Unidad 2 / Escenario 4

Lectura fundamental

Arquitecturas SOA, punto a punto y multiprocesador

Contenido

Palabras clave: arquitectura, servicios, SOA, punto a punto, P2P, multi procesador

Introducción

En esta Lectura fundamental se describen las tres arquitecturas adicionales: la arquitectura orientada a servicios, en la que se exponen los métodos de comunicación necesarios para su funcionamiento. La arquitectura punto a punto, mediante la cual se explican dos tipos de modelos. La arquitectura multiprocesador, conocida por ser una de las que ha permitido que se puedan prestar servicios a gran escala en Internet y de forma hibrida.

Adicionalmente, se dan los fundamentos de la programación paralela, que es la herramienta fundamental para trabajar en ambientes distribuidos.

1. Arquitectura orientada a servicios

Esta arquitectura es comúnmente llamada SOA, que quiere decir *Service Oriented Architecture*. Es quizá una de las arquitecturas más populares, debido a la masificación de Internet.

Las aplicaciones empresariales debían evolucionar para un ambiente mucho más dinámico y más desacoplado, porque las arquitecturas anteriores cliente/servidor y objetos distribuidos estaban orientadas a aplicaciones que existían y presentaban un mejor desempeño en una ambiente local y más controlado. Sin embargo, si una organización debía comunicarse con otra, su medio de comunicación era Internet y, debido a la dinámica de comunicación de este medio, garantizar un esquema confiable a inicios de los años 2000 (Ryan, 2010) aún era extremadamente complejo y costoso para las organizaciones.

Así se crea la arquitectura orientada a servicios, donde parte de su estrategia está basada en el esquema de objetos distribuidos, en el que existe un proveedor de servicios y un receptor del mismo (Coulson & Baichoo, s.f.). Sin embargo, el esquema orientado a servicios necesita de un *middleware* o de un bus de *software* que hace que sea posible la comunicación. En la arquitectura orientada a servicios, la prestación de servicio se conoce como servicio Web.

¿Qué es un servicio Web?

Un servicio web es una suplencia estándar de un recurso publicado en algún lugar de Internet, el cual provee servicios, ya sea de tipo computacional o informacional, para que puedan ser utilizados por un usuario u otro programa de *software*. Esto quiere decir que un proveedor de servicio es un ente independiente del *software* que usa dicho servicio. Lo anterior logra el nivel de desacoplamiento necesario para garantizar la dinámica requerida por las aplicaciones.

A continuación, se describe el modelo conceptual empleado por la arquitectura orientada a servicios para garantizar su funcionamiento.

1.1. Arquitectura conceptual

La arquitectura orientada a servicios está compuesta por tres componentes fundamentales:

- Proveedor de servicios
- Solicitante de servicios
- Registro de servicios

El servicio más complejo es el proveedor de servicios, que tiene como propósito exponer una interfaz. Esta interfaz define cómo se puede consumir dicho servicio, qué parámetros necesita y cuál es la ubicación lógica del servicio dentro del servidor o contenedor, el cual es el encargado de publicar dicho servicio. Los proveedores de servicios se encargan de desarrollar servicios especializados y ofertarlos o ponerlos al servicio de cierto número de usuarios que consumen servicios de diferentes organizaciones.

Arquitectura orientada a servicios (Ciclo de vida de los servicios)

