How to do 100K+ TPS at less than 1ms latency

Martin Thompson & Michael Barker

QCon SF 2010

LMAX

Agenda

- Context Setting
- Tips for high performance computing (HPC)
- What is possible on a single thread???
- New pattern for contended HPC
- Q & A

LMAX

Who/What is LMAX?

- The London Multi-Asset Exchange
- Spin-off from Betfair into retail finance
- Access the wholesale financial markets on equal terms for retail traders
- We aim to build the highest performance financial exchange in the world

LMAX

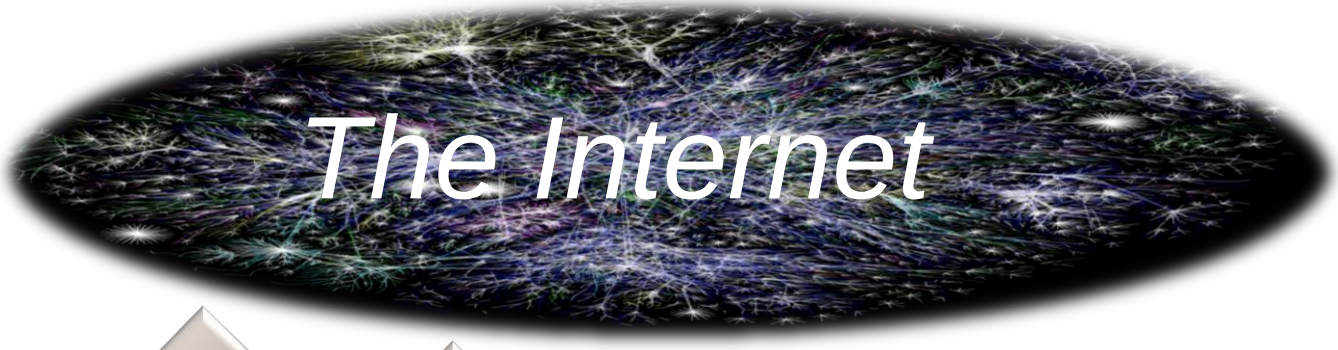
What is Extreme Transaction Processing (XTP)?

The Betfair Experience

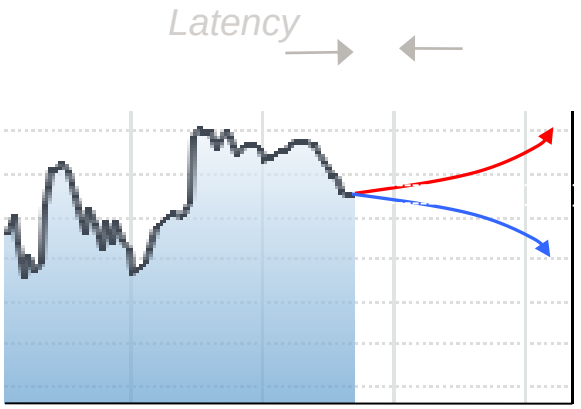
V

LMAX

What is Extreme Transaction Processing (XTP)?



The LMAX Model



LMAX

How not to solve this problem

LMAX

Phasers or Disruptors?

LMAX

Tips for high performance computing

1. Show good “Mechanical Sympathy”
2. Keep the working set In-Memory
3. Write cache friendly code
4. Write clean compact code
5. Invest in modelling your domain
6. Take the right approach to concurrency

