Actualización de Servicios

Unidad 5
Gestión de dispositivos y usuarios

Índice

Bibliografía

[CAT+01] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin Vahdat, Ronald P. Doyle: “Managing Energy and Server Resources in Hosting Centres”. 18th ACM Symp. on Oper. Syst. Princ. (SOSP), Chateau Lake Louise, Banff, Alberta, Canada, octubre 2001: 103-116.

[CCW06] Frederick Chong, Gianpaolo Carraro, Roger Wolter: “Multi-Tenant Data Architecture”, Microsoft Developer Network, junio 2006.

[JDU+74] David S. Johnson, Alan J. Demers, Jeffrey D. Ullman, M. R. Garey, Ronald L. Graham: “Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms”. SIAM J. Comput. 3(4): 299-325 (1974)

[LHM+13] Ziyang Liu, Hakan Hacigümüs, Hyun Jin Moon, Yun Chi, Wang-Pin Hsiung: “PMAX: tenant placement in multitenant databases for profit maximization”. EDBT 2013: 442-453

[LPN09] Willis Lang, Jignesh M. Patel, Jeffrey F. Naughton: “On energy management, load balancing and replication”. SIGMOD Record 38(4): 35-42 (2009)

[LSP+12] Willis Lang, Srinath Shankar, Jignesh M. Patel, Ajay Kalhan: “Towards Multi-tenant Performance SLOs”. 28th IEEE Intnl. Conf. on Data Eng. (ICDE), abril 2012: 702-713

[Sto86] Michael Stonebraker: “The Case for Shared Nothing”. IEEE Database Eng. Bull. 9(1): 4-9 (1986)

Objetivos

- Revisar y analizar las implicaciones del multiarrendamiento en la gestión de recursos de un sistema de computación en la nube.
- Conocer y evaluar la incidencia del multiarrendamiento sobre el rendimiento global y los SLA.
- Analizar y aplicar adecuadamente diferentes estrategias de planificación y distribución de carga para minimizar el consumo energético en una infraestructura escalable.

Índice

1. Introducción

- La unidad 1 presentó el concepto de servicio...
  - Las empresas clientes se despreocupan de su desarrollo.
    - Así reducen riesgos y gastos.
    - Se espera que los proveedores generen servicios...
      - Más eficientes.
      - Mejor administrados.
      - Más económicos.
    - … debido a la economía de escala.

1. Introducción

- Los proveedores...
  - Deben ser capaces de ofrecer un mismo servicio a múltiples empresas clientes.
  - Adaptándolo a las necesidades y preferencias de cada una.
- Esto conduce al “multiarrendamiento” (“*multi-tenancy*”).
- Estudiemos qué dificultades entrañará esta gestión.

Índice

2. Multiarrendamiento ("Multitenancy")

- Arquitectura de multiarrendamiento:
  - Una misma instancia de la aplicación es utilizada por múltiples empresas clientes simultáneamente.
    - Típica en la computación en la nube con modelo de servicio SaaS.
    - Ya se utilizó en la década de los 60.
      - Sistemas de tiempo compartido.
      - Basados en ordenador central.
      - Compartido por múltiples usuarios.

2. Multiarrendamiento ("Multitenancy")

- Comparar con arq. multi-instancia:
  - Cada usuario instala la aplicación en "su" máquina.
    - En exclusiva.
  - Típico en entornos personales.
  - No admite economía de escala.
    - El usuario prevé sus necesidades.
    - Adquiere y configura su(s) máquina(s).
    - Administra la aplicación.
    - Mayores costes, generalmente.

2. Multiarrendamiento ("Multitenancy")

- Aspectos a analizar:
  - 2.1. Objetivos.
  - 2.2. Clases.
  - 2.3. Gestión de datos.
  - Impacto del multiarrendamiento sobre SLA / SLO.
    - Ver sección 3.

2.1. Objetivos

- Un modelo de servicio SaaS:
  - Proporciona transparencia sobre el número de máquinas utilizadas para proveer un servicio.
  - Desde un punto de vista lógico, sólo hay una instancia del servicio.
    - Utilizada por múltiples clientes.
    - Implantada en un número variable de máquinas.
      - Arquitectura de multiarrendamiento.

2.1. Objetivos

- Cliente:
  - Percibe la imagen de que el servicio se ofrece en exclusiva para él.
    - Aislamiento del resto de arrendatarios.
    - En algunos casos, personalización de la interfaz.
    - Arrendatarios empresariales: SLA.
      - Rendimiento previsible.
  - Menores riesgos y costes.
    - Servicio desarrollado por un equipo de programadores experto en esa área.
    - Sistema elástico. Economía de escala.

2.1. Objetivos

- Eficiencia:
  - Algunos recursos / datos únicamente requieren accesos de lectura.
    - Pueden ser compartidos por todos los arrendatarios.
    - La ratio usuarios/réplica puede ser muy alta.
      - Gestión más eficiente.
      - El proveedor necesita poco espacio para guardar esa información.

2.1. Objetivos

- Datos persistentes:
  - Será el recurso que requerirá una gestión más compleja.
  - Objetivos contrapuestos:
    - El arrendatario busca aislamiento, seguridad y rendimiento.
    - El proveedor soporta compartición y concurrencia.
      - Para reducir costes.
      - Sin comprometer SLA.

2.1. Objetivos

- Datos persistentes (ii):
  - Como la aplicación/servicio es el mismo...
    - Cada cliente debería necesitar un mismo esquema en su BD.
    - Las diferencias residirían en el contenido manejado por cada empresa cliente.
      - Pero no en el esquema.
  - ¿A qué nivel de compartición se debe llegar?

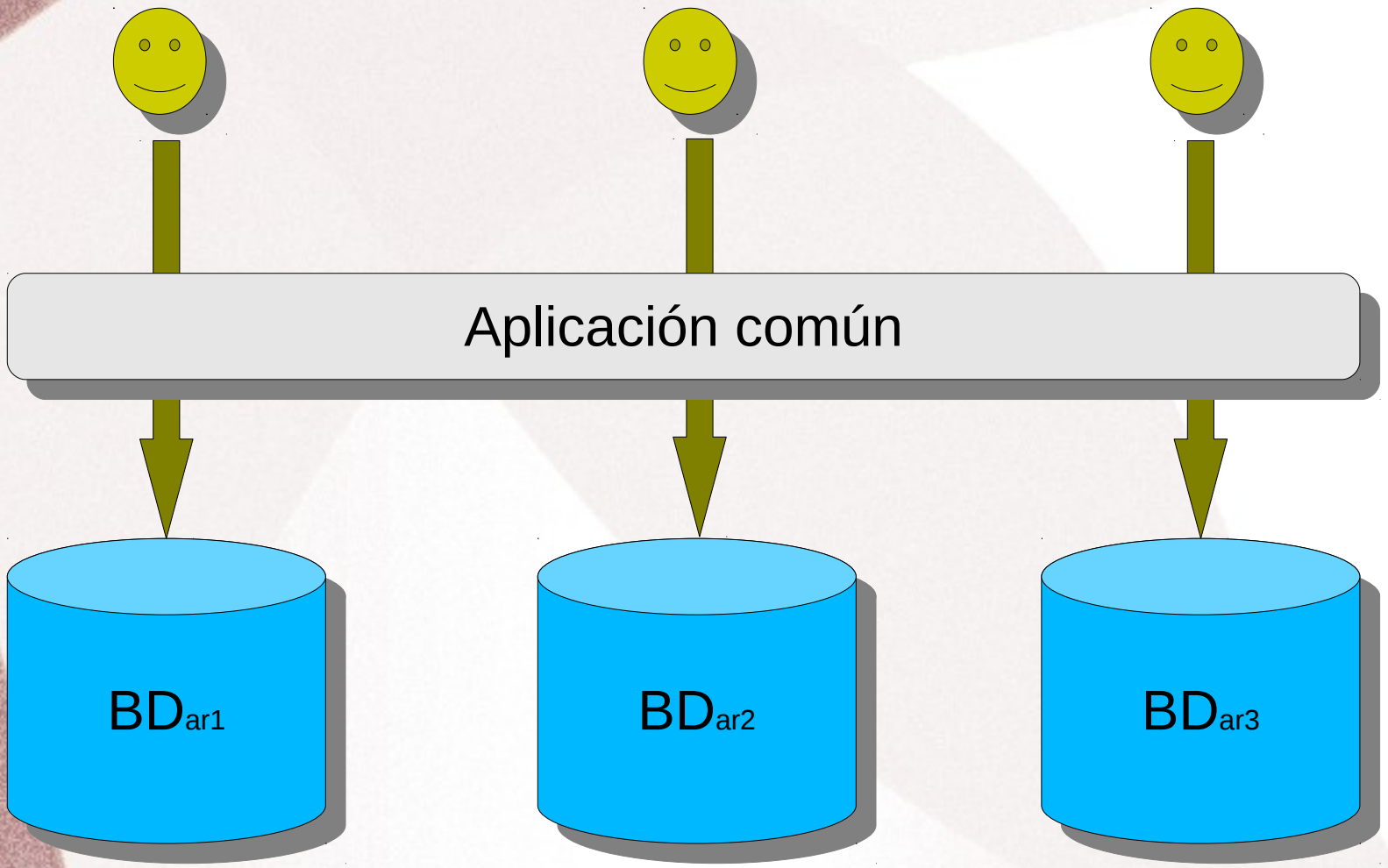
2.2. Clases

- Arquitecturas de gestión de datos persistentes bajo multiarrendamiento.
- Tres clases [CCW06]:
  - BB.DD. separadas.
  - Esquemas separados.
  - Esquema compartido.

2.2.1. BB.DD. separadas

- Los arrendatarios comparten el código de la aplicación, pero cada uno tiene su propia BD.
- Ventajas:
  - Aislamiento óptimo.
  - Garantías de seguridad.
  - Personalización sencilla.
- Inconvenientes:
  - Coste económico alto.
  - Resulta difícil compartir recursos “hardware”.

