Design and Implementation of a Content Gateway in Erlang

Xu Cao and Haobo Yu

LightPole, Inc.

Roadmap

- Introduction: who, what, why
- Architecture
- Implementation issues
- Discussion

Who are we?

- LightPole
  - Founded in 2007, development center in Beijing
- Focus on mobile software development
  - LBS
  - Mobile banking
  - Mobile airline services

Mobile applications in our eyes

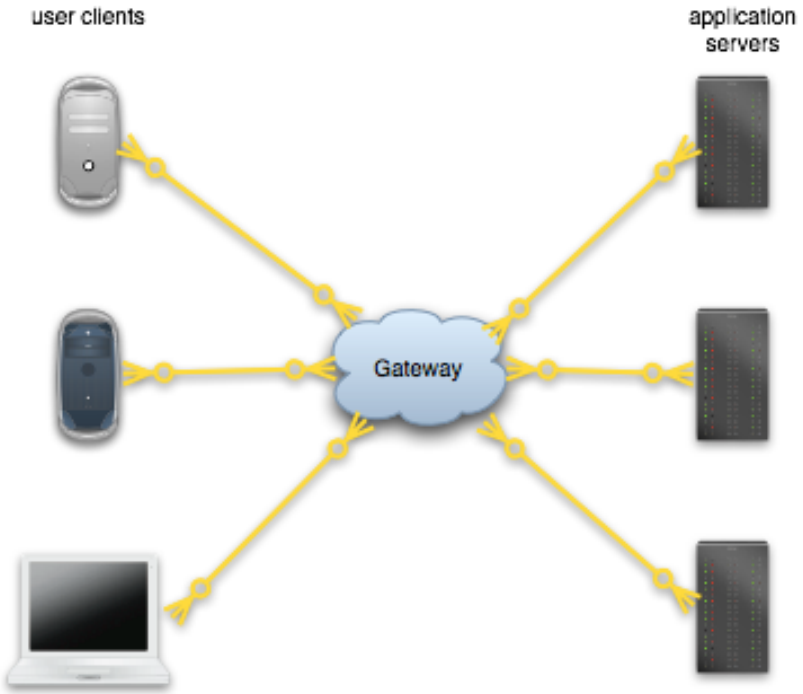
Anatomy of a mobile banking app

Banking
- Account management
- Credit card
- Balance transfer

Value-added Services
- Travel services
- Phone bill payment
- Movie tickets

Customer Services
- Call center
- Online help
- Account setup

Requirements



- Mobile client
  - Supports all major platforms
  - Customized browser
  - Persistent connection
- Content gateway
  - Connects to many app servers via XML, JSON and others
  - Connects to many mobile clients

What does the gateway do?

- Interface with many app servers, process and transform data
- Generate mobile client pages from internal data representations
- Support storage of intermediate states, such as cookies and forms
- Support typical database systems
- Provide OTA and version control of mobile clients for major platforms

