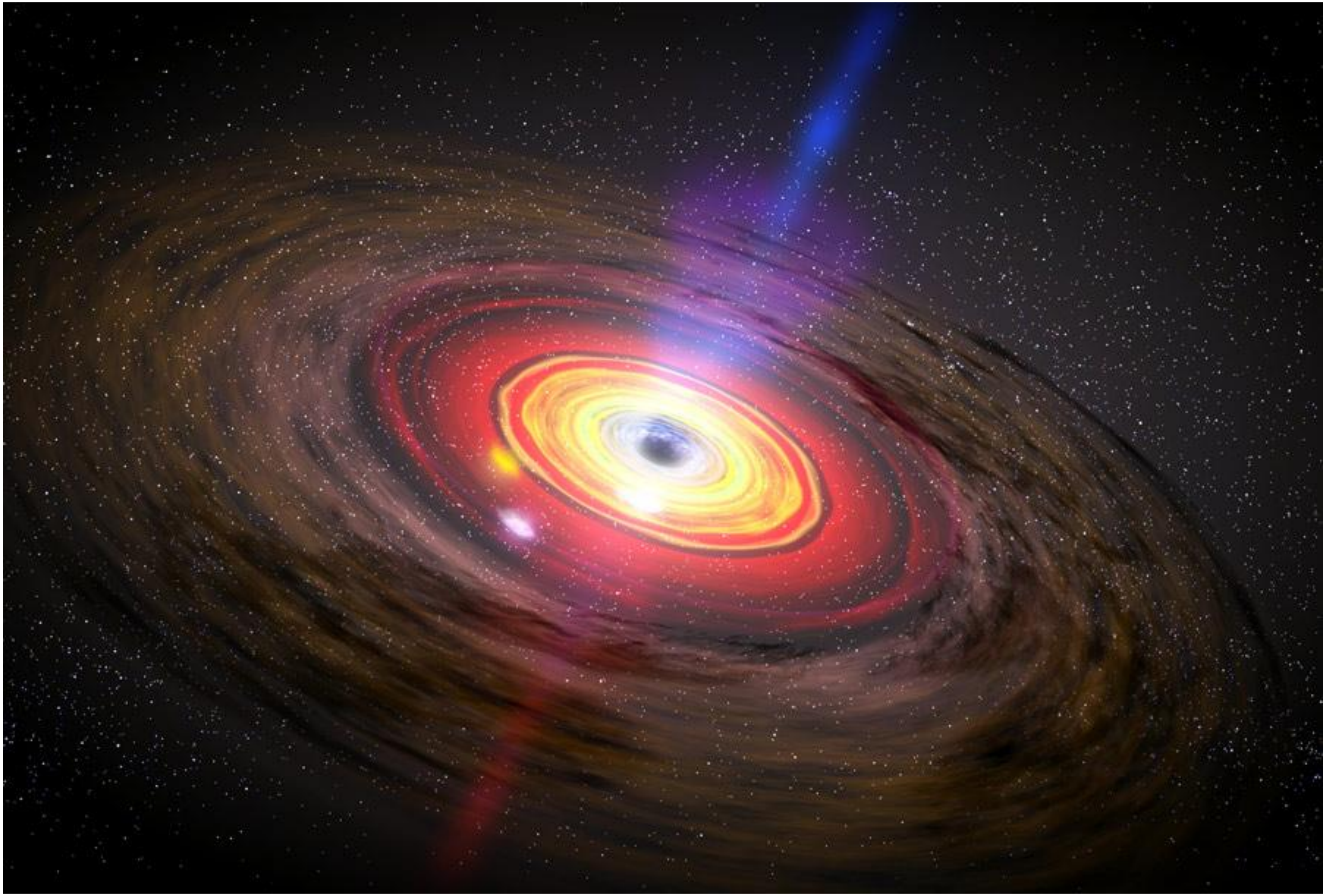
Real Logic

ACCELERATING SOFTWARE

Event Sourced Architectures for High Availability

Martin Thompson - @mjpt777

Our Disaster Recovery Plan Goes Something Like This...
HELP! HELP!

