Sun Java™ System

# Sun Java Enterprise System 2005Q1 Guía de planificación de la implementación

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
EE.UU.

Nº de pieza: 819-1917

Capítulo 5

# Diseño de implementación

Durante la etapa de diseño de implementación del ciclo de vida de la solución, se diseña una arquitectura de implementación general y una especificación de implementación detallada y se prepara una serie de planes y especificaciones necesarias para implementar la solución. La aprobación del proyecto se produce en la fase de diseño de implementación.

Este capítulo incluye los siguientes apartados:

# Información acerca del diseño de implementación

El diseño de implementación se inicia con el escenario de implementación que se crea durante las etapas de diseño lógico y requisitos técnicos del ciclo de vida de la solución. El escenario de implementación contiene una arquitectura lógica y los requisitos de calidad del servicio (QoS) de la solución. Los componentes identificados en la arquitectura lógica se asignan en servidores físicos y otros dispositivos de red para crear una arquitectura de implementación. Los requisitos de QoS proporcionan directrices sobre las configuraciones de hardware para obtener especificaciones relativas a la calidad del servicio como el rendimiento, la disponibilidad o la escalabilidad.

El diseño de la arquitectura de implementación es un proceso iterativo. Normalmente, se revisan los requisitos de QoS y se vuelven a examinar los diseños preliminares. Se tienen en cuenta las relaciones entre los distintos requisitos de calidad de servicio, equilibrando los cambios y el coste de propiedad para obtener una solución óptima que en última instancia satisfaga los objetivos del proyecto.

## Aprobación del proyecto

La aprobación del proyecto se produce en la etapa de diseño del mismo, normalmente después de haber creado la arquitectura de implementación. Con la arquitectura de implementación y las especificaciones que se describen a continuación, se calcula el coste real de la implementación y se envía a los participantes para su aprobación. Una vez que se aprueba el proyecto, se firman los contratos para la realización de la implementación y se asignan los recursos necesarios para implementar el proyecto.

# Resultados del diseño de implementación

Durante la etapa de diseño de implementación, deberá preparar alguno de los siguientes planes y especificaciones:

- **Arquitectura de implementación.** Una arquitectura general que muestra la asignación de una arquitectura lógica en un entorno físico. El entorno físico incluye los nodos informáticos de un entorno de intranet o Internet, procesadores, memoria, dispositivos de almacenamiento y otros dispositivos de red y hardware.
- **Especificaciones de implementación.** Especificaciones detalladas que se utilizan como referencia para crear la implementación. Estas especificaciones proporcionan detalles específicos acerca del hardware de red e informático para obtener y describir el diseño de la red para la implementación. Las especificaciones de implementación también incluyen especificaciones para los servicios de directorios, incluidos los detalles sobre el árbol de información de directorios (DIT) y los grupos y funciones definidos para el acceso a los directorios.
- **Planes de implementación.** Un grupo de planes que cubre distintos aspectos de la implementación de una solución de software empresarial. Los planes de implementación incluyen:
  - **Plan de migración.** Describe la estrategias para migrar datos de la empresa y actualizar el software empresarial. Los datos migrados deben ser compatibles con los formatos y estándares de las aplicaciones empresariales recién instaladas. Todo el software empresarial debe encontrarse en los niveles de versión correctos para que se pueda relacionar entre sí.
  - **Plan de instalación.** Obtenido a partir de la arquitectura de implementación, especifica los nombres del servidor de hardware, los directorios de instalación, la secuencia de instalación, los tipos de instalación para cada nodo y la información de configuración necesaria para instalar y configurar una implementación distribuida.
  - **Plan de gestión de usuarios.** Incluye estrategias de migración para los datos en los directorios y bases de datos existentes, especificaciones de diseño de directorio que tienen en cuenta el diseño de duplicación especificado en la arquitectura de implementación y procedimientos para proporcionar nuevo contenido a los directorios.

- **Plan de prueba.** Describe los procedimientos para probar el software implementado, incluidos planes específicos para desarrollar implementaciones prototipo y piloto, pruebas de estrés que determinan la capacidad para tratar las cargas previstas y pruebas funcionales que determinan si la función planificada actúa de la forma prevista.
- **Plan de despliegue.** Describe los procedimientos y programación para cambiar la implementación de un entorno de planificación y prueba a uno de producción. Normalmente, el cambio de la implementación a la producción se produce en varias etapas. Por ejemplo, la primera etapa puede ser la implementación del software para un grupo limitado de usuarios y aumentar la base de usuarios con cada etapa hasta que se complete la implementación. La implementación en fases también puede incluir una implementación programada de paquetes de software específicos hasta que se complete toda la implementación.
- **Plan de recuperación tras desastre.** Describe los procedimientos que se deben llevar a cabo para restaurar el sistema cuando se producen fallos imprevistos en el mismo. El plan de recuperación incluye procedimientos para los fallos a pequeña y gran escala.
- **Plan de operaciones (Manual de ejecución).** Un manual de operaciones que describe los procedimientos de supervisión, mantenimiento, instalación y actualización.
- **Plan de formación.** Contiene procesos y procedimientos para formar a los operadores, administradores y usuarios finales acerca del software empresarial recién instalado.

# Factores que afectan al diseño de implementación

Varios factores influyen en la toma de decisiones durante el diseño de la implementación. Tenga en cuenta los siguientes factores clave:

- **Arquitectura lógica.** La arquitectura lógica detalla los servicios funcionales en una solución propuesta y las relaciones entre los componentes que proporcionan dichos servicios. Utilice la arquitectura lógica como el factor clave para determinar la mejor manera de distribuir los servicios. Un escenario de implementación contiene la arquitectura lógica junto con los requisitos de calidad de servicio (se describe a continuación).
- **Requisitos de calidad de servicio.** Los requisitos de calidad de servicio (QoS) especifican varios aspectos del funcionamiento de una solución. Utilice los requisitos de calidad de servicio para ayudar a desarrollar estrategias que permitan obtener los objetivos de rendimiento, disponibilidad, escalabilidad, facilidad de mantenimiento, etc. definidos.. Un escenario de implementación contiene la arquitectura lógica junto con los requisitos de calidad de servicio (como se ha descrito anteriormente).
- **Análisis de uso.** El análisis de uso, desarrollado durante la etapa de requisitos técnicos del ciclo de vida de la solución, proporciona información acerca de los patrones de uso que pueden ayudar a calcular la carga y el estrés de un sistema implementado. Utilice el análisis de uso para ayudar a aislar los atascos de rendimiento y desarrollar estrategias para satisfacer los requisitos de QoS.
- **Casos de uso.** Los casos de uso, desarrollados durante la etapa de requisitos técnicos del ciclo de vida de la solución, muestran varias interacciones de usuario identificadas para una implementación, a menudo identificando los casos de uso más comunes. Aunque los casos de uso están englobados en el análisis de uso, al evaluar un diseño de implementación se debe hacer referencia a los casos de uso para asegurar que se tratan adecuadamente.
- **Acuerdos de nivel de servicio.** Un acuerdo de nivel de servicio (SLA) especifica los requisitos de rendimiento mínimos y, cuando no se cumplen dichos niveles, el nivel y alcance del servicio técnico al cliente que se debe prestar. Un diseño de implementación debería satisfacer fácilmente los requisitos de rendimiento especificados en un acuerdo de nivel de servicio.

- **Coste total de propiedad.** En el diseño de implementación, se analizan las soluciones potenciales que se encargan de los requisitos de QoS para la disponibilidad, rendimiento, escalabilidad, etc. Sin embargo, para cada solución que se considera, también se debe tener en cuenta el coste de dicha solución y cómo repercute en el coste total de propiedad. Asegúrese de que tiene presente los cambios asociados con las decisiones y que optimiza los recursos para obtener los requisitos empresariales dentro de las limitaciones existentes.
- **Objetivos empresariales.** Los objetivos empresariales se establecen durante la etapa de análisis empresarial del ciclo de vida de la solución e incluyen los requisitos y limitaciones empresariales para satisfacer dichos objetivos. En última instancia, el diseño de implementación se juzga por su capacidad para cumplir los objetivos empresariales.