LMAX

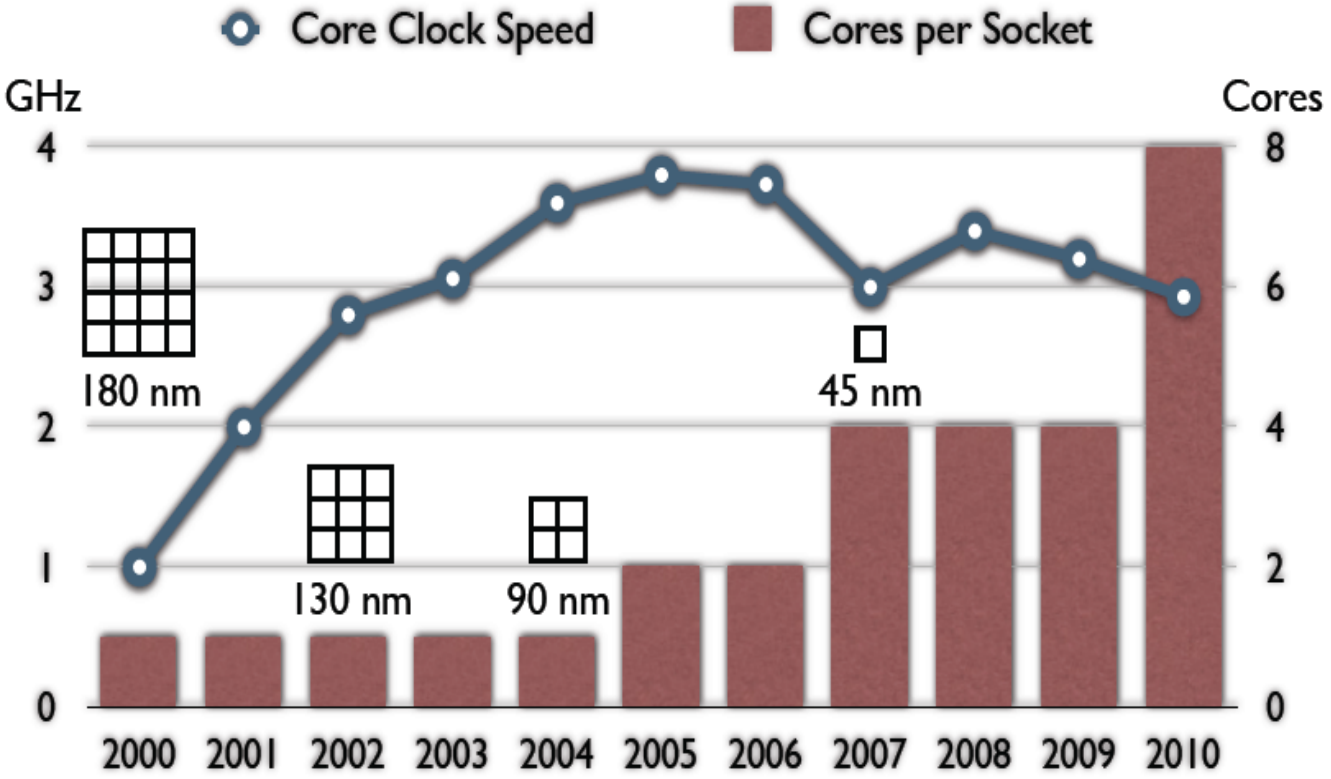
1. Mechanical Sympathy – 1 of 2

Memory

- Latency not significantly changed
- Massive bandwidth increase
- 144GB in a commodity machine

CPUs

- The GHz race is over
- Multi core
- Bigger smarter caches

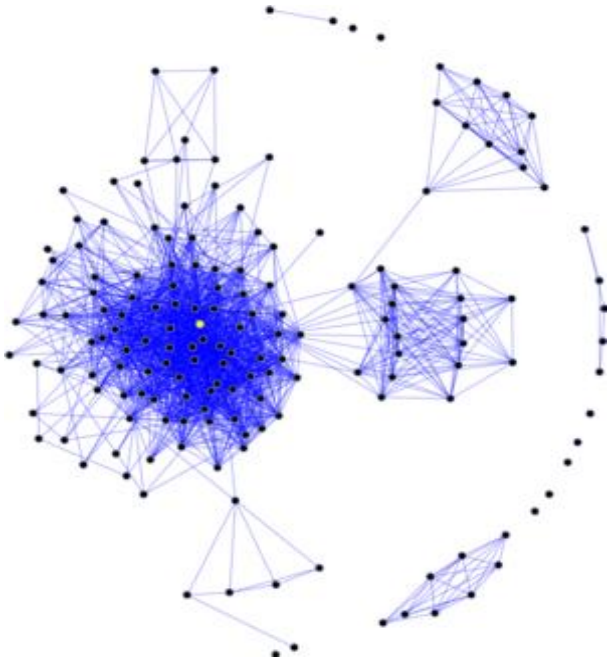


LMAX

1. Mechanical Sympathy – 2 of 2

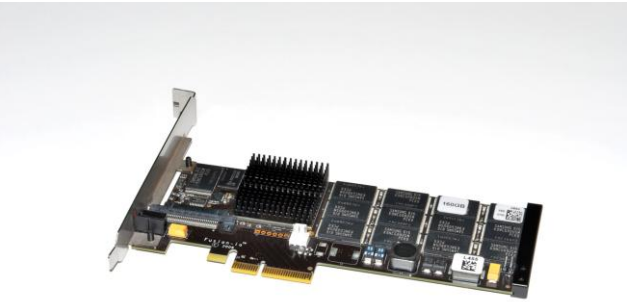
Networks

- Sub 10 microseconds for local hop
- Wide area bandwidth is cheap
- 10GigE is now a commodity
- Multi-cast is getting traction



