io_thrash benchmark

1. Download the io_thrash tarball from http://sourceforge.net/projects/fis-gtm/files/Benchmarking/io_thrash/io_thrash_20130512.tgz and unpack it in a folder.
2. Compile it following the instructions at the top of the file `io_thrash.c`.
3. Run the benchmark. Files `081106-1a_io_thrash_Readme.odt` and `081106-1a_io_thrash_Readme.pdf` describe the workload and command line options.

Running the benchmark

Change to a directory on the filesystem to be benchmarked, and run `io_thrash` in a `screen` session (since elapsed times can be very long). In the actual benchmarks with the following script, `io_thrash_db` was the pattern used for files, with `io_thrash_db_0.db` being the only file created for this set of inputs. Note that each set of inputs is occurs three times.

```
#!/bin/sh
rm -f io_thrash_db_0.db
dd if=/dev/zero of=io_thrash_db_0.db bs=1024000 count=5000
/usr/bin/time -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000 999999999 8 1024 0 20000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000 999999999 8 1024 0 20000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000 999999999 8 1024 0 20000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 50000 999999999 8 1024 0 200000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 50000 999999999 8 1024 0 200000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 50000 999999999 8 1024 0 200000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 500000 999999999 8 1024 0 2000000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 500000 999999999 8 1024 0 2000000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 500000 999999999 8 1024 0 2000000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000000 999999999 8 1024 0 20000000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000000 999999999 8 1024 0 20000000 999999999 60 3 999999999
/usr/bin/time -a -o io_thrash.out -f ”%e %S %U” io_thrash 1 0 io_thrash_db 5000000 999999999 8 1024 0 20000000 999999999 60 3 999999999
```

3n+1 benchmark

Download GT.M and the threeen1fbint program

1. Download the latest version of the GT.M installer[1] from http://sourceforge.net/projects/fis-gtm/files/GT.M%20Installer/ - as of May 2013, this is v0.12 (http://sourceforge.net/projects/fis-gtm/files/GT.M%20Installer/v0.12/gtminstall).
2. Make the `gtminstall` shell script executable, and execute it as root to download and install the latest version of GT.M (`gtminstall --help` lists options and `gtminstall --verbose` provides information as it runs; the default installation directory is `/usr/lib/fis-gtm/V`<version>_<platform>, e.g., `/usr/lib/fis-gtm/V6.0-002_x86_64`): `chmod +x gtminstall ; sudo gtminstall`
3. Download the threeen1fbint program (http://sourceforge.net/projects/fis-gtm/files/Benchmarking/threeen1/threeen1fbint.tgz) and unpack to extract the source program `threeen1fbint.m`.

Initial configuration of a benchmark directory

1. Set the environment variable `gtmdir` to a directory in the filesystem to be benchmarked, e.g., `export gtmdir=/testarea1`
2. Source `gtmprofile` to assign values to environment variables, e.g, `source /usr/lib/fis-gtm/V6.0-002_x86_64/gtmprofile`
3. Copy the threeen1fbint program to the `r/` subdirectory: `cp threeen1fbint.m $gtmdir/r/`
4. Optionally, delete the default database and journal file, change the parameters as needed, and create a new database file; if journaling is to be used create the new journal files. The commands below delete the default database & journal files, replacing them with an unjournaled database file of 5,000,000 1KiB blocks, that extends as needed by 65,535 blocks, and has a shared memory buffer pool of 65,536 blocks

```
rm $gtmdir/$gtmver/g/gtm.{dat,mjl*}
gde
change -segment default -block_size=1024 -allocation=5000000 -extension_count=65535 -global_buffer_count=65536
exit
mupip create
```

Running the benchmark

Create an input file with starting and ending values of the range of numbers for which 3n+1 sequences are to be computed (the starting number is always 1), the number of concurrent worker processes and the the size of blocks of integers that each worker process should solve at a time. For the actual benchmark, the input used was:

```
$ cat threeen1fbint.in
1 100000 8 500
```

[1] Note: first check whether package fis-gtm is in the repositories of your distribution - as of mid 2013, there are active efforts to include it, starting with the Debian repositories.

```
1 100000 8 500
1 100000 8 500
1 1000000 8 5000
1 1000000 8 5000
1 1000000 8 5000
1 10000000 8 50000
1 10000000 8 50000
1 10000000 8 50000
1 100000000 8 500000
1 100000000 8 500000
1 100000000 8 500000
$
```

Note that each set of inputs is occurs three times. To run the benchmark:

1. Set the environment variable `gtmdir` to a directory in the filesystem to be benchmarked, e.g., `export gtmdir=/testarea1`
2. Source `gtmprofile` to assign values to environment variables, e.g, `source /usr/lib/fis-gtm/V6.0-002_x86_64/gtmprofile`
3. Run the benchmark in a screen session with `mumps -run threeen1fbint <threeen1fbint.in | tee threeen1fbint.out`

A sample output is:

```
$ cat threeen1fbint.out
1 100,000 8 500
 350 1,570,824,736 1 218,005 318,005 218,005 318,005
1 100,000 8 500
 350 1,570,824,736 0 217,995 317,995
1 100,000 8 500
 350 1,570,824,736 1 217,670 317,670 217,670 317,670
1 1,000,000 8 5,000
 524 56,991,483,520 5 2,169,652 3,169,652 433,930 633,930
1 1,000,000 8 5,000
 524 56,991,483,520 4 2,169,963 3,169,963 542,491 792,491
1 1,000,000 8 5,000
 524 56,991,483,520 5 2,170,137 3,170,137 434,027 634,027
1 10,000,000 8 50,000
 685 60,342,610,919,632 108 21,731,573 31,731,573 201,218 293,811
1 10,000,000 8 50,000
 685 60,342,610,919,632 83 21,732,809 31,732,809 261,841 382,323
1 10,000,000 8 50,000
 685 60,342,610,919,632 74 21,733,130 31,733,130 293,691 428,826
1 100,000,000 8 500,000
 949 2,185,143,829,170,100 11,622 217,292,108 317,292,108 18,697 27,301
1 100,000,000 8 500,000
 949 2,185,143,829,170,100 8,695 217,294,775 317,294,775 24,991 36,492
```

```
1 100,000,000 8 500,000
 949 2,185,143,829,170,100 8,458 217,292,076 317,292,076 25,691 37,514
$
```

Where the first line of each pair of lines is the input, and the second line consists of:

1. the longest 3n+1 sequence encountered, e.g., 949
2. the largest integer encountered in any 3n+1 sequence, e.g., 2,185,143,829,170,100
3. the number of elapsed seconds for the run, e.g., 8,458
4. the total number of database updates, e.g., 217,292,076
5. the total number of database reads, e.g., 317,292,076
6. the database update rate, e.g., 25,691 updates/second
7. the database read rate, e.g., 37,514 reads/second