Kauê Soares da Silveira
171671

Trabalho Obrigatório 2: Comparação de Mecanismos de Comunicação

INF01151 - Sistemas Operacionais II N

Professor: Alexandre Carissimi

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Conteúdo

1 Introdução

O objetivo do trabalho é analisar a facilidade de implementação, uso e desempenho dos mecanismos de comunicação UDP / TCP, RPC e RMI. Para isto, foram implementadas diferentes versões de dois serviços (null e sort, descritos a seguir), uma para cada um dos mecanismos.

O serviço null é um serviço que não faz nada. Não recebe argumentos de entrada e não retorna resultado para o cliente [1].

Já o serviço sort recebe um vetor de 250 inteiros inicializado com a identidade e tem por objetivo ordená-lo em ordem descrescente. O vetor resultado é então retornado.

[1] Na realidade o serviço implementado não foi exatamente esse, ver Cap. 5

2 *Metodologia Empregada*

Cada medição de tempo foi feita através da média de 1000 repetições. Este processo foi efetuado 30 vezes, para cada forma de comunicação abordada, com intervalos de 1 minuto para garantir o encerramento das conexões TCP, obtendo-se a média e o desvio padrão, e permitindo calcular o intervalo de confiança (Tab. 4.3).

Para os serviços utilizando o protocolo TCP, foi considerado o tempo de estabelecimento e encerramento da conexão, assim como o tempo da requisição em si.

Todos os clientes / servidores foram escritos na linguagem de programação C, exceto os referentes ao mecanismo RMI, os quais foram escritos na linguagem Java.

3 *Descrição da Plataforma Experimental*

3.1 Cliente

Sistema Operacional (uname -a): Linux gabriela 2.6.31-19-generic #56-Ubuntu SMP Thu Jan 28 01:26:53 UTC 2010 i686 GNU/Linux

Processador (cat /proc/cpuinfo):

model name : AMD Athlon(tm) 64 Processor 3000+

cpu MHz : 1999.843

cache size : 512 KB

Memória (cat /proc/meminfo):

MemTotal : 509012 kB

Conexão : 100 mbps

JVM : java version "1.6.0_0"; OpenJDK Runtime Environment (IcedTea6 1.6.1) (6b16-1.6.1-3ubuntu3); OpenJDK Server VM (build 14.0-b16, mixed mode).

gcc :

Target : i486-linux-gnu

Thread model : posix

gcc version : 4.4.1 (Ubuntu 4.4.1-4ubuntu9)

3.2 Servidor

Sistema Operacional (uname -a): Linux liana 2.6.31-19-generic #56-Ubuntu SMP Thu Jan 28 01:26:53 UTC 2010 i686 GNU/Linux

Processador (cat /proc/cpuinfo):

model name : Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz

cpu MHz : 1596.000

cache size : 3072 KB

Memória (cat /proc/meminfo):

MemTotal : 2058944 kB

Conexão : 100 mbps

JVM : java version "1.6.0_0"; OpenJDK Runtime Environment (IcedTea6 1.6.1) (6b16-1.6.1-3ubuntu3); OpenJDK Server VM (build 14.0-b16, mixed mode).

gcc :

Target : i486-linux-gnu

Thread model : posix

gcc version : 4.4.1 (Ubuntu 4.4.1-4ubuntu9)

4 Análise

4.1 Facilidade de Implementação

Sockets tcp e udp estão num nível mais baixo de abstração, o que faz com sejam de implementação mais difícil e mais sujeita a erros. Já o RPC, por estar num nível de abstração um pouco mais alto, também é um pouco mais fácil de implementar. Seguindo esta mesma linha de raciocínio, RMI é o mecanismo com nível de abstração mais elevado e também o mais fácil de ser implementada.

4.2 Facilidade de Uso

A classificação quanto á facilidade apresenta os mesmos critérios supramencionados, sendo RMI o mais fácil de utilizar, seguido do RPC, ficando por último as sockets TCP e UDP.