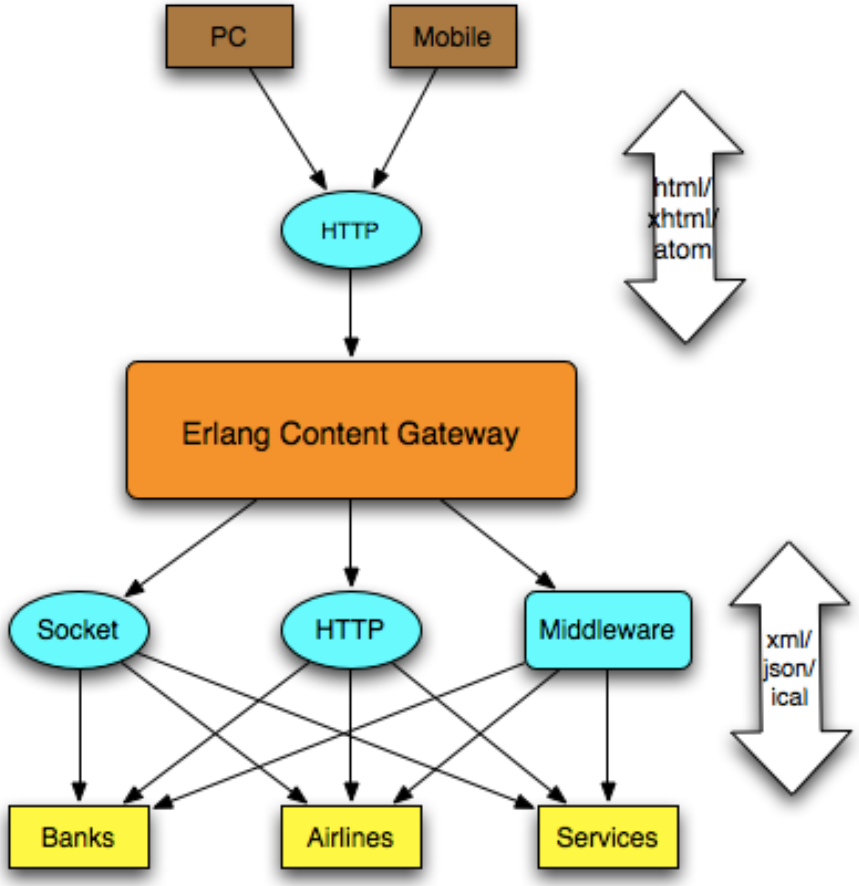
Why Erlang?

- Efficient support of asynchronous communication and process abstraction
- Safe programming protections
- Scalability in a multi-core environment
- Easy to interface with C++ and Java components

Roadmap

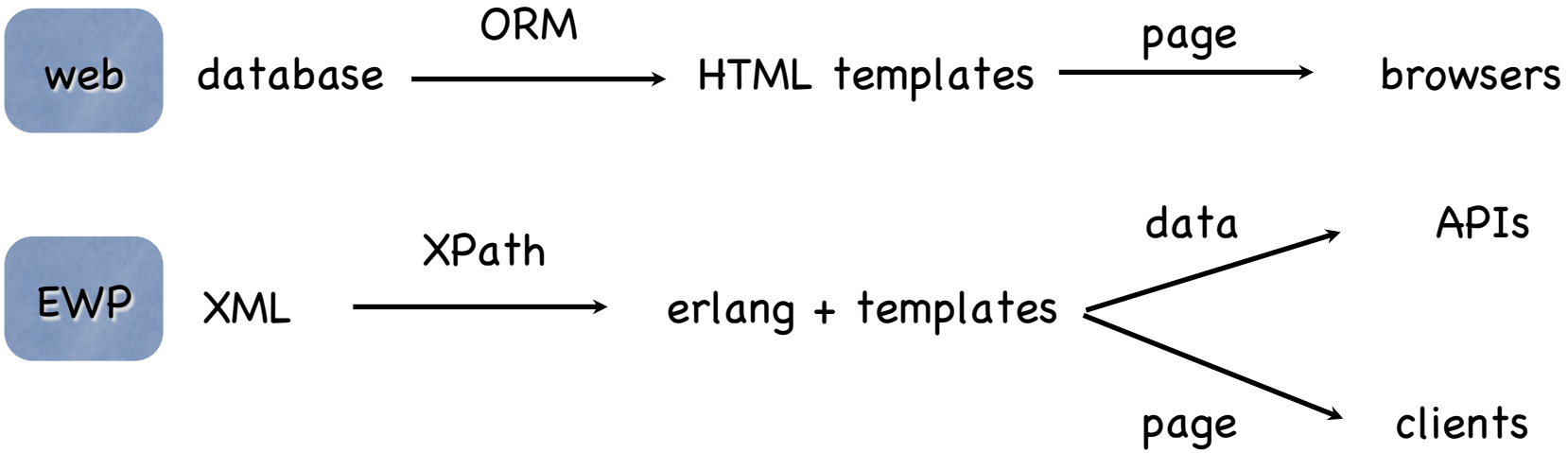
- Introduction: who, what, why
- Architecture
- Implementation issues
- Discussion

