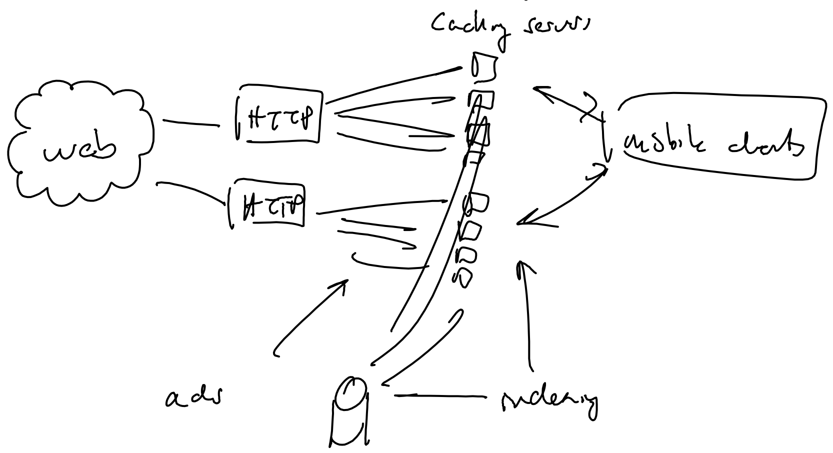
Thorn

Thursday, November 10, 2011
13:29

Thorn

- yet another actor based language
  - No Threads
  - isolated components communicating via messages
- focusing on script-y features
  - caveats
    - no reflection
    - no manipulating fields / methods on the fly
    - static structure imposed by classes
    - no dynamic code ~~loading~~ → no "eval"
- pattern matching
- language-integrated query mechanisms (the SQL stuff they talk about)

Thorn is a good match for running twitter



you can use thorn to glue all of this together.
(not yet but in the future)

Concurrency Model

- low-level send / recvs

interface #1 - low-level mailbox like interface

```
p <<< v
receive {
    m_1 => {...}
    :
    m_k => {...}
}
```

interface #2 - high level

p <-> m(...)	blocks for p's answer	sync m(...)
p <-- m(...)	doesn't block on m	async m(...)
serve	builds a handler for all sync/async methods that are defined	

See Thorn mmo.rpg example on the site

you can do an exception handler as well

→ timeouts

```
p<-> m(...) timeout(n) {deal with it();}
```

```
spawn {
    var done := false;
        async quit() prio 100 {done := true}
        sync go() {...}
        body { while (!done) {serve;}}
    }
```

```
sync cmd(x) {
    r = worker <-> subcmd (x)
    r
}
```

```
}

sync cmd(x) envelope e {
      r = waiter <-- subcmd (e,x);
      throw splitsync ();
}
async subcmd (e,x) {
    r = ...
    sync reply (e,r);
}

    var x := 0
     class counter { def inc() { x=x+1 } }

        p <<< counter()
```