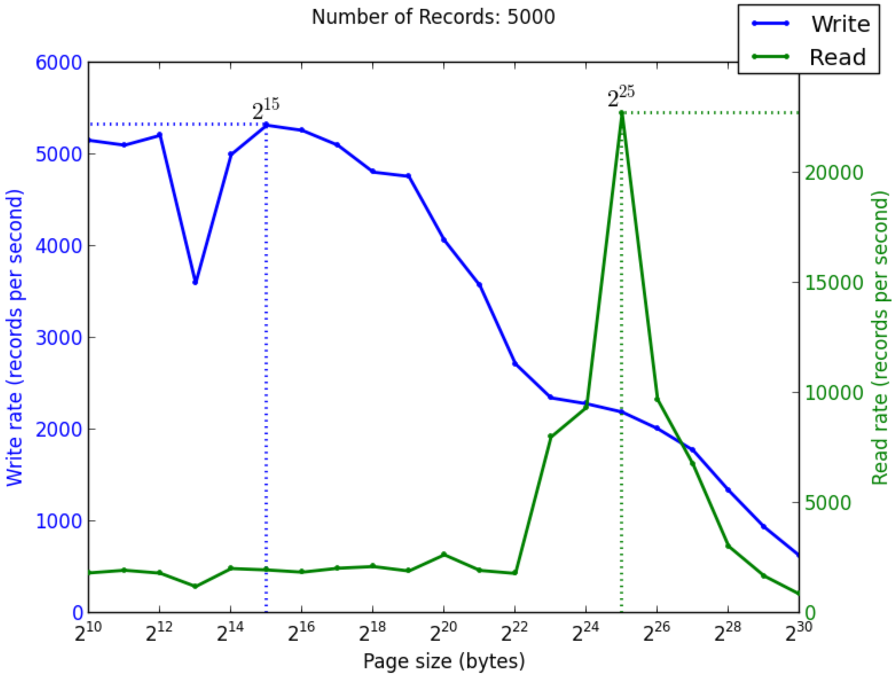
CSC443 Practical Learning 2

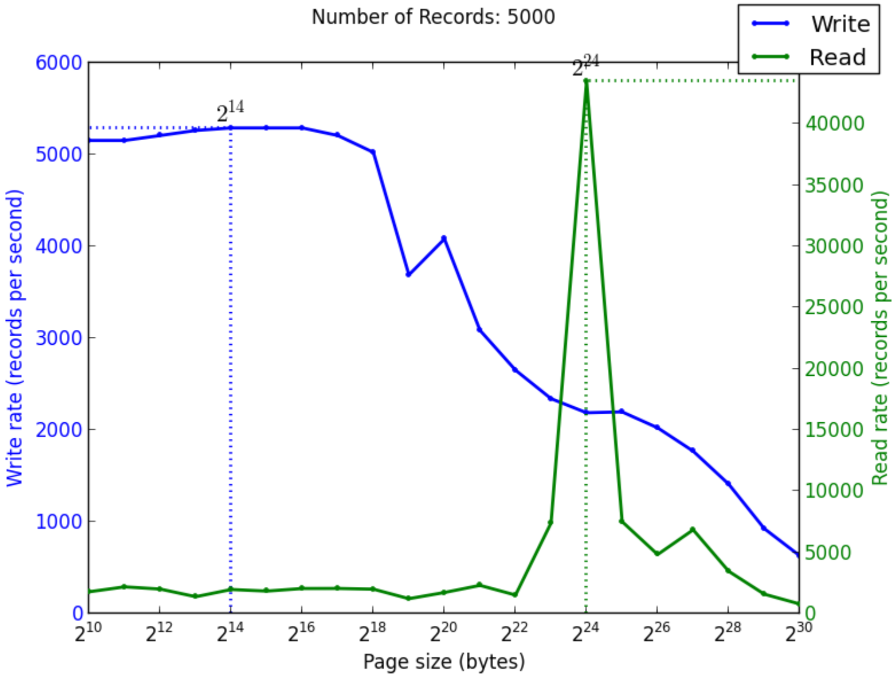
Abhinav Gupta, Zeeshan Qureshi

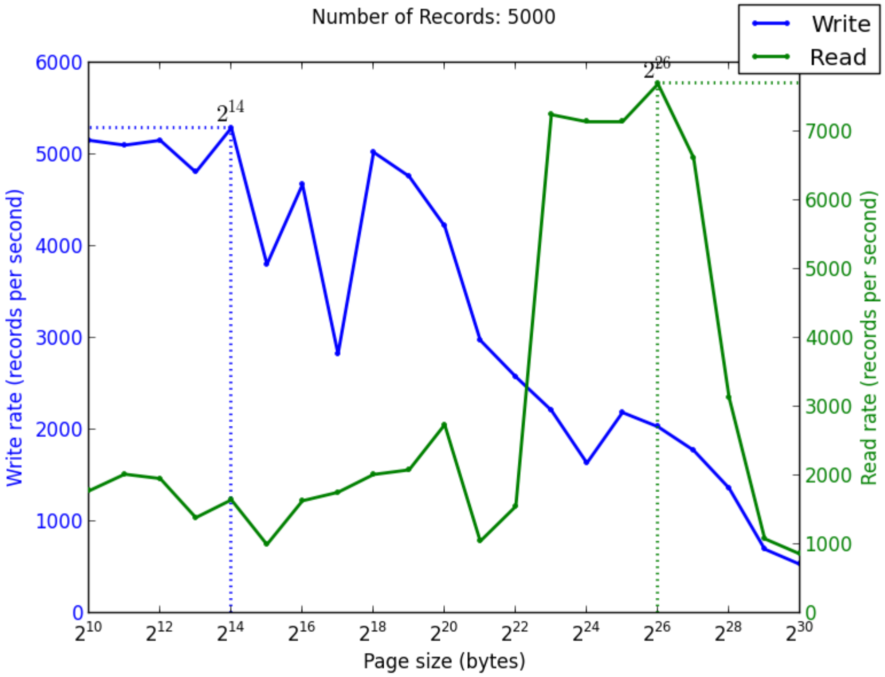
23 Feb 2013

Page Layout

We used `mkcsv.py` to generate a CSV file with 5,000 entries. We then used `write_fixed_len_pages` and `read_fixed_len_pages` to store and retrieve the data into heap files. After running the experiment three times on the data, we got the following results:


Number of Records: 5000
Write
Read
2^{15}
2^{25}
Write rate (records per second)
Read rate (records per second)
0
1000
2000
3000
4000
5000
6000
0
5000
10000
15000
20000
2^{10}
2^{12}
2^{14}
2^{16}
2^{18}
2^{20}
2^{22}
2^{24}
2^{26}
2^{28}
2^{30}
Page size (bytes)


Number of Records: 5000
Write
Read
2^{14}
2^{24}
Write rate (records per second)
Read rate (records per second)
0
1000
2000
3000
4000
5000
6000
0
5000
10000
15000
20000
25000
30000
35000
40000
2^{10}
2^{12}
2^{14}
2^{16}
2^{18}
2^{20}
2^{22}
2^{24}
2^{26}
2^{28}
2^{30}
Page size (bytes)



Comparison to PL1

All three runs exhibited similar behavior: the best write rate was around page size 2^{14} bytes (16 kB) and the best read rate was around page size 2^{25} (32 MB). Note that The experiments performed on block sizes in the last PL assignment exhibited similar performance characteristics.

In all cases, whether it was write or read, performance took a massive drop by the time it got to 2^{30} bytes (1 GB). This is expected because the system is reading from or writing to a 1 GB block of memory for reads and writes pertaining to a record that consumes only 1,000 bytes, so resources spent accessing all the extra memory are wasted.

Variable Length Encoding

Even if variable length encoding was used for records, the overall performance behaviour of the system as page size is increased will be the same. The only difference would be that records will consume slightly more memory and so the page will contain slightly fewer records.

Versus CSV

Page-based format is obviously superior to CSV. CSV is an expensive serialization format in both, storage and performance.

For one, there is no way to traverse a CSV file but linearly. Since records are not guaranteed to be fixed length, there's no way of seeking to an absolute position within the CSV file, and so the whole file has to be traversed to get to the record that was required. Even if the CSV was sorted, there would still be no way to seek directly to specific records in the file.

Shortcomings

The way we currently organize pages is on a first-empty-slot basis. There is no organization of the data inside the pages and so a page has to be traversed completely to find a record whose offset within the page is not known.

Secondly, the pages themselves are thrown into a heap file with no ordering among them. This means that to search the file for a specific record, all records of all pages will have to be checked linearly.

Heap File

TODO

Heap file experiment info goes here.