Working with MIG

mobile interactive group | a velti company

- Our robust technology has been used by major broadcasters and media clients for over 7 years
- Voting, Polling and Real-time Interactivity through second screen solutions
- Incremental revenue generating services integrated with TV productions
- Facilitate 10,000+ interactions per second as standard across our platforms
- Platform and services have been audited by Deloitte and other compliant bodies
- High capacity throughput for interactions, voting and transactions on a global scale
- Partner of choice for BBC, ITV, Channel 5, SKY, MTV, Endemol, Fremantle and more:

BBC itv CHANNEL 5 sky TELECINCO 7 MTV MUSIC TELEVISION O2

THE X FACTOR FREMANTLEMEDIA endemol SONY vodafone

QCon

mVoy Products

mobile interactive group | a velti company

High volume mobile messaging campaigns & mobile payments

Social Interactivity & Voting via Facebook, iPhone, Android & Web

Create, build, host & manage mobile commerce, mobile sites & apps

Interactive messaging & multi-step marketing campaigns

QCon

MIG Technologies

mobile interactive group | a velti company

- Erlang
- RIAK & leveldb
- Redis
- Ubuntu

- Ruby on Rails
- Java
- Node.js
- MongoDB
- MySQL

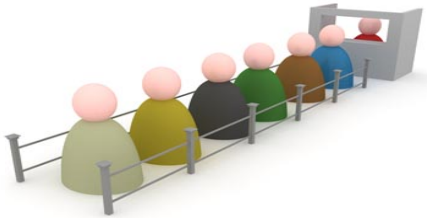
QCon

Battle Stories

- Building a wallet

- Optimizing your hardware stack

- Building a robust queue



Building a wallet

mobile interactive group | a velti company

- Fast
  - Over 10,000 debits / sec (votes)
  - Over 1,000 credits / sec

- Scalable
  - Double hardware == Double performance

- Robust / Recoverable
  - Transactions can not be lost
  - Wallet balances recoverable in the event of multi-server failure

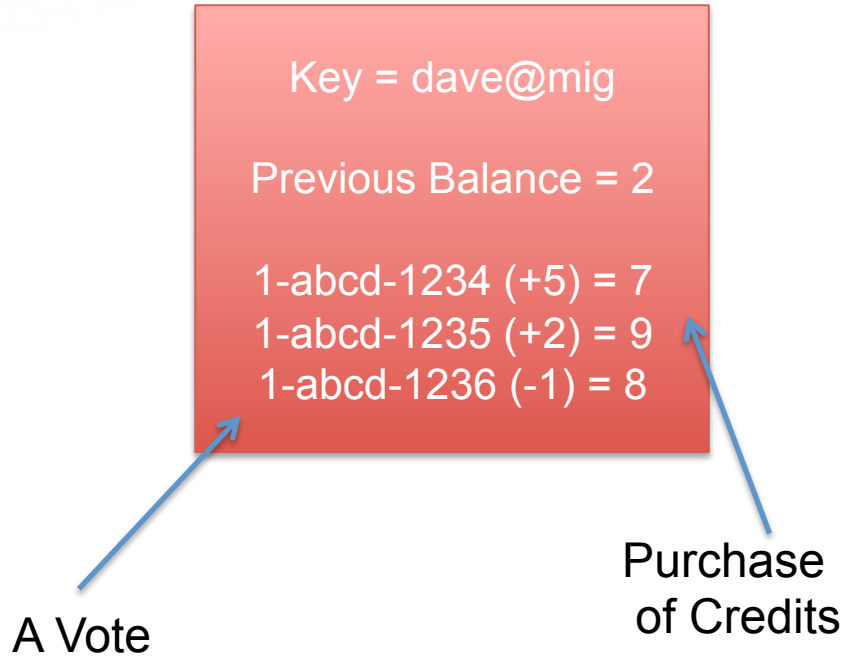
- Auditable
  - Complete transaction history

