CSC443 PL1 report

Mihai Nicolae, g1mihai, 998584367
Qingsong Meng, g1dameng, 998383093

2.3. Experiments

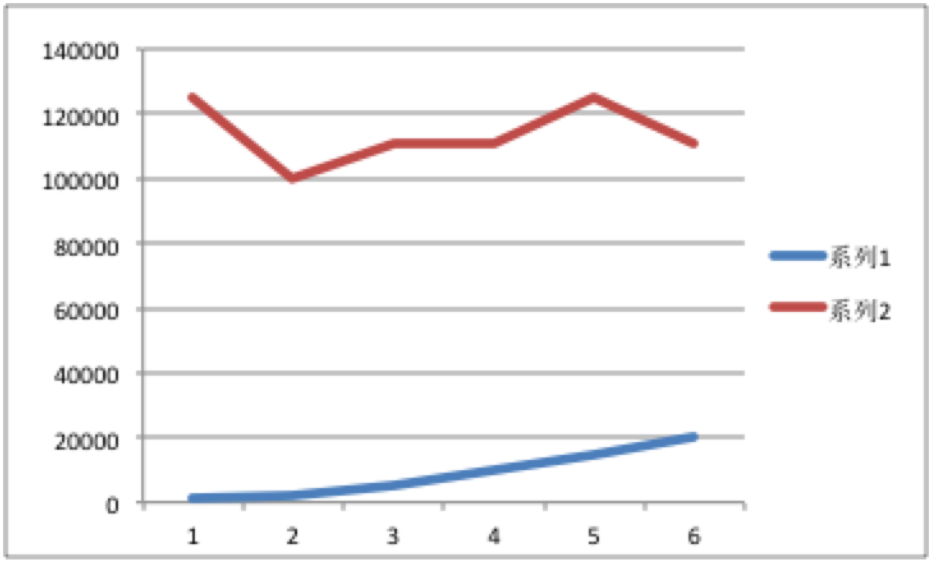
We assume that there is only *one* table schema. There are 100 attributes, and each attribute is 10 bytes each. So, records in the table are fixed length.

- Calculate the size of fixed length serialization of records in the table.
    - The size of fixed length serialization is simply the number of attributes (100) multiplied by the size of an attribute (10). That is 1000 bytes.
- Use fixed_len_sizeof() to check if it agrees with your calculation.
    - Yes, this does agree with our calculation.

