UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

Y.A.M.A.

Documento de pruebas

Cátedra de Sistemas Operativos

Trabajo práctico Cuatrimestral

- 2C2017 -
Versión [1.1]

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 & 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.

Los juegos de datos (datasets) y scripts se podrán encontrar en su correspondiente repositorio de github.

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

Se recuerda a los grupos, **la necesidad de la ejecución del comando** `export LC_ALL=C` **en cuando inicien una sesión**. Esto surge de que lenguajes como Python, C y la mayoría que comparan strings según su valor ASCII, se necesita que el comando sort use el mismo criterio de ordenamiento. Esto se logra seteando la variable LC_ALL=C, que indica que se use el pseudo-locale "C" para comparar, y mediante el export, se asegura que todos los procesos ejecutados en dicha sesión tengan configurado el mismo locale.

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

Pruebas

Configuración para la evaluación

Configuración del sistema:

VM1 IP: Master 1 Nodo 1 <htop>[2] <consola limpia>	VM2 IP: Master 2 Nodo 2 <htop> <consola limpia>
VM3 IP: YAMA Master 3 Nodo 3 <htop>	VM4 IP: FileSystem Master 4 Nodo 4 <htop>

Nodos

Los Nodos deben tener el siguiente esquema de espacio de datos:

Nodo1	Nodo2	Nodo3	Nodo4
60MB	90MB	150MB	150MB

Configuración de los Master

Cada job N°X va a ejecutar los scripts para obtener la cantidad de personas nacidas entre el 1920 y el 2015 que compartan el mismo nombre.

Variable	Valor
Transformador	transformador.py
Reductor	reductor.py
Archivos	nombres.csv (dataset reducido o completo)
Resultado	yamafs:/out/jobX/resultado.csv
MD5(resultado)	8b11d4503f9c0e4d515cbf838fe4b114 (dataset reducido) 9aefc44eda8acf0fff18a2bb70e8c662 (dataset completo)

[2] Ver Anexo Comandos Útiles

Detalle de los set de Datos

Los scripts de transformación y reducción corresponden al mismo set de datos en dos sabores, en un mismo archivo y divididos en varios archivos más pequeños.

Juego de Datos:

- Completo: nombres.csv (c48348d0f25469a239f5533d23645e37) [212 MB]
- Reducido: nombres.csv (0f25d7d69f1fca92bf9e3c045f577e81) [34 MB]

Juego de Datos Completo divididos y sanitizados:

- nombres-1920-1924.csv (8e0889006104c379122aaea357f8f0bf) [803 KB]
- nombres-1925-1929.csv (9505a7a66835ba6bf2e66ca321cffa1a) [2.7 MB]
- nombres-1930-1934.csv (9028dce969ef29fbda60bd07efc27c92) [5.0 MB]
- nombres-1935-1939.csv (554c55626ef07679c44c04330100b1a1) [6.5 MB]
- nombres-1940-1944.csv (67d756ffbf9620e26f66088a6ff27821) [7.5 MB]
- nombres-1945-1949.csv (694dc0ed4a4751cc554fdd10151296fe) [8.2 MB]
- nombres-1950-1954.csv (50cd4eb0ee0cf5f37f0e3339d6301c85) [8.7 MB]
- nombres-1955-1959.csv (087d6f930dbc354d3612578d427f4c29) [9.2 MB]
- nombres-1960-1964.csv (b74ca29a6541f723416ea5dde82f0fc5) [11 MB]
- nombres-1965-1969.csv (1316cff6cbc1fbd001b79646b93b94dc) [10 MB]
- nombres-1970-1974.csv (6ca134a99519a3252a44eeadf8100bc6) [12 MB]
- nombres-1975-1979.csv (8cec55bd1d9292fbc758c1539889b71c) [14 MB]
- nombres-1980-1984.csv (a45274e36ca3af8b6907cbf4140a4099) [14 MB]
- nombres-1985-1989.csv (24c229a26fa857c434edd61f51e2b311) [17 MB]
- nombres-1990-1994.csv (a5ed42d689fa716d069f7fb471b30a60) [20 MB]
- nombres-1995-1999.csv (1e0963b2e7f4e64f48e39c8c3212ea09) [19 MB]
- nombres-2000-2004.csv (68d581ed57d21e535b5ceb7cae1ffa7b) [19 MB]
- nombres-2005-2009.csv (a8d65db97884b8119f2ffac56624ab84) [17 MB]
- nombres-2010-2014.csv (bb77b78dbd3dd1c53a465e79c536f321) [20 MB]
- nombres-2015.csv (aecd04fe69320c929c152417ed63630f) [4.7 MB]