Storage

- Disk is the new tape! Fast for sequential access
- SSDs for random threaded access
- PCI-e connected storage



LMAX

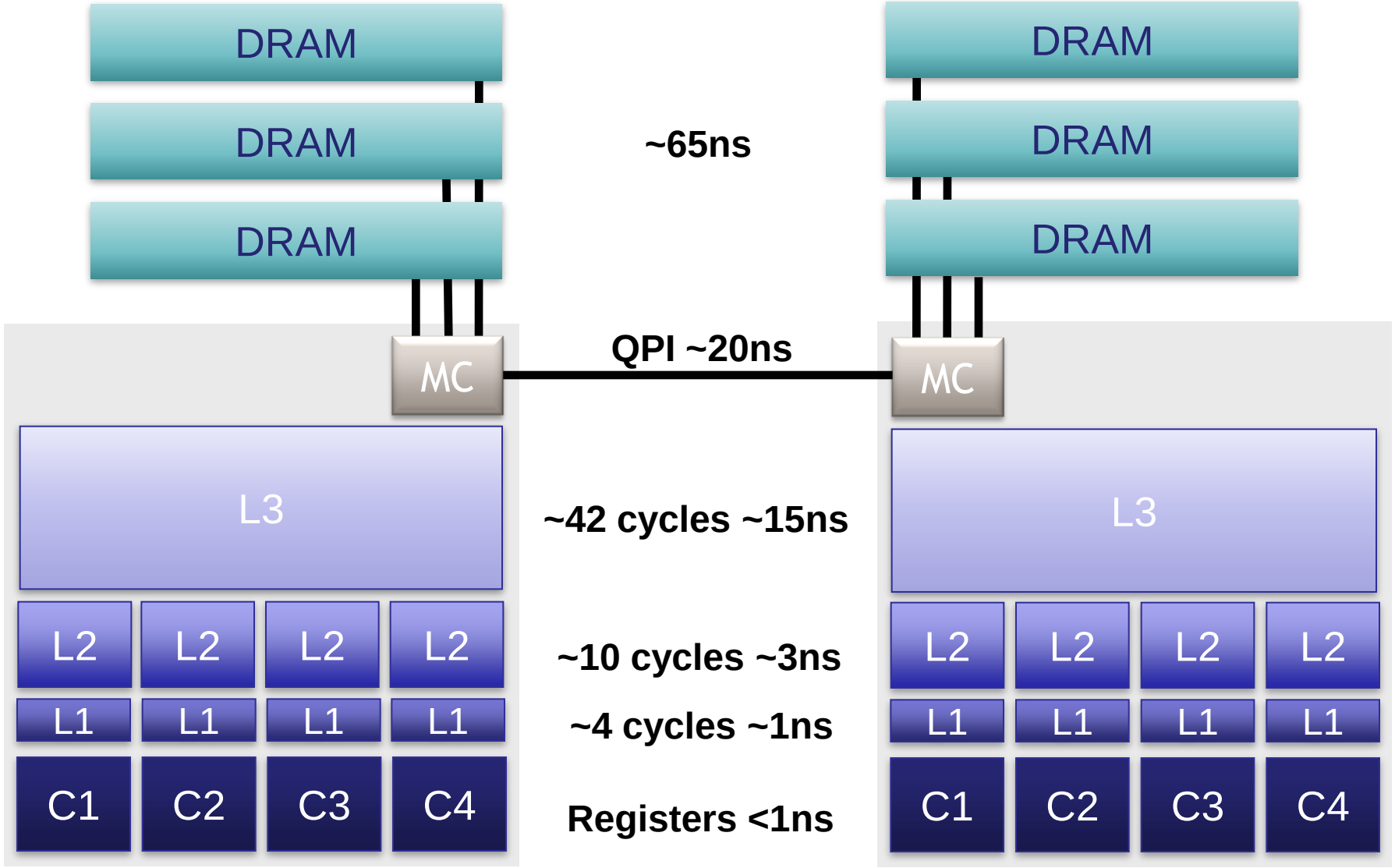
2. Keep the working set In-Memory

Does it feel awkward working with data remote from your address space?

