Shared Memory Sort

Bobo Liang A20356451

1. Brief description of the problem

Large file can't be sorted by loaded them all into memory, so in order to sort large files, so we need use divide concur algorithm. Split the large files into small pieces file, sort these chunk files, and then merge them.

2. Methodology

For 2GB file, I divide it into 16 chunk files, sort each file. Create 8 threads, allocate two input string array of size about 10MB, read part of strings from two input chunk file, And compare, merge the two input array into one output string array of size 20MB, write the output array to intermediate files, loop this until all input file has been sorted. Then merge the intermediate files again until finally one large file.

For 20GB file, I divide it into 32 chunk files. Create 16 threads, each thread use two 10MB input memory size and 20MB output memory size.

My sort algorithm is similar with linux sort algorithm, which is:

UNIX sort treats keys are lines (strings), the algorithm followed by unix sort is in fact the R-Way merge. Let the input file size be IN_SIZE.

1. Choosing Run Size:

The sizes of the initial runs are chosen from the total physical memory (TOTAL_PHY) and available memory (AVAIL_PHY). RUN_SIZE = (MAX(TOTAL_PHY/8,AVAIL_PHY))/2

maximum of 1/8th of TOTAL_PHY and AVAIL_PHY and divided by 2. See function "default_sort_size (void)" in the code.

2. Creating Runs:

Unix sort creates a temporary file for every run. So it creates IN_SIZE/RUN_SIZE (celing) temporary files. Internally it uses merge sort to sort internally it uses an optimization mentioned in Knuth volume 3 (2nd edition).

3. Merging:

The number of runs merged at any time is hard coded in the program see macro NMERGE , NMERGE is defined to be 16 so it merges exactly 16 runs at any time.

3. Performance

Table 1: Performance evaluation of TeraSort

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)
Compute Time (sec)	182	41	1500	489
Data Read (GB)	4x2=8	3x2=6	6x20=120	7x20=140
Data Write (GB)	4x2=8	3x2=6	6x20=120	7x20=140
I/O Throughput (MB/sec)	88M/sec	293M/sec	160M/sec	573M/sec

Performance analysis

My Shared Memory sort is about 4 times lower than the linux sort.

My20GB sort **can't finished within 20 minutes**, it stops at pass 4, while totally there are 32 input files need to merge into one, that is 5 passes. So approximately it will run 25 minutes to finish.

The reason I think should be the values of file chunk number, threads number, input memory size and output memory size are not very proper, they should be tuned better, I need to reduce the pass number to 4, 16 chunk file number, as the linux sort does.

4. Logs

Linsort2GB.log

start linsort2GB
real 0m25.878s
user 0m32.364s
sys 0m9.300s

Linsort20GB.log

start linsort20GB
real 6m52.585s
user 6m39.236s

sys 1m30.380s

mysort2GB.log

```
start mysort
inputfile has been splitted into 16 chunks
Thread 0 is working on pass1run0
Thread 1 is working on pass1run1
Thread 2 is working on pass1run2
Thread 3 is working on pass1run3
Thread 4 is working on pass1run4
Thread 5 is working on pass1run5
Thread 6 is working on pass1run6
Thread 7 is working on pass1run7
Thread 0 has joined
Thread 1 has joined
Thread 2 has joined
Thread 3 has joined
Thread 4 has joined
Thread 5 has joined
Thread 6 has joined
Thread 7 has joined
Thread 0 is working on pass2run0
Thread 1 is working on pass2run1
Thread 2 is working on pass2run2
Thread 3 is working on pass2run3
Thread 0 has joined
Thread 1 has joined
Thread 2 has joined
Thread 3 has joined
Thread 0 is working on pass3run0
Thread 1 is working on pass3run1
Thread 0 has joined
Thread 1 has joined
Thread 0 is working on pass4run0
Thread 0 has joined

real    1m48.955s
user    2m46.244s
sys     0m25.704s
```

start valsort
First unordered record is record 6
sump pump fatal error: pfunc_get_rec: partial record of 68 bytes found at end of input