Unidad 1 / Escenario 2

Lectura fundamental

Diseño de los sistemas distribuidos

Contenido

Palabras clave: diseño, transparencia, flexibilidad, confiabilidad, desempeño, escalabilidad

Introducción

Esta Lectura fundamental está compuesta por un único tema: la construcción y diseño de los sistemas distribuidos, que tiene como objetivo garantizar mecanismos de calidad y control a las aplicaciones a las cuales los componentes adicionales prestarán sus servicios.

Antecedentes

La construcción y el diseño de un servicio tecnológico, que será empleado por usuarios o por otros sistemas, de forma masiva y distribuida geográficamente o no, no es tarea fácil, pues es necesario tener en cuenta variables en cuanto a la prestación del servicio y a la operación del mismo.

Muchas de las problemáticas que los profesionales en ingeniería de sistemas están llamados a solucionar dentro del sector empresarial están orientadas al diseño y desarrollo de software. Sin embargo, el diseño de los sistemas que le dan soporte a estos aplicativos, una vez ya están desarrollados, es un proceso que, por lo general, se toma a la ligera o al que no se le presta la suficiente atención, al menos en los momentos iniciales de las fases de despliegue de un proyecto.

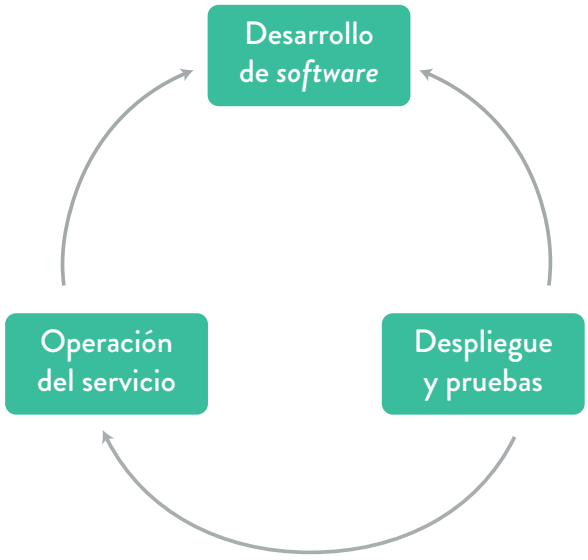

Figura 1. Macroprocesos de la creación de un servicio de tecnología
Fuente: elaboración propia

Dentro del proceso de elaboración se puede ver, inicialmente, el ciclo de vida clásico de desarrollo de *software*. Una vez el desarrollo está completo, se procede al despliegue en servidores de prueba, para finalizar con la operación y la prestación del servicio hacia el público objetivo.

1. Diseño de los sistemas distribuidos

Ningún sistema es igual a otro y esto se debe a la naturaleza del servicio que cada uno presta y a los aspectos que conforman su diseño. Comprender estos aspectos significa comprender el sistema y sus características, no relacionadas con la funcionalidad del *software* o servicio, sino con lo que se conoce como a los requerimientos no funcionales de una aplicación (Pressman, 2005).

Estos aspectos de diseño son:

- Flexibilidad
- Confiabilidad
- Desempeño
- Escalabilidad
- Transparencia

Ningún aspecto es más importante que otro, pero suele ser un reto poder garantizar los cinco aspectos. Por eso se recomienda que al momento del levantamiento de la información sobre los requerimientos del sistema distribuido se defina cuál de estos tiene mayor importancia o cual tiene prioridad al momento de prestar el servicio a los usuarios finales del sistema.

Estos aspectos de diseño aportan varios lineamientos para la construcción del sistema, ya sea, tanto de la capa física como de la capa lógica, es decir de los componentes 1 y 2 del sistema distribuido: equipos de cómputo y *hardware* adicional, y middleware y *software* transversal.

Es así como estos lineamientos pueden partir de preguntas sencillas como:

- ¿De cuántos componentes físicos y lógicos está compuesto el sistema?
- ¿Cómo están interactuando los componentes físicos?
- ¿Cuál es el nivel de servicio de la aplicación?
- ¿Hay alguna preparación en caso de dificultad o desastre?
- ¿Cuántos usuarios soporta el sistema?