- Keep data and behaviour co-located
- Affords rich interaction at low latency
- Enabled by 64-bit addressing

LMAX

3. Write cache friendly code

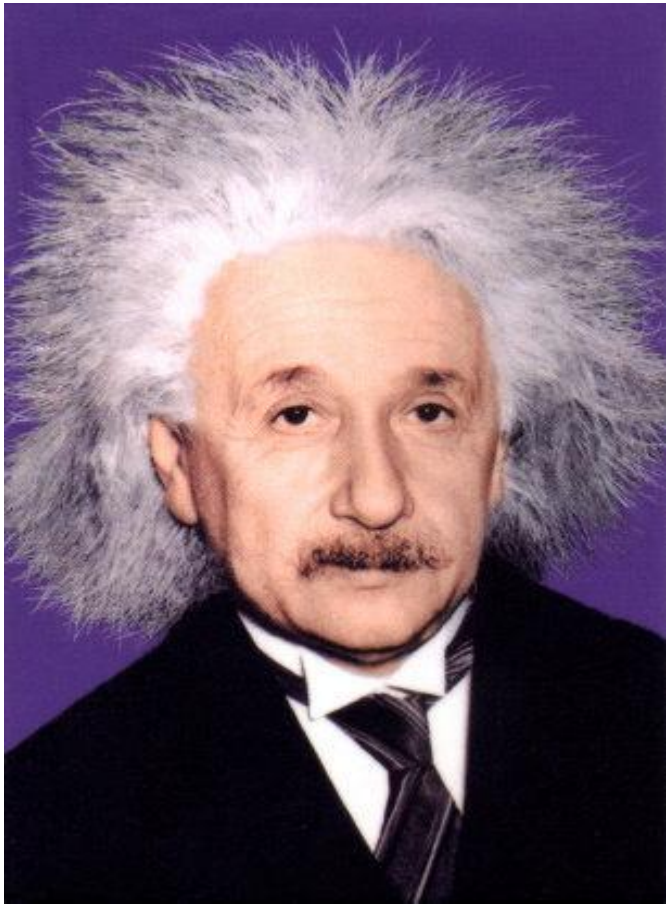


LMAX

4. Write clean compact code

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius -- and a lot of courage -- to move in the opposite direction."

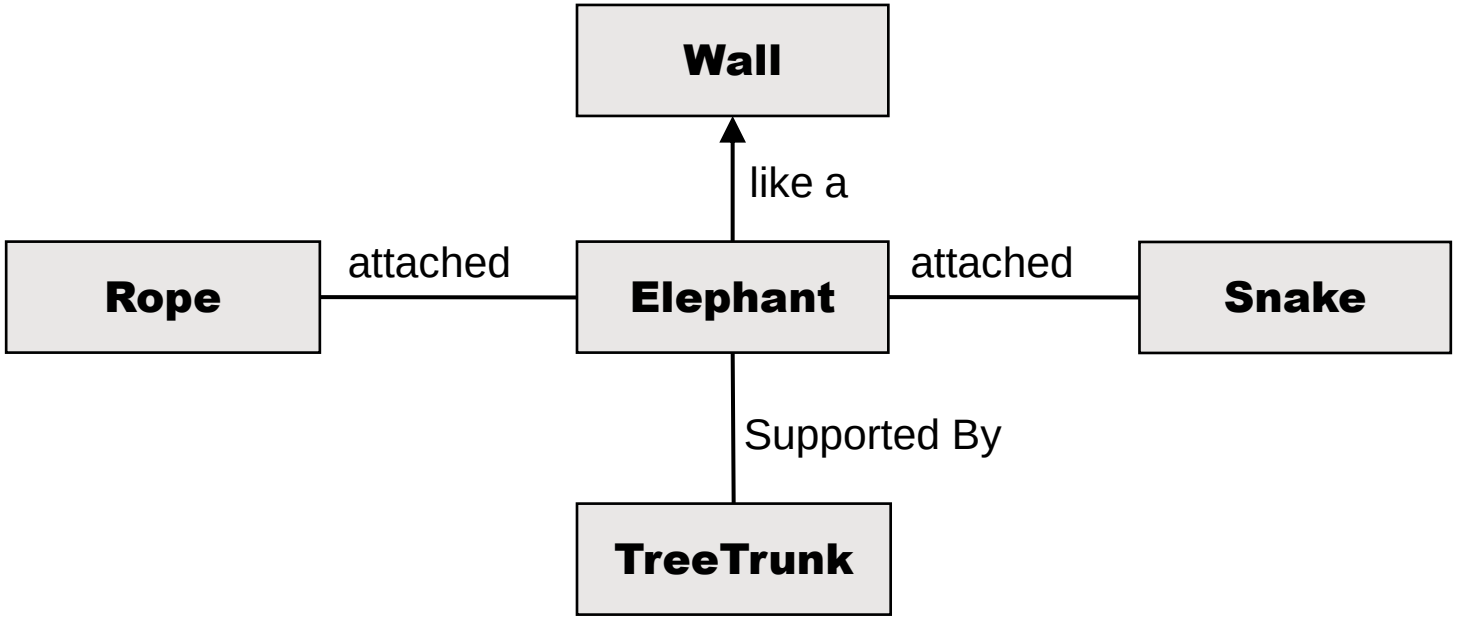
- Hotspot likes small compact methods
- CPU pipelines stall if they cannot predict branches
- If your code is complex you do not properly understand the problem domain
- Nothing in the world is truly complex other than Tax Law