# Metodología del diseño de implementación

Al igual que otros aspectos de la planificación de la implementación, el diseño es tanto un arte como una ciencia y no se puede detallar con procedimientos y procesos específicos. Algunos de los factores que contribuyen a un diseño de implementación realizado con éxito son la experiencia previa en diseño, el conocimiento de la arquitectura de sistemas, el conocimiento del campo y el pensamiento creativo aplicado.

Normalmente, el diseño de implementación se centra en la obtención de los requisitos de rendimiento a la vez que se cumplen los requisitos de calidad de servicio. Las estrategias que se utilicen deben equilibrar los cambios implicados con las decisiones del diseño para optimizar la solución. Normalmente, la metodología que utilice implica las siguientes tareas:

- **Cálculo de los requisitos del procesador.** A menudo, el diseño de implementación comienza con el cálculo del número de CPU necesarias para cada componente de la arquitectura lógica. Comience con los casos de uso que representen la mayor carga y continúe con cada caso de uso. Tenga en cuenta la carga en todos los componentes que forman los casos de uso y modifique los cálculos consecuentemente. Considere también la experiencia previa que tenga con el diseño de sistemas empresariales.
- **Cálculo de los requisitos de procesadores para un transporte seguro.** Estudie los casos de uso que requieren un transporte seguro y modifique los cálculos de CPU de manera correspondiente.

- **Duplicación de los servicios para disponibilidad y escalabilidad.** Una vez que haya realizado los cálculos de procesadores, cambie el diseño para tener en cuenta los requisitos de calidad de servicio en cuanto a disponibilidad y escalabilidad. Considere soluciones de equilibrado de carga que traten las cuestiones de disponibilidad y recuperación tras error.

  En el análisis, considere los cambios derivados de las decisiones del diseño. Por ejemplo, ¿cuáles son los efectos de la estrategia de disponibilidad y escalabilidad sobre la facilidad de mantenimiento del sistema? ¿Cuáles son los otros costes de las estrategias?

- **Identificación de cuellos de botella.** A medida que continúa con su análisis, examine el diseño de implementación para identificar cualquier cuello de botella que produzca que la transmisión de los datos esté por debajo de los requisitos y realice los ajustes pertinentes.

- **Optimización de los recursos.** Revise el diseño de implementación en cuanto a gestión de los recursos y considere las opciones que minimizan los costes a la vez que satisfacen los requisitos.

- **Gestión de los riesgos.** Vuelva a estudiar los análisis técnicos y empresariales con respecto al diseño, realizando las modificaciones necesarias para tener en cuenta los eventos o situaciones que no se hubieran previsto en la planificación anterior.

# Cálculo de los requisitos de procesadores

Esta sección plantea un proceso para estimar el número de CPU y la memoria correspondiente que son necesarios para poder prestar los servicios en un diseño de implementación. La sección incluye un ensayo de un proceso de cálculo para un ejemplo de escenario de implementación de comunicaciones.

La estimación de la potencia de cálculo de la CPU es un proceso iterativo que considera las siguientes opciones:

- Componentes lógicos y sus interacciones (como se indica en las relaciones de dependencia entre los componentes en la arquitectura lógica)
- Análisis de uso para los casos de uso identificados
- Requisitos de calidad de servicio
- Experiencia anterior con diseño de implementación y con Java Enterprise System
- Consulta con los servicios profesionales de Sun que tienen experiencia en el diseño e implementación de distintos tipos de escenarios de implementación

El proceso de cálculo incluye los siguientes pasos. El orden de estos pasos no es vital, pero proporciona una forma de considerar los factores que afectan al resultado final.

1. Determine un cálculo base de CPU para los componentes identificados como puntos de entrada del usuario al sistema.

   Una decisión de diseño es si se deben cargar total o parcialmente las CPU. Las CPU completamente cargadas maximizan la capacidad del sistema. Para aumentar la capacidad, se aumentarán los costes de mantenimiento y el posible tiempo de inactividad al añadir CPU adicionales. En algunos casos, puede elegir añadir equipos adicionales para satisfacer el aumento de los requisitos de rendimiento.

   Las CPU cargadas parcialmente permiten tratar el aumento de requisitos de rendimiento sin tener que incurrir inmediatamente en costes de mantenimiento. Sin embargo, hay un gasto adicional asociado a los sistemas infrautilizados.

2. Ajuste los cálculos de CPU para tener en cuenta las interacciones entre los componentes.

   Estudie las interacciones entre los componentes en la arquitectura lógica para determinar la carga extra necesaria debido a los componentes dependientes.

3. Estudie el análisis de uso de los casos de uso específicos para determinar las cargas máximas del sistema, y realice los ajustes necesarios en los componentes que tratan estas cargas máximas.

   Comience con los casos de uso que más ponderan (aquellos que requieren la mayor carga) y continúe con cada caso de uso para asegurarse de que tiene en cuenta todos los escenarios de uso proyectados.

4. Ajuste los cálculos de CPU para que reflejen los requisitos de seguridad, disponibilidad y escalabilidad.

Este proceso de cálculo proporciona puntos de inicio para determinar la potencia de procesamiento real que necesita. Normalmente, se crean implementaciones prototipo basadas en dichos cálculos y luego se realizan comprobaciones estrictas con respecto a los casos de uso previstos. Sólo después de comprobaciones iterativas puede determinar los requisitos de procesamiento reales para un diseño de implementación.

# Ejemplo de cálculo de los requisitos de procesadores

Esta sección ilustra una metodología para estimar la potencia de procesamiento necesaria para una implementación de ejemplo. La implementación de ejemplo está basada en la arquitectura lógica para la solución de comunicaciones basada en identidades para una empresa de tamaño medio de entre 1.000 y 5.000 empleados, como se describe en la sección "Ejemplo de comunicaciones basadas en identidades" en la página 65.

Las cifras de CPU y memoria que se utilizan en el ejemplo son cálculos arbitrarios utilizados sólo a modo de referencia. Estas cifras están basadas en datos arbitrarios en los que se basa el ejemplo teórico. Es necesario un análisis exhaustivo de los distintos factores para calcular los requisitos de procesadores. Este análisis debería incluir, aunque no debería limitarse a ello, la siguiente información:

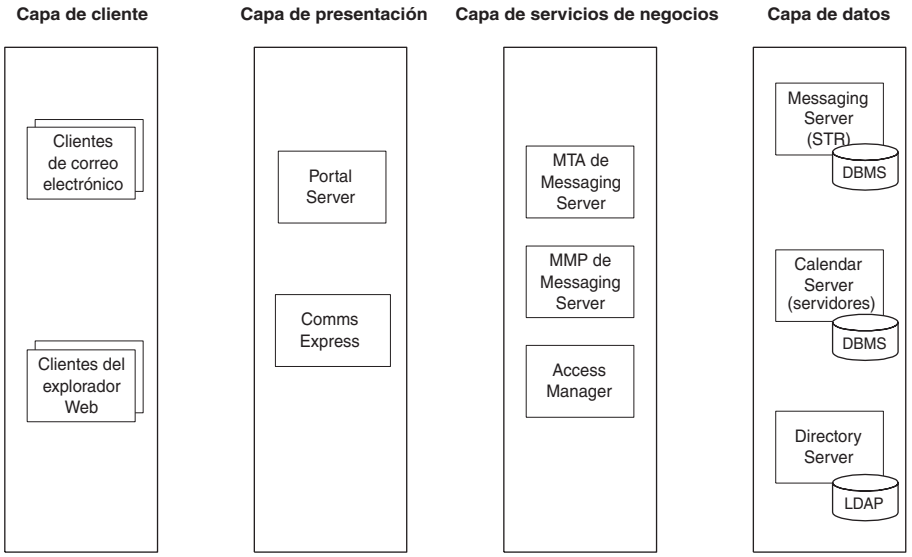
- Casos de uso detallados y análisis de uso basados en un análisis exhaustivo de negocio
- Requisitos de calidad de servicio determinados por el análisis de los requisitos empresariales
- Costes y especificaciones específicos del hardware de procesamiento y de red
- Experiencia previa en implementaciones de despliegues parecidos

**PRECAUCIÓN** La información que se presenta en estos ejemplos no representa ningún consejo de implementación y sólo se deberá considerar como un ejemplo de un proceso que puede utilizarse al diseñar un sistema.

