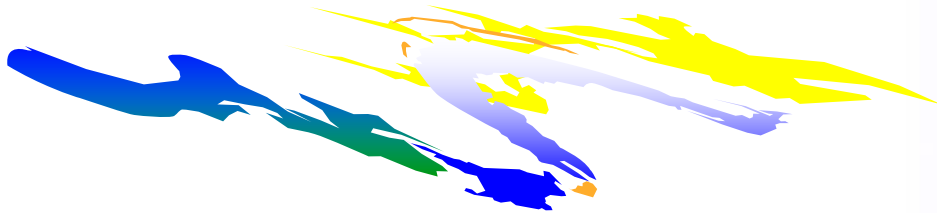




IBM Deep Computing Team

Modular I/O



John Bauer

Deep Computing, IBM (bauerj@us.ibm.com)

Overview

- What is MIO?
- How does it work?
- How to use MIO?
- Examples

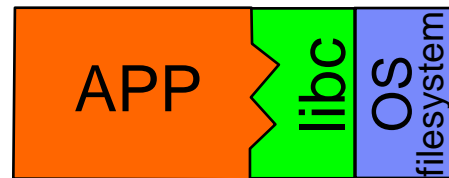
What is MIO?

- **Automatic I/O instrumentation**
 - Uses LD_PRELOAD to “get under the covers”
 - No recompiling
 - No relinking
 - Available on x86 Linux, AIX, Linux on Power

- **Key components**
 - libTKIO
 - libMIO
 - DataView

- **Output**
 - mio statistics file for each program invoked
 - trace module statistics for every file opened
 - events files for analysis with DataView

What is MIO?

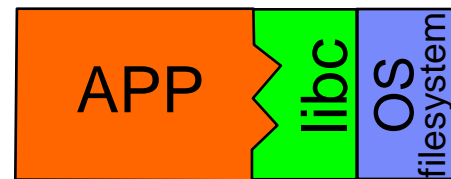


LD_PRELOAD

- Affects all subsequent programs invoked
- `export LD_PRELOAD=libTKIO.so`
- Entry points in libTKIO.so are used in lieu of link specification



What is MIO?



libTKIO

- **contains shared object entry points for the major POSIX I/O**
 - open open64 unlink truncate truncate64 access
 - stat stat64 lstat lstat64
 - lseek lseek64
 - read pread pread64
 - write pwrite pwrite64
 - close fcntl sync ftruncate ftruncate64
- **Should be transparent to applications**
 - The cost is one extra subroutine call per I/O event
- **Allows runtime selection of I/O function replacements**

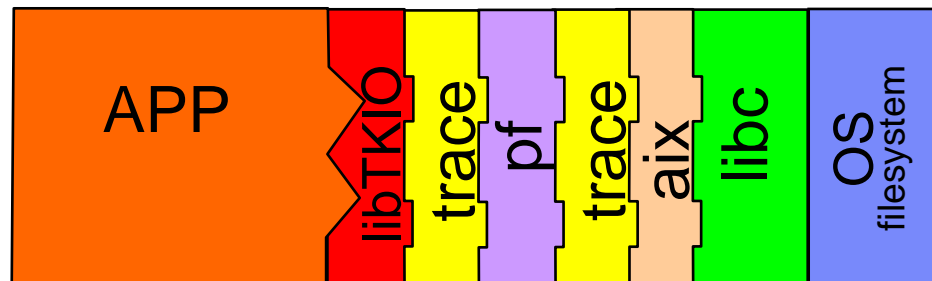


What is MIO?



libMIO

- **Collection of independent I/O modules**
 - Each module obeys the “laws of I/O”
 - A given module is not aware of the existence of other modules
 - Analogous to piping unix commands, but data flow is 2-way
- **Key modules**
 - trace –instrumentation
 - pf –user space caching and prefetching
 - async –handles aio requests with pthreads
 - aix – interface back to libc calls.



How does MIO work?

trace

trace module

- summary of file activity
- binary events file
- low cpu overhead
- typical options
 - /stats
 - /bytes /kbytes /mbytes /gbytes /tbytes
 - /events=*mio.evt*

How does MIO work?

pf

pf module

- detects sequential I/O and prefetches data
- aggregates smaller requests into page size requests
- user memory buffering
- Commonly used options
 - /global=*any_name*
 - /cache_size=1g
 - /page_size=1m
 - /prefetch=1
 - /stride=1
 - /direct
 - /stats

How does MIO work?

async

async module

- uses pthreads to perform asynchronous I/O
- minimal overhead
- Commonly used options
 - /nthread=1
 - /nchild=1
 - /naio=40

How does MIO work?



aix

aix module (soon to be named posix)

- interface to POSIX in libc
- allows bit manipulation
- Commonly used options
 - /sector_size=
 - /alt_name=
 - /notrunc

How to use MIO?