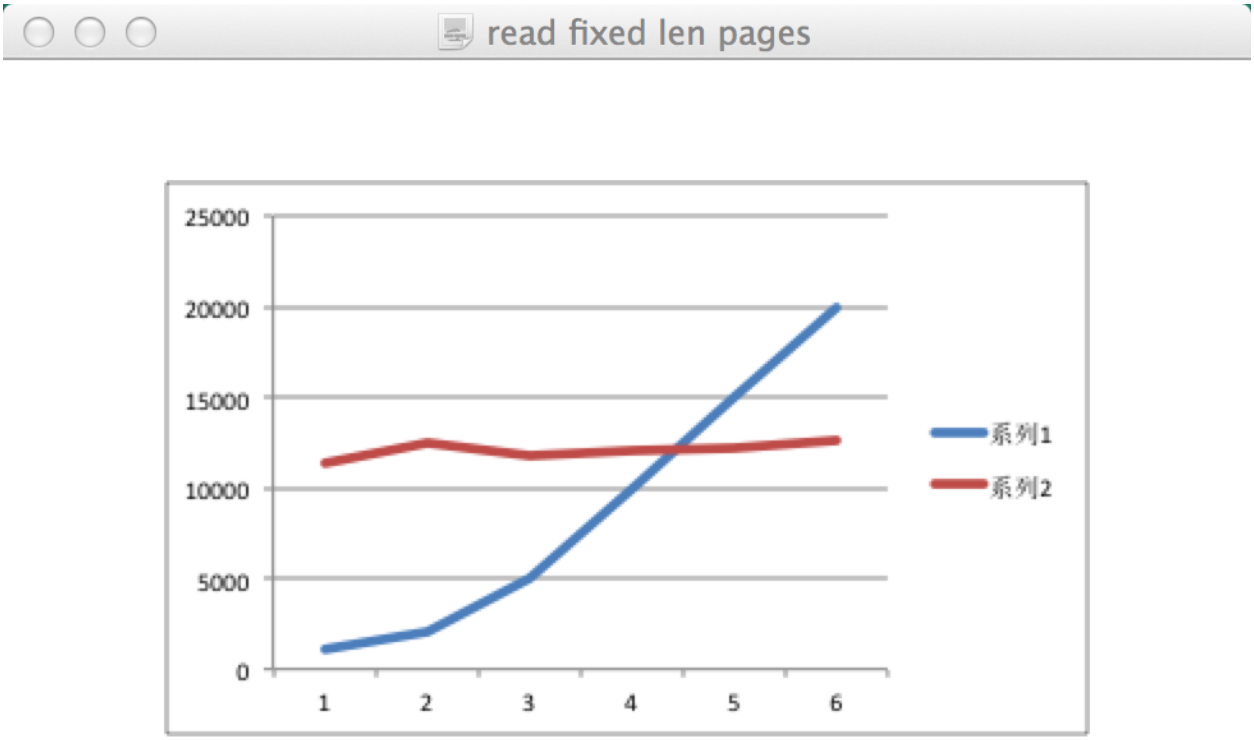
3.2. Experiment

- Plot the performance (records / second) versus page size for write and read.
    - We noticed that the performance decreases as the page size decreases.
- Compare this to the blocked disk I/O characteristics you observed in the tutorial
- Discuss why page based format is superior to storing records using a CSV file.
    - Page-based format is better than storing records in a CSV file. CSV is an expensive serialization format in both storage and performance.
    - Firstly, the CSV file can only be traversed linearly. Secondly, there is no method of seeking to an absolute position because there is no guarantee on the fixed length of the records.
- Discuss the shortcomings of the way we organize pages.
    - For insertion, we find the available (meaning it has enough space to store a record) page and insert there. Data is not sorted on pages, so a page has to be traversed fully to find the desired record.

write fixed len pages



Clipping contents: TIFF image



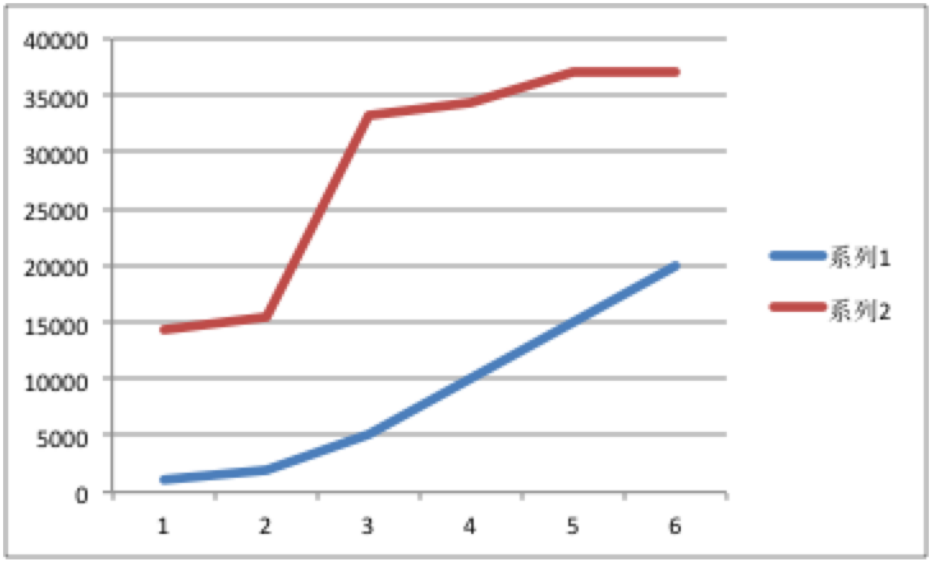
4.3. Experiment

Measure the performance of csv2heapfile, comment on how the page size affects the performance of load.

