FLYWHEEL
TRANSACTION PROCESSING SYSTEM

Matt Youill, Chief Technologist

ATG, Betfair

HPTS Workshop, Asilomar CA, October 2007

INDUSTRY / TRADITIONAL BOOKMAKING.

- Bets are only made between the customer and the Bookmaker.
- Bookmaker sets odds (prices) and factors in a margin.

Traditional Bookmakers are still major players in the gaming industry.

betfair

INDUSTRY / BETFAIR.

- New bookmaker Betfair appears in 2000. Defined the Betting Exchange concept.

- Customer's bets matched between themselves.
- Customers set the odds (prices).

- An exchange has "perfect" risk management and therefore, a lower margin.
- Commission on winnings.
- Prices around 20% better than Traditional Bookmakers.

INDUSTRY / BETFAIR.

- Betfair operates a betting exchange, games exchange, poker room, and casino.
- Annual revenues in excess of $300 million.
- Over 1,000,000 registered users.
- Over 1000 employees in offices globally.

- 4 billion page views/week.
- Almost half of all global traffic to gambling sites comes to Betfair.
- $4,000 deposited every minute.

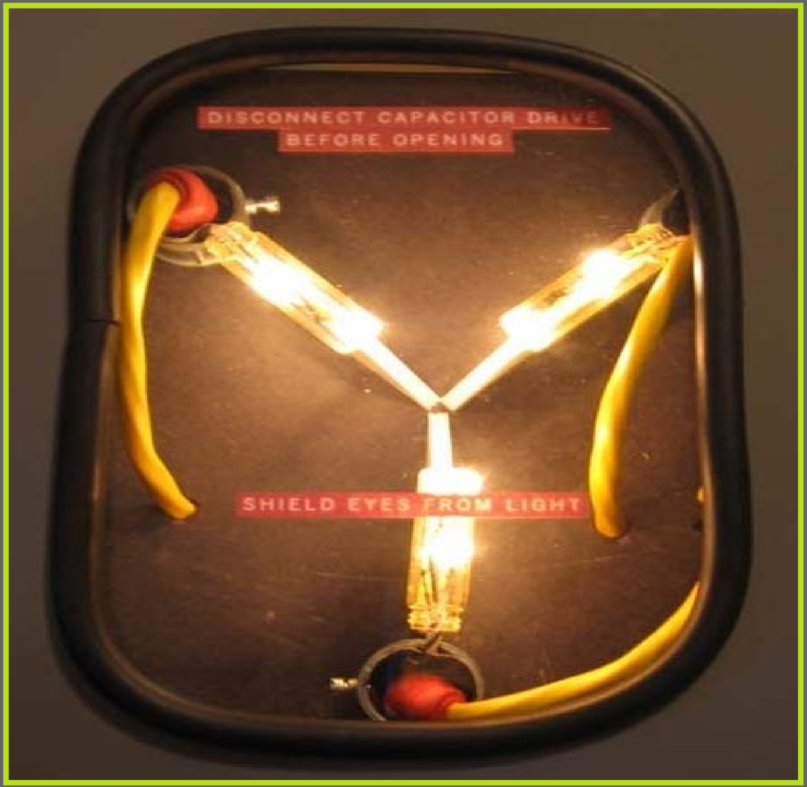
- World's leading betting exchange.

betfair

CHALLENGE / BETTING EXCHANGE.

- The exchanges have very large capacity requirements.
- Currently, up to 1000 transactions per second.
- Primary Flywheel objective of 50,000 low cost transactions per second.
- Plus increased reliability, maintainability, etc.
- A transaction = a bet placement, a cancel or an edit.

CHALLENGE / EXCHANGE TRANSACTION ENGINE.

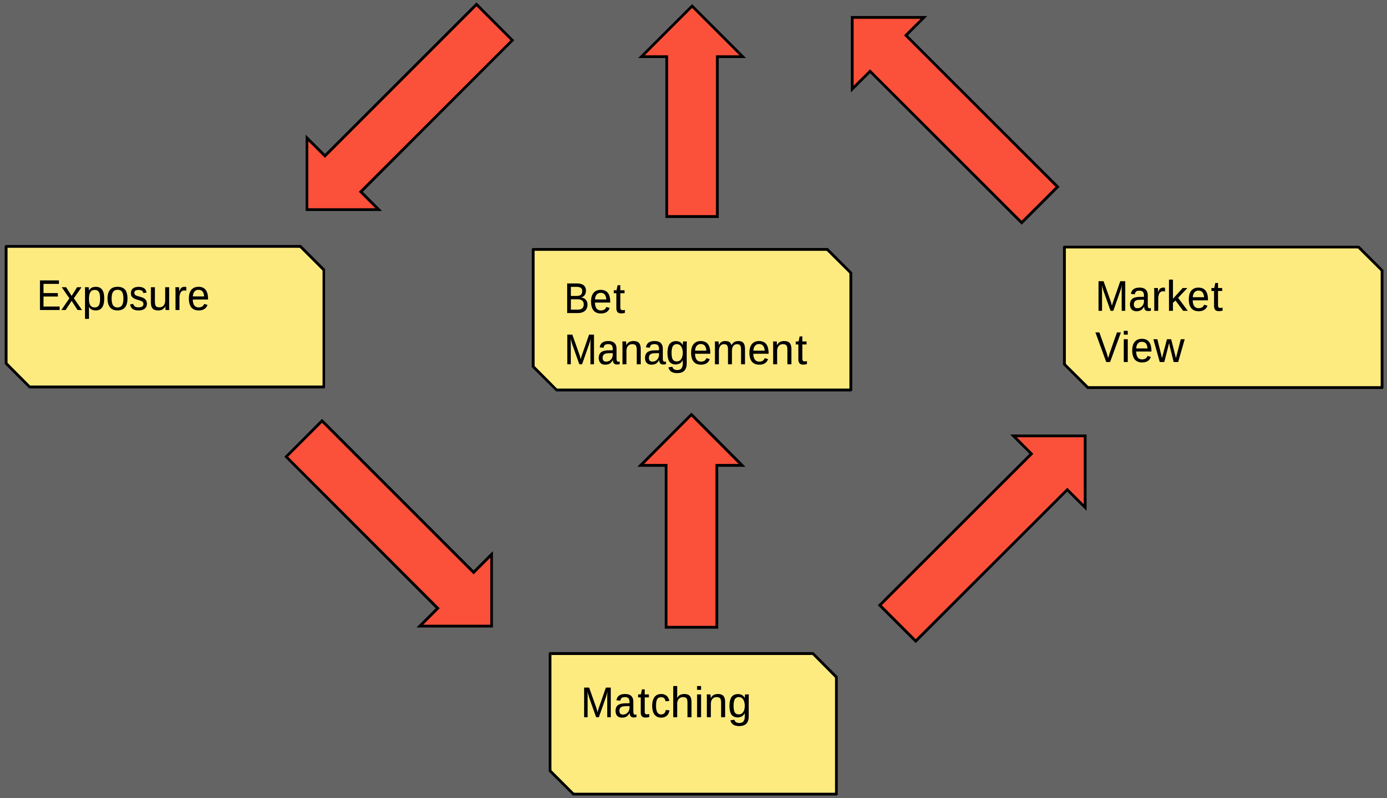


- The Betfair exchange's "flux capacitor" is the *Exchange Transaction Engine (ETE).*

- It provides 4 key functions:

- **Exposure**: Validates and stores bet orders, reserves customer funds.
- **Matching**: Matches customer's bets and reports the result.
- **Bet Management**: Allows customers to view the status of their bets.
- **Market View**: Allows customers to view a summary of all the bets placed on an event.

CHALLENGE / EXCHANGE TRANSACTION ENGINE.



CHALLENGE / EXCHANGE TRANSACTION ENGINE.

- Currently implemented in PL/SQL on a single Oracle instance.
- Rated as one of top 5 “hottest” Oracle databases in the world.

- Since Betfair was founded it has been a constant struggle to satisfy capacity demands.
- Why?... Growing pains aside, it is because Betfair’s rules of fairness present a challenge when scaling the Matching component of the ETE.

CHALLENGE / MATCHING.

- Betfair has two key business rules…

  Best Execution (Best customer value)

  Each bet placed is matched against opposing bets in order of descending odds.

  First come, First served (Fairness for all)

  The first bet placed is the first matched exclusive of others.