## Determine el cálculo base de CPU para los puntos de entrada del usuario

Comience calculando el número de CPU necesarias para tratar la carga prevista de cada componente que actúa como punto de entrada de usuario. La siguiente figura muestra la arquitectura lógica para un escenario de comunicación basado en identidades descrito anteriormente en el Capítulo 4, "Diseño lógico" en la página 65.

**Figura 5-1** Arquitectura lógica para un escenario de comunicaciones basado en identidades




La siguiente tabla muestra los componentes en la capa de presentación de la arquitectura lógica que trabaja directamente con los usuarios finales de la implementación. La tabla incluye cálculos de CPU base obtenidos a partir del análisis de requisitos técnicos, casos de uso, análisis de uso específico y experiencia previa con este tipo de implementación.

**Tabla 5-1** Cálculos de CPU para componentes que contienen acceso a los puntos de entrada de usuario

| Componente | Número de CPU | Descripción |
|---|---|---|
| Portal Server | 4 | Componente que es un punto de entrada de usuario. |
| Communications Express | 2 | Dirige los datos a los canales de calendario y mensajería de Portal Server. |

## Incluya los cálculos de CPU para las relaciones de dependencia de los servicios

Los componentes que proporcionan puntos de entrada de usuario requieren compatibilidad con otros componentes de Java Enterprise System. Para continuar especificando requisitos de rendimiento, incluya los cálculos de rendimiento para que tengan en cuenta la compatibilidad requerida por otros componentes. El tipo de interacciones entre los componentes debería detallarse al diseñar la arquitectura lógica, tal y como se describe en los ejemplos de arquitectura lógica de la sección "Ejemplo de arquitecturas lógicas" en la página 60.

**Tabla 5-2** Cálculos de CPU para compatibilidad de componentes

| Componente | CPU | Descripción |
|---|---|---|
| Messaging Server MTA (entrada) | 1 | Dirige los mensajes de correo entrantes de clientes de Communications Express y de correo electrónico. |
| Messaging Server MTA (salida) | 1 | Dirige los mensajes de correo salientes a los destinatarios. |
| Messaging Server MMP | 1 | Acceso al almacén de mensajes de Messaging Server para clientes de correo electrónico. |
| Messaging Server STR (almacén de mensajes) | 1 | Recupera y almacena mensajes de correo electrónico. |
| Access Manager | 2 | Proporciona servicios de autorización y de autenticación. |
| Calendar Server (servidor) | 2 | Recupera y guarda datos de calendario para Communications Express, aplicación de usuario de Calendar Server. |
| Directory Server | 2 | Proporciona servicios de directorio LDAP. |
| Web Server | 0 | Proporciona compatibilidad con contenedor web para Portal Server y Access Manager.<br>(No se precisa de ciclos de CPU adicionales.) |

## Estudie los casos de uso para la utilización de carga máxima

Vuelva a los casos de uso y análisis de uso para identificar las áreas de utilización de carga máxima y realice los ajustes oportunos a los cálculos de CPU.

Por ejemplo, suponga que para este ejemplo identifica las siguientes condiciones de carga máxima:

- Aumento inicial de usuarios a medida que se registran simultáneamente
- Intercambios de correo electrónico durante franjas horarias específicas

Para tener en cuenta esta utilización de carga máxima, realice ajustes en los componentes que prestan estos servicios. La siguiente tabla describe los ajustes que puede tener en cuenta a la hora de evaluar esta utilización de carga máxima.

**Tabla 5-3** Ajustes en el cálculo de CPU para carga máxima

| Componente | CPU (ajustadas) | Descripción |
|---|---|---|
| Messaging Server MTA entrada | 2 | Añada 1 CPU por correo electrónico entrante pico |
| Messaging Server MTA salida | 2 | Añada 1 CPU por correo electrónico saliente pico |
| Messaging Server MMP | 2 | Añada 1 CPU por carga adicional |
| Messaging Server STR (almacén de mensajes) | 2 | Añada 1 CPU por carga adicional |
| Directory Server | 3 | Añada 1 CPU por búsquedas LDAP adicionales |

## Modifique los cálculos para otros estados de carga

Continúe con los cálculos de CPU para tener en cuenta otros requisitos de calidad de servicio que puedan afectar a la carga:

- **Seguridad.** A partir de la etapa de requisitos técnicos, determine cómo puede afectar el transporte seguro de los datos a los requisitos de carga y realice las modificaciones correspondientes en sus cálculos. La siguiente sección, "Cálculo de los requisitos de procesadores para transacciones seguras" en la página 84, describe un proceso para realizar los ajustes.
- **Duplicación de los servicios.** Ajuste los cálculos de CPU para tener en cuenta la duplicación de los servicios para incluir las consideraciones de disponibilidad, equilibrado de carga y escalabilidad. La siguiente sección, "Determinación de las estrategias de disponibilidad" en la página 88, debate el tamaño de las soluciones de disponibilidad. La sección "Determinación de estrategias para la escalabilidad" en la página 97 plantea las soluciones que implican contar con acceso a los servicios de directorios.
- **Capacidad latente y escalabilidad.** Modifique los cálculos de CPU según sea necesario para permitir una capacidad latente para cargas grandes imprevistas en la implementación. Compruebe los hitos previstos para el escalamiento y el aumento de carga previsto en el tiempo para asegurarse de que puede alcanzar cualquier hito previsto para escalar el sistema, ya sea horizontal o verticalmente.

## Actualice los cálculos de CPU

Normalmente, se redondea el número de CPU a un número par. Esta operación permite dividir de forma igual los cálculos de CPU entre dos servidores físicos y también añade un pequeño factor para la capacidad latente. Sin embargo, el redondeo de acuerdo con las necesidades específicas necesita una duplicación de servicios.

Como norma general, permita 2 gigabytes de memoria para cada CPU. La memoria real necesaria depende de la utilización específica y se puede determinar en la comprobación.

La siguiente tabla muestra los cálculos finales para el ejemplo de comunicaciones basado en identidades. Estos cálculos no incluyen la potencia de cálculo adicional que se podría añadir para la seguridad y disponibilidad. Los totales para seguridad y disponibilidad se agregarán en las siguientes secciones.

**Tabla 5-4** Ajustes de cálculos de CPU para compatibilidad de componentes

| Componente | CPU | Memoria |
|---|---|---|
| Portal Server | 4 | 8 GB |
| Communications Express | 2 | 4 GB |
| Messaging Server (MTA; entrada) | 2 | 4 GB |
| Messaging Server (MTA; salida) | 2 | 4 GB |
| Messaging Server (MMP) | 2 | 4 GB |
| Messaging Server (almacén de mensajes) | 2 | 4 GB |
| Access Manager | 2 | 4 GB |
| Calendar Server | 2 | 4 GB |
| Directory Server | 4 | 8 GB (redondeado desde 3 CPU/6 GB de memoria) |
| Web Server | 0 | 0 |

# Cálculo de los requisitos de procesadores para transacciones seguras

El transporte seguro de los datos implica el tratamiento de transacciones en un protocolo de transporte seguro como nivel de sockets seguro (SSL) o seguridad de nivel de transporte (TLS). Normalmente, las transacciones tratadas en un transporte seguro requieren una potencia de cálculo adicional para en primer lugar establecer una sesión segura (conocida como protocolo de enlace) y luego para cifrar y descifrar los datos transportados. En función del algoritmo de cifrado utilizado (por ejemplo, algoritmos de cifrado de 40 bits o 128 bits), la potencia de cálculo adicional puede ser importante.

Para que las transacciones seguras se realicen al mismo nivel que las transacciones no seguras, debe planificar la potencia de cálculo adicional. En función de la naturaleza de la transacción y los servicios de Sun Java™ Enterprise System que las tratan, las transacciones seguras pueden requerir hasta cuatro veces más de potencia de cálculo que las transacciones no seguras.

Al calcular la potencia de cálculo necesaria para tratar las transacciones seguras, analice los casos de uso para determinar el porcentaje de transacciones que requieren un transporte seguro. Si los requisitos de rendimiento para las transacciones seguras son iguales que para las transacciones no seguras, modifique los cálculos de CPU para tener en cuenta la potencia de cálculo adicional necesaria para las transacciones seguras.

En algunos escenarios de uso, el transporte seguro sólo puede ser necesario para la autenticación. Una vez que un usuario se autentica en el sistema, no son necesarias más medidas de seguridad para transportar los datos. En otros escenarios, el transporte seguro puede ser necesario para todas las transacciones.

