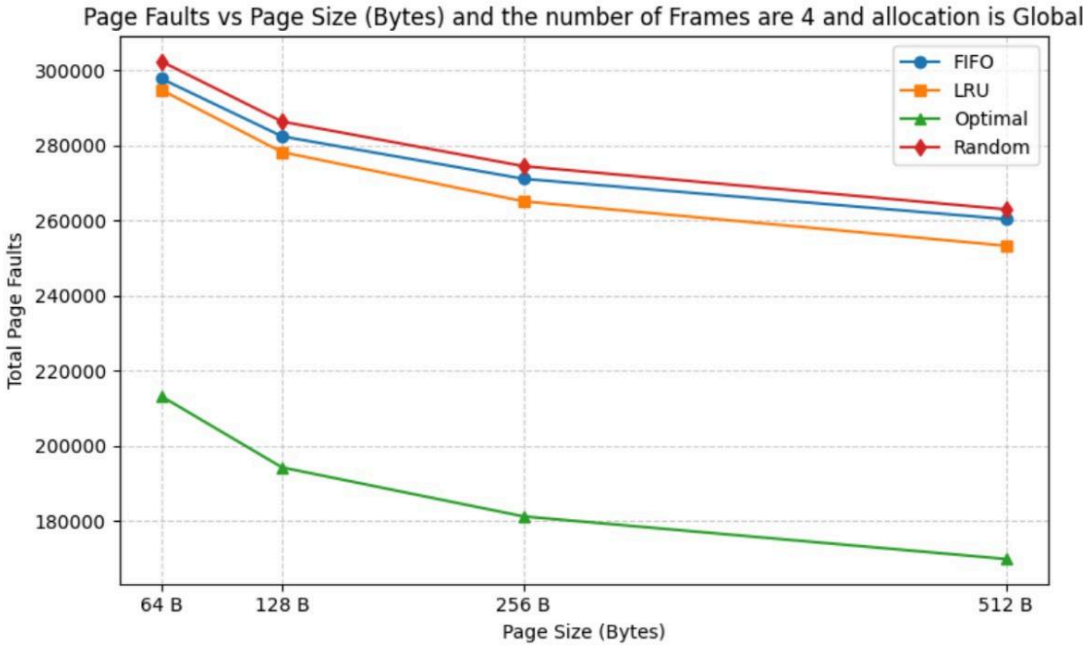
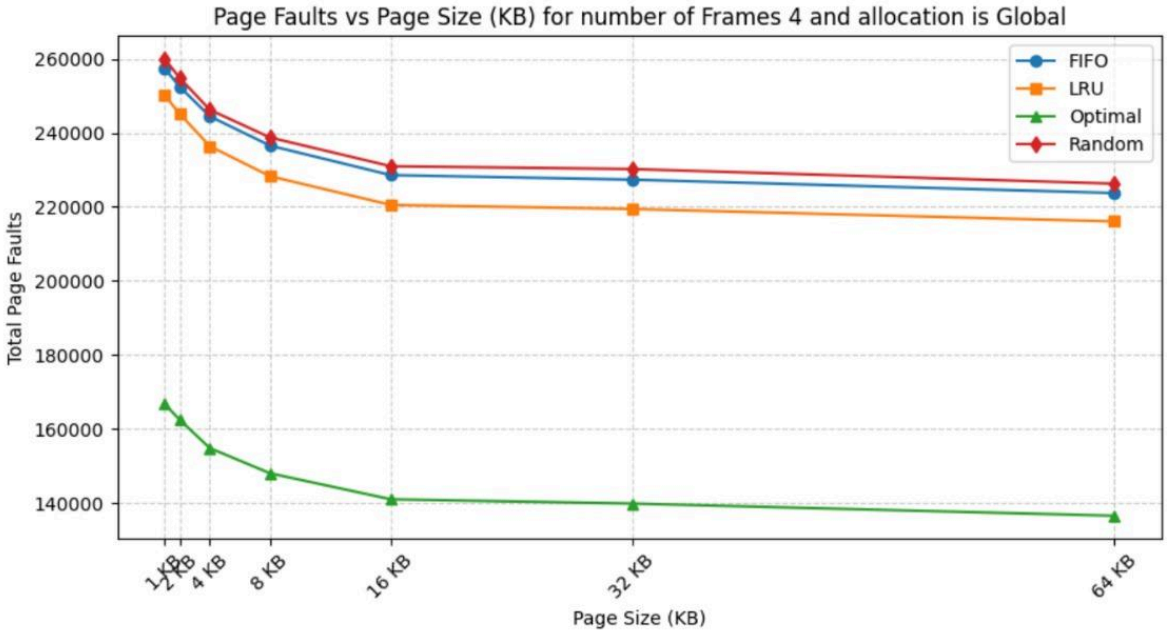
LAB-4(CS23BT013,CS23BT063)

1. **Allocation: Global**
   Number of frames : 4
   Trace file : combined.trace





Conclusion:

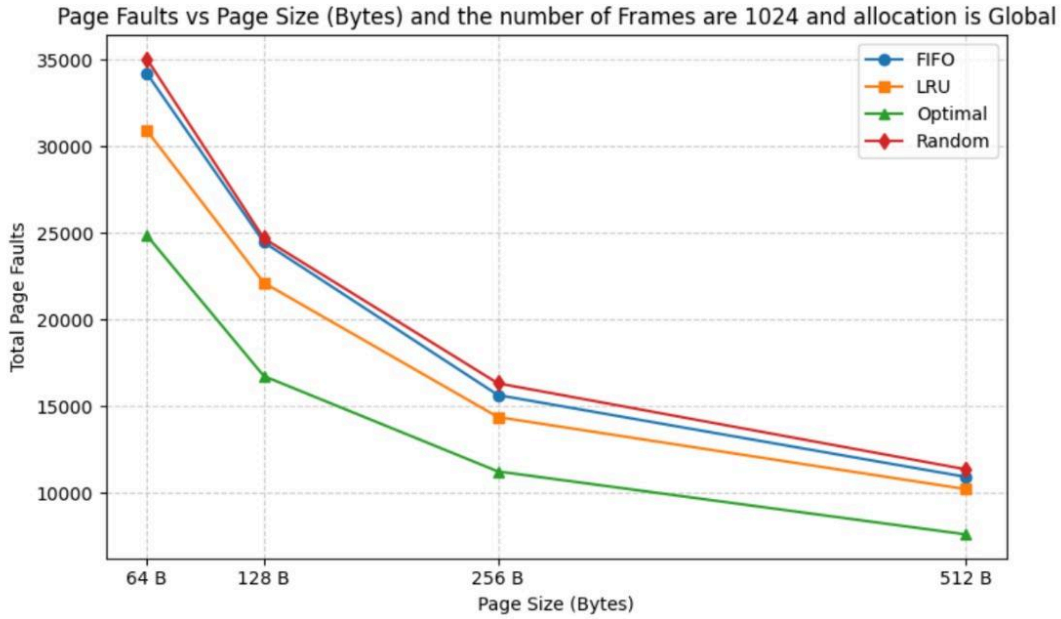
At smaller frame numbers like 4, all page replacement algorithms experience very high page faults because the memory cannot hold the working set of the processes.

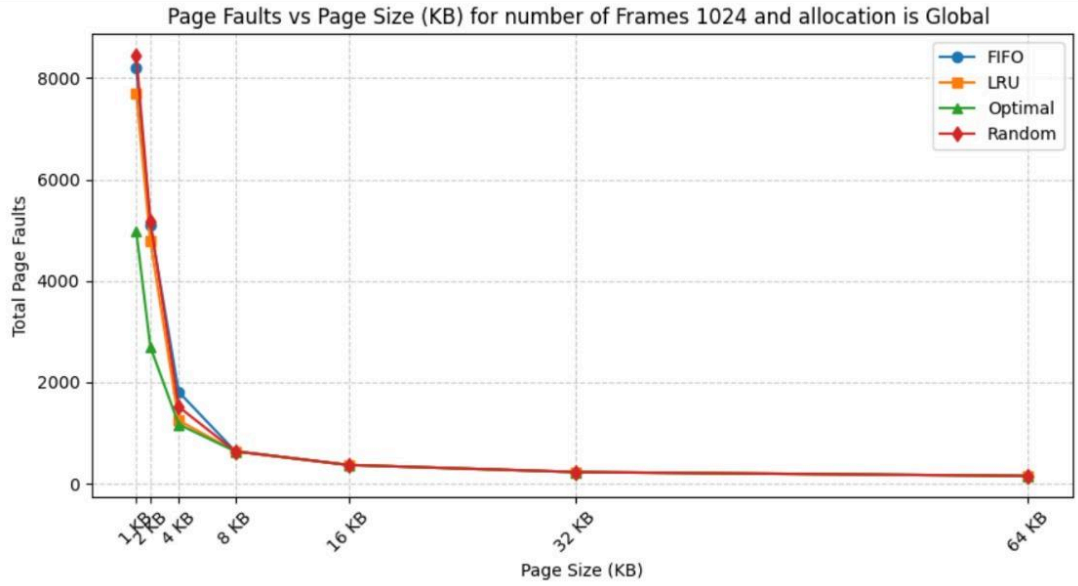
Among them, Optimal performs best, followed by LRU, FIFO, and Random

2. Allocation : Global

Number of frames : 1024

Trace file : combined.trace





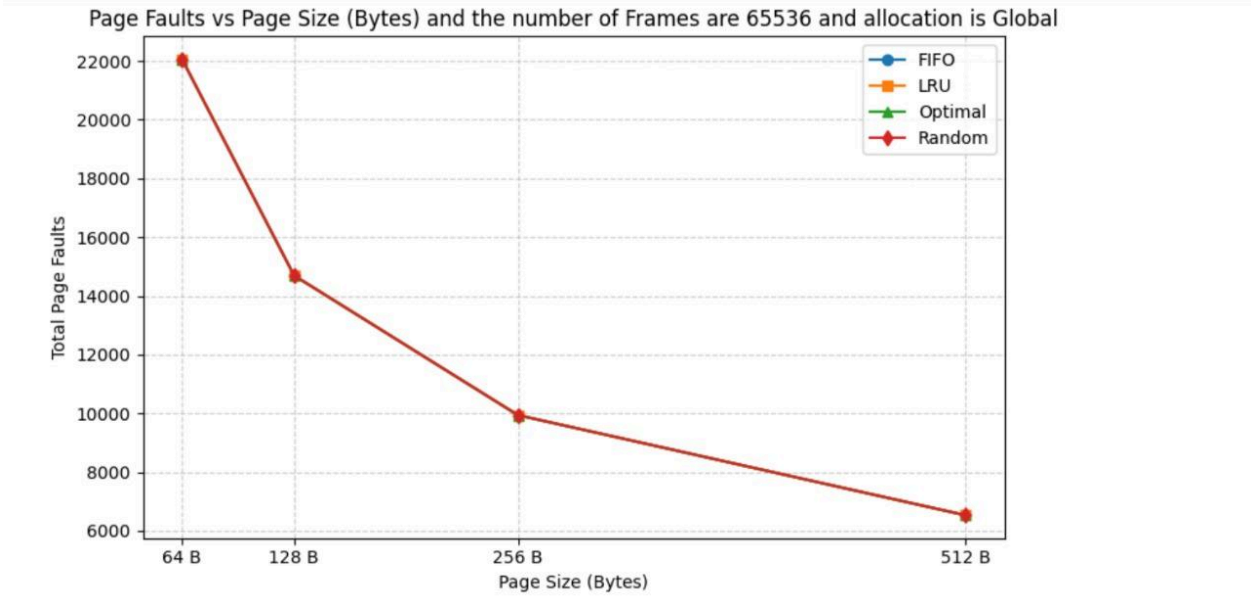
Conclusion:

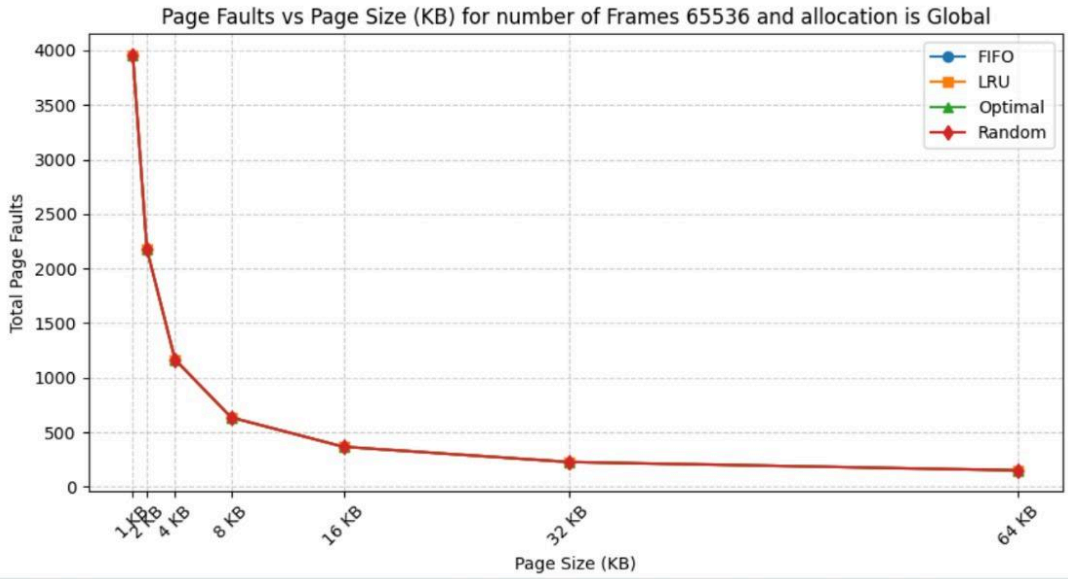
With more frames available(1024), total page faults reduce significantly for all algorithms.

Optimal again performs the best, while LRU closely follows it due to its ability to approximate recent page usage.

FIFO and Random lag behind because they may evict frequently used pages.

3. Allocation : Global
Number of frames : 64k(65536)
Trace file : combined.trace





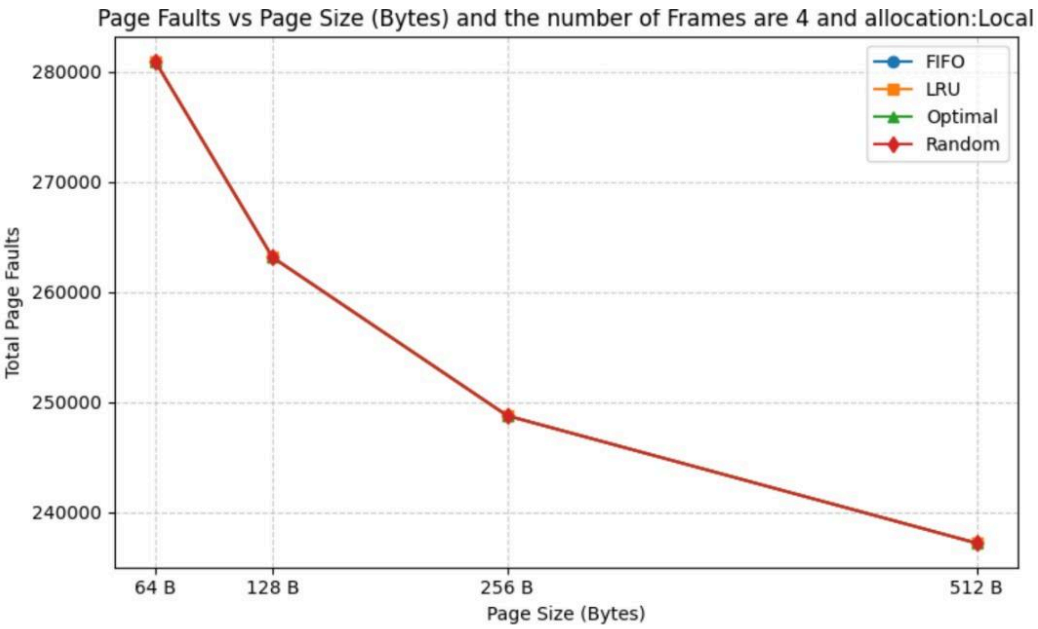
Conclusion:

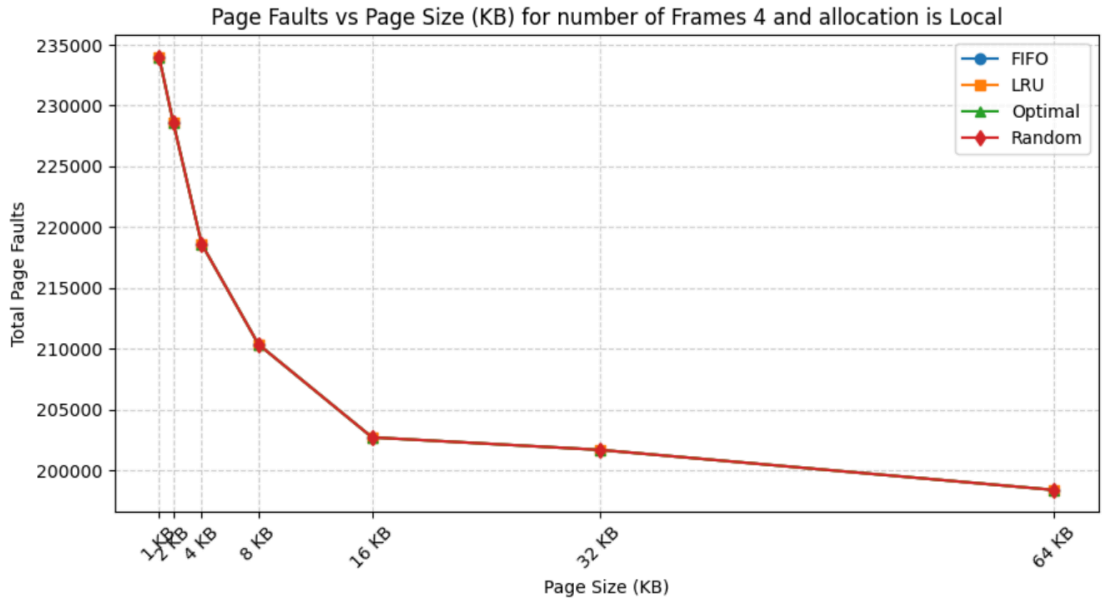
At this very high frame count, nearly all pages fit in memory.
All algorithms converge to similar page fault values, approaching the minimum possible.
The replacement policy has almost no impact at this scale, since the memory can hold most or all required pages.