Resultado esperado (md5): 72a8ee320e4d5dda419fccb90022d869

Otros scripts de Transformación

Si bien el script de reducción es el mismo para todas las diferentes transformaciones, existen dos transformaciones extras que podrán ser utilizadas durante la prueba:

1. El conjunto formado con el *transformador_iniciales.py*: El cual obtiene la cantidad de personas que tienen la mismas iniciales, nacidas del 1920 hasta el 2015.
    a. **md5** dataset Completo: 975f82dcc9da6722b712bd981f1ef822
    b. **md5** dataset Reducido: *b20e8cce4cd27a6b843c949e20941430*
2. El conjunto formado con el *transformador_anuales.py*: El cual obtiene la cantidad de personas que nacieron en cada año entre el 1920 hasta el 2015.
    a. **md5** dataset Completo: 610e5e7e04db3a5753d796041f8a1098
    b. **md5** dataset Reducido: *16512ddf3406cb209c70a705b7f6595e*

Etapa 1

Se iniciará el **yamafs:** conectando sus correspondientes nodos. Se deberá formatear el mismo y crear los directorios */base*, */out/job1* y /out/job2.
Copiar los archivos de los tests a **yamafs:/base** y evaluar la asignación y distribución de bloques en los procesos Nodos. Modificar la ruta de origen de los archivos copiados en **yamafs** y corroborar que:

1. Al hacer un *md5* sobre dichos archivos originales movidos y sobre los archivos dentro de nuestro sistema **yamafs** los resultados sean iguales.
2. Se hayan distribuido equitativamente los bloques dentro de los nodos del yamafs. Los nodos deben tener un carga equitativa de datos

Etapa 2

Iniciar el proceso **YAMA** con algoritmo W-Clock y retardo en 500ms. Luego iniciar el proceso **Master 1 y Master 2**. Corroborar durante la ejecución que:

1. Se cumpla el orden de las etapas en cada Job y la existencia de los archivos temporales en el *yamafs*
2. Se respete el balanceo de carga en los nodos.

Una vez finalizados ambos Jobs corroborar que:

1. Los resultados obtenidos mediante la comparación de sus MD5 sean iguales.

Etapa 3

Iniciar los procesos **Master 3 y Master 4**. Durante la ejecución de la etapa de transformación, desconectar un **Nodo** y validar que:

1. El sistema re planifica las transformaciones pérdidas.
2. Se respeta la carga del sistema en la replanificación.
3. Corroborar que se cumpla el orden de las etapas en cada Job y la existencia de los archivos temporales
4. El o los job/s replanificado/s terminan y al ejecutar dos nuevos master estos finalizan con normalidad (*re ejecución de etapa 2*)
5. El o los job/s replanificado/s terminan correctamente y su resultado por medio de sus MD5 son iguales.

Cambiar el algoritmo de balanceo a **Clock**, ejecutar nuevamente **Master 1** y **Master 2**. Chequear que:

1. Se cumpla el orden de las etapas en cada Job y la existencia de los archivos temporales en el ***yamafs***
2. Se respete el balanceo de carga en los nodos.
3. Los resultados obtenidos mediante la comparación de sus MD5 sean iguales.

Etapa 4

Desconectar el proceso FileSystem. Volver a ejecutarlo y corroborar que se ha iniciado en **estado no-seguro**. Luego iniciar los Nodos en orden numérico corroborando, el cambio a **estado seguro** luego de poseer por lo menos la 1 copia de cada archivo.

Desconectar un Nodo. Ejecutar Master 1 y verificar que el sistema responde con normalidad. Una vez finalizado, conectar nuevamente el Nodo y ejecutar Master 2. Validar que:

1. El nodo conectado nuevamente comienza a tener carga.
2. Los resultados obtenidos de los master por comparación de sus MD5 sean iguales.