Data flow

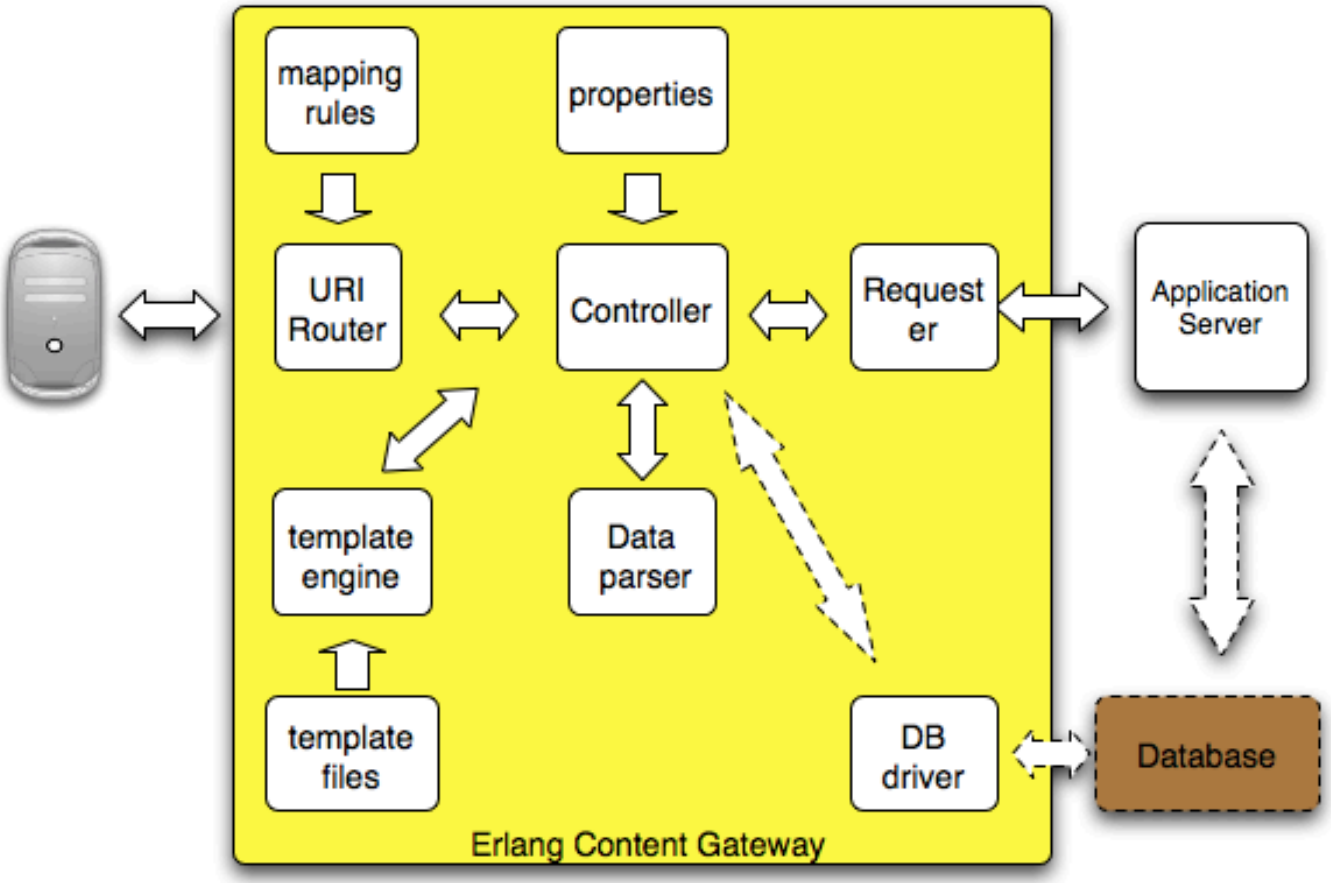

PC
Mobile
HTTP
html/
xhtml/
atom
Erlang Content Gateway
Socket
HTTP
Middleware
xml/
json/
ical
Banks
Airlines
Services

