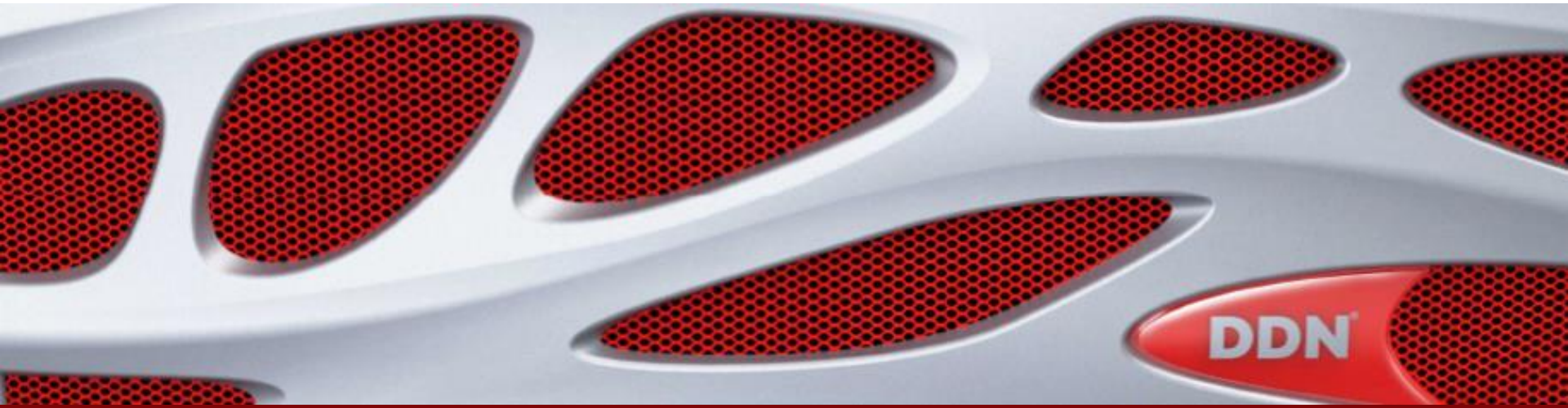
1

IO-Profiling

Björn Olausson, SE Germany



2

How

- FS/HW independent
- Easy
- All-embracing
- Low overhead

3

Applicable tools

- Darshan
- Strace (IOApps)
- SFA stats
- Systemtap
- DIO-pro

4

DARSHAN

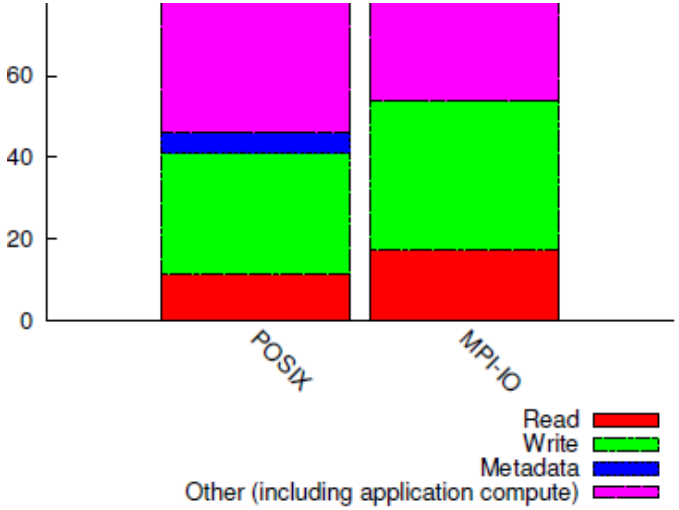
5

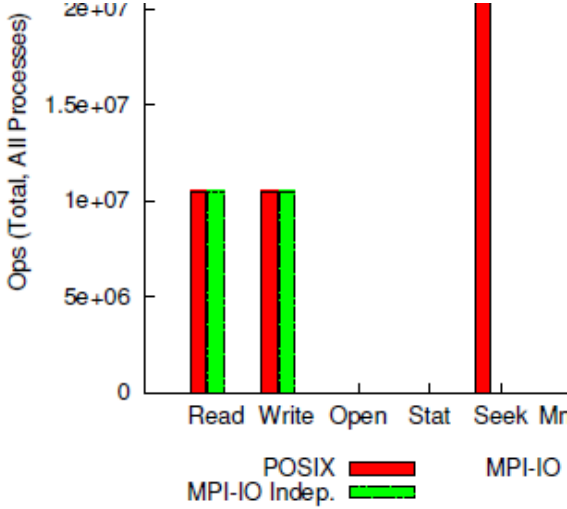
Capabilities – Darshan

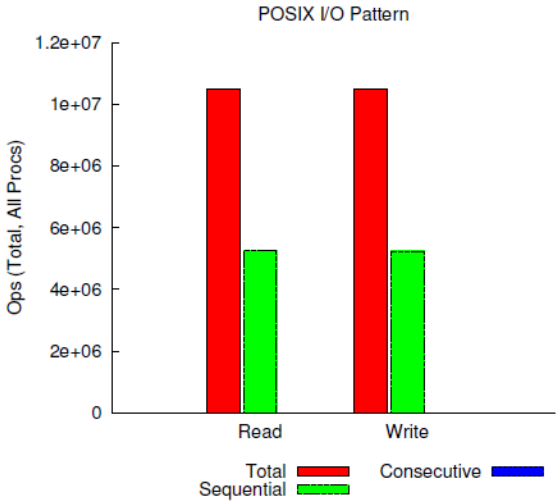
- **Ease of deployment**
  - LD_PRELOAD
  - PDF reports
- **What it captures**
  - MPI-IO
  - POSIX IO
  - HDF IO