LMAX

5. Invest in modelling your domain

Model of an elephant based on blind men touching one part each



- **Single responsibility – One class one thing, one method one thing, etc.**
- **Know your data structures and cardinality of relationships**
- **Let the relationships do the work**

LMAX

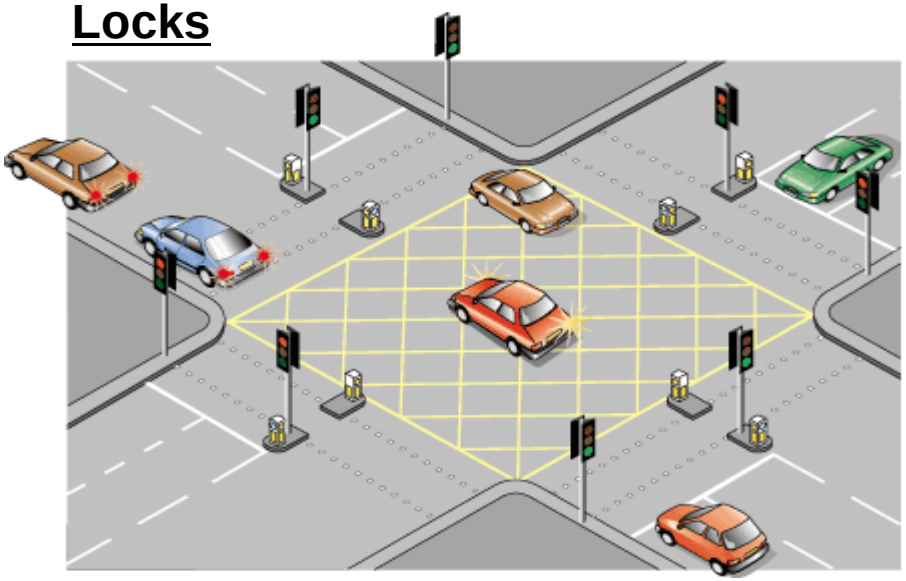
6. Take the right approach to concurrency

Concurrent programming is about 2 things:

***Mutual Exclusion*:** Protect access to contended resources

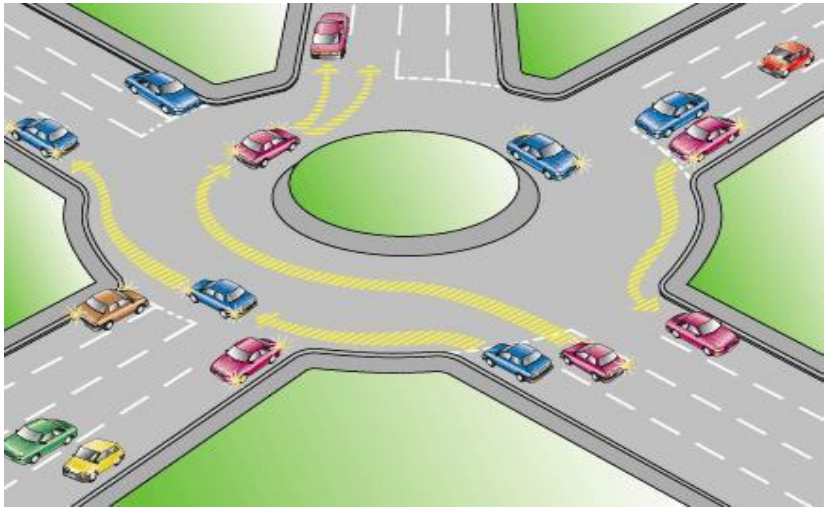
***Visibility of Changes*:** Make the result public in the correct order