Using miostats

- **% miostats a.out arg1 arg2 arg3 ...**
 - Will generate a stats file **a.out.0.stats** in the current directory
- **% miostats gmake simple.exe**
 - **as.0.stats cc1.0.stats collect2.0.stats gcc.0.stats**
gmake.0.stats ld.0.stats strip.0.stats

How to use MIO?

miostats options

- **-v**
 - verbose, prints to stderr the environment variables used to control TKIO and MIO

- **-p prg1[:prg2:prg3...]**
 - A colon separated list of programs that **TKIO** should use **MIO** with
 - Default is "*"
 - Wildcards are permitted (ensure you use quotes to protect from expansion by shell)
 - -p "analysis.*:/home/bauerj/bin/*"
 - -p "/usr/**"

- **-f file1[:file2:file3...]**
 - A colon separated list of files that **MIO** should use the trace module with
 - Default is "*"
 - Wildcards are permitted
 - -f "input.dat:output.txt"
 - -f "/scratch/bauerj/*"
 - -f "/scratch/**"

- **-e NAME=VALUE**
 - To set an environment variable

- **-mpirun (HPMPI)**
 - Append necessary miostats environment variables to MPIRUN_OPTIONS

How to use MIO?

miostats options (cont)

- **-tb (default)**

- Trace module to generate brief statistics

- Trace close : mpp971 <-> aix : LFct62 1156 : (673398/1274.12)=528.52 mbytes/s

- **-ts**

- Trace module to generate full statistics

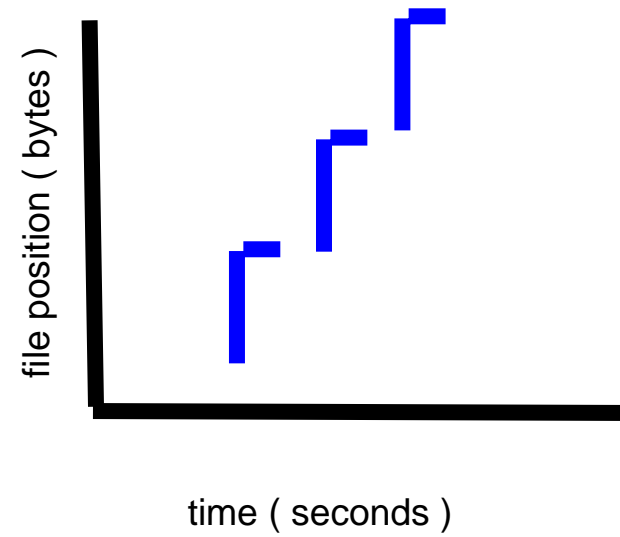
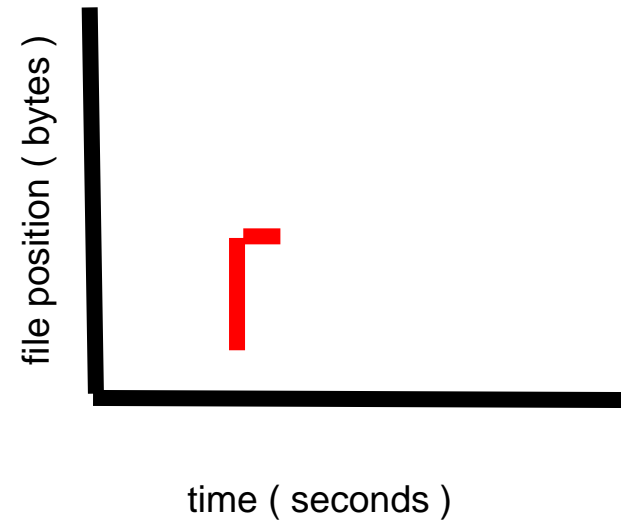
```
Trace close : mpp971 <-> aix : LFct62 : 1156 : (673398/1274.12)=528.52 mbytes/s
    demand rate=146.75 mbytes/s=673398/(4741.44-152.69))
    open_size=0, current_size=1156    max_size=1156
mode =0666  FileSystemType=EXT2 sector size=4096  inode=25100358
oflags =0x42=RDWR  CREAT
open          1      0.00
write        111026   19.40      3469 ==      3469      32768      32768
read         21437712 1254.72    669928 ==    669928      32768      32768
seek         21548738  10.76
                1 forward  seeks average=86048768
                30573 backward seeks average=23095548
fstat64       1      0.00
close         1      0.00
```