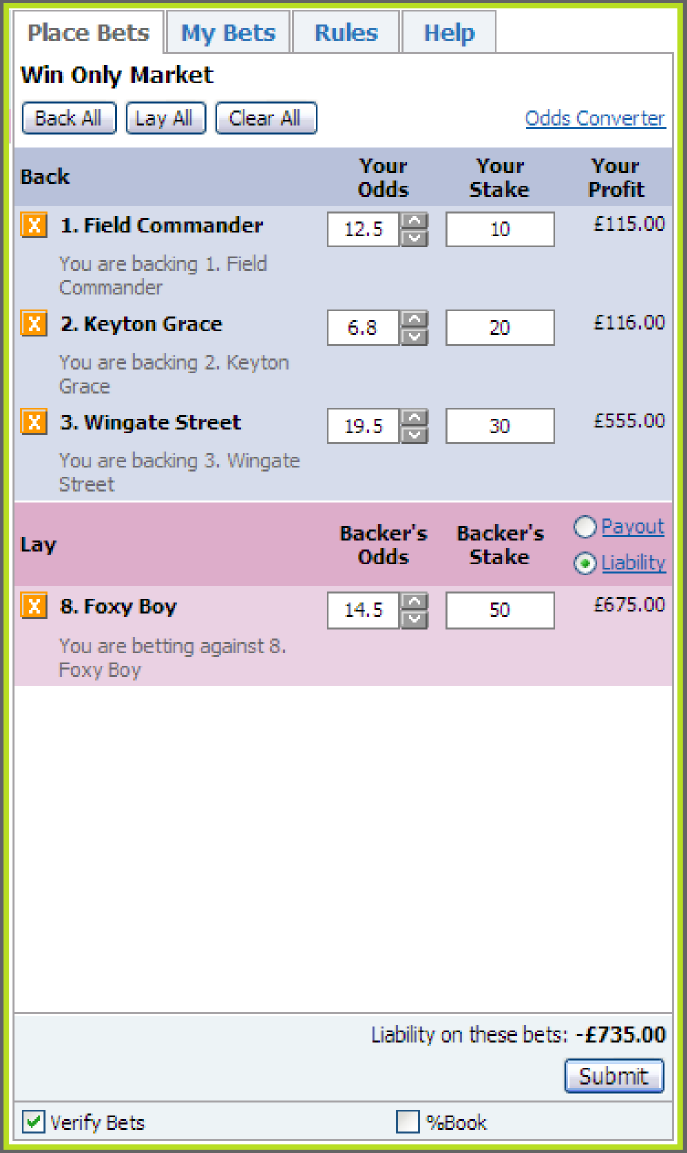
- Means everything processed serially.

- ***There is an unavoidable <u>traffic jam</u> in the business rules.***

CHALLENGE / MATCHING.

- Rules only apply per event.
- Means system can be scaled by processing each event in parallel. (Similar to the concurrent execution of trades on financial instruments).
- But…
- The nature of big events, particularly horse racing, is that they are short lived and start times rarely coincide. (Unlike the more evenly spread activity on financial exchanges).
- Over 75% of betting activity at any one time is on a single Betfair event – the “hot market”.

CHALLENGE / EXPOSURE AND BET MANAGEMENT.



Win Only Market

Back	Your Odds	Your Stake	Your Profit
1. Field Commander You are backing 1. Field Commander	12.5	10	£115.00
2. Keyton Grace You are backing 2. Keyton Grace	6.8	20	£116.00
3. Wingate Street You are backing 3. Wingate Street	19.5	30	£555.00

Lay	Backer's Odds	Backer's Stake	Payout [✓ Liability]
8. Foxy Boy You are betting against 8. Foxy Boy	14.5	50	£675.00

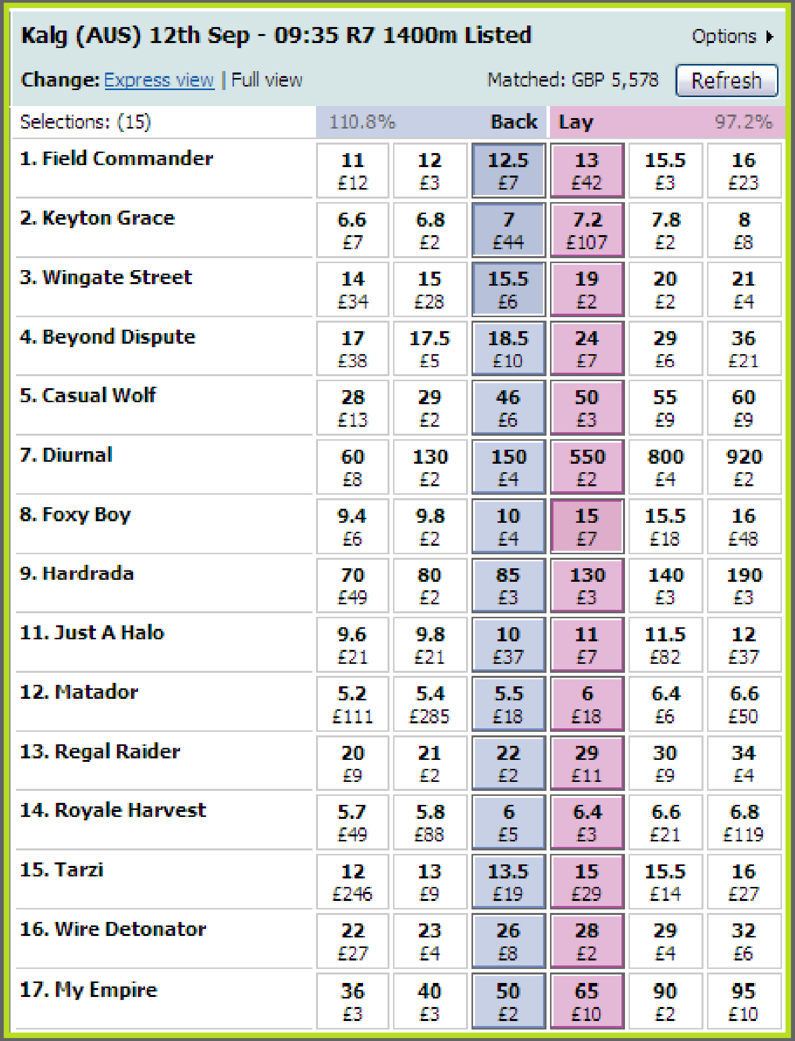
Liability on these bets: **-£735.00**

✓ Verify Bets ☐ %Book

- Exposure and bet management are less of a challenge as they occur on a per customer (account) basis.
- With activity roughly evenly spread across accounts, and peak activity on individual accounts relatively low, it's possible to simply partition by account.
- Parallelisation, partitioning and distribution provides sufficient capacity.
- Except, of course, in the case of a big individual users. But none of them are that big...yet.

betfair

CHALLENGE / MARKET VIEW.



Kalg (AUS) 12th Sep - 09:35 R7 1400m Listed Options ▸

Change: Express view | Full view Matched: GBP 5,578 Refresh

Selections: (15)	110.8%		Back	Lay		97.2%
1. Field Commander	11 £12	12 £3	12.5 £7	13 £42	15.5 £3	16 £23
2. Keyton Grace	6.6 £7	6.8 £2	7 £44	7.2 £107	7.8 £2	8 £8
3. Wingate Street	14 £34	15 £28	15.5 £6	19 £2	20 £2	21 £4
4. Beyond Dispute	17 £38	17.5 £5	18.5 £10	24 £7	29 £6	36 £21
5. Casual Wolf	28 £13	29 £2	46 £6	50 £3	55 £9	60 £9
7. Diurnal	60 £8	130 £2	150 £4	550 £2	800 £4	920 £2
8. Foxy Boy	9.4 £6	9.8 £2	10 £4	15 £7	15.5 £18	16 £48
9. Hardrada	70 £49	80 £2	85 £3	130 £3	140 £3	190 £3
11. Just A Halo	9.6 £21	9.8 £21	10 £37	11 £7	11.5 £82	12 £37
12. Matador	5.2 £111	5.4 £285	5.5 £18	6 £18	6.4 £6	6.6 £50
13. Regal Raider	20 £9	21 £2	22 £2	29 £11	30 £9	34 £4
14. Royale Harvest	5.7 £49	5.8 £88	6 £5	6.4 £3	6.6 £21	6.8 £119
15. Tarzi	12 £246	13 £9	13.5 £19	15 £29	15.5 £14	16 £27
16. Wire Detonator	22 £27	23 £4	26 £8	28 £2	29 £4	32 £6
17. My Empire	36 £3	40 £3	50 £2	65 £10	90 £2	95 £10

- The Market View or the view of the odds and stakes currently available on an event is challenging but in a different way.

- Every transaction will potentially change the view of a market.
- Hence, every transaction requires the new view to be delivered to customers.

- This is a lot of information that needs to go to a lot of people many times every second.
- Multicasting scales effectively but in the world of the web it is harder requiring manual polling, AJAX, Comet, etc.

APPROACH / 100X PROGRAMME.