Locks



- Context switch to the kernel
- Can always make progress
- Difficult to get right

Atomic/CAS Instructions



- Atomic read-modify-write primitives
- Happen in user space
- Very difficult to get right!

LMAX

What is possible when you get this stuff right?

On a single thread you have ~3 billion instructions per second to play with:

10K+ TPS

- If you don't do anything too stupid

100K+ TPS

- With well organised clean code and standard libraries

1m+ TPS

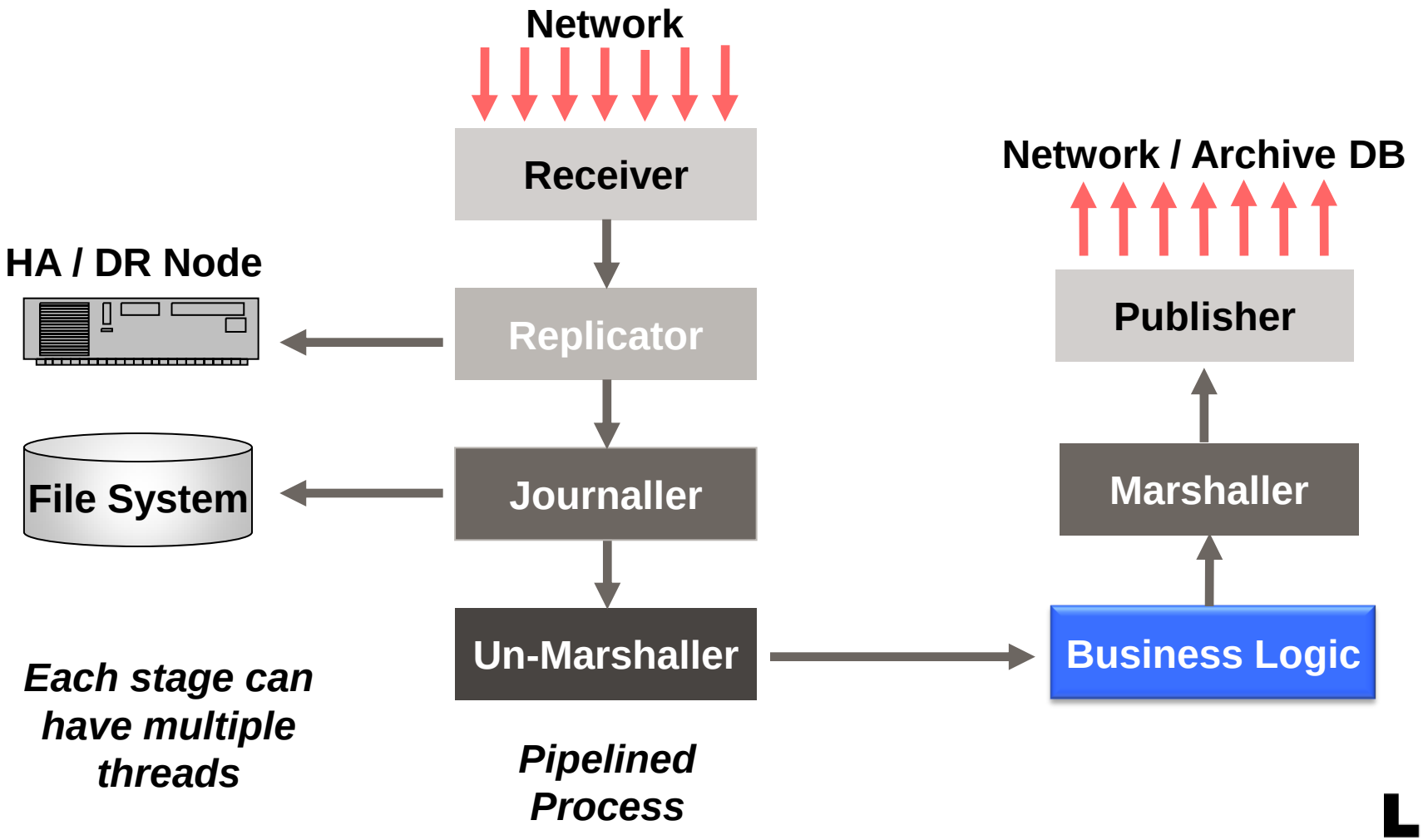
- With custom cache friendly collections
- Good performance tests
- Controlled garbage creation
- Very well modelled domain

- BTW writing good performance tests is often harder than the target code!!!