1.1. Flexibilidad

La flexibilidad atraviesa varias características de los sistemas distribuidos, desde lo más básico de los componentes. Como ya sabe, existen muchos fabricantes de equipos, y estos, generalmente, deben seguir ciertos estándares y ser compatibles con diferentes sistemas operativos, esta capacidad se llama **heterogeneidad**.

De la misma manera, en el caso del software, se espera que el sistema tenga la habilidad de interactuar y comunicarse con otros sistemas, de diferentes proveedores; esta habilidad se conoce como **apertura**.

Y, finalmente, así como un sistema debe estar en constante desarrollo, el sistema distribuido debe tener la capacidad de tener nuevas funcionalidades o capacidades; esto se conoce como capacidad de **evolución**.

1.1.1. Heterogeneidad

En el Escenario 1 se estudió la arquitectura Von Neuman (Godfrey & Hendry, 1993), que ha permitido estandarizar las características de un computador, y el ordenamiento de las máquinas disponibles en el mercado de acuerdo con sus características grandes o pequeñas, y sus especificaciones.

Otro aspecto fundamental que refleja la heterogeneidad es la capacidad de un sistema para soportar diferentes sistemas operativos, tal como se mencionó anteriormente.

1.1.2. Apertura

Significa que el sistema es capaz de intercambiar información y servicios con otros sistemas diferentes, como es el caso de Facebook y Google en su papel de proveedores de identidad. Esto ocurre cuando una aplicación externa, al conectarse con estos sistemas, pone en práctica ciertos protocolos orientados a la autenticación, de acuerdo con el protocolo establecido.

Esta característica es muy importante, pues ha permitido la evolución de Internet en los últimos 20 años, en la medida en que, a partir de instrucciones estandarizadas, cualquier persona puede construir un servicio e interactuar con el protocolo con el cual fue construido, dando lugar al *software* libre.

1.1.3. Capacidad de evolución

Tal y como lo indica el ciclo de vida del desarrollo del *software* (Pressman, 2005), el sistema debe estar en constante evolución y crecimiento. El sistema distribuido debe tener la capacidad de aceptar nuevas funciones o cambios en las que ya tiene establecidas a lo largo del tiempo. Es decir, un sistema distribuido no es un sistema estático, sino que evoluciona orgánicamente para realizar nuevos procesos e intercambios con otras funcionalidades y sistemas externos de la organización.

1.2. Confiabilidad

Puede entenderse como la capacidad de un sistema de cómputo para seguir activo después de un tiempo determinado. Tal característica también se conoce como disponibilidad. Sin embargo, un sistema complejo con muchos componentes y subsistemas tiene una alta probabilidad de fallo y por esto, otra característica de la confiabilidad es la capacidad de recuperarse ante las fallas, que se conoce como tolerancia a fallos.

1.2.1. Disponibilidad

En un sistema informático, en cualquier contexto, está dada según el tiempo de operación continuo sin fallos que puede tener; este tiempo está medido en porcentaje (%), en un rango que va desde 0%, que sería total indisponibilidad en el año, hasta 100%, que significaría total disponibilidad del servicio de forma ininterrumpida en el año.

Para medir la disponibilidad del sistema informático y verificar que es capaz de ejecutar las funciones para lo que fue diseñado, se hace el cálculo del tiempo en porcentaje mediante la siguiente fórmula:

$$\%D = \frac{Tt - \sum_1^f Tf}{Tt}$$

Donde cada una de las variables son:

%D: porcentaje de disponibilidad

Tt: tiempo total del servicio (generalmente equivale a un año – 31.536.000 seg.)

f: número total de fallos en el tiempo total del servicio

Tf: tiempo individual, en segundos, de los fallos ocurridos

Cuando los servicios informáticos necesitan múltiples computadores para su funcionamiento, requieren una infraestructura adicional, la cual se presta, en la mayoría de los casos, desde un Data Center. Los Data Center, según la norma TIA-942 (Telecommunication Industry Association, 2005), están clasificados de uno a cuatro, siendo uno el más bajo y cuatro el más alto.

