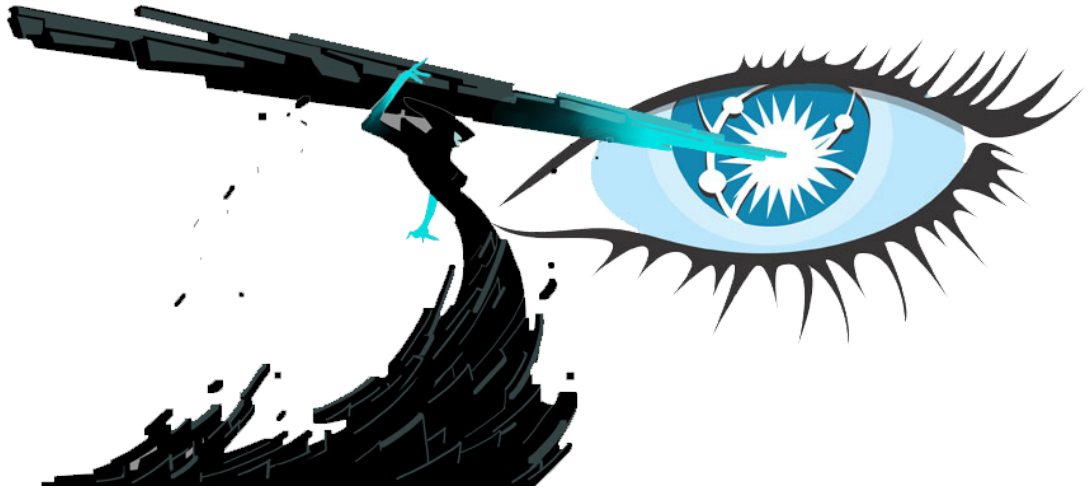
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

Lissandra

Documento de pruebas

Cubriéndonos de aca a invierno



Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 1C2019 -
Versión 1.0

Requisitos y notas de la evaluación

Los requisitos expuestos a continuación se encuentran ampliados en las Normas del Trabajo Práctico, que por practicidad, se han resumido a continuación.

Deploy y Setup

Es condición necesaria para la evaluación que **el Deploy y Setup del trabajo se realice en menos de 10 minutos.** Pasado este tiempo el grupo perderá el derecho a la evaluación.
Los archivos de configuración requeridos **para los diversos escenarios de pruebas** deberán ser preparados con anticipación por el grupo con todos los valores requeridos prefijados dejando los sólo los parámetros desconocidos (ej: IP) incompletos.

Compilación y ejecución

La compilación debe hacerse en la máquina virtual de la cátedra en su edición Server (no se pueden usar binarios subidos al repositorio).
Será responsabilidad del grupo verificar las dependencias requeridas para la compilación, y en caso de requerir bibliotecas provistas por la cátedra, descargarlas. También es responsabilidad de los integrantes del grupo conocer y manejar las herramientas de compilación desde la línea de comandos. Ver Anexo - Comandos Útiles

Evaluación

Cada grupo deberá llevar **una** copia impresa de la planilla de evaluación[1] **con los datos de los integrantes completos** (dejando los campos “Nota” y “Coloquio” en blanco) y una copia de los presentes tests.

Para la aprobación, un Trabajo Práctico deberá contar con todos los ítems marcados como “**Contenidos Mínimos**”.

Una vez alcanzada la aprobación, los ítems marcados como **“Detalle”** tienen un valor asociado que suman puntaje en caso de ser cumplidos, teniendo como base la nota 6 (seis) y como máximo la nota 10 (diez).
Las pruebas **pueden ser alteradas o modificadas entre instancias de entrega** para verificar el correcto funcionamiento y desempeño del sistema desarrollado.
En estos casos el documento será actualizado y re-publicado para reflejar estos cambios.

