

# The Erlang Web

an Open Source Fault Tolerant and  
Scalable Web Framework

Erlang User Conference, Stockholm, Sweden  
November 13, 2008



Erlang Training and Consulting Ltd  
[www.erlang-consulting.com](http://www.erlang-consulting.com)

Michał Ptaszek, Michał Slaski, Michał Zajda  
[michal@erlang-consulting.com](mailto:michal@erlang-consulting.com)

---

# Contents

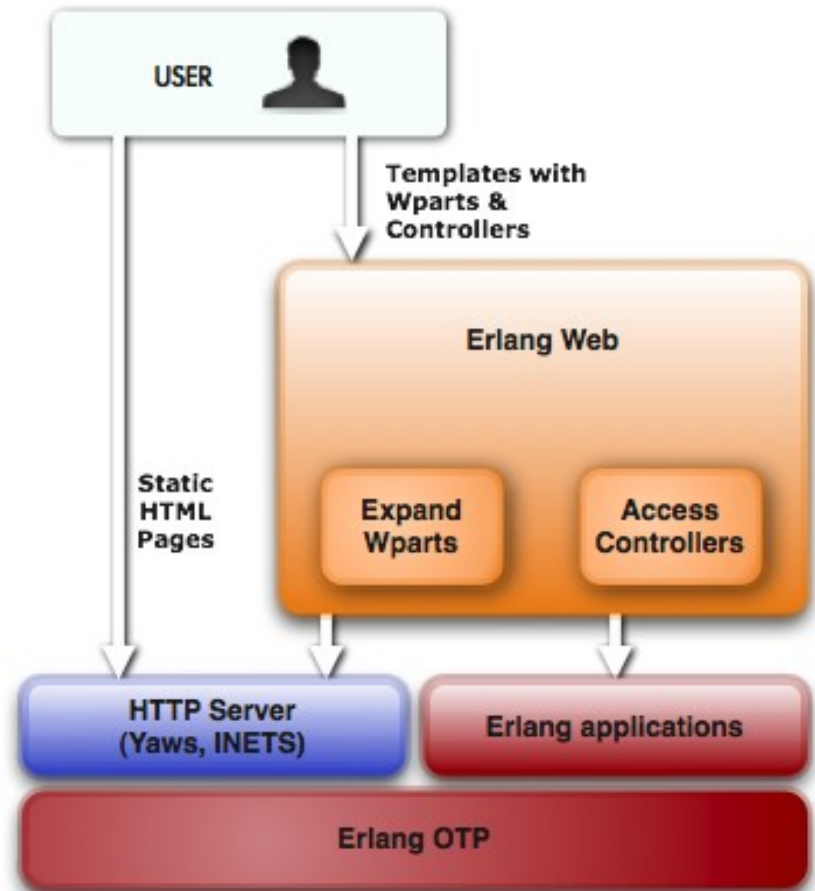
- The reason to have the Erlang Web
- How it works
- Erlang Web 1.1 features
- License
- Scalability
- Fault tolerance
- Stress testing
- Example sites

# The reason to have the Erlang Web

- Concurrent Erlang good for web applications
- Yaws, INETS and Mnesia
- Same platform and share same memory space
- MVC paradigm
- No Erlang code in HTML templates
- Reusable components
- Move from early adopters to early majority

# How it works

- Built on top of HTTP server
- Calls controller functions
- Expands templates with dynamic parts



# How it works

- Dispatcher
  - *regular expressions used to match on URL*
- Controllers
  - *functions accessed through dispatcher*
- Templates
  - *XHTML files and wtpl engine with inheritance*

# How it works

- **Wparts**
  - *tags put in templates*
- **Wtypes**
  - *data types for which validation and formatting can be done automatically*
- **i18n support**
  - *files with multilingual translations*
- **Request dictionary**
  - *mechanism for accessing data from wparts*

# Dispatcher

- A collection of files with regular expressions

```
{dynamic, "^/index.html$", {main, home}}.
```

```
{dynamic, delegate, "^/user", "config/dispatcher/user.conf"}.
```

```
{static, "^/about$", "about.html"}.
```

```
{static, "^style.css$", enoent}.
```

# Controllers - simple case

- Request to URL: `http://hostname/app/module/func`
  - Will use `module.erl`
  - Will call `validate/1` function
  - If successful, will call controller function `func/n`
- `func/n` should:
  - accept arguments returned from `validate/1`
  - return one of:
    - `{template, Path}`
    - `{json, Data}`
    - `{redirect, Url}`