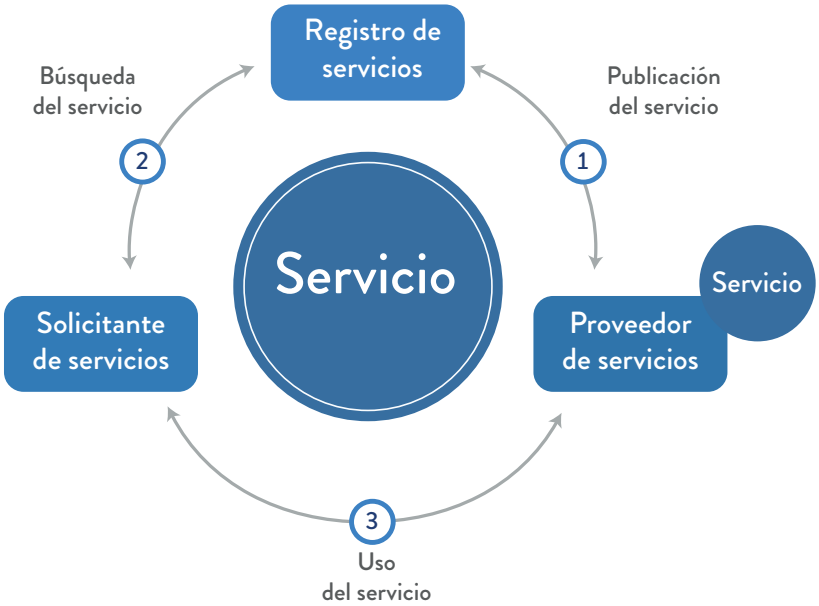


Figura 1. Arquitectura orientada a servicios

Fuente: elaboración propia

El ciclo de servicio de la Figura 1 es bastante sencillo y está compuesto de tres pasos:

1. Una vez el servicio o el proveedor de servicios está en línea, inicia el proceso de publicar el servicio en el registro de servicios. Esto tiene como propósito notificar que el proveedor de servicios se encuentra activo y está listo para recibir solicitantes.
2. Una vez el servicio está activo y registrado, es susceptible de ser encontrado por un solicitante que busque el servicio. Este paso genera como respuesta al solicitante, la ubicación del servicio y un token para validar el uso seguro del servicio.
3. El solicitante hace uso del servicio directamente en la interfaz estándar y validándose con el *token* asignado en el servicio *web*.

Una de las características más importante de esta arquitectura es la estandarización en el proceso de definición de interfaces y definición de funcionalidades del servicio mismo; esta característica permite no solo el desacoplamiento de la aplicación sino de la flexibilidad de la misma, ya que habilita una capacidad de heterogeneidad bastante alta. Lo anterior se debe a que la interfaz estandarizada permite que tanto proveedores de servicios como solicitantes puedan desarrollarse en casi cualquier lenguaje, mientras cumplan con el estándar definido.

1.2. Diferencias entre servicios y objetos

Una de las diferencias más fuertes entre objetos y servicios es la posibilidad que tienen los servicios de ser interorganizacionales, esto quiere decir que el proveedor de servicios dentro del esquema SOA, puede prestarles servicios a clientes, tanto dentro como fuera de su propia organización.

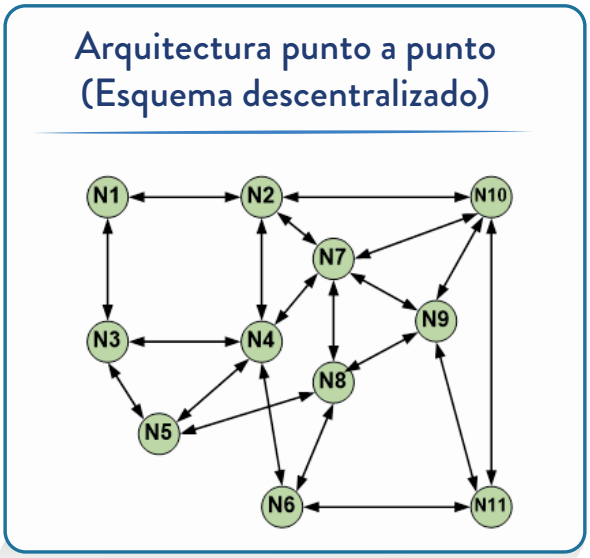
Esto también les da ventajas considerables a los clientes o a las aplicaciones con las que interactúa el usuario final debido a que estas aplicaciones pueden integrar varios servicios de diferentes organizaciones con el fin de poder reutilizar funcionalidades ya previamente desarrolladas. Por ejemplo, en la actualidad existe un modelo de negocio muy lucrativo el cual se basa en servicios *web*, y es el de las API. Las API son librerías previamente diseñadas para un propósito específico, con la única diferencia de que las funcionalidades están expuestas para que se puedan utilizar, cobrando al usuario por su uso, ya sea individual o por cantidad de transacciones, como es el caso de IBM Bluemix (Stifani, s.f.).