Aunque la norma es mucho más explícita en cuanto a los componentes en lo que se refiere a medidas de seguridad perimetral, instalación, prevención y atención de desastres, entre otros componentes, aquí se enfocan los requerimientos de seguridad relacionada con los equipos de cómputo.

- **TIER I:** ofrece el nivel más básico de protección y ninguna redundancia para la infraestructura de misión crítica de la organización.
- **TIER II:** necesita cumplir con las medidas anteriores y, adicionalmente, con capacidad redundante en los sistemas críticos y aislamiento para dichos componentes.
- **TIER III:** cumplir con las medidas del TIER II y, adicionalmente, todos los equipos deben tener redundancia a nivel eléctrico, mecanismos de generación de energía o fuente de energía alterna.
- **TIER IV:** cumplir con las medidas del TIER III y, adicionalmente, múltiples y distintos caminos eléctricos, de servicio y red, cambio automático para momentos de falla a los sistemas de respaldo, y enfriamiento continúo para todos los equipos en el sitio.

Estas son algunas de las características de los centros de datos comúnmente llamados *Data Center*. Según su categoría, a continuación, se expone una tabla en la cual se observan más datos relevantes según el *ranking* previamente expuesto.

Tabla 1. Niveles de disponibilidad según nivel del centro de datos

Nivel del Centro de Datos	% de Disponibilidad	Tiempo en el que el centro de datos NO está disponible
Tier I	99671%	28.82 horas
Tier II	99741%	22.68 horas
Tier III	99982%	1.57 horas
Tier IV	99995%	52.56 minutos

Fuente: elaboración propia

Se puede observar el porcentaje de disponibilidad del *Data Center*, según su clasificación en TIER y también el tiempo de indisponibilidad máximo que puede tener, según su *ranking*.

1.2.2. Tolerancia a fallos

La tolerancia a fallos se puede definir como la capacidad que tiene el sistema de sostener su operatividad al momento de enfrentar una falla súbita.

Para poder incrementar y mejorar la tolerancia de un sistema es necesaria la **redundancia**, dada por una cantidad determinada de equipos de cómputo configurados que se encuentran listos para actuar según el nivel de redundancia necesaria. En la mayoría de los casos, estos equipos de cómputo están alojados dentro de un Data Center y cuentan con el respaldo adicional de las configuraciones efectuadas en los TIERS.

A continuación, se presenta una tabla con los niveles de redundancia existentes según la necesidad de tolerancia deseada.

Tabla 2. Niveles de redundancia de sistemas informáticos

Nombre de Redundancia	Ecuación de Redundancia
Configuración de Capacidad	N
Redundancia Aislada	N + 1
Redundancia en paralelo	N + 1
Redundancia Sistema * Sistema	2 N
Doble redundancia en paralelo	2 (N + 1)
Doble redundancia distribuida	2 (N + 1)

Fuente: elaboración propia

A continuación, se explica cada uno de los niveles de redundancia y en qué se diferencian unos de otros.

