The Smarkets Platform

Hunter Morris
CTO & Co-founder

smarkets

Intro to Prediction Markets

- “Binary event futures”
- Members trade pre-written contracts with others

smarkets

Exchange Mechanics

- Price/time priority
- *Single sorted order book*
- Not trivially parallelisable

Exchange Mechanics

- Other parts are easy to distribute
  - Individual accounts
  - Individual orders
  - Data streams

Member Activity

- Social streams of data
- “Challenges”
- Privacy filters
- Leaderboards
- Badges

smarkets

Early Days

- Architecture is message-heavy
- Erlang is a natural fit
- Functional style is easier to prove
- Reliability is **king**

Erlang *pitfalls*

- Core web functionality missing
- Email?
- String manipulation?
- Web developer ease of use
- Web frameworks?

General Decimal Arithmetic

- http://speleotrove.com/decimal/
- IEEE 754 + IEEE 854
- Loads of unit tests

General Decimal Arithmetic

- Open source?
- Anybody interested?
- 50% implemented (enough for our purposes)

NIH

- How difficult is it to create a web framework?

NIH

- Very difficult

REST

- Building an API
- Tight HTTP
- Multiple representations
  - JSON
  - XHTML
  - Atom

REST

- “Member-facing” front-end uses our own API
- Eating our own dog food

EWGI

- Erlang Web Gateway Interface
- Conceived by Filippo Pacini
- Similar to Python's WSGI
- PEP 333

EWGI

- Components are middleware
- Standard interface for building components
- Functional

EWGI

- Multiple hosting implementations
  - Mochiweb
  - Yaws
  - Inets
  - Nginx C module

Smak

- REST “toolkit”
- Flexible URL routing
  - Regular expressions
  - Erlang patterns

Smak

- Define resources as property lists
- Base resource handling by default
- Sort of a light-weight object orientation

Smak

- Stateless
- Each component is easy to test independently
- Mock requests/data

Persistence

- Append-only files
- Minimal memory overhead
- No file seeks

Persistence

- Some data is *very* unstructured

Persistence

- CouchDB
  - You may have heard of it
  - If not, it's very cool, so you should check it out
  - Naturally distributable

Persistence

- Couch can be used for “bank transactions”
- Eventual consistency is (often) perfectly acceptable

Safety

- Node failures happen often
- Traditional backups are too infrequent

Safety

- Process transaction logs
- Slave processes maintain state
- Snapshots

Safety

- Introducing AMQP
- Open protocol for messaging
- Crosses many domains
- “Durable” message queues
  - *See talk from yesterday*

Eventual consistency

- Messaging topography
- Not everything must happen straight away

Eventual consistency

- A kind of data “quality of service”
- Prioritise sets of data
  - Members don't need account history every instant, for example

Logging

- RabbitMQ has very flexible topography
- Separate physical location
- Back up data

Request statelessness

- Try wherever possible
- Allows for simple scalability

Request statelessness

- Independently testable
- No need for node affinity
- Buffering processes allow front-end architecture to vary independently from back-end

Testing

- State machines
- Transitions are predictable
  - Coverage analysis
  - Natural unit tests

Testing - Requests

- Mock HTTP requests
- EWGI makes this very simple

Testing - Data

- CouchDB easy to “imitate”
- Mock service

Testing – Messaging

- Slightly more complex
- Sequence is important

Questions?

- Hunter Morris
- hunter.morris@smarkets.com
- http://smarkets.com/
- http://github.com/skarab/