REMOTE PROCEDURE CALL – CONCEITO E EXEMPLO

ERIC MARQUES, JONATHAN SANTIAGO

erc_m@hotmail.com, jonathansilvajss@gmail.com

Sistemas Distribuídos – Alexandre Zamberlan

Universidade Franciscana (UFN)

2 CONCEITO

- É uma tecnologia para criação de programas distribuídos cliente/servidor.
- Provê um paradigma de comunicação de alto nível no sistema operacional.
- Necessita da existência de um protocolo de transporte (exemplo TCP/IP ou UDP).
- Possibilita a utilização de procedimentos remotos como se fossem chamadas locais.
- Não precisa controlar possíveis diferenças (arquiteturas diferentes entre cliente e servidor).

3 HISTÓRIA

- O RPC é datado de 1976.
- Primeiro uso comercial pela *Xerox* em 1981.
- Primeira implementação popular para Unix foi o Sun RPC (atual ONC RPC).
- Network Computing System (NCS) Implementação pioneira em Unix, posteriormente utilizada como fundação do DCE/RPC no Distributed Computing Environment (DCE).
- Uma década depois a Microsoft adotou como base o DCE/RPC para criar o MSRPC.
- De forma análoga, atualmente utiliza-se XML e JSON como linguagem de descrição de interface e HTTP como protocolo de rede para formar serviços web, cujas implementações incluem SOAP, REST e XML-RPC.

4 FUNCIONAMENTO

- Similar a uma chamada local.
  - Argumento passados para o procedimento por um local específico e o controle é passado ao processo.
  - O controle retorna ao processo que disparou o procedimento e a execução continua.
- O RPC trabalha de forma semelhante.
  - O controle muda entre o processo cliente e o processo servidor.
  - Primeiro o processo cliente envia a mensagem que contém os parâmetros do procedimento para o servidor, e fica aguardando uma resposta.

5 FUNCIONAMENTO

- No servidor o processo que estava inativo até a chegada da mensagem, extrai os parâmetros, computa os resultados e posteriormente envia uma mensagem de resposta e aguarda a mensagem seguinte.
- Cliente recebe a resposta, extrai os resultados da mensagem e resume sua execução.

- Nesse modelo apenas um dos dois processos (cliente e servidor), estão ativos em determinado momento
- O RPC não faz restrições à implementação de um modelo concorrente.
- Uma implementação pode ser assíncrona.

6 VANTAGENS E LIMITAÇÕES

- Vantagens
  - facilita o desenvolvimento de sistemas distribuídos pois permite a utilização de semânticas claras e diretas.
  - Facilita a transferência do gasto computacional para os servidores.
  - Diminui a necessidade de reescrita de código.
- Limitações
  - O RPC não segue necessariamente um padrão.
  - Existem inúmeras implementações que não são compatíveis entre si, e como as mensagens são enviadas através de uma rede de comunicação, existem atrasos e até possíveis falhas na interação.

7 Exemplos - addsub(C)

```
/* addsub.x : definição da interface */

#define PROGRAM_NUMBER 12345678
#define VERSION_NUMBER 1

/* tipo de dado que será passado aos procedimentos remotos */

struct operands{
        int x;
        int y;
};
```

Exemplos - addsub(C) - continuação

```
/* Definição da interface que será oferecida aos clientes */
program ADDSUB_PROG{
   version ADDSUB_VERSION {
     int ADD (operands) = 1;
     int SUB (operands) = 2;
   } = VERSION_NUMBER;
}= PROGRAM_NUMBER;
```

Exemplos - Server(C)

```
/* Cliente RPC*/

#include <stdio.h>
#include "addsub.h"

/* implementação da função add */
int * add_1_svc (operands *argp, struct svc_req *rqstp){
   static int result;
   printf ("Recebi chamado: add %d %d\n", argp->x, argp->y);
   result = argp->x + argp->y;
   return (&result);
}
```

Exemplos - Server(C) - continuação

```
/* implementação da função sub */
int * sub_1_svc (operands *argp, struct svc_req *rqstp){
   static int result;

   printf ("Recebi chamado: sub %d %d\n", argp->x, argp->y);
   result = argp->x - argp->y;
   return (&result);
}
```

Exemplos - Client(C)

```
/* Cliente RPC*/

#include <stdio.h>
#include "addsub.h"

/* função que chama a RPC add_1 */
int add (CLIENT *clnt, int x, int y){
   operands ops;
   int *result;
```

Exemplos - Client(C) - continuação

```
 /* junta os parâmetros em um struct */
   ops.x = x;
   ops.y = y;

   /* chama a função remota */
   result = add_1 (&ops,clnt);
   if (result == NULL)    {
     printf ("Problemas ao chamar a função remota\n");
     exit (1);
   }
   return (*result);
}
```

Exemplos - Client(C) - continuação

```
/* função que chama a RPC sub_1 */
int sub (CLIENT *clnt, int x, int y){
   operands ops;
   int *result;

   /* junta os parâmetros em um struct */
   ops.x = x;
   ops.y = y;
```

Exemplos - Client(C) - continuação

```
/* chama a função remota */
   result = sub_1 (&ops,clnt);
   if (result == NULL)   {
      printf ("Problemas ao chamar a função remota\n");
      exit (1);
   }
   return (*result);
}
```

Exemplos - Client(C) - continuação

```
int main( int argc, char *argv[]){
   CLIENT *clnt;
   int x,y;
   /* verifica se o cliente foi chamado corretamente */
   if (argc!=4)   {
      fprintf (stderr,"Usage: %s hostname num1  num2\n",argv[0]);
      exit (1);
   }
```

16 Exemplos - Client(C) - continuação

```
/* cria uma struct CLIENT que referencia uma interface RPC */
   clnt = clnt_create (argv[1], ADDSUB_PROG, ADDSUB_VERSION, "udp");

   /* verifica se a referência foi criada */
   if (clnt == (CLIENT *) NULL)   {
      clnt_pcreateerror (argv[1]);
      exit(1);
   }
```

Exemplos - Client(C) - continuação

```
  /* obtém os dois inteiros que serão passados via RPC */
   x = atoi (argv[2]);
   y = atoi (argv[3]);

   /* executa os procedimentos remotos */
   printf ("%d + %d = %d\n", x, y, add (clnt,x,y));
   printf ("%d - %d = %d\n", x, y, sub (clnt,x,y));

   return (0);
}
```

Exemplos - Server(Python)

```
import datetime, xmlrpc.client
from xmlrpc.server import SimpleXMLRPCServer

def today():
    today = datetime.datetime.today()
    return xmlrpc.client.DateTime(today)

server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")
server.register_function(today, "today")
server.serve_forever()
```

19 Exemplos - Client(Python)

```
import xmlrpc.client, datetime

proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")

today = proxy.today()
# convert the ISO8601 string to a datetime object
converted = datetime.datetime.strptime(today.value, "%Y%m%dT%H:%M:%S")
print("Today: %s" % converted.strftime("%d.%m.%Y, %H:%M:%S"))
```

20 REFERÊNCIAS

VARGAS P. Kayser. Chamada de procedimento (2019). Disponível em <https://pt.wikipedia.org/wiki/Chamada_de_procedimento_remoto>. Acessado em: 04/2019.

Hoekstra, M.H., Machado, P. Remote Procedure Call. Disponível em <http://www.deinfo.uepg.br/~alunoso/2017/RPC/oquee.html>. Acessado em: 04/2019.

Milhorim, D. Exemplo de uso de RPC. Disponível em <http://www.professordiovani.com.br/sd/exemplo_rpc.htm>. Acessado em 04/2019.