Design choices

- HTTP as universal transport
  - Avoid problems traversing carrier gateways
- Framework to manipulate data flows
  - Embedded in a web server, similar to but different from a web framework

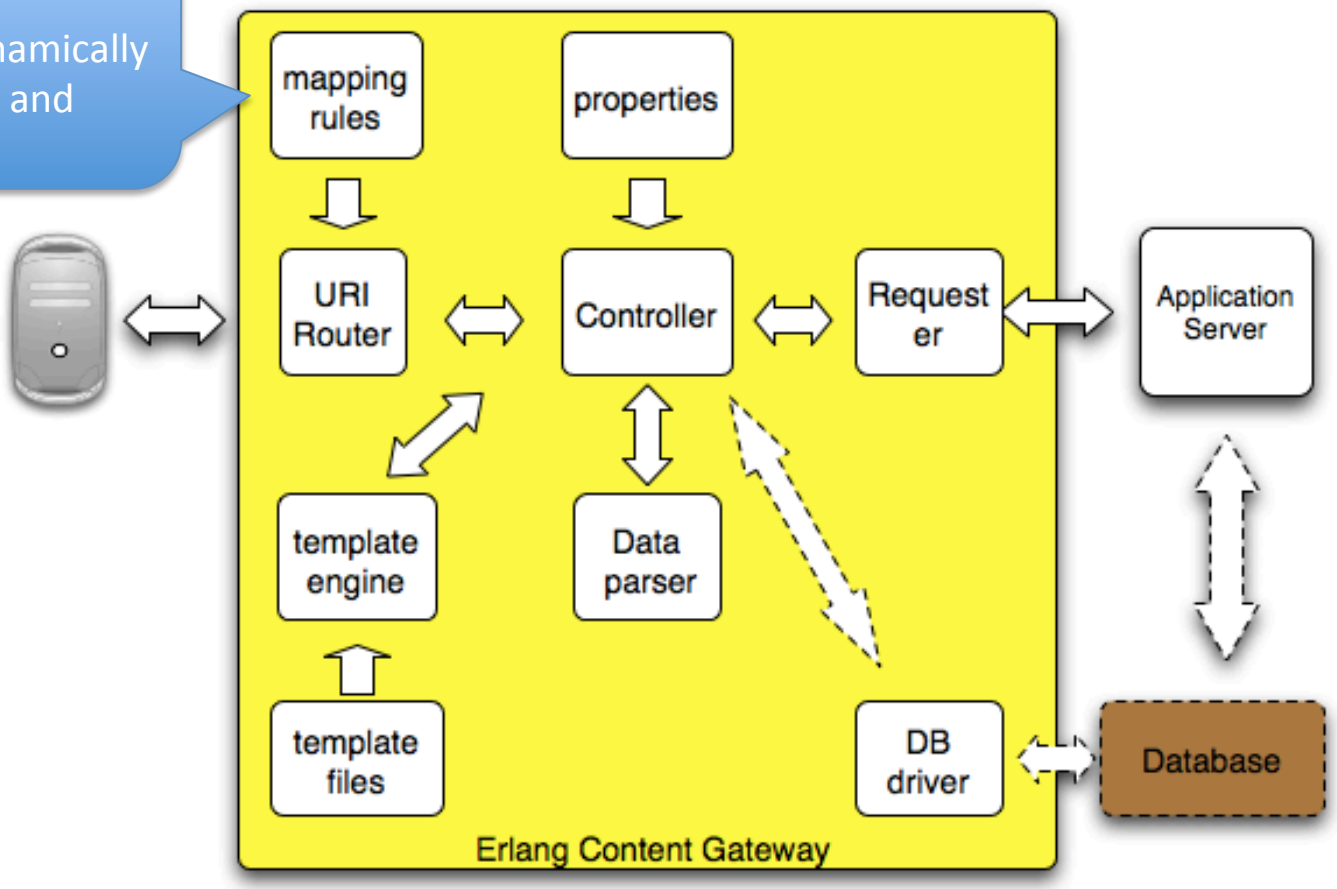


Architecture

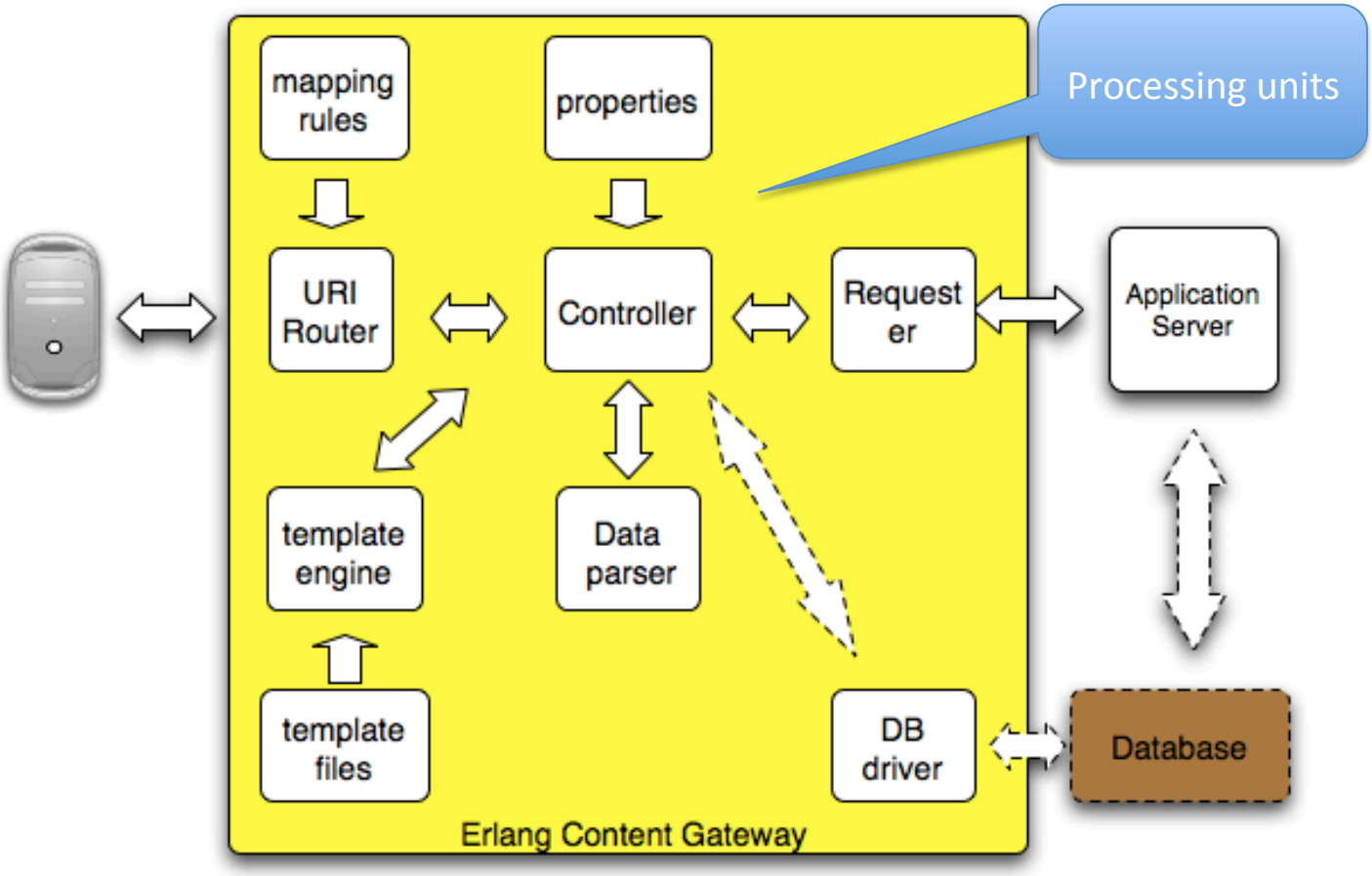


Architecture