- **Configuración de capacidad (N):** es la configuración básica necesaria para la prestación del servicio, sin ningún componente adicional. Es la capacidad mínima necesaria para mantener el servicio, pero ante el requerimiento de cualquier mantenimiento es necesario crear una interrupción de servicio. Generalmente, se aplican en cuartos de datos artesanales o *Data Center* tipo TIER I.
- **Redundancia aislada (N+1):** esta es una de las configuraciones mínimas para la redundancia y se conoce como esquema Activo-Pasivo. En este nivel se requiere otro componente adicional al mínimo, en stand-by, el cual entra a operar como servicio principal, disminuyendo considerablemente la cantidad de tiempo de indisponibilidad. Generalmente, se aplican en cuartos de datos artesanales o *Data Center* tipo TIER II.
- **Redundancia en paralelo (N+1):** esta configuración se conoce como esquema Activo-Activo, de modo que ambos componentes se encuentran prestando el servicio de forma sincronizada, lo que significa que existe capacidad adicional prestando el servicio. En caso de fallo, toda la carga pasaría al componente que continua activo, y así se garantiza que el servicio se siga prestando. Generalmente, se aplican en cuartos de datos artesanales o *Data Center* tipo TIER II y III.
- **Redundancia sistema-sistema (2N):** este esquema está relacionado con la duplicación de los componentes requeridos para la prestación del servicio, así como los esquemas anteriores que agregaban recursos adicionales este los duplica, previniendo garantizando que en caso de fallas los usuarios podrán tener el servicio con toda capacidad, adicionalmente cabe aclarar que este esquema permite que los recursos aparte de estar redundantes, estén distribuidos geográficamente, lo que aumenta la probabilidad de continuidad en los niveles de servicio. Generalmente se aplican en cuartos de datos artesanales o *datacenters* tipo TIER IV mientras exista la posibilidad de tener dos proveedores eléctricos independientes y redundantes para cada unidad.
- **Doble redundancia en paralelo (2(N+1)):** Este esquema de servicio es de los más confiables existentes, los cuales combina los puntos tres y cuatro (3 y 4), garantizando el esquema de servicio mínimo, más el componente adicional del +1, teniendo doble redundancia de componentes primarios y adicionalmente manejando de forma paralela un esquema de servicio activo-activo.
- **Redundante distribuida (2(N+1)):** Para finalizar, este esquema tiene en cuenta los esquemas presentados en los puntos dos y cuatro (2 y 4), con el fin de conservar la capacidad de servicio adicional del n+1, pero con los componentes replicados en una ubicación geográfica totalmente diferente.

Para finalizar este apartado sobre la confiabilidad y la tolerancia a fallos, se usa un procedimiento empleado que sirve para definir y medir la confiabilidad de un elemento de software o de una aplicación informática, esta métrica permite conocer el tiempo medio entre fallas o llamado en inglés (MTBF) que es el acrónimo para Mean Time Between Failures. El MTBF normalmente está expresado en horas, comienzan desde que se produce el error en un componente y es necesario corregir. Para calcular el MTBF, se usa la siguiente ecuación:

$$MTBF = \frac{Tt - \sum_1^f Tf}{f}$$

Ecuación 2. Fórmula para el cálculo del tiempo medio entre errores

Fuente: elaboración propia

Donde cada una de las variables son.

MTBF: Tiempo medio entre fallas (Men Time Between Failures)

Tt: Tiempo total del Servicio (generalmente 1 año – 31.536.000 segs)

f : Número total de fallos en el tiempo total del servicio

Tf: Tiempo individual en segundos de los fallos ocurridos

1.3. Desempeño

El desempeño está relacionado con una rama de la computación, la cual se explora más adelante en este Módulo. Sin embargo, para explicar los aspectos de diseño es necesario hacer un acercamiento a este tema y cómo afecta el desarrollo y el proceso de diseño de una aplicación distribuida.

La Computación de Alto Desempeño (CAD) o High Performance Computing (HPC) ha jugado un papel muy importante en el avance de la ciencia en los últimos años, ya que para procesar, manipular y almacenar datos a gran escala estos son fundamentales, no se podrían probar las teorías propuestas en las diferentes ramas del conocimiento en el mundo sin este procesamiento.

La computación de alto desempeño es el uso de procesamiento en paralelo para el uso avanzado más eficiente, fiable y rápido de aplicaciones. El término se usa especialmente para designar los sistemas cuya función es de 1 teraflop o 1012 operaciones de punto flotante por segundo (Yang and Guo 2006).

El término HPC es ocasionalmente usado como sinónimo para supercomputación, aunque técnicamente un supercomputador es un sistema que realiza o está cerca de la tasa de operación más alta para los computadores. Algunos supercomputadores trabajan a más de un petaflop o 1015 operaciones de punto flotante por segundo.

1.3.1. Modelos de programación distribuida

Una parte importante de la computación de alto desempeño son los modelos de programación utilizados en este tipo de escenarios. Estos permiten darles un mejor uso a estas nuevas tecnologías, incrementando rendimiento, reduciendo tiempos de espera en operaciones bloqueantes y mejorando la estructura del programa u algoritmo en sí.