Por ejemplo, al examinar un catálogo de productos en un sitio de comercio electrónico, todas las transacciones pueden ser no seguras hasta que el cliente realice todas las selecciones y esté listo para realizar la compra. Sin embargo, algunos escenarios de uso, como las implementaciones para los bancos o agentes de bolsa, requieren que la mayoría o todas las transacciones sean seguras y aplican el mismo estándar de rendimiento para las transacciones seguras y no seguras.

## Cálculos de CPU para transacciones seguras

Esta sección continúa la implementación de ejemplo para mostrar cómo calcular los requisitos de CPU para un caso de uso teórico que incluye transacciones seguras y no seguras.

Para calcular los requisitos de CPU para las transacciones seguras, lleve a cabo los siguientes cálculos:

1. Comience con una cifra de base para los cálculos de CPU (como se mostraba en la sección anterior, "Ejemplo de cálculo de los requisitos de procesadores" en la página 79).
2. Calcule el porcentaje de transacciones que requieren un transporte seguro y calcule las CPU previstas para las transacciones seguras.
3. Calcule el número de CPU reducido para las transacciones no seguras.
4. Anote el número para las transacciones seguras y para las no seguras para calcular el número total de CPU.
5. Redondee el número de CPU total a un número par.

La Tabla 5-5 muestra un cálculo de ejemplo basado en casos de uso y análisis de uso para Portal Server que suponen lo siguiente:

- Todos los inicios de sesión requieren autenticación segura.
- Todos los inicios de sesión suponen un 10% de la carga total de Portal Server.
- El requisito de rendimiento para las transacciones seguras es el mismo que el requisito de rendimiento para transacciones no seguras.

Para contar con la potencia de cálculo extra necesaria para tratar las transacciones seguras, el número de CPU necesarias para tratar estas transacciones se aumentará por cuatro. Al igual que otras cifras de CPU del ejemplo, este factor es arbitrario y simplemente tiene una finalidad ilustrativa.

**Tabla 5-5** modificación de los cálculos de CPU para transacciones seguras

| Paso | Descripción | Cálculo | Resultado |
|---|---|---|---|
| 1 | Comience con un cálculo de base para todas las transacciones de Portal Server. | El cálculo de base a partir de la Tabla 5-3 en la página 82 es 4 CPU. | - - - - - |
| 2 | Calcule el número de CPU adicional para las transacciones seguras. Suponga que las transacciones seguras requieren cinco veces más de potencia de CPU que las transacciones no seguras. | El diez por ciento del cálculo de base requiere un transporte seguro:<br>`0,10 x 4 CPU = 0,4 CPU`<br>Aumente la potencia de CPU para transacciones seguras en un factor de cuatro:<br>`4 x 0,4 = 1,6 CPU` | 1,6 CPU |
| 3 | Calcule el número de CPU reducido para las transacciones no seguras. | El noventa por ciento del cálculo de base son transacciones no seguras:<br>`0,9 x 4 CPU = 3,6 CPU` | 3,6 CPU |
| 4 | Calcule el número total de CPU ajustado para las transacciones seguras y no seguras. | Cálculo para transacciones seguras + no seguras = total:<br>`1,6 CPU + 3,6 CPU = 5,2 CPU` | 5,2 CPU |
| 5 | Redondee a un número par. | `5,2 CPU ==> 6 CPU` | 6 CPU |

A partir de los cálculos para transacciones seguras de este ejemplo, debería modificar el número total de CPU en la Tabla 5-5 en la página 86 añadiendo dos CPU y cuatro gigabytes de memoria para obtener el siguiente total para Portal Server.

**Tabla 5-6** Ajustes de cálculos de CPU para transacciones seguras en Portal Server

| Componente | CPU | Memoria |
|---|---|---|
| Portal Server | 6 | 12 GB |

# Hardware especializado para tratar las transacciones SSL

Los dispositivos de hardware especializados, como las tarjetas aceleradoras SSL y otros dispositivos, están disponibles para ofrecer potencia de cálculo y establecer sesiones seguras y el cifrado y descifrado de los datos. Al utilizar hardware especializado para las operaciones SSL, se dedica una potencia de cálculo para parte de los cálculos SSL, normalmente la operación de protocolo de enlace que establece una sesión segura.

Este hardware puede ser beneficioso para la arquitectura de implementación final. Sin embargo, debido a la naturaleza especializada del hardware, calcule en primer lugar los requisitos de rendimiento para transacciones seguras en términos de potencia de CPU y luego considere las ventajas de utilizar hardware especializado para tratar la carga adicional.

Algunos factores que se deben considerar al utilizar hardware especializado es si los casos de uso admiten la utilización de hardware (por ejemplo, casos de uso que requieren un gran número de operaciones de protocolo de enlace SSL) y la capa añadida de complejidad que este hardware aporta al diseño. Esta complejidad incluye la instalación, configuración, pruebas y administración de estos dispositivos.

# Determinación de las estrategias de disponibilidad

Al desarrollar una estrategia para requisitos de disponibilidad, estudie las interacciones entre componentes y el análisis de uso para determinar las soluciones de disponibilidad que debe considerar. Realice el análisis componente por componente, determinando la solución que mejor se ajuste a los requisitos de disponibilidad y recuperación tras error.

Los siguientes elementos son ejemplos del tipo de información que recopila para determinar las estrategias de disponibilidad:

- ¿Cuál es el porcentaje de disponibilidad especificado?
- ¿Cuáles son las especificaciones de rendimiento con respecto a las situaciones de recuperación tras error (por ejemplo, al menos un 50% de rendimiento durante la recuperación)?
- ¿El análisis de uso identifica los tiempos de uso máximo?
- ¿Cuáles son las consideraciones geográficas?

La estrategia de disponibilidad que elija también debe tener en cuenta los requisitos de facilidad de mantenimiento, como se plantea en "Determinación de estrategias para la escalabilidad" en la página 97. Evite las soluciones complejas que requieren un gran esfuerzo de mantenimiento y administración.

## Estrategias de disponibilidad

Las estrategias de disponibilidad para las implementaciones de Java Enterprise System incluyen:

- **Equilibrado de carga.** Utiliza los componentes de hardware y software redundantes para compartir una carga de procesamiento. Un equilibrador de carga dirige todas las solicitudes de un servicio a uno de las varias instancias simétricas del servicio. Si cualquiera de estas instancias fallase, hay otras instancias disponibles para asumir una mayor carga.
- **Recuperación tras error.** Implica la gestión del hardware y software redundante para proporcionar un acceso continuo a los servicios y seguridad para los datos críticos en caso de que alguno de los componentes falle.

El software Sun Cluster proporciona una solución de recuperación tras error para los datos críticos gestionados por los componentes de servidores como el almacenamiento de mensajes de Messaging Server y los datos de calendario de Calendar Server.

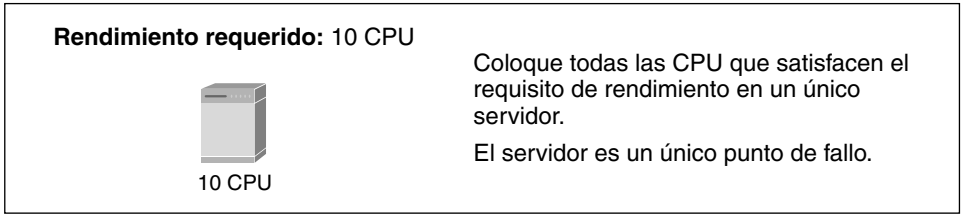
- **Duplicación de servicios.** La duplicación de servicios proporciona varias fuentes para acceder a los mismos datos. Directory Server proporciona varias estrategias de duplicación y sincronización para el acceso al directorio LDAP.

Las siguientes secciones proporcionan algunos ejemplos de soluciones de disponibilidad que proporcionan varios niveles de equilibrado de carga, recuperación tras error y duplicación de servicios.

## Sistema de un servidor

Coloque todos los recursos de cálculo de un servicio en un único servidor. Si el servidor falla, todo el servicio falla.

