Table 2: What is weird about FFTW? What type of access pattern must it be doing?

FFTM benefits significantly from 4 MB superpages, but not from 64 KB and 512 KB superpages.

FFTW must have a strided access pattern.

Because large superpages will benefit most when large contiguous blocks of memory are accessed in order.
There are few page table entries, less TLB misses, and less cache misses.
So FFTW must have a sequential access pattern or a strided access pattern.

Also, for 64 KB and 512 KB superpages, FFTW doesn't benefit much from the larger page size.
The performance is almost the same as the base system. However, for 4 MB superpages, FFTW shows significant improvement. If FFTW had a sequential access pattern, we should expect the performance gains to be more gradual. This suggests that FFTW has a strided access pattern, and it is possible that the stride size is greater than 512 KB.

An implicit but overriding principle of the superpage paper is primum non nocere ("first, do no harm") in that they try to never be worse than the base system. Give two examples of choices they made that satisfy this principle and one example of a choice that does not.

Satisfy "first, do no harm":

1. Incremental promotions. We only promote regions to superpages that are fully populated by the application.
   This is a conservative approach to promotion because only when the population count reaches the next larger superpage size, the system promotes the region to the next size.
   If it continues to be populated, another promotion occurs to the next size, and so on.
   This avoids unnecessary promotion of pages, and thus avoids causing harm to the system.

2. Fragmentation control. When failing to allocate a contiguous region of the preferred size, the deamon will attempt to traverse the inactive list and move those pages that contribute to the goal to the cache.
   Thus, the fragmentation will be coalesced and the next allocation request will be satisfied, while the pages are still in the cache.
   There should be no harm to the system.

Not:

1. Multi-list reservation scheme, buddy allocator. This technique involves maintaining separate lists of reservations for different superpage sizes, preempting the least recently allocated reservation. It makes

makes preemption simple and reduces possible contention. However, mesa shows a small performance degradation because buddy allocator does not differentiate zeroed-out pages from other free pages. When the OS allocates a page that needs to be subsequently zeroed out, it requests the memory allocator to preferentially allocate an already zeroed-out page if possible.