La programación concurrente es uno de los modelos de programación más usados en los ambientes de alto desempeño, consiste en un grupo de subprogramas que pueden ser ejecutados en paralelo (Ben-Ari, 2006) (Losada, Rodríguez y Carrillo, 2009).

En otras palabras, programas concurrentes pueden ser ejecutados secuencialmente en un solo procesador o ejecutados en paralelo, asignando cada proceso computacional a un grupo de procesadores que pueden estar cerca o distribuidos a través de una red de computadores. En algunos sistemas computacionales concurrentes la comunicación entre los componentes está oculta del programador, mientras que en otros debe estar muy explícita.

La comunicación explicita puede ser dividida en 2 clases:

- **Comunicación de memoria compartida:** ocurre cuando los componentes concurrentes se comunican alterando el contenido de memorias compartidas. Este estilo de programación concurrente generalmente requiere de la aplicación de algunas maneras de asegurado (e.g. mutexs, semáforos o monitores) para coordinar entre hilos (Ben-Ari, 2006).
- **Comunicación de paso de mensajes:** es cuando los componentes concurrentes se comunican intercambiando mensajes que pueden ser cargados asíncronamente o que puede usarse un estilo *"rendezvous"* en el cual la entidad que envía bloquea hasta que el mensaje es recibido (Quinn y J, 2004). Asincrónicamente, el paso de mensajes puede ser de confianza o de desconfianza. Una amplia cantidad de teorías matemáticas para el entendimiento y el análisis de sistemas de paso de mensajes están disponibles (Ben-Ari, 2006).

La programación paralela es una forma de computación en la cual muchos cálculos son llevados o son ejecutados simultáneamente (Almasi & Gottlieb, 1989), operando el principio de que se pueden dividir en tareas más pequeñas, las cuales son resueltas concurrentemente en paralelo. La computación en paralelo básicamente usa unidades de procesamiento independientes para resolver un problema, con esto se rompe un problema de dependencia en el cual cada tarea puede resolverse de manera individual y simultánea con otras.

Una de las maneras para medir el paralelismo es por medio de variables como el tiempo de ejecución (Tp), sobrecarga paralelizada (To), *speed-up* (s), la eficiencia (E) y costo (PTp) de los algoritmos implementados (Bernstein, 1966) ((Grama, 2003). Estas variables se miden por medio de modelos matemáticos los cuales dan una aproximación del nivel de optimización en el cual un algoritmo ha sido implementado y generan como resultado curvas en los cuales se detectan los picos de máxima paralelización de una aplicación sobre una plataforma específica.

1.3.2. Ley de Moore

Lo anterior ha hecho que los requerimientos de operaciones de punto flotante por segundo (FLOPS) sean mayores, lo que hace que las tareas de computo no sean solo para científicos en supercomputadores, los cuales llegan a precios astronómicos y a un tamaño que ocupa habitaciones enteras, sino que mediante tecnologías accesibles gestiona un mejor uso de los recursos, y se generan nuevos paradigmas que permiten alcanzar niveles de procesamiento bastante elevados a costos mucho más bajos de lo que vale un super computador.

Año	Microoprocesador	Cantidad
1971	4004	2.250
1972	8008	2.500
1974	8080	5.000
1978	8086	29.000
1982	286	120.000
1985	intel 386	275.000
1989	Pentium	1.180.000
1993	Pentium II	3.100.000
1997	Pentium III	7.500.000
1999	Pentium IV	24.000.000
2000	Intel Itaium	42.000.000
2002	Intel Itaium	220.000.000
2003	Intel Itaium	410.000.000
2010	Core i7	1.000.000.000
2015	Core i7	10.000.000.000.000

Figura 2. Ley de Moore

Fuente: elaboración propia a partir de los datos de la revista universitaria de la UNAM

1.4. Escalabilidad

Este aspecto de diseño hace referencia a la capacidad de un sistema de crecer según la cantidad usuarios concurrentes requiriendo sus servicios. Este concepto está directamente ligado con la confiabilidad, específicamente con el sub ítem de tolerancia a fallos, donde inicialmente un sistema se despliega para una cantidad de usuarios concurrentes, este valor lo explicamos como n y representa una cantidad de usuarios concurrentes más un porcentaje adicional, generalmente definido por el arquitecto de la solución al momento de considerar el despliegue de la aplicación en la infraestructura destinada para alojar dicha aplicación.