# Controllers with data flow

- The controller module should export `dataflow/1` that returns a list of functions to be called
- Functions will be called one by one
- If something goes wrong, `error/2` function is called
- In this way basic aspects can be implemented

# Controllers with data flow - example

```
dataflow(save) -> [validate, log].
```

```
validate(save, _) ->  
  Name = eptic:fget("post", "example_name"),  
  case is_valid_name(Name) of  
    true -> {ok, [Name]};  
    false -> {error, bad_name}  
  end.
```

```
log(save, Name) -> ...
```

```
error(save, bad_name) ->  
  {template, "templates/error/error_1.html"}.
```

```
save(Name) ->  
  db:save(Name),  
  {template, "templates/save/form.html"}.
```

# Templates

- XHTML files parsed with Xmerl
- Xmerl structures of parsed templates are stored on disk as binaries for better performance
- Template inheritance with wtpl engine
- Automatic generation of forms used in CRUD operations

# Wpart

- HTML template contains `<wpart:somename />`
- During expanding the tag is replaced with value returned from
  - `wpart_somename:handle_call(Element)`
- `handle_call` should accept the current tag as an Xmerl record
- `handle_call` should return any Xmerl construct

```
handle_call(E) ->
```

```
Name = wpartlib:has_attribute("attribute::name", E),
```

```
#xmlText{value = "Hello " ++ Name}.
```

# Wtype

- **Basic wtype**
  - *Used for formatting and validation*
  - *Wtypes for dates, integers, text, etc. are built in*

- **In template:**

```
<form> <wpart:integer name="id"/> </form>  
<wpart:lookup key="id" format="integer"/>
```

- **In controller function:**

```
wtype_integer:validate({[], ID})
```

# Wtype

- **Complex wtype**
  - *Used for defining data structures from the model*
  - *Can consist of basic and complex wtypes*
  - *Can be constructed using Erlang record syntax*
  - *Or can be constructed/updated on the fly*

```
-record(person, {id, name, age}).
```

```
-record(person_types, {  
    id = {integer, []},  
    name = {string, []},  
    age = {integer, [{min, 1}, {max, 140}]}}  
)).
```

# Request dictionary

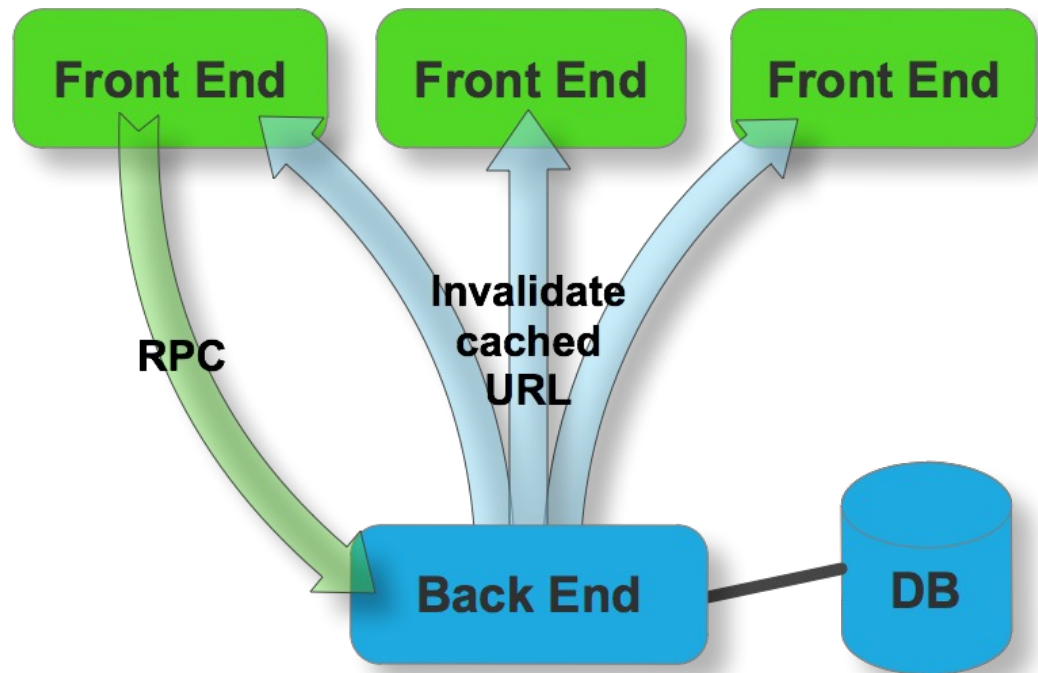
- Temporary storage created per each HTTP request
- Implemented as ETS table
- Stores
  - *GET/POST variables*
  - *Data passed from the model to the view*