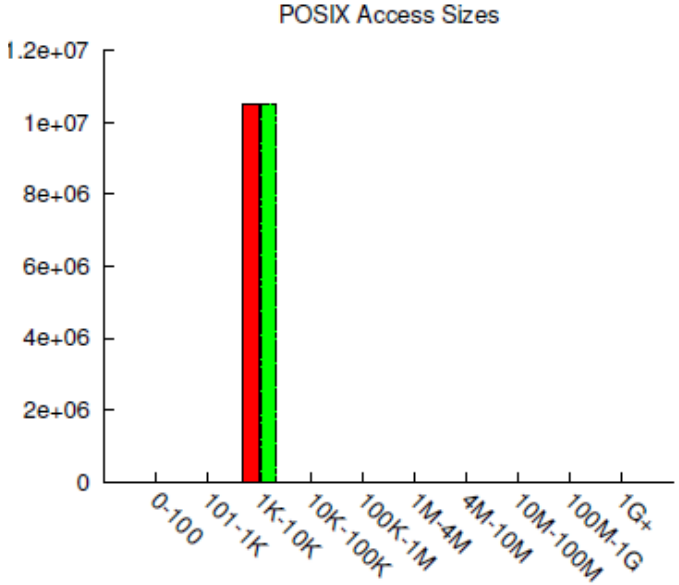
6

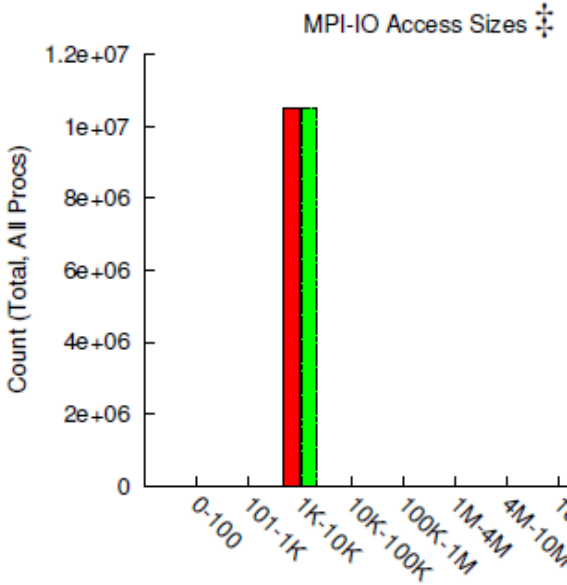
What it looks like











7

Limitations – DARSHAN

- No file access patterns

8

STRACE

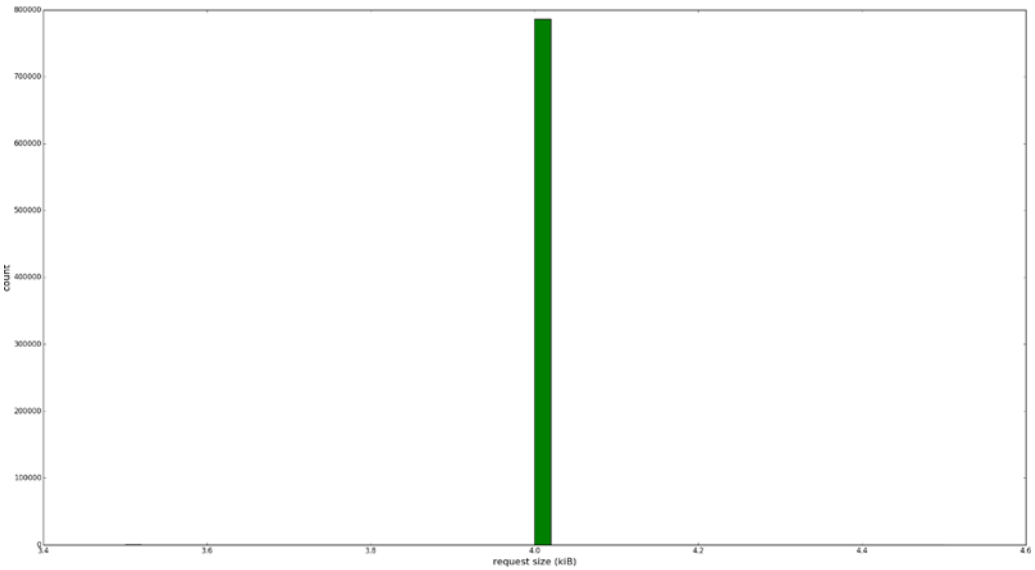