**Figura 5-2** Sistema de un servidor




Sun ofrece servidores de gama alta que proporcionan las siguientes ventajas:

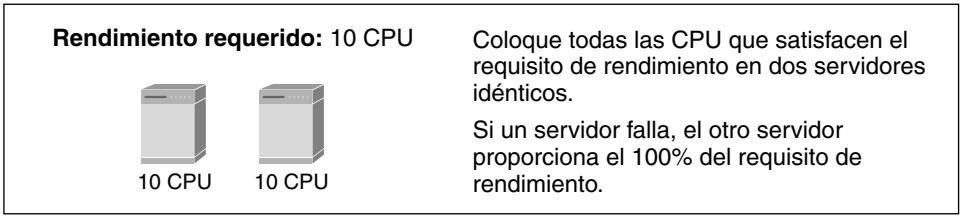
- Sustitución y reconfiguración de componentes de hardware mientras el sistema se está ejecutando
- Capacidad para ejecutar varias aplicaciones en dominios a prueba de fallos en un servidor
- Posibilidad de actualizar la capacidad, la velocidad de rendimiento y la configuración de E/S sin reiniciar el sistema

Normalmente, un servidor de gama alta cuesta más que un sistema multiservidor comparable. Sin embargo, un único servidor proporciona ahorros en cuanto a los costes de administración, supervisión y hospedaje con respecto a los servidores en un centro de datos. Las funciones de equilibrador de carga, recuperación tras error y eliminación de puntos de fallo son más flexibles en los sistemas multiservidor.

## Sistemas redundantes horizontalmente

Existen varias maneras de aumentar la disponibilidad con servidores redundantes paralelos que proporcionan equilibrado de carga y recuperación tras errores. La siguiente figura muestra dos servidores duplicados que proporcionan un sistema de recuperación tras error N+1. Un sistema N+1 cuenta con un servidor adicional para proporcionar el 100% de capacidad en caso de que un servidor falle.

**Figura 5-3** Sistema de recuperación tras error N+1 con dos servidores

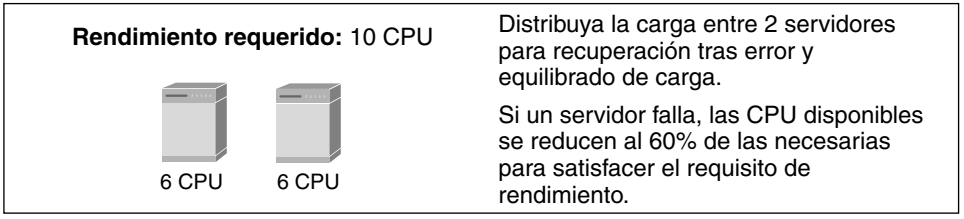



La potencia de cálculo de cada servidor en la Figura 5-3 anterior es idéntica. Sólo un servidor trata los requisitos de rendimiento. El otro servidor proporciona el 100% del rendimiento cuando llama a un servicio como copia de seguridad.

La ventaja de un diseño de recuperación tras error N+1 es un 100% de rendimiento durante una situación de recuperación tras error. Las desventajas incluyen un aumento en los costes de hardware sin una ganancia correspondiente en el rendimiento general (debido a que un servidor se encuentra en modo en espera para utilizarse únicamente en las situaciones de recuperación tras error).

La siguiente figura ilustra un sistema que implementa el equilibrado de carga más una recuperación de errores que distribuye el rendimiento entre dos servidores.

**Figura 5-4** Equilibrado de carga más recuperación tras error en dos servidores

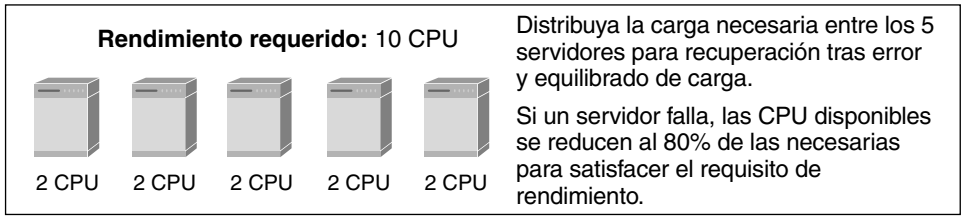


En el sistema que se muestra en la Figura 5-4 anterior, si un servidor falla, todos los servicios están disponibles, aunque a un porcentaje de la capacidad completa. El servidor restante proporciona 6 CPU de potencia de cálculo, que es el 60% del requisito de 10 CPU.

Una ventaja de este diseño es la capacidad latente de 2 CPU cuando ambos servidores están disponibles.

La siguiente figura ilustra una distribución entre un número de servidores para el rendimiento y equilibrado de carga.

**Figura 5-5** Distribución de la carga entre *n* servidores




Debido a que hay cinco servidores en el diseño que se muestra en la Figura 5-5, si un servidor falla, el resto de los servidores proporcionan un total de 8 CPU de potencia de cálculo, que es el 80% del requisito de rendimiento de 10 CPU. Si añade un servicio adicional con una capacidad de 2 CPU al diseño, tendrá efectivamente un diseño N+1. Si un servidor falla, se satisface el 100% del rendimiento por los servidores restantes.

Este diseño incluye las siguientes ventajas:

- Rendimiento adicional en caso de que un servidor falle
- Disponibilidad incluso cuando falla más de un servidor
- Los servidores se pueden rotar para realizar el mantenimiento y llevar a cabo actualizaciones
- Normalmente, varios servidores de gama baja cuestan menos que un único servidor de gama alta

Sin embargo, los costes de administración y mantenimiento pueden aumentar significativamente con servidores adicionales. También se debe considerar el coste del hospedaje de los servidores en un centro de datos. En un momento dado, se obtiene una menor rentabilidad al agregar servidores adicionales.

### Software de Sun Cluster

Para las situaciones que requieren un mayor grado de disponibilidad (como cuatro o cinco nueves), deberá considerar el software de Sun Cluster como parte de su diseño de disponibilidad. Un sistema de clúster es el acoplamiento de servidores redundantes con almacenamiento y otros recursos de red. Los servidores de un clúster se comunican continuamente entre sí. Si uno de los servidores pierde la conexión, el resto de los dispositivos del clúster aíslan el servidor y recuperan cualquier aplicación o datos que haya fallado desde el nodo que haya fallado u otro nodo. Este proceso de recuperación tras error se obtiene de forma relativamente rápida con una pequeña interrupción del servicio a los usuarios del sistema.

El software de Sun Cluster requiere un hardware adicional dedicado y conocimientos especializados para realizar la configuración, administración y mantenimiento.

## Ejemplo de diseño de disponibilidad

Esta sección contiene dos ejemplos de estrategias de disponibilidad basadas en la solución de comunicaciones basadas en identidades para una empresa de tamaño medio con un número de empleados entre 1.000 y 5.000, como se describió anteriormente en "Ejemplo de comunicaciones basadas en identidades" en la página 65. La primera estrategia de disponibilidad ilustra el equilibrio de carga para Messaging Server. La segunda ilustra una solución de recuperación tras error que utiliza software de Sun Cluster.

### Ejemplo de equilibrado de carga para Messaging Server

La siguiente muestra los cálculos de potencia de CPU para cada componente lógico de Messaging Server en la arquitectura lógica. Esta tabla repite el cálculo final determinado en la sección "Actualice los cálculos de CPU" en la página 83.

**Tabla 5-7** Ajustes de cálculos de CPU para compatibilidad de componentes

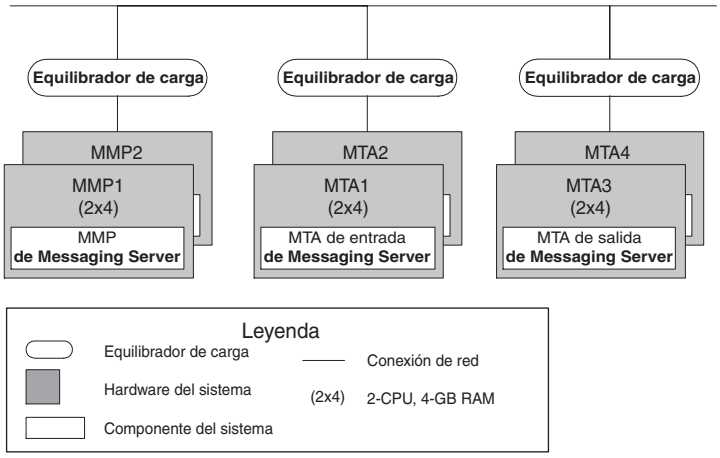
| Componente | CPU | Memoria |
|---|---|---|
| Messaging Server (MTA; entrada) | 2 | 4 GB |
| Messaging Server (MTA; salida) | 2 | 4 GB |
| Messaging Server (MMP) | 2 | 4 GB |
| Messaging Server (almacén de mensajes) | 2 | 4 GB |

Para este ejemplo, asuma que durante la etapa de requisitos técnicos, se especificaron los siguientes requisitos de calidad de servicio:

- **Disponibilidad.** La disponibilidad general del sistema debe ser 99,99% (no incluye el tiempo de inactividad programado). El fallo de un sistema informático individual no debe traducirse en un fallo del servicio.
- **Escalabilidad.** Ningún servidor debe tener más de un 80% de capacidad utilizada en la carga máxima diaria y el sistema debe ser capaz de aumentar un 10% anualmente.