- **-te**

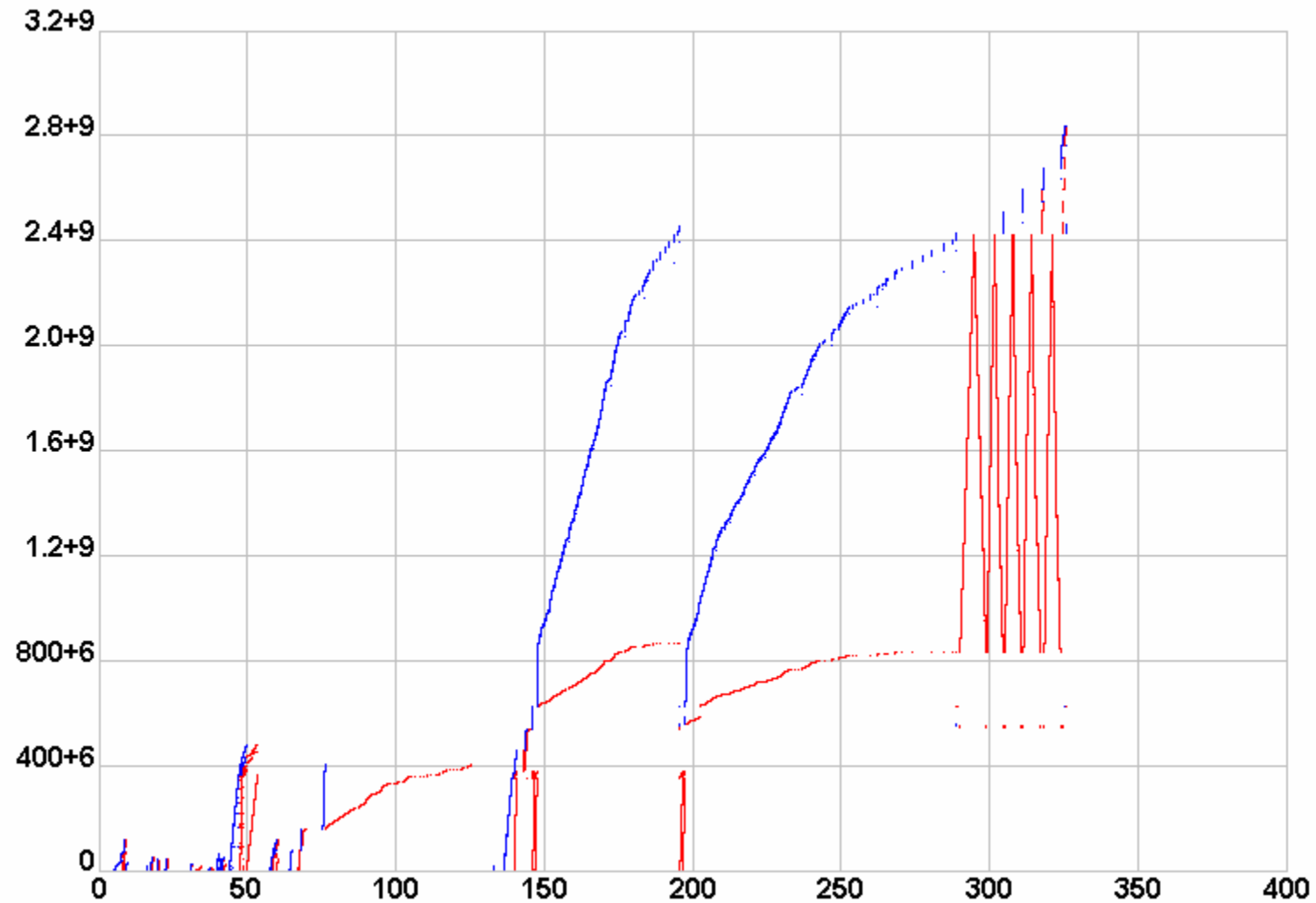
- Trace module to generate full statistics and evt files for DataView

DataView : overview

- **java application**
 - Works on windows Linux, and AIX
 - Reads trace module evt files
- **Invoking DataView**
 - `export CLASSPATH=./dv.jar`
 - `java dv.DataView [evtfile]`



DataView : *file activity plot*



MIO Summary

- demonstrated performance gains
- simple to implement
- flexible run time interface
- delivered as a shared object library for aix
- released with AIX 5.3H
- available for Linux/x86_64 for IBM customers