"100 times (100X) more transactions per second"

"Infinite capacity at zero cost"

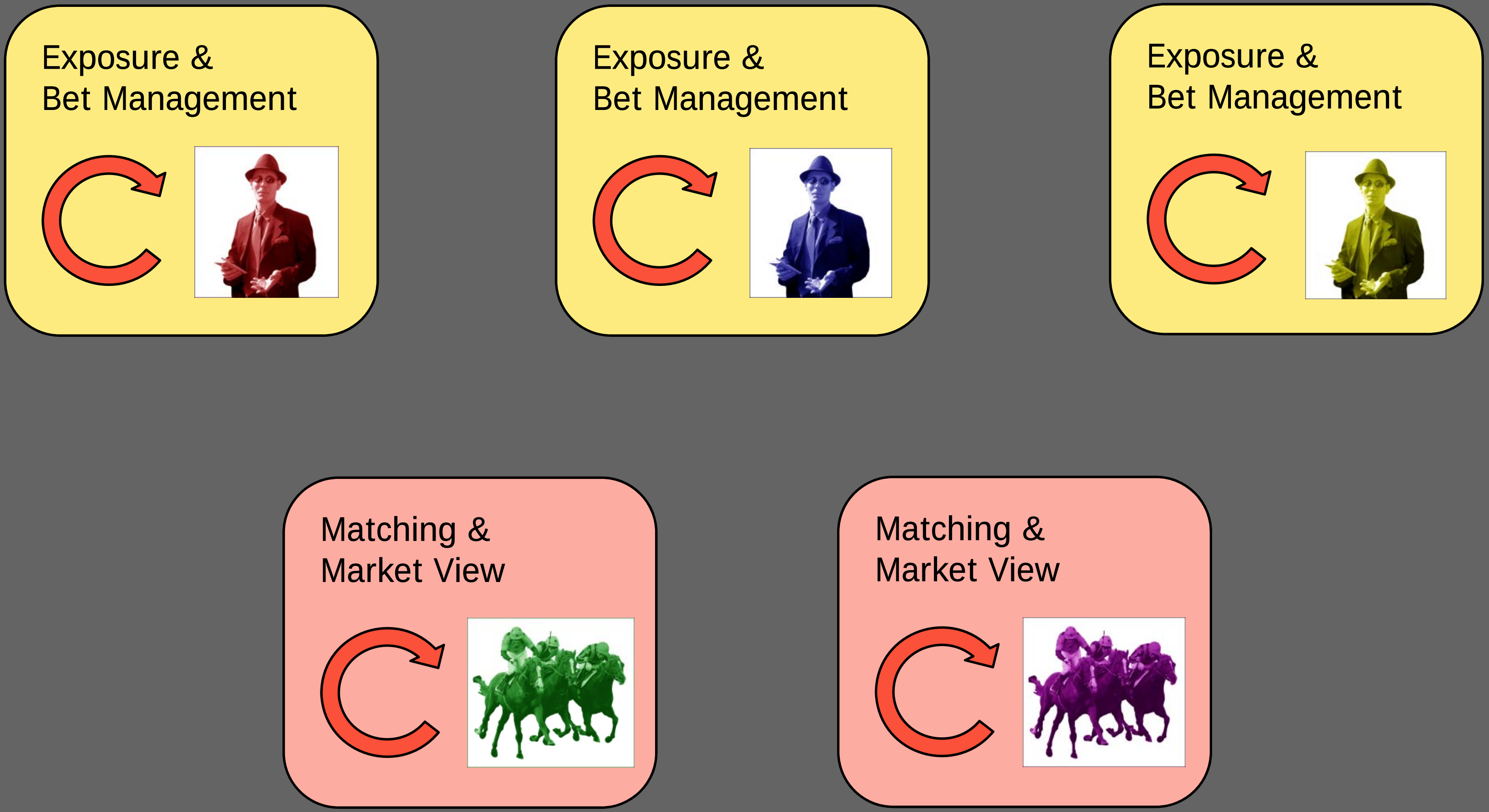
- At the time of initiation (500 TPS -> 50,000 TPS)
- Challenge existed for many years, well defined. Solution less so ☺.

Phases	
Investigation	Had this problem or anything like it been solved before?
Research & Design	Sketch and elaborate various architectures.
Proof of Concept	Multiple Vendors including Betfair Engineering. Build, test and compare.
Integration	How to get into production? What changes and compromises needed?
Build and Deploy	Build the "Lite" version and plug it into the live system.

SOLUTION / CONCURRENCY (ISOLATION).

- Each of the major functions in the ETE require exclusive access to either an account or event.
- Rather than client sessions acquiring and releasing locks on particular pieces of data, each instance of an entity is assigned to a single execution unit (i.e. Actor). Unit A may own Account 123, or unit B may own Market 456 for example.
- With execution units tied to individual entities, access is inherently serialised and “transactions” are isolated from one another.

SOLUTION / CONCURRENCY (ISOLATION).



SOLUTION / CONCURRENCY (ISOLATION).

Exposure &
Bet Management

Account

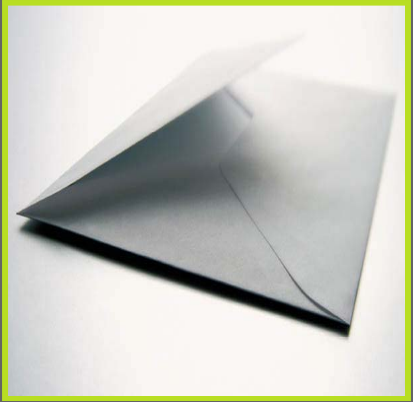
Account ID	Funds	Exposure
1	$1,000	$735

Bets

Bet ID	Type	Runner	Odds	Stake
1	Bet For	Field Commander	12.5	$10
2	Bet For	Wingate Street	6.8	$20
3	Bet For	Keyton Grace	19.5	$30
4	Bet Against	Foxy Boxy	14.5	$50

- Only one execution unit can work on this data. It is “owned” by the Actor.
- No interleaved/inconsistent updates.

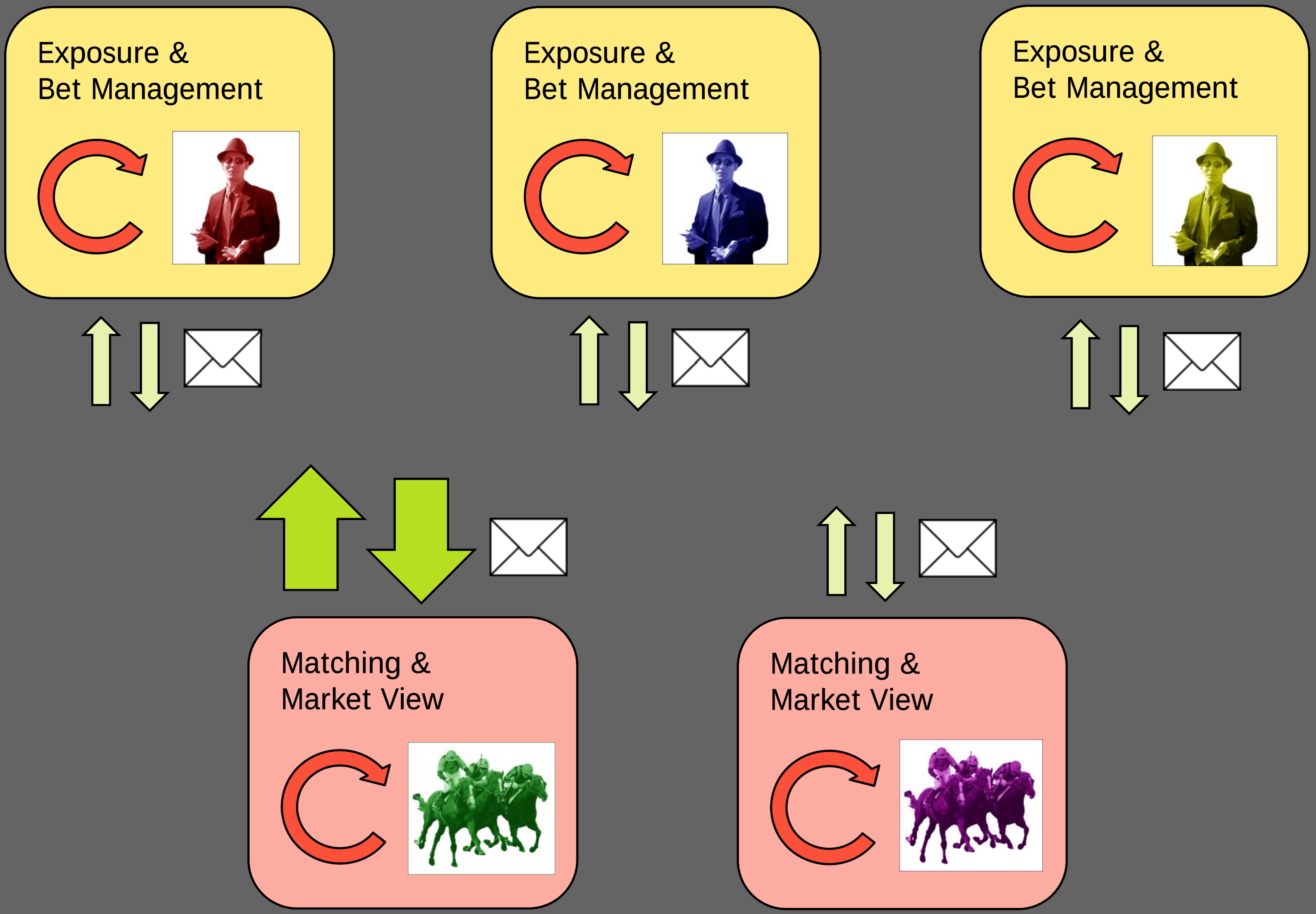
SOLUTION / COMMUNICATIONS.