Se iniciará nuevamente el **yamafs** con el flag de limpio. Validar que se han eliminado las estructuras administrativas. Conectar sus correspondientes nodos y formatear dicho filesystem. Crear los directorios /prueba, /prueba/data, /directorios.

Copiar el archivo *nombres.csv* (completo) en el directorio prueba y validar que todos los bloques estén asignados. **Renombrar** el archivo /prueba/nombres.csv a **txt_procesado.csv**

Intentar **copiar** el archivo *nombres.csv (reducido)* y verificar que **falla por no disponer bloques suficientes**. Eliminar el archivo *txt_procesado.csv* y volver a intentar. Verificar que se haya copiado el archivo correctamente revisando que **solo ocupe un único bloque**. Finalmente, **listar** los elementos del directorio */prueba* y mostrar el contenido del archivo *nombres.csv*.
Tratar de **eliminar** el directorio */prueba* y verificar que falla por poseer elementos. Luego eliminar el directorio */prueba/data* y **mover** el archivo *nombres.csv* al directorio /directorios y volver a realizar la eliminación, esta vez de forma exitosa.

Finalmente, haciendo uso del copiar-pegar, crear 96 directorios dentro de /directorios, y validar que no es posible crear otro más. Luego, formatear el **yamafs** nuevamente y validar que las estructuras administrativas se han eliminado y no hubo mensajes a los Nodos.

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 **solo** la última versión del código (en vez de todo el repositorio)

```
curl -u '[usuario]' -L -o [archivo] [url_repo]
```

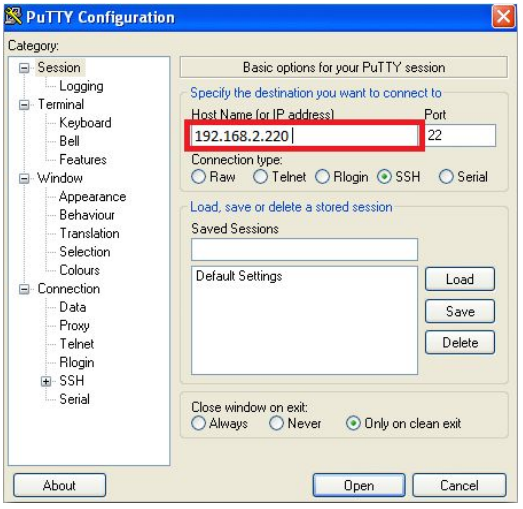
Ejemplo:

```
curl -u 'gatonprieto' -L -o commons.tar
https://api.github.com/repos/sisoputnfrba/so-commons-library/tarball/master
```

Este comando debe ejecutarse sin salto de línea. **Luego descomprimir con:** `tar -xvf commons.tar`

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

Planilla de Evaluación - TP2C2017

Nombre del Grupo	Nota (Grupal)

Legajo	Apellido y Nombres	Nota (Coloquio)

Evaluador/es Práctica	
Evaluador/es Coloquio	

Observaciones:

Sistema Completo	
Contenidos Mínimos	
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 MD5 de un archivo final producto de del sistema es el mismo que si dicha tarea fuera realizada de forma manual.	
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	

Proceso Master	
Contenidos Mínimos	
El proceso Master respeta las etapas de preplanificación según la especificación	
La desconexión del proceso Master deja al sistema en un estado estable	
El proceso Master informa correctamente la cantidad de tareas ejecutadas en un Job y el tiempo de ejecución del mismo	
Envía de forma correcta los scripts de las operaciones de transformación/reducción a los Workers	
El proceso master informa correctamente el tiempo promedio de ejecución de cada etapa y la cantidad de tareas realizadas al simultáneo	
Contenidos de Promoción	
El proceso master informa correctamente los fallos ocurridos durante la ejecución de los Jobs.	
La desconexión de un proceso Master interrumpe la ejecución de todos los Workers, e impacta en la carga total del sistema.	
Al recibir una instrucción de replanificación desde YAMA, el proceso Master delega las tareas en forma paralela y efectiva a los Workers, respetando las etapas de transformación, reducción local y reducción global	