QCon

Building a wallet - attempt #1

mobile interactive group | a velti company

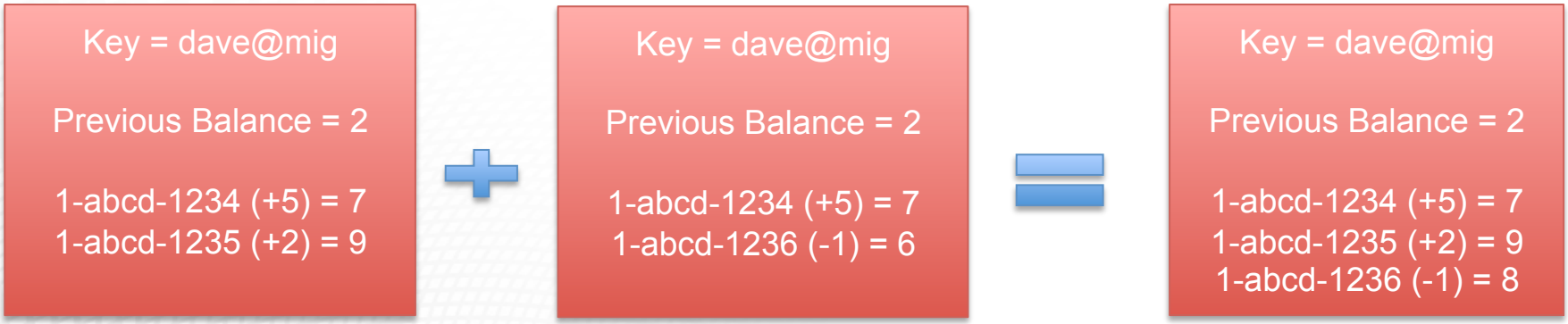
- Use RIAK Only
  - Keep things simple
  - Less moving parts
- A wallet per user containing:
  - Previous Balance
  - Transactions with unique IDs
  - Rolling Balance
  - Credits (facebook / itunes)
  - Debits (votes)



QCon

Building a wallet - attempt #1

- RIAK = Eventual Consistency
  - In the event of siblings
  - Deterministic due to unique transactions ID's
  - Merge the documents and store



Building a wallet - attempt #1

- Compacting the wallet
  - Periodically
  - In event it grows to large

Key = dave@mig

Previous Balance = 2

1-abcd-1234 (+5) = 7
1-abcd-1235 (+2) = 9
1-abcd-1236 (-1) = 8
…
1-abcd-9999 (+1) = 78

Compactor

Key = dave@mig

Previous Balance = 78

Building a wallet - attempt #1

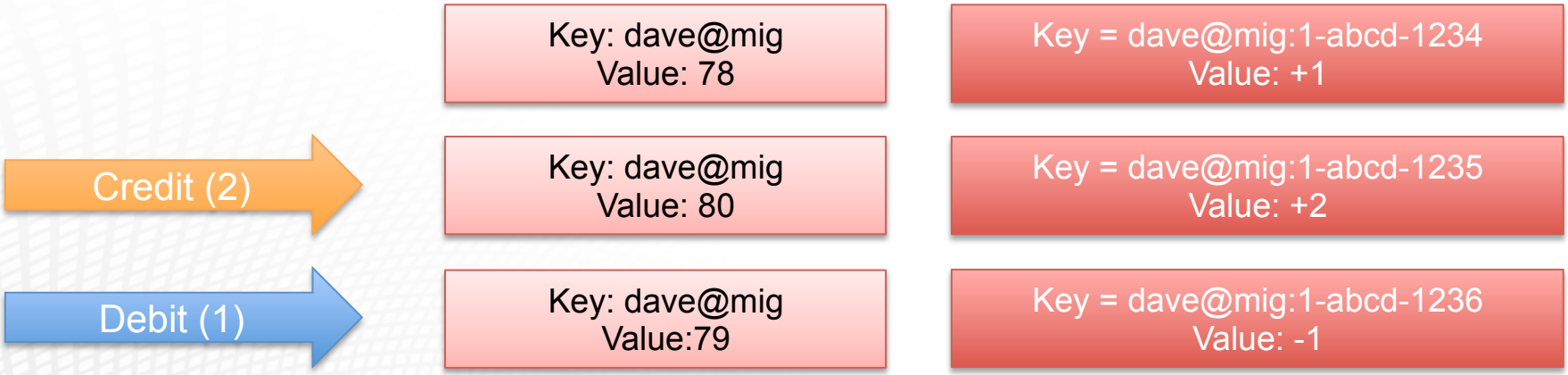
mobile interactive group | a velti company