2. Arquitectura punto a punto

La arquitectura punto a punto o P2P es un sistema naturalmente descentralizado. Su característica principal es que el sistema es altamente colaborativo y generalmente sirve para compartir información o recursos. Es decir que cada uno de los participantes es cliente y servidor al tiempo. Sin embargo, cabe aclarar que en esta arquitectura no hay distinción entre clientes y servidores, solo participantes. Por otro lado, en cuanto a la estructura, cuando un participante o nodo ingresa al sistema no existe jerarquía alguna para la coordinación del mismo.

En aplicaciones P2P, el sistema es diseñado teniendo en cuenta que se debe aprovechar la red de computadores que pueden tener una escala muy grande. El objetivo es aprovechar la fuerza computacional y la disponibilidad de almacenamiento de las que disponen estas grandes centrales informáticas, usando los protocolos y los estándares que ya existen para facilitar las comunicaciones entre los nodos, dado que los sistemas distribuidos son esclavos de las comunicaciones. Sin sistema de comunicaciones no hay sistemas distribuidos. Para que la operación funcione en cada uno de los nodos se debe ejecutar una copia del *software* que compone la aplicación.

Las técnicas punto a punto *(peer-to-peer)* son y fueron usadas para sistemas *stand alone* personalizados, pero hoy aún se usas en redes de pc. También se usan mucho en *software* cooperativo que tolera procesos distribuidos.

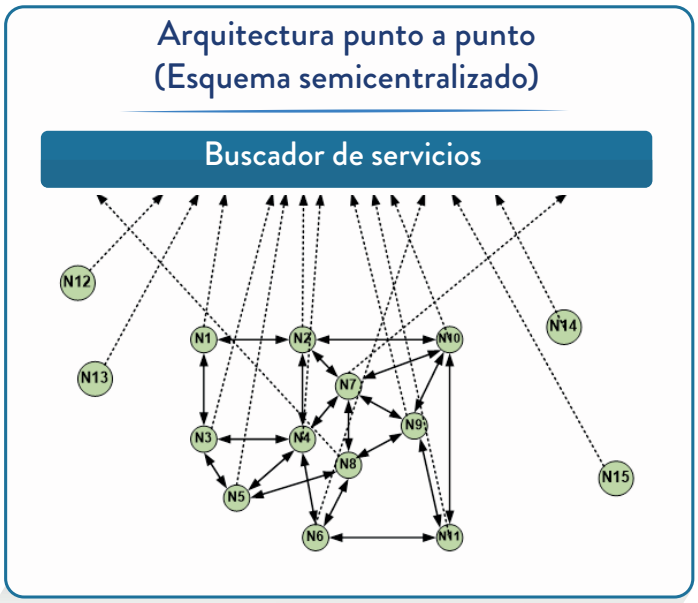


Las redes *peer-to-peer* tienen la gran dificultad para encontrar información, dado que no está **centralizada** y como tal tienen que buscar nodo por nodo, utilizando otros nodos como puente y resulta demasiado caro y se vuelve una tarea desgastante.

Observe la Figura 2 y suponga que N1 necesita un archivo que está en N11. Para alcanzarlo debe abrirse camino por N2 y N3 y aplicar un algoritmo de ruta crítica para encontrar la ruta más económica que es N2->N10->N11.

Figura 2. Arquitectura punto a punto – Modelo descentralizado

Fuente: elaboración propia



La arquitectura punto a punto tiene como ventajas que es tolerante a fallos y muy redundante. Si un nodo se desconecta de otro nodo de la red, se puede reconectar buscando otra ruta, pero esto implica sobrecarga en los demás nodos y en la red porque muchos otros nodos deben asumir un proceso de búsqueda más riguroso con los respectivos costos de comunicaciones.

Arquitectura P2P semi-centralizada. Es una arquitectura en la cual más de un nodo actúa como servidor de servicios, lo que facilita las comunicaciones entre los demás nodos.

Figura 3. Arquitectura punto a punto – Modelo semi-centralizado
Fuente: elaboración propia

En una arquitectura semi-centralizada, el papel del servidor es ayudar a establecer contacto entre iguales en la red o para coordinar los resultados de un cálculo.

