Apache Multipurpose Infrastructure for Network Applications
Building Scalable Network Applications

Dan Pritchett
Rearden Commerce
What are network applications?

- Application logic is primarily protocol handling

  Good Examples
  - SMTP
  - LDAP
  - FTP

- Tend to have a high ratio of I/O to application logic.
How have we solved this?

- Worker thread per request pattern
  - Listener thread accepts connections.
  - Connection established with client, delegated to a worker thread.
  - Worker remains bound to client until service request is complete.
Why doesn’t that work?

- Majority of thread time in I/O wait
- Threads are expensive
  - 500KB stack per thread
  - Context switching overhead
Enter SEDA

- **Staged Event Driven Architecture**
  - Non-blocking I/O generates events
  - Stages perform operations on events
Benefits of SEDA

- Decouple threads from I/O
  - One thread can service many I/O channels
  - Improved resource utilization
- Higher throughput
  - Linear degradation in service response time.
  - Equally important more balanced performance
SEDA Performance

**Throughput**

- **Apache**
- **Haboob**

**Response Time Distribution (ms)**

**Jain Fairness**

- Colorful bars and points for comparison.
SED A Implementations

- Apache MINA
- Haboob/Sandstorm
- LimeWire
- TerraLycos
- Rimfaxe Web Server
- Apache Excalibur
- SwiftMQ
- MULE
- Ocean Store
Apache MINA

- Multipurpose Infrastructure for Network Applications
  ➢ Apache implementation of SEDA
- A pattern and framework for developing network services.
- Java 5.0 NIO implementation
Projects Using MINA

- Apache Qpid
- Avis
- SubEthaSMTP
- Jive
- OpenLSD
- Red5
- QuickFIX/J
- Apache Directory Project
- Beep4j
MINA Application

Components

- Accept inbound connections
- Establish outbound connections
- Transform data
- Ancillary Processing
- Protocol Logic
MINA Filters

- Filters are the primary source of code factoring
- Categories of filter methods
  - Data Flow
    - messageReceived, filterWrite, messageSent
  - Session Management
    - sessionCreated, sessionOpen, sessionClosed, sessionIdle
  - Filter Management
    - onFilterPre/PostAdd, onFilterPre/PostRemove
  - Errors
    - exceptionCaught
Data Flow Events

- **messageReceived**
  - Inbound events

- **filterWrite**
  - Outbound events, prior to I/O

- **messageSent**
  - Outbound events, after I/O
Session Management

Events

- **sessionCreated**
  - Prior to I/O connection established

- **sessionOpened**
  - Once connection is established

- **sessionClosed**
  - Once connection is closed
Filter Lifecycle & Errors

- Filter additions
  - onPreAdd, onPostAdd

- Filter removals
  - onPreRemove, onPostRemove

- Errors
  - exceptionCaught - All calls from framework catch Throwable and route through this method.
  - This method should avoid throwing exceptions
Handlers

- Handlers implement the service logic.
- Handlers are a special filter.
  - Operations provided by IoHandler interface.
  - Same methods as filter, except filter lifecycle.
  - Installed automatically as last filter in chain
    - Last to receive inbound events
    - First to receive outbound events
Filter Chains

- Filters are attached to acceptors and connectors.
- Events pass through the chain in order.
  - Filter can control if events propagate or stop.
Typical MINA Application

- Transform streams to objects
- Track collection of objects as logical
- Administration and management.
- Protocol Semantics
Codec

- Encoder/Decoder
  - Translates byte streams into objects
  - Streams arrive as byte buffers
    - Buffers and parsing units orthogonal.
      - For example
        - XML document may be in many buffers
        - One buffer may contain many HTTP headers
      - Don’t assume anything about buffer boundaries and protocol boundaries.
Session Management

- Protocols may involve multiple events in a logical session.
  - HTTP for example:
    - Request
    - Headers
    - 0 or more body blocks
- Session filter provides aggregation of events for downstream filters.
Logging/Administration

- Loosely coupled metrics
  - Log protocol information
  - Statistics
  - Flow control
- Each capability can be written as a single purpose, reusable filter.
Example Service

- Simple Time Server
  - TCP/IP protocol
  - Client sends their time, ISO formatted GMT, line terminated.
  - Service responds with two lines:
    - Server time, ISO formatted GMT
    - Delta in milliseconds between the client and server.
Step 1: Service Configuration

SocketAcceptorConfig config = new SocketAcceptorConfig();
config.setBacklog(5);
config.setReuseAddress(true);
DefaultIoFilterChainBuilder chain = config.getFilterChain();
TextLineCodecFactory factory = new TextLineCodecFactory();
ProtocolCodecFilter codec = new ProtocolCodecFilter(factory);
chain.addLast("codec", codec);
chain.addLast("date", new ISODateFilter());
chain.addLast("logger", new LoggingFilter());
chain.addLast("counter", new ServiceCounterFilter());
Step 2: Acceptor Setup

TimeServiceHandler handler = new TimeServiceHandler();
SocketAcceptor acceptor = new SocketAcceptor();
InetSocketAddress listenAddr = new InetSocketAddress(8011);
acceptor.bind(listenAddr, handler, config);
public class ISODateFilter extends IoFilterAdapter {
    private static final DateFormat formatter = new SimpleDateFormat("yyyy-MM-ddTHH:mm:ss.S");

    public void messageReceived(NextFilter next, IoSession session, Object message) {
        Date date = formatter.parse((String)message);
        next.messageReceived(session, date);
    }

    public void filterWrite(NextFilter next, IoSession session, WriteRequest request) {
        Response response = (Response)request.getMessage();
        StringBuilder text = new StringBuilder();
        text.append(formatter.format(response.date)).append("\r\n");
        text.append(Long.toString(response.delta)).append("\r\n");
        WriteRequest encoded = new WriteRequest(text.toString());
        next.filterWrite(session, encoded);
    }
}
public class ServiceCounterFilter extends IoFilterAdapter {
    private int requests = 0;

    public void messageReceived(NextFilter next, IoSession session, Object message) {
        requests += 1;
        next.messageReceived(session, message);
    }

    public int getRequests() {
        return requests;
    }
}
public class TimeServiceHandler extends IoHandlerAdapter {
    public void messageReceived(IoSession session, Object message) {
        Date client = (Date) message;
        Response response = new Response(client);
        session.write(response);
    }
}

Dan Pritchett — Apache Multipurpose Infrastructure for Network Applications Building Scalable Network Applications
public class Response {
    Date date;
    long delta;

    public Response(Date client) {
        date = new Date();
        delta = client.getTime() - date.getTime();
    }
}
Tips: Threads

- I/O Threads - Associated with Acceptors & Connectors
  - Manage NIO channels and deliver events into the chain.
  - Take care to avoid starving NIO channels.
- ExecutorFilter dispatches event on separate thread
  - Can be installed anywhere in filter chain
- Remember context switching adds to client latency
Tips: Factoring

- Filters are the unit of factoring
  - Filter over head is small.
  - Small, single purpose filters are preferable to handler logic.

- Filters vs Objects
  - No simple rule for using one vs. the other.
  - In fact, they are complimentary. Hierarchy of filters is reasonable.
Further Reading

- **SEDA**
  - [http://www.eecs.harvard.edu/~mdw/proj/seda/](http://www.eecs.harvard.edu/~mdw/proj/seda/)

- **MINA**