LMAX Disruptor:

100K TPS at less than 1ms latency

Dave Farley

Martin Thompson

GOTO Århus 2011

LMAX

LMAX History

- Spin-off from Betfair the world’s largest sports betting exchange
- Massive throughput and customer numbers

- LMAX has the fastest order execution for retail trading
- Institutional market makers providing committed liquidity
- Real-time risk management of retail customers

Disruptor

How not to solve this problem

Disruptor

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

1. Mechanical Sympathy

Is it really “Turtles all the way down”?

What is under all these layers of abstraction?

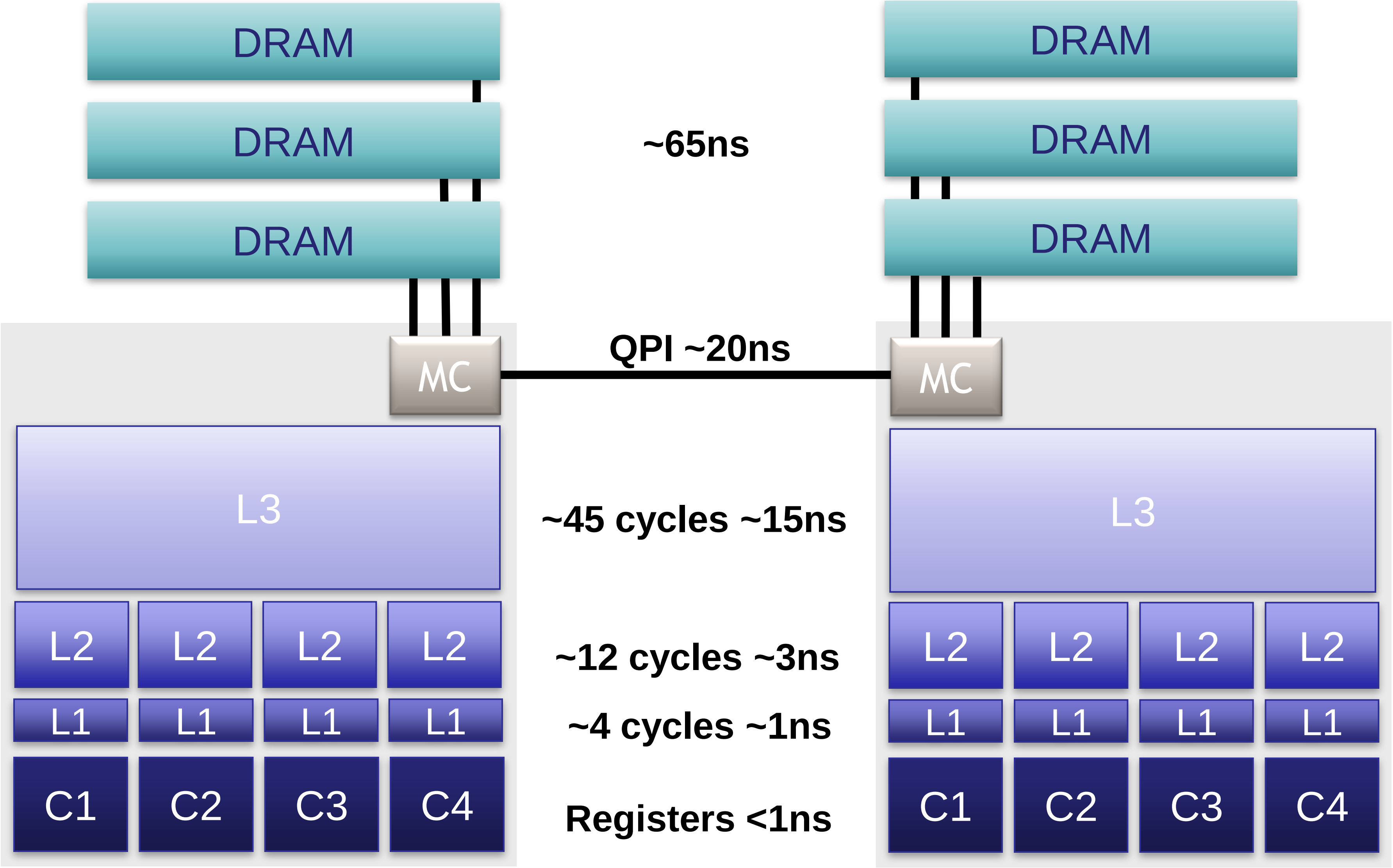
> *"The most amazing achievement of the computer software industry is its continuing cancellation of the steady and staggering gains made by the computer hardware industry.“*
>
> *- Henry Peteroski*