# License

- Erlang Web Public License
  - *Derivative work of the Erlang Public License, Version 1.1*
  - *Changes definition of Modifications to include also wpart/wtype modules*

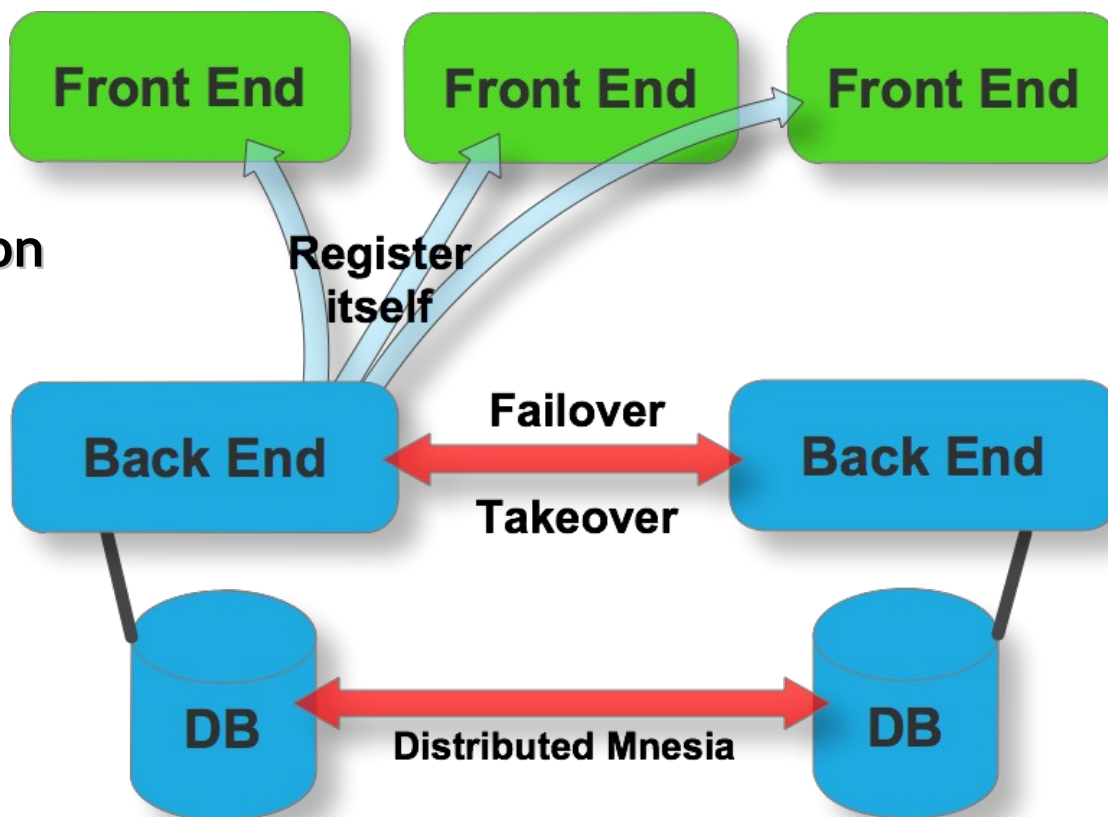
# Scalability

- Front End
  - *HTTP server*
  - *Dispatcher*
  - *Cache tables*
  - *Static content*
- Back End
  - *Invalidator*
  - *Controllers*
  - *Templates*
  - *Database*



# Fault tolerance

- Two Back End nodes
- running Erlang Distributed Application
- which registers at Front Ends during startup