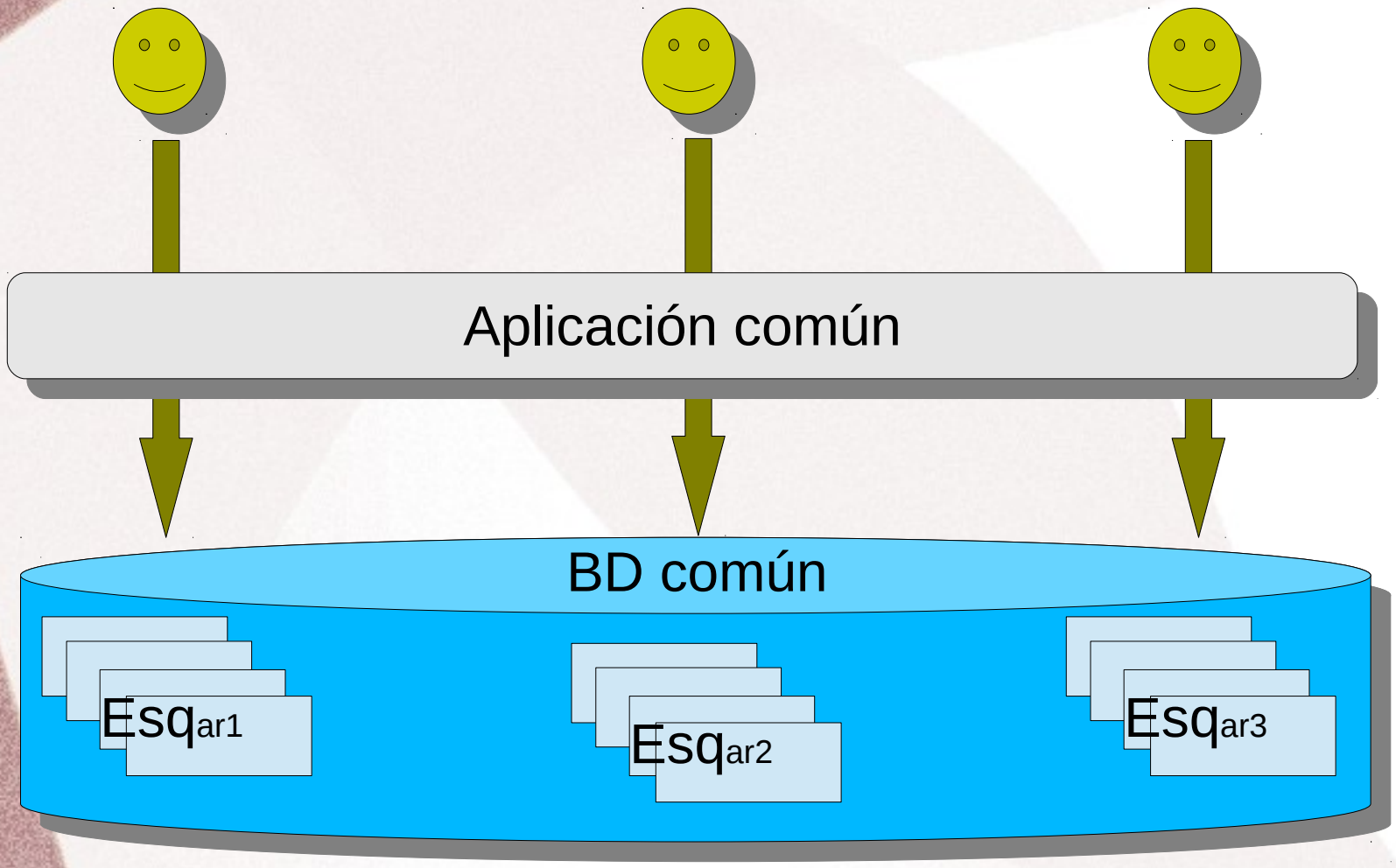
2.2.1. BB.DD. separadas



2.2.2. Esquemas separados

- BD idéntica para todos los arrendatarios.
  - Cada uno maneja su copia propia del conjunto de tablas de la BD.
- Ventajas:
  - Las mismas del caso anterior.
  - Leve rebaja en costes.
- Inconvenientes:
  - Si el servidor del SGBD falla, afecta a todos los arrendatarios.
  - Si hay que restaurar desde una copia de seguridad, otros arrendatarios pueden perder sus modificaciones recientes.

2.2.1. Esquemas separados



2.2.3. Esquema compartido

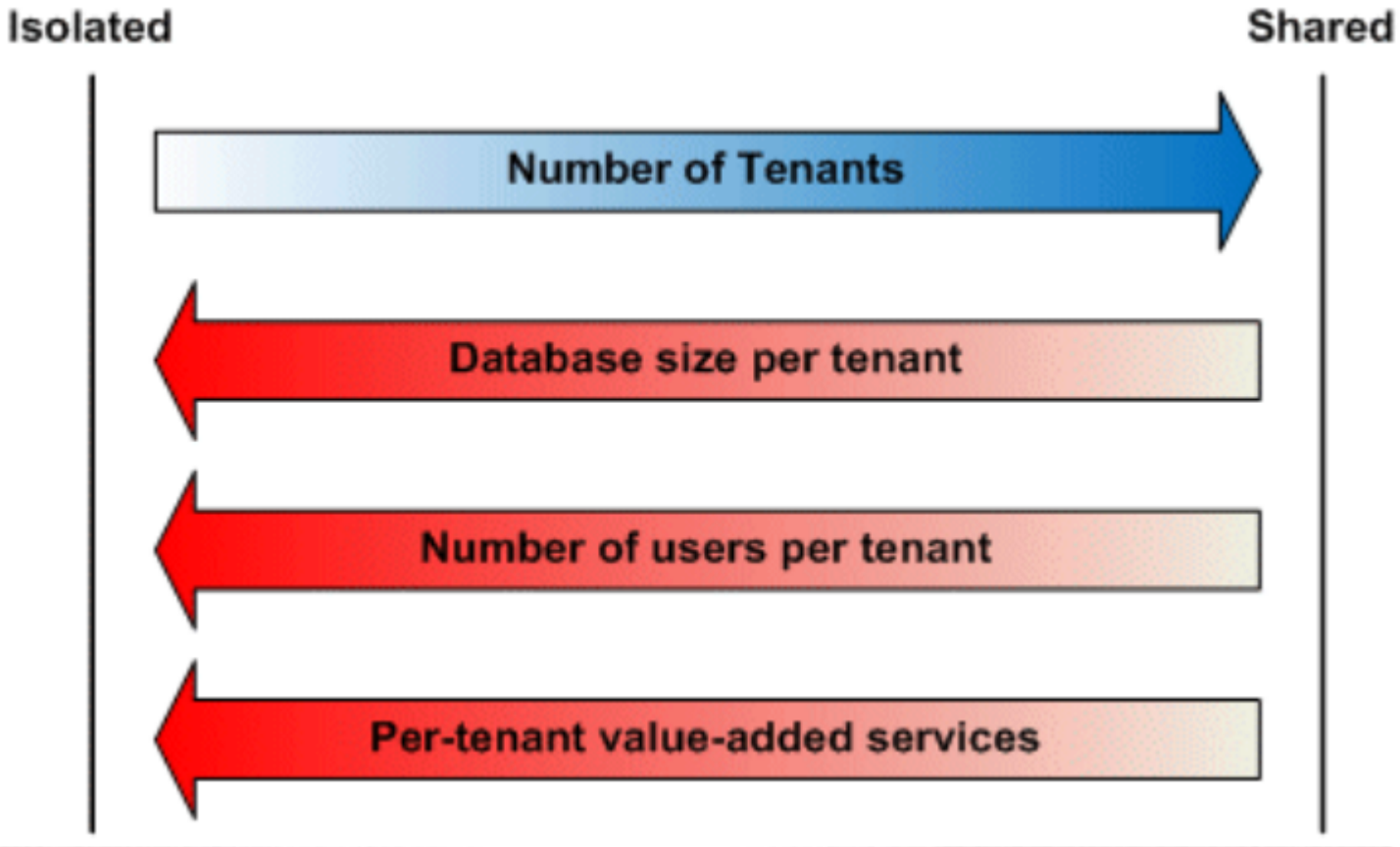
- Existe una sola BD y un solo conjunto de tablas.
  - Compartidos por todos los arrendatarios.
- Cada tabla:
  - Incluye una columna con el identificador de arrendatario.
    - Así se distinguen las filas de cada empresa cliente.

2.2.3. Esquema compartido

- Ventajas:
  - Mínima inversión en HW.
- Inconvenientes:
  - Personalización difícil.
    - Esquema idéntico para todos los arrendatarios.
  - Mínimas garantías de seguridad.
    - Diseño más complejo.
    - Incrementa los costes de desarrollo y depuración.
  - Pésima gestión de fallos.
    - Si un arrendatario recuperase sus datos desde una copia de seguridad, entorpecería a todos los demás.

2.2. Clases

- ¿Qué clase interesa más? [CCW06]



2.3. Gestión de datos

- Aspectos a considerar [CCW06]:
  - Seguridad.
  - Extensibilidad / personalización.
  - Escalabilidad.
- Solución sencilla con BB.DD. separadas y compleja con esquema compartido.
  - Analicemos ese segundo caso.

2.3.1. Seguridad

- Mecanismos existentes:
  - Listas de control de acceso asignables a tablas o filas.
    - Permite especificar qué arrendatarios pueden acceder y de qué manera a cada elemento.
  - Definición de vistas.
    - Cada vista puede ser asignada a un arrendatario diferente.
  - Cifrado de elementos críticos.
    - Evita que esa información pueda ser utilizada por otros arrendatarios, pero ralentiza los accesos a su propietario.

2.3.1. Seguridad

- Esos mecanismos...
  - Gestionan los aspectos de seguridad entre arrendatarios diferentes.
    - Pero los arrendatarios no suelen ser los usuarios finales.
    - Son empresas que ofrecen sus servicios a otros usuarios.
    - ¿Cómo gestionar ese segundo nivel de protección?
      - Los SGBD no suelen facilitar otros mecanismos para refinar más.
      - ¡Por eso el diseño y desarrollo de una gestión compartida es más difícil!!

2.3.2. Personalización

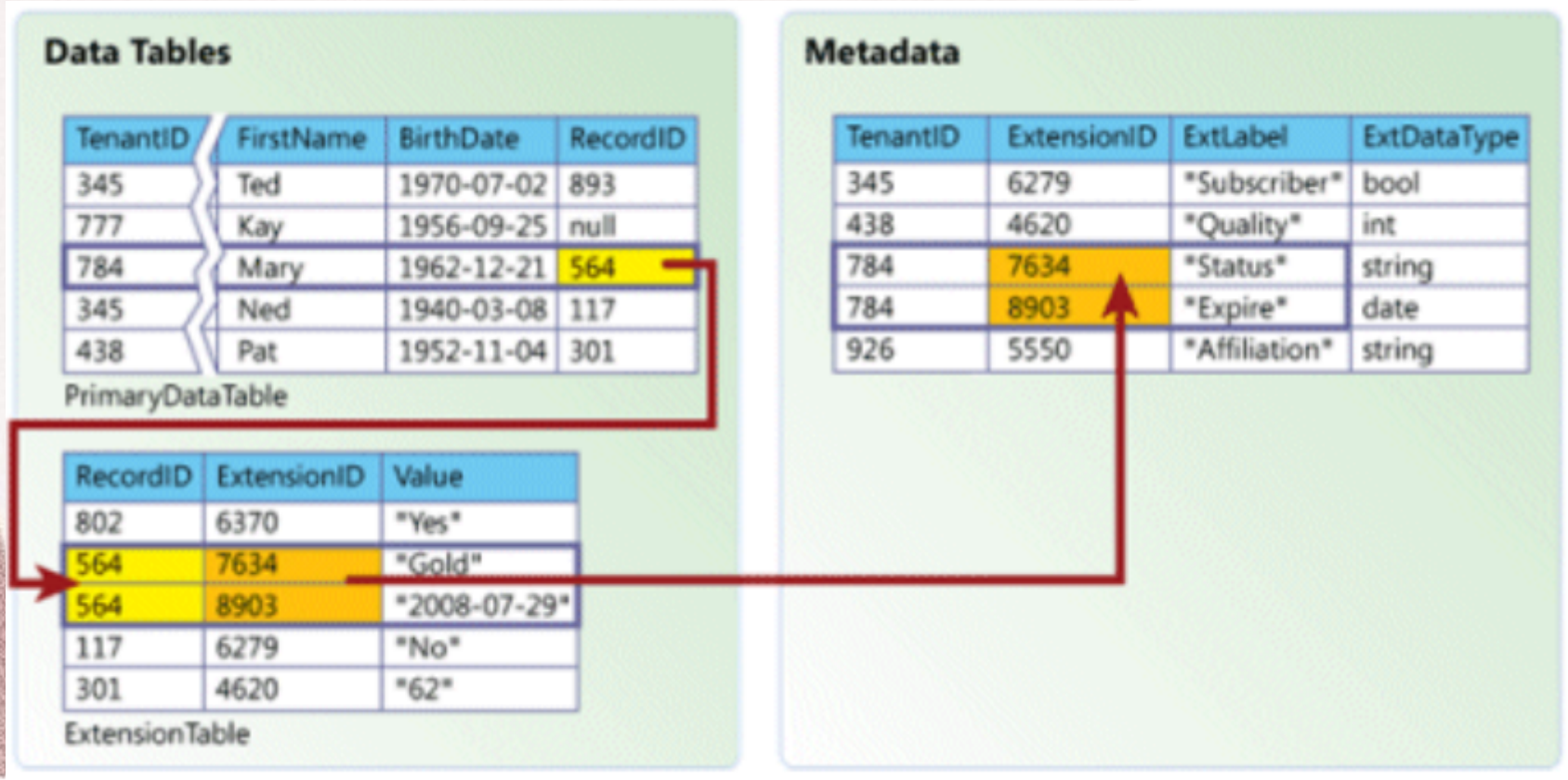
- En las tres clases se asume:
  - Aplicación común.
  - Esquema inicial idéntico
- ...para todos los arrendatarios.
- En BB.DD. separadas y esquemas separados...
  - No hay restricciones para modificar el esquema de cada arrendatario.
  - ¡Pero en la clase “esquema compartido” resulta muy difícil!!