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

3. Write cache friendly code



~65ns

QPI ~20ns

~45 cycles ~15ns

~12 cycles ~3ns

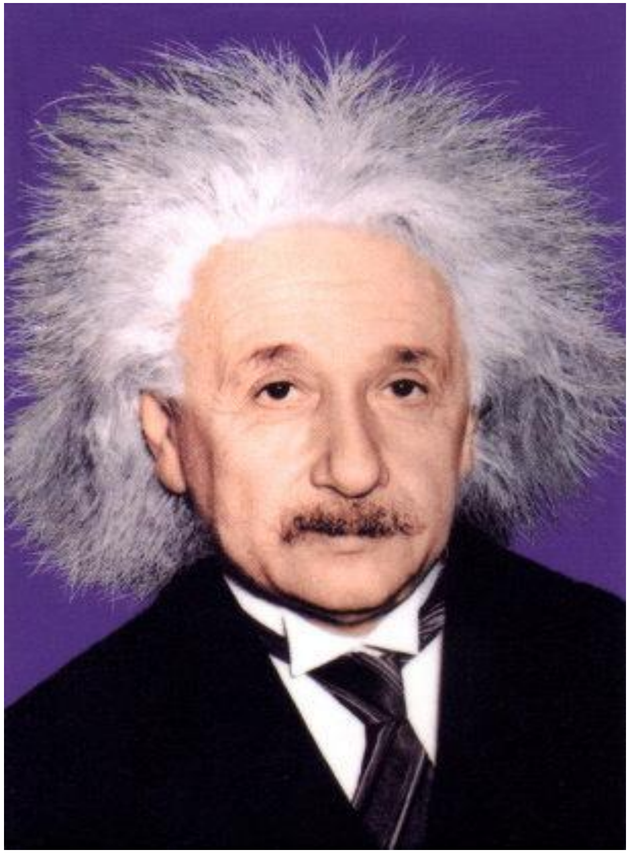
~4 cycles ~1ns

Registers <1ns

4. Write clean compact code

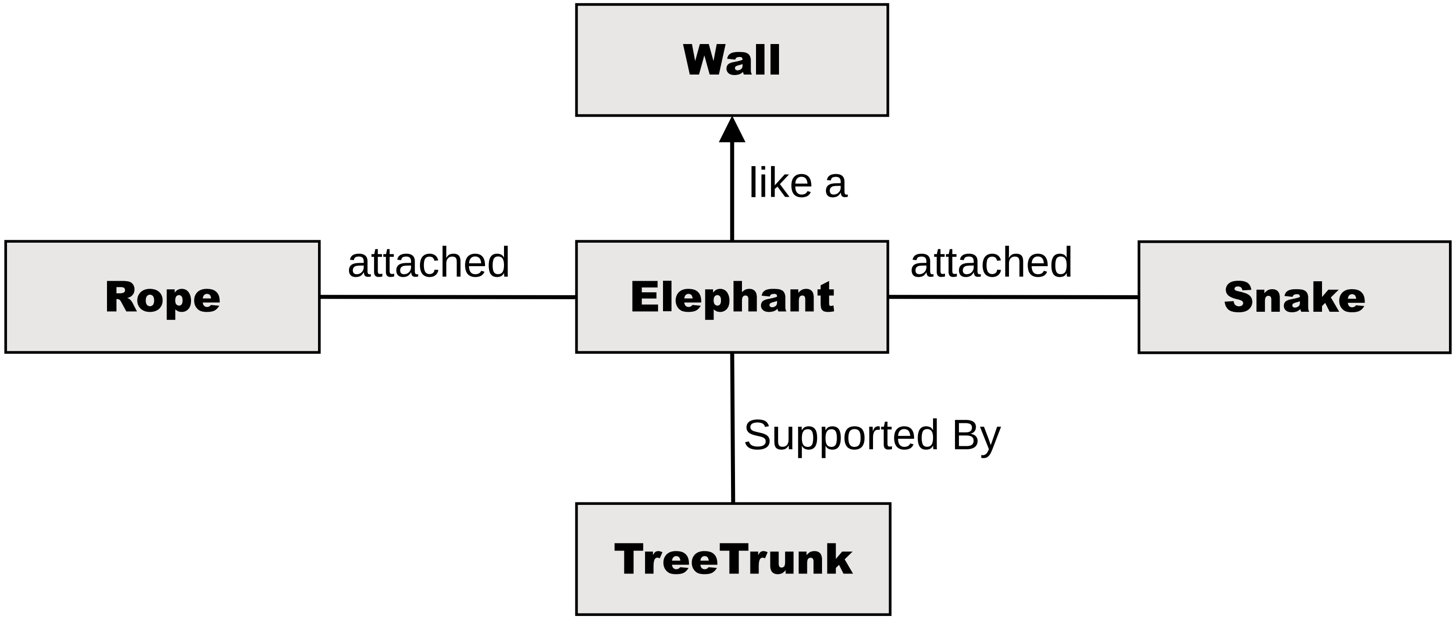
"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius -- and a lot of courage -- to move in the opposite direction."

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



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

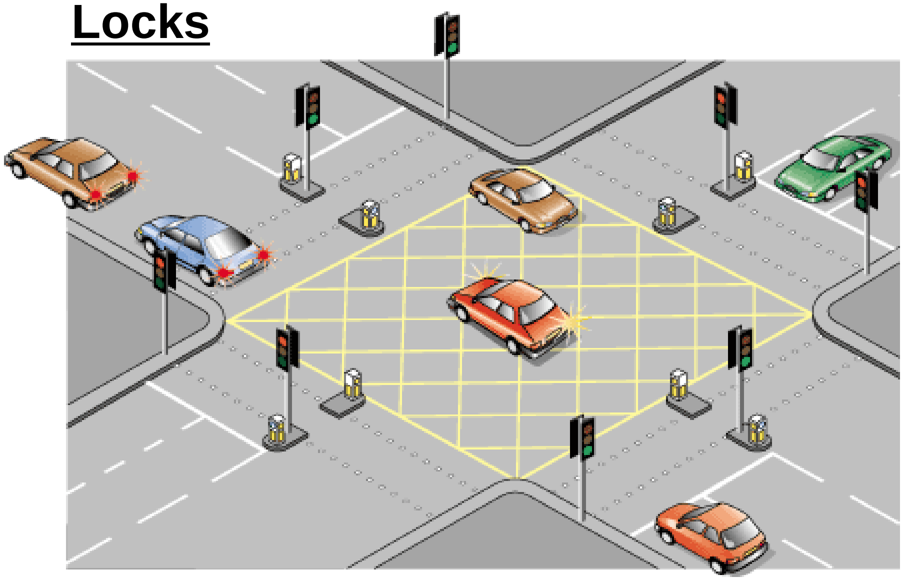
6. Take the right approach to concurrency

Concurrent programming is about 2 things:

Mutual Exclusion: Protect access to contended resources

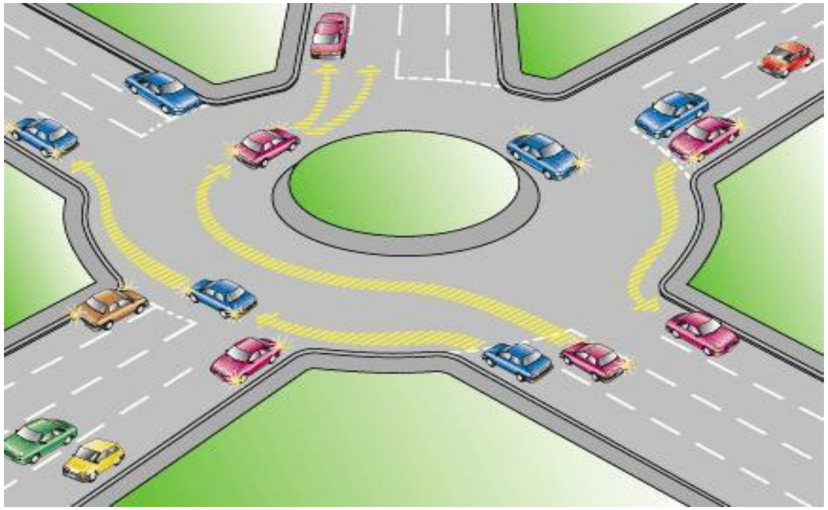
Visibility of Changes: Make the results public in the correct order

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!