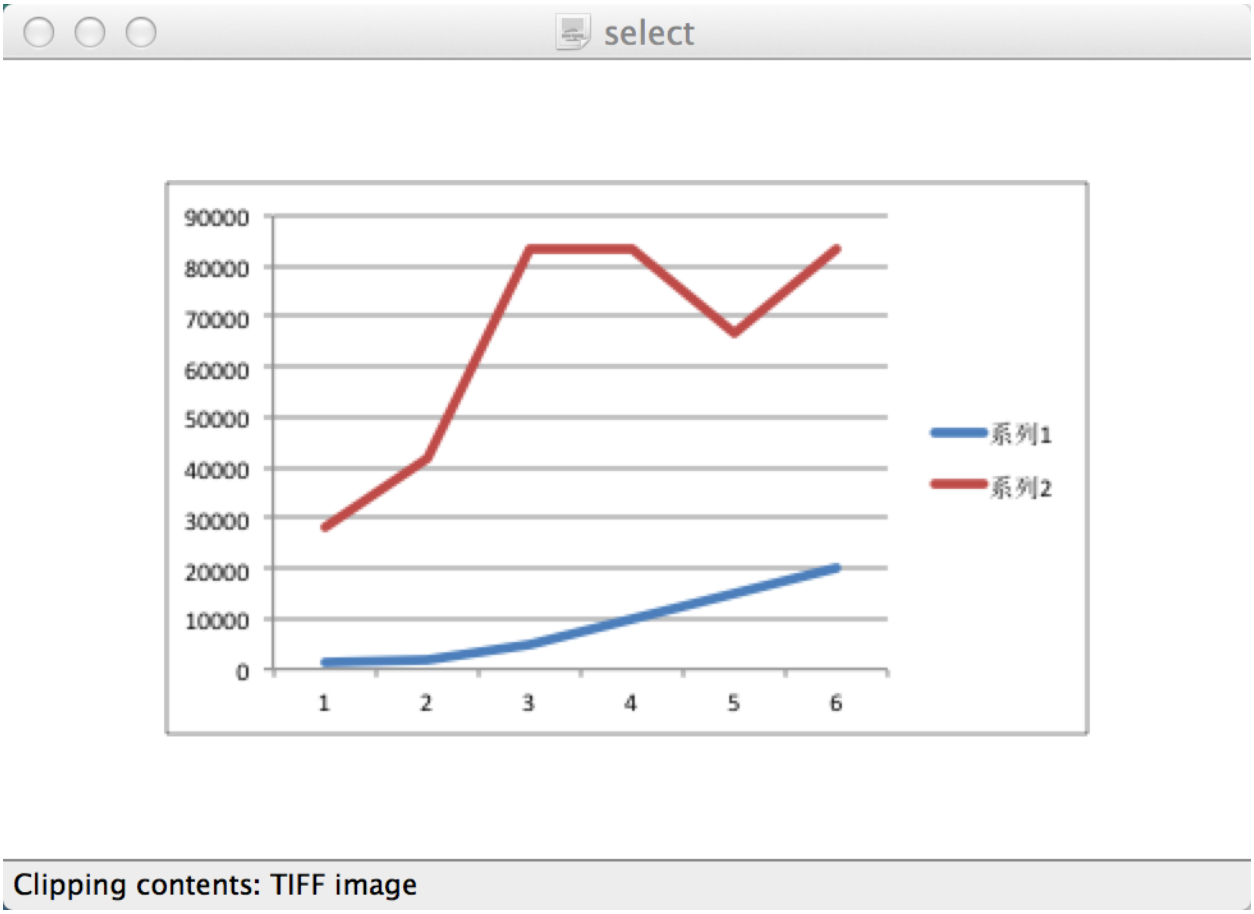
There is a negligible performance drawback on csv2heapfile for smaller page sizes. We observed a performance of 5 miliseconds for a page size of 1100 and a performance of 4 miliseconds for a page size of 11000.

- Measure the performance of the query versus page size.
    - Similarly, there is a negligible performance impact on the query given the page size.
- Comment on the choice of page size and the effects of the range from *start* and *end* on the performance of the query.
    - The data on the heap file is not sorted so we will do a full file scan regardless of the start, end values. We already observed that page size does not affect the performance of the query.

