Design Document: Contiguous Frame Pool Implementation

Files Modified:

- cont_frame_pool.h
- cont_frame_pool.cpp

No other files were modified for this implementation.

cont_frame_pool.h

- A singly linked list is used to store all the kernel frame pools
- **static ContFramePool *head**: Used to store the head of the linked list containing the list of frame pools.
- **ContFramePool *next:** Used to store the address of the next frame pool
- **unsigned char * bitmap:** A character array is used to store the status of the frames. One character will be able to status of 4 frames (since 2 bits are used for a frame and a character contains 8 bits)
- All other initialized variables are self-explanatory.

cont_frame_pool.C

<u>get_state and set_state</u>
The bitmap uses 2 bits per frame to store states. Each byte stores 4 frame states.

- Free: 00
- Used: 01
- HoS: 11

set_state is used to set the state of the frame according to the above configuration, and get_state returns the state of the given frame.

<u>Constructor implementation</u>
This function first adds the current object to the linked list containing the addresses of the frame pools. Then, it initializes the class variables, followed by initializing the bitmap based on the _info_frame_no. If info_frame_no is zero, it uses the first frame to store the bitmap. It then sets the status of all the frames in the frame pool to FREE and the status of the information frame to HoS (since the information frame should be used to store process data, it can't be in FREE state).

get_frames
This function implements a search algorithm to find a contiguous block of free frames. Starting from the beginning of the frame pool, it iterates over the frames, counting contiguous free frames until it reaches the requested size (_n_frames). Once a suitable block is found, it iterates over this block again to update the state of each frame, marking the first frame differently if needed to indicate the start of a block. The search and update mechanisms are carefully designed to handle edge cases, such as reaching the end of the pool without finding a sufficient block.

release_frames
To deallocate a block of frames, release_frames iterates through the linked list to find the frame pool containing the given base frame number. It then iterates through the frames until a Free frame or HoS is found, setting their state back to free.

mark_inaccesible
The function calculates the index for each frame in the state management structure and directly modifies the state to mark the frame as inaccessible(the first frame being set as HoS and the following frames being set as Used), using a loop to apply this state change over the specified range.

needed_info_frames
needed_info_frames calculates how much space is needed to store the frame pool's management information by considering the total number of frames and the storage efficiency of the chosen state management structure (e.g., a bitmap). In our implementation, Each byte can store four frames. Thus, we need _n_frames/(4*FRAME_SIZE) info frames.