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

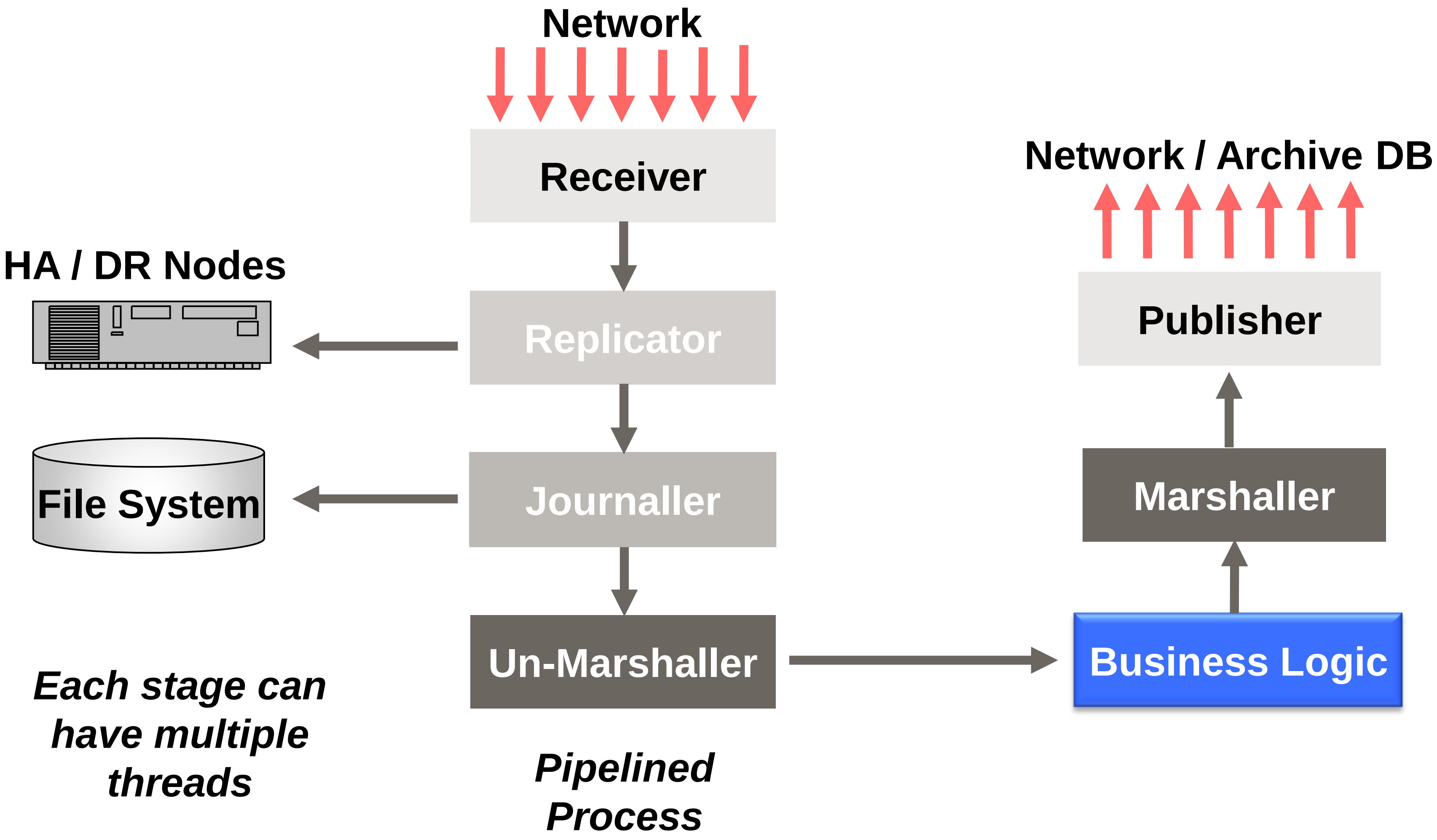
1m+ TPS

- 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!!!

How to address the other non-functional concerns?