- Our experiences
  - Open to abuse
  - As wallet grows, performance decreases
  - Risk of sibling explosion
  - User can go over drawn

QCon

Building a wallet - attempt #2

- Introduce REDIS
  - REDIS stores the balance
  - RIAK stores individual transactions



Building a wallet - attempt #2

mobile interactive group | a velti company

- Keeping it all in sync
  - Periodically compare REDIS and RIAK
- Disaster Recovery
  - Rebuild all balances in REDIS
  - Using transactions from RIAK

QCon

Building a wallet - attempt #2

mobile interactive group | a velti company

- Our experiences
  - It works
  - Fast 10,000 votes / sec (6 x HP DL385)
  - Used wallet recovery (Data Center Power Fail)
- The future
  - Possible use of levelDB backend for RIAK
  - Faster wallet recovery

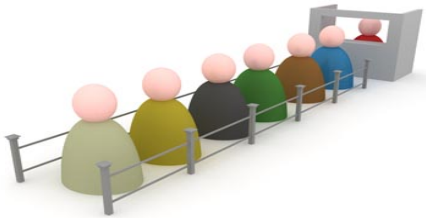
QCon

Battle Stories

- Building a wallet

- Optimizing your hardware stack

- Building a robust queue



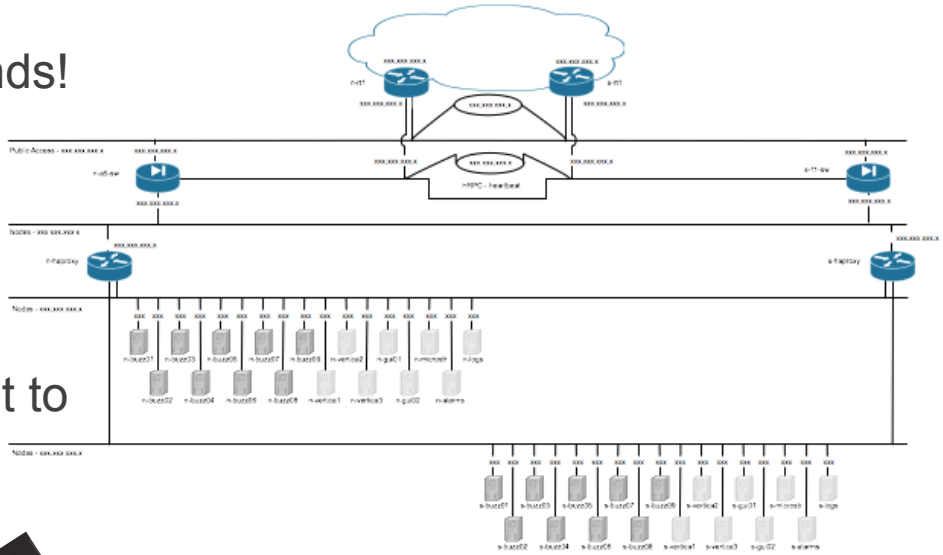
Hardware optimisation

- Observed ‘time outs’
  App ⇔ RIAK DB
- Developed sophisticated balancing mechanisms to code around them, but they still occurred
- Especially under load

Photograph and Logo © 2010 Time Out Group Ltd.

Nature of the problem

- Delayed responses of up to 60 seconds!
- Our live environment contains:
  - 2 x 9 App & RIAK Nodes
  - HP DL385 G6
  - 2 x AMD Opteron 2431 (6 cores)
- We built a dedicated test environment to get to the bottom of this:
  - 3 x App & RIAK Nodes
  - 2 x Intel Xeon (8 cores)



Looking for contention…

Contention options

- CPU

Less than 60% utilisation

- Disk IO

?