4.3 Desempenho

nome	média	desvio padrão	intervalo de 95% de confiança para a média
tcp/null	36.50	12.30	[32.10, 40.90]
tcp/sort	39.25	14.01	[34.24, 44.26]
udp/null	11.85	2.29	[11.03, 12.67]
udp/sort	13.38	2.83	[12.36, 14.39]
rpc/null/tcp	220.25	104.33	[182.94, 257.56]
rpc/null/udp	53.75	23.93	[45.19, 62.31]
rpc/sort/tcp	232.00	97.64	[197.08, 266.92]
rpc/sort/udp	60.50	17.38	[54.28, 66.72]
rmi/null	349.00	53.20	[329.97, 368.03]
rmi/sort	821.75	53.06	[802.77, 840.73]

Tabela 4.1: Média, Desvio Padrão e Intervalo de Confiança de cada um dos métodos (em ns.)

4.4 Comparação Qualitativa

nome	facilidade de uso	facilidade de implementação	nível de abstração	desempenho
udp / tcp	baixa	baixa	baixo	alto
rpc	média	média	médio	médio
rmi	alta	alta	alta	baixo

Tabela 4.2: Comparação qualitativa das abordagens

4.5 Discussão

Como espeficicado na tabela, facilidade de uso, de implementação e nível de abstração estão intimamente relacionados, e são inversamente proporcionais ao desempenho. A escolha por um ou outro método vai depender da linguagem que se pretende usar e do grau de desempenho implicado pelo domínio da aplicação que se deseja programar.

Comparando tcp com udp, é possível perceber que a segunda apresenta uma média de tempo de execução 30% menor, o que é de se esperar, visto que não é um protocolo orientado à conexão (diferentemente do tcp), e portanto acaba apresentando um overhead de execução menor.

Já comparando os desempenhos dos serviços null e sort, podemos ver que o aumento no tempo de execução ficou entre 10 e 20%, a não ser no caso de RMI em que este aumento foi de mais de 100%. Isso nos mostra que o tempo para ordenar um vetor de 250 elementos não foi significante em relação ao tempo necessário para a comunicação em si, exceto no RMI, em que a manipulação de vetores realmente se mostrou bastante onerosa comparada com a comunicação.

5 *Principais Problemas Encontrados*

Funções send (respectivamente, recv) do TCP / UDP precisam enviar (respectivamente, receber) algum dado, ou seja, não é possível implementar um servidor NULL exatamente da forma como foi especificado. Em vista disso, mas ainda com a finalidade de minimizar a comunicação, foi utilizado um parâmetro do tipo char, tanto para a chamada quanto para o retorno. Afim de preservar a justiça das comparações, parâmetros semelhantes foram utilizados nas outras implementações do serviço NULL.

O protocolo TCP prevê um status TIMED_WAIT após o pedido de término da execução e antes que esta seja efetivamente terminada. Isso gera problemas com testes automatizados de alta carga, pois eventualmente ocorre excesso de conexões e estas começam a ser recusadas. Para evitar este problema, foram introduzidas paradas periódicas de um minuto, como explicado no capítulo sobre a metodologia empregada (Cap. 2).

A escassez de documentação sobre o Java RMI fez com que houvessem alguns problemas nas implementações referentes ao mesmo, principalmente em relação ao CLASSPATH (as classes "stub" não eram "enxergadas" por cliente e servidor ao mesmo tempo). Problema resolvido após algumas tentativas, modificando a hierarquia dos arquivos das classes e reiniciando o rmiregistry.

6 Conclusão

Os diferentes mecanismos de comunicação de processos tem suas vantagens e desvantagens, assim como aplicações onde seu uso é mais ou menos adequado. Alguns dos aspectos mais relevantes destes mecanismos foram aqui comparados (facilidade de implementação, uso e desempenho).

Porém, existem outras características que devem consideradas, tais como tolerância à falhas e segurança, pois estas também têm grande importância.

É fundamental termos em mente todos estes aspectos para poder decidir com confiança qual dos mecanismos é o mais adequado para nosso software, levando em consideração o contexto em que este estará executando.