Por ejemplo, si la Figura 3 representa un sistema de mensajería instantánea, entonces los nodos de la red se comunican con el computador (maestro) que desempeña las funciones de buscador de servicios (líneas punteadas), el cual indica cuales nodos están libres. Cuando los nodos libres son encontrados, se establece comunicación directa y ya no se requiere la comunicación con el servidor maestro. En la Figura 3, también se observa que los nodos N12, N13, N14 y N15, tienen comunicación directa.

En esta arquitectura P2P, cuando se usa para procesamiento masivo o procesamiento a gran escala, algunos se encargan de distribuir el trabajo, otros reúnen y comprueban resultados y la gran mayoría de nodos son dedicados al procesamiento de datos. Este tipo de arquitectura presenta mucha sobrecarga, sobre todo cuando se tiene que saltar de un nodo a otro en busca de información. El sistema se vuelve lento, la red se sobrecarga y la autenticidad y seguridad no están resueltas completamente.

Los sistemas distribuidos P2P no son usados para aplicaciones críticas, se usan con sistemas de baja critica.

3. Arquitectura multiprocesador

Las arquitecturas multi procesador se clasifican en dos categorías: fuertemente acopladas y débilmente acopladas.

Este tipo de arquitecturas se usan tanto para procesamiento a gran escala como para procesamiento particular. Un portátil i7 es una arquitectura fuertemente acoplada porque todos sus *cores* están integrados al computador y se pueden desarrollar procesos usando hilos con *Parallel threats* (Figura 5).

Existen servidores de más de un procesador, donde cada procesador tiene más de un core (núcleo). Por ejemplo, un servidor de 8 procesadores, con 36 *cores* por procesador. En tal caso, es mucha la capacidad de cálculo que se tiene.

Por otro lado, las Figuras 4 y 6 representan arquitecturas multiprocesador **débilmente** acopladas, porque los procesadores están en cada uno de los servidores y para comunicar los procesos deben usar la **red** de datos.

Una organización computacional de un millón de servidores, donde cada servidor tenga 8 procesadores, y cada procesador tenga 64 *cores* es una organización computacional muy poderosa. Un ejemplo es el centro de supercomputación más poderoso del mundo, ubicado en Estados Unidos: el DOE/SC/Oak Ridge National Laboratory.

Las organizaciones computacionales débilmente acopladas usan un agendador para organizar las tareas, y un despachador para enviar los procesos desde el servidor maestro al servidor que va a realizar la tarea (Figura 4), para poder organizar los procesos y llevar el control de la ejecución. No sobra decir que usar muchos procesadores de muchos cores mejora en gran magnitud la capacidad de cálculo.