2.3.2. Personalización

- Soluciones:
  - Disponer de columnas / atributos adicionales en cada tabla.
    - P.ej., de tipo cadena.
    - La aplicación ya realizará el “casting” una vez obtenido el dato.
    - Problemas:
      - Se malgasta espacio si los arrendatarios no llegan a utilizar estos campos.
      - ¿Y si algún arrendatario necesitase más campos?
        - Solución poco flexible.

2.3.2. Personalización

- Soluciones (ii):
  - Utilizar una tabla de extensión y una columna adicional en la tabla extendida.
  - Ver ejemplo:



2.3.2. Personalización

- Soluciones (ii):
  - El valor insertado en la columna adicional (si no es nulo)...
    - Permite averiguar cuántos campos adicionales se utilizan en esa fila, y el código (“ExtensionID”) y valor asignados a cada uno.
    - Después con el código y el identificador de arrendatario, se podrá consultar en otra tabla de metadatos el tipo y el nombre de los campos adicionales.

2.3.2. Personalización

- ¡Pero la dificultad no sólo está en la extensión del esquema!!!
  - Se suponía que la aplicación debía ser común para todos los arrendatarios.
  - Si personalizamos el esquema para algunos de ellos, habrá que generar código adicional en la aplicación para que sepa gestionar esas diferencias.

2.3.3. Escalabilidad

- ¿Cómo mejorarla?
  - Mediante particionado [Sto86] horizontal.
    - Repartir conjuntos de filas consecutivas entre diferentes nodos.
    - Debe tenerse en cuenta a qué arrendatario pertenece cada rango.
    - Debe considerarse también la frecuencia de acceso sobre cada rango.
    - ¡El objetivo es también equilibrar la carga!!!

Índice

3. Planificación, SLO y multiarrendamiento

- El multiarrendamiento exige una gestión más compleja si...:
  - El servicio ofertado gestiona un gran volumen de información persistente.
    - DaaS
  - El SLA considera:
    - Disponibilidad
    - **¡Rendimiento (transac/seg, TPS)!!**
  - El coste del HW y del SGBD es alto.

3. Planificación, SLO y multiarrendamiento

- En ese caso...:
  - Hay que gestionar adecuadamente la distribución de los arrendatarios
    - Cumpliendo sus SLO.
    - Minimizando el coste para el proveedor.
      - Ordenadores.
      - Discos.
  - Es un problema de optimización.

3. Planificación, SLO y multiarrendamiento

- Lang et al. [LSP+12] propone estas etapas para resolver el problema:
  1. Evaluar el rendimiento de cada tipo de nodo servidor ("*Stock Keeping Unit*", SKU) con una carga homogénea de arrendatarios.
  2. Ídem con carga heterogénea.
     - Proporciona un modelo de caracterización de rendimiento por SKU.
  3. Obtener una estrategia óptima de despliegue de la carga. Proporciona:
     - Política de provisión de HW.
     - Política de planificación de arrendatarios.

3. Planificación, SLO y multiarrendamiento

- La tercera etapa utiliza como entradas:
  - Un conjunto de clases de SKU con...
    - Coste específico de cada clase.
    - Características de rendimiento.
  - Un conjunto de arrendatarios, con los requisitos (SLO) de cada uno.

3. Planificación, SLO y multiarrendamiento

- Lang et al. utilizan estas etapas en un sistema con estos SKU:

TWO SERVER CONFIGURATIONS (SKUS)

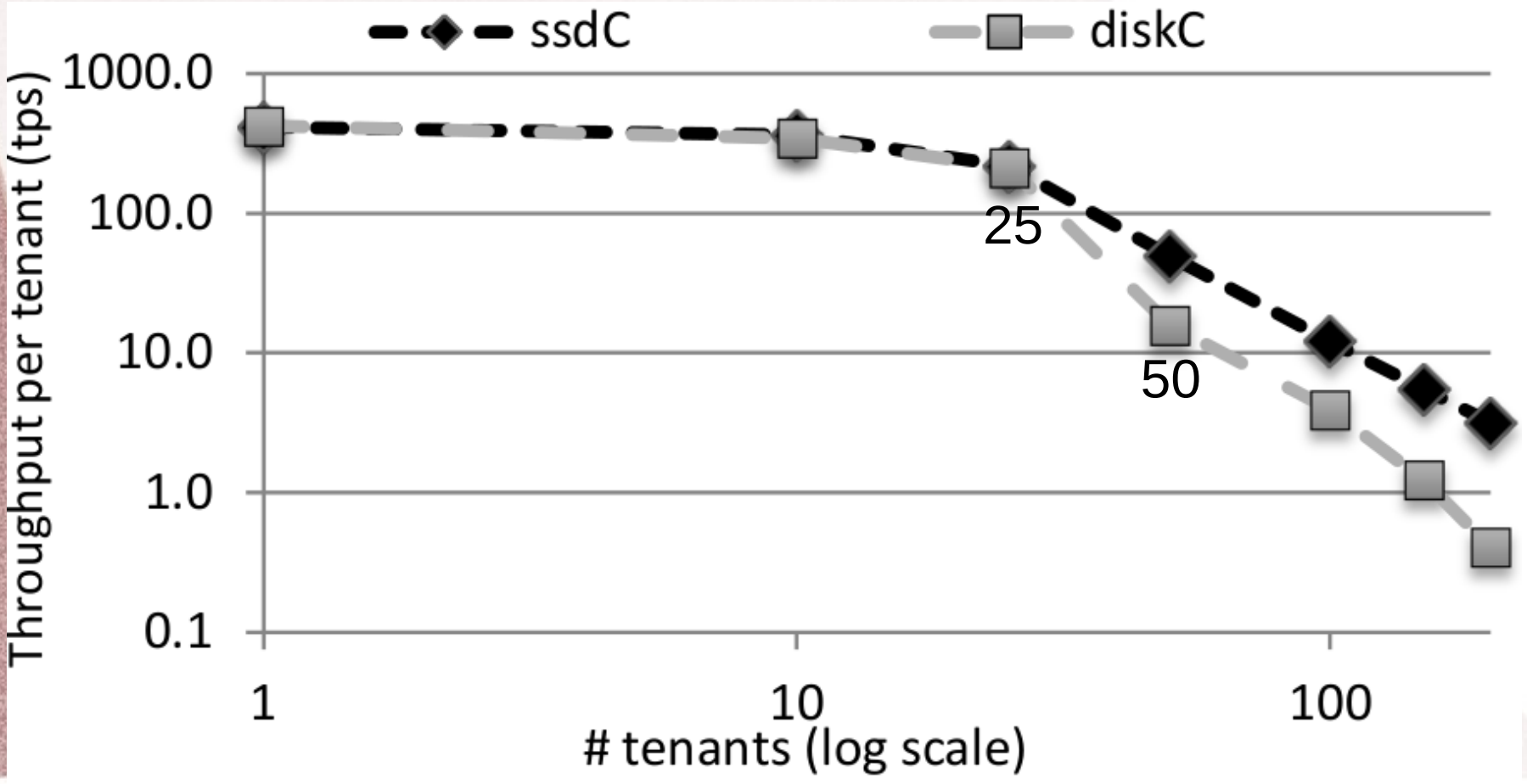
	ssdC	diskC
CPU	2X Intel L5630	2X Intel L5630
RAM	32GB	32GB
OS Storage	10K SAS 300GB	10K SAS 300GB
DB Data	2X Crucial C300 256GB	2X 10K SAS 300GB
DB Log	Crucial C300 256GB	10K SAS 300GB
RAID Cntlr w/BBC	YES	YES
Cost	$4,500	$4,000

3. Planificación, SLO y multiarrendamiento

- Otras consideraciones:
  - BD relacional (SQL Server).
  - Arrendatarios introducen una carga similar a la utilizada en el benchmark TPC-C.
    - BD de 1 GB por arrendatario.
  - Modelo de multiarrendamiento:
    - BD separadas (aunque sobre un SGBD compartido).

3.1. Primera etapa

- Evaluar el rendimiento de cada SKU ante cargas homogéneas.
  - Asume un único tipo de arrendatario.
    - En cuanto a la carga introducida.



3.2. Segunda etapa

- Evaluar el rendimiento de cada tipo de SKU ante cargas heterogéneas.
  - Se asume que habrá varias clases de arrendatarios.
    - En función de su carga introducida o de sus SLO.
  - Definición 1: Sea $S=\{s_1,s_2,...,s_k\}$ el conjunto de k SLO publicados por un proveedor DaaS.
    - Por tanto, k clases de arrendatarios.

3.2. Segunda etapa

- Definición 2:
  - Para un SKU dado, sea B = [b_1, b_2,..., b_k], donde b_i representa el número de arrendatarios de la clase s_i asignados al servidor.
  - Para este servidor, la *función de caracterización del rendimiento del SKU*, f(B), representa el rendimiento obtenido para un intervalo de tiempo determinado con diferentes políticas de planificación / reparto de arrendatarios.

3.2. Segunda etapa

- Definición 3:
  - Dado un servidor SKU y un vector B (según la Def. 2), una *función booleana simplificada de caracterización de rendimiento* del SKU, *f'(B)*, devuelve cierto si todos los arrendatarios alcanzan sus respectivos SLO de rendimiento y falso en otro caso.

3.2. Segunda etapa

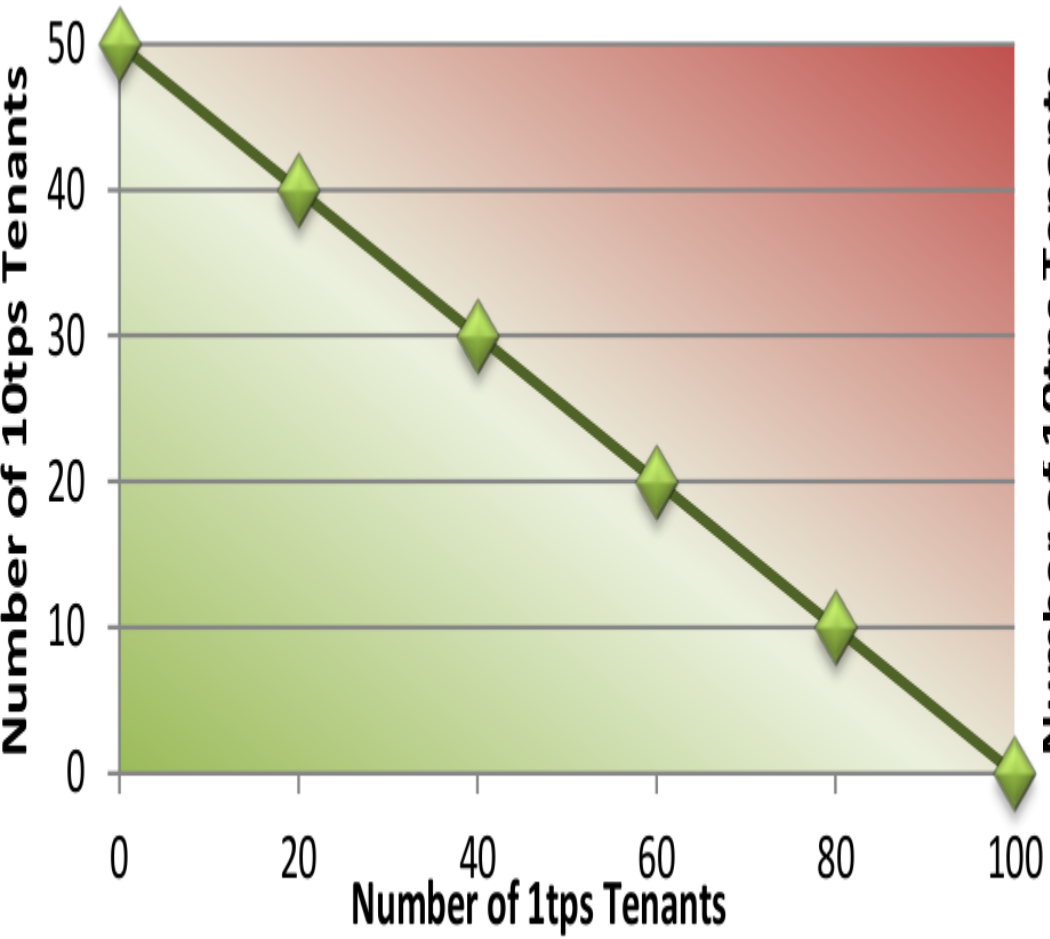
- Otras consideraciones:
  - Decidir cómo repartir recursos entre diferentes clases de arrendatarios que comparten un SKU.
    - Cada clase recibe una instancia SQL Server en el SKU.
    - Memoria asignada a este servidor proporcional a dos factores:
      - Proporción de TPS en esta clase frente al agregado de todas las clases.
      - Proporción de arrendatarios en esta clase frente al agregado de todas las clases.

