

Diffusion

Anatomy of a high-performance Java server

Philip Aston

Docklands.LJC, November 2016

Product Architect at Push Technology

- paston@pushtechnology.com

Ex-BEA, ex-Oracle

- WebLogic Server / Coherence / Oracle Event Processing
- *Professional Oracle WebLogic Server* (Wrox 2009)
- *J2EE Performance Testing* (Expert Press 2002)

The Grinder

The Product Slide

Push Technology

- Real-time data distribution for web, mobile, and IoT

Diffusion

- Real-time messaging at scale
- Java server
- Android, iOS/OS X, JavaScript, .NET, C, and Java SDKs

Reappt

- Diffusion As A Service on IBM BlueMix and AppDirect

What's coming up?

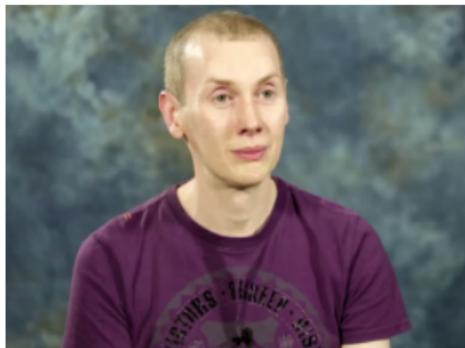
High Performance Broadcast, in Java

Publish and Subscribe

High Performance Broadcast

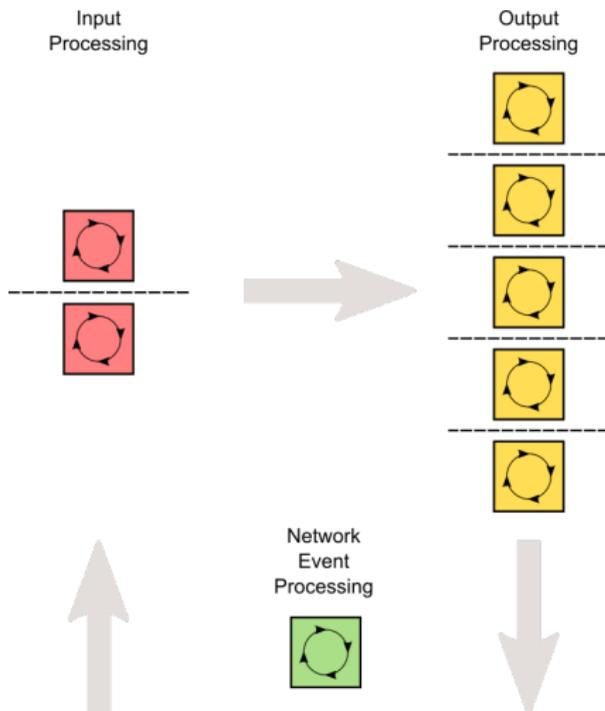
Applied "Mechanical Sympathy"

Martin Thompson

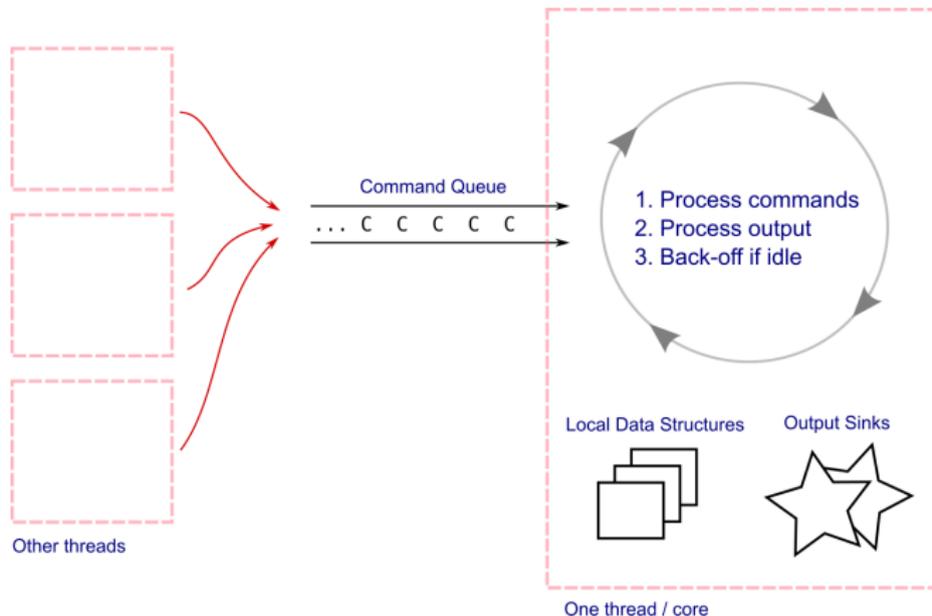


[mechanical-sympathy mail list](#)