- Network IO

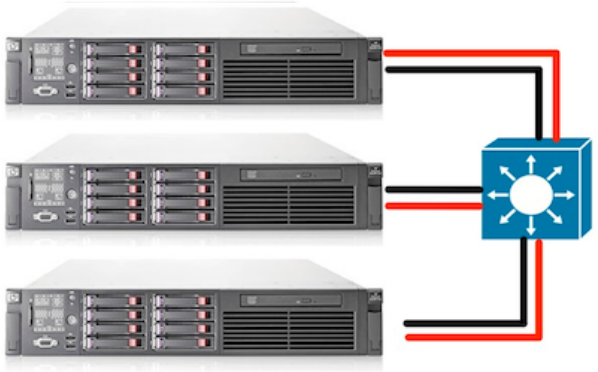
?

Disk I/O contention?

- Got SSD drives (10 x access speed)
- Three independent makes
- RIAK data directory = SSD
- Logs, OS, etc = HDD

Network I/O contention?

- RIAK cluster is I/O hungry
- Wired up second NICs
- Dedicated RIAK VLAN
- Keep apps traffic on other VLAN



Divorce!

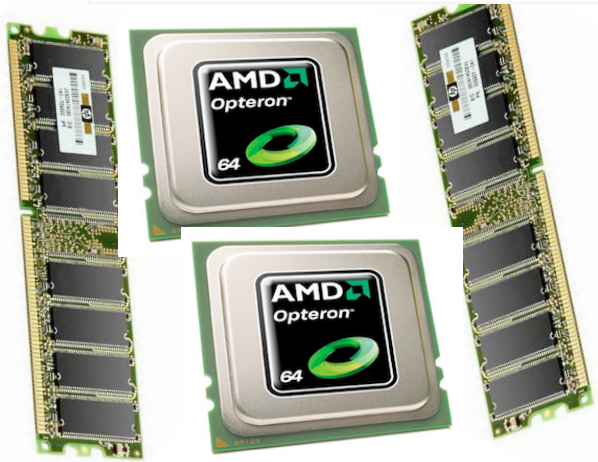
- Bought two more servers
- Separated Apps & RIAK DB
- APP = Two nodes
- RIAK = Three nodes

We could have stopped there…

Memory contention / NUMA

mobile interactive group | a velti company

- Looking at the 60% again
  - ***Non-Uniform Memory Access*** *(**NUMA**) is a computer memory design used in Multiprocessing, where the memory access time depends on the memory location relative to a processor.* - Wikipedia
- In the 1960s CPUs became faster then memory
- Race for larger cache memory
- Cache algorithms
- Multi processors accessing the same memory leads to contention and significant performance impact
- Dedicate memory to processors/cores/threads
- BUT, - most memory data is required by more then one process. => ccNUMA
- Linux threading allocation is challenged
- Cache-coherence attracts significant overheads, especially for processes in quick succession!



QCon

Gain control! - NUMACTL

mobile interactive group | a velti company

- Processor affinity – Binds a particular process type to a specific processor
- Instruct memory usage to use different banks
- For example: numactl --cpunodebind 1 –interleave all erl
- Get it here: apt-get install numactl

- => No timeouts
- => 20%+ speed increase when running App & RIAK
- => Full use of existing hardware

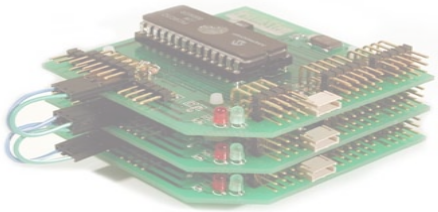
QCon

Battle Stories

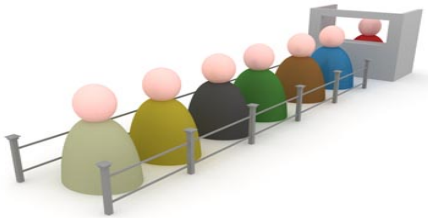
mobile interactive group | a velti company