OVERALL CONCLUSION FOR GLOBAL ALLOCATION:

Global allocation efficiently reduces total page faults by dynamically distributing frames among processes.
As the number of frames increases, the difference between policies narrows.

4. Allocation : Local
Number of frames : 4
Trace file : combined.trace





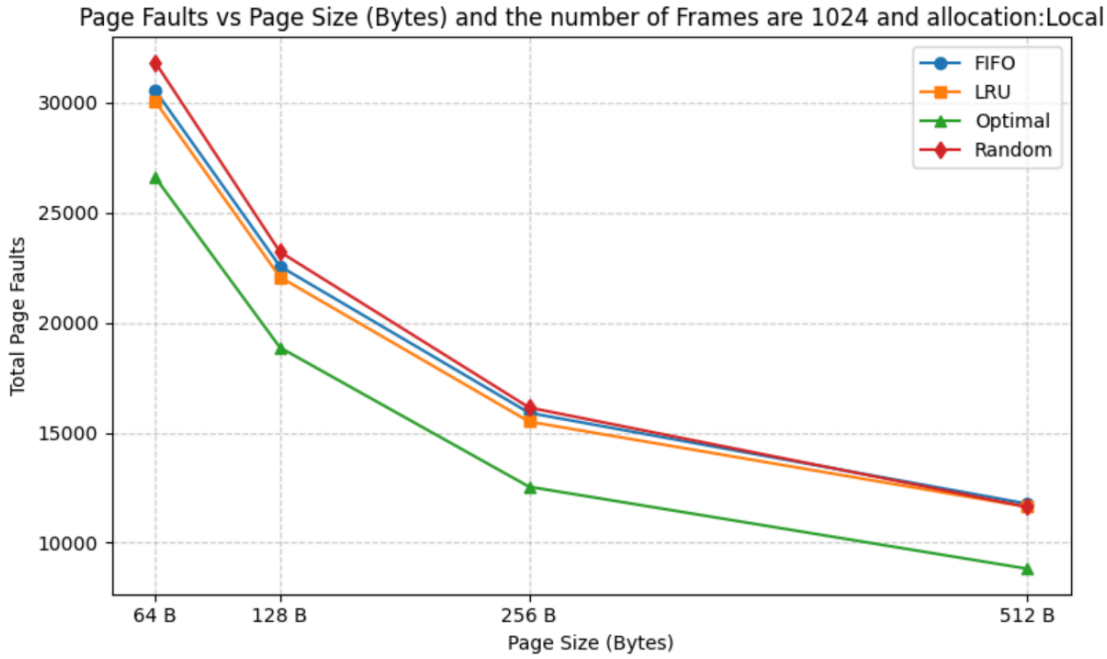
Conclusion:

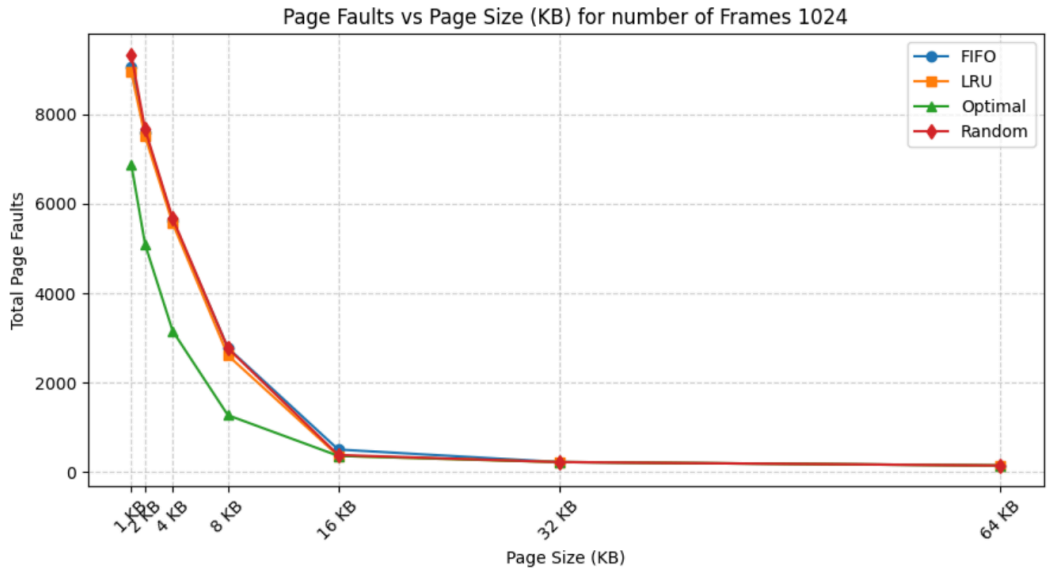
When frames are very limited and allocated locally (equally among processes), each process suffers frequent page faults.

Unlike global allocation, frames cannot be borrowed from other processes, so the total fault count remains high.