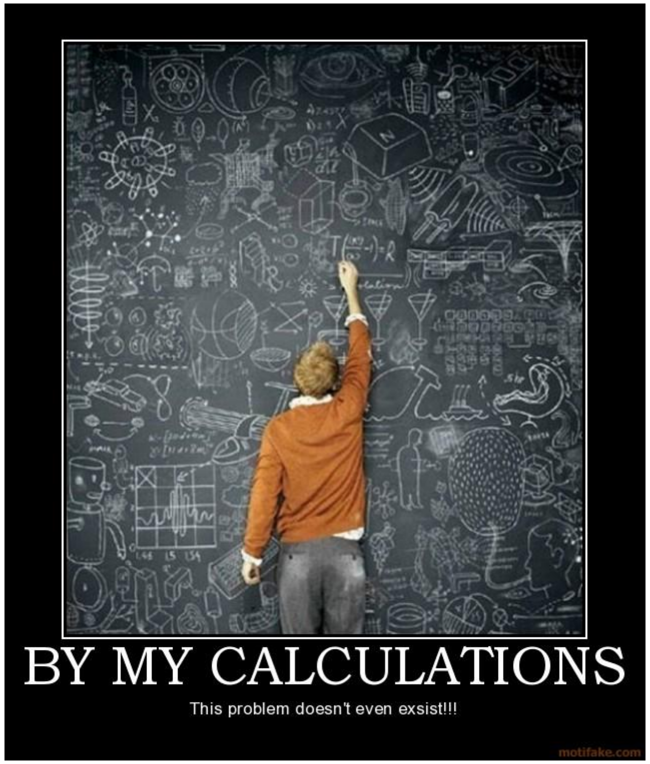


What Is *“High Availability”*?

- Availability refers to ability of the user community to access a system – not about Uptime!
- By “High” availability we generally mean the system is always there when we need it
- The 9’s are the typical way this is measured
    - 99.999%? When did the issue occur?
- MTBF – Mean Time Between Failures
- MTTR – Mean Time To Recover !!!
- Bathtub curve for Failure Rates
- System pauses (e.g. Garbage Collection)
- What about hot upgrade?

The “Truth” About Production Outages

- Admin “Cock-ups”
- Clustering Software
- Hardware Failures
- Software Bugs



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x00000000,0x00000002,0x00000000,0xFCBAC2A4)


***   CRASHDD.SYS - Address FCBAC2A4 base at FCBAC000, DateStamp 36bb6f3c

Beginning dump of physical memory
Dumping physical memory to disk: 100
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

High Availability: The Good, The Bad, The Ugly!

- *The Good*: Queries
  - > Go parallel with lots of replicas
- *The Bad*: Updates
  - > Some problems cannot be made parallel but some can
  - > Lock step clusters
- *The Ugly*: Distributed Resilience
  - > Latency
  - > Eventual Consistency
  - > Data Loss
  - > CAP Theorem

Transaction Processing & High Availability

1. Migrate between known good states
2. Replicate the step

Databases

> Oracle: SCNs, RAC nodes, replication

> MySQL Cluster: Shards, 2PC, deltas and snapshots

> MySQL: Clustered file systems, replication

- Tandem NonStop – hardware & software stack with a message passing kernel
- IMS TM transaction queue (Apollo Program)

“Event Sourced Design”

“Capture all changes to an application state as a sequence of events” – Fowler (2005)

“Apply a sequence of change events to a model in order” – *Thompson*

Modern References:

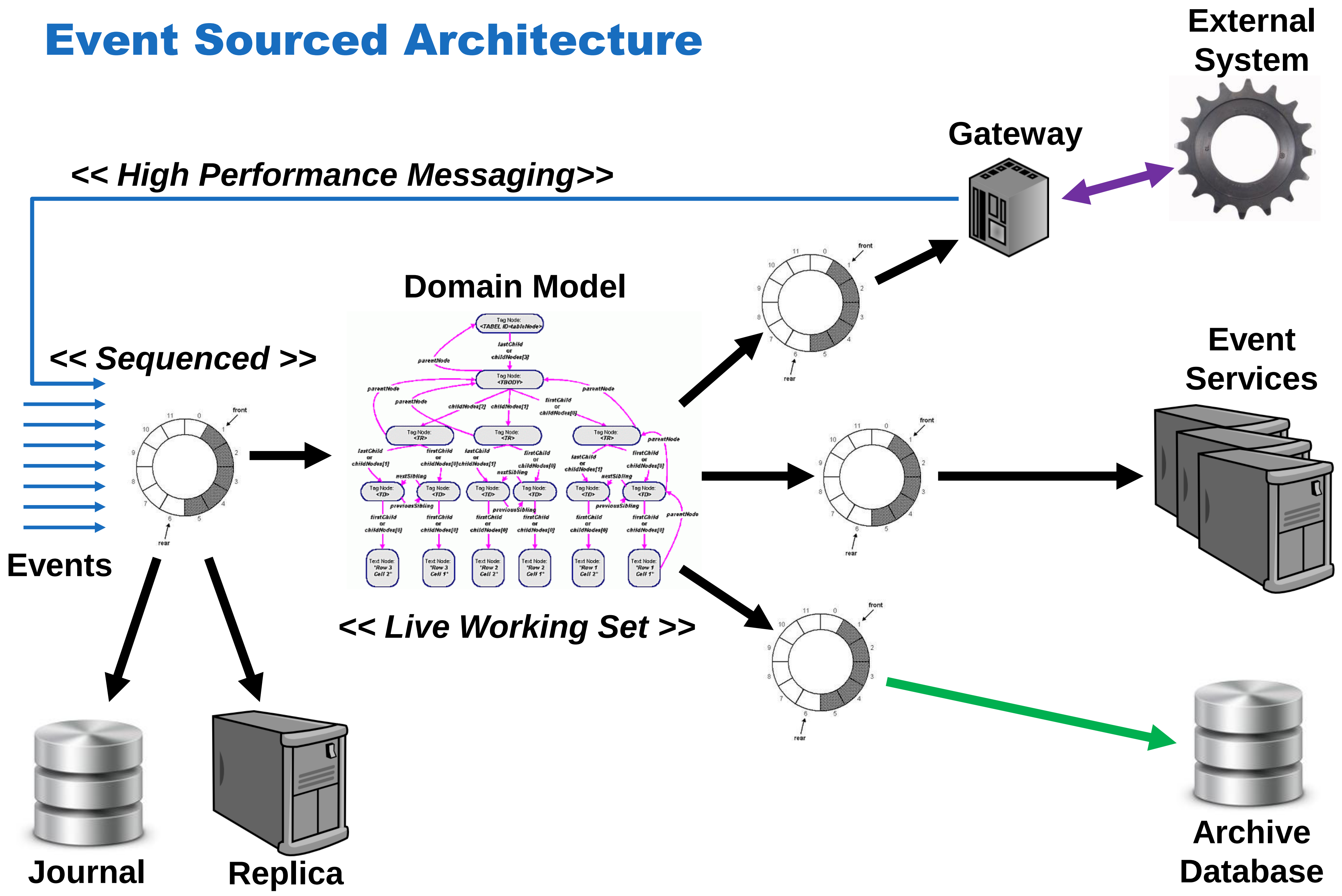
> **“*Object Prevalence*” – *Klaus Wuestefeld (2001)***
>
> **Node.js**
>
> **Nginx, G-WAN**

