Java NewIO

Java Network Programming
Chapter 12
p384

Example Code:
http://examples.oreilly.com/9780596007218/

Further examples/discussion
http://gee.cs.oswego.edu/dl/cpjslides/nio.pdf
http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html

Servers

- Goal: performance
  - how many connections can you handle a minute
- Traditional architecture:
  - threads/connections
  - why? how does it work? what is the advantage?
  - threads use, minimally, about 1 MB of RAM*
- Problems?
  - overhead?
- Clients?

* Java Network Programming, p384

New IO

- Goal: Even better performance
  - how do we lower overhead
  - non-blocking IO
  - Socket / File / etc

Event driven

  - Channel
  - Selector
  - SelectionKey
  - ByteBuffer

In some circumstances, you may not need NIO: tune thread stack size

http://stackoverflow.com/questions/4057853/java-i-o-vs-java-new-i-o-nio-with-linux-nptl

Thread Pool

- Set of running threads
  - no create/destroy overhead
  - limited in number
    - but can grow
  - Why is a many connections to one thread ratio acceptable?

Client Example (chargen RFC 864)

```
try
{
  SocketAddress address = new InetSocketAddress(args[0], port);
  SocketChannel client = SocketChannel.open(address);

  ByteBuffer buffer = ByteBuffer.allocate(74);
  WritableByteChannel out = Channels.newChannel(System.out);

  while (client.read(buffer) != -1) {
    buffer.flip();
    out.write(buffer);
    buffer.clear();
  }
}
catch (IOException ex)
{
    ex.printStackTrace();
}
```

http://examples.oreilly.com/9780596007218/

Server

```
ServerSocketChannel serverChannel;
Selector selector;
try
{
  serverChannel = ServerSocketChannel.open();
  ServerSocket ss = serverChannel.socket();
  InetSocketAddress address =
          new InetSocketAddress(port);
  ss.bind(address);
  serverChannel.configureBlocking(false);
  selector = Selector.open();
  serverChannel.register(selector,
                         SelectionKey.OP_ACCEPT);
}
catch (IOException ex)
{
  ex.printStackTrace();
  return;
}
```

http://examples.oreilly.com/9780596007218/

while(true)

```
selector.select();

Set readyKeys = selector.selectedKeys();
Iterator iterator = readyKeys.iterator();
while (iterator.hasNext())
{
   SelectionKey key = (SelectionKey) iterator.next();
   iterator.remove();
    if(key.isAcceptable())
    {
       // next slide
    }
    else if(key.isWriteable())
    {
       // next next slide
    }
}
```

Various try/catch blocks removed for clarity

http://examples.oreilly.com/9780596007218/

key.isAcceptable()

```
ServerSocketChannel server = (ServerSocketChannel)
                                    key.channel();
SocketChannel client = server.accept();

System.out.println("Accepted connection from " + client);
client.configureBlocking(false);

SelectionKey clientKey = client.register(selector,
                                  SelectionKey.OP_WRITE);
ByteBuffer buffer = ByteBuffer.allocate(74);

buffer.put(rotation, 0, 72); // rotation: global array
buffer.put((byte) '\r');
buffer.put((byte) '\n');
buffer.flip();
clientKey.attach(buffer);
```

http://examples.oreilly.com/9780596007218/

key.isWriteable()

```
SocketChannel client = (SocketChannel) key.channel();
ByteBuffer buffer = (ByteBuffer) key.attachment();

if (!buffer.hasRemaining())
{
  // Refill the buffer with the next line
  // details removed
  buffer.put((byte) '\r');
  buffer.put((byte) '\n');
  // Prepare the buffer for writing
  buffer.flip();
}
// try to write entire buffer
// update the position in the buffer
client.write(buffer);
```

http://examples.oreilly.com/9780596007218/

Full Duplex

NonblockingSingleFileHTTPServer.java

```
else if (key.isWritable())
{
  SocketChannel channel =
      (SocketChannel) key.channel();
  ByteBuffer buffer =
      (ByteBuffer) key.attachment();
  if (buffer.hasRemaining())
  {
    channel.write(buffer);
  }
  else
  {  // we're done
    channel.close();
  }
}
```

```
else if (key.isReadable())
{
  SocketChannel channel =
      (SocketChannel) key.channel();
  ByteBuffer buffer =
          ByteBuffer.allocate(4096);
  //read but ignore HTTP request
  channel.read(buffer);

  // switch channel to
  // write-only mode
  key.interestOps(
      SelectionKey.OP_WRITE);
  key.attach(
      data);
}
```

http://examples.oreilly.com/9780596007218/

Thread Pool

- java.util.concurrent
  - Class ThreadPoolExecutor

```
ThreadPoolExecutor
(int corePoolSize,                          // threads to keep
int maximumPoolSize,                        // max threads
long keepAliveTime,                         // how long will a thread
                                            // stay idle before being
                                            // removed?
TimeUnit unit,                              // units for above
BlockingQueue<Runnable> workQueue, // hold tasks
ThreadFactory threadFactory,                // how to create new thread
                                            // can change priority,
                                            // ThreadGroup, etc.
RejectedExecutionHandler handler)           // how to handle overflow
```

Practical Concerns

```
select()

if (isReadable())
{
  Connection conn = (Connection) key.attachment();
  // add to thread pool

}
else if( isWriteable() )
{
  // add to thread pool

}
```

C Code version

```
// POSIX

#include <sys/select.h>

int select(int nfds, fd_set *restrict readfds,
           fd_set *restrict writefds, fd_set *restrict errorfds,
           struct timeval *restrict timeout);

or

int pselect(int nfds, fd_set *readfds, fd_set *writefds,
            fd_set *exceptfds, const struct timespec *timeout,
            const sigset_t *sigmask);
```

- In which set of file descriptors will a read/write not block?
  - input sets overwritten to indicate which FDs are ready
  - *man select_tut*