Universidade Federal de Pernambuco

Bacharelado em Ciência da Computação
Proposta do Trabalho de Graduação

Análise e Implementação de um middleware baseado em RPC utilizando Erlang

Autor: Miguel Rodrigues Araújo
Orientador: Carlos André Guimarães Ferraz

Setembro, 2016

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

70 ANOS
TEMPOSTRANSVERSOS

Conteúdo

1 Contexto
2 Objetivo
3 Cronograma
4 Possíveis Avaliadores
5 Referências
6 Assinaturas

1 Contexto

Um sistema distribuído é composto por computadores conectados em rede (hardware e software) que se comunicam e coordenam suas ações somente através do envio de mensagens (Coulouris et al., 2001). Exemplo clássico de sistemas distribuídos inclui, entre outros, a Internet[1]. Ela pode ser vista como um imenso número de servidores e clientes se comunicando constantemente, através de *n* protocolos.

Existem várias razões para que sistemas distribuídos sejam usados. Por exemplo, se um problema pode ser quebrado em vários subproblemas que podem ser resolvidos independentemente, dividir esses subproblemas em várias máquinas pode ser uma solução interessante. Além disto, se uma aplicação pode ser distribuída sem muita perda, distribuir tal aplicação evita que ela tenha um ponto único de falha (VÖLTER et al., 2004, p. 1).

Pela sua natureza, sistemas distribuídos são complexos de serem implementados, e exigem um vasto conjunto de requisitos (heterogeneidade, *openess*, segurança, escalabilidade, tolerância a falhas, concorrência e transparência). Um exemplo importante é a tarefa de prover transparência: o sistema tem que ser elaborado de forma que o usuário final não perceba que o sistema é distribuído (MISHRA; TRIPATHI, 2014).

O melhor cenário indica que o desenvolvedor necessita somente em resolver o problema, esquecendo os passos inerentes ao fato da aplicação ser distribuída. Para permitir este cenário, foi criada uma camada extra na infraestrutura chamada *middleware*. Ela é um conjunto de serviços que ajuda a resolver problemas de heterogeneidade e distribuição (Bernstein 1996). Especificamente, esta camada utiliza-se dos mecanismos de comunicação de "baixo nível" providos pelo sistema operacional para fornecer uma comunicação de "alto nível" para as aplicações distribuídas. Toda a complexidade da distribuição é "absorvida" pelo *middleware*. Ou seja, ele tem o propósito de abstrair as complexidades de um sistema ou hardware, permitindo o desenvolvedor da aplicação focar todo o seu esforço na tarefa a ser resolvida, sem se distrair com conceitos ortogonais do sistema ou hardware [4].

Remote Procedure Calls (RPC) foi o protocolo escolhido para a construção do middleware neste trabalho (BIRREL; NELSON, 1984). RPC é um protocolo que um programa pode usar para requisitar um serviço de outro programa localizado em outro computador na rede sem ter entendimento sobre os detalhes da rede. RPC usa o modelo cliente/servidor[2]. O programa requisitante é um cliente e o programa fornecedor é um servidor. Semelhante a uma chamada de procedimento local, um RPC é uma operação síncrona que requisita alguma operação de um procedimento remoto e, enquanto o resultado dessa operação não é devolvido, a programa permanece bloqueado. No entanto, o uso de processos leves que compartilham o mesmo espaço de endereço permite que várias chamadas de procedimento remoto sejam executadas simultaneamente [6].

Erlang[3] foi a linguagem adotada para o desenvolvimento do middleware. Erlang é uma linguagem funcional focada em sistemas com as seguintes características: concorrente, distribuída, robusta, tolerante a falhas, alta disponibilidade e *hot code upgrade* [7].

Deste modo, é possível identificar que Erlang se encaixa perfeitamente na proposta do trabalho. Portanto, este trabalho avalia as vantagens e desvantagens de se construir um middleware utilizando Erlang.

---

[1] https://en.wikipedia.org/wiki/Internet

[2] https://pt.wikipedia.org/wiki/Cliente-servidor

[3] http://www.erlang.org

2 Objetivo

O objetivo deste trabalho é construir um middleware com suporte a chamadas de procedimentos remotos utilizando Erlang. Além disto, o trabalho foca em fazer fazer uma análise aprofundada sobre as vantagens e desvantagens de ter utilizado Erlang e as decisões de projeto, provavelmente avaliando quais seriam as perdas e ganhos se o middleware fosse construído em uma linguagem de paradigma orientado a objetos. Por fim, será feita uma avaliação de eficiência com outros middleware implementados no Centro de Informática ou sugeridos pelo orientador e/ou avaliador do trabalho durante o desenvolvimento do projeto.

3 Cronograma

Atividades	Ago	Set	Out	Nov	Dez
Definição de Proposta					
Revisão Bibliográfica					
Estudo sobre sistemas distribuídos					
Pesquisa aprofundada sobre Erlang e *Remoting Patterns*					
Desenvolvimento do middleware					
Elaboração do relatório					
Preparação da defesa					
Defesa do Trabalho de Graduação					

4 Possíveis Avaliadores

Nelson Souto Rosa

5 Referências

[1] Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair (2011). **Distributed Systems: Concepts and Design (5th Edition)**. Boston: Addison-Wesley. ISBN 0-132-14301-1.

[2] VÖLTER, Markus; KIRCHER, Michael; ZDUN, Uwe. **Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware**. Chichester: Wiley, 2004.

[3] MISHRA, Kamal Sheel; TRIPATHI, Anil Kumar. **Some Issues, Challenges and Problems of Distributed Software System**. International Journal Of Computer Science And Information Technologies. Varanasi, India, p. 4922-4925. jan. 2014.

[4] S. Neely, S. Dobson, and P. Nixon, **Adaptive middleware for autonomic systems**, Ann. Técommun., vol. 61, nos. 9–10, pp. 1099–1118, 2006.

[5] BIRRELL, Andrew D.; NELSON, Bruce Jay. **Implementing remote procedure calls**. Acm Trans. Comput. Syst., [s.l.], v. 2, n. 1, p.39-59, 1 fev. 1984. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2080.357392.

[6] ROUSE, Margaret; BOWMAN, Phil; HAZAN, Fred. **Remote Procedure Call**. Disponível em: <http://searchsoa.techtarget.com/definition/Remote-Procedure-Call>. Acesso em: 15 set. 2016.

[7] ARMSTRONG, Joe; DÄCKER, Bjarne; LINDGREN, Thomas. **Open-source Erlang - White Paper:** Erlang Overview. Elaborada pelo time Erlang na Ericsson. Disponível em: <https://web.archive.org/web/20111025022940/http://ftp.sunet.se/pub/lang/erlang/white_paper.html>. Acesso em: 16 set. 2016.

6 Assinaturas

_____

Orientador: Carlos André Guimarães Ferraz

_____

Aluno: Miguel Rodrigues Araújo