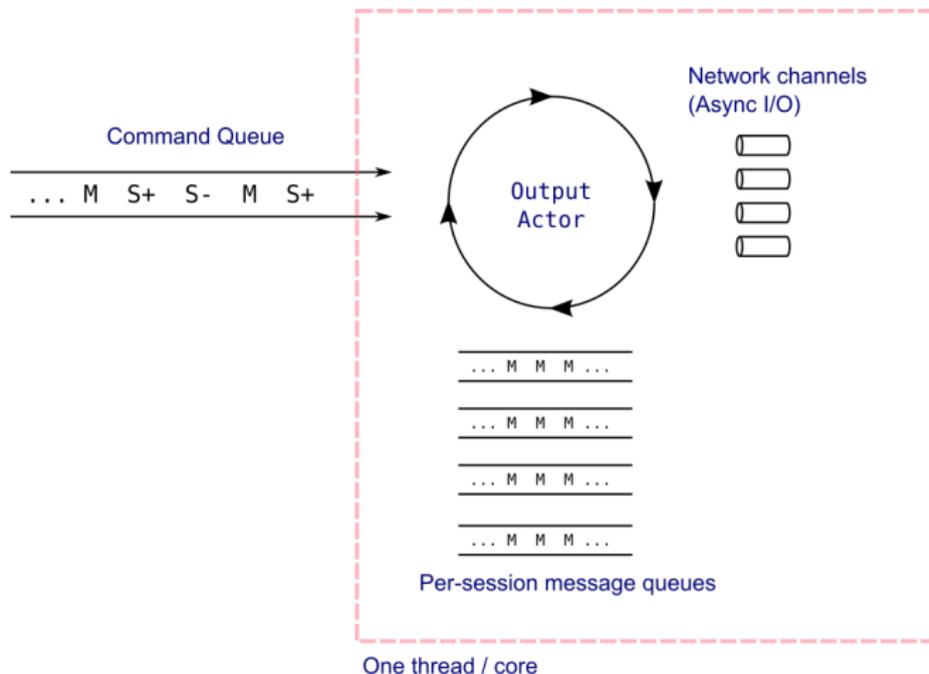
The big picture



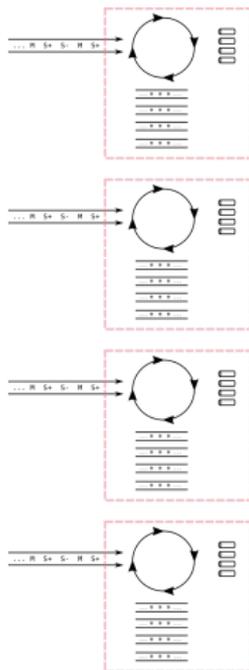
Actor - a processing pattern



Output processing



Benefits



Simple sequential programming

- Non-thread safe data structures
- Consistent order

Memory hierarchy friendly

- Hot threads bound to single cores
- Thread-exclusive data structures

Parallel

- Scales across cores and sockets
- Pipelining

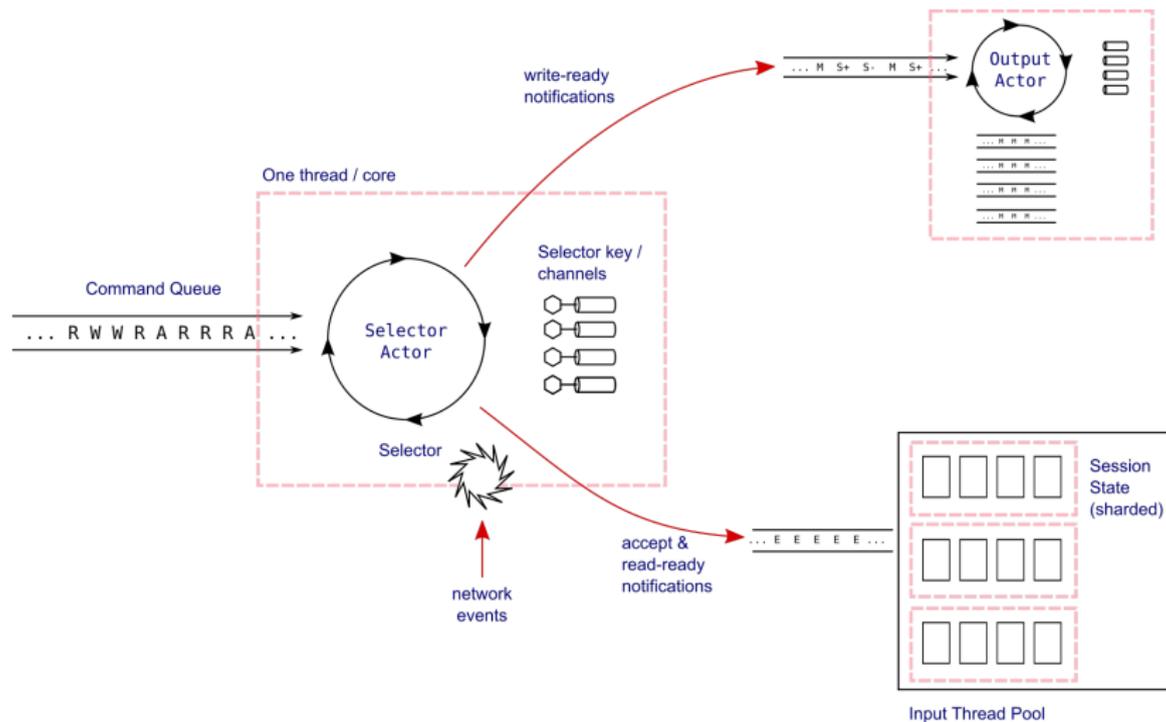
Selector

- Selected key set is not thread-safe
- Easiest if registration for CONNECT, ACCEPT, READ, WRITE, and close all occur on a single thread

Channel

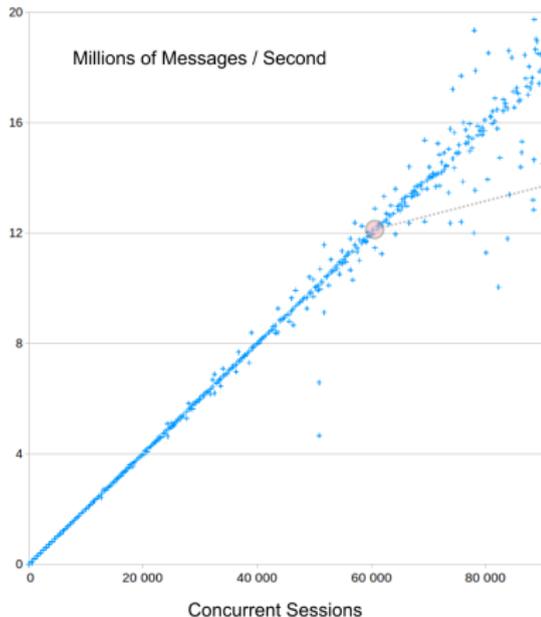
- For consistent, ordered processing
 - Read from single thread
 - Write from a single thread

Selector Actor

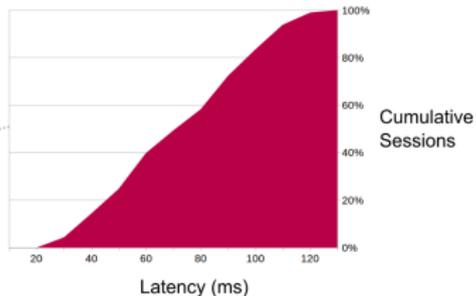


Benchmark results

Throughput vs Sessions



Latency @ 60K Sessions



Dell Power Edge R720
Dual socket Xeon E5-2630 6-core hyper-threaded
32GB RAM
10G network interface

Inter-thread communication

Choosing a concurrent, in-memory queue

JDK options don't cut it

- `ArrayBlockingQueue`
 - Single lock synchronises producers and consumers
- `ConcurrentLinkedQueue`
 - allocates for each offer
 - is unbounded

Nitsan Wakart



- [PsyLobSaw blog](#)
- [JCTools](#)

MpscArrayQueue

- Multi-producer, single consumer, bounded queue
- Lock-free

Details

- Separate consumer and producer fields
 - False sharing avoidance
 - Optimised memory barriers
- Producers use CAS to update fields
- Consumer uses *Fast Flow* method

See Lock Free Queues or Queue Evolution: from 10M to 470M ops/sec for much more.

Publish and Subscribe

Diffusion pub/sub - it's different

- 1 Value-based interface
 - Developers focus on application **data**, not **messages**
 - SDKs provide idiomatic native API
- 2 Stateful topics - the *inverted data grid*
 - The server maintains current value for each topic
 - Clients synchronised with topic value on subscription
 - Deltas follow
- 3 Interest-based subscription
 - Clients provide topic selectors
 - Server matches selectors against topics and authorisation

Take aways

- 1 Event-oriented / non-blocking throughout
- 2 Threads aligned to processing flow
 - Separate input and output processing
 - Minimal inter-thread communication via queues
- 3 Thread-exclusive data
 - Sessions partitioned across threads
 - Processing batched where possible
 - Global state shared as immutable snapshots

<http://docs.pushtechnology.com/>

<http://developer.reappt.io/docs/index.html>

Thank you - Any Questions?