# Stress testing

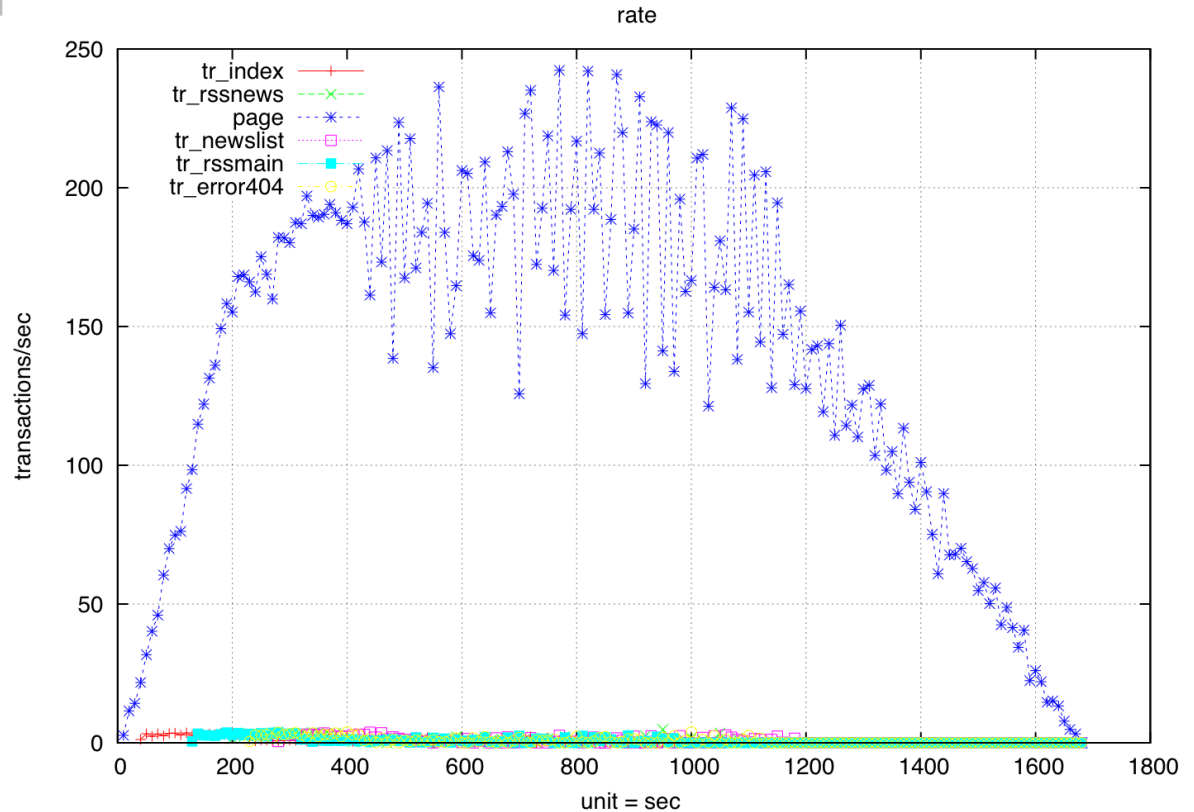
- One machine for Erlang Web application and one for Tsung
  - *Intel Core 2 Due 2.4 GHz*
  - *3 GB DDR2 RAM*
  - *Erlang R12-4, no hiipe*
  - *Suse 11*
- Application implemented with
  - *Yaws 1.73*
  - *Mnesia as database*
  - *dispatcher, template inheritance*

# Stress testing

- No caching
  - *All requests are served by back end*
- Inets http client
  - *Response time = 4ms*
- Tsung (HTTP 1.1 connections)
  - *185 pages / sec*
- Caching on front end
  - *All requests are served by front end*
- Inets http client
  - *Response time = 1ms*
- Tsung (HTTP 1.1 connections)
  - *1000 pages / sec*