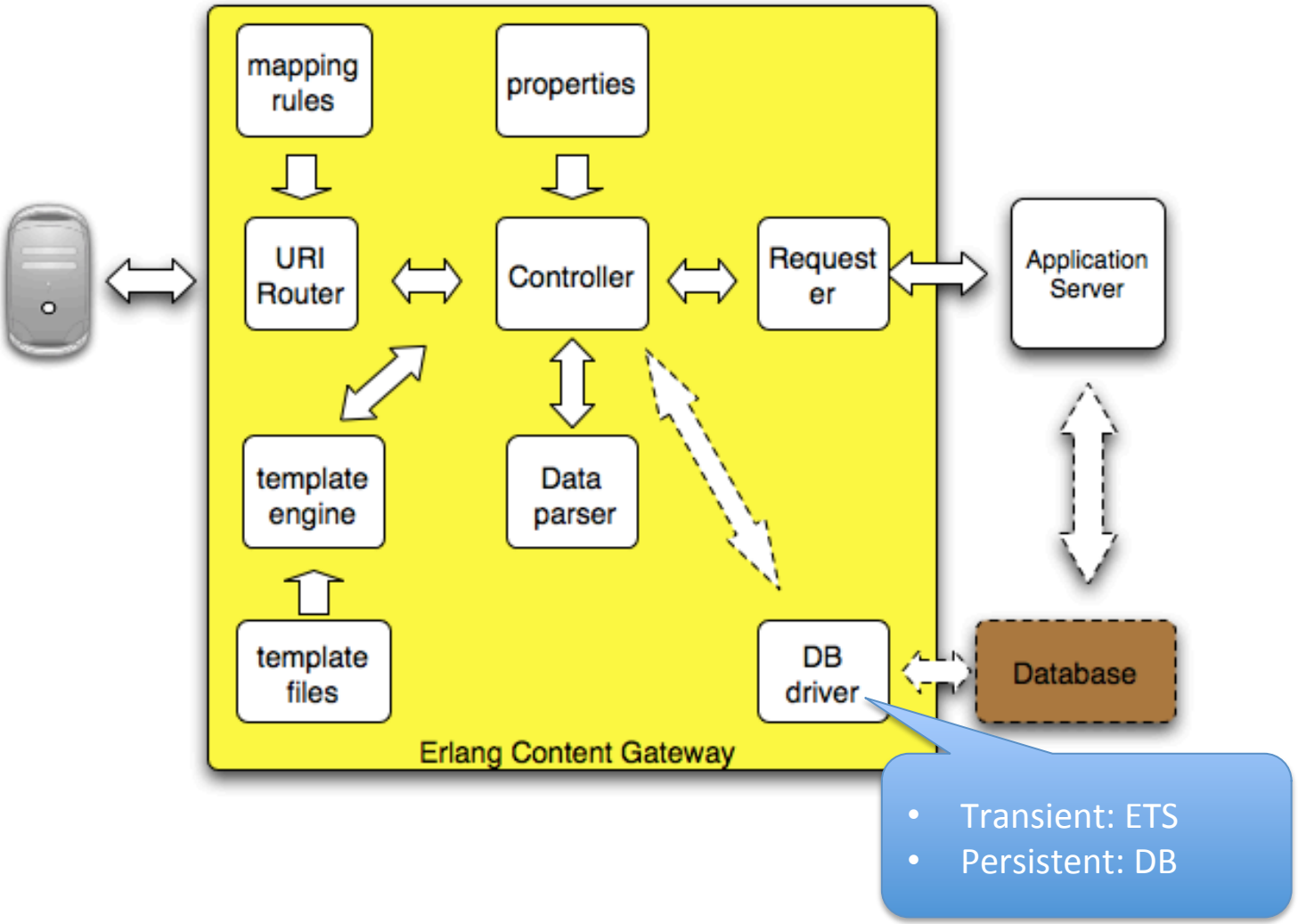
- Entry pint of all requests
- Can be dynamically configured and updated



Architecture



Architecture



How to use it?