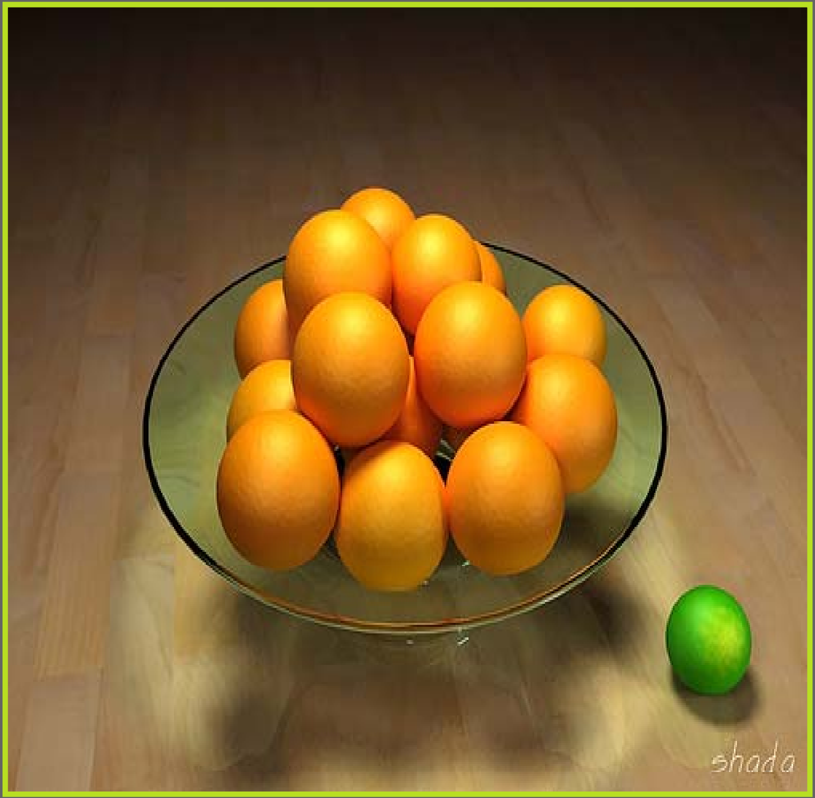
- Each Actor operates on the data it owns as a result of messages (operations) sent to it.
- Each Actor may in turn send more messages to further execution units.
- Message passing has its benefits, but also its costs...

Can distribute widely, conceptually simple concurrency model, threads can be detached from sessions, messages can be batched, easier to take advantage of asynchronous IO, etc.	But messages can be lost, arrive out of order, duplicated, corrupted, need correlation, no system wide consistency or atomicity, enqueue/dequeue costs, etc.

SOLUTION / COMMUNICATIONS.



SOLUTION / CONSISTENCY.



- No Actor in the system has a complete and consistent view of the entire system and there is no strict integrity across units.
- Each only responsible for its individual view of the world. These chunks of reality define the boundaries of consistency.
- Ultimately full system wide consistency is desirable but at any instance in time, global consistency may vary. That is - global consistency is <u>weak</u> and achieved only <u>eventually</u>.
- When something inconsistent is found, explicit correcting actions need to be taken.

SOLUTION / CONSISTENCY.

Exposure & Bet Management

Bets

Bet ID	Type	Runner	Odds	Stake
1	Bet For	Field Commander	12.5	$10

Matches

Bet ID	Matched Bet ID	Matched Odds	Matched Stake
1	5	12.5	$5

Matching & Market View

Unmatched Bets

Bet ID	Type	Runner	Odds	Unmatched Stake
1	Bet For	Field Commander	12.5	$5

Example: These must add up - eventually

- Data exists under two Actors control, but no explicit consistency.
- Don't become inconsistent in the first place. State and operations must not be lost, corrupted, duplicated, etc. But if something does, take explicit compensating action.

SOLUTION / ATOMICITY.

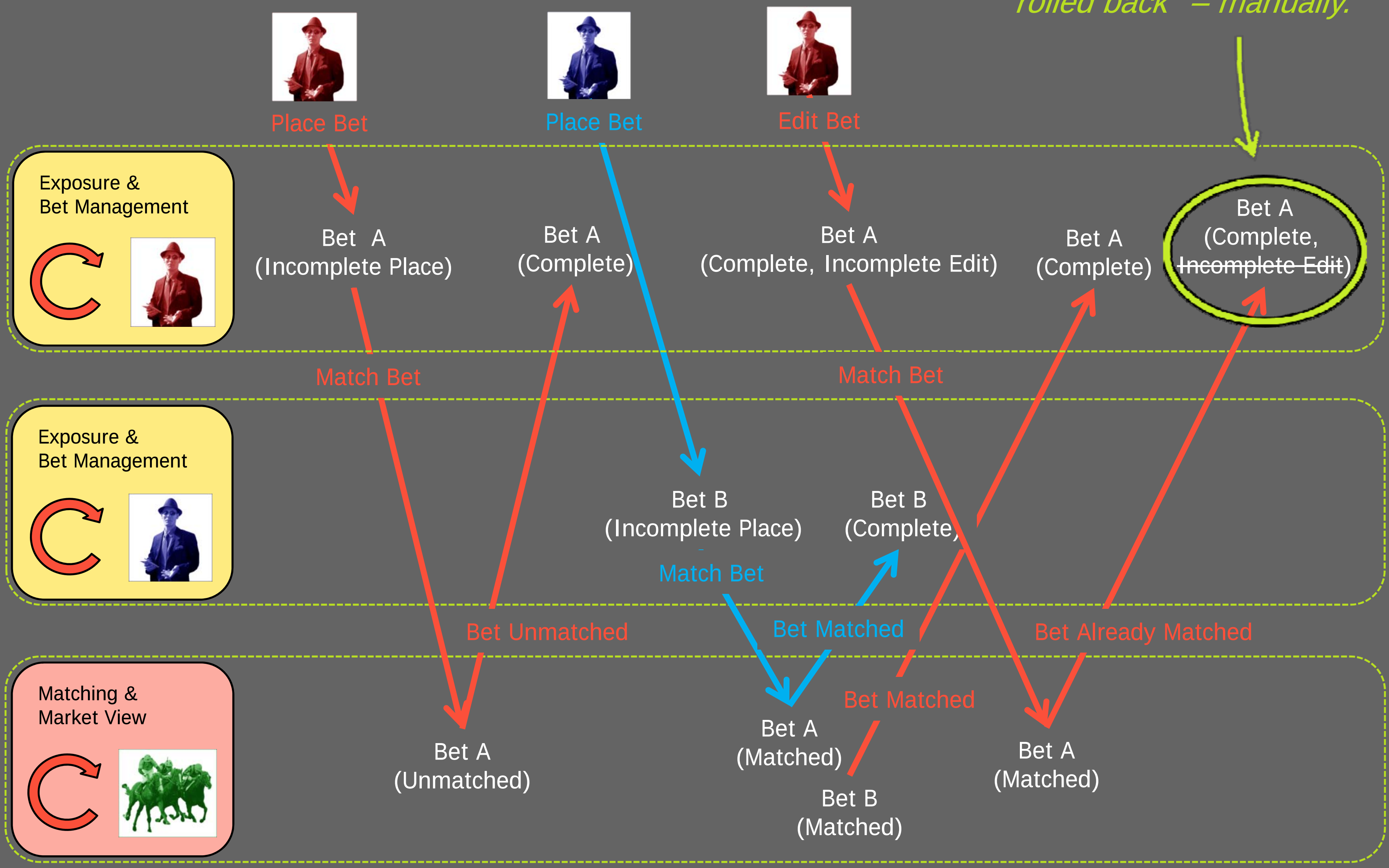
- Each operation sent to an Actor defines the scope of a transaction. The instructions that the operation performs is defined by the deterministic function that it executes.

- Receipt and journaling of that operation defines the success of a transaction. The instructions need not be executed, only captured.

- In most case, transactions across Actors require explicit compensating logic.
- Would be nice to have one Actor and avoid spanning transactions. Not practical though.

SOLUTION / ATOMICITY.

The edit can't complete, the edit work done so far must be "rolled back" – manually.



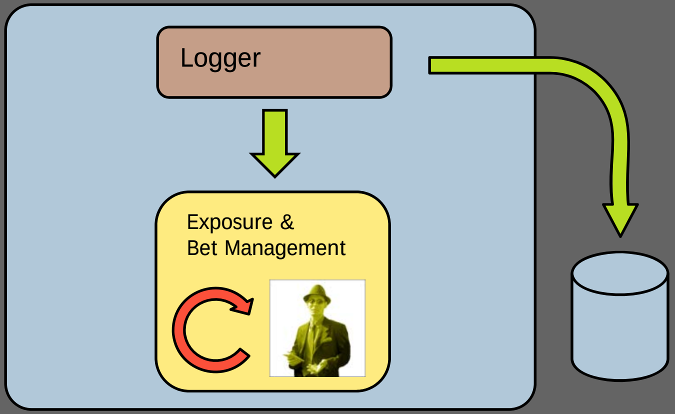
betfair

SOLUTION / WHEN THINGS GO WRONG.

- Highly distributed architecture allows for partial failures and disruptions.
- Large chunks of the system can fail and the system as a whole will keep running.
- Durable state and message reliability means system stabilises eventually in the event of a failure.

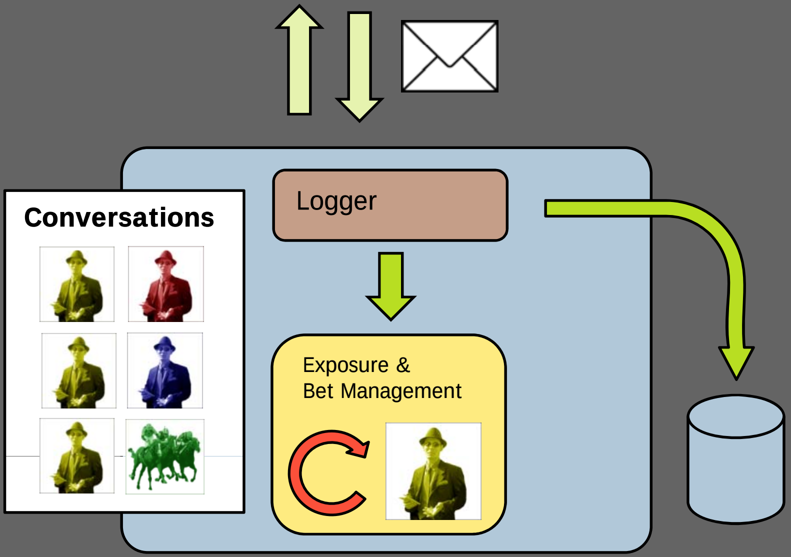
Design isn't overly fussy. "Simple" design important.