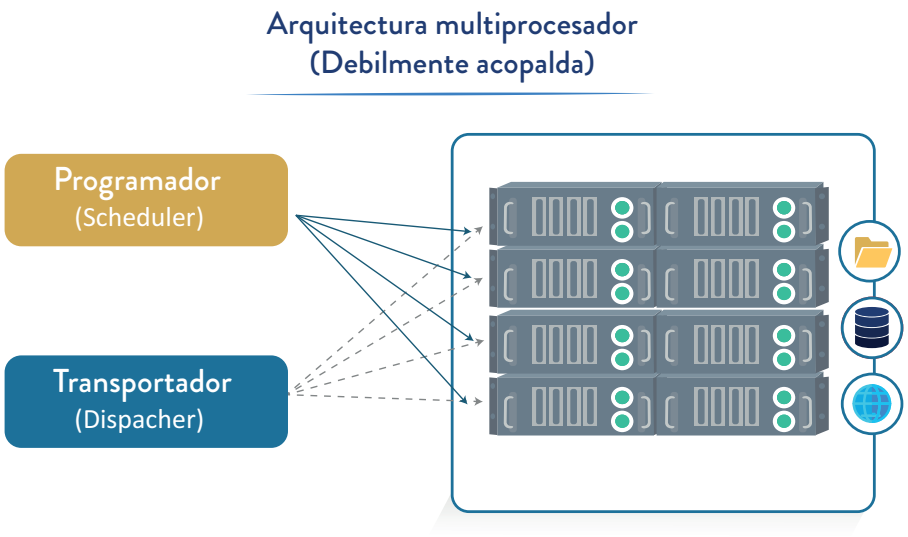


Figura 4. Arquitectura multiproceso débilmente acoplada

Fuente: elaboración propia

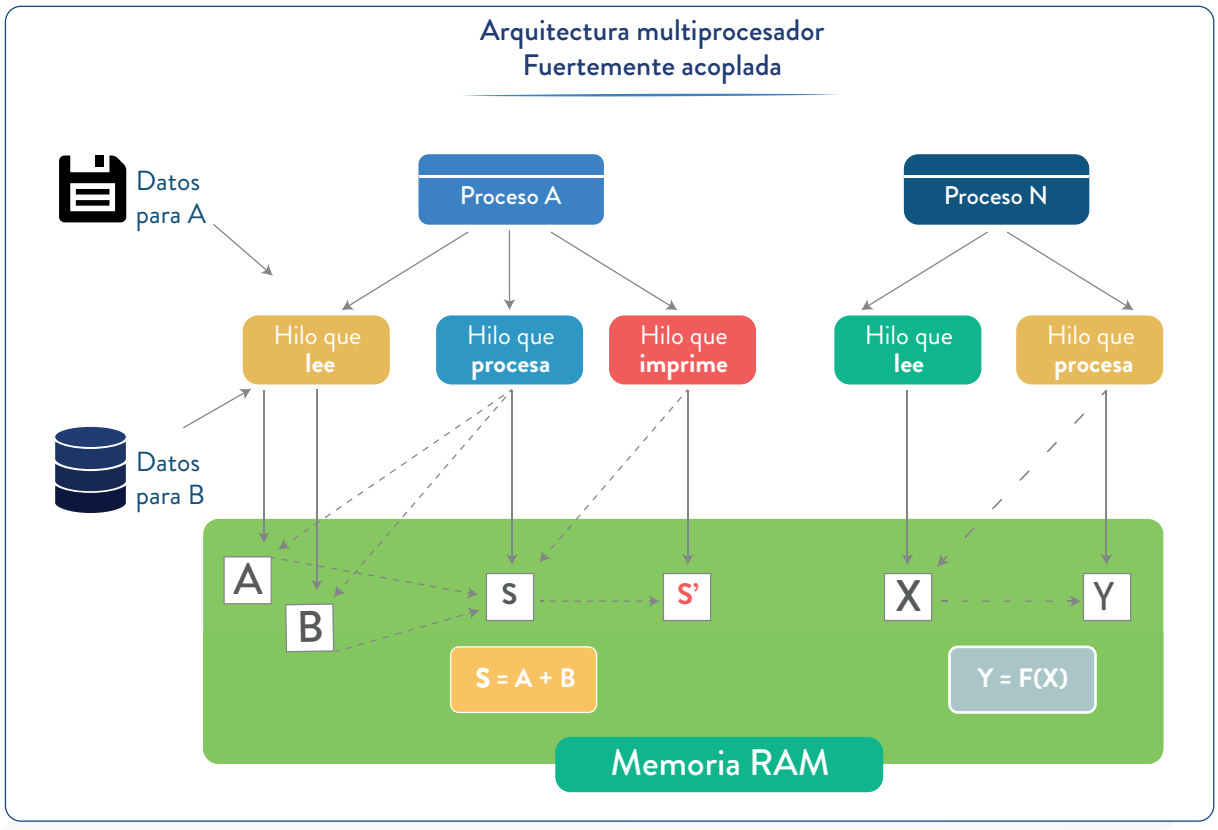


Figura 5. Arquitectura multiproceso fuertemente acoplada

Fuente: elaboración propia

4. Fundamentos de programación paralela

Con la llegada de los computadores *multicore* y las tecnologías de *Cloud Computing*, la computación en paralelo ha tomado relevancia. No es un área específica en el mundo de la computación, pero si es una de las mejores alternativas para lograr eficiencia, mejor uso de los recursos, ahorro de energía, bajar la temperatura, aumentar la velocidad de frecuencia de reloj, etc.