LMAX

How to address the other non-functional concerns?

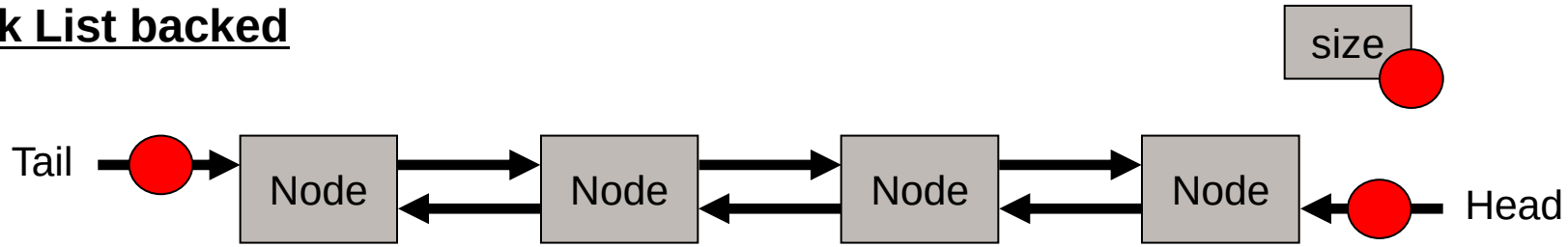
- **With a very fast business logic thread we need to feed it reliably**
  - > Did we trick you into thinking we can avoid concurrent programming?



LMAX

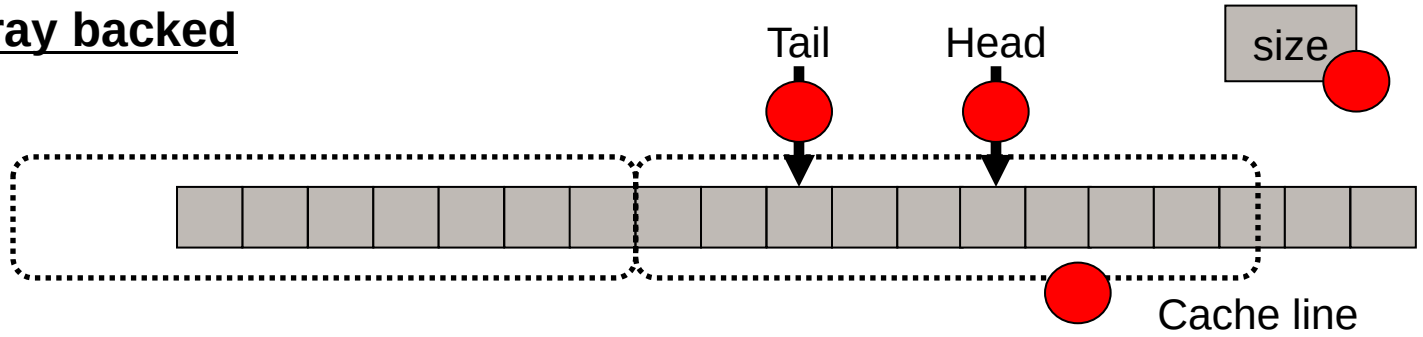
Concurrent access to Queues – The Issues

Link List backed



- Hard to limit size
- O(n) access times if not head or tail
- Generates garbage which can be significant