9

Capabilities – STRACE

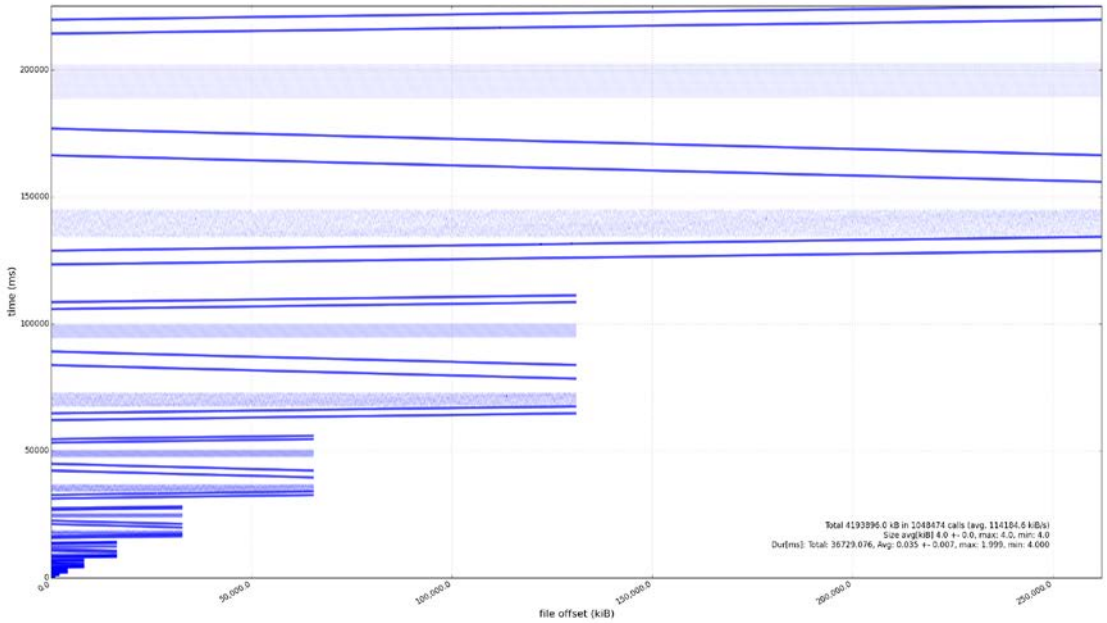
- **Ease of deployment**
  - Strace <your application>
  - Tools to process raw logs (IOApps)
- **What it captures**
  - All calls