Abstractions

- System = [App, …]
  - An app ~ a mobile app
  - Can be loaded, unloaded, updated dynamically
- App = [Controllers, …]
  - Defined by a configuration file, consists of a set of beam and resource files
- Channel = [Channel, rules, …]
  - A channel ~ an interface element in a client
  - Has a unique URL, consists of a set of rules to perform a task, can be nested in any given way
  - Hides details of remote requests to app servers

An example

- Create an app skeleton via script
- .app conf file: request routing rules
  - http://ebank/index -> ebank_controller:index

```
{controllers,
 [{{"ebank","index"},{ebank,index},[{verify,
false}, ...]
}, ...
```

- Template

```
<?cs include:"boc_util”?>
<form method='post' action='#{cs   var:base_url}#/boc_c2c/
0102’>
  ...
</form>
```

- Package it as a zip, loaded into a running gateway

Roadmap

- Introduction: who, what, why
- Architecture
- Implementation issues
- Discussion

Problems

- String implementation is CPU and memory intensive
  - E.g., during XML parsing and template generation
- Support databases and middleware used by enterprise application
  - E.g., Oracle, Sybase, DB2, …
- Lock competition on ETS table in multi-core environment

Solutions

string processing

- Use linked-in driver/NIF to integrate efficient C/C++ libs like XercesC++, XQilla and ClearSilver for string processing and template generation
- Avoid manipulation on string in Erlang directly, pass binary between driver and Erlang process

Template Engine

- Build on top of XercesC and ClearSilver
  - Accept XML as input
  - High performance template generation
- Support nested input data structure
  - Like {key, [{a, v1}, {b,[…]}]}

Result

Xml parser

XML Size	Linked-in Driver (Xerces)	Xmerl
2Kb	0.56ms	0.9ms
5Kb	1.3ms	2.1ms
16Kb	1.4ms	6.7ms

Template language

Page Size	Linked-in Driver (ClearSilver)	ErlyDTL
2Kb	0.04ms	0.12ms
11Kb	0.09ms	0.4ms

Solutions

database and system support

- A linked-in database driver using C/C++ APIs
  - Support elegant API syntax from ErlyDB
  - Support typical enterprise databases: Oracle, DB2, Sybase, MySQL
- Use Java Node to integrate enterprise system library such as IBM CICS Client

Result

CentOS 5.1, MySQL 5.1
10000 insertions and selects repeated 4 times
Select 200K data

Operation	Linked-in driver (ms)	ErlyDB	ODBC
insert	0.5ms	0.868ms	49ms
select	98ms	226ms	115ms

Solutions

resources competition

- Horizontal split policy
  - Split R/W locks for ETS tables (what the write_concurrency optimization did in R14)
  - One exclusive instance of resource for each scheduler
- After optimization, 3 times throughput
  - 4*2.5Ghz, Intel Q8300

About AIX

- Problem
  - AIX 6.1, 4 CPUs
  - SMP-enable VM, R14B01, linked-in driver/NIF could consume equivalent to one CPU
  - This happened to crypto and our own driver
- Workaround
  - One Erlang node for each CPU core (for AIX)

Roadmap

- Introduction: who, what, why
- Architecture
- Implementation issues
- Discussion

Discussion

- Better AIX support?
  - We are willing to work with others interested. Please contact us!
- Optimize it
- Open source it

The End

Thank you!