Anteriormente, para aumentar el rendimiento de una aplicación bastaba con moverla a una máquina con un procesador más rápido. Sin embargo, hoy en día esto ya no vale, pues el rendimiento se obtiene a través de sacarle partido a más núcleos y no aumentando la velocidad.

A partir de 1986 - 2002, los microprocesadores se aceleraron como un cohete, lo que aumentó el rendimiento en un promedio del 50% anual. Después del 2002, esta aceleración se redujo en el rendimiento de los procesadores, en un 20% por año. Este cambio tan dramático se asoció con el diseño de los microprocesadores.

Entonces, en lugar de diseñar y construir microprocesadores más rápidos, se decidió poner varios procesadores en un solo circuito integrado.

Actualmente, la potencia computacional va en aumento, pero también los problemas y necesidades de computación. Los problemas que se pensaban irresolubles con la programación en serie, han sido resueltos por las tecnologías paralelas, como la decodificación del genoma humano. Sin embargo, los problemas más complejos todavía están esperando a ser resueltos.

¿Como hacer programas paralelos?

1. Por paralelismo de tareas; se divide la tarea en subtareas y así se resuelve el problema con apoyo de los núcleos.
2. Por paralelismo de datos; se hace una partición de los datos para resolver el problema con el apoyo de los núcleos. Cada núcleo lleva a cabo operaciones similares con su parte de datos.

Suponga que el profesor P tiene que dictar una cátedra denominada Literatura inglesa, que tiene 300 estudiantes y llega la hora de evaluar. El profesor P cuenta con tres asistentes, y se deben calificar 300 exámenes de 15 preguntas cada uno. Para realizar la tarea dispone de 1000 minutos y calificando cada examen se demora 10 minutos. Usa sus tres asistentes/cores y le asigna 100 exámenes a cada uno. El profesor hace de agendador y transportador.

Por paralelización de datos, se usa el mismo programa en varios servidores con diferentes bloques de datos. Se parte la tarea principal en subtareas que se resuelven con el mismo programa ejecutado en varios *cores*. Para el caso del profesor, se asignan las preguntas 1 a 5 al primer asistente, 6 a 10 para el segundo asistente y 11 a 15 para el tercer asistente. Todos califican los 300 exámenes, pero solo hacen su segmento de preguntas, Cuando el primer asistente califica el primer examen, se lo envía al segundo asistente para que califique sus preguntas y este, a su vez, cuando termina de calificar sus preguntas, le envía el examen al tercer asistente para que califique las últimas cinco preguntas. Una vez se cierra el ciclo, se tiene un proceso en paralelo, secuencializado, que requiere de un nivel alto de sincronización.

Figura 6. Datacenter multiprocesador para computación paralela, débilmente acoplada
Fuente: Bsdrouin

Referencias bibliográficas

Coulson, G. y Baichoo, S. (s.f.). *A distributed object platform for multimedia applications. IEEE International Conference on Multimedia Computing and Systems 2*, 122–126. Recuperado de: https://doi.org/10.1109/MMCS.1999.778193

Ryan, J. (2010). *A history of the Internet and the digital future. Reaktion Books*. Recuperado de: https://books.google.com.co/books/about/A_History_of_the_Internet_and_the_Digita.html?id=I0OYhHefumoC&redir_esc=y&hl=en

Stifani, R. (s.f.). *IBM Bluemix The Cloud Platform for Creating and Delivering Applications.* Recuperado de: https://www.redbooks.ibm.com/redpapers/pdfs/redp5242.pdf

Referencias de imágenes

Vaeenma (s.f.). Centro de computo de altas prestaciones (Datacenter). [Fotografía]. Recuperado
de https://www.gettyimages.es/detail/foto/medical-network-diagram-imagen-libre-de-derechos/517851799

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Sistema Distribuidos

Unidad 2: Arquitecturas cliente-servidor y objetos distribuidos y arquitecturas SOA, punto a punto y multiprocesador

Escenario 4: Arquitecturas SOA, punto a punto y multiprocesador

Autor: Alexis Rojas

Asesor Pedagógico: Jeimy Lorena Romero Perilla

Diseñador Gráfico: Brandon Steven Ramírez Carrero

Asistente: Maria Elizabeth Avilán Forero