SOLUTION / RELIABILITY (STATE).



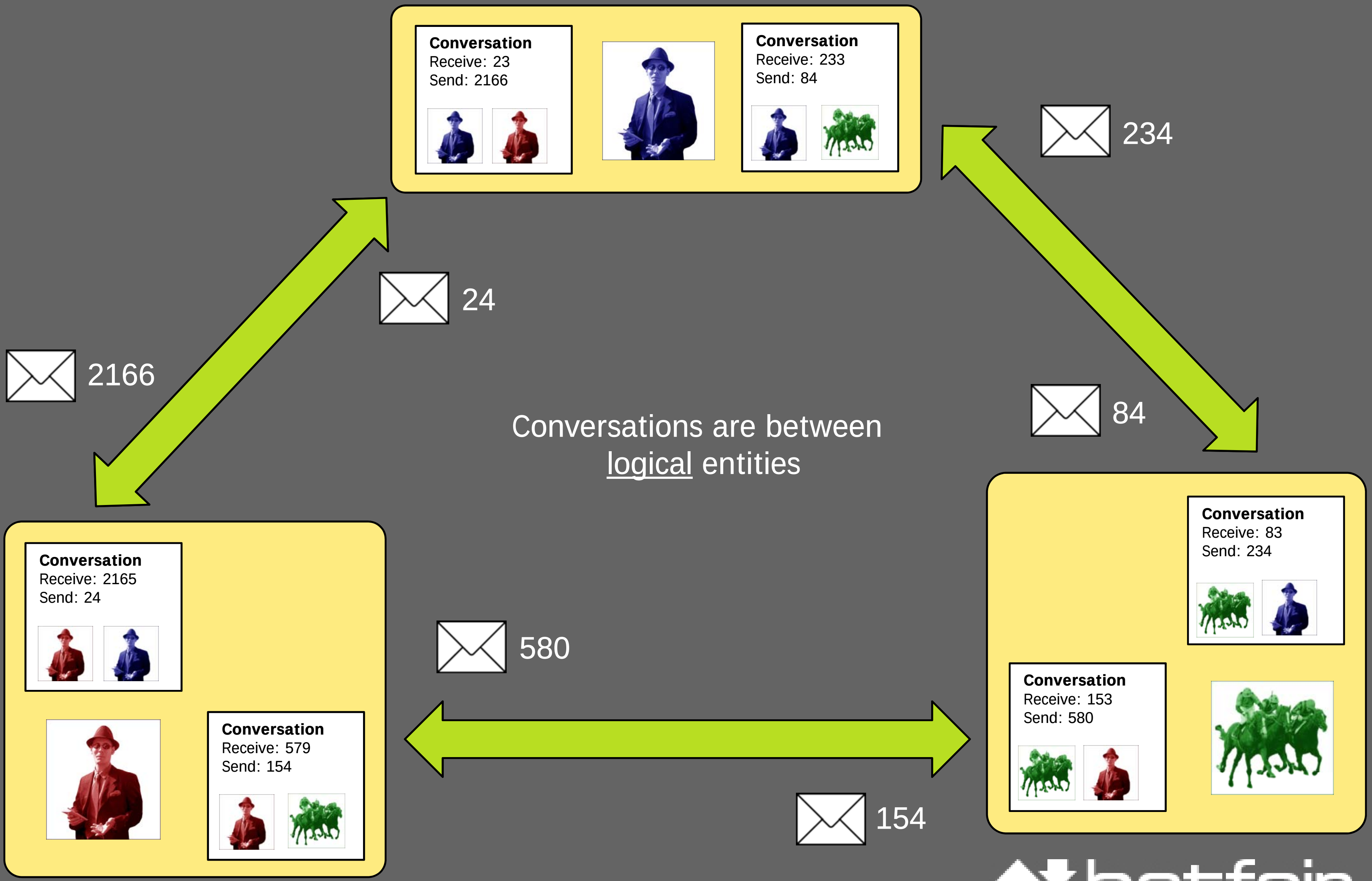
- Actor's inbox turned into a journal. State persistence provided by logging all messages.
- Replay of journal restores state. Simple ☺
- All functions following the log must be deterministic so under replay everything is restored to exactly how it was.
- Disk shared between nodes in the event of a full node failure.
- Adding check pointing reduces replay time and prevents log exhaustion.

SOLUTION / RELIABILITY (MESSAGING).



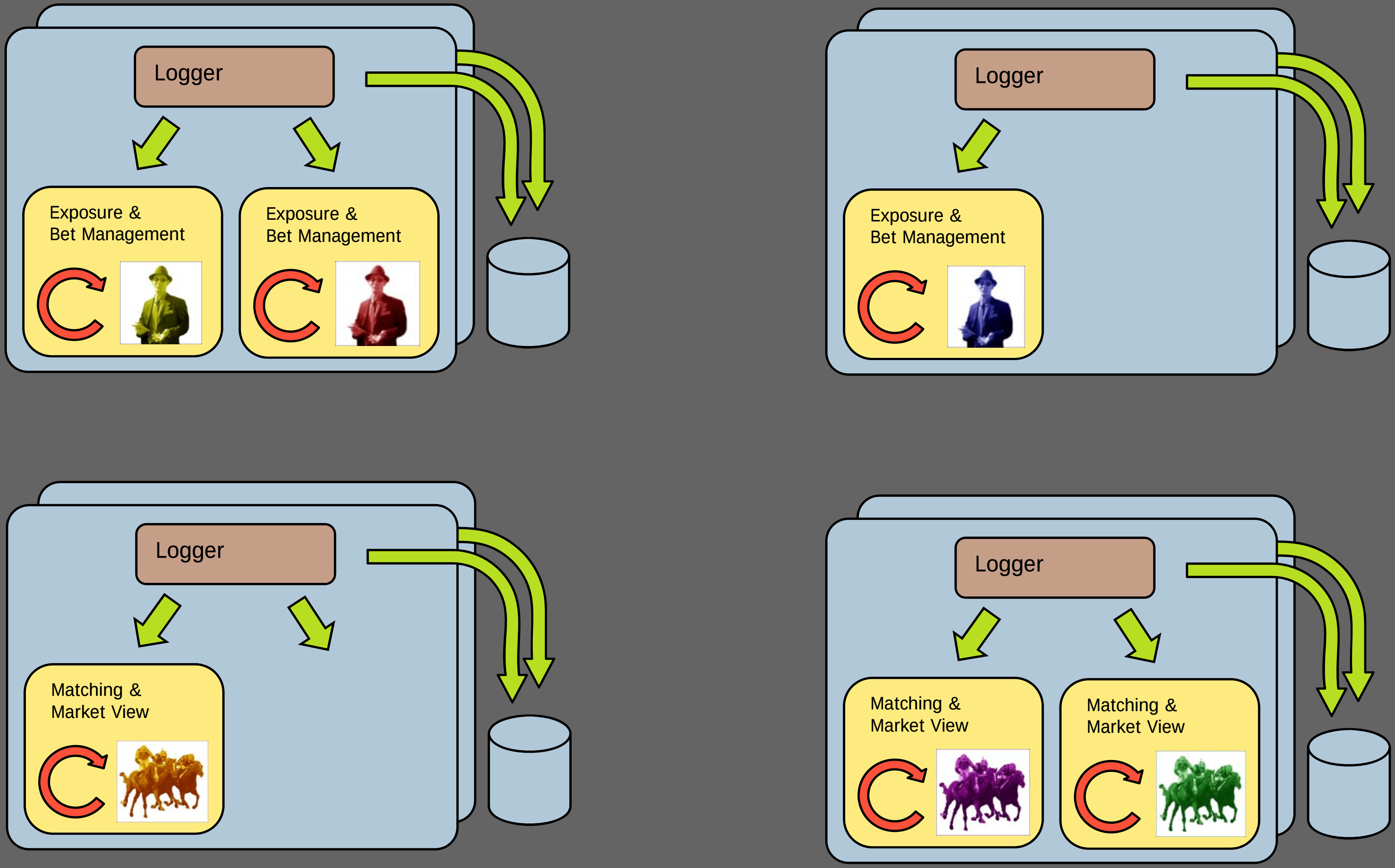
- The log also serves as the basis for reliable messaging.
- Journal replay also restores the state of message counters along with application state.
- These message counters (or conversations) are maintained for each sending and receiving pair.
- Using the counters, lost messages can be detected, duplicate messages can be removed and in doubt messages can be retried safely.

SOLUTION / RELIABILITY (MESSAGING).



Conversations are between <u>logical</u> entities

SOLUTION / DEPLOYMENT.



SOLUTION / HIGH AVAILABILITY.