3.2. Segunda etapa

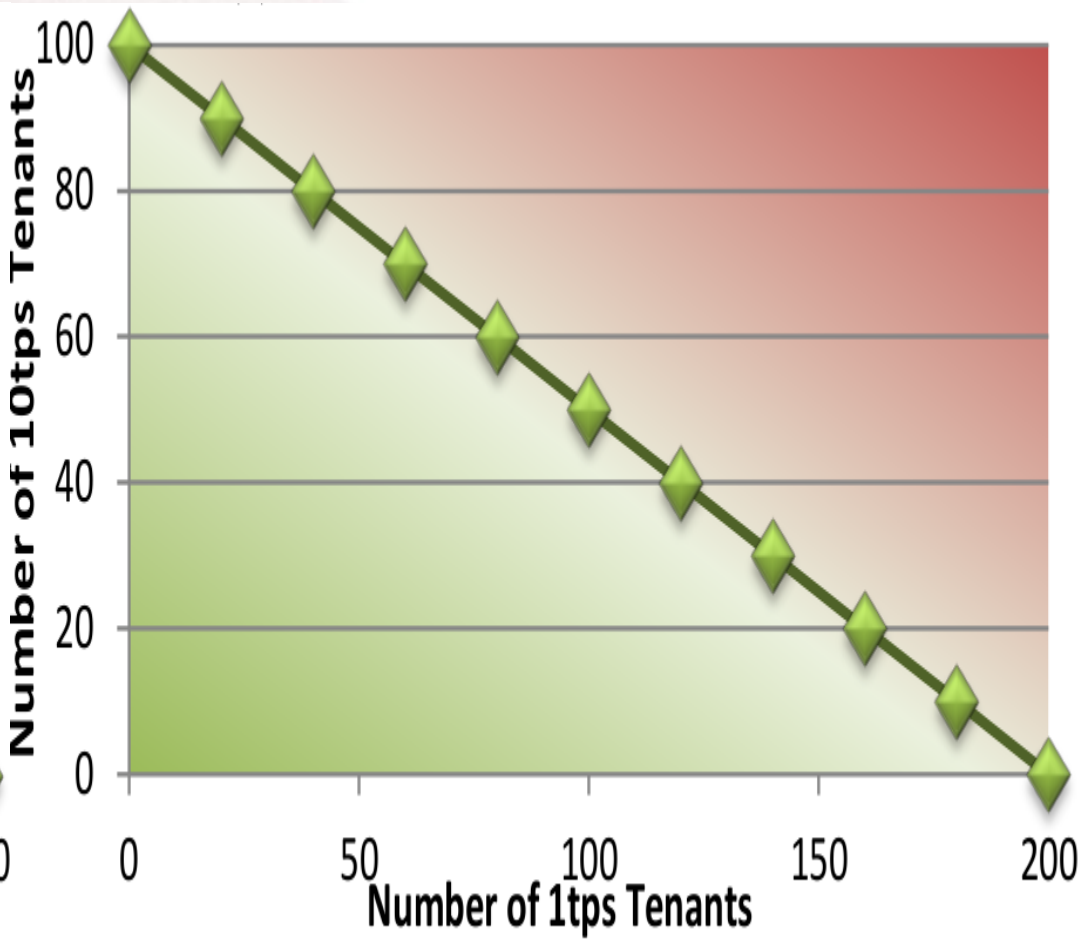
- Según los resultados de la etapa 1:
  - Ambos SKU admiten unos 25 o 26 arrendatarios a 100 TPS.
    - Cuando llenan la memoria RAM.
  - Para un rendimiento de 10 TPS:
    - Aprox. 50 arrend. en diskC SKU.
    - Aprox. 100 arrend. en ssdC SKU.
  - Para un rendimiento de 1 TPS:
    - Aprox. 200 arrend. en diskC SKU.
    - > 500 arrend. en ssdC SKU.

3.2. Segunda etapa

- Resultados para S={10tps,1tps}.



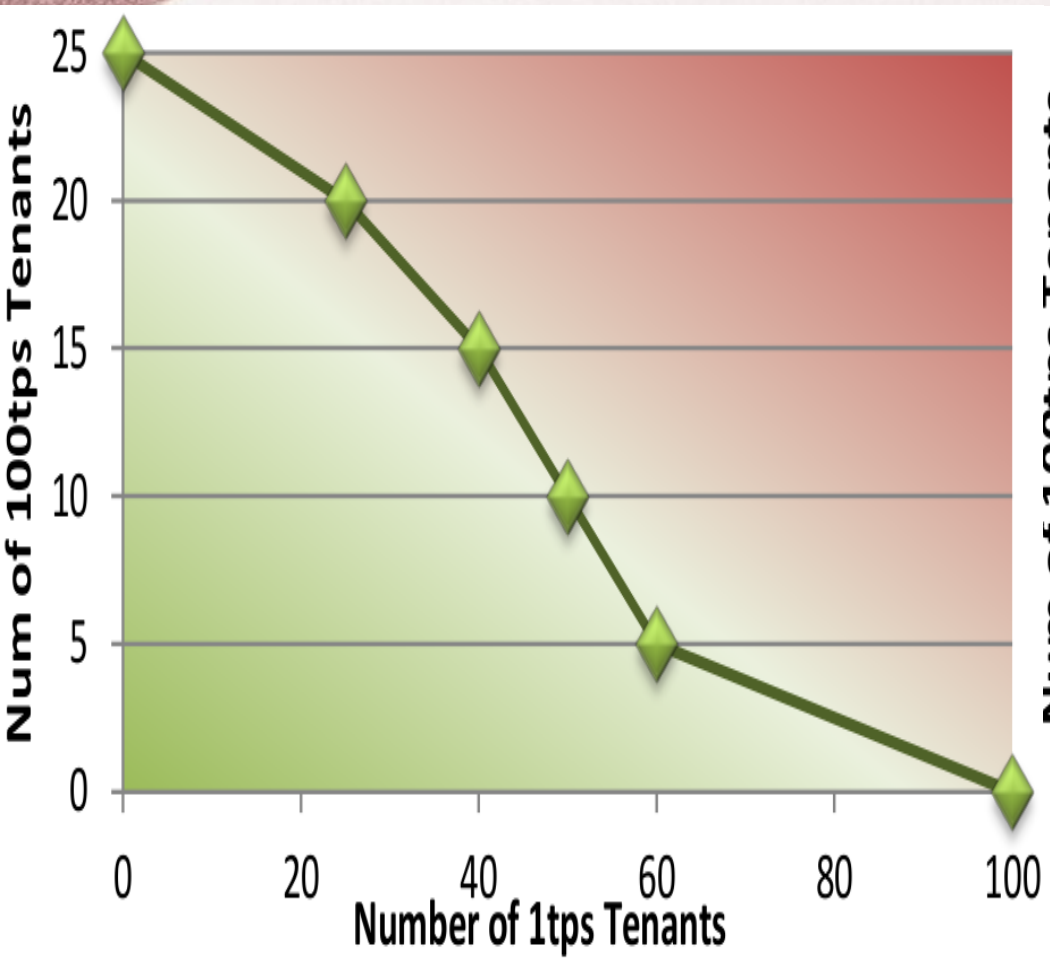
(a) Performance on the diskC SKU



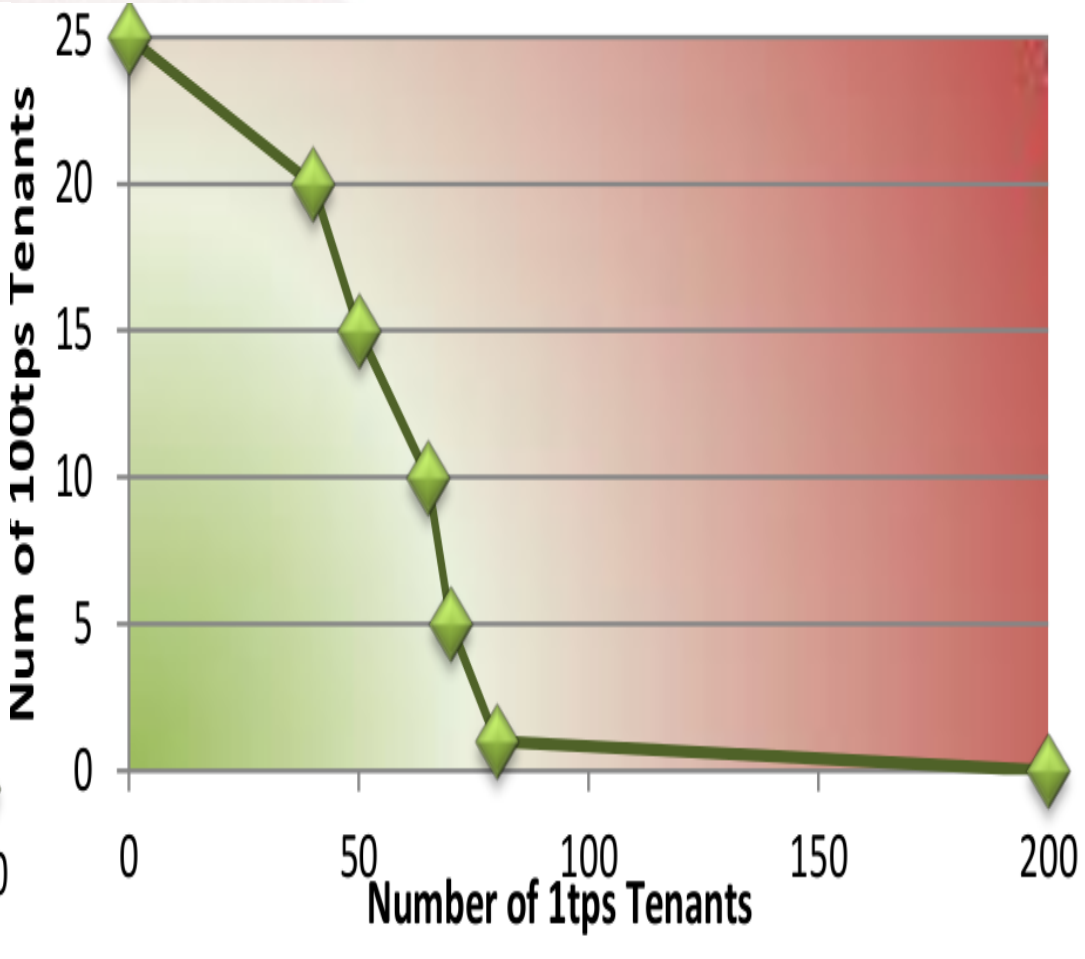
(b) Performance on the ssdC SKU

3.2. Segunda etapa

- Resultados para S={100tps,1tps}.



