you need to
load the file into memory, just choose a small file, for example, any
file whose size is like 1M. Given the fact page size is 4K, that would
be 256 pages:

Pseudocode:

```
Global Var: file
1. Procedure test_virtualization()
2. load_file_once_into_memory (file);
3. time1 ← clock gettime();  // record the clock time before we write to each page of the file
4. for each page of file in memory do
5.     write to that page;
6. time2 ← clock gettime();  // record the clock time after we write to each page of the file
7. t1 ← diff(time1,time2);
8. load_file_twice_into_memory (file);
9. sleep (NUM OF SECONDS); // sleep and hope the two copies will be merged
10. time1 ← clock gettime(); // record the clock time before we write to. each page of the file
11. foreach page of file in memory do
12.     write to that page;
13. time2 ← clock gettime(); // record the clock time after we write to each page of the file
14. t2 ← diff(time1,time2);
15. ratio ← t2/t1;
16. if ratio > KSM_THRESHOLD then
17.     return 1;
18. else
19.     return 0;
```

Once you have t1 and t2 (for each page), you can draw a picture and
show the significant gap between t1 and t2. Or you can compute the
ratio like the above pseudo code does, and make the decision based on
any selected threshold, it's up to you.

And this is an example code snippet calculating t1 - you can write
anything to each page, in this example, just write a 'b' to each page.
The file being loaded is called "httpd".

```
char *content0;
double t1,t2;
struct timespec time1, time2;

/*Load file once and record the write access time*/
size1 = load_file_to_memory("httpd", &content0);
if (size1 < 0)
{
    DPRINTF("Error loading file\n");
    free(content0);
    return 0;
}
pages=size1/4096;
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time1);
do{
    content0[pages*4096-1]='b';
    pages--;
}
while(pages > 0);

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time2);
```

```
t1=diff(time1,time2).tv_nsec;
printf("t1=%f\n",t1);
```