

# Rendimiento y escalabilidad

## Introducción



USAC - Análisis y diseño de sistemas 2 - 1er semestre 2016  
Ing. Ricardo Morales



# Introducción

# Definiciones

---

## Rendimiento

- Mide que tan rápido y eficientemente puede un sistema de software completar ciertas tareas de computación

## Escalabilidad

- Mide la tendencia del rendimiento con incrementos a la carga



# Métricas

---

Para tareas de computación tipo OLTP, consistentes de actividades interactivas del usuario, la métrica de tiempo de respuesta es usada para medir que tan rápido puede responder un sistema a las solicitudes de usuarios interactivos

Para jobs batch no interactivos, la métrica de desempeño (throughput) es utilizada para medir el número de transacciones que un sistema puede completar en un período de tiempo



# Rendimiento y escalabilidad

---

- ▶ El rendimiento y la escalabilidad son inseparables, no tiene sentido hablar de escalabilidad si un sistema no tiene buen rendimiento, sin embargo un sistema puede tener buen rendimiento pero no escalar



# Factores que impactan

---

- ▶ Los 3 factores que mas impactan el rendimiento y escalabilidad de un sistema de software son:

Las capacidades de la  
plataforma de hardware

La madurez de la  
plataforma de software  
(sistema operativo, drivers,  
stack de máquinas  
virtuales, ambiente de  
runtime, etc.)

Su diseño e  
implementación





# Stack de Software

# Hardware

---

- ▶ Entender el rendimiento y escalabilidad del software empieza con entender las computadoras, porque es el hardware el que eventualmente determinará el rendimiento y escalabilidad de un programa de software, después de realizar todas las optimizaciones y afinaciones en el lado del software

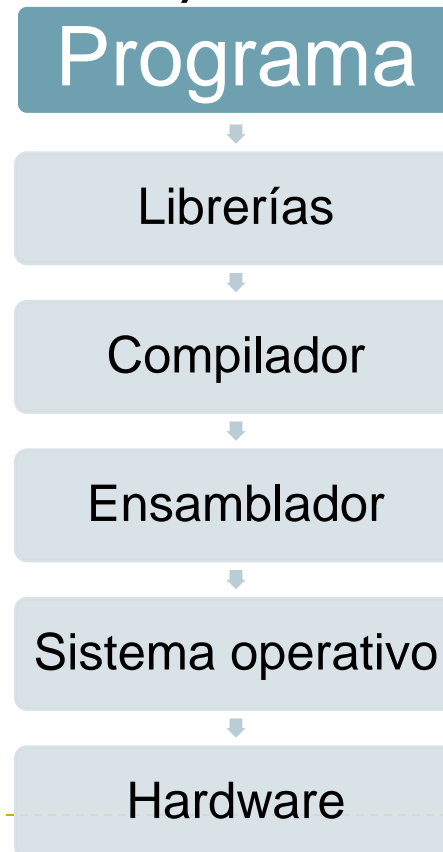




# Software stack

---

- ▶ Consiste de múltiples capas
- ▶ El rendimiento y escalabilidad de cada capa es crítico para el rendimiento y escalabilidad del sistema como un todo



## Software stack (II)

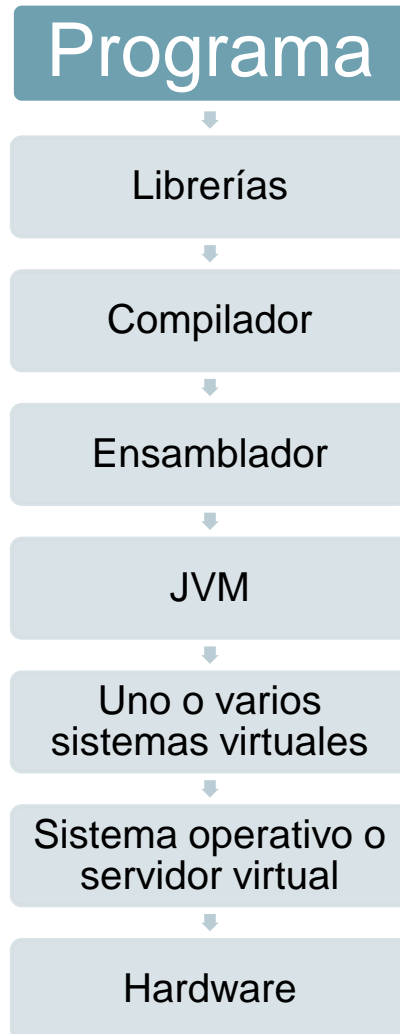
---

- ▶ Cada capa mostrada en la gráfica anterior es un factor de rendimiento y escalabilidad para los programas contruidos para correr sobre dicho stack
- ▶ Algunas capas se pueden afinar mas que otras, desde las perspectivas de rendimiento y escalabilidad de software
- ▶ Es importante asegurarse que todas las oportunidades de optimización en cada capa han sido aplicadas apropiadamente