- With a very fast business logic thread we need to feed it reliably

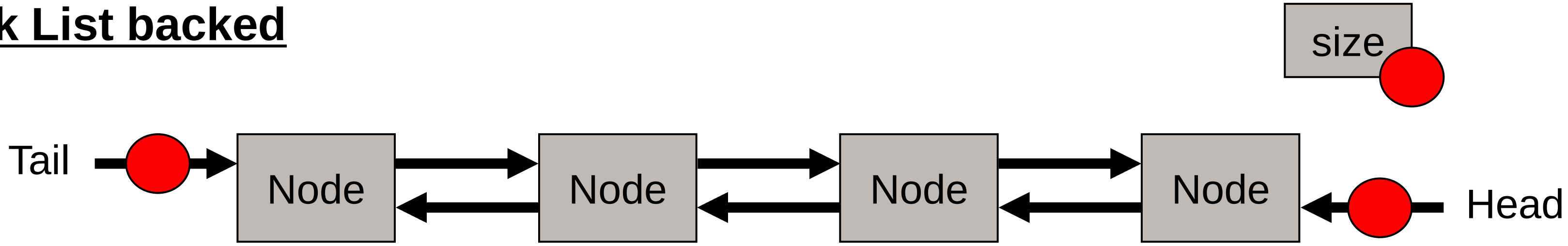


Each stage can have multiple threads

Pipelined Process

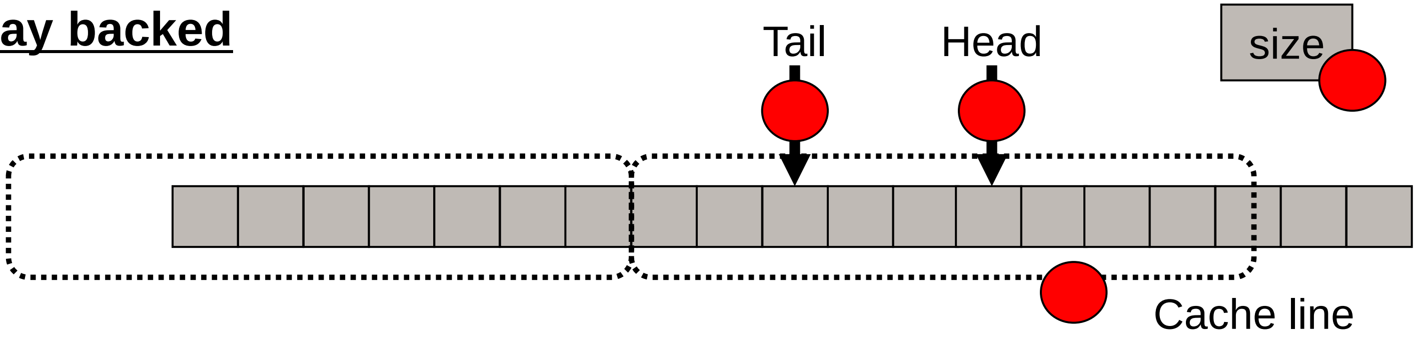
Concurrent access to Queues – The Issues