Con el concepto de escalabilidad el sistema crece sin necesidad de:

- Afectar el servicio de los usuarios existentes.
- Incrementar la cantidad de usuarios del servicio.
- Garantizar las todas las funcionalidades del sistema.

Para lo anterior existen dos tipos de escalabilidad, ambas están relacionadas por igual con el incremento del hardware y el software, ya sea por necesidades de replicación de componentes, incremento de capacidades o balanceo de funcionalidades del sistema.

1.4.1. Escalabilidad vertical

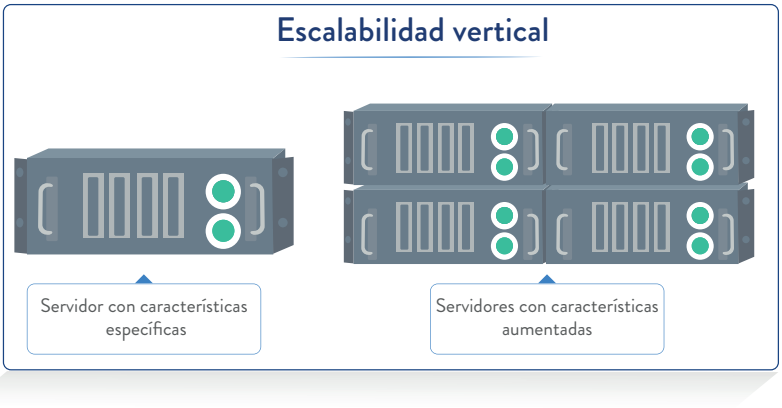

Figura 3. Escalabilidad vertical

Fuente: elaboración propia

Este esquema de crecimiento está orientado al crecimiento de los componentes del sistema; estos componentes pueden ser el crecimiento de un servidor en capacidad, con el fin de que con las mismas unidades físicas se presten servicios a más usuarios. Este tipo de crecimiento es el más básico y generalmente tiene un límite, dado por la capacidad del *hardware* al momento en el que se va a realizar el crecimiento.

Como se puede ver en el servidor a la izquierda, cuenta con unas características específicas al momento del despliegue de la aplicación. El servidor a la derecha es el mismo servidor con características aumentadas al momento de necesitar el escalado del sistema distribuido.

1.4.2. Escalabilidad horizontal

Este tipo de escalabilidad es la más recomendada para un sistema distribuido, porque a medida que más usuarios utilizan el sistema, los administradores pueden adicionar más servicios de forma lateral de modo de que el sistema siempre podrá aceptar el aumento. Sin embargo, este tipo de escalabilidad, si bien aprovecha el *hardware* tipo *commodity*, necesita que desde la misma fase de diseño de la aplicación sean contemplados los siguientes métodos de escalamiento horizontal.

Balanceo de funcionalidades: este esquema también conocido como balanceo de carga, es necesario cuando los componentes de la aplicación están fuertemente acoplados, esto quiere decir que los módulos que prestan el servicio hacia los usuarios necesitan que la información esté disponible en tiempo de ejecución en la misma instancia de la aplicación, es decir que el balanceo extiende la aplicación en más de un ambiente con el fin de que pueda "ocupar más espacio". Por ejemplo, este tipo de escalabilidad es la que se utiliza en contenedores de aplicaciones que se encuentran distribuidos en múltiples máquinas, pero juntos forman una instancia de la aplicación.

En este método de escalabilidad horizontal se tienen un grupo de máquinas de capacidades moderadas, las cuales se encuentran sincronizadas por un aplicativo, en este caso un *middleware*, con el fin de proveer un mismo ambiente con la capacidad conjunta de las máquinas y la posibilidad de adicionar más máquinas según la necesidad de servicio del sistema distribuido.