Para satisfacer el requisito de disponibilidad, para cada componente de Messaging Server se deben proporcionar dos instancias, una de cada servidor de hardware separado. Si un servidor de un componente falla, el otro presta el servicio. La siguiente figura muestra el diagrama de red para esta estrategia de disponibilidad.

**Figura 5-6** Ejemplo de estrategia de disponibilidad para Messaging Server



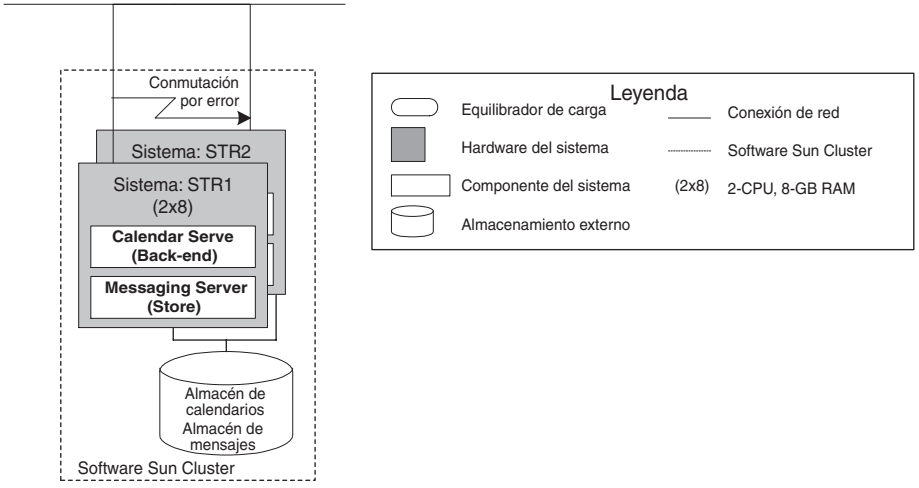

En la figura anterior, el número de CPU se ha duplicado desde su cálculo original. Las CPU se doblan por los siguientes motivos:

- En el caso de que se produzca un fallo en el servidor, el servidor restante proporciona la potencia de CPU necesaria para encargarse de la carga.
- Para el requisito de escalabilidad de que ningún servidor utilice más del 80% de su capacidad en una carga pico, la potencia de CPU ofrece este margen de seguridad.
- En cuanto al requisito de escalabilidad de un aumento anual del 10% de la carga, la potencia de CPU adicional añade la capacidad latente que puede tratar el aumento de cargas hasta que se necesite un escalamiento adicional.

## Ejemplo de recuperación tras error utilizando el software de Sun Cluster

La siguiente figura muestra un ejemplo de estrategia de recuperación tras error para Calendar Server de servidor y el almacén de mensajes de Messaging Server. El almacén de mensajes y entorno de servidor de Calendar Server se duplican en servidores de hardware independientes y se configuran para la recuperación tras error con el software de Sun Cluster. El número de CPU y la memoria correspondiente se duplica en cada servidor de Sun Cluster.

**Figura 5-7** Diseño de recuperación tras error utilizando el software de Sun Cluster




## Ejemplo de duplicación de servicios de directorio

Los servicios de Directory Server se pueden duplicar para distribuir las transacciones en varios servidores, proporcionando una alta disponibilidad. Directory Server proporciona varias estrategias para la duplicación de los servicios, como las siguientes:

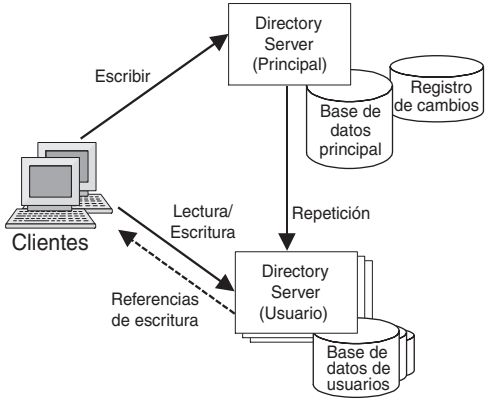
- **Varias bases de datos.** Almacena distintas partes del árbol de directorios en bases de datos separadas.
- **Encadenamiento y referencias.** Vincula los datos distribuidos en un único árbol de directorios.
- **Repetición de una réplica principal.** Proporciona un origen centralizado para la base de datos principal que luego se distribuye en réplicas de usuario
- **Repetición de varias réplicas principales.** Distribuye la base de datos principal entre varios servidores. Cada una de estas réplicas principales distribuye su base de datos entre sus réplicas de usuarios.

Las estrategias de disponibilidad para Directory Server es un tema complejo que no se engloba dentro del ámbito de esta guía. La siguientes secciones, "Repetición de una réplica principal" y "Repetición de varias réplicas principales" proporcionan una descripción general de las estrategias básicas de duplicación. Para obtener información detallada acerca de las estrategias de disponibilidad para Directory Server, consulte la *Directory Server Guía de planificación de la implementación*, `http://docs.sun.com/doc/817-7607`.

## Repetición de una réplica principal

La siguiente figura muestra una estrategia de duplicación de una réplica principal que ilustra los conceptos de replicación básicos.

**Figura 5-8** Ejemplo de repetición de una réplica principal




En la duplicación de una réplica maestra, una instancia de Directory Server gestiona la base de datos de directorios principal, registrando todos los cambios. La base de datos principal se duplica en cualquier número de bases de datos de usuario. Las instancias del usuario de Directory Server se optimizan para las operaciones de lectura y búsqueda. Cualquier operación de escritura recibida por un usuario se devuelve a la base de datos principal. La base de datos principal actualiza periódicamente las bases de datos de los usuarios.

Entre las ventajas de la repetición de una única réplica principal se incluyen:

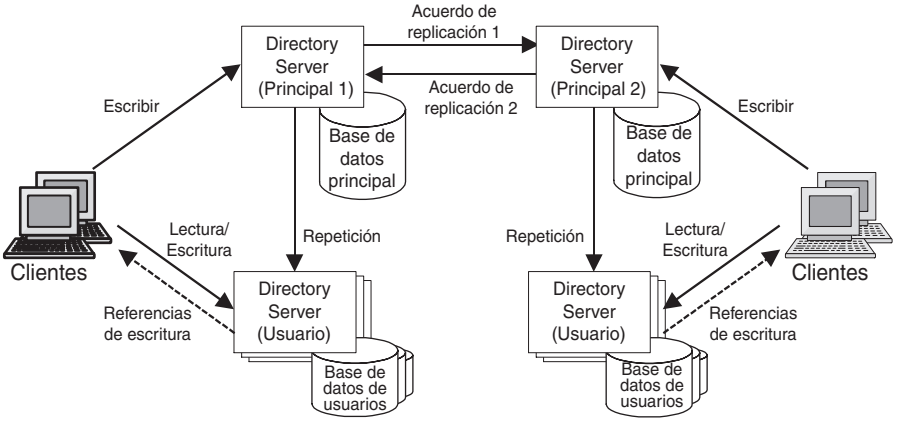
- Una instancia de Directory Server optimizada para las operaciones de lectura y escritura de la base de datos
- Un número de instancias del usuario de Directory Server optimizadas para las operaciones de lectura y búsqueda.
- Escalabilidad horizontal para las instancias de usuario de Directory Server

## Repetición de varias réplicas principales

La siguiente figura muestra una estrategia de repetición de varias réplicas principales que se puede utilizar para distribuir globalmente el acceso a los directorios.