Link List backed



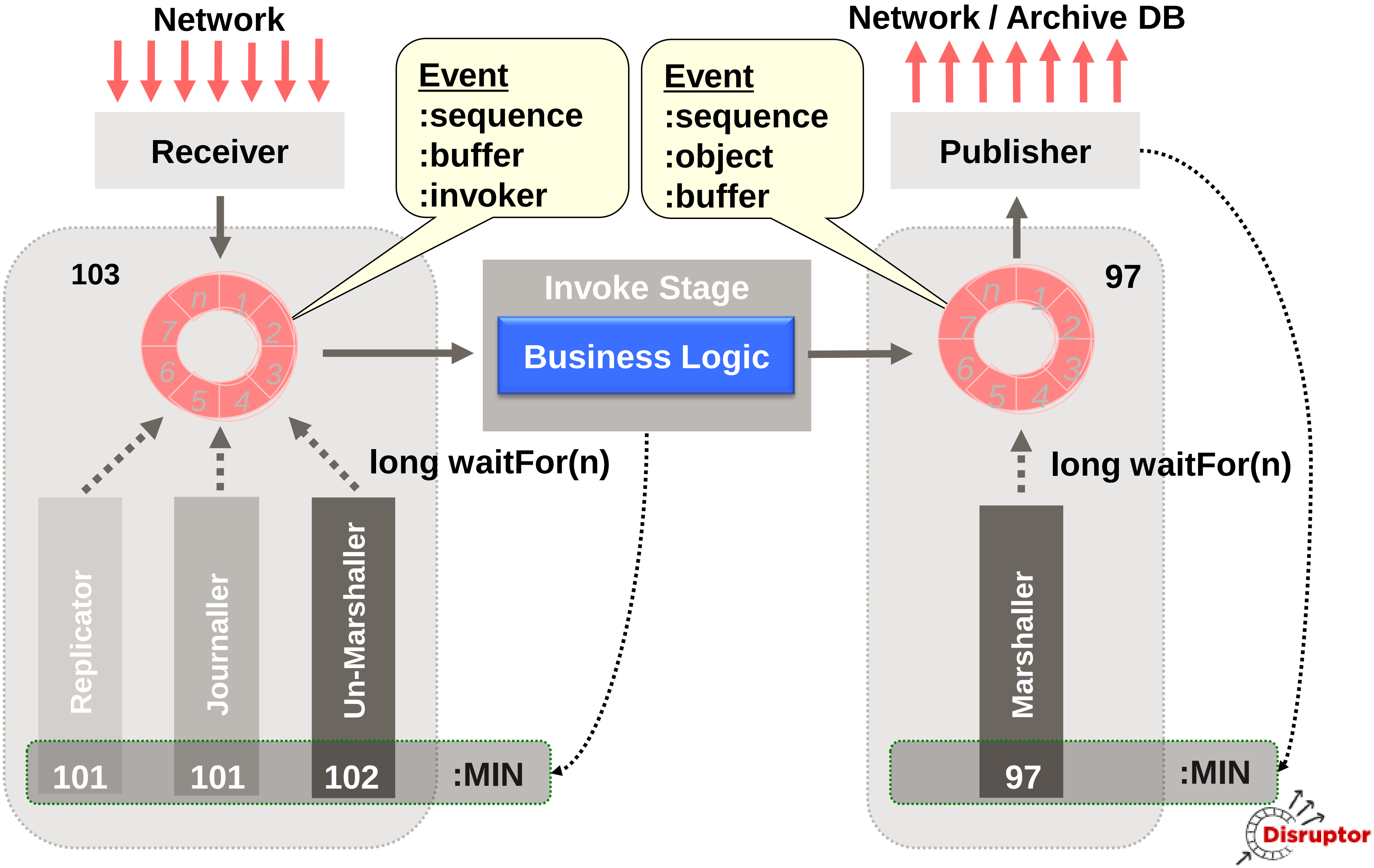
- Hard to limit size
- O(n) access times if not head or tail
- Generates garbage which can be significant

