Caso 2 - Informe Máquina Virtual

Grupo 19
Sergio Soler Garzon - 202310103
Raúl Ruiz: 202321332

Tecnología e Infraestructura Computacional

Ricardo Gómez

Universidad de los Andes
Ingeniería de Sistemas y Computación
Bogotá D.C.
2025

1. **Descripción del algoritmo usado para generar las referencias de página**

Se generan referencias para sumar dos matrices M1 y M2 en M3 (enteros de 4 bytes) en orden row-major. Para cada celda (i,j) se calculan direcciones virtuales lineales: baseM1 + (i*N + j)*4, baseM2 + (i*N + j)*4, baseM3 + (i*N + j)*4; luego se derivan VPN = dir//TP y offset = dir%TP. La secuencia por celda es: leer M1[i,j], leer M2[i,j], escribir M3[i,j]. En nuestro código, esto lo implementa el generador en App.Generador mediante Core.Referencia (matrizId, fila, columna, direccionVirtual, numeroPagina, desplazamiento, op). El uso de orden row-major asegura que los accesos sigan la misma organización física que el lenguaje C/Java utiliza en memoria. Esto permite que un conjunto de elementos consecutivos caigan en la misma página, reduciendo fallos. Si el orden fuera column-major, las referencias estarían más dispersas, generando un mayor número de fallos por salto entre páginas.

2. **Descripción de las estructuras de datos usadas para simular el comportamiento del sistema de paginación y cómo usa dichas estructura**

- Tabla de páginas (Core.TablaPaginas): Map<VPN, idMarco> por proceso para resolver hits rápidamente.
- Marcos (Core.Marco): por cada marco se guarda pidDueno, vpnCargada y ultimaReferencia (timestamp) para LRU/aging.
- Estadísticas (Core.Estadisticas): contadores de aciertos, fallos y swaps; se derivan tasas.
- Planificador RR: cola (Deque) que rota procesos por referencia; al finalizar un proceso, sus marcos se reasignan al proceso activo con más fallos (política implementada en SimuladorMotor.finalizarProceso).

En el funcionamiento del simulador, estas estructuras se actualizan dinámicamente en cada referencia. La tabla de páginas se modifica al ocurrir un fallo de página, agregandolo al mapeo de la nueva página cargada, y se limpia cuando una página es reemplazada. Los marcos actualizan el campo ultimaReferencia en cada acceso, sea hit o fallo, usando un reloj lógico global que permite implementar la política de LRU/envejecimiento. Las **estadísticas** se incrementan con base en el resultado de cada referencia: un hit aumenta los aciertos, un fallo suma a los fallos y, en caso de reemplazo, se registra un swap. Finalmente, el planificador Round-Robin reorganiza la cola de procesos tras cada acceso, reasignando marcos de procesos terminados a aquel que acumula más fallos, con el fin de balancear la carga de memoria.

3. **Corra su programa con matrices de 100x100 variando el tamaño de página. Cree una gráfica que ilustre el comportamiento del número de fallas de página con respecto al tamaño de página**

Se midieron fallos para TP ∈ {512, 1024, 2048, 4096, 8192, 16384, 32768} con 64 marcos totales (LRU). El patrón row-major reduce fallos cuando el TP crece hasta estabilizarse. A continuación, la gráfica:

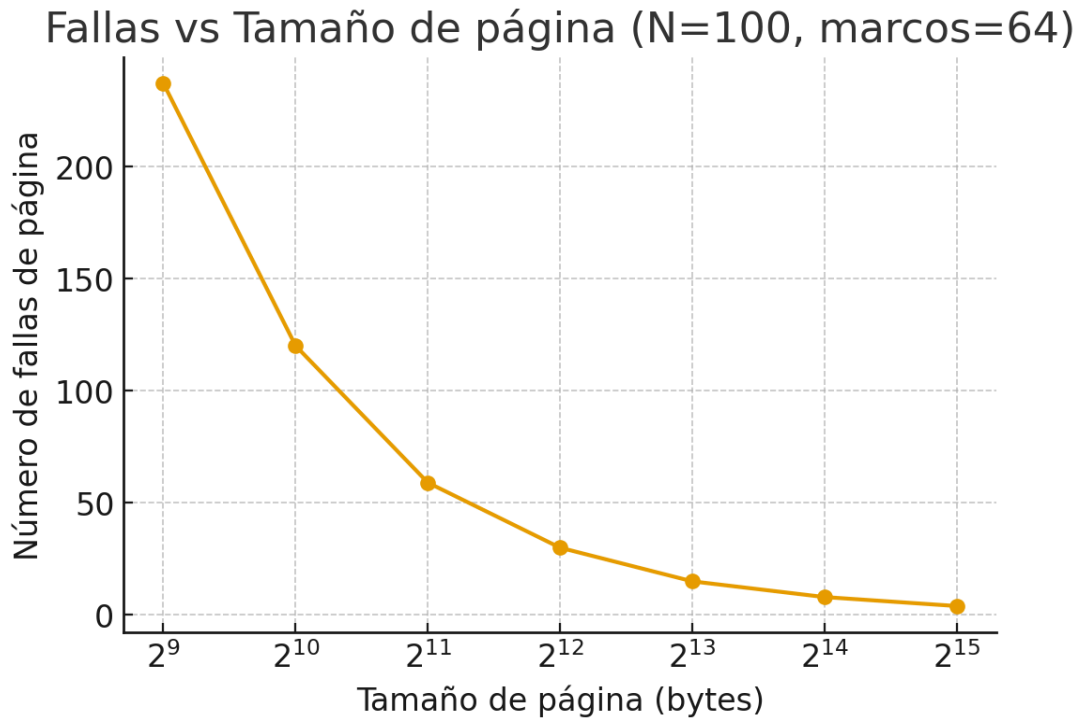

Tabla resumen (TP vs Fallos/Swaps/Referencias):

TP (bytes)	**Fallos**	**Swaps**	**Referencias**
512	237	173	30000
1024	120	56	30000
2048	59	0	30000
4096	30	0	30000
8192	15	0	30000

16384	8	0	30000
32768	4	0	30000

Los resultados muestran una tendencia clara: con páginas pequeñas, el número de fallos es elevado porque una sola fila de la matriz se fragmenta en múltiples páginas. A medida que el tamaño de página aumenta, cada fila ocupa menos páginas, lo que reduce significativamente los fallos. A partir de **2048 bytes**, la reducción se estabiliza y los swaps desaparecen, porque la huella de trabajo de las filas activas cabe en los marcos asignados. Esto refleja la importancia de seleccionar un tamaño de página adecuado: muy pequeño produce sobrecarga por fallos, mientras que demasiado grande puede generar fragmentación interna, aunque en este caso el efecto negativo no es tan evidente por el patrón secuencial de acceso.

4. **Otras 4 configuraciones. Por ejemplo, variando el tamaño de las matrices y el número de marcos, para entender cómo afecta la memoria virtual el desempeño del programa**

Se variaron N y marcos con TP=4096:

1) N=50, marcos=32, la presión es baja y la tasa de fallos también.
2) N=100, marcos=32, la presión sube y los fallos aumentan.
3) N=100, marcos=128, prácticamente todas las páginas caben, los fallos se reducen.
4) N=150, marcos=64, el mayor tamaño de la matriz compensa los marcos extra, manteniendo un nivel intermedio de fallos.

5. **Escriba su interpretación de los resultados: ¿corresponden a los resultados que esperaba? Explique su respuesta**

Los resultados corresponden con la teoría: el acceso por filas produce alta localidad espacial y temporal; al aumentar TP cae el número de fallos hasta un umbral. Con más marcos, disminuyen fallos y swaps porque la huella activa de páginas de la fila actual (M1, M2, M3) cabe con mayor probabilidad. A mayor N, crecen las referencias totales y la presión sobre marcos si éstos se mantienen constantes.

6. **¿Cómo se afecta el tiempo de respuesta de un proceso por el número de accesos a SWAP?**

Cada swap implica I/O a disco, órdenes de magnitud más lento que RAM. Más swaps conlleva más tiempo bloqueado en E/S, es decir mayor latencia por referencia y peor tiempo de respuesta del proceso. Reducir fallos y swaps con TP adecuado y suficientes marcos mejora el tiempo de respuesta.

¿Sumar matrices representa un problema de localidad alta, media o baja? Explique su respuesta en un párrafo de máximo 10 líneas

Sumar matrices presenta un patrón de localidad alta. El recorrido se hace en orden row-major, lo que implica que las direcciones de memoria accedidas son contiguas dentro de cada fila, aprovechando así la localidad espacial. En cada celda (i,j) se leen de inmediato los valores de M1 y M2 y se almacena el resultado en M3, generando además localidad temporal. Cuando el tamaño de página es suficiente, varias posiciones consecutivas se resuelven en la misma página sin generar fallos adicionales. Del mismo modo, si se asigna un número adecuado de marcos, las páginas de las filas activas permanecen en memoria y se reducen fallos y swaps. Por ello, la suma de matrices constituye un problema con localidad claramente alta, lo cual explica el buen desempeño observado en la simulación al variar tanto el tamaño de página como la cantidad de marcos.

Conclusiones

Finalmente, el simulador implementa LRU mediante timestamps (ultimaReferencia) y reparto inicial parejo de marcos por proceso, con reasignación de marcos al terminar un proceso hacia el que más fallos presenta. Las mediciones confirman la relación entre TP, marcos y localidad del algoritmo de suma de matrices.