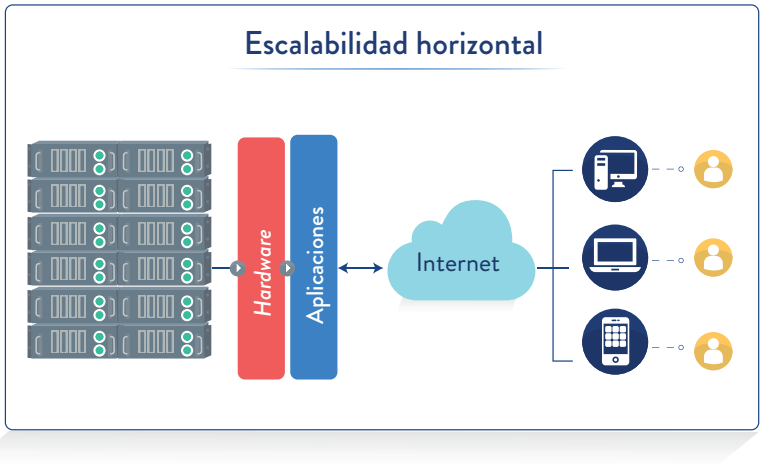

Figura 4. Escalabilidad horizontal

Fuente: elaboración propia

Replicación o distribución de componentes: este tipo de replicación es posible cuando el *software* definido posee características o módulos débilmente acoplados, esto quiere decir que los componentes pueden ejecutarse en ambientes de memoria separados y que se comunican por medio de mensajes o protocolos específicos. Este tipo de escalabilidad es una de las más sencillas al momento de realizar un despliegue de la aplicación a la infraestructura ya que es posible relegar trabajo a máquinas específicas. Por ejemplo, este tipo de extensión de capacidad es comúnmente usado en las arquitecturas orientadas a servicios.

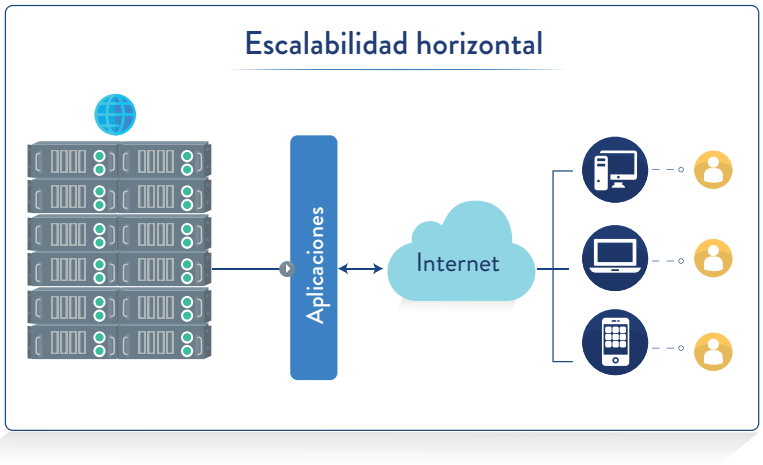

Figura 5. Escalabilidad horizontal – Replicación o distribución de componentes

Fuente: elaboración propia

En este método de escalabilidad horizontal se tienen un grupo de máquinas de capacidades moderadas. En cada máquina se encuentra un servicio diferente, de abajo hacia arriba se encuentran dos servidores de bases de datos, dos de aplicación (negocio), un servidor *web*.

Para finalizar es importante recalcar que no hay una única respuesta al momento de definir un tipo de escalabilidad en el diseño de una aplicación, generalmente se recomienda estudiar desventajas y ventajas de cada una de los anteriores temas según:

- Requerimientos de los usuarios
- Tiempo de desarrollo del proyecto
- Infraestructura disponible
- Esquema de servicio y operación
- Presupuesto

1.5. Transparencia

La transparencia hace referencia a la capacidad que tiene el sistema distribuido de verse como un todo a pesar de los procesos que estén ejecutándose en el mismo. Así mismo el usuario debe percibir que está interactuando con una sola entidad (sistema) y no con varias entidades más pequeñas que se complementan en funcionalidad entre ellas a pesar de que el sistema esté compuesto de varias partes, ya sean servidores, aplicativos, etc.