Proceso YAMA	
Contenidos Mínimos	
El proceso YAMA es agnóstico a la ejecución de las operaciones	
Preplanifica de forma adecuada la carga entre los procesos Worker	
La ejecución de algoritmo N-Clock al pre-planificar es correcta	
La ejecución de algoritmo W-Clock al pre-planificar es correcta	
Informa correctamente el estado de todos los Nodos y sus Etapas	
Replanifica operaciones de **transformación** ante una desconexión de un Nodo en una copia del bloque	
La carga del Sistema se balancea correctamente con la entrada y salida de Jobs	
Al fallar una operación el sistema queda estable, eliminando los jobs que fallaron de la carga del sistema	
Contenidos de Promoción	
Al replanificar la carga es pareja y se respeta el algoritmo de re-planificación	
Ante la necesidad de replanificar, se permite al proceso Job terminar de ejecutar devolviendo el resultado correcto	
Al fallar una operación el sistema queda estable, cancelando las operaciones de ese Job que se encontraban ejecutando hasta el momento	
Si no es posible replanificar una etapa de transformación, el proceso YAMA debe dar por fallida la operación total	
Maneja Señales de forma correcta sin desestabilizar al sistema	
Detalle	
Los nombres de archivos temporales son aleatorios y únicos	
Recarga la configuración al recibir la señal SIGUSR1	
El proceso de re-planificación toma en cuenta el cambio del Nodo Encargado	

Sistema FileSystem	
Contenidos Mínimos	
Divide los archivos a escribir en bloques de tamaño fijo con su respectiva copia en un nodo dif.	
Informa de forma correcta la cantidad de bloques libres y usados de cada nodo y el espacio total	
Realiza correctamente el balanceo de los bloques libres ante una petición de escritura	
El formateo se realiza de forma lógica, sin impactar sobre los datos almacenados	
El MD5 de un archivo copiado a proceso Filesystem es idéntico al del archivo original	
Las estructuras del FileSystem se persisten y son correctas	
El proceso FileSystem puede reconstruir su estado anterior, iniciando como no-seguro	
El sistema aprovecha el paralelismo de los DataNodes para acceder a la información	
El sistema respeta correctamente los escenarios de estabilidad y disponibilidad del sistema	
El FileSystem guarda correctamente archivos cuyos bloques no estén alineados con el tamaño del bloque (<1MB)	
Se la reconexión de un nodo caído al FileSystem	
Contenidos de Promoción	
Las operaciones sobre archivos(eliminar, renombrar, mover) y directorios (borrar, crear, renombrar, mover) se realizan de forma lógica, sin impactar sobre los datos almacenados	
La consola del FIleSystem cumple con el comportamiento de todos los comandos requeridos (rm | rename | mv | cat | mkdir | cpfrom/to | cpnode | ls | info)	
No se permite crear más de 100 directorios	
No permite que existan directorios con nombres duplicados	
Se pueden guardar archivos con el mismo nombre en distintos directorios	
Detalle	
El proceso FS informa correctamente los nodos que componen a un archivo	
No se puede eliminar un directorio que posee contenido	
Si se encuentra en un estado no-seguro, rechaza la conexión de YAMA	
El FileSystem soporte +2 copias de un bloque de un archivo	
La consola provee una buena interacción con el usuario	

Componente Nodo	
Contenidos Mínimos	
Los procesos DataNode y Worker son agnósticos al contenido del data.bin	
El proceso Worker es agnóstico al contenido del programa recibido para la etapa a realizar	
El proceso Worker ejecuta las operaciones de las etapas mediante redirecciones de stdin y stdout utilizando pipelines con fork/execv o la función system()	
Luego de una etapa de transformación, se efectúa correctamente un sort	
Los archivos de cada operación de reducción local y global se aparean antes de comenzar las mismas	
El proceso Worker del Nodo marcado como Encargado, se comunica correctamente con los demás nodos	
Los procesos Workers permiten recibir solicitudes de tanto procesos Master como otros procesos Workers actuando como Encargados	
Contenidos de Promoción	
La interfaz especificada entre los DataNode y el proceso FileSystem se respeta	
Las estructuras administrativas se almacenan en memoria y no en el data.bin	
Detalle	
Se ha utilizado mmap() para el acceso a los datos del archivo.bin	
No se ha utilizado la función system() para ejecutar los scripts	