Building the Next Generation Web Services with Erlang: Scalability and High Availability at Stake

Mickaël Rémond <mickael.remond@erlang-fr.org>

19th June 2001

Abstract

The Internet model is waiting for its third generation of services and applications to emerge. The traditionnal players, like SUN MICROSYSTEMS and IBM are proposing a model that is very traditionnal but in a way that is not yet mature when apply to web services. I propose to explore other directions based on Erlang ERICSSON's middleware.

This paper is written for the 2001 PLI Erlang Workshop.
See *http://www.erlang.se/workshop/program.html*

Contents

1 Introduction

Two main evolutions are structuring the web today:

1. The need for scalable services. The web has become a mission critical tool for many companies and must face more and more users. The Web applications thus now need to scale to a point that were not imaginable several years ago.

2. The development of what is called web services. A web site already use several services provided by third-parties (DNS, SMTP routing, ...). But those services are also using business services, like exchange rate, weather, and so on. There is now a tight and complex interdependancy between those services implying that they must be scalable and fault-tolerant to ensure that a huge part of the web activity will not be stuck by a failure of one particular service.

Erlang has been made as a tool capable of solving those two constraints.

2 The main Erlang features

...

3 The traditional 3-tiers approach

The main initiative to fullfill this requirement come from SUN MICROSYSTEMS. They are trying to offer a model basically based on the traditional 3-tiers approach.

The Java approach promotes Applications servers as the execution environment and Enterprise Java Beans (EJB) as the development model. The application server is on the middleware part. The front-end is the presentation layer relying on a Web application or a stand-alone Java application. The back-end is the data access layer, encapsulating access to legacy system and databases.

The middle tiers is probably the most important. Accessing business object in the middle tier can make it possible to build several different presentation layer using the the business logic and thus preserving consistency.

This model is not yet very mature. It offers poor performance and reliability issues, mainly because you are supposed to developp everything in Java. The current Java Virtual Machines (JVM) are still huge memory consumers and are not yet very stable.

You cannot start building critical application on a technology that is not yet mature.

The alternative I am proposing is efficient and mature. You can build clustered fault tolerant robust server with Erlang and rely on one of the most efficient framework to develop user interface (client side: QT from TROLLTECH).

Schema showing a comparison of the traditionnal 3-tiers architecture and the Erlang approach architecture: Data-tiers, middle tier and presentation layer.

As a proof of concept, I will show a development made to meet such requirements. The tool is a news publishing system. The news are published on the Web and managed by a traditionnal back-end to ensure that an easy key-in by the users. The back-end has been developed with the QT framework (TROLLTECH) and directly use the Erlang protocol.

4 Complex, robust and scalable web services

4.1 SOAP services

SOAP (*Simple Object Access Protocol*) is basically a way to call remote procedure, over an http transport protocol and with an XML formalism.

Here is what a SOAP message looks like: ... Schema ...

But the SOAP remote procedure calls are now becoming a standard to build complex Web services composed with smaller components. With a SOAP interface for Erlang, robust Erlang servers can be integrated with SOAP compliant applications. As the implementation and plateform does not really matter in this approach, we will soon be able to switch from standard Web services to robust Erlang based services.

An Erlang implementation of SOAP has been made by IDEALX (http://www.IDEALX.org). I will describe here how you can build a SOAP access to the previously described application.

We a SOAP connector we can integrate to traditionnal company and Information System.

4.2 Manderlbot

The Manderlbot project does not try to surf the SOAP and XML hype to prove Erlang strength in mission critical application.

Instead, it aims at building a robust and scalable network services, setup and driven by a web interface.

Manderlbot aims at proposing an IRC (Internet Relay Chat) bot hosting system. You can setup a bot for a channel via a classic web interface. The bot you create will then connect to your IRC channel and offer special services on that channel.

This tool is experimental and aims at testing Erlang scability as IRC bot are usualy very demanding services.

4.3 Worldforge

Worldforge is a project aiming at developping an online role playing game servers, with massively multi-user constraints. It means that thousands of players may come and play in the very same world.

The needs for such a development are importants. We must be able to manage a great numbers of network connections and to be able to react very quickly to incoming events. We also have to compute interaction between the characters and send back the results to all client programs.

So, the main technical goals for such a platform are :

- extrem network orientation (lot of connections, very demanding information routing) ;
- very high processing level which seems impossible to reach on a single machine. Whatever the program's power, it is always possible to add clients and then overflow the program's capabilities.

Those two needs show by themselves that we must be think about developping a distributed server yet.

The Erlang language has intrinsics features which make it possible to develop this distributed server much more easily than in any other language. It offers precious possibilities for such a server, such as running code hot swapping, without service interruption. Lastly, Erlang was written by Ericsson for its routers, with similar requirements than Worldforge server: high numbers of simultaneous connections ; ability to bring information to the right destination. At this point, the analogy with Worldforge server is obvious.

The idea of using Erlang for such development is then appealing. So Thierry Mallartd decided to experiment it for a Worldforge server. I will briefly present the point we reach so far.

5 Conclusion

The approach I described could certainly be apply to many service. Maybe the network computing promised by Sun can revive, based on Erlang web services.

Maybe Erlang will help improving the rise of the new fault-tolerant service. The time has come I think. Companies are yet building there web services with a complex technical architecture organised in a big cluster of machines. They can now improve the software that run over the cluster of computer by using technical and business oriented services made with the Erlang development environment.

We will keep on working in this direction and provide you with feedback on Erlang middleware usage outside the telco industry.