10

What it looks like



Write size



Read & write patterns

11

Limitations – STRACE

- **Heavy performance impact**

Strace – Really?!

time ./create-file.py

Without tracing: ~ 5 seconds

time strace -o strace.log -T -ttt -e trace=write,read,lseek ./create-file.py

With **strace**: ~ 11 seconds

* average over 5 consecutive runs

13

SFA-STATS

14

Capabilities – SFA stats

- **Ease of deployment**
  - CLI tool for
    - Cache stats
    - VD/PD/CHANNEL counter
  - Spreadsheet
- **What it captures**
  - All IO

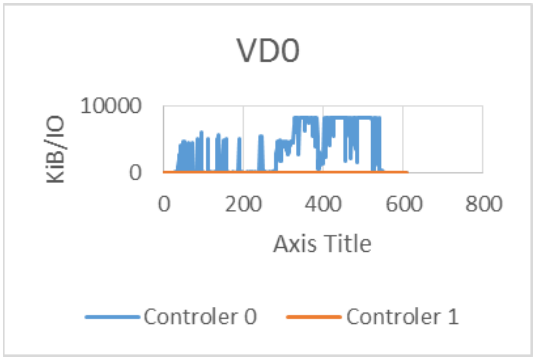
15

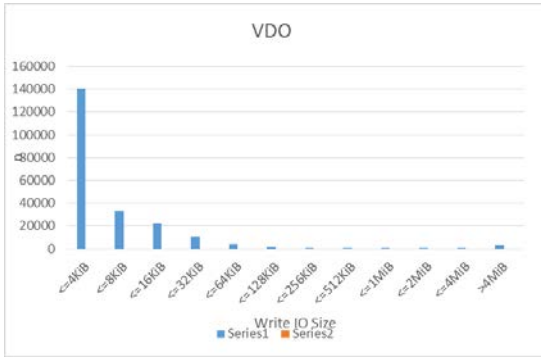
What it looks like (SFA-STATS)

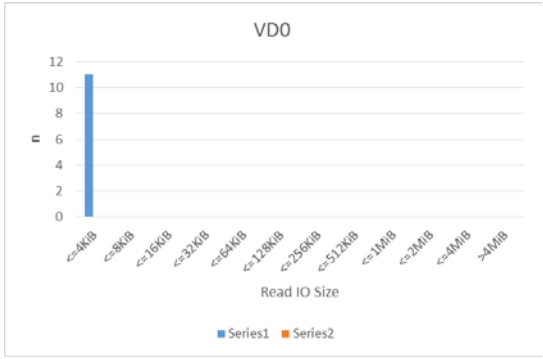
	Controler 0							Controler 1						
	VD0							VD0						
Time	IOs/sec	KiB/sec	KiB/IO	Fwd IO/s	Fwd KiB/s	R KiB/s	W KiB/s	Time	IOs/sec	KiB/sec	KiB/IO	Fwd IO/s	Fwd KiB/s	R KiB/s	W KiB/s
0,352	0	0	0	0	0	0	0	0,352	0	0	0	0	0	0	0
0,704	0	0	0	0	0	0	0	0,704	0	0	0	0	0	0	0
1,056	0	0	0	0	0	0	0	1,056	0	0	0	0	0	0	0

Write IO Size													
	140707	33296	22686	10627	3733	1317	516	161	132	191	1027	3706
	VD0											
Time	<=4KiB	<=8KiB	<=16KiB	<=32KiB	<=64KiB	<=128KiB	<=256KiB	<=512KiB	<=1MiB	<=2MiB	<=4MiB	>4MiB
0,352	0	0	0	0	0	0	0	0	0	0	0	0
0,704	0	0	0	0	0	0	0	0	0	0	0	0
1,056	0	0	0	0	0	0	0	0	0	0	0	0