5. Allocation : Local

Number of frames : 1024
Trace file : combined.trace



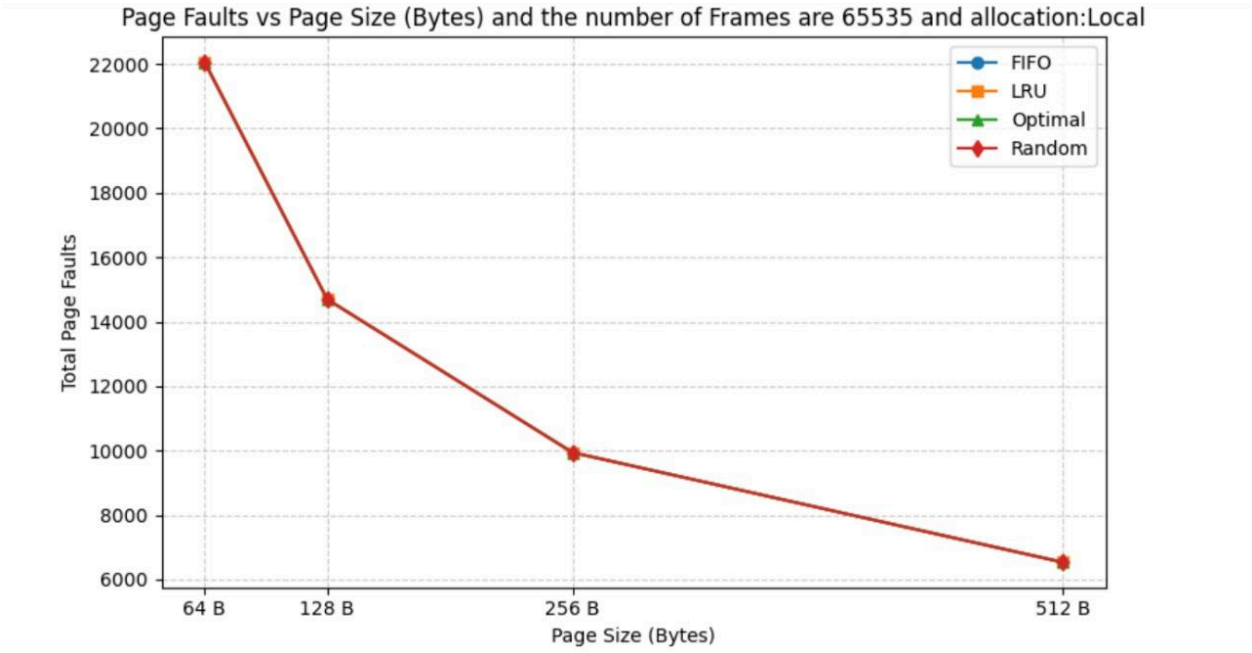


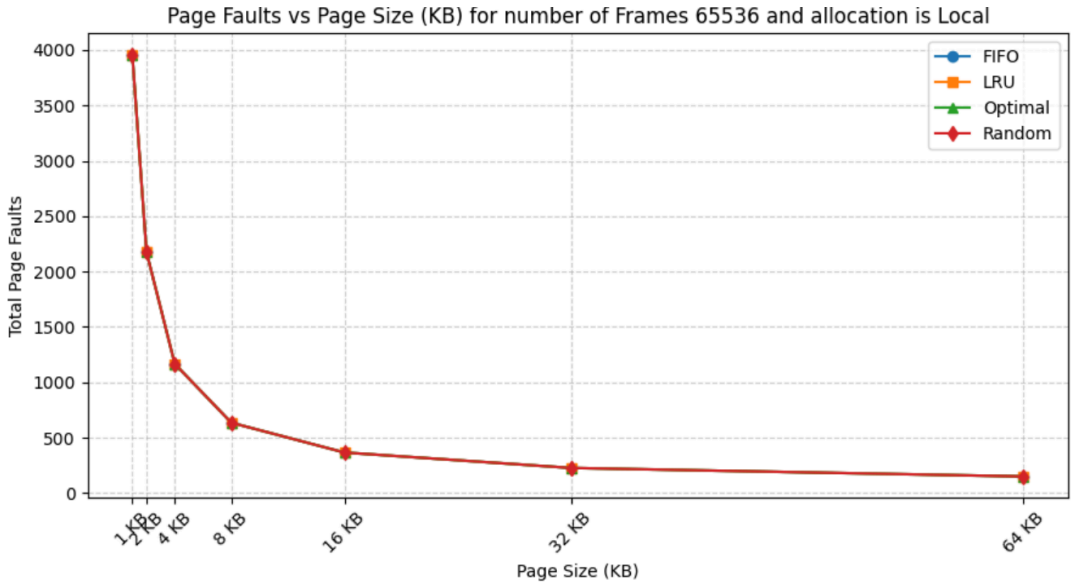
Conclusion:

As frames increase, page faults reduce for all algorithms, but Local allocation gives slightly higher total faults than Global allocation because it cannot reassign unused frames from inactive processes.

LRU and Optimal again show better performance than FIFO and Random.