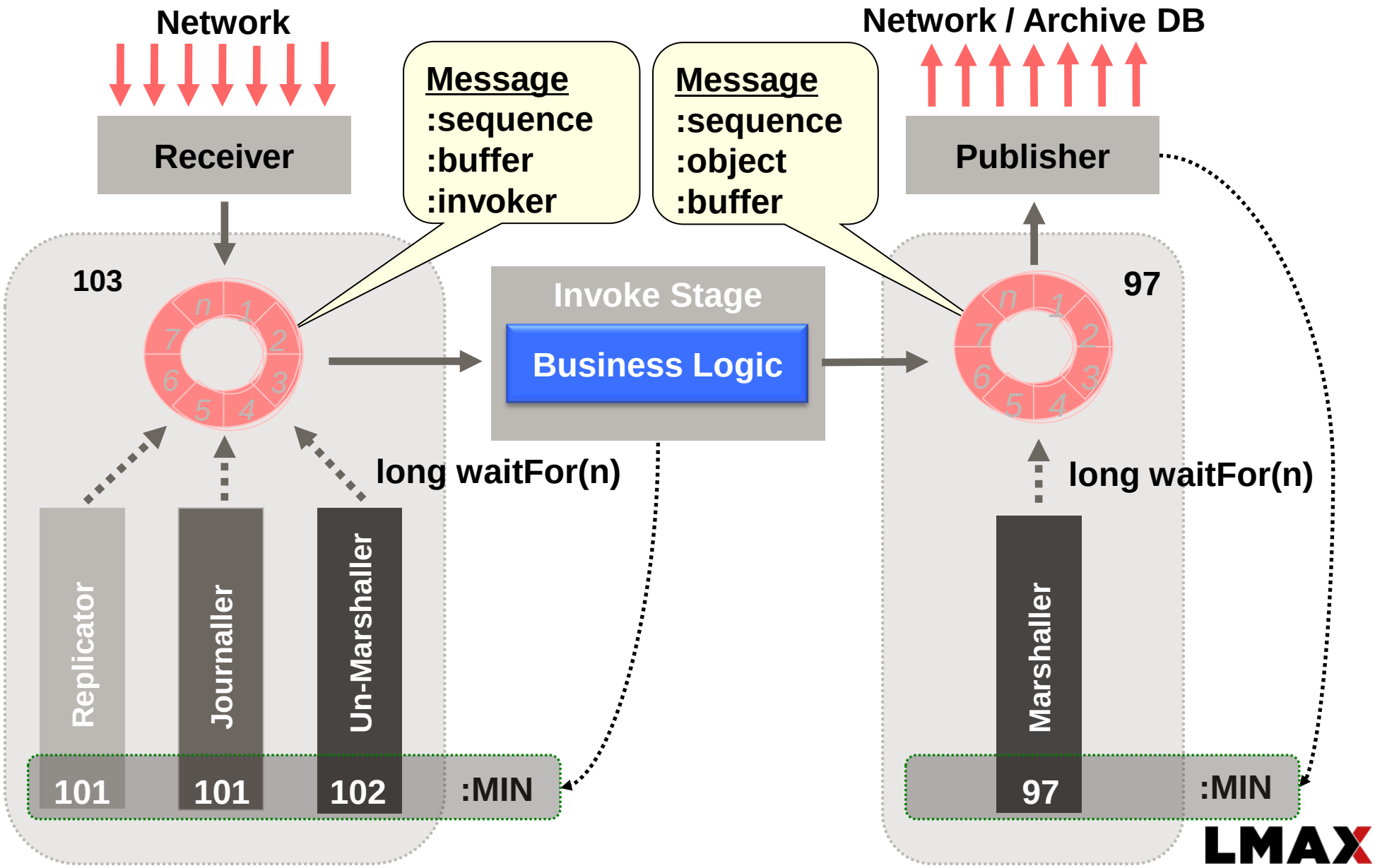
Array backed



- Cannot resize easily
- Difficult to get *P *C correct
- O(1) access times for any slot and cache friendly

LMAX

Disruptor


Network
Receiver
Message
:sequence
:buffer
:invoker
Message
:sequence
:object
:buffer
Network / Archive DB
Publisher
103
97
Invoke Stage
Business Logic
long waitFor(n)
long waitFor(n)
Replicator
Journaller
Un-Marshaller
Marshaller
101
101
102
:MIN
97
:MIN
LMAX

Quick Recap

- **Most developers have an incorrect view of hardware and what can be achieved on a single thread**
- **On modern processors a cache miss is your biggest cost**
- **Push concurrency into the infrastructure, and make it REALLY fast**
- **Once you have this, you have the world that OO programmers dream of:**
  - Single threaded
  - All in-memory
  - Elegant model
  - Testable code
  - No infrastructure or integration worries

LMAX

Wrap up

Q & A

jobs@lmax.com