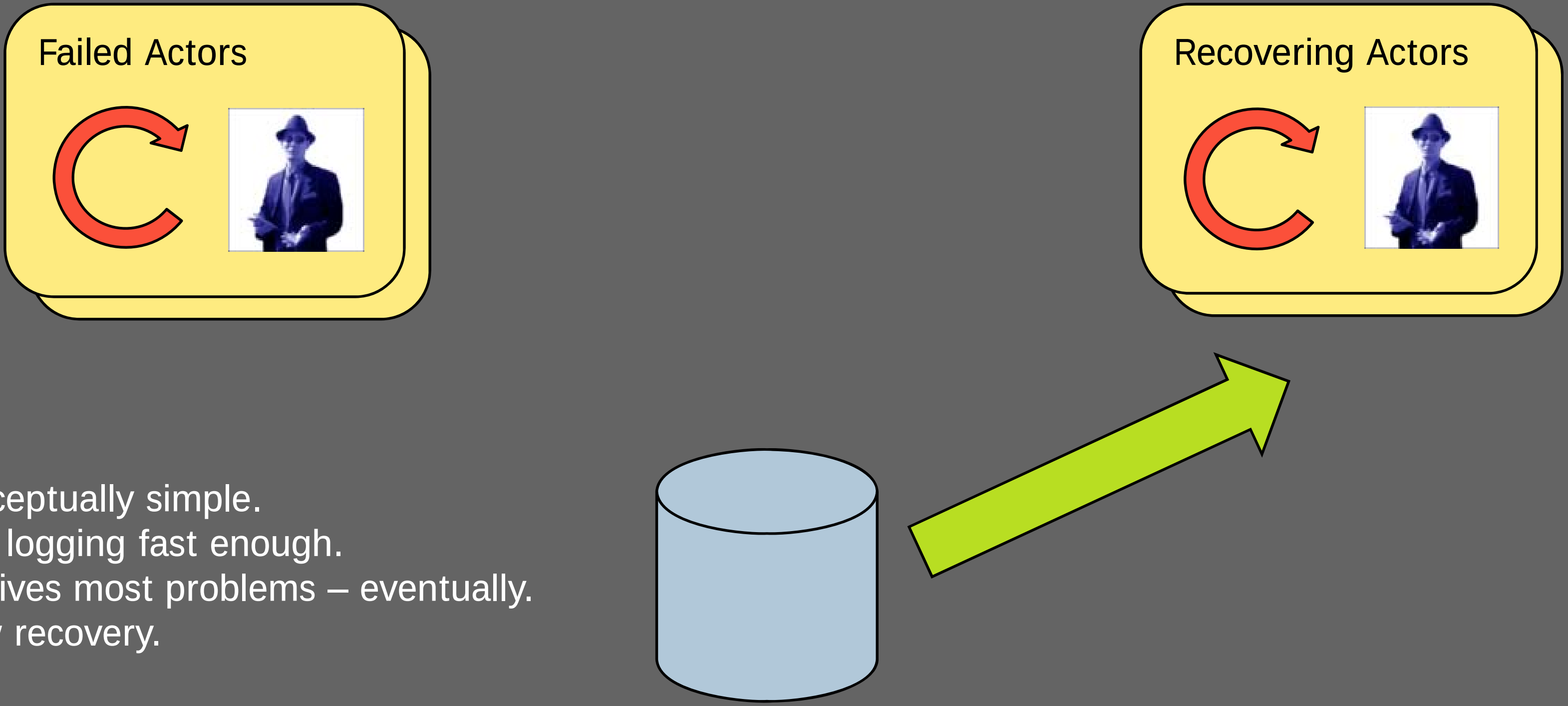
- Log replay means faults are tolerated… slowly.
- Need fast(er) recovery.

- Two flavours of HA – catastrophic and fast.
- Need a solution for both.
- We need a back up ready to go, but if both “disappear” then all is not lost.

- Lots of different ways to do this, but no perfect solution.

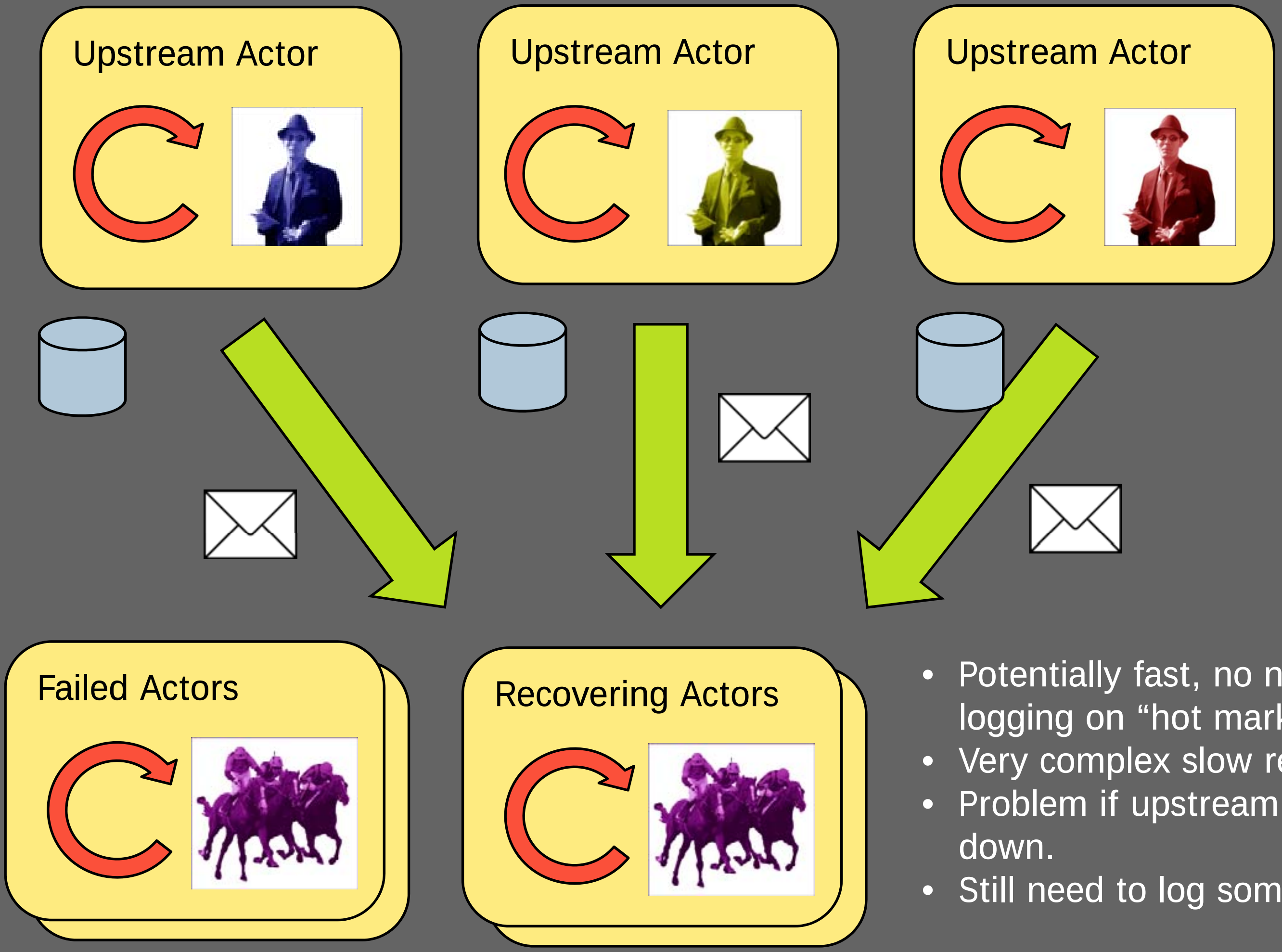
- We tried various ways – about 8. Mostly well established methods.

SOLUTION / CATASTROPHIC FAILURES (1. LOG REPLAY).



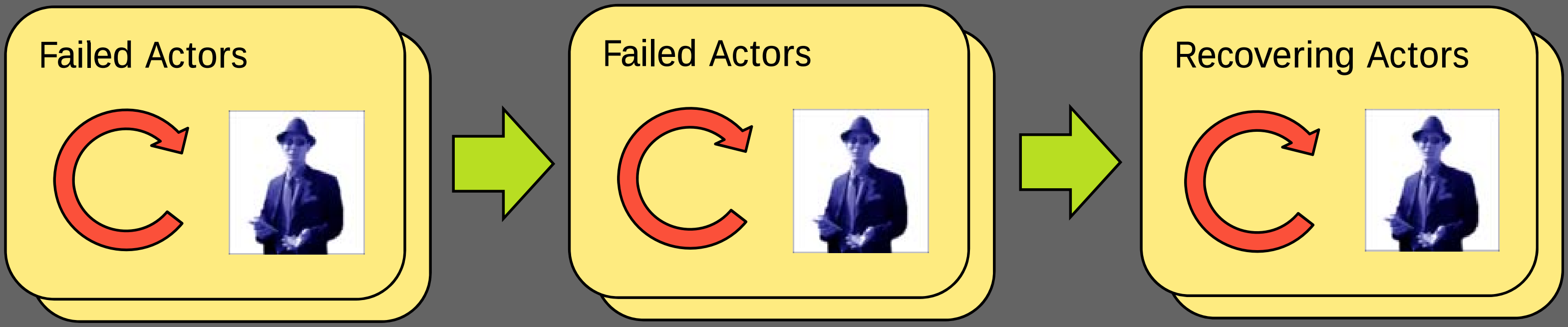
- Conceptually simple.
- Disk logging fast enough.
- Survives most problems – eventually.
- Slow recovery.