Read IO Size												
11	0	0	0	0	0	0	0	0	0	0	0	0
VD0												
Time	<=4KiB	<=8KiB	<=16KiB	<=32KiB	<=64KiB	<=128KiB	<=256KiB	<=512KiB	<=1MiB	<=2MiB	<=4MiB	>4MiB
0,352	0	0	0	0	0	0	0	0	0	0	0	0
0,704	0	0	0	0	0	0	0	0	0	0	0	0
1,056	0	0	0	0	0	0	0	0	0	0	0	0







What it looks like (CACHE-STATS)

RP0

Ops

Time	Write Stripe	WT	NFSA	Total	Read Miss	Hit	Total
1	0	0	0	0	0	0	0
2	0	0	112	112	0	0	0

IOPS

Time	Write Stripe	WT	NFSA	Total	Read Miss	Hit	Total
1	0	0	0	0	0	0	0
2	0	0	112	112	0	0	0

MB/s

Time	Write Stripe	WT	NFSA	Total	Read Miss	Hit	Total
1	0	0	0	0	0	0	0
2	0	0	4	4	0	0	0

Front

Ops

Time	Send Pblocks	Blocks	Total	Receive Pblocks	Blocks
1	470	0	470	0	0

IOPS

Time	Send Pblocks	Blocks	Total	Receive Pblocks	Blocks
1	467	0	467	0	0

MB/s

Time	Send Pblocks	Blocks	Total	Receive Pblocks	Blocks
1	4	0	0	0	0

Ops

Time	Send Mxfers	Total	Received Mxfers(slots)	Total
1	0	983	959	982
2	0	23711	22225	23621

IOPS

Time	Send Mxfers	Total	Received Mxfers(slots)	Total
1	0	978	954	977
2	0	23711	22225	23621

MB/s

Time	Send Mxfers	Total	Received Mxfers(slots)	Total
1	13	0	0	0
2	1181	0	0	0

ICL

Ops

Time	Flush Stripe	RMW	WT	Parity	Total	Read Prefetch
1	16	476	0	492	489	0
		1040		1174	1171	

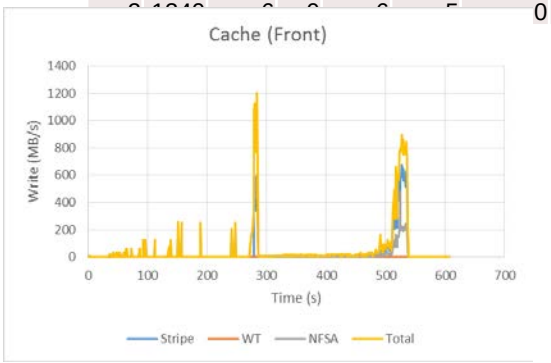
IOPS

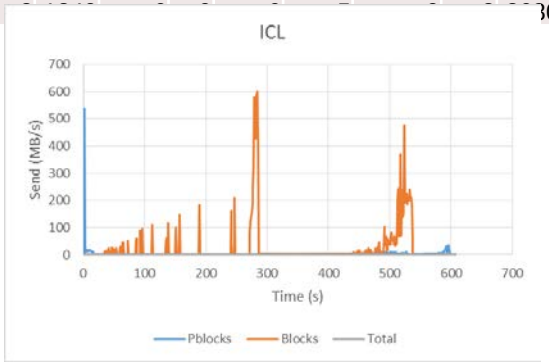
Time	Flush Stripe	RMW	WT	Parity	Total	Read Prefetch
1	15	473	0	489	486	0
		1040		1174	1171	

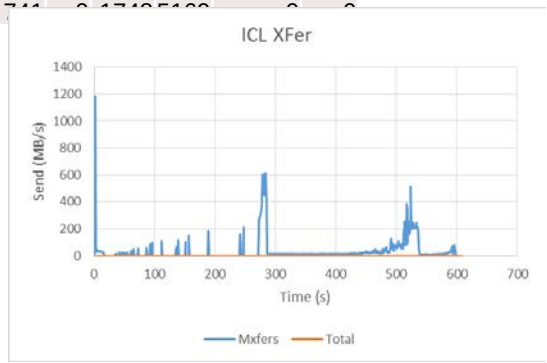
MB/s

Time	Flush Stripe	RMW	WT	Parity	Total	Read Prefetch	Total
1	31	5	0	18	55	0	0

Back







17

Limitations SFA

- Obviously you need a DDN system ;-)
- File access pattern and calls not obvious
- Timing only on a seconds scale
- No per file stats, only global