[1] Al final de este documento

Anexo - Comandos Útiles

Copiar un directorio completo por red

```
scp -rpC [directorio] [ip]:[directorio]
```

Ejemplo:

```
scp -rpC tp-1c2015-repo 192.168.3.129:/home/utnso
```

Descargar bibliotecas en un repositorio (como las commons)

git clone [url_repo]

Ejemplo:

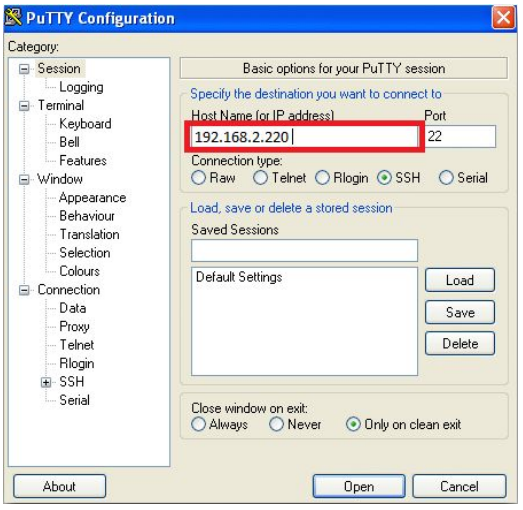
git clone https://github.com/sisoputnfrba/so-commons-library

PuTTY

Este famoso utilitario nos permite desde Windows acceder de manera simultánea a varias terminales de la Máquina Virtual, similar a abrir varias terminales en el entorno gráfico de Ubuntu.

Ya se encuentra en las computadoras del laboratorio y se puede descargar desde aquí

Al iniciar debemos ingresar la IP de nuestra máquina virtual en el campo **Host Name (or IP address)** y luego presionar el botón **Open** y loguearnos como **utnso**



Se recomienda investigar:

- Directorios y archivos: cd, ls, mv, rm, ln (creación de symlinks)
- Entorno: export, variable de entorno LD_LIBRARY_PATH
- Compilación: make, gcc, makefile
- Criptografía: md5sum
- Visor de procesos del sistema: htop

Generador de scripts

Les planteamos dos alternativas para que cada grupo pueda auto generar scripts para la prueba de su trabajo práctico.

1)Utilizando el siguiente comando se pueden imprimir por consola scripts autogenerados junto con los resultados, es decir, el valor final que debería tomar cada una de las claves.

```
curl
"https://utnso.com/lql-script?selectProbability=15&keyMaxLength=100&valueMaxLength=60&lines=10&tables=ANIMALS,PEOPLE,SINGERS"
```

2)Utilizando el siguiente comando se puede escribir dentro de un archivo el script lql generado y dentro de otro archivo los resultados esperados.

```
curl
"https://utnso.com/lql-script?selectProbability=15&keyMaxLength=100&valueMaxLength=60&lines=10&tables=ANIMALS,PEOPLE,SINGERS" | csplit --elide-empty-files - '/^$/' \"{*}\" && mv xx00 nombre_archivo_script.lql && mv xx01 nombre_archivo_resultados.txt
```

Parámetros a utilizar:

1) selectProbability: La probabilidad de que una instrucción generada sea un SELECT. Caso contrario, será un INSERT.
2) keyMaxLength: tamaño máximo de las Keys
3) valueMaxLength: tamaño máximo del Value
4) lines: cantidad de líneas que tendrá el script
5) tables: tablas que se quiere que tenga el script. El generador se encargará de asignarle a las claves valores semánticamente correctos para la temática[2] de la tabla que se eligió.

Aclaración: todos los parámetros tienen valores default por lo que es opcional completarlos.

[2] Ejemplos de temáticas: computers, tvs, companies, governments, careers, wood, presidents, books, animals, remotes, illumination, conspiracies, whiskies, hamburgers, food, dictatorships, positions, beverages, ads, views

Prueba Base

Para la evaluación del trabajo práctico se requiere ejecutar esta prueba completa sin interrupciones. En caso que esto no se cumpla o los resultados no sean los esperados no se procederá a continuar la evaluación del trabajo práctico.

Es posible que el ayudante decida alterar / generar scripts nuevos con cambios en las keys, manteniendo el espíritu de las pruebas.

Configuración del sistema:

VM1 IP: Proceso Kernel <htop> <consola limpia>	VM2 IP: Proceso Memoria 1 <htop> <consola limpia>
VM3 IP: Proceso LFS <htop> <consola limpia>	VM4 IP: Proceso Memoria 2 <htop> <consola limpia>

Archivos de Configuración

LFS

```
PUERTO_ESCUCHA=5003
PUNTO_MONTAJE="/home/utnso/lfs-base/"
RETARDO=0
TAMAÑO_VALUE=60
TIEMPO_DUMP=20000
```

Memoria 1

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA2]
PUERTO_SEEDS=[8001]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=1280
RETARDO_JOURNAL=60000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=1
```

Memoria 2

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=1280
```

```
RETARDO_JOURNAL=60000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=2
```

Kernel

```
IP_MEMORIA=IP_MEMORIA1
PUERTO_MEMORIA=8001
QUANTUM=3
MULTIPROCESAMIENTO=1
METADATA_REFRESH=15000
SLEEP_EJECUCION=100
```

Metadata FileSystem

```
BLOCK_SIZE=64
BLOCKS=4096
MAGIC_NUMBER=LISSANDRA
```

Actividades:

1. Ejecutar todos los módulos (iniciando las memorias por la Memoria 1) y esperar a que el sistema sea consistente, es decir, que el Kernel conozca ambas memorias.
2. Asociar la Memoria 1 al criterio SC y ejecutar el script `simple_sc.lql` y esperar a que finalice.
3. Asociar la Memoria 1 y Memoria 2 al criterio EC y ejecutar el script `simple_ec.lql` y esperar a que finalice.
4. Asociar la Memoria 1 y Memoria 2 al criterio SHC y ejecutar el script `simple_shc.lql` y esperar a que finalice.
5. Ejecutar el comando JOURNAL en el kernel y esperar al dump de la memtable.

Resultados Esperados:

1. Inicialmente el Kernel solo debe conocer a la Memoria 1 y luego mediante el proceso de gossiping debe conocer a la Memoria 2.
2. Todos los pedidos deben ser resueltos normalmente y los SELECT deben dar los valores insertados previamente.
3. Los pedidos deben ser ejecutados de manera alternada entre las dos memorias. Uno de los primeros dos SELECT debe fallar ya que el dato no se encuentra en disco y de los subsiguientes dos deben dar valores diferentes (cumpliendo la lógica de Eventual Consistency).
4. Para las tablas cuya consistencia es SHC, los pedidos deben ser distribuidos por medio de la Key y todos los pedidos sobre la misma Key deben ir a la misma memoria.
5. Verificar que se creen los archivos temporales cumpliendo:
    a. El archivo temporal de la tabla “COLORS” debe tener 2 registros.
    b. El archivo temporal de la tabla “MARCAS” debe tener 1 registro.
    c. El archivo temporal de la tabla “CELULARES” debe tener 2 registros.
6. Las Keys deben tener los siguientes valores:

Tabla	Key	Valor
COLORS	1	Black
COLORS	2	Red
MARCAS	1	Adidas
CELULARES	1	iPhone
CELULARES	2	Android

Prueba Kernel

Configuración del sistema:

VM1 IP: Proceso Kernel <htop> <consola limpia>	VM2 IP: Proceso Memoria 2 y 4 <htop> <consola limpia>
VM3 IP: Proceso LFS <htop> <consola limpia> Proceso Memoria 1	VM4 IP: Proceso Memoria 3 <htop> <consola limpia>

Archivos de Configuración

LFS

```
PUERTO_ESCUCHA=5003
PUNTO_MONTAJE="/home/utnso/lfs-prueba-kernel/"
RETARDO=0
TAMAÑO_VALUE=60
TIEMPO_DUMP=60000
```

Memoria 1

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
PUERTO_SEEDS=[IP_MEMORIA3]
RETARDO_MEM=1000
RETARDO_FS=0
TAM_MEM=4096
RETARDO_JOURNAL=70000
RETARDO_GOSSIPING=5000
MEMORY_NUMBER=1
```

Memoria 2

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA4]
PUERTO_SEEDS=[]
RETARDO_MEM=1000
RETARDO_FS=0
TAM_MEM=2048
RETARDO_JOURNAL=30000
RETARDO_GOSSIPING=5000
MEMORY_NUMBER=2
```

Memoria 3

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
```

```
PUERTO_SEEDS=[]
RETARDO_MEM=1000
RETARDO_FS=0
TAM_MEM=2048
RETARDO_JOURNAL=70000
RETARDO_GOSSIPING=5000
MEMORY_NUMBER=3
```

Memoria 4

```
PUERTO=8002
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA1]
PUERTO_SEEDS=[]
RETARDO_MEM=1000
RETARDO_FS=0
TAM_MEM=4096
RETARDO_JOURNAL=30000
RETARDO_GOSSIPING=5000
MEMORY_NUMBER=4
```

Kernel

```
IP_MEMORIA=IP_MEMORIA3
PUERTO_MEMORIA=8001
QUANTUM=1
MULTIPROCESAMIENTO=1
METADATA_REFRESH=15000
SLEEP_EJECUCION=1000
```

Metadata FileSystem

```
BLOCK_SIZE=128
BLOCKS=4096
MAGIC_NUMBER=LISSANDRA
```

Actividades:

1. Iniciar todos los módulos (iniciando las memorias por la Memoria 1) y esperar a que el sistema sea consistente, es decir, que el Kernel conozca a todas las memorias.
2. Asociar la Memoria 1 al criterio SC, las Memorias 2 y 4 al criterio SHC y las Memorias 3 y 4 al criterio EC.
3. Ejecutar los scripts `animales.lql`, `comidas.lql` y `internet_browser_falla.lql`
4. Ejecutar el comando METRICS en la consola del Kernel
5. Una vez que finalicen los scripts anteriores, modificar el valor de quantum a 2
6. Ejecutar los scripts `misc_1.lql`, `misc_2.lql` y `cosas_falla.lql`
7. Desconectar la Memoria 3
8. Ejecutar las siguientes request en la consola del Kernel:

```
INSERT POSTRES 63 "Flan"
INSERT COLORES 326 "Gris"
SELECT COSAS 332
SELECT SERIES 89
```

9. Desconectar la Memoria 2 y reconectar la Memoria 3 asociada al criterio SHC
10. Ejecutar el comando METRICS en la consola del Kernel
11. Ejecutar el comando JOURNAL en el kernel y esperar al dump de la memtable.

Resultados Esperados:

1. Inicialmente el Kernel solo debe conocer a la Memoria 3 y luego mediante el proceso de gossiping debe conocer a las demás memorias
2. Los scripts animales, comidas, misc_1 y misc_2 terminan correctamente, mientras que los internet_browser_falla y cosas_falla terminan debido a una falla.
3. Todos los pedidos deben ser resueltos normalmente y los SELECT deben dar los valores insertados previamente.
4. Los pedidos deben ser ejecutados respetando la consistencia de las tablas, siendo ejecutados en la Memoria correcta.
5. Para las tablas cuya consistencia es SHC, los pedidos deben ser distribuidos por medio de la Key y todos los pedidos sobre la misma Key deben ir a la misma memoria.
6. Los scripts se planifican correctamente siguiendo un diagrama de estados de forma secuencial respetando el algoritmo Round Robin y el Quantum indicado.
7. Los scripts son ejecutados en paralelo, respetando el valor que indica el campo de multiprocesamiento

Prueba LFS

Configuración del sistema:

VM1 IP: Proceso Kernel <htop> <consola limpia>	VM2 IP: Proceso Memoria 1 <htop> <consola limpia>
VM3 IP: Proceso LFS <htop> <consola limpia> Proceso Memoria 2	VM4 IP: Proceso Memoria 3 <htop> <consola limpia>

Archivos de Configuración

LFS

```
PUERTO_ESCUCHA=5003
PUNTO_MONTAJE="/home/utnso/lfs-compactacion/"
RETARDO=0
TAMAÑO_VALUE=60
TIEMPO_DUMP=5000
```

Memoria 1

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA2]
PUERTO_SEEDS=[8001]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=320
RETARDO_JOURNAL=60000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=1
```

Memoria 2

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA3]
PUERTO_SEEDS=[8001]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=320
RETARDO_JOURNAL=60000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=2
```

Memoria 3

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
```

```
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=320
RETARDO_JOURNAL=60000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=3
```

Kernel

```
IP_MEMORIA=IP_MEMORIA1
PUERTO_MEMORIA=8001
QUANTUM=3
MULTIPROCESAMIENTO=1
METADATA_REFRESH=15000
SLEEP_EJECUCION=10
```

Metadata FileSystem

```
BLOCK_SIZE=64
BLOCKS=4096
MAGIC_NUMBER=LISSANDRA
```

Actividades:

1. Ejecutar todos los módulos (iniciando las memorias por la Memoria 1) y esperar a que el sistema sea consistente, es decir, que el Kernel conozca ambas memorias.
2. Asociar la Memoria 1, 2 y 3 al criterio EC y ejecutar el script `compactacion_larga.lql` y esperar a que finalice.
3. Ejecutar el comando JOURNAL desde el Kernel.
4. Desconectar el proceso LFS y relevantarlo
5. Ejecutar dentro de la consola del LFS las siguientes request:
   ```
   SELECT METALS 198
   SELECT METALS 135
   SELECT ANIMALS 110
   ```
6. Ejecutar dentro de la consola del kernel:
   ```
   DESCRIBE
   SELECT METALS 198
   SELECT METALS 135
   SELECT ANIMALS 110
   ```

Resultados Esperados:

La prueba forzará en todo momento JOURNAL de las distintas memorias y dado el bajo tiempo de dump se generarán múltiples archivos. Dadas las condiciones de tiempos se espera que esta prueba esté ejecutando por lo menos 1 minuto.

1. Inicialmente el Kernel solo debe conocer a la Memoria 1 y luego mediante el proceso de gossiping debe conocer a la Memoria 2.
2. Se debe verificar en el transcurso de la prueba que se hagan múltiples JOURNAL de las distintas memorias y ver cómo se crean los distintos archivos temporales gracias al proceso dump.

3. Los archivos temporales creados mediante el proceso de dump deben ser una réplica exacta de lo que se encontraba en la memtable de cada tabla
4. Verificar que durante el proceso de la prueba se ejecute la compactación y los archivos temporales sean eliminados y compactados.
5. Al finalizar la ejecución del script, esperar a que vuelva a ejecutarse la compactación. Se debe ejecutar el comando SELECT a disposición del ayudante sobre distintas keys y tablas dando el resultado que se encuentra en `compactacion_larga.txt`
6. El LFS debe poder levantarse ante una caída, manteniendo las Keys y sus valores.

Prueba Memoria

Configuración del sistema:

VM1 IP: Proceso Kernel <htop> <consola limpia>	VM2 IP: Proceso Memoria 1 <htop> <consola limpia>
VM3 IP: Proceso LFS <htop> <consola limpia>	VM4 IP:

Archivos de Configuración

LFS

```
PUERTO_ESCUCHA=5003
PUNTO_MONTAJE="/home/utnso/lfs-prueba-memoria/"
RETARDO=0
TAMAÑO_VALUE=60
TIEMPO_DUMP=3000
```

Memoria 1

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
PUERTO_SEEDS=[8001]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=340
RETARDO_JOURNAL=1000000
RETARDO_GOSSIPING=10000
MEMORY_NUMBER=1
```

Kernel

```
IP_MEMORIA=IP_MEMORIA1
PUERTO_MEMORIA=8001
QUANTUM=3
MULTIPROCESAMIENTO=1
METADATA_REFRESH=15000
SLEEP_EJECUCION=0
```

Metadata FileSystem

```
BLOCK_SIZE=64
BLOCKS=4096
MAGIC_NUMBER=LISSANDRA
```

Actividades:

1. Ejecutar todos los módulos

2. Asociar la Memoria 1 al criterio SC y ejecutar el script `reemplazo1.lql` y esperar a que finalice.
3. Ejecutar el comando JOURNAL desde Memoria.
4. En la consola del LFS ejecutar los siguientes request:
   ```
   INSERT POSTRES 819 “Alfajor”
   INSERT POSTRES 178 “Mousse”
   ```
5. Ejecutar el script `reemplazo2.lql` y esperar a que finalice.
6. Ejecutar por consola de LFS las siguientes request:
   ```
   SELECT POSTRES 100
   SELECT POSTRES 101
   SELECT POSTRES 1888
   SELECT POSTRES 80
   SELECT POSTRES 120
   SELECT POSTRES 999
   ```

Resultados Esperados:

1. Se debe verificar que inicialmente todos los inserts solicitados en reemplazo1.lql se escriben en distintas páginas de la Memoria sin necesidad de ejecutar el algoritmo de reemplazo
2. Se debe verificar que al correr el script reemplazo2.lql todas las páginas inicialmente tengan el flag de modificado en 0 y una vez que se realizan los inserts todas cambien a 1
3. Al correr el script reemplazo2.lql las páginas que contienen las Keys 819 y 178 deben ser reemplazadas
4. Al querer ejecutar el comando que inserta en la Key 999 se debe encontrar que la Memoria esta full, informandole al Kernel de este estado y realizando un JOURNAL forzoso a partir de que este se lo solicite
5. Cuando se ejecuten los SELECT en la consola del LFS todos los valores devueltos deben estar actualizados
6. Una vez realizado el JOURNAL la memoria debe estar completamente vacía

Prueba de Stress

Configuración del sistema:

VM1 IP: Proceso Kernel <htop> <consola limpia>	VM2 IP: Proceso Memoria 2 y 4 <htop> <consola limpia>
VM3 IP: Proceso LFS <htop> <consola limpia> Proceso Memoria 1	VM4 IP: Proceso Memoria 3 y 5 <htop> <consola limpia>

Archivos de Configuración

LFS

```
PUERTO_ESCUCHA=5003
PUNTO_MONTAJE="/home/utnso/lfs-stress/"
RETARDO=0
TAMAÑO_VALUE=60
TIEMPO_DUMP=20000
```

Memoria 1

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=4096
RETARDO_JOURNAL=70000
RETARDO_GOSSIPING=30000
MEMORY_NUMBER=1
```

Memoria 2

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA1]
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=2048
RETARDO_JOURNAL=30000
RETARDO_GOSSIPING=30000
MEMORY_NUMBER=2
```

Memoria 3

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[]
```

```
PUERTO_SEEDS=[IP_MEMORIA4, IP_MEMORIA2]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=256
RETARDO_JOURNAL=70000
RETARDO_GOSSIPING=30000
MEMORY_NUMBER=3
```

Memoria 4

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA1]
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=4096
RETARDO_JOURNAL=30000
RETARDO_GOSSIPING=30000
MEMORY_NUMBER=4
```

Memoria 5

```
PUERTO=8001
PUERTO_FS=5003
IP_SEEDS=[IP_MEMORIA2, IP_MEMORIA3]
PUERTO_SEEDS=[]
RETARDO_MEM=0
RETARDO_FS=0
TAM_MEM=1024
RETARDO_JOURNAL=50000
RETARDO_GOSSIPING=30000
MEMORY_NUMBER=5
```

Kernel

```
IP_MEMORIA=IP_MEMORIA5
PUERTO_MEMORIA=8001
QUANTUM=1
MULTIPROCESAMIENTO=3
METADATA_REFRESH=15000
SLEEP_EJECUCION=10
```

Metadata FileSystem

```
BLOCK_SIZE=128
BLOCKS=8192
MAGIC_NUMBER=LISSANDRA
```

Actividades:

1. Ejecutar todos los módulos (iniciando las memorias por la Memoria 1) y esperar a que el sistema sea consistente, es decir, que el Kernel conozca a todas las memorias.
2. Asociar la Memoria 5 al criterio SC, las Memorias 1, 2, 3 y 4 al criterio SHC y las Memorias 2, 3 y 5 al criterio EC.
3. Ejecutar los scripts `cities_countries.lql`, `games_computer.lql` y `internet_browser_falla.lql`
4. Antes de que finalicen los scripts anteriores, modificar el valor del quantum a 3
5. Ejecutar los scripts `library_study.lql`, `nintendo_playstation.lql` y `cosas_falla.lql`
6. Desconectar la Memoria 4 y reconectarla al criterio EC
7. Desconectar la Memoria 2 y reconectarla asociada al criterio SHC
8. Ejecutar el comando METRICS en la consola del Kernel
9. Ejecutar el comando JOURNAL en el kernel y esperar al dump de la memtable.

Resultados Esperados:

1. Inicialmente el Kernel solo debe conocer a la Memoria 3 y luego mediante el proceso de gossiping debe conocer a las demás memorias
2. Todos los scripts finalizan correctamente
3. Todos los pedidos deben ser resueltos normalmente y los SELECT deben dar los valores insertados previamente.
4. Los pedidos deben ser ejecutados respetando la consistencia de las tablas, siendo ejecutados en la Memoria correcta.
5. Para las tablas cuya consistencia es SHC, los pedidos deben ser distribuidos por medio de la Key y todos los pedidos sobre la misma Key deben ir a la misma memoria.
6. Los scripts se planifican correctamente siguiendo un diagrama de estados de forma secuencial respetando el algoritmo Round Robin y el Quantum indicado.
7. Los scripts son ejecutados en paralelo, respetando el valor que indica el campo de multiprocesamiento

Planilla de Evaluación - TP1C2019

Nombre del Grupo	Nota (Grupal)

Legajo	Apellido y Nombres	Nota (Coloquio)

Evaluador/es Práctica	
Evaluador/es Coloquio	

Observaciones:

Sistema Completo	
El deploy se hace de forma automatizada y en un tiempo límite de 10 a 15 minutos	
Los procesos ejecutan de forma simultánea y la cantidad de hilos y subprocesos en el sistema es la adecuada	
Los procesos establecen conexiones TCP/IP y se comunican mediante un protocolo	
El sistema no registra casos de Espera Activa ni Memory Leaks	
El sistema responde de forma resiliente a la interacción con el entorno	
Se utilizaron de forma criteriosa los métodos estudiados para el manejo de múltiples conexiones *(multiplexado y arquitecturas multi-hilos)*	
El log permite determinar en todo momento el estado actual y anterior de los diversos procesos y del sistema junto con sus cambios significativos	
El sistema continúa su funcionamiento ante comandos erróneos o paths inexistentes (informándole al usuario el error)	
El sistema no requiere permisos de superuser (sudo/root) para ejecutar correctamente	
El sistema no requiere de Valgrind o algún proceso similar para ejecutar correctamente	

General Lissandra (todos los módulos)	
El módulo responde correctamente los SELECT	
El módulo responde correctamente los INSERT	
El módulo responde correctamente los CREATE	
El módulo responde correctamente los DESCRIBE	
(*) El módulo responde correctamente los DROP	
Todas las operaciones pueden correrse desde la consola del módulo	
El sistema detecta y responde a modificaciones en los archivos de configuración en tiempo de ejecución	

Proceso Lissandra File System	
Genera correctamente los DUMPS luego del tiempo estipulado por archivo de configuración	
Genera archivos temporales con el contenido exacto de la Memtable	
Las estructuras del FileSystem se actualizan correctamente ante cambios en los archivos	
Ante cambios en las estructuras del FileSystem el sistema informa correctamente cuáles son las operaciones realizadas (bloques que cambiaron, metadatas, archivos)	
El sistema compacta correctamente las particiones con sus archivos temporales	

Al correr una compactación no se bloquea el sistema	
(*) La compactación bloquea la tabla por un período mínimo de tiempo	
Responde pedidos en paralelo, utilizando threads para procesar los mismos	
Las estructuras, tablas y registros tan solo se bloquean siguiendo las condiciones de Bernstein	
El LFS continua correctamente su ejecución luego de ser desconectado y relevantado	

Proceso Memoria (Standalone)	
Funciona utilizando la implementación indicada de Segmentación Paginada	
Las páginas se guardan en un único espacio **contiguo** de memoria principal	
(*) Al no quedar frames libres, el sistema reemplaza las mismas siguiendo el algoritmo LRU	
Informa correctamente cuando se encuentra FULL	
La memoria realiza journaling en los períodos de tiempo indicado	
Luego de realizar journaling, la memoria vacía su contenido	
Puede ejecutar journaling a través del comando JOURNAL en consola	
Utiliza correctamente las técnicas de manejo de múltiples conexiones	

Proceso Memoria (Cluster)	
(*) Las memorias pueden autodescubrirse mediante el proceso de Gossiping	

Proceso Kernel	
Mantiene los estados de ejecución indicados, e informa por consola la transición de los scripts entre los mismos	
Permite correr scripts LQL utilizando el comando RUN	
Puede forzar a las memorias conectadas a realizar journaling mediante el comando FORCE JOURNAL	
Redescubre las memorias del sistema utilizando gossiping	
Permite correr varios scripts en paralelo utilizando múltiples estados de EXEC	
(*) Imprime las métricas del sistema	
Identifica las consistencias de las tablas y elige correctamente el criterio a ejecutar	
El criterio EVENTUAL CONSISTENCY elige correctamente la memoria a quien enviar el pedido	

El criterio STRONG CONSISTENCY elige correctamente la memoria a quien enviar el pedido	
(*) El criterio STRONG HASH CONSISTENCY elige correctamente la memoria a quien enviar el pedido	
Permite asociar memorias a los criterios utilizando el comando ADD	