- Building a wallet

- Optimizing your hardware stack



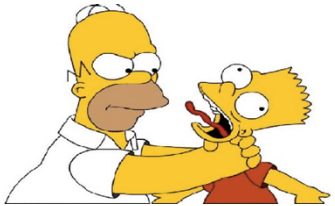
- Building a robust queue



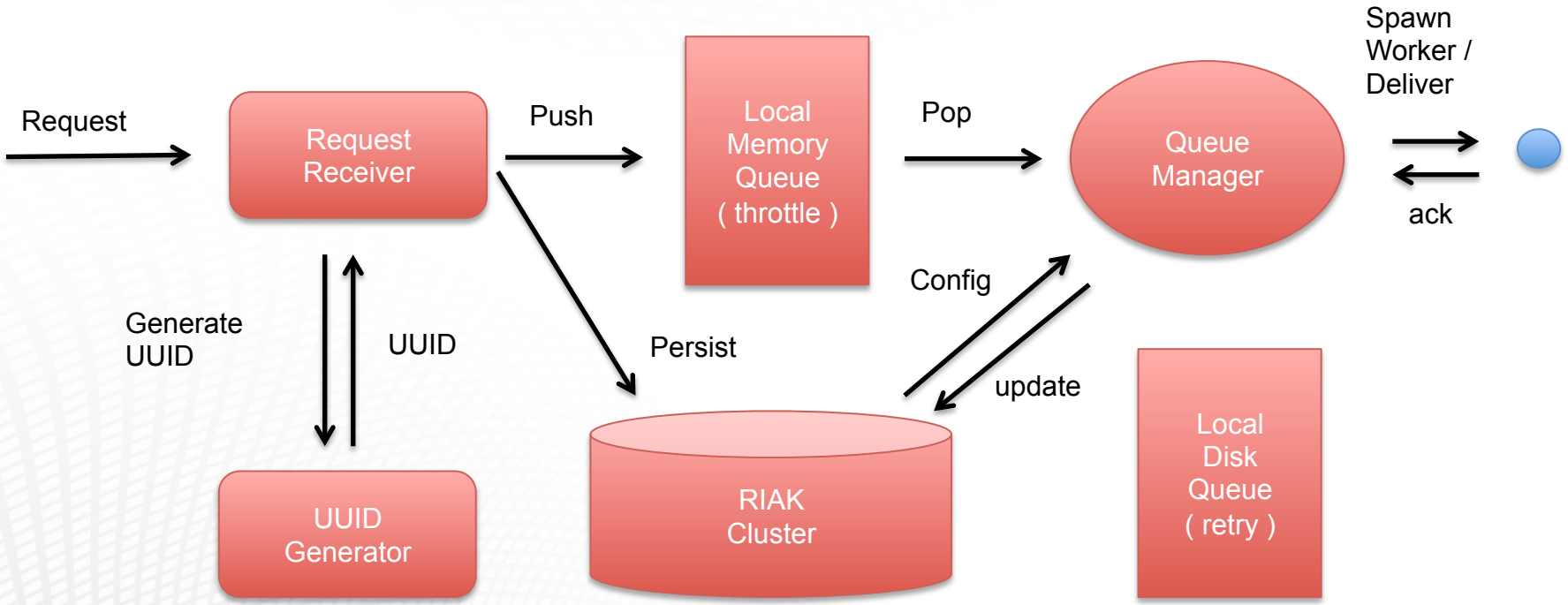
QCon

Building a Queue

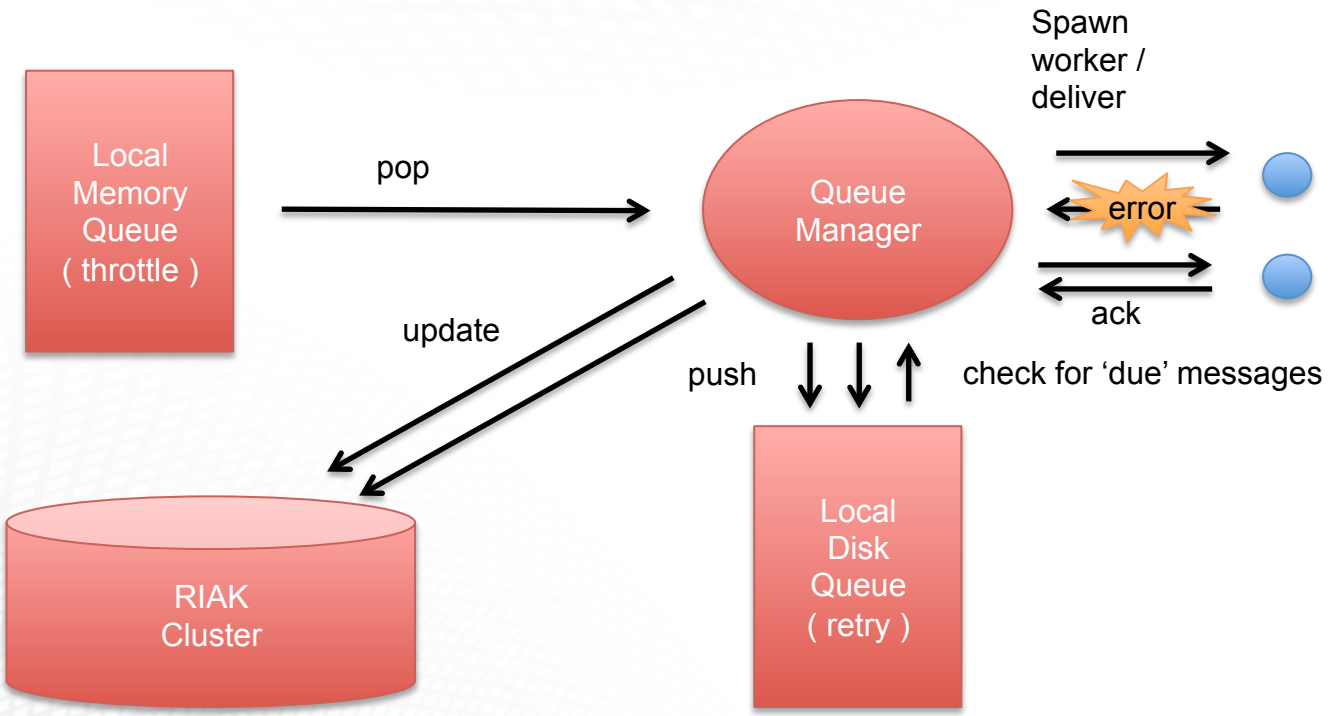
- Similar requirements to the wallet
  - Fast
  - Scalable
  - Robust / Recoverable
- Scheduling
  - Ability to send a message later
  - Retry queues with incrementing delay
- Throttling
  - Rate at which we process requests
  - Rate at which we can send messages



Building a Queue - Throttling

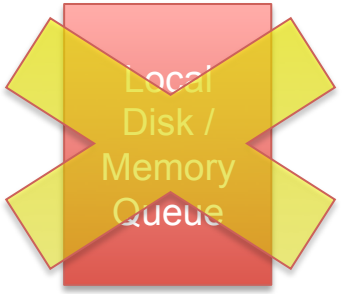


Building a Queue -Retry



Building a Queue - Recovery

- Queue crashes / dies
  - Memory (throttle)
  - Disk (retry)
- Query RIAK
  - Physical node name (e.g. sender)
- RIAK provides 3 different techniques
  - Map reduce
  - Key filtering
  - Secondary index's



Building a Queue - Recovery

- Map reduce
  - Slowest to execute recovery
  - RIAK bitcask backend (very fast)
  - Cost is (1 + N) RIAK operations
    (N = Number of nodes)
- Key filtering
  - Faster to execute than Map reduce
  - RIAK bitcask backend
  - Cost is (1 + 2N) RIAK operations
- Secondary indexes
  - Fastest to execute recovery
  - RIAK leveldb backend (slower than bitcask)
  - Cost is (1 + N) RIAK operations

