Netty Contributor Meet-Up 2021-10-28

- Buffer Updates (Chris V)
- Netty Core/Contrib repo separation (Chris V)
- HTTP Caching (Aayush A)

Buffer Updates

Chris Vest

Buffer Updates

What’s new since the incubator repo & the netty/netty PR

- Simpler life cycle
- All feedback addressed
- More integrated + Porting has started
- More convenience APIs

Simpler life cycle

- Simplified API — allocate/split/copy/receive → close
- No visible reference counting — no more acquire/release
- No “is it owned or not” mental overhead — buffers are always owned
- No more aliasing — required for “always owned”
- No more slice/duplicate — would cause aliasing
- Instead we have split, copy, send

All feedback addressed

- All feedback addressed from incubator, PR
- CompositeBuffer now an interface with static factory methods
- ByteCursor no longer supports iterating longs at a time
- Buffers always big-endian — no longer configurable
- Read-only state is irreversible — but use constBufferSupplier for constants

More integrated

- ChannelConfig.get/setBufferAllocator
- ChannelHandlerContext.bufferAllocator
- DefaultGlobalBufferAllocator.DEFAULT_GLOBAL_BFUFER_ALLOCATOR’
- New leak detector
- Native memory address now hidden — avail. in forEach*Component
  - Can cause aliasing, but necessary evil for I/O touch points

More convenience APIs

- New CompositeBuffer.decomposeBuffer → Buffer[]
- New Buffer.skipReadable/Writable
- New Buffer.bytesBefore(byte)
- New Buffer.writeCharSequence
- New Buffer.read/writeSplit
- BufferAllocator.allocate now support zero-sized buffers
- BufferAllocator.copyOf for byte[] & ByteBuffer
- New Send.map combinator

Questions on Buffer Updates?

Netty Core/Contrib Repo. Sepr.

Chris Vest

The Plan

netty/netty →

netty/
- netty ← *core module*
- netty-contrib-xml ← *pilot module*
- netty-contrib-transport-sctp
- netty-contrib-haproxy
- netty-contrib-memcache
- netty-contrib-mqtt
- netty-contrib-redis
- netty-contrib-smtp
- netty-contrib-socks
- netty-contrib-stomp
- netty-contrib-handler-proxy

+ stuff in netty-codec? pcap in handler?

Netty 5 only

Benefits

- Independent sets of maintainers
- Independent release schedule
    - Contrib modules are not changed very often
    - Contrib modules should depend on netty-core version *ranges*
- Separation of concerns
    - Contrib modules should not depend on netty internals
- Reduced maintenance burden on netty core

Phase 1

- Create pilot module/repository
- https://github.com/netty/netty-contrib-xml
- Move code
- Change java package (io.netty.contrib.*)
- Set up builds
- Set up 1-click release process (ideally)

Phase 2

- Repeat for all other non-core modules
- Want maintainership of something?
  - Discuss with Norman/Trustin
- Will we port over to the new Buffer API before moving code?
  - No

Future phases?

- Move SslHandler to a netty-tls module?
- Move BoringSSL from tcnative into a netty-tls-native module?
- Move compression algorithms to modules?
- Move NIO transport to its own module?
- Avoid Java module package name conflicts

Questions on Core/Contrib separation?