Actualmente, la Organización Internacional para la Estandarización (International Organization for Standardization - ISO), en el documento de código ISO/IEC 19793 del 2015, ha definido ocho (8) tipos de transparencia. Sin embargo, habiendo estudiado las características anteriores se puede decir que estos tipos de transparencia podrían ser sub catalogados en 2 tipos: los relacionados con otros aspectos de diseño y los que no están relacionados con otros aspectos de diseño.

Transparencia de concurrencia: también conocida como paralelismo, posibilita al sistema que el usuario utilice varios recursos en el mismo instante de tiempo, producto del procesamiento de una tarea más grande o más compleja.

Transparencia de replicación: autoriza el uso de recursos adicionales para completar la tarea en menor tiempo. Es un efecto de la anterior.

Transparencia de fallos: si ocurre un fallo en el sistema, existe algún mecanismo de control que permite ya sea salvar la tarea del usuario y moverla a otro recurso activo o corregir el fallo y continuar con la tarea en el mismo recurso activo nuevamente.

Transparencia de prestaciones: permite que el sistema cambie con el fin de poder aumentar la eficiencia en el uso de los recursos asignados al sistema o de recursos adicionales, está directamente relacionado con la velocidad del procesamiento de una tarea.

Transparencia de migración: habilita el movimiento de instancias, objetos, sesiones, etc. a otro recurso por la razón que sea, temas relacionados con el servicio o con fallos del mismo. Este también es un efecto de la transparencia de fallos y de prestaciones.

Transparencia de escalado: tolera el crecimiento o el decrecimiento del tamaño del sistema según las necesidades del mismo. Esto generalmente está asociado a la carga de los usuarios y a la cantidad concurrente de los mismos en el sistema. Este es uno de los principios del *Cloud*, llamado Elasticidad.

Transparencia de localización: permite que los procesos que el usuario ejecute puedan encontrar, información o recursos, sin importar en qué lugar del sistema se encuentren.

Transparencia de acceso: permite el uso de la información o recursos previamente descubiertos, con el fin de que los usuarios perciban una única entidad o un sistema como un todo y no como un conjunto de partes.

Referencias bibliográficas

Almasi, G. y Gottlieb, A. (1989). *Highly parallel computing*. Benjamin/Cummings. Recuperado de: https://dl.acm.org/citation.cfm?id=160438

Ben-Ari, M. (2006). *Principles of concurrent and distributed programming*. Addison-Wesley.

Bernstein, A. J. (1966). Analysis of Programs for Parallel Processing. *IEEE Transactions on Electronic Computers*, EC-15(5), 757–763. Recuperado de: https://doi.org/10.1109/PGEC.1966.264565

Godfrey, M. y Hendry, D. (1993). The computer as von Neumann planned it. *IEEE Annals of the History of Computing*, 15(1), 11–21. Recuperado de: https://doi.org/10.1109/85.194088

Grama, A. (2003). *Introduction to parallel computing.* Addison-Wesley.

Losada, Rodríguez y Carrillo. (2009). *Programación Avanzada, Concurrente y Distribuida* (2nd ed.). España: Universidad Politécnica de Madrid.

Mead, C. (s.f.). *Excerpts from A Conversation with Gordon Moore: Moore's Law Moore's Law*, 2. Recuperado de: http://large.stanford.edu/courses/2012/ph250/lee1/docs/Excepts_A_Conversation_with_Gordon_Moore.pdf

Pressman, R. S. (2005). *Software engineering : a practitioner's approach*. McGraw-Hill.

Quinn, M. J. (2004). *Parallel programming in C with MPI and openMP*. McGraw-Hill. Recuperado de: https://dl.acm.org/citation.cfm?id=1211440

Rincón, D. (2018). Conceptos Generales de los Sistemas Distribuidos. *Módulo de Sistemas Distribuidos* (p. 60).

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Sistema Distribuidos

Unidad 1: Conceptos Generales de los Sistemas Distribuidos – Diseño de un SD

Escenario 2: Arquitecturas de los sistemas distribuidos, diseño de un SD

Autor: Alexis Rojas

Asesor Pedagógico: Jeimy Lorena Romero Perilla

Diseñador Gráfico: Brandon Steven Ramírez Carrero

Asistente: Maria Elizabeth Avilán Forero