Bucket: recovery
Key: abcd-1234-1234
Value:
node: sender

Bucket: recovery
Key: sender-abcd-1234-1234

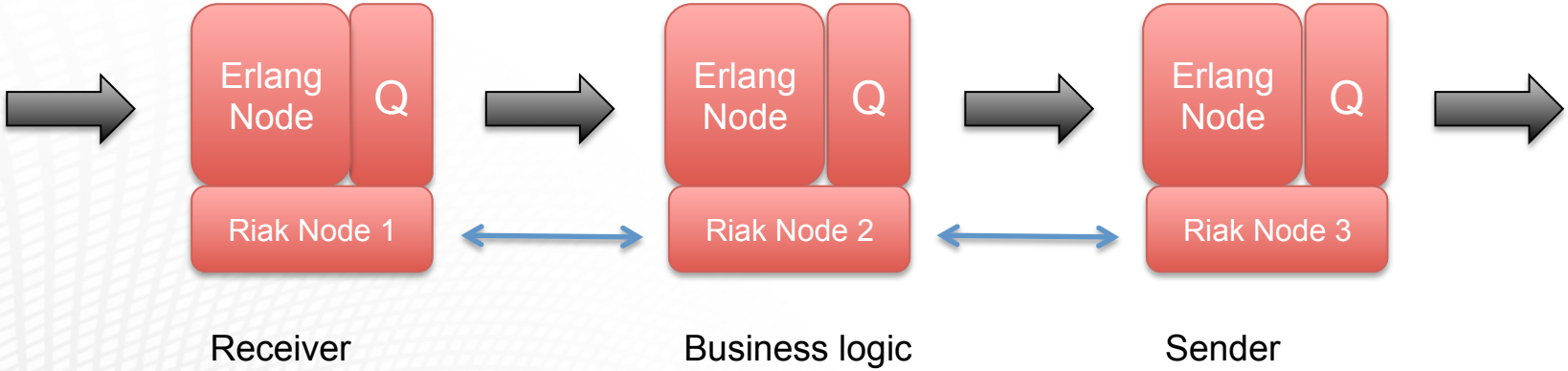
Bucket: recovery
Key: abcd-1234-1234
Indexes:
node: sender

Building a Queue - Recovery

- After testing we choose levelDB & Secondary indexes
  - Good compromise between
    - Speed of running recovery
    - Performance impact on the queues

Building a Queue - Flow

- Insert: [bucket: recovery, key: abcd-1234-1234, index: **node - receiver**]
- Update: [bucket: recovery, key: abcd-1234-1234, index: **node - business logic**]
- Update: [bucket: recovery, key: abcd-1234-1234, index: **node - sender**]
- Delete: [bucket: recovery, key: abcd-1234-1234]



Building a Queue – Today

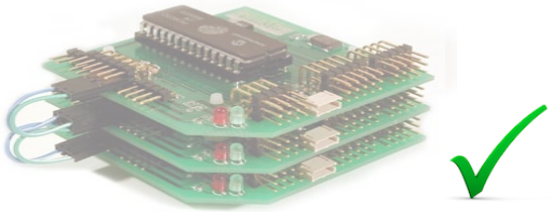
- We have a standalone prototype queue based on levelDB
  - Undergraduate final year project
    - Pathfinder scheme
    - Dan Fernandez
    - https://github.com/mitadmin/dupQ

Battle Stories

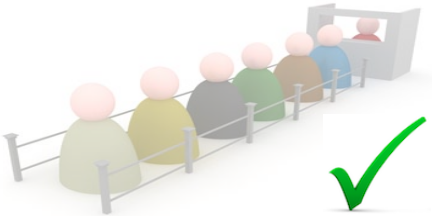
mobile interactive group | a velti company

- Building a wallet

- Optimizing your hardware stack



- Building a robust queue



QCon

Thank You

mobile interactive group | a velti company

Questions?

If you'd like to work *with* or *for* MIG please contact the MIG Team:

David Dawson
+44 7900 005 759
David@MIGcan.com

https://github.com/DangerDawson

Marcus Kern
+44 7932 661 527
Marcus@MIGcan.com

QCon