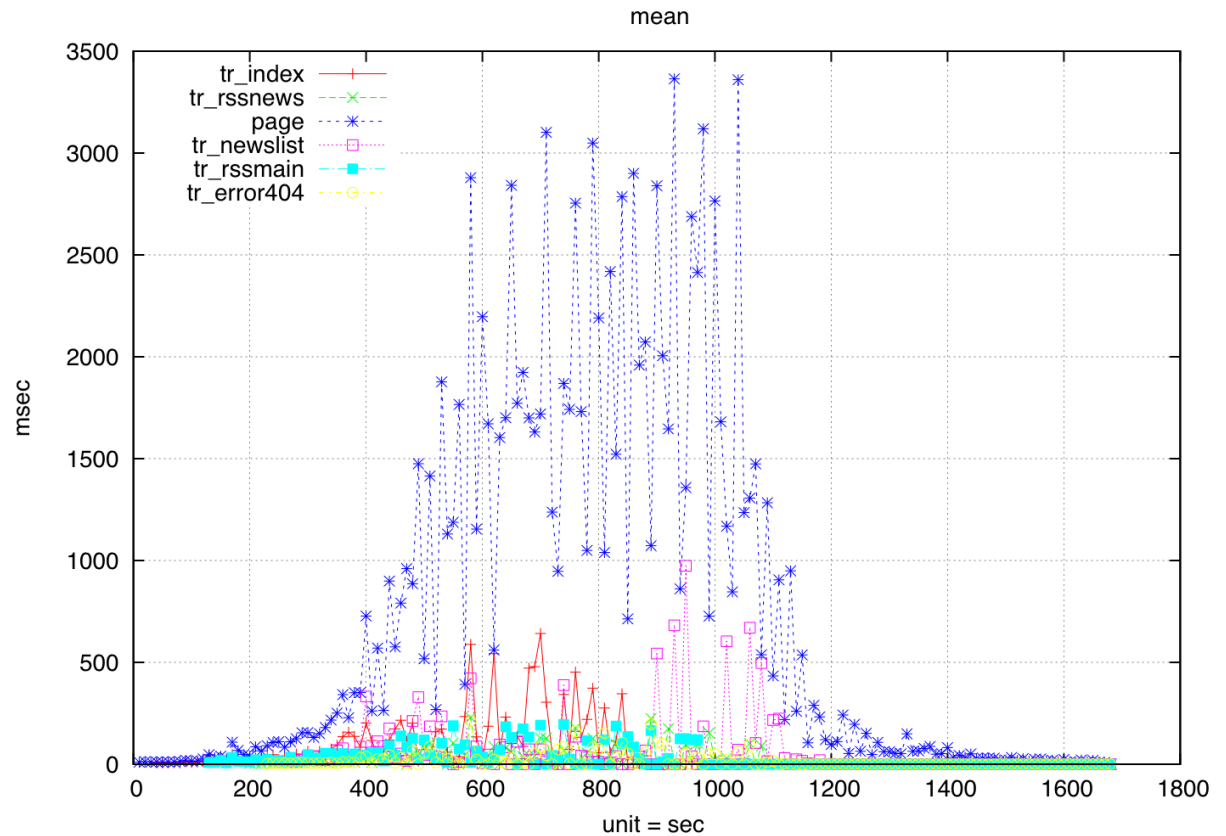
# One node, no caching

- Pages per second



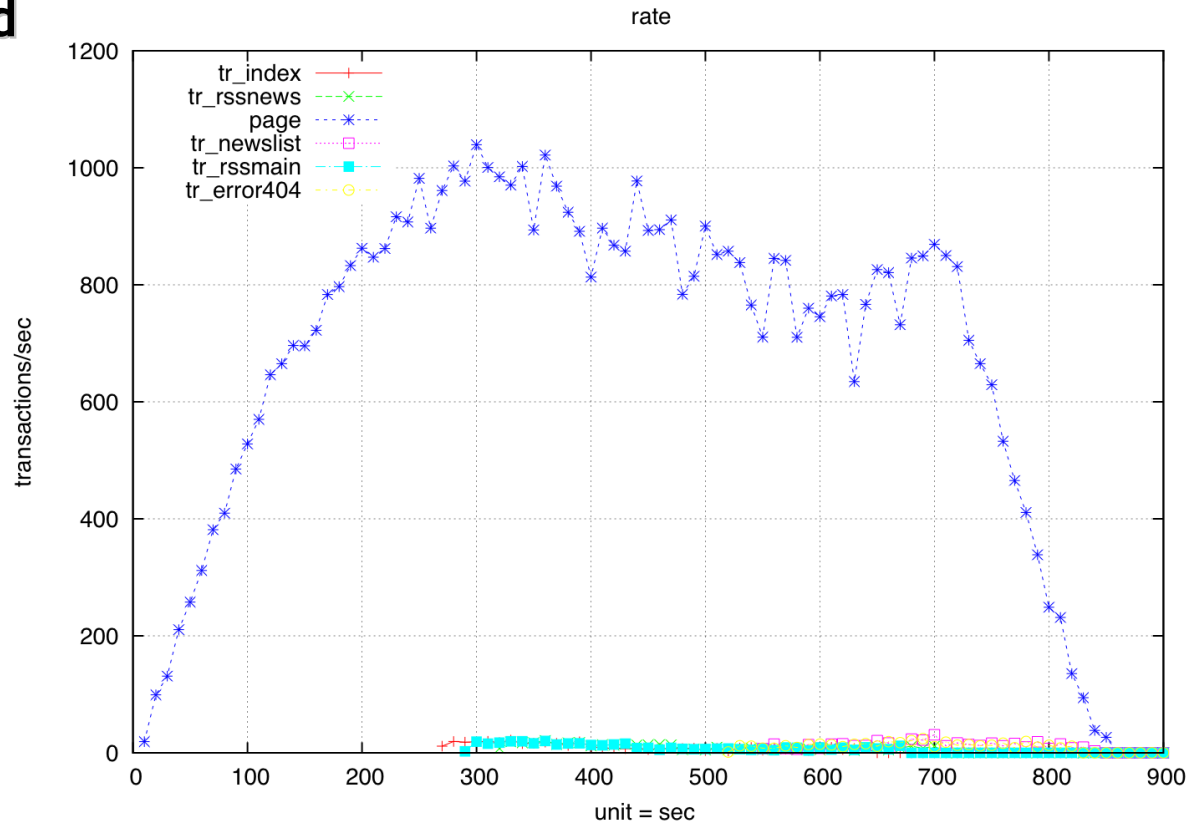
# One node, no caching

- Response time
- in milliseconds



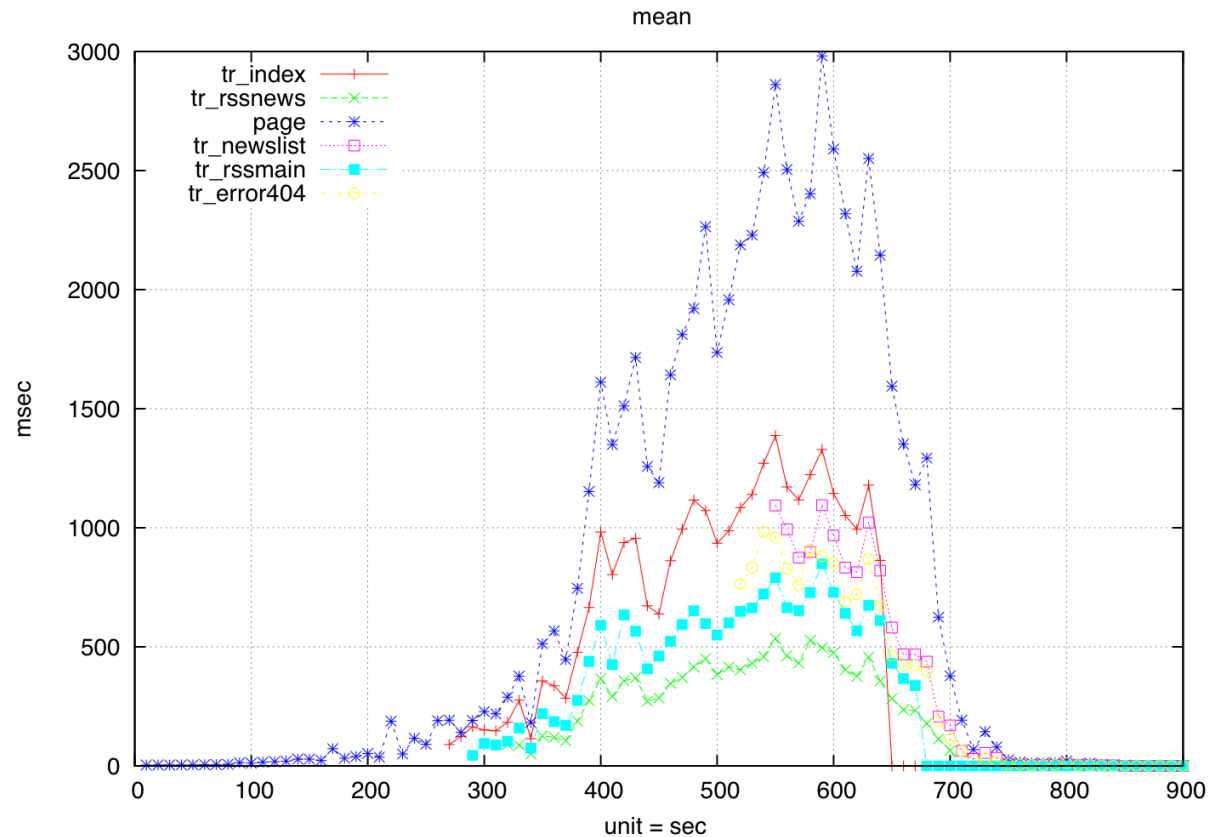