SOLUTION / CATASTROPHIC FAILURES (2. UPSTREAM REPLAY).



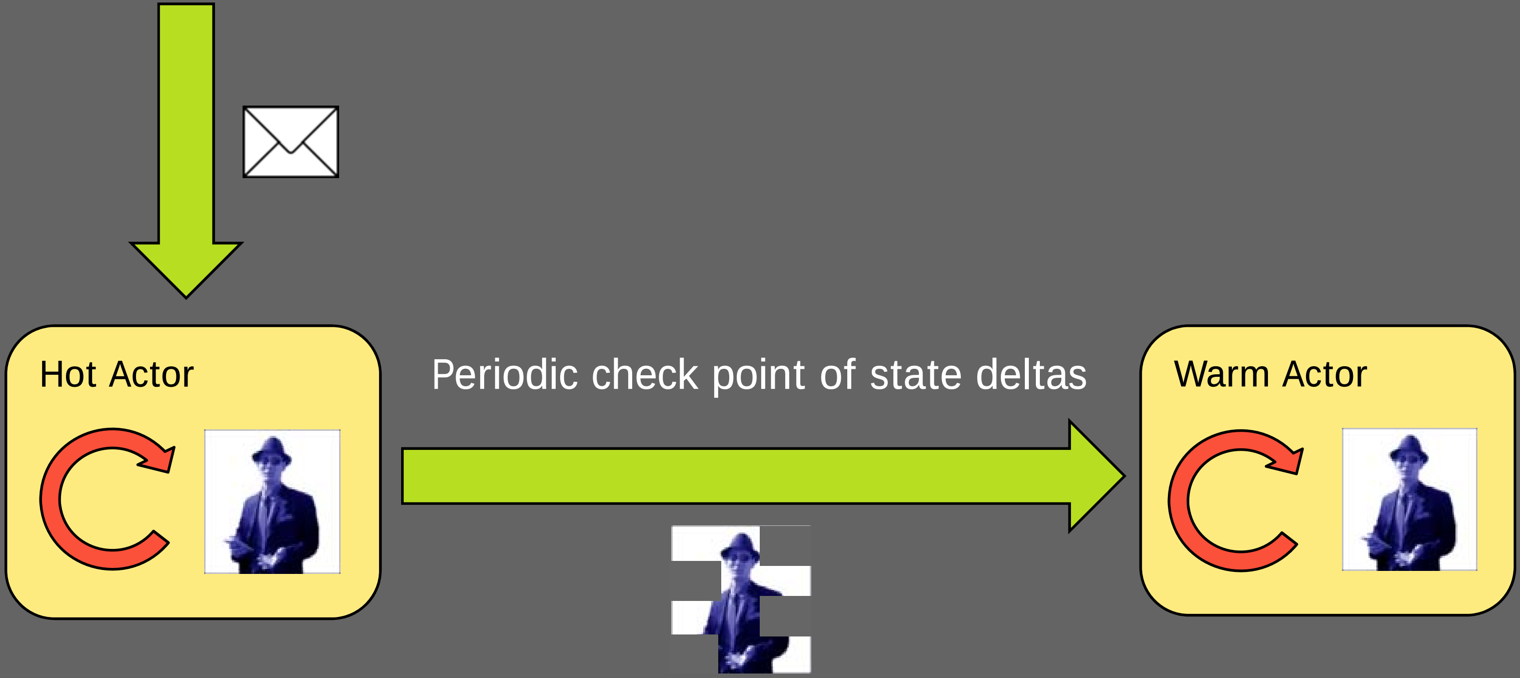
- Potentially fast, no need for slow logging on “hot market” units.
- Very complex slow recovery.
- Problem if upstream actor is also down.
- Still need to log somewhere.

SOLUTION / CATASTROPHIC FAILURES (3. MULTI-ORDER MIRRORING).



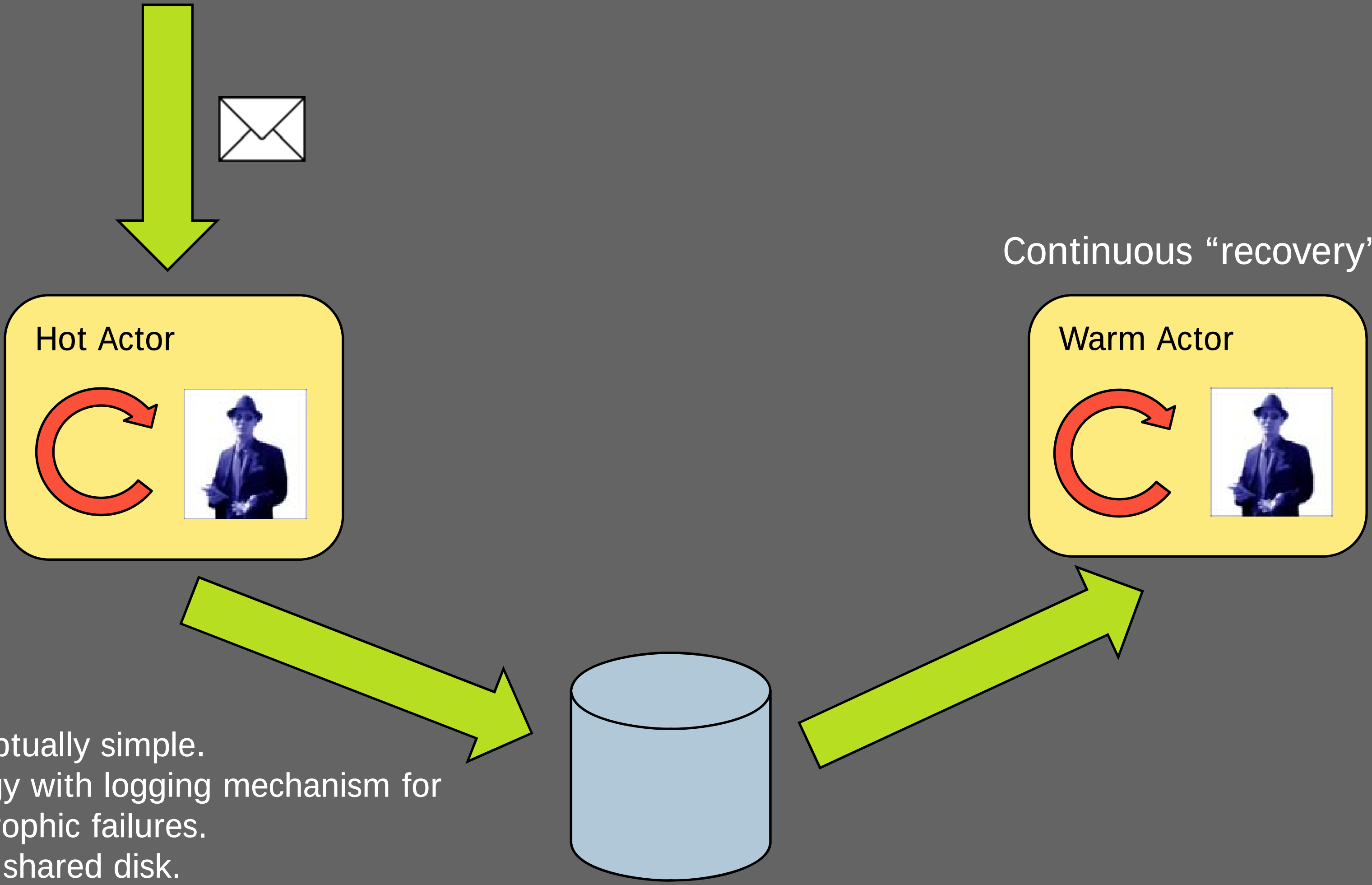
- Appears simple, but can be tricky.
- Slow.
- Won’t survive high multi-order failures.

SOLUTION / FAST FAILURES (4. STATE MIRRORING).