6. Allocation : Local
Number of frames : 64k(65536)
Trace file : combined.trace

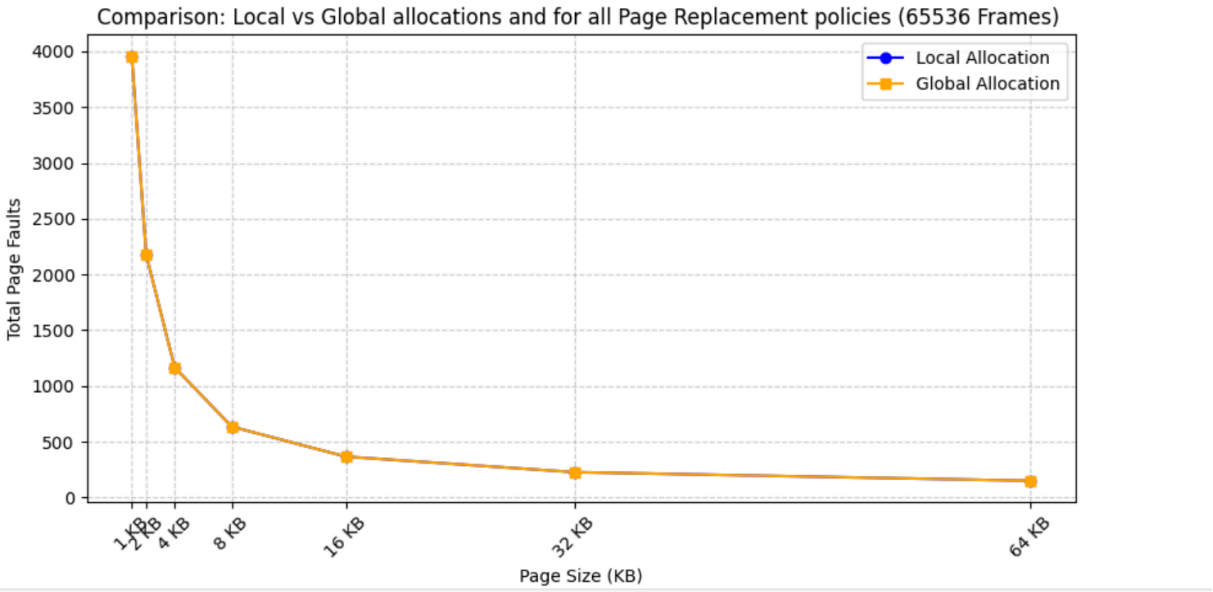




Conclusion:
At large frame counts, Local and Global allocations perform almost identically since every process gets enough frames to store its working set.
OVERALL CONCLUSION FOR LOCAL ALLOCATION:
Local allocation ensures fairness among processes but results in slightly higher total page faults compared to Global allocation.
The difference diminishes as the number of frames increases.
Optimal and LRU remain the best-performing policies.

GRAPH FOR GLOBAL VS LOCAL ALLOCATIONS FOR combined.trace :

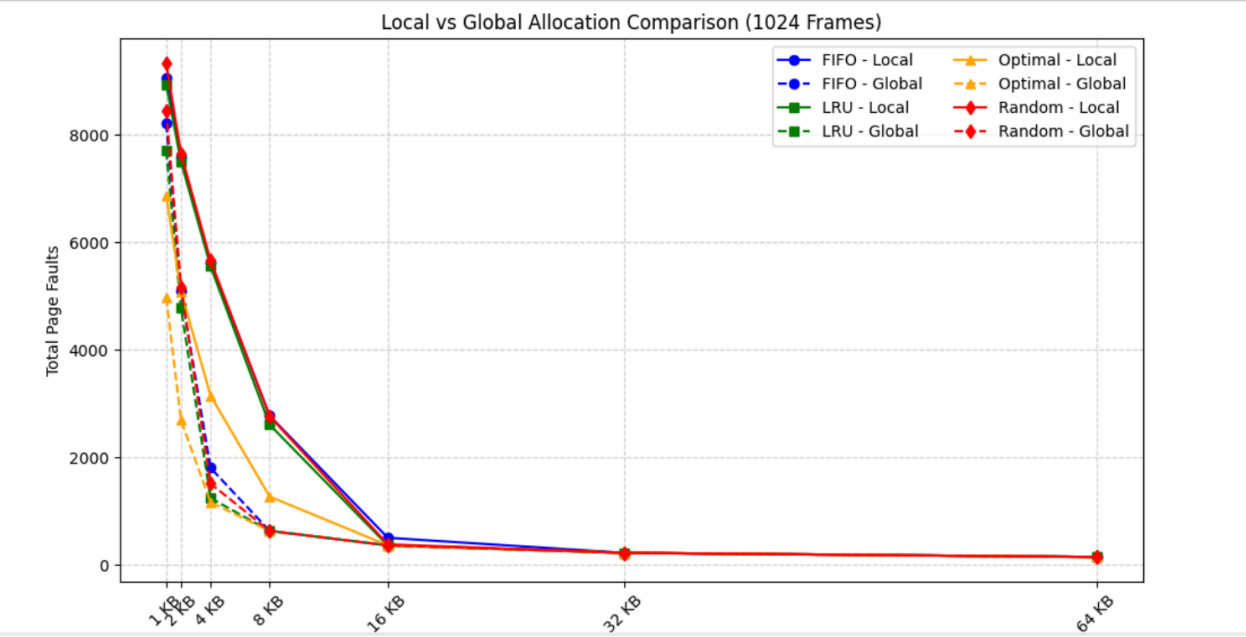
1. Number of frames = 65536



Conclusion:
At high memory capacity, both Global and Local allocations show nearly identical results across all algorithms.

Each process can retain its entire working set in memory, so page replacement becomes less frequent.

2.number of frames = 1024



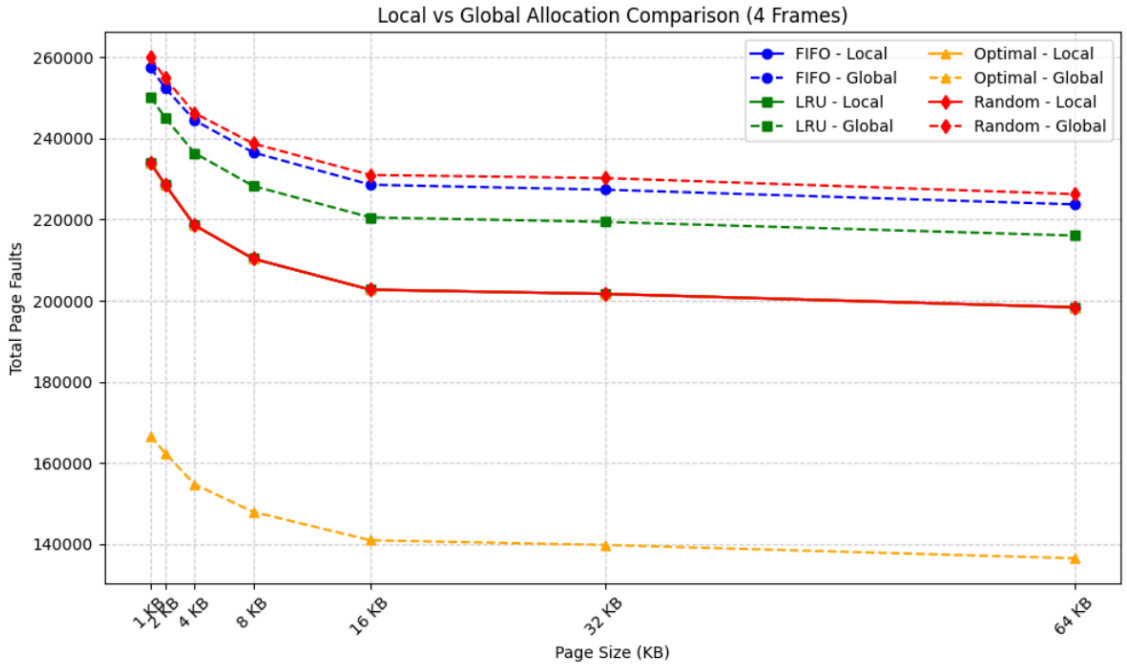
Conclusion:

The Global allocation shows fewer total page faults compared to Local because it can dynamically adjust frame usage across processes.

Local allocation, while fairer, limits frame sharing, causing a slightly higher fault count.

The trend is most visible for FIFO and Random.

3.number of frames = 4



Conclusion:

At very low memory capacity, both Global and Local allocations experience extremely high fault rates.

Global still performs slightly better, as it can prioritize active processes.

OVERALL CONCLUSION:

Across all experiments, Optimal consistently achieves the fewest page faults, providing a benchmark for comparison.

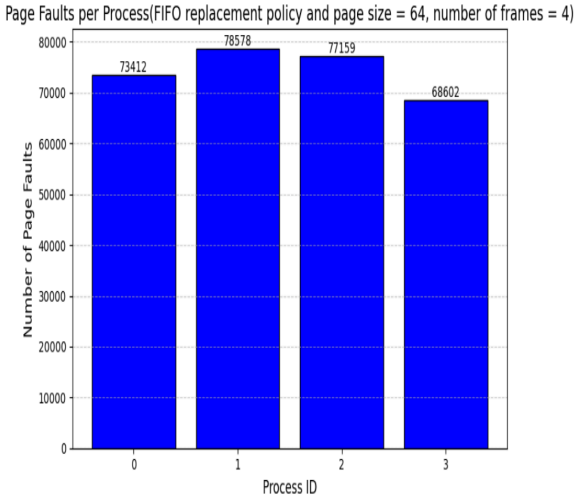
LRU performs closest to Optimal, effectively exploiting temporal locality.

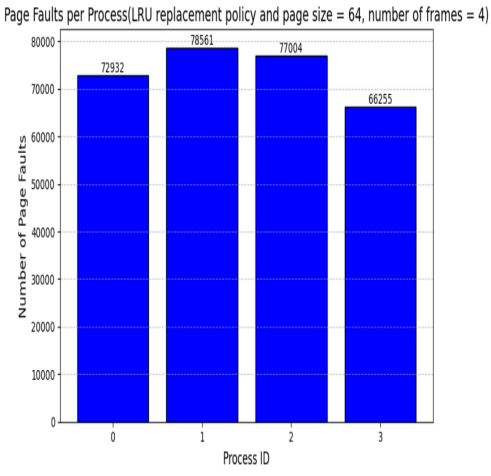
FIFO and Random show higher fault rates due to lack of recency awareness.

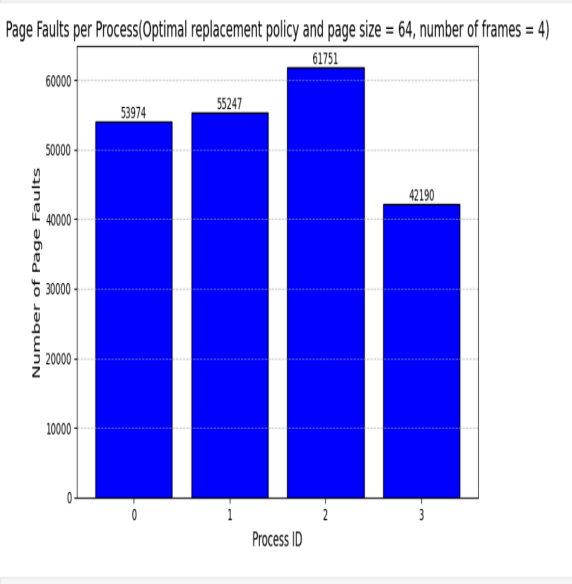
Global allocation minimizes total faults through flexible frame sharing, while Local allocation offers fair but slightly less efficient performance.
Increasing either page size or number of frames reduces total page faults significantly.

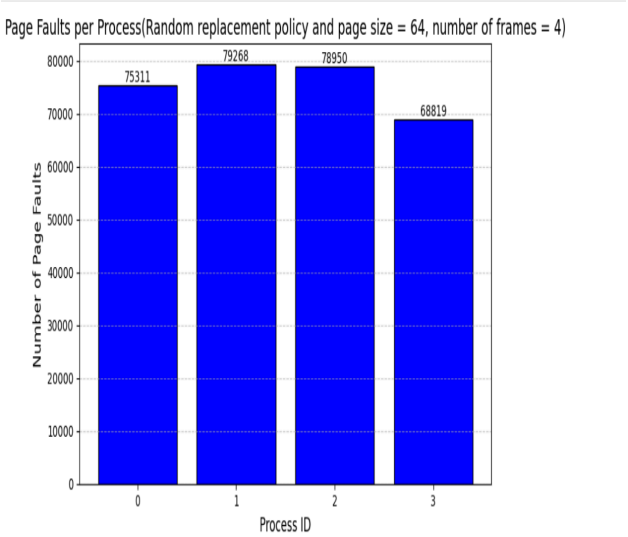
FOR combined_short.trace:
The page fault results are the same for all replacement policies (FIFO, LRU, Optimal) and both allocation policies (global and local) because the memory trace is very small and contains very few distinct pages per process. Even the smallest number of frames used is enough to hold all the pages referenced, so no page replacements occur. Changing page sizes or frame counts in this scenario does not create enough distinct pages to trigger differences between policies.