(a) Performance on the diskC SKU

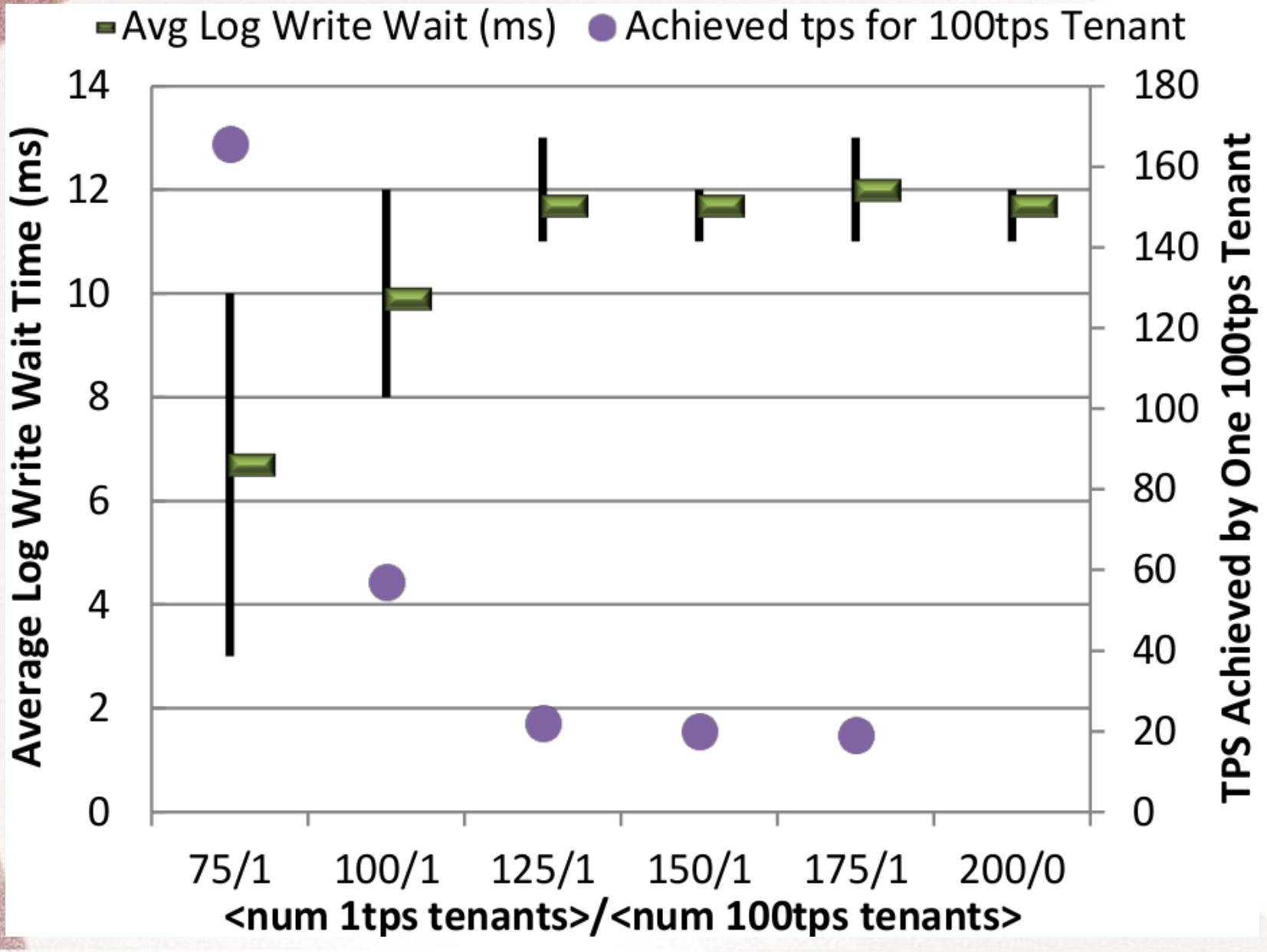


(b) Performance on the ssdC SKU

3.2. Segunda etapa

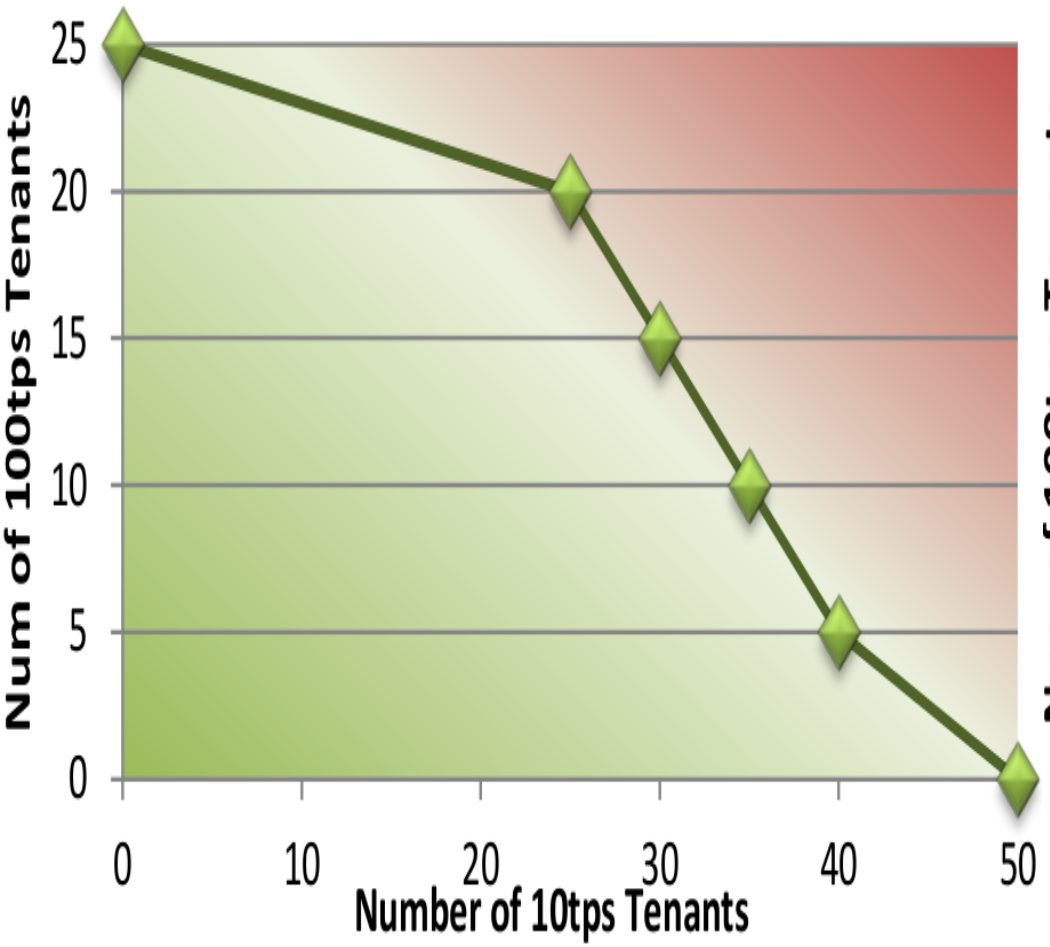
- La configuración S={100tps,1tps} ya no ofrece una frontera lineal.
- ¿Por qué?
  - Al introducir un número alto de arrendatarios a 1 TPS, el disco de logging se satura.
    - Supera los 10ms por escritura.
  - Estos valores dependen de la arquitectura del SGBD.
- Ver figura siguiente, en un ssdC SKU.

3.2. Segunda etapa

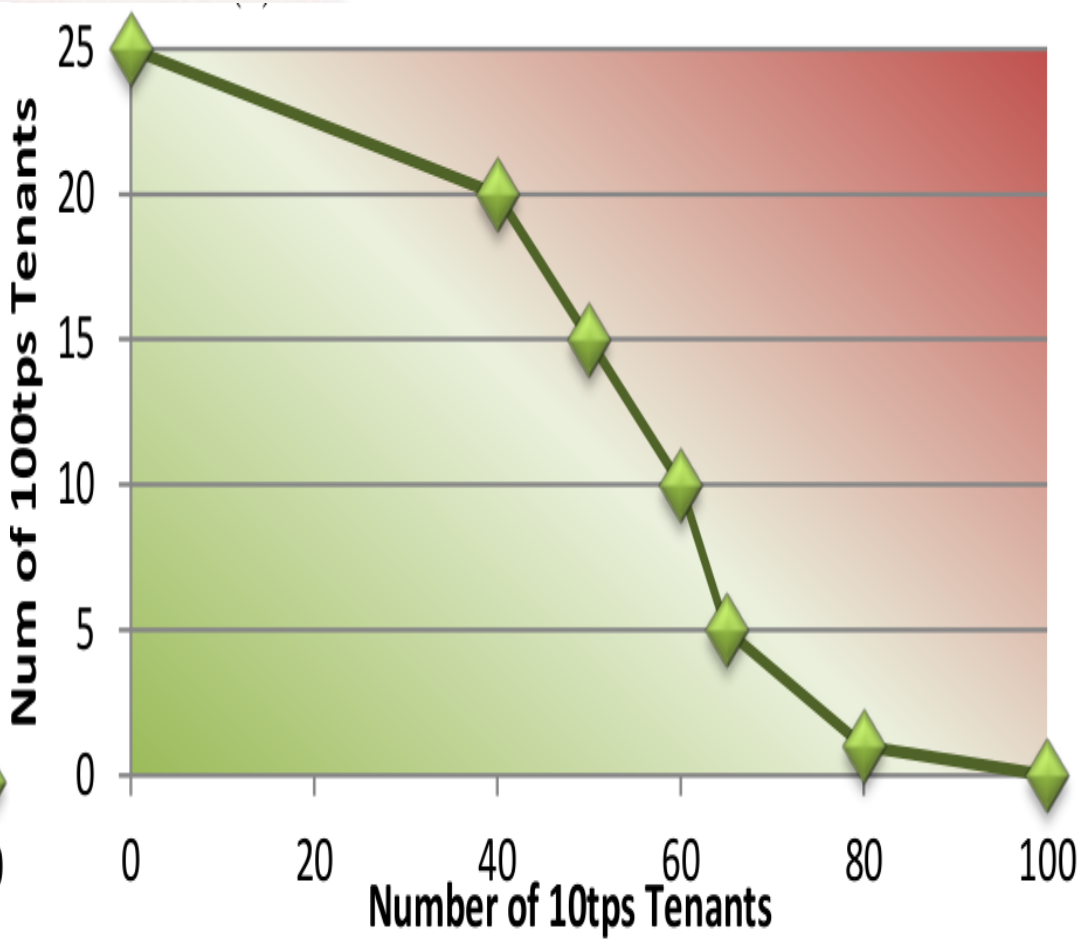


3.2. Segunda etapa

- Resultados para S={100tps,10tps}.



(a) Performance on the diskC SKU



(b) Performance on the ssdC SKU

3.3. Tercera etapa

- Se necesitan definiciones previas:
  - Definición 4: M es un multiconjunto $\{m_1,m_2,...,m_p\}$ donde cada m_j representa a un servidor SKU definido por el par (f'_j,c_j) donde la función f'_j es la dada en la Def. 3 y c_j representa el coste operativo amortizado mensual del servidor.
  - Definición 5: Sea t_i un conjunto de arrendatarios que suscribe el SLO s_i especificado en la Def. 1. Representaremos a todos los arrendatarios mediante el conjunto $T=\{t_1,t_2,...,t_k\}$.