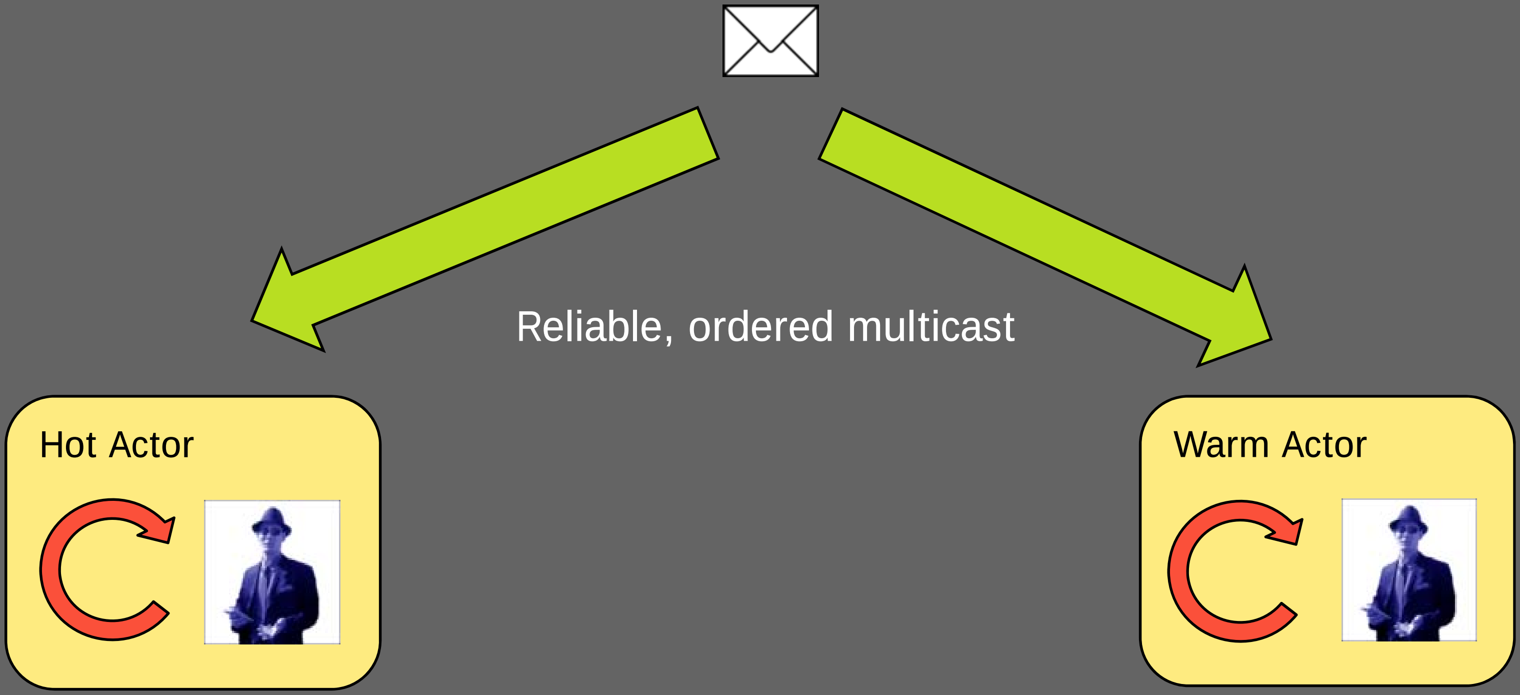
- Appears simple.
- Somewhat slow.
- Reintroducing a failed node can be tricky.
- Synergy with logging (check pointing) mechanism for catastrophic failures.

SOLUTION / FAST FAILURES (5. LOG TAILING).



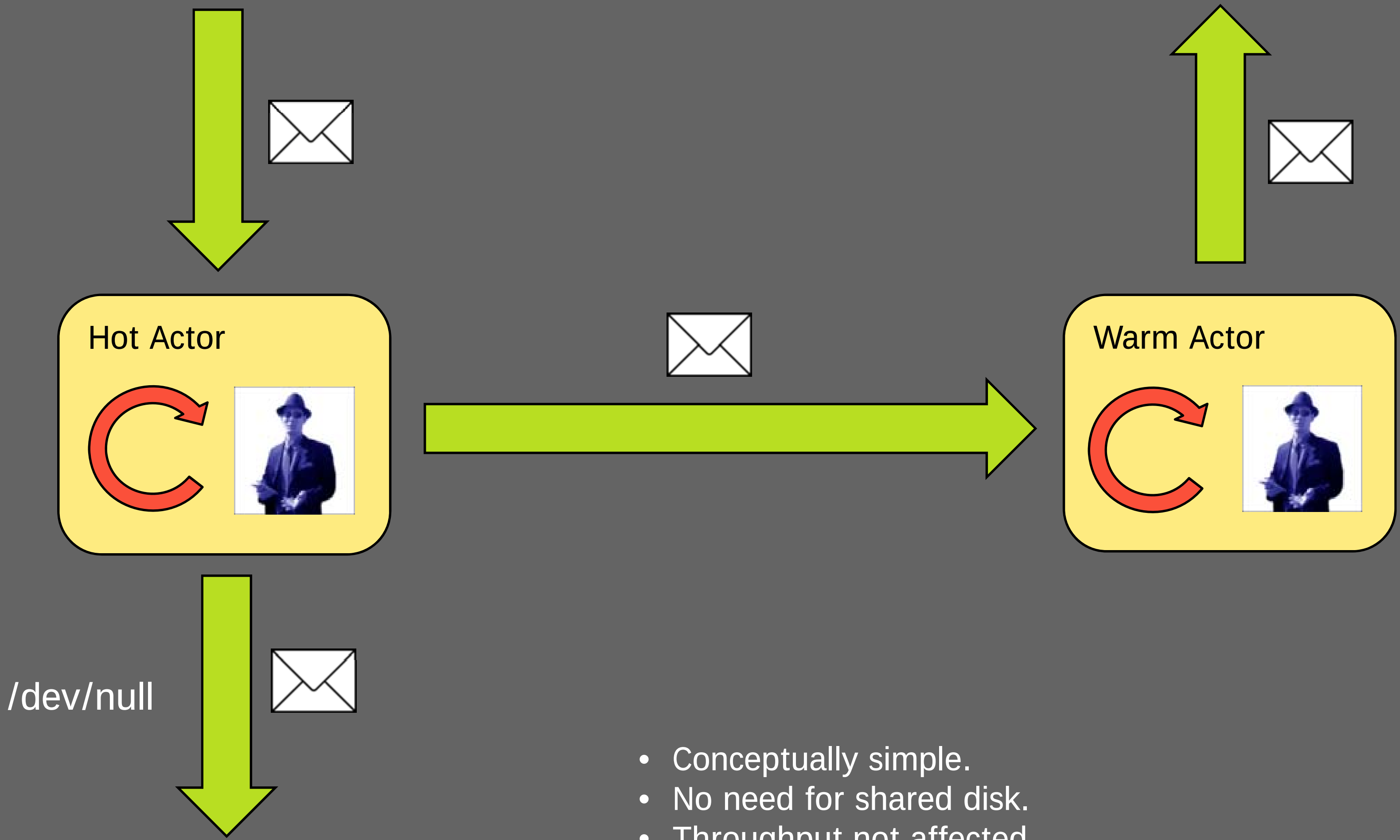
- Conceptually simple.
- Synergy with logging mechanism for catastrophic failures.
- Needs shared disk.

SOLUTION / FAST FAILURES (6. SIMULTANEOUS DELIVERY).



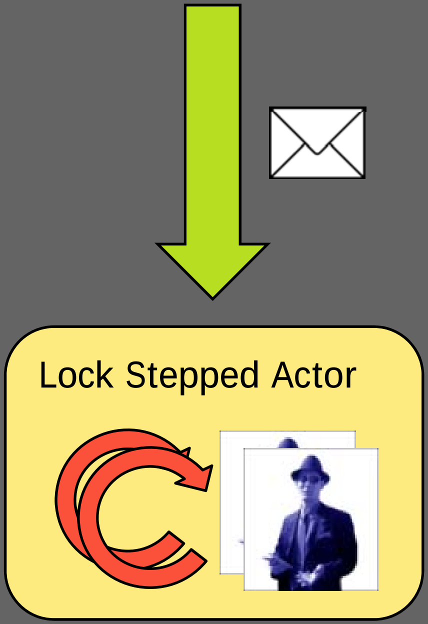
- Slow and tricky.
- No need for shared disk.

SOLUTION / FAST FAILURES (7. RELAYING).



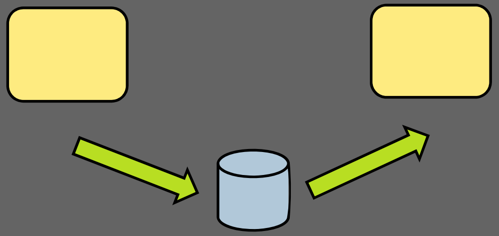
- Conceptually simple.
- No need for shared disk.
- Throughput not affected.
- Though increases latency.

SOLUTION / FAST FAILURES (8. LOCK STEPPING).



- Off the shelf solution (NonStop, Stratus, Qumranet, etc).
- Specialised.

SOLUTION / LOG TAILING



- Best compromise.
- Simple to understand and implement.
- Catches both catastrophic and fast failure requirements.

RESULTS.

- 70K TPS (hot market throughput test at Sun Solution Centre, Manchester, England).
  - Single biggest cost, by far, was serialization on and off network and disk.
- Design patent pending.
- Winner of CNET UK Technology Awards.

Results				
Load Injectors	"Account Controllers"	"Market Controllers"	Throughput	Latency
2	2	1	29,950 TPS	250 ms
3	3	1	73,370 TPS	550 ms
6	5	2	136,150 TPS	679 ms

Machines Specification:
2 x Dual Core 2Ghz AMD Opteron Processor
8GB Memory
Copper Gbit Network
Red Hat Enterprise Linux 4 O/S
Java 1.5 VM
Sun 3510 FC Array Storage

- "Lite" (Production) version currently in testing.
  - Scaled up instead of out and integrated with "classic" Oracle ETE.
  - Step towards "Heavy".

betfair

FURTHER WORK.

- Geographic Separation.
  - Issues when running across continents largely solved.
  - Latency issue remains. Faster than light travel? (eep!)
    - Problem for fairness.
    - Problem for DR.

- Generality.
  - Can the architecture be generalised and applied to more problem domains across Betfair. Ignoring the betting, it's just a reliable state machine.

More importantly: Build and deploy to production – happening now.

END