Ranking 1v1 Games with Erlang
How few tools solve large problems

Jesper Louis Andersen
jesper.louis.andersen@erlang-solutions.com
http://jlouisramblings.blogspot.com

March 2, 2012

1v1 Games

Typical 1v1 Games:

- Chess
- Go
- WordFeud
- StartCraft II 1v1 games
- Quake Live Duels

Rating Systems

- Track a *belief* in the skill of a player
- Base on historical data
- One simple measure: Did the player win, tie or lose?

Common rating system: ELO

- Arpad Elo, implemented 1960, chess rating
- Simple system - hand calculation possible
- Assumes Normal Distribution of player skill around a rating
- Variance is fixed - fixed confidence

ELO weaknesses:

- Player A has 32 matches, Player B has 157 - same belief?
- Player A plays regurlarly, Player B is returning after a 4 month hiatus - same belief?
- Player A is consistently beating players above his Rating.

Glicko 2

- Created by Mark E. Glickman
- Tracks three variables for every player: R, RD and σ:
- The current rating R
- The variance of the rating RD
- The *volatility* σ - which is a measure of consistency.
- σ makes the system able to cope with fast-changing players.

Glicko 2 (cont.)

- We report a players rating as an Interval
- Players start out with $R = 1500$, $RD = 350$
- System has 95% confidence in the range: $R \pm 2 \cdot 350$
- Initial players are somewhere between 800 and 2200 points.
- As players play more games, we lower the RD adjust the R
- σ is adjusted as well if the player is fooling the rating system.

Quake Live

- A Modern update of the Quake 3 Engine.
- A game of skill with a high skill ceiling
- Can best be described as "Chess + Hand/Eye coordination"
- Weapons, Health, Armor, Timing, Position, "Aim"
- Game is extremely strong in EU and Russia - Poland has scores of players (Russia, US, Poland are top nations).

Implementing G2 is easy:

- Full description which is algorithmic.
- Algorithm is a functional chain - 7 steps
- Examples for each step
- As of 22nd of Feb a non-converging loop bug was fixed (QuickCheck! - but took many tests to find)

- Data is “public”.
- Write scraper, backed by modules `re`, `httpc` and the `jsx` application (JSON data)
- Let *PostgreSQL* provide the stable storage

- Players refreshed every 5 days. Provides matches.
- Matches are fetched once, then analyzed separately.
- The system base does *not* care if 3000 refreshers are spawned.
- Database is built such that it is *idempotent*, i.e., $s(s(x)) = s(x)$
- System can restart at any point and the database is consistent.

Enter JOBS, by Ulf Wiger:

```
{ql_fetch,
  %% Queue Section
  [{max_time, 300000},
   {max_size, 300},
   {type, fifo},
  %% Regulator section
   {regulators,
     [{counter, [{limit, 2}]},
      {rate, [{limit, 1}]}]}]}
```

We don't use dampeners since they are buggy it turns out

```
case jobs:ask(ql_fetch) of
    {ok, _Opaque} ->
        case qlg_overload:ask() of
            yes ->
                fetch_and_store(State);
            no ->
                ok
        end,
        {stop, normal, State}
    {error, Reason} ->
       ...
       {stop, normal, State}
end;
```

Overload Handling

- 2 state FSM: Normal/Overloaded
- Errors or Consistently slow response times trigger overload
- Waits for 5 to 15 minutes

Fetch code

- Doesn't care about limitation, let JOBS do it
- Players, Matches and so on go to same Queue
- Just keep queue filled up
- A couple of process pools on simple 1-1 supervisors is all it takes

- "Tournaments of one week"
- Construct "battle graph" $G = (V, E)$ for ranking period
- Vertices, V are the players
- Edges, E, are labeled matches, $A \to B$ if A won over B.
- Ranking is done by a JOBS queue as well
- Counter 4, chunks of 4000 matches are enqueued
- JOBS handles parallelism

- Current performance limit is the DB query time
- Lots of ways to query-optimize
- No HiPE (native code) and G2 is an obvious candidate for it
- A typical one-week tournament is ranked in under 30 secs on 2 cores - which is acceptable.

Results

Use R and ggplot2 to plot:

```
p <- ggplot(df, aes(x=reorder(x, y), y=y,
                    colour=Volatility, ymin=ylo, ymax=yhi)) +
  geom_pointrange() +
  geom_point(aes(y=ylo), colour="black", size=1.5) +
  coord_flip() +
  geom_hline(aes(x=0), lty=2) ...

return(p)
```

Current Problems and work

- JOBS needs some more maturity
- I've seen corner cases of infinite message loops in JOBS
- I plan on fixing JOBS
- Also document JOBS

JOBS is awesome! We just need to push its use some more and get it into shape!

Further work

- Detect banned players and remove them
- Tune the rating system - run simulated annealing to find optimal values of σ and τ for G2 and Quake Live
- Requires optimization
- Scan more matches. Around 125000 matches scanned right now