Array backed



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

Disruptor in Action



Disruptor – Concurrent Programming Framework

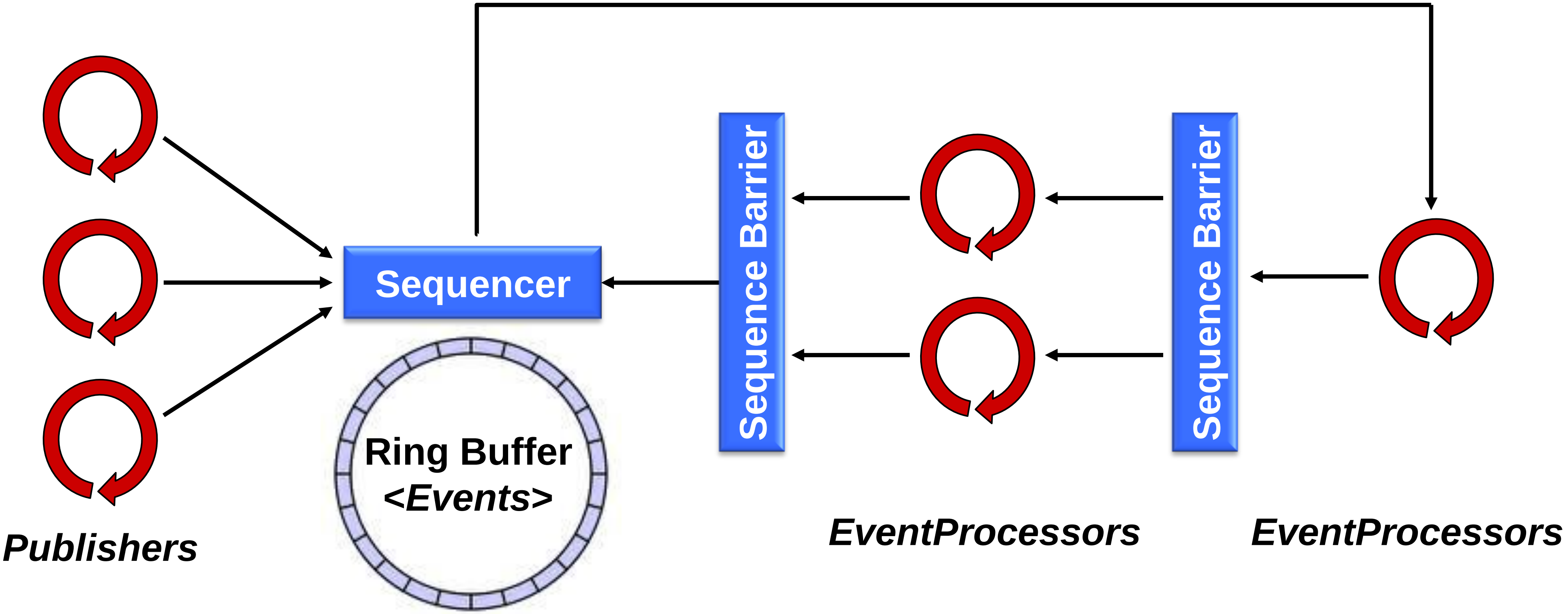
Open Source project: http://code.google.com/p/disruptor/

- Very active with ever increasing performance and functionality
- Wide applicability
  - Exchanges/Auctions, Risk Models, Network Probes, Market Data Processing, etc.

How do we take advantage of multi-core?

- Pin threads to cores for specific steps in a workflow or process
- Pass messages/events between cores with “Mechanical Sympathy”
- Understand that a “cache miss” is the biggest cost in HPC
- Measure! Don’t believe everything you read and hear
  - Let’s bring back science in computer science!

The Disruptor Pattern



CPU Core per thread

Disruptor

Wrap UP

http://code.google.com/p/disruptor/

http://www.davefarley.net/

http://mechanical-sympathy.blogspot.com/

jobs@lmax.com