3.3. Tercera etapa

- Con las def. 1 a 5 se especifica el problema de minimización:
  - Dados los conjuntos S, T y el multiconjunto M...
    - Calcular los vectores
      - $A = [a_1, a_2, ..., a_p]$
        - a_i es el número de servidores de tipo m_i
      - $B = [b_1, b_2, ..., b_p]^T$
        - b_i es un vector de longitud k que indica cuántos arrendatarios de cada clase de SLO serán asignados a cada servidor de tipo m_i.
    - Para que la función objetivo $C=a_1 c_1 + a_2 c_2 + ... + a_p c_p$ obtenga un valor mínimo y satisfaga las restricciones:
      - R1: $AB = [|t_1|, |t_2|, ..., |t_k|]$ (se cubran todos los arrendatarios)
      - R2: $f'_i(b_i)$ devuelva cierto para todo i, $1<i<p$ (se cumplan todos los SLO)

3.3. Tercera etapa

CASO DE ESTUDIO

- Se asume una vida útil para los servidores de 3 años.
  - Su coste amortizado mensual sería:
    - SsdC: 125 $
    - DiskC: 111 $
      - Se utiliza también una configuración con el mismo rendimiento, pero más barata (3150$) del SKU diskC.
        - Coste mensual: 88$
- Se asume un total de 10000 arrendatarios.

3.3. Tercera etapa

- Configuraciones a evaluar:

Scenario	SLO set	Tenant Ratio	diskC Amortized Cost
SC1	S_2={100tps, 1tps}	20:80	$111
SC2	S_2={100tps, 1tps}	50:50	$111
SC3	S_2={100tps, 1tps}	80:20	$111
SC4	S_2={100tps, 1tps}	20:80	$88
SC5	S_2={100tps, 1tps}	50:50	$88
SC6	S_2={100tps, 1tps}	80:20	$88
SC7	S_3={100tps, 10tps}	20:80	$111
SC8	S_3={100tps, 10tps}	50:50	$111
SC9	S_3={100tps, 10tps}	80:20	$111
SC10	S_3={100tps, 10tps}	20:80	$88
SC11	S_3={100tps, 10tps}	50:50	$88
SC12	S_3={100tps, 10tps}	80:20	$88

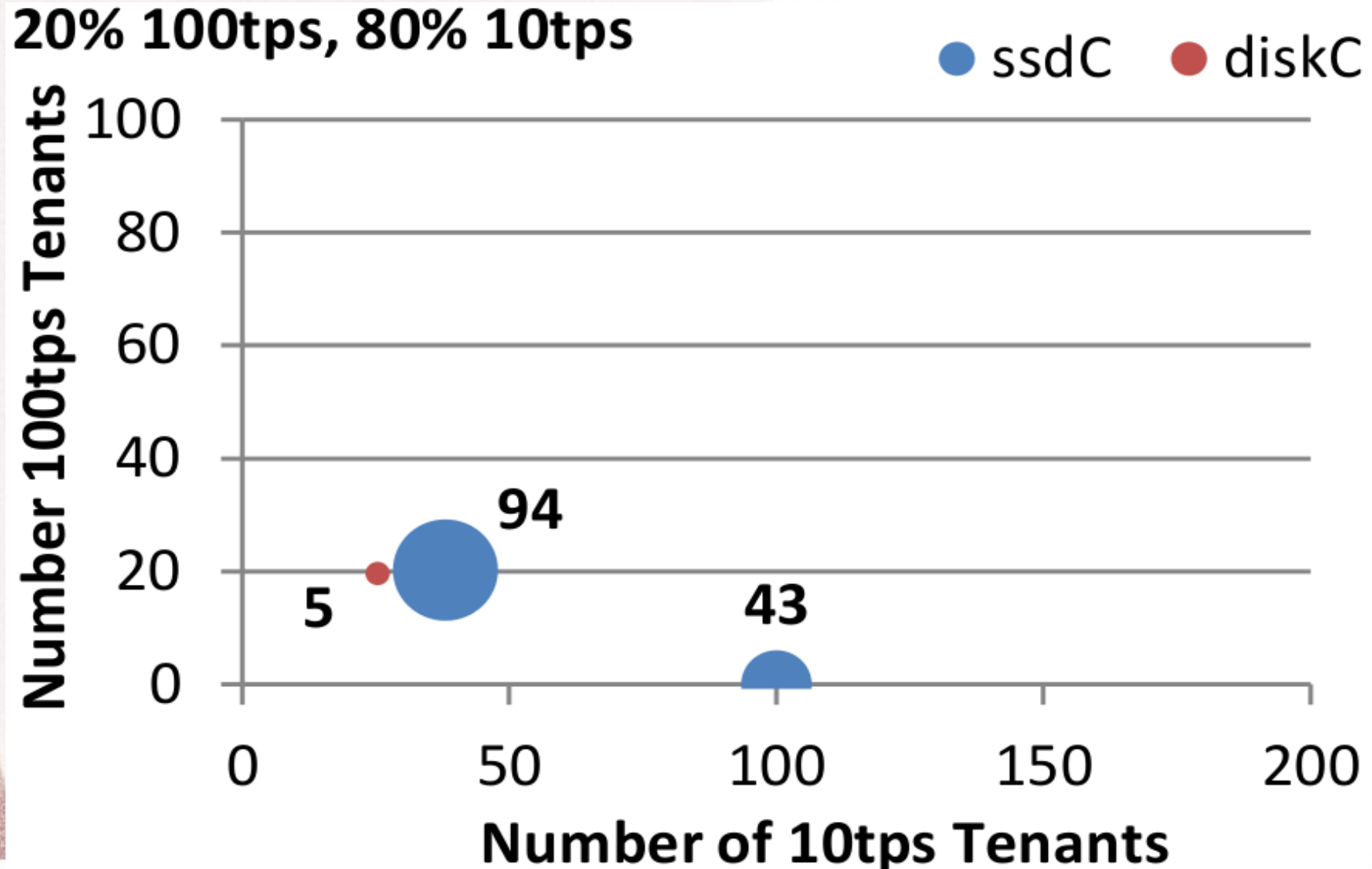
3.3. Tercera etapa

Resultados para algunas configuraciones

Config.	#ssdC	#diskC	Ar1/ssd	Ar2/ssd	Ar1/dis	Ar2/dis	Coste
SC1	38	82	0	200	24,39	4,878	13852$
SC2	19	206	0	200	24,272	5,825	25241$
SC3	0	330	0	0	24,242	6,061	36630$
SC4	35	86	0	200	23,256	11,628	11943$
SC5	15	211	0	200	23,697	9,478	20443$
SC6	0	330	0	0	24,242	6,061	29040$
SC8	179	37	23,385	27,315	22	3	26482$
SC9	67	260	24,47	29,852	24,47	0	27235$
SC11	0	240	0	200	20,833	20,833	21120$
SC12	0	336	0	0	23,810	5,952	29568$

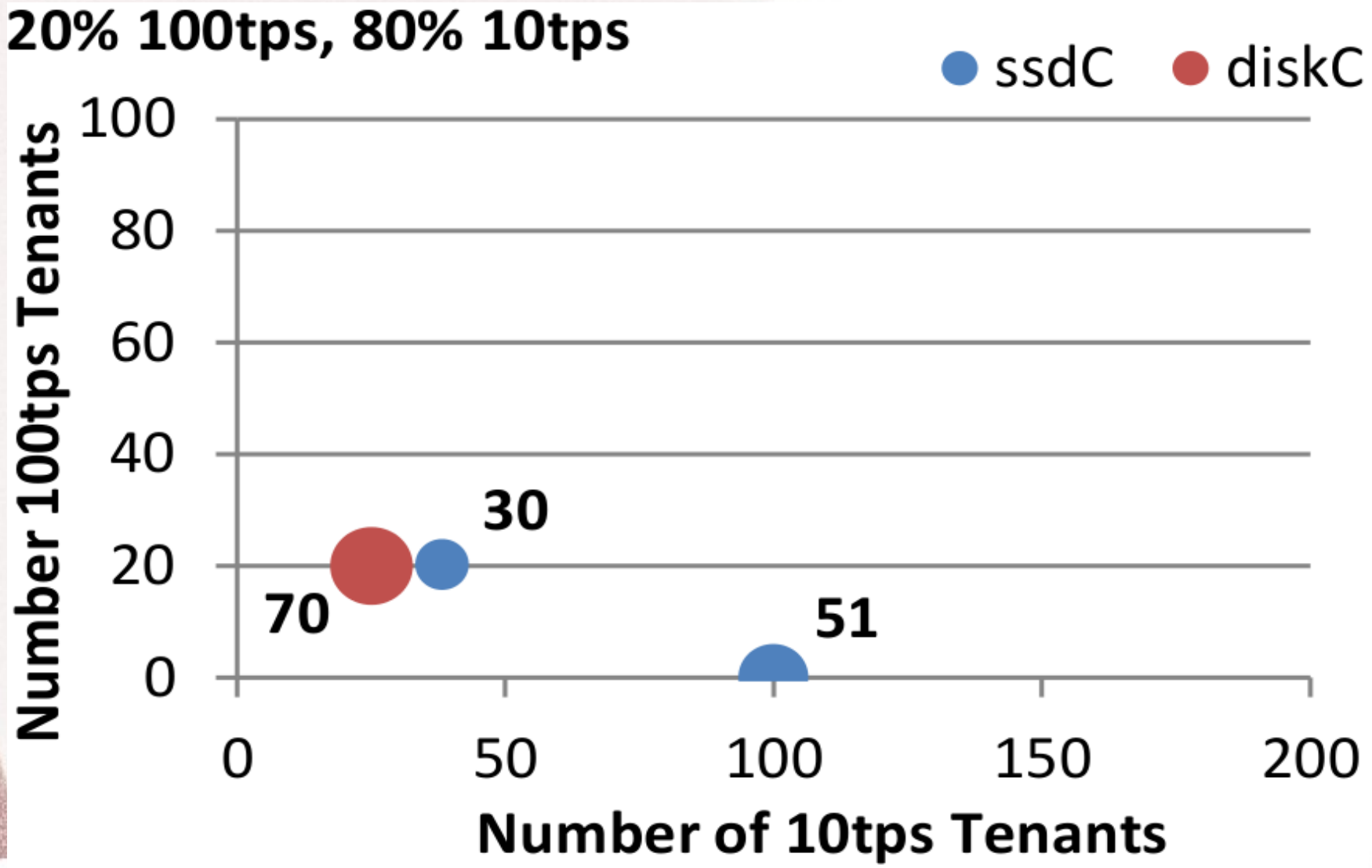
3.3. Tercera etapa

- Configuración SC7: Coste 17681$.



3.3. Tercera etapa

- Configuración SC10: Coste 16250$.



3.4. Revisión

- Si analizamos los resultados...
  - Las configuraciones seleccionadas...
    - ¡Asignan los SKU con mejores prestaciones a las configuraciones menos exigentes!!
    - No siempre utilizan ambos tipos de servidores SKU.
  - Hay otros costes no considerados...
    - Administración y monitorización de equipos.
    - Refrigeración.
    - Mantenimiento del centro de datos.
    - ¡Quizá interesen otras configuraciones más sencillas! Por ejemplo, homogéneas.
      - Que reduzcan esos costes no contemplados en el modelo.