SystemTap

https://sourceware.org/systemtap/

SystemTap provides free software (GPL) infrastructure to simplify the gathering of information about the running Linux system

What was required to make it run?

1. One single node with:
   - yum install kernelname-devel-version
   - debuginfo-install kernelname-version
   - yum install systemtap systemtap-runtime

2. Test:
   - stap -v -e “probe vfs.read {printf("read performed\n"); exit()}”

3. DONE

Running IOR with SystemTap

1. Start SystemTap:
   - stap iomon.stp -w -Gbinfilter=ior -o /tmp/stap.log -F -S 1000,1 --remote IP1 --remote IP2
     - -w → Shut Up!
     - -o → Write output to file
     - -Gbinfilter=ior → Defines variable “binfilter”. Log only calls realted to “ior”
     - -F → “Flight-Recorder” mode (runs in background)
     - -S → max Filesize and number of files to keep in “Flight-Recorder” mode
     - --remote → SSH to remote host and execute stap

2. Start IOR:
   - mpiexec.hydra -np 8 -hosts IP1, IP2 -ppn 4 /lustre/bjoern/ior/bin/ior
   - Wait for IOR to finish

3. Stop SystemTap:
   - for IP in IP1 IP2; do ssh IP “killall stapio ; killall stapio” ; done

4. DONE!

Now the stunning results!

LITTLE IMPACT ON PERFORMANCE OR RUNTIME

(compared to native and strace)

Sorry for all CAPS – I got a bit excited!

Let me prove it! (on a small example)

IOR setup

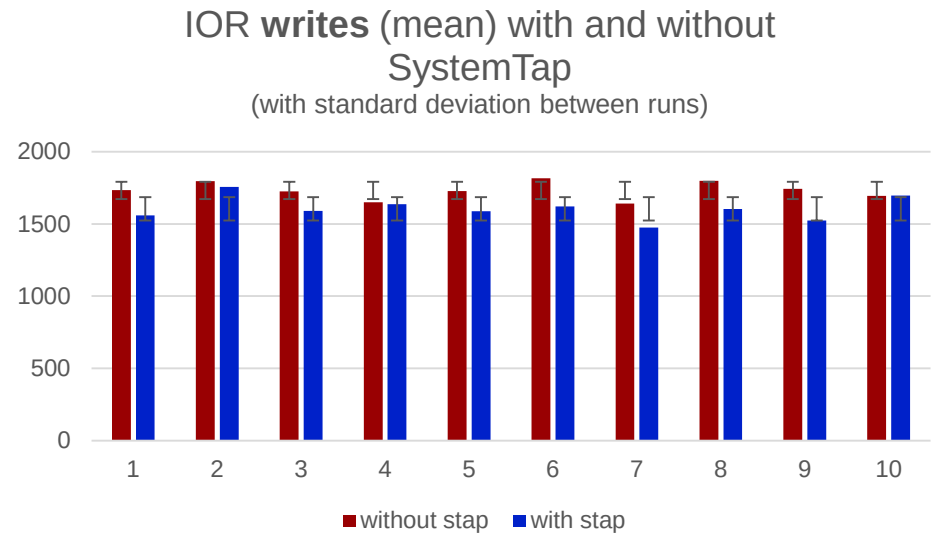
```
IOR START
        testFile = /lustre/bjoern/ior_systemtap.file
        filePerProc=0
        api=POSIX
        fsync=0
        randomOffset=0
        useStridedDatatype=0
        useO_DIRECT=0
        repetitions=3
        verbose=1
        blockSize=200m
        transferSize=128k
        RUN
IOR STOP
```