PER PROCESS COMPARISONS FOR combined.trace file(Global):









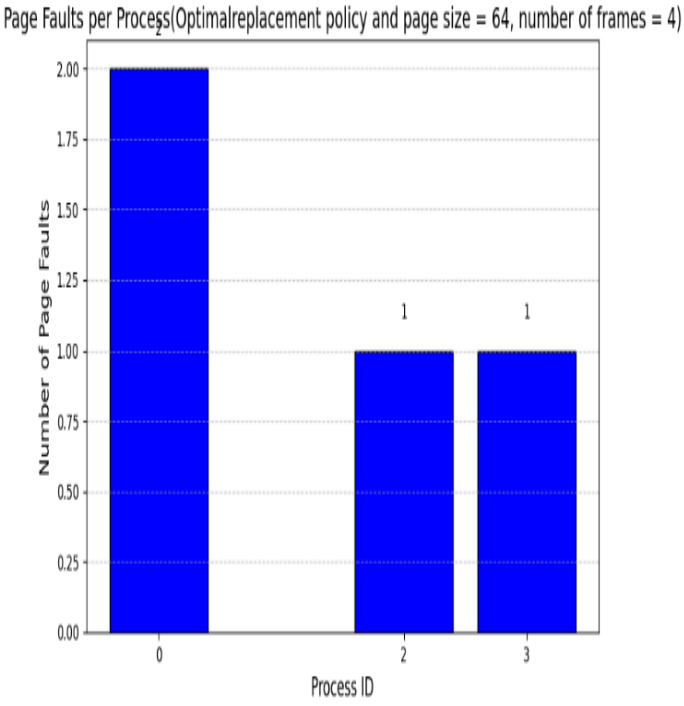
Conclusion:

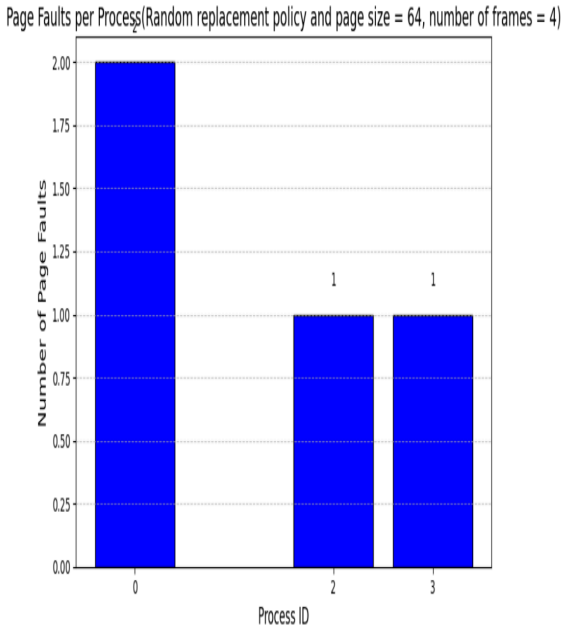
Processes 0,1 and 2 generally have **more number of page faults** than process 3.

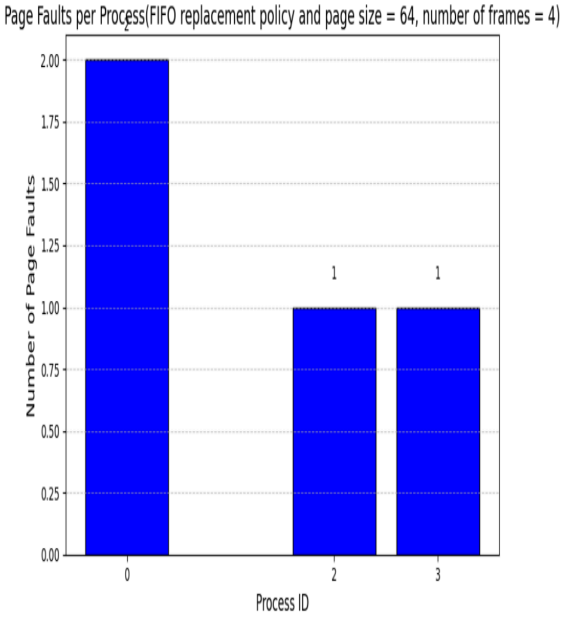
This may indicate that process 3 has **smaller working set size** or better **spatial locality**.

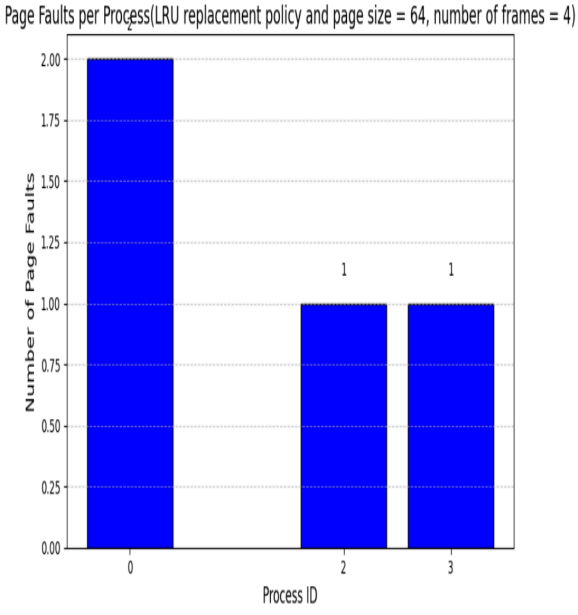
If the trace file mixes memory references from multiple programs, this shows that **some workloads are more memory-efficient** than others.

PER PROCESS COMPARISONS FOR combined_short.trace(Global and local)









Conclusion:

As this trace file contains only a few memory accesses for each process. Because of this, the working set of each process fits easily within the assigned frames, making the allocation strategy have little to no effect…tha means the local and global allocations almost have the same result.