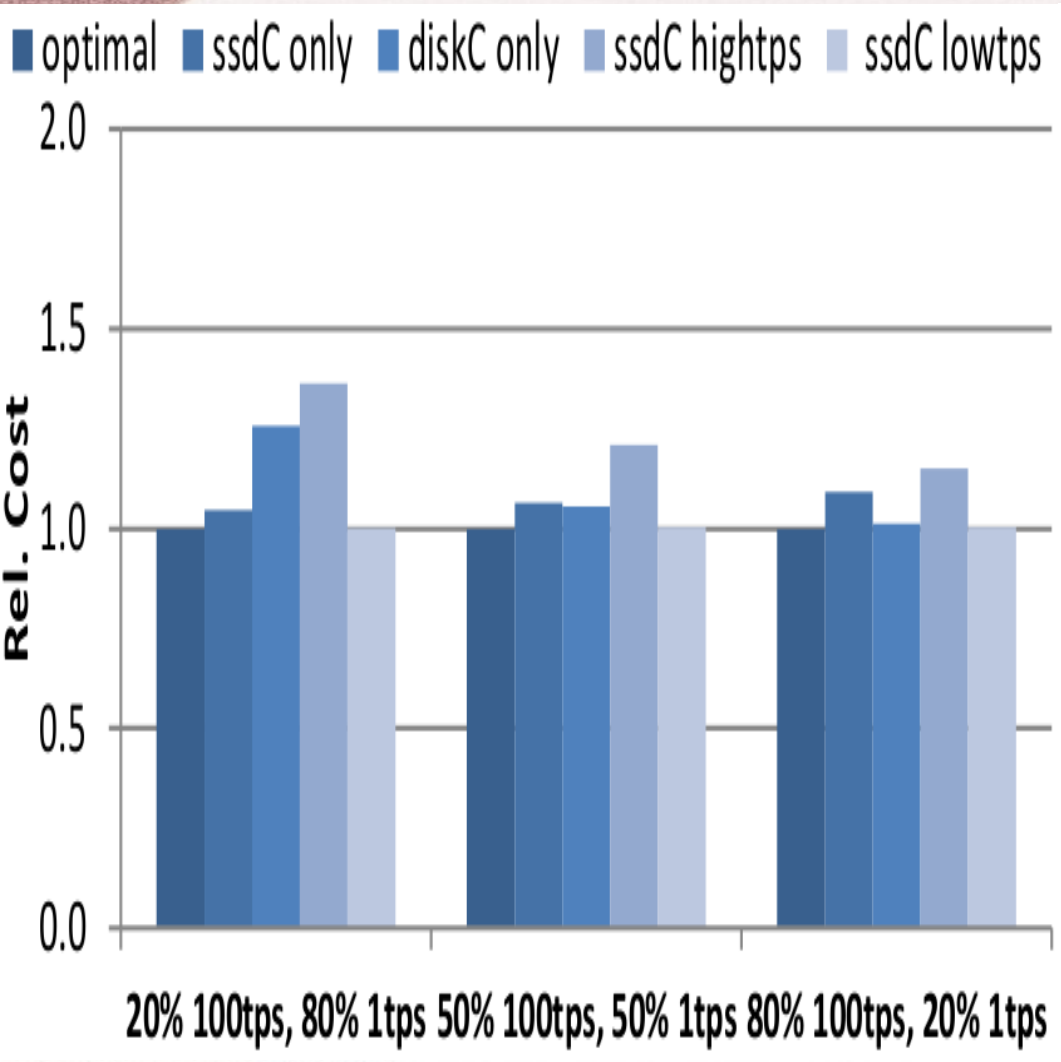
3.4. Revisión

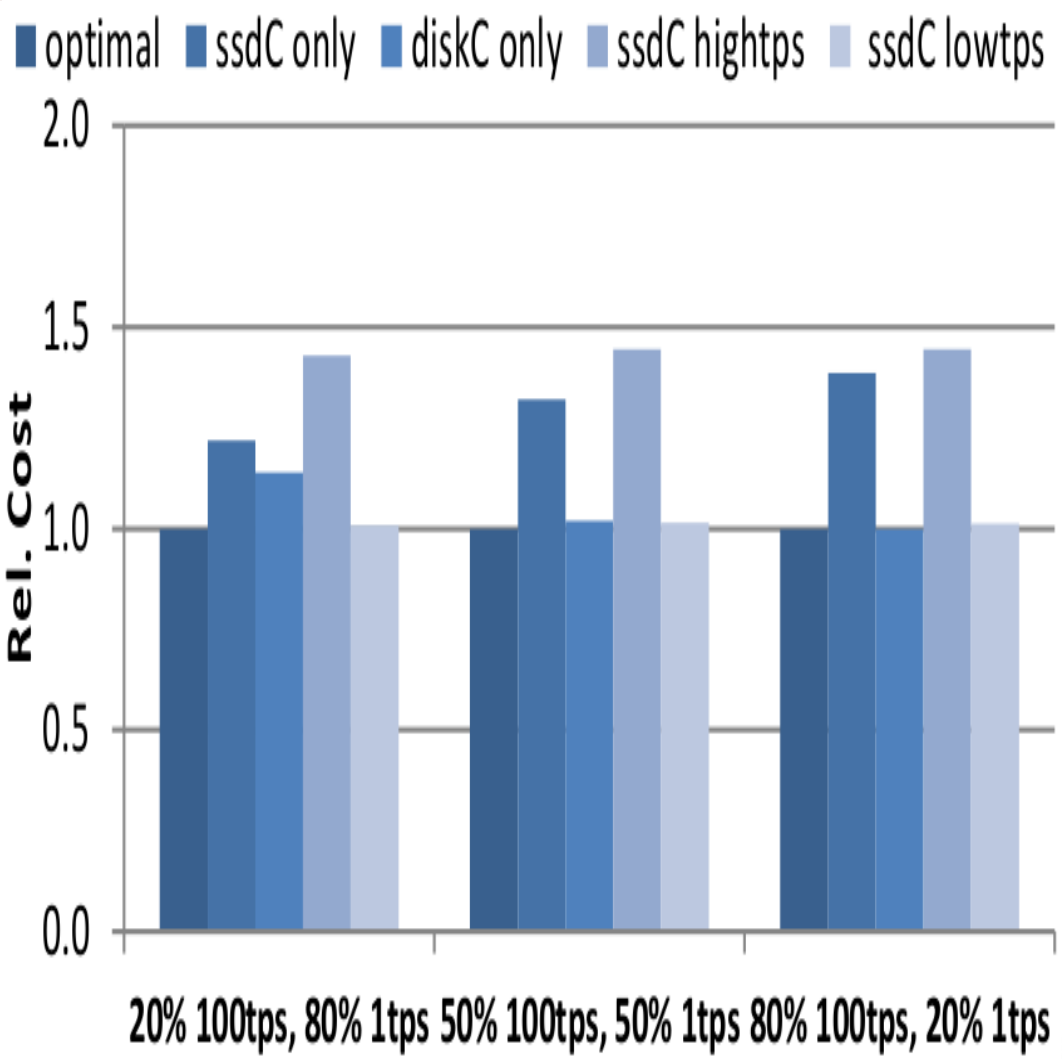
- Serían configuraciones subóptimas.
  - En cuanto a costes.
  - Pero más sencillas de administrar.
- Lang et al. estudian estos casos:

Methods	ssdC SKU	diskC SKU
Optimal	heterogeneous SLOs	heterogeneous SLOs
ssdC-only	heterogeneous SLOs	–
diskC-only	–	heterogeneous SLOs
ssdC-hightps	homogeneous high-perf	homogeneous low-perf
ssdC-lowtps	homogeneous low-perf	homogeneous high-perf

3.4. Revisión

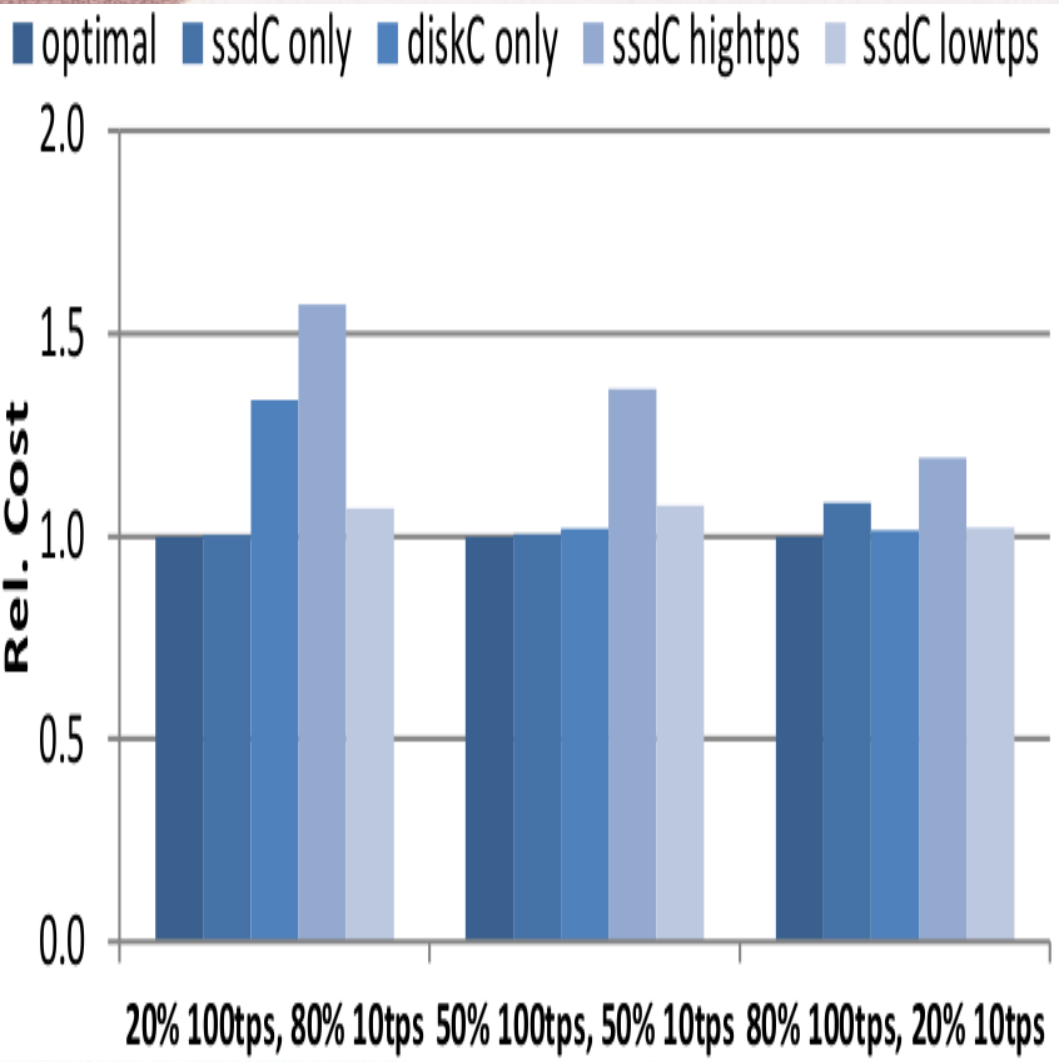
- Resultados para escenarios SC1 a SC6.