# Two nodes, front end with cache

- Pages per second



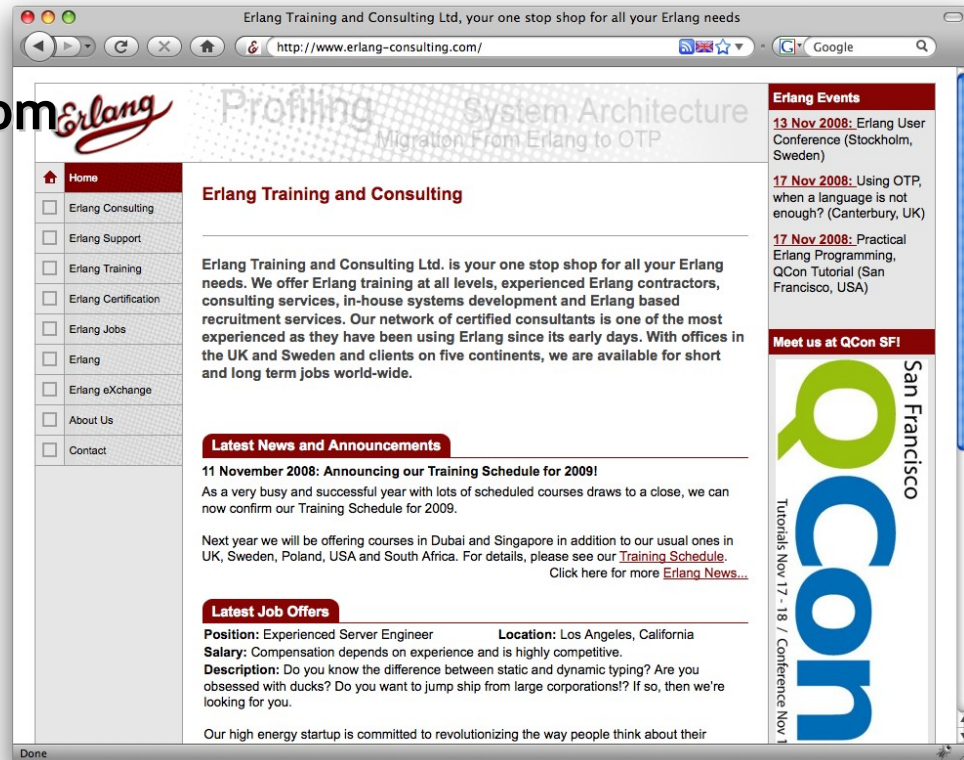
# Two nodes, front end with cache

- Response times
- in milliseconds



# Example sites

- erlang-consulting.com