However the ideas have been around a long time…

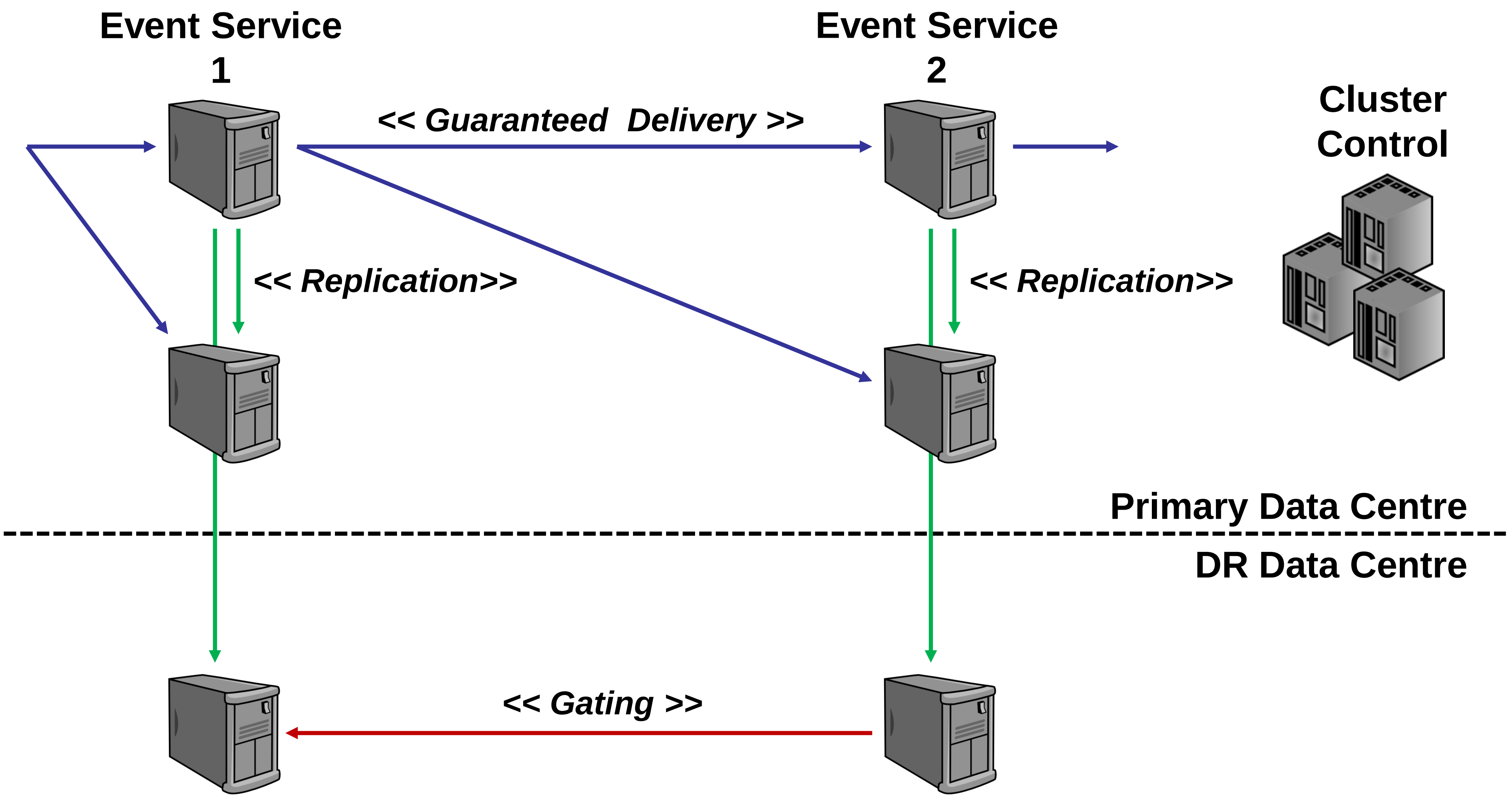
Persistence and Recovery

- Transaction Log
  - > Record input sequence of events
  - > Replay to rebuild system state on recovery
  - > Great for performance testing and debugging!
- Snapshots
  - > Used to speed up recovery
  - > Do not need to keep transaction logs forever
- Data Migration
  - > Change model when system is to be upgraded
  - > Fix data issues