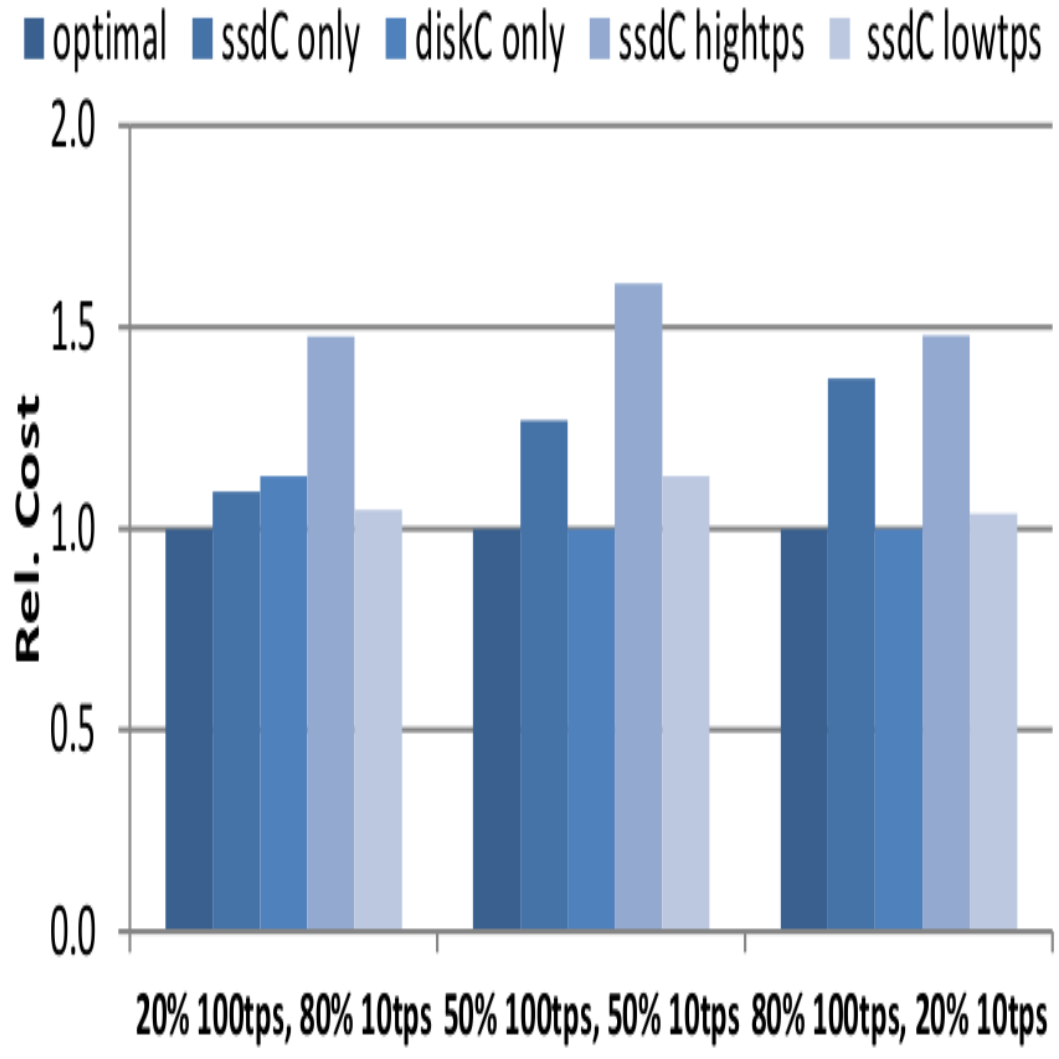




3.4. Revisión

- Resultados para escenarios SC7 a SC12.





3.4. Resultados

- Los costes relativos son más altos que en el caso óptimo.
  - Las peores configuraciones se dan en los escenarios SC7 y SC11.
    - Superan el 52% y 59% de sobrecoste, respectivamente.
  - Pero en otros escenarios llegamos a encontrar hasta dos configuraciones más sencillas con costes idénticos al caso óptimo.
    - ¡Conviene utilizarlas!!!

Índice

4. Consideraciones energéticas

- Cuando la carga no sea alta...
  - Interesará parar máquinas (físicas)
    - Sin incumplir SLA
    - Minimizando las migraciones
    - Sin penalizar el rendimiento
    - Con ello se minimiza el consumo energético y se minimiza el coste económico.
- Redistribución de máquinas virtuales
  - Aproximación básica:
    - Utilizar un algoritmo de mejor ajuste ("*best fit*") [JDU+74] para dirigir las migraciones.
    - Ejemplos / evoluciones: [LHM+13]

4. Consideraciones energéticas

- Principios a utilizar:
  - Objetivo inverso al equilibrado de carga.
    - Se busca concentrar la carga en unos pocos nodos.
    - Liberando al resto, que podrá ser parado.
  - Los nodos que queden activos deben ser capaces de atender la carga que reciban.
    - En algunos casos se tolerará cierta sobrecarga inicial.
    - Implicación en la gestión del SLA.

4. Consideraciones energéticas

- Principios (ii):
  - Cerca del 60% del consumo de electricidad en un ordenador se debe a su fuente de alimentación [CAT+01].
    - Se ahorra poco dejando el equipo en suspensión.
    - Si un nodo no se va a utilizar, conviene apagarlo.

4. Consideraciones energéticas

- Migración imposible en caso de datos persistentes:
  - Requiere excesivo tiempo.
  - Pero los datos suelen estar replicados
    - Si el grado de replicación es suficientemente alto...
      - Bastaría con decidir qué réplicas parar.
      - Redistribuyendo su carga entre las demás.

4. Consideraciones energéticas

- Propuestas relevantes para gestionar los paros:
  - Chase et al. [CAT+01]
    - Una de las primeras publicaciones en esta área.
    - Solución general.
  - Lang et al. [LPN09]
    - Solución específica para datos.
    - Consultar artículo.

4.1. Chase et al.

- La gestión de energía es otro factor a contemplar en la optimización / asignación de recursos.
  - Ver sección 3.
- Términos utilizados en [CAT+01]:
  - μ_i/μ_{max}: Unidades de recurso asignadas al servicio i / totales en el sistema.
  - t: Época / instante. Intervalo temporal en el que tanto μ_{max} como el número de servicios desplegados y la carga soportada permanezcan constantes.
  - $U_i(t,\mu_i)$: "Utilidad" → Beneficio obtenido por el proveedor al asignar μ_i recursos al servicio i.
  - Recurso: Ordenador / servidor.

4.1. Chase et al.

- Términos (ii):
  - $\lambda_i(t,\mu_i)$: Rendimiento obtenido por el servicio i durante el instante t con su asignación de recursos.
  - $bid_i(\lambda_i(t,\mu_i))$: Ingresos del proveedor al generar cierto rendimiento para un servicio i.
  - r_i: Recursos contratados por el servicio i.
    - Cargas altas: $\mu_i > r_i$.
    - Cargas bajas: $r_i > \mu_i$.

4.1. Chase et al.

- Términos (iii):
  - ρ_i: Utilización de los recursos asignados al servicio i.
    - Depende de la carga soportada.
  - ρ_{target}: Utilización idónea según el SLA del servicio i.
  - cost(t): Coste variable (para el proveedor) medio por unidad de recurso utilizada en el instante t. Incluye:
    - Consumo eléctrico.
    - Mantenimiento.
  - $penalty_i(t,\mu_i)$: Penalización económica para el proveedor si en el instante t se da:
    - $\mu_i < r_i \quad Y \quad \rho_i > \rho_{target}$

4.1. Chase et al.

- Objetivo: Maximizar los beneficios...

$$profit(t) = \sum_i^N (U_i(t, \mu_i) - \mu_i * cost(t))$$

$$U_i(t, \mu_i) = bid_i(\lambda_i(t, \mu_i)) - penalty_i(t, \mu_i)$$

- ...respetando la restricción...

$$\mu = \sum_i^N \mu_i \leq \mu_{max}$$

4.1. Chase et al.

- Términos (iv):
  - $price_i(\mu_i) = U_i(t,\mu_i+1) - U_i(t,\mu_i)$
  - grow(i,target)
    - Asigna recursos adicionales al servicio i hasta que su precio sea "*target*".
    - Así $price_i(\mu_i)$ baja.
  - shrink(i,target)
    - Requisa recursos del servicio i hasta que su precio sea "*target*".
    - Así $price_i(\mu_i)$ sube.

4.1. Chase et al.

- Algoritmo:

1. *Reclaim resources with negative return.* For any service i with $price_i(\mu_i) \leq cost(t)$: $shrink(i, cost(t))$.

2. *Allot more resources to profitable services.* If $\mu < \mu_{max}$, then there are idle resources to deploy. Determine the highest bidder i with maximum $price_i(\mu_i)$ and the next highest bidder j. Greedily allot resources to the highest bidder i: $grow(i, price_j(\mu_j))$. Repeat, adding resources until $\mu = \mu_{max}$, or the highest remaining bidder's $price_i(\mu_i) < cost(t)$.

3. *Reclaim overcommitted resources.* If $\mu > \mu_{max}$, then reclaim the least profitable resources. This may be needed if browndown or some other failure has reduced μ_{max} from the previous epoch. Determine the lowest-loss service i with minimal $price_i(\mu_i)$, and the next lowest-loss service j. Reclaim from i: $shrink(i, price_j(\mu_j))$. Iterate with the next lowest-loss service, reclaiming resources until $\mu = \mu_{max}$.

4. *Adjust for highest and best use.* If any services i and j exist such that $price_i(\mu_i) < price_j(\mu_j)$, then shift resources from i to j until $price_i(\mu_i) \approx price_j(\mu_j)$. This is done by shifting resources from the lowest to the highest bidders until equilibrium is achieved.

4.1. Chase et al.

- Resultados:
  - El algoritmo termina cuando los “precios” de todos los servicios llegan a un equilibrio.
    - La utilización de cada recurso depende de:
      - El servicio al que haya sido asignado.
      - El SLA de ese servicio.
      - El beneficio que obtenga el proveedor de ese servicio.
  - Si la carga global es baja:
    - Habrá instancias de recurso no asignadas.
    - Se ahorrará energía.

4.1. Chase et al.

- Resultados (ii):
  - Mejor control del consumo energético.
  - Tiempo de respuesta más uniforme.

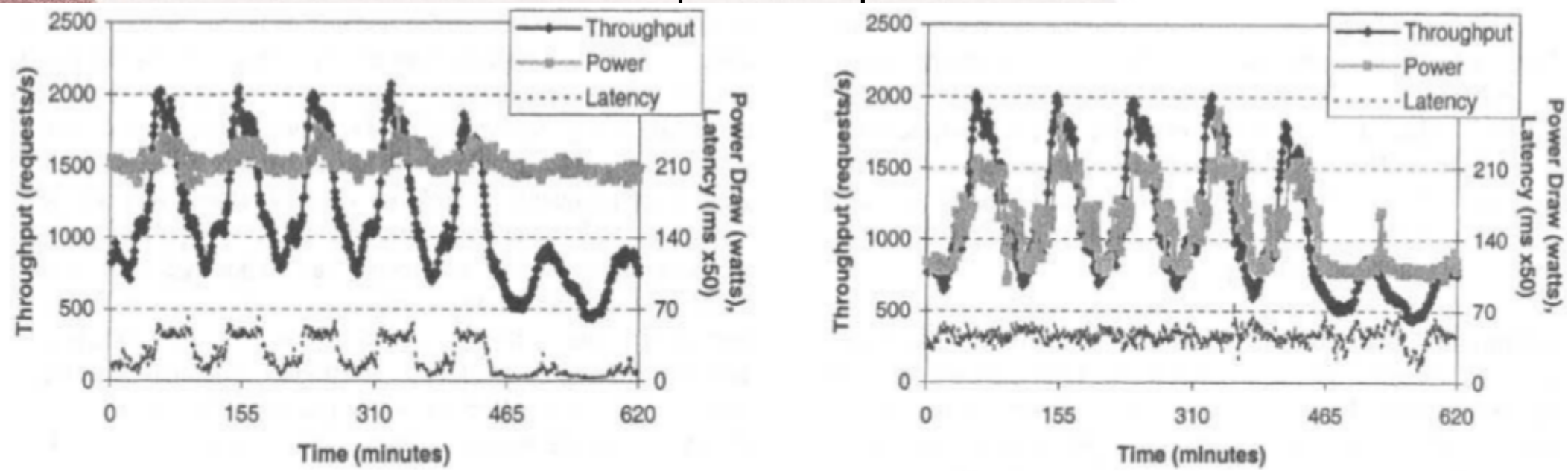


Figure 11: Throughput, power, and latency for the IBM trace at 16x with (right) and without (left) energy-conscious provisioning.

Índice

5. Resultados de aprendizaje

- Al finalizar esta unidad, el alumno debe ser capaz de:
  - Conocer las variantes de multiarrendamiento existentes.
  - Gestionar adecuadamente el multiarrendamiento en la provisión de recursos.
  - Evaluar la incidencia del multiarrendamiento sobre el rendimiento global y los SLA.
  - Aplicar diferentes estrategias de planificación y distribución de carga para minimizar el consumo energético en una infraestructura escalable.