Llamadas a Procedimientos Remotos

Virginia Padilla

Universidad Nacional Experimental de Guayana

virginiapadillas@gmail.com

9 de julio de 2021

Contenido

RPC

Tema principal

Se analizan tres aspectos que son importante para comprender este concepto:

- el estilo de programación promovido por RPC - programación con interfaces;
- la semántica de la llamada asociada con RPC;
- la cuestión clave de la transparencia y su relación con las llamadas a procedimientos remotos.

Programación con Interfases

Interfases

- La interfaz de un módulo especifica los procedimientos y las variables a las que se pueden acceder a través de otros módulos.
- En sistemas distribuidos son programas distribuidos donde los módulos se ejecutan en procesos separados.
- En cliente-servidor cada servidor proporciona un conjunto de procedimientos que están disponibles para uso de los clientes.

Programación con Interfases

Programación con Interfases: Implementación

- No hay detalle de implementación lenguaje de programación.
- No hay acceso a variables mediante ejecución de procesos remotos ni mecanismos pase de parámetros
- Direcciones en procesos locales no son válidas en los procesos remotos
- Mecanismo RPC se integra con el lenguaje de programación e incluye la notación para la definición de interfaces con mensajes input/output
- Escrita en variedades de lenguajes, C++, Java, Python
- Lenguajes de definición de interfaces (IDL) están diseñados para permitir que los procedimientos implementados en distintos lenguajes puedan ser invocados por otros

Ejemplo: Corba IDL

Corba IDL

- Ejemplo en el IDL `CORBA`.

```
// In file Person.idl
struct Person {
        string name;
        string place;
        long year;
};
interface PersonList {
        readonly attribute string listname;
        void addPerson(in Person p) ;
        void getPerson(in string name, out Person p);
        long number();
};
```

- *PersonList* especifica los métodos disponibles para la invocación remota en el objeto remoto que implementa esa interfase.
- *addPerson* especifica sus argumentos como argumentos de entrada
- *getPerson* busca una instancia de *Person* por el nombre establecido en su segundo argumento de salida.

Semántica de llamadas de RPC

doOperation puede dar garantías de entrega en el mensaje:

- Reintentar mensaje de solicitud:si retransmisión del mensaje de solicitud hasta se recibe una respuesta o supone falla del servidor.
- Filtrado duplicado: si se usan las retransmisiones y si se debe filtrar solicitudes duplicadas en el servidor.
- Retransmisión de resultados: si se debe tener un historial de mensajes enviados para permitir que los resultados perdidos se retransmitan.

Fault tolerance measures			*Call semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

Semántica de llamadas de RPC

Semántica de llamadas de RPC

- **Puede ser**: el invocador de la llamada recibe un resultado, en cuyo caso el invocador de la llamada sabe que el procedimiento se ejecutó al menos una vez, o recibe una excepción.
- **Al menos una vez**, la semántica puede ser logrado mediante la retransmisión de mensajes de solicitud, que enmascara las fallas de omisión del mensaje de solicitud o resultado.
- **Como máximo**, la persona que llama recibe un resultado, en cuyo caso la persona que llama sabe que el procedimiento se ejecutó exactamente una vez, o una recibe una excepción, entonces el procedimiento ha sido ejecutados ya sea una vez o no lo ha sido.

Semántica de llamadas de RPC

Semántica de llamadas de RPC: Fallas

La semántica puede sufrir lo siguiente tipos de falla:

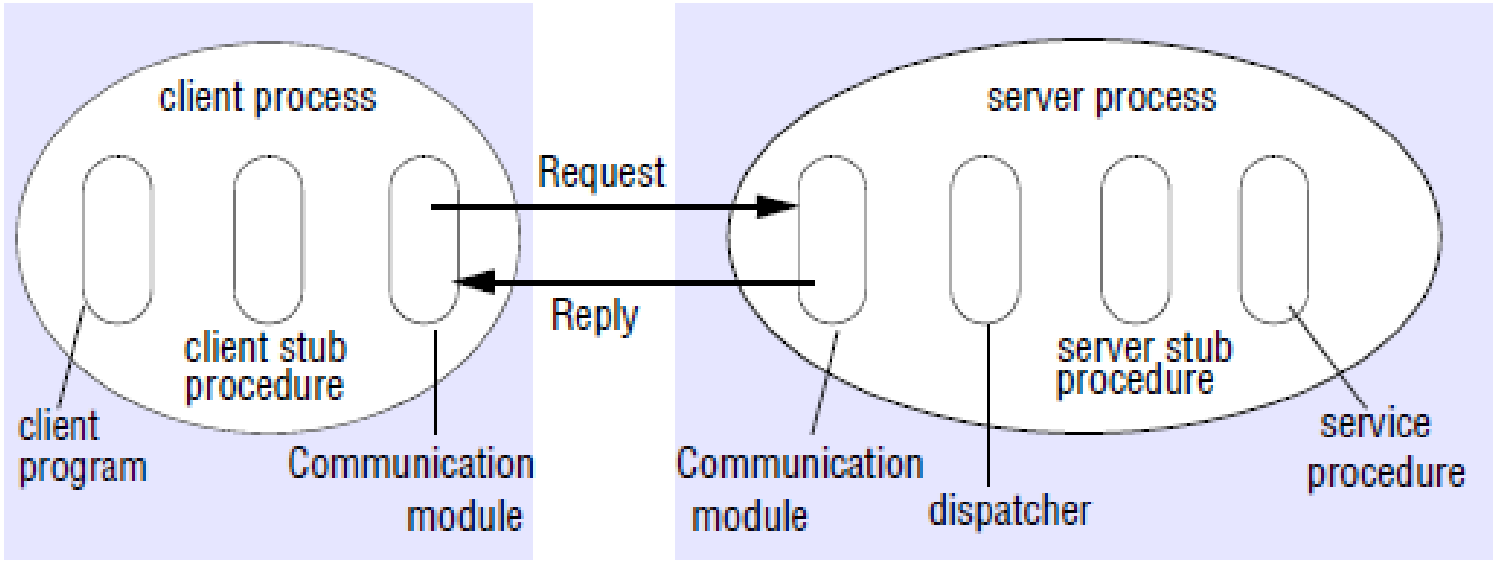
- fallas de bloqueo cuando falla el servidor que contiene el procedimiento remoto;
- fallas arbitrarias: en los casos en que el mensaje de solicitud se retransmite, el servidor puede recibirlo y ejecutar el procedimiento más de una vez, posiblemente causando valores incorrectos para ser almacenados o devueltos.

Transparencia

Transparencia

- La elección de si el RPC debe ser transparente también está disponible para los diseñadores de IDL.
- Por ejemplo, en algunos IDL, una invocación remota puede generar una excepción cuando el cliente no puede comunicarse con un procedimiento remoto. Esto requiere que el programa cliente maneje tales excepciones, permitiéndole lidiar con tales fallas.
- Un IDL también puede proporcionar una facilidad para especificar la semántica de llamada de un procedimiento.
- Esto último puede ayudar al diseñador del servicio: si se elige la semántica de llamada al menos una vez para evitar los gastos generales de una vez como máximo, las operaciones deben diseñarse para que sean idempotentes.

Arquitectura de RPC



Implementación de RPC

Implementación de RPC

- El cliente que accede a un servicio incluye un procedimiento *stub* para cada procedimiento definido en la interfaz de servicio.
- El procedimiento *stub* se comporta como un procedimiento local para el cliente, pero en lugar de ejecutar la llamada, ordena (empaqueta) el identificador del procedimiento y los argumentos en un mensaje de solicitud, que envía a través de su módulo de comunicación al servidor.
- Cuando llega el mensaje de respuesta, desarma (desempaqueta) los resultados.
- El proceso del servidor contiene un despachador junto con un procedimiento de código stub del servidor y un procedimiento de servicio para cada procedimiento en la interfaz de servicio.

Implementación de RPC

Implementación de RPC

- El despachador selecciona uno de los procedimientos *stub* del servidor, según el identificador de procedimiento en el mensaje de solicitud.
- El procedimiento de stub del servidor, desempaqueta los argumentos en el mensaje de solicitud, llama al procedimiento del servicio correspondiente y calcula los valores de retorno para el mensaje de respuesta.
- Los procedimientos de servicio implementan los procedimientos en la interfaz de servicio. Los procedimientos de stub de cliente, servidor y el despachador puede ser generado automáticamente por un compilador de interfaz a partir de la definición de interfaz del servicio.

Implementación de RPC

Implementación de RPC

- RPC se implementa generalmente sobre un protocolo solicitud-respuesta.
- El contenido de los mensajes de solicitud y respuesta es el mismo que se ilustra para los protocolos de solicitud-respuesta.
- RPC puede implementarse para tener una de las opciones de semántica de invocación discutidas, generalmente se elige al menos una vez o como máximo una vez.
- El módulo de comunicación implementará las opciones de diseño deseadas en términos de retransmisión de solicitudes, tratamiento de duplicados y retransmisión de resultados.

References

Van Steen M , Tanenbaum, A (2017)

Distributed Systems

Pearson Education, Inc.

Coulouris G, Dollimore J, Kindberg T, Blair G (2011)

Distributed Systems: Concepts and Design

Addison-Wesley Publishing Company.

Veríssimo, P. and Rodrigues, L. (2012)

Distributed Systems for System Architects

Springer.

Limoncelli T, Strata R, Hogan C. (2014)

The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems

Addison Wesley.

Fin