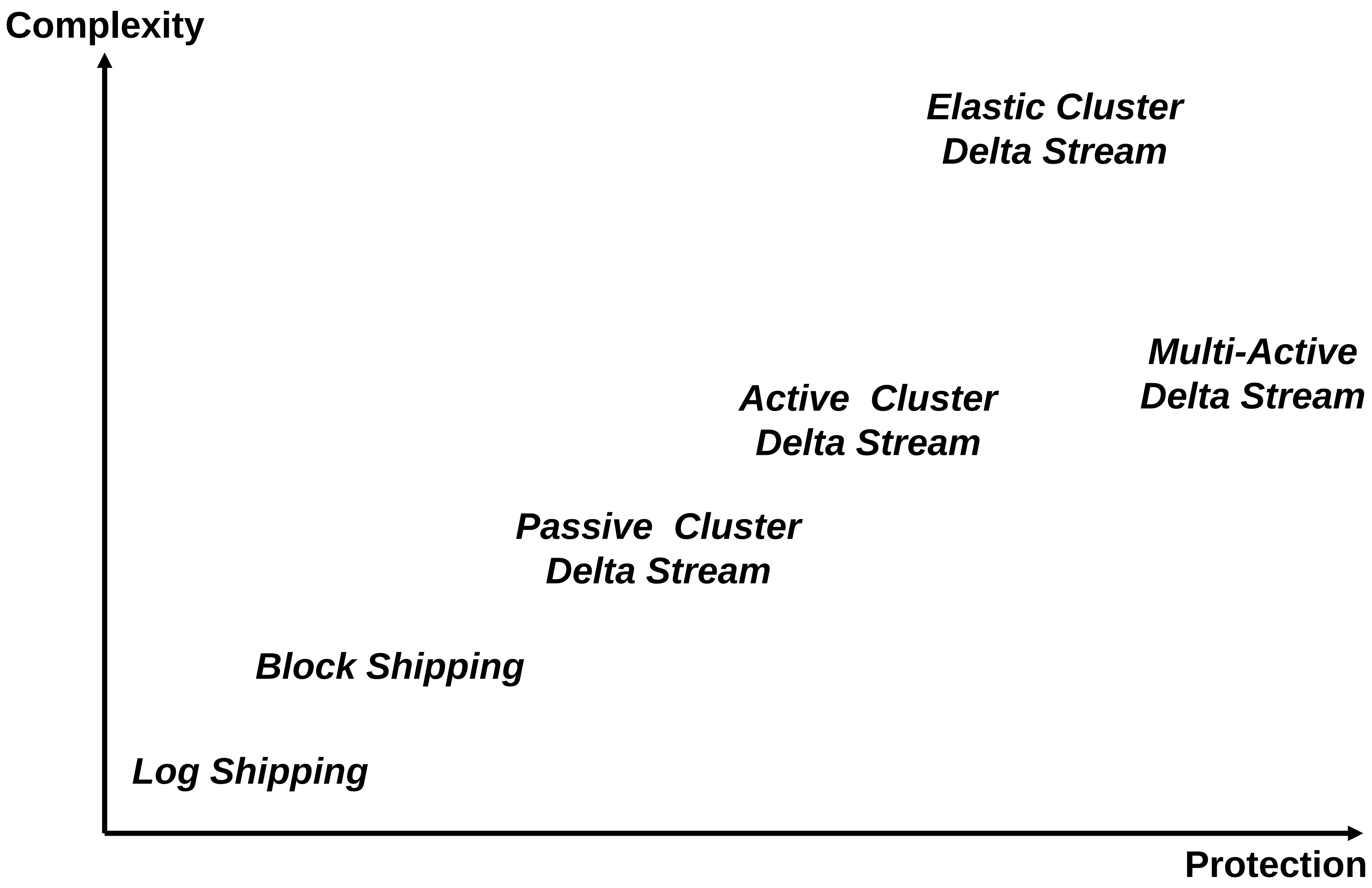
Event Sourced Architecture



HA Clusters


Event Service 1
Event Service 2
<< Guaranteed Delivery >>
Cluster Control
<< Replication>>
<< Replication>>
Primary Data Centre
DR Data Centre
<< Gating >>

Replication Models & Failure Detection


Complexity
Elastic Cluster Delta Stream
Multi-Active Delta Stream
Active Cluster Delta Stream
Passive Cluster Delta Stream
Block Shipping
Log Shipping
Protection

Importance of Design & Testing

- Unit & Acceptance Tests in CI
- Defensive argument checking
- Aggregate methods for “transactions”
- Exception handling

- Getting this stuff right is easier than concurrent programming in the business model!

- These approaches are amazing for helping you learn
  - Replay production logs for analysis and bug fixing

Scaling Event Sourced Architectures

- CQRS – Command Query Responsibility Segregation
  - Multiple read nodes/threads from same event stream
- Shards
  - People, Stuff, and Deals
  - Can partition on nodes/threads
- Complex Transactions
  - Same approach as CQRS if single shot
  - Most complex transactions are best broken down into a state model with steps

Note: In-memory asynchronous designs give great performance!

Questions?

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

Twitter: @mjpt777