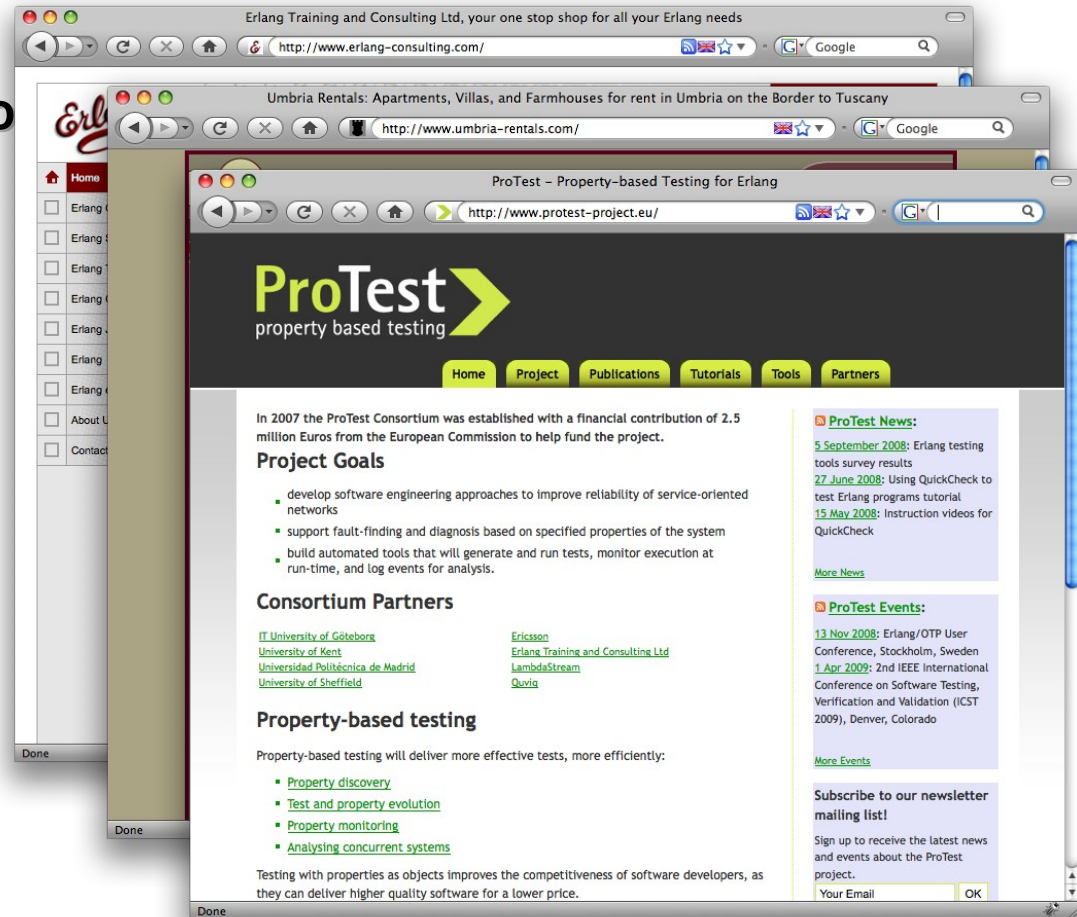
# Example sites

- erlang-consulting.co
- umbria-rentals.com



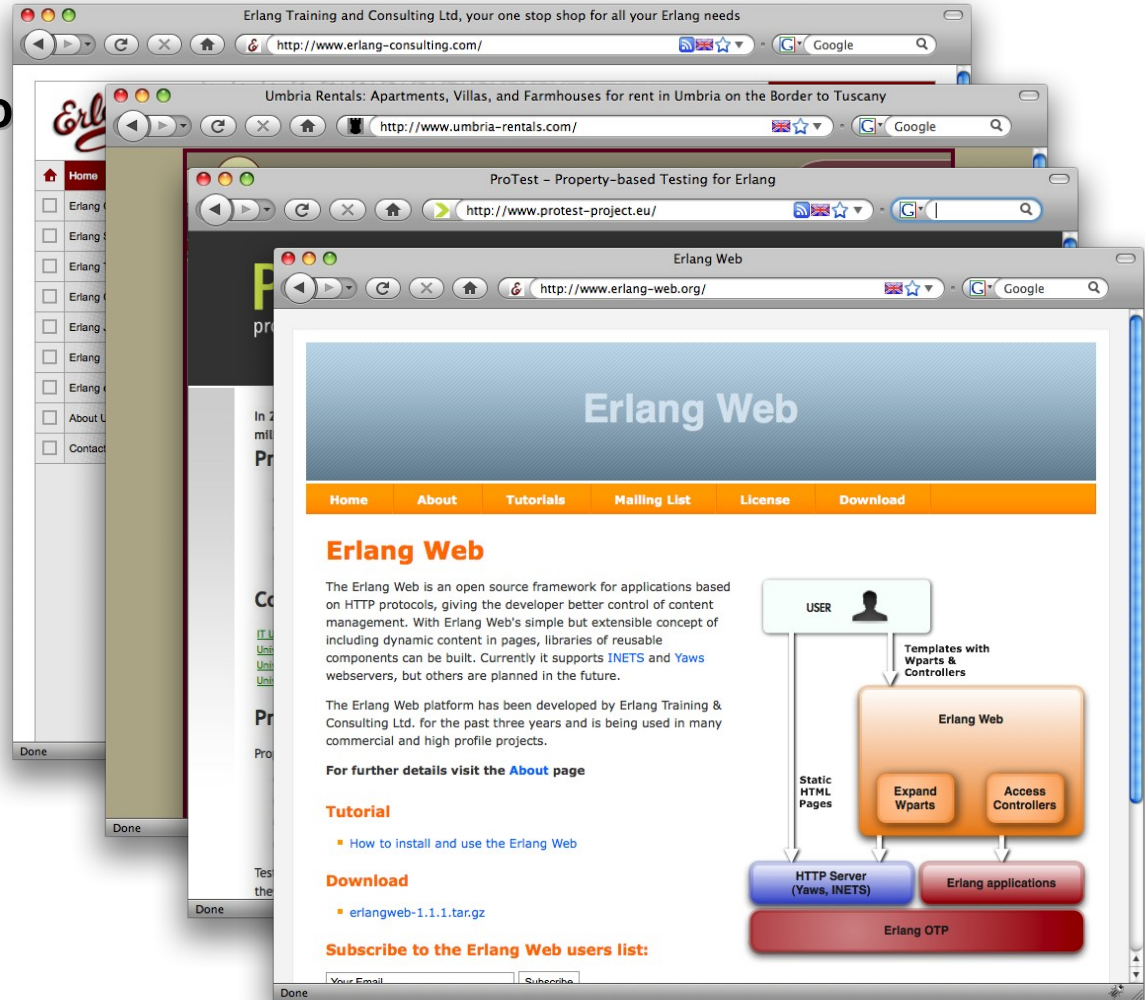
# Example sites

- erlang-consulting.co
- umbria-rentals.com
- protest-project.eu



# Example sites

- erlang-consulting.co
- umbria-rentals.com
- protest-project.eu
- erlang-web.org



# Questions

---



---

# Thank You!

For more information and to download visit  
<http://www.erlang-web.org>