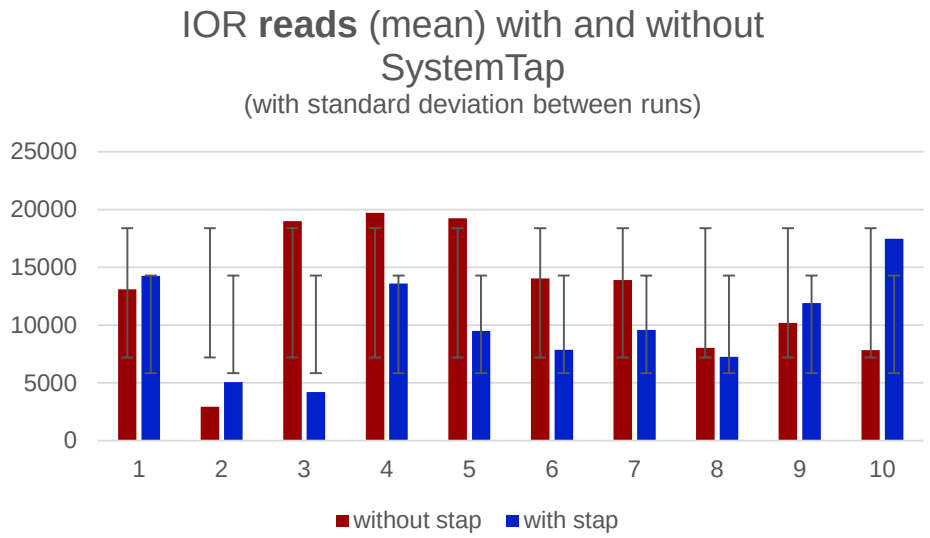
IOR output

```
Test 0 started: Mon May  2 21:53:46 2016
Path: /lustre/bjoern
FS: 171.0 TiB   Used FS: 10.3%   Inodes: 223.0 Mi   Used Inodes: 2.3%
Participating tasks: 8
Summary:
                api               = POSIX
                test filename     = /lustre/bjoern/ior_systemtap.file
                access            = single-shared-file
                pattern           = segmented (1 segment)
                ordering in a file = sequential offsets
                ordering inter file= no tasks offsets
                clients           = 8 (4 per node)
                repetitions       = 3
                xfersize          = 131072 bytes
                blocksize         = 200 MiB
                aggregate filesize = 1.56 GiB
```

Let me prove it! (on a small example)

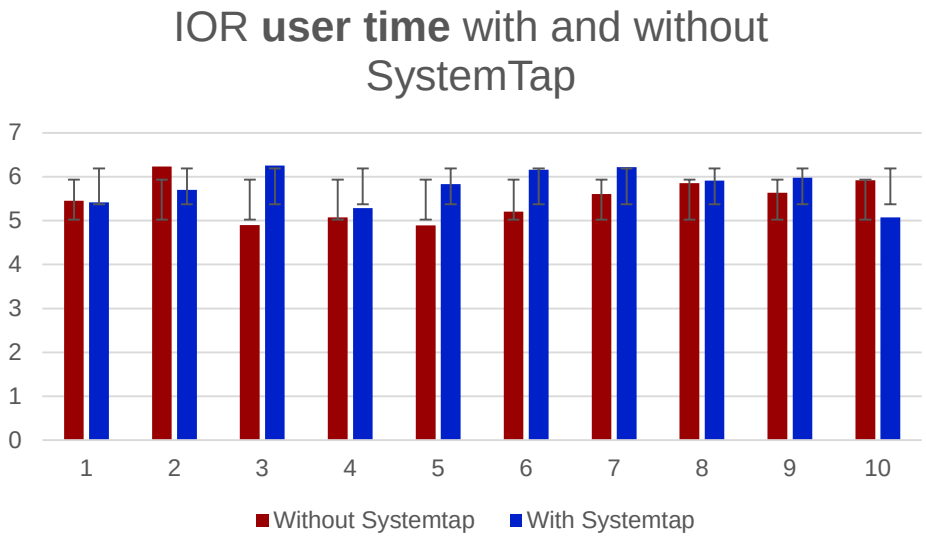
IOR results