En una repetición de varias réplicas principales, una o más instancias de Directory Server gestionan la base de datos de directorios principal. Cada réplica principal cuenta con un acuerdo de duplicación que especifica los procedimientos para sincronizar las bases de datos principales. Cada réplica principal se duplica en una serie de bases de datos de usuario. Al igual que con la repetición de una única réplica principal, las instancias de usuario de Directory Server se optimizan para el acceso de lectura y escritura. Cualquier operación de escritura recibida por un usuario se devuelve a la base de datos principal. La base de datos principal actualiza periódicamente las bases de datos de los usuarios.

**Figura 5-9** Ejemplo de repetición de varias réplicas principales




La estrategia de repetición de varias réplicas principales proporciona todas las ventajas de la repetición de una única réplica maestra más una estrategia de disponibilidad que puede proporcionar el equilibrado de carga para realizar las actualizaciones a las réplicas principales. También puede implementar una estrategia de disponibilidad que proporciona un control local de las operaciones de directorio, que es una consideración importante para las empresas con centros de datos distribuidos globalmente.

# Determinación de estrategias para la escalabilidad

La escalabilidad es la capacidad para aumentar la capacidad del sistema, normalmente mediante la adición de recursos de sistema, pero sin realizar cambios en la arquitectura de implementación. En el análisis de requisitos, normalmente se realizan predicciones del crecimiento previsto para un sistema basado en los requisitos empresariales y en los siguientes análisis de uso. Estas predicciones del número de usuarios de un sistema y la capacidad del sistema para satisfacer sus necesidades son a menudo cálculos que pueden variar en gran medida con respecto a los números reales del sistema implementado. El diseño debe ser lo suficientemente flexible como permitir una variación con respecto a las predicciones.

Un diseño escalable incluye la suficiente capacidad latente para tratar el aumento de la carga hasta que un sistema se pueda actualizar con recursos adicionales. Este tipo de diseños se puede escalar fácilmente para tratar el aumento de la carga sin tener que volver a diseñar el sistema.

## Capacidad latente

La capacidad latente es uno de los aspectos de escalabilidad en los que se incluyen los recursos de disponibilidad y rendimiento adicionales en el sistema de forma que se puedan afrontar cargas pico inusuales. También puede supervisar cómo se utiliza la capacidad latente en un sistema implementado para ayudar a determinar cuándo se debe escalar el sistema añadiendo recursos. La capacidad latente es una manera de crear seguridad en el diseño.

El análisis de casos de uso puede ayudar a identificar los escenarios que pueden crear cargas pico inusuales. Utilice este análisis de cargas máximas inusuales más un factor para cubrir el crecimiento imprevisto para diseñar la capacidad latente que crea seguridad en el sistema.

El diseño del sistema debe ser capaz de gestionar la capacidad prevista durante un tiempo razonable, normalmente los 6 a 12 primeros meses de funcionamiento. Los ciclos de mantenimiento se pueden utilizar para agregar recursos o aumentar la capacidad según se necesite. Idealmente, se deberían poder programar las actualizaciones del sistema periódicamente, pero a menudo, la predicción de los aumentos en la capacidad necesarios es una tarea difícil. Para determinar el momento de actualizar un servicio, se deberá realizar una supervisión exhaustiva de los recursos y de las previsiones empresariales.

Si tiene previsto implementar su solución en etapas progresivas, deberá programar el aumento de la capacidad del sistema para que coincida con otras mejoras programadas en cada etapa.

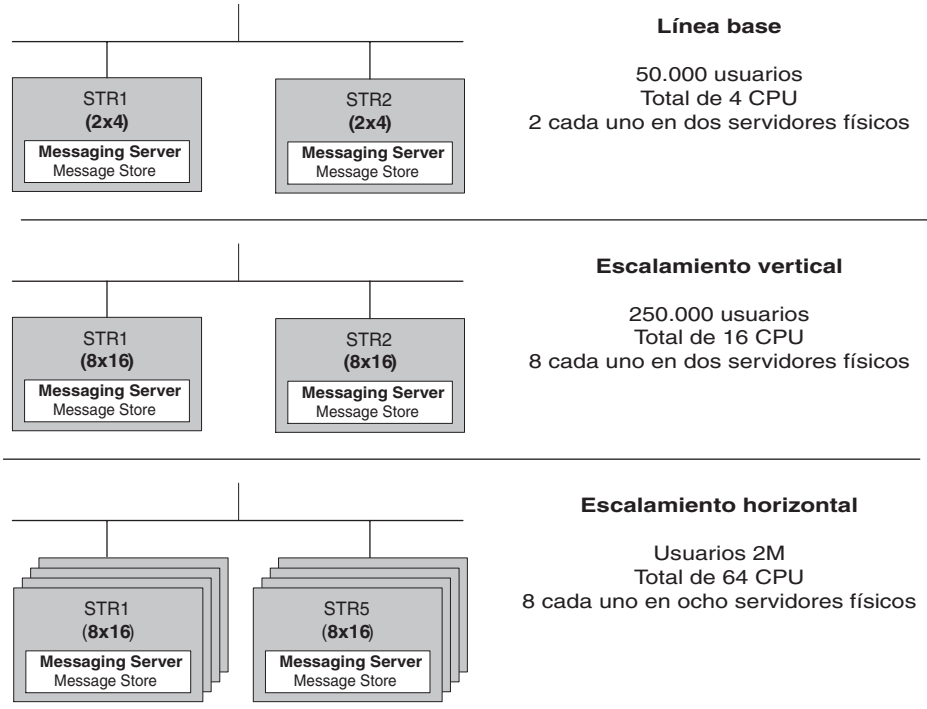
# Ejemplo de escalabilidad

El ejemplo de esta sección muestra el escalamiento horizontal y vertical de una solución que implementa Messaging Server. Para el escalamiento horizontal, se añaden CPU adicionales a un servidor para que pueda gestionar el aumento de la carga. Para el escalamiento vertical, se gestiona el aumento de la carga mediante la adición de servidores para poder distribuir la carga.

La base de este ejemplo supone una base de 50.000 usuarios que reciben el servicio mediante dos instancias de almacenamiento de mensajes que están distribuidas para el equilibrio de carga. Cada servidor cuenta con dos CPU para un total de cuatro CPU. La siguiente figura muestra cómo se puede escalar el sistema para tratar el aumento de cargas para 250.000 usuarios y 2.000.000 usuarios.

**NOTA** La Figura 5-10 muestra las diferencias entre el escalamiento vertical y el escalamiento horizontal. Esta figura no muestra otros factores que se deben tener en cuenta al realizar el escalamiento, como el equilibrado de carga, la recuperación tras error y los cambios en los patrones de uso.

**Figura 5-10** Ejemplos de escalamiento horizontal y vertical




# Identificación de cuellos de botella de rendimiento

Una de las claves para un diseño de implementación con éxito consiste en identificar los cuellos de botella del rendimiento y desarrollar una estrategia para evitarlos. Un cuello de botella del rendimiento se produce cuando la velocidad de acceso a los datos no cumple con los requisitos especificados para el sistema.

Los cuellos de botella se pueden ordenar de acuerdo con distintos tipos de hardware, tal y como se indica en la siguiente tabla de puntos de acceso de un sistema. Esta tabla también sugiere las soluciones potenciales para los cuellos de botella en cada clase de hardware.

**Tabla 5-8** Puntos de acceso a los datos

| Clase de hardware | Velocidad de acceso relativa | Soluciones para la mejora del rendimiento |
|---|---|---|
| Procesador | Nanosegundos | Escalamiento vertical: Añada más potencia de procesamiento, mejore la caché del procesador |
| | | Escalamiento horizontal: Añada potencia de procesamiento paralelo para el equilibrado de carga |
| Memoria del sistema (RAM) | Microsegundos | Memoria del sistema dedicada para tareas específicas |
| | | Escalamiento vertical: añada memoria adicional |
| | | Escalamiento horizontal: cree instancias adicionales para el procesamiento paralelo y el equilibrado de carga |
| Lectura y escritura en disco | Milisegundos | Optimice el acceso al disco con matrices de disco (RAID) |
| | | Dedique el acceso al disco a funciones específicas, como sólo lectura o sólo escritura |
| | | Introduzca en caché los datos a los que se accede frecuentemente en la memoria del sistema |
| Interfaz de red | Varía en función del ancho de banda y la velocidad de acceso de los nodos de la red | Aumente el ancho de banda |
| | | Añada hardware acelerador al transportar los datos seguros |
| | | Mejore el rendimiento de los nodos de la red, de forma que los datos estén accesibles más rápidamente |

**NOTA** La Tabla 5-8 muestra las clases de hardware en función de la velocidad de acceso relativa, implicando que los puntos de acceso lentos, como los discos, tienen una mayor probabilidad de ser la fuente de los cuellos de botella. Sin embargo, unos procesadores que no tengan la suficiente potencia para gestionar grandes cargas también son una fuente de cuellos de botella.

Normalmente, se comienza el diseño de implementación con los cálculos de potencia de procesamiento de base para cada componente de la implementación y sus relaciones de dependencia. A continuación se determina cómo evitar los cuellos de botella relacionados con la memoria del sistema y acceso a los discos. Por último, se examina la interfaz de la red para determinar los cuellos de botella potenciales y centrarse en las estrategias que se deben implantar para solucionarlos.

# Optimización de acceso al disco

Un componente vital del diseño de implementación es la velocidad de acceso al disco para los conjuntos de datos a los que se accede frecuentemente, como los directorios LDAP. El acceso al disco ofrece el acceso más lento a los datos y es probablemente el origen de un cuello de botella de rendimiento.

Una forma de optimizar el acceso al disco es separar las operaciones de escritura de las de lectura. Las operaciones de escritura no sólo son más caras que las de lectura, sino que normalmente las operaciones de lectura (operaciones de búsqueda en los directorios LDAP) se producen con mucha más frecuencia que las de escritura (actualizaciones a los datos en los directorios LDAP).

Otra manera de optimizar el acceso al disco es mediante discos dedicados en función de los distintos tipos de operaciones de E/S. Por ejemplo, proporcione un acceso al disco separado para las operaciones de registro de Directory Server, como registros de transacciones y de eventos, y operaciones de lectura y escritura LDAP.

Asimismo, considere la implementación de una o más instancias de Directory Server dedicadas a las operaciones de lectura y escritura y la utilización de instancias duplicadas distribuidas en servidores locales para el acceso de lectura y de búsqueda. Las opciones de encadenamiento y enlace también están disponibles para optimizar el acceso a los servicios de directorios.

El capítulo “Tamaño del sistema” en la *Directory Server Guía de planificación de la implementación*, http://docs.sun.com/doc/817-7607, plantea varios factores en la planificación del acceso al disco. Entre los temas tratados en este capítulo, se incluyen:

- **Requisitos mínimos de memoria y espacio en disco.** Ofrecer una estimación de la memoria y capacidad de disco necesarios para los distintos tamaños de directorios.
- **Establecimiento de tamaño de memoria física para el acceso a caché.** Ofrece unas directrices para estimar el tamaño de la caché de acuerdo con la utilización prevista de Directory Server y con la planificación de utilización total de la memoria.
- **Establecimiento del tamaño de los subsistemas de disco.** Ofrece información sobre los requisitos de espacio en disco de acuerdo con los sufijos de directorio y los factores de Directory Server que afectan a la utilización del disco y a la distribución de los archivos en los discos, incluidas varias alternativas de matrices de discos.

# Diseño para la óptima utilización de los recursos

El diseño de la implementación no es simplemente el cálculo de los recursos necesarios para satisfacer los requisitos de calidad de servicio. En el diseño de la implementación también se analizan todas las opciones disponibles y se selecciona la mejor solución que minimiza los costes pero que satisface los requisitos de calidad del servicio. Debe analizar los cambios asociados a cada decisión del diseño para asegurar que las ventajas que se ofrecen en un área no se ven anuladas por un incremento del coste en otra.

Por ejemplo, el escalamiento horizontal para la disponibilidad puede aumentar la disponibilidad general pero al coste de un mayor mantenimiento y servicio. El escalamiento vertical para el rendimiento puede aumentar la potencia de cálculo sin incurrir en gastos adicionales, pero la potencia adicional se puede utilizar de forma poco eficaz en algunos servicios.

Antes de completar la estrategia de diseño, examine las decisiones para asegurarse de que ha equilibrado la utilización de los recursos con las ventajas generales de la solución propuesta. Este análisis implica normalmente el examen de cómo afectan las calidades del sistema en un área a las de otra. La siguiente tabla muestra algunas calidades del sistema y las consideraciones correspondientes para la gestión de recursos.

**Tabla 5-9** Consideraciones para la gestión de recursos

| Calidad del sistema | Descripción |
|---|---|
| Rendimiento | En el caso de soluciones de rendimiento que concentran CPU en servidores individuales, ¿podrán los servicios utilizar eficazmente la potencia de cálculo? (Por ejemplo, algunos servicios tienen un límite sobre el número máximo de CPU que se pueden utilizar de forma eficaz.) |
| Capacidad latente | ¿La estrategia gestiona cargas que superan los cálculos de rendimiento? |
| | ¿Las cargas excesivas se tratan mediante el escalamiento vertical en los servidores, equilibrando la carga en otros servidores o con ambas acciones? |
| | ¿La capacidad latente es suficiente para tratar las cargas pico inusuales hasta que alcance el siguiente hito de escalamiento de la implementación? |
| Seguridad | ¿Ha previsto el aumento de rendimiento necesario para tratar las transacciones seguras? |
| Disponibilidad | Para las soluciones redundantes horizontalmente, ¿ha calculado los gastos de mantenimiento a largo plazo? |
| | ¿Ha tenido en cuenta el tiempo de inactividad programado necesario para mantener el sistema? |
| | ¿Ha equilibrado los costes entre servidores de gama alta y de gama baja? |
| Escalabilidad | ¿Ha calculado hitos para escalar la implementación? |
| | ¿Cuenta con una estrategia adecuada para ofrecer la suficiente capacidad latente para tratar los aumentos de carga previstos hasta que se alcancen los hitos para escalar la implementación? |
| Facilidad de mantenimiento | ¿Ha tenido en cuenta los costes de administración, supervisión y mantenimiento en el diseño de disponibilidad? |
| | ¿Ha considerado soluciones de administración delegada (que permiten que los usuarios finales realicen algunas tareas de administración) para reducir los costes de administración? |

# Gestión de riesgos

La mayor parte de la información en la que está basada el diseño de la implementación, como los requisitos de calidad de servicio y análisis de uso, no está compuesta por datos empíricos sino por datos basados en los cálculos y proyecciones obtenidos en última instancia de los análisis de negocios. Estas previsiones pueden ser imprecisas por varios motivos, como circunstancias imprevistas en el clima empresarial, métodos defectuosos de recopilación de los datos o simplemente debido a errores humanos. Antes de completar un diseño de implementación, vuelva a comprobar los análisis en los que está basado el diseño y asegúrese de que el diseño tiene en cuenta cualquier desviación razonable de los cálculos o previsiones.

Por ejemplo, si el análisis de uso subestima la utilización real del sistema, corre el riesgo de crear un sistema que no pueda manejar la cantidad de tráfico que se encuentre. Un diseño que tenga un rendimiento inferior al previsto se considerará un fallo.

Por otro lado, si crea un sistema que es mucho más potente de lo necesario, estará utilizando recursos que se podrían utilizar en otro lugar. La clave es incluir un margen de seguridad por encima de los requisitos pero evitar el uso desmesurado de los recursos.

El uso desmesurado de los recursos se traduce en un error del diseño porque los recursos infrautilizados se podrían aplicar a otras áreas. Además, los participantes pueden percibir estas soluciones de recursos excesivos como que no cumplen con los contratos en buena fe.

# Ejemplo de arquitectura de implementación

La siguiente figura representa una arquitectura de implementación completada para la implementación de ejemplo presentada anteriormente en esta guía. Esta figura proporciona una idea de cómo se debe presentar una arquitectura de implementación.

**PRECAUCIÓN** La arquitectura de implementación de la siguiente figura sólo es válida para fines ilustrativos. No representa una implementación que se haya diseñado, creado o probado y no debería considerarse como un consejo para la planificación de la implementación.

**Figura 5-11** Ejemplo de arquitectura de implementación