHandler
- Important Methods
  - connect

IoProcessor

- Internal component
- Handles reading and writing data to an underlying connection
- Each connection is associated with a single IoProcessor (shared amongst multiple connections)

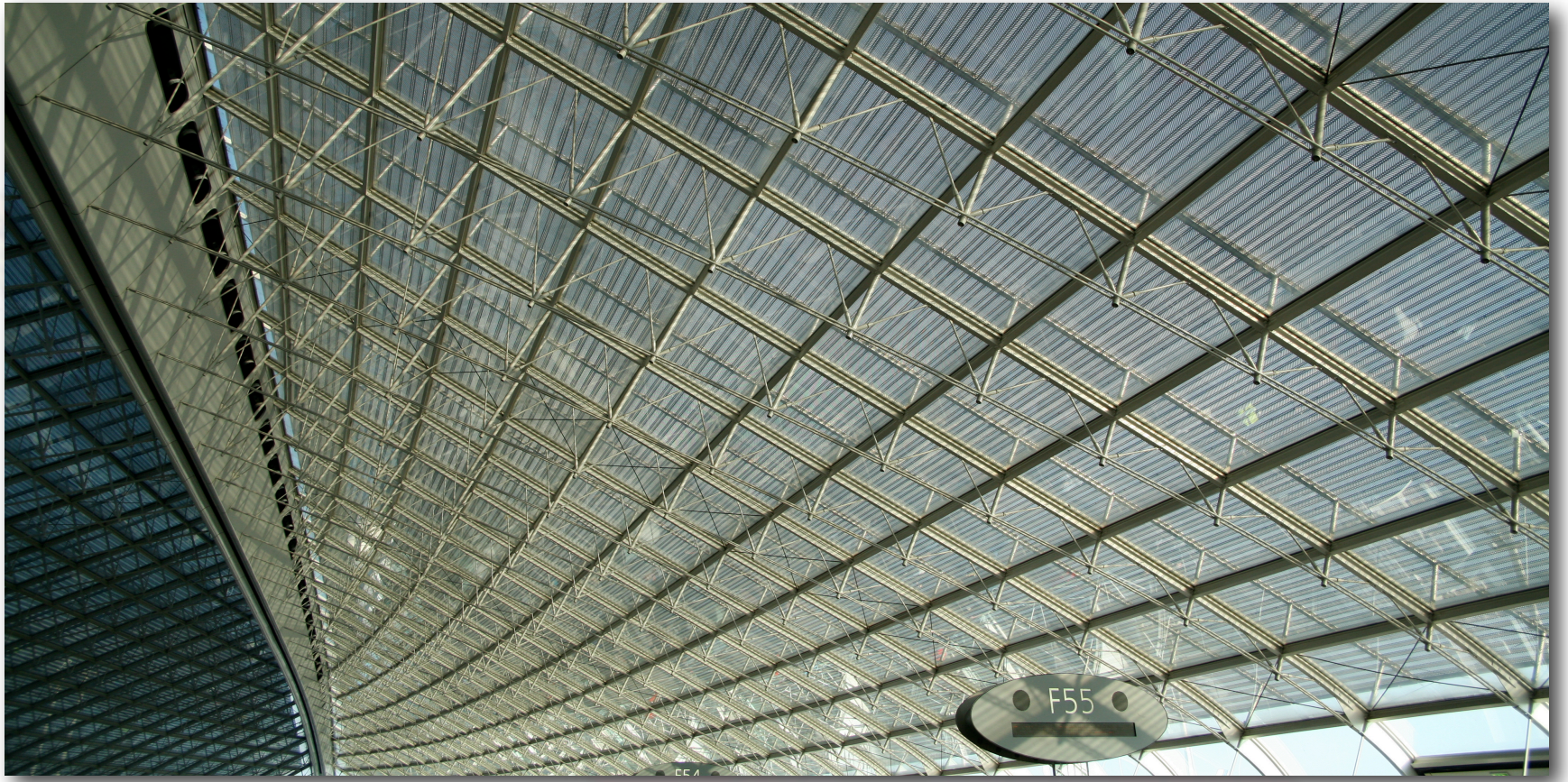


Our Sample Application

- Persistent connections from clients
- Serialize java objects across the wire
- Clients connect and are given a *unit of work*, which in this case, is just an instruction for how long to wait until getting their next instruction.

Monitoring Performance

- Thread activity via `jconsole`
- CPU Activity via Activity Monitor
  - (or your favorite tool)



Limitations

Scalability

- JVM limit on number of threads
- The lovely `java.lang.OutOfMemoryError: unable to create new native thread`

Lets convert to MINA!

- Server side first
  - (Client to come soon)

Re-testing

New Limitations?

- Java Serialization takes up CPU time
  - (a profiler would reveal this)
- OS limit of per-process open files
  - (consult the documentation for your OS)
  - sysctl / ulimit to view/change on unix-like systems

MINA on the client

- Since we will be using MINA's built-in support for building protocols, the *ProtocolCodecFilter*
  - Any socket client can talk to MINA
  - We're using MINA on both sides for simplicity in our examples.

Client is just like the server

- IoHandler and IoFilter's
- Can re-use filters on both client and server sides.

Implementing the protocol.

It still works!

Filters that ship today

- Logging
- Compression
- Blacklist
- SSL
  - Requires Java 5

Filters we are working on

- Traffic Shaping
- Rate Limiting
- Performance Monitoring

Some things built on MINA

- LDAP - Apache DS <http://directory.apache.org/subprojects/apacheds/index.html>
- Flash - red5 <http://www.osflash.org/red5>
- HTTP - AsyncWeb <http://asyncweb.safehaus.org/>

Performance Tips

- Set the number of IoProcessor's to be equal to the number of CPU cores.
- Benchmark! Users have found both heap and direct buffers, pooled and not pooled, to be beneficial depending on their workloads.
- For ultra-low latency with small message sizes on a local lan, disable Nagle's algorithm; the TCP_NODELAY flag.

Questions?

Thank You!