csv 2 heapfile



Clipping contents: TIFF image

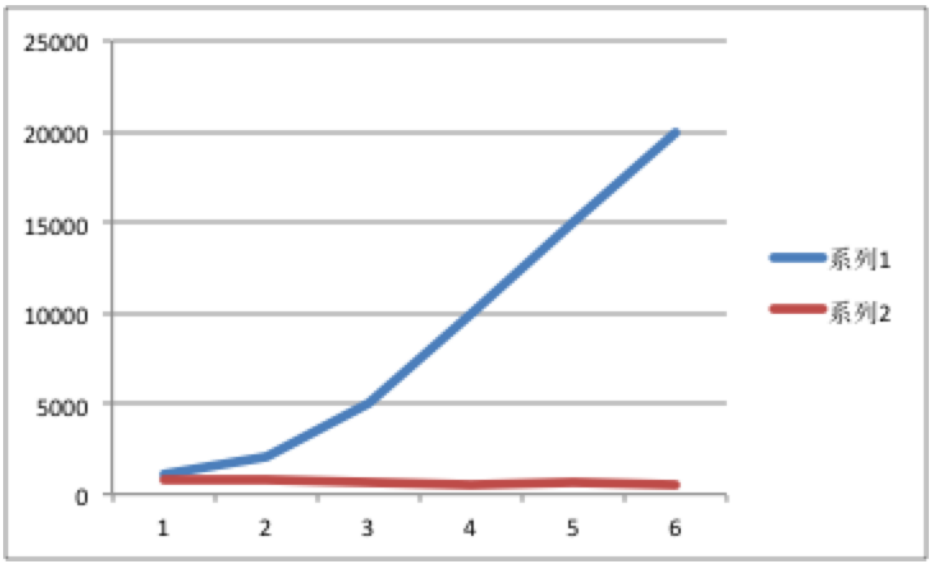


5.2. Experiment

- Measure the performance of csv2colstore against different page sizes.
    - The performance of csv2colstore is constant with respect to different page sizes.
- Compare the result with that of csv2heapfile in the previous section. Comment on the difference.
    - Compared to csv2heapfile, csv2colstore is about 100 times slower. That is expected, as we are performing I/Os on 100 times more heapfiles.
- Compare the performance of select2 with that of select in the previous section. Comment on the difference.
    - The performance of select2 is about 10 times better than that of select. This is expected, as for select we perform a full scan of the heap file, whereas in select2 we perform a full scan of a heap file that is 50 times smaller than the heap file of select. The reason why the heap file of select2 is 50 times smaller than the heap file of select is because we are storing the tuple-id for each attribute value, and we allocated 10 bytes for each tuple-id.

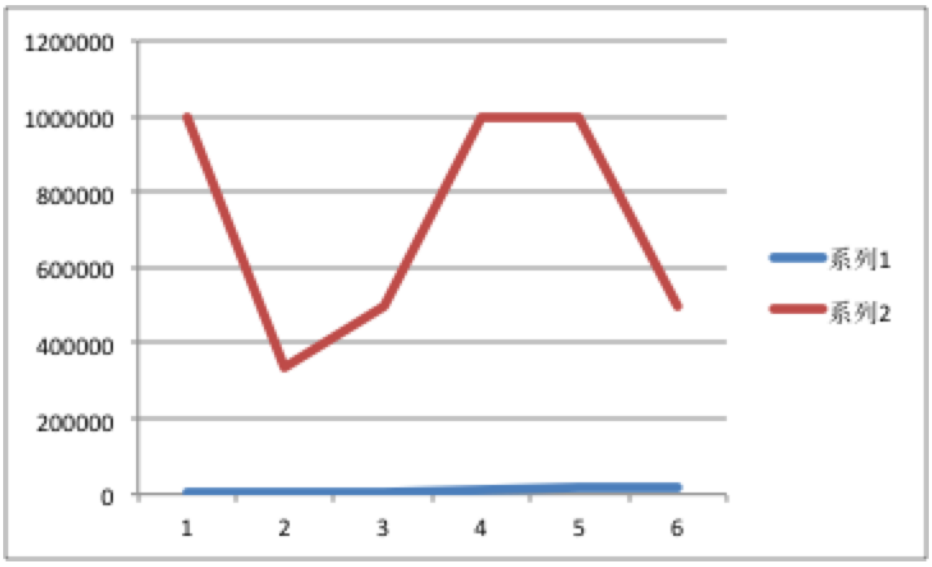
- Compare the performance of select3 with that of select and select2. Comment on the difference especially with respect to different selection ranges (different values of start and end).
    - We did not implement select3, so we cannot comment on its performance.

csv2colstore



Clipping contents: TIFF image

select2



Clipping contents: TIFF image