# Máquinas virtuales y sistemas virtuales en el stack

---




# Multithreading

---

- ▶ Primero que nada, un thread es diferente a un proceso en ciencias de la computación
- ▶ Un proceso es un programa stand-alone, mientras que los threads generalmente son generados dentro de un proceso, sin cruzar los límites de dicho proceso
- ▶ Multithreading es una de las tecnologías de software mas importantes para aumentar el rendimiento y escalabilidad de todos los tipos de software
- ▶ Permite que varias tareas se ejecuten simultáneamente o en paralelo para obtener lo mejor del hardware con varios CPUs
- ▶ Siempre que se analiza un problema de rendimiento o escalabilidad, se debe iniciar con el uso de CPU del sistema a analizar





## Probando rendimiento y escalabilidad de software

# Valor de las pruebas

---

- ▶ Pruebas completas de rendimiento y escalabilidad pueden proveer una base cuantitativa para oportunidades posteriores de optimizaciones y afinaciones
- ▶ Es importante pensar con cuidado como se deben diseñar y ejecutar las pruebas de rendimiento y escalabilidad
- ▶ Así mismo, como evaluar los resultados cuantitativos y como transformar ese trabajo de pruebas en valor para la organización



# Equipos de hardware

---

- ▶ Con relación a los equipos, una regla general es que los sistemas que se usan para pruebas de rendimiento y escalabilidad deben ser como mínimo 2 o 4 veces mas potentes que los sistemas de desarrollo o QA, en términos de poder de CPU (total de gigahertz de todos los CPUs) y frecuencia de CPU



# Alcance de pruebas de rendimiento y escalabilidad

---

- ▶ Las pruebas no funcionales de rendimiento y escalabilidad de software se pueden clasificar en las siguientes categorías:

Pruebas de regresión de rendimiento, para rastrear cambios en rendimiento debido a cambios en el código de un release a otro

Optimizaciones de rendimiento y pruebas de afinamiento, para soportar esfuerzos de optimización y afinamiento

Pruebas de rendimiento de benchmarking, para saber que rendimiento deben esperar los clientes del software si usan cargas reales con hardware similar

Pruebas de escalabilidad, para chequear si el software puede satisfacer las necesidades crecientes del negocio



# Pruebas de regresión de rendimiento

---

- ▶ Las pruebas de regresión son una medida para verificar si nuevos cambios en el código han arruinado algo que funcionaba previamente
- ▶ El concepto aplica también al rendimiento, no solo a la parte funcional
- ▶ Es importante no solo para minimizar los riesgos a los clientes del negocio, sino también para un rastreo interno de degradaciones y mejoras de rendimiento debido a cambios a nivel de código fuente
- ▶ La frecuencia de estas pruebas no es la misma que la de las pruebas funcionales, generalmente se pueden correr por iteración



# Optimizaciones de rendimiento y pruebas de afinamiento

---

- ▶ Existen 2 tipos de pruebas
  - ▶ Pruebas de rendimiento para optimizar software al nivel implementación de código (profiling, análisis de queries)
  - ▶ Pruebas de rendimiento para afinar el rendimiento del software, al variar los parámetros de configuración de la plataforma de hardware, plataforma de software y aplicación, sin incurrir en cambios a nivel de código fuente



# Pruebas de rendimiento

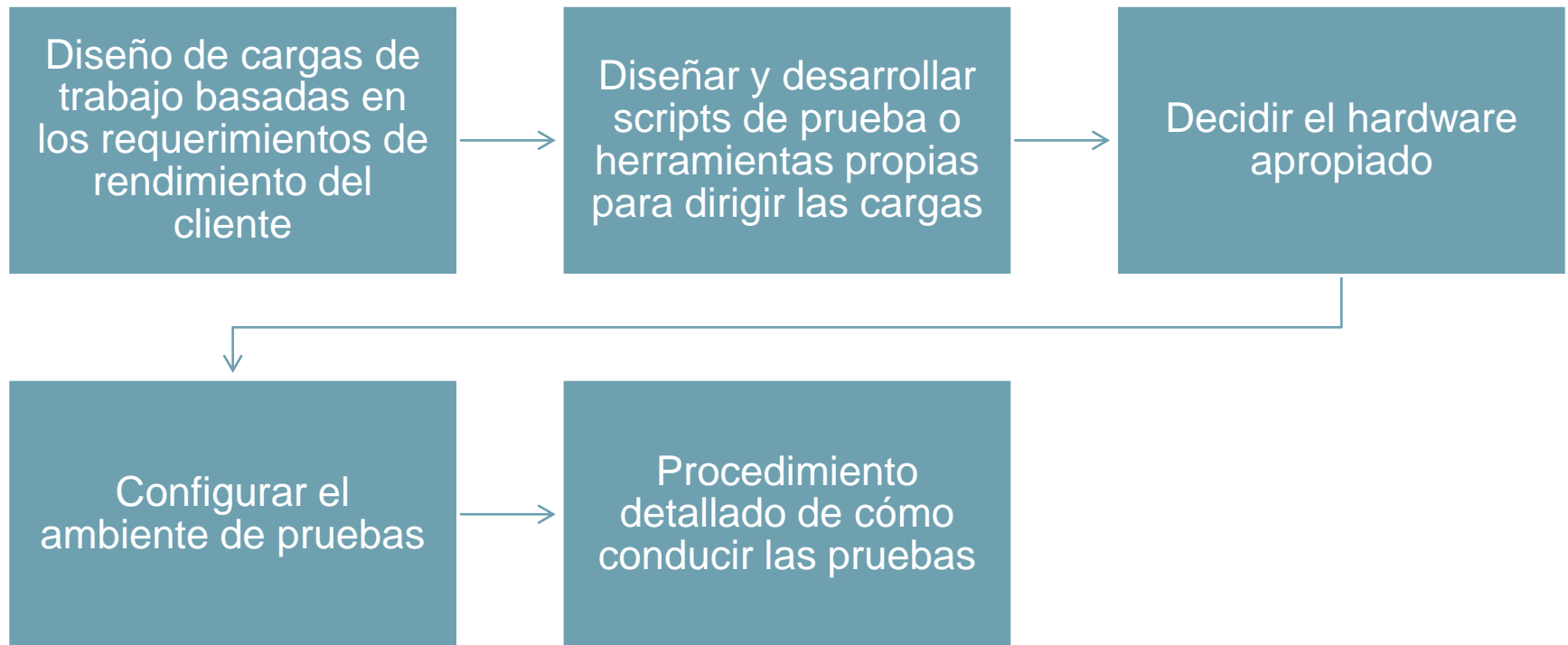
---

- ▶ Las pruebas de rendimiento deben ser parte del ciclo entero de desarrollo de software
- ▶ Se debe estar refinando las metodologías y herramientas de pruebas para alcanzar la mas alta productividad y eficiencia posibles
- ▶ El no alcanzar lo anterior se puede deber a:
  - ▶ Las pruebas de rendimiento de software tienen sus propios retos comparado con pruebas de funcionalidad y escribir código
  - ▶ Las pruebas de rendimiento requieren que se piense cuidadosamente acerca de cada detalle de las pruebas
  - ▶ En un sentido mas general, un trabajo de rendimiento de software requiere habilidades de agudeza, independencia y pensamiento creativo



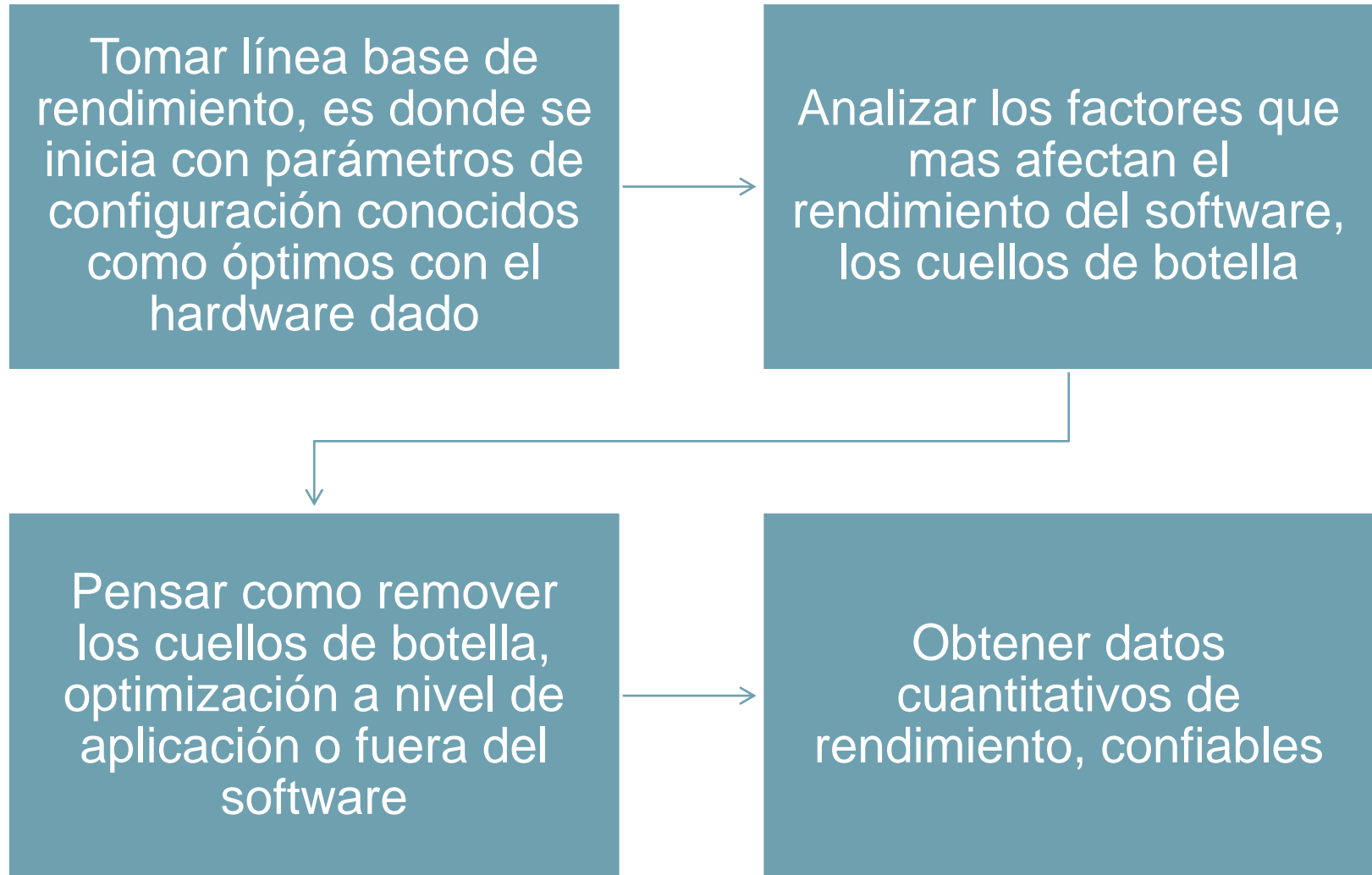
# Pasos para pruebas de rendimiento

---



## Pasos para pruebas de rendimiento (cont.)

---



# Entregables de pruebas de rendimiento

---

- ▶ Números de rendimiento, tales como tiempos de respuesta o desempeño, que importen mas a los clientes
- ▶ Una lista exacta de las condiciones usadas para obtener los resultados:
  - ▶ Especificaciones de hardware
  - ▶ Stack de software usado
  - ▶ Configuraciones de hardware y software
  - ▶ Reportes de actividad de base de datos
  - ▶ Métricas de rendimiento de servidores
  - ▶ Un análisis de cuellos de botella, que brinden indicaciones de cómo optimizar o afinar el rendimiento del software



# Pruebas de rendimiento de benchmarking

---

- ▶ Es una forma de mostrar que tan bueno es el rendimiento del software y que también dicho software puede alcanzar las necesidades de los clientes, en relación a los competidores, al usar cargas de trabajo mas realísticas y hardware cercano al de producción



# Pruebas de escalabilidad

---

- ▶ Es similar a pruebas de stress en las cuales se llevan las pruebas de rendimiento al extremo para encontrar cuales son los límites
- ▶ Es generalmente una extensión de las pruebas regulares de rendimiento, con mas usuarios y sets de datos mas grandes





# Definiendo rendimiento de software

---

- ▶ Rendimiento es esencialmente la velocidad a la cual alguna actividad o trabajo es completado, de inicio a fin
- ▶ En el caso del software lo que interesa son 2 tipos de tareas de computación o cargas de trabajo: OLTP y batch jobs
- ▶ Una carga OLTP es simplemente una representación de las actividades interactivas del usuario, usualmente se caracteriza por el tipo de actividad del usuario y el número de usuarios asociados a cada una
- ▶ Una carga de batch job especifica la cantidad de trabajo a ser procesado sin la necesidad de la intervención del usuario



# Métricas de rendimiento para cargas de trabajo OLTP

---

- ▶ El rendimiento de un sistema de software tipo OLTP es medido por el tiempo de respuesta que obtienen los usuarios
- ▶ El tiempo de respuesta es la duración en tiempo medida desde la perspectiva del usuario, desde que se inicia una acción hasta recibir respuesta del sistema
- ▶ Es importante no confundir el tiempo de respuesta con el tiempo de pensar.
- ▶ El tiempo de pensar mide la duración en tiempo usada por el usuario en preparar entradas y digerir el contenido de la respuesta hasta iniciar la siguiente acción en el sistema



# Métricas de rendimiento para cargas de trabajo OLTP (II)

---

- ▶ La diferencia entre ambos tiempo, lleva a la diferenciación entre usuarios activos y usuarios concurrentes: 
$$N_{concurrent} = N_{active} \left( \frac{Tiempo\_respuesta}{Tiempo\_respuesta + Tiempo\_pensar} \right)$$
- ▶ Los usuarios activos son aquellos que están estresando el sistema
- ▶ El tiempo total en una transacción tiene 3 partes:

$$OLTP\_txn\_time = User\_time + System\_time + Browser\_time$$



# Métricas de rendimiento para batch jobs

---

- ▶ El rendimiento de un batch job es medido en desempeño, en contraste con el tiempo de respuesta para cargas OLTP
- ▶ El desempeño mide el número de tareas completadas o el número de objetos creados dentro de un período de tiempo dado



# Naturaleza estocástica de las medidas de rendimiento de software

---

- ▶ Es un hecho que ambos, números de tiempo de respuesta y números de desempeño, son estocásticos (patrón que puede ser analizado estadísticamente, pero no puede ser predicho precisamente) por naturaleza
- ▶ En general, se debe ser cuidadoso al interpretar cambios en números de rendimiento menores de 10%, como mejoras o degradaciones asociadas a cambios de código o de configuración
- ▶ Se debe tener en mente que los resultados de rendimiento son números estadísticos y las estadísticas contienen errores



# Ley de Amdahl

---

- ▶ Originada de exploraciones hace varias décadas en cuanto a como mejorar el rendimiento de un sistema de computación usando múltiples procesadores paralelos
- ▶ Se adapta para evaluar el rendimiento de un sistema que consiste de una serie de subsistemas tales como un sistema típico de 3 capas, compuesto por un web server, un application server y un servidor de base de datos
- ▶ Ya que es mejor explorar múltiples factores, uno a la vez, se asume que se tiene un sistema que consiste de 2 subsistemas



# Ley de Amdahl (II)

---

## ► Donde

- G=ganancia de rendimiento
- n=mejora de tiempo esperado, n veces
- $f=T2/(T1+T2)$ , factor de impacto

$$G = \frac{1}{1 - f + \frac{f}{n}}$$



# Factores de rendimiento y escalabilidad de software

---

Rendimiento y escalabilidad de la plataforma de hardware. Hay 4 categorías de factores de hardware: CPU, memoria, almacenamiento y red

Como está configurado el hardware

Plataforma del sistema operativo

Plataforma del sistema de base de datos

Configuración del software en si mismo

Como está diseñado e implementado el producto de software





# Contador de rendimiento

---

- ▶ Es una entidad lógica que representa uno de los aspectos de un recurso cuantitativamente, por ejemplo:
  - ▶ ¿Qué tan ocupado está el CPU?
  - ▶ ¿Qué tanta memoria está siendo usada por la aplicación bajo pruebas?
  - ▶ ¿Qué tan ocupados están los discos de almacenamiento?
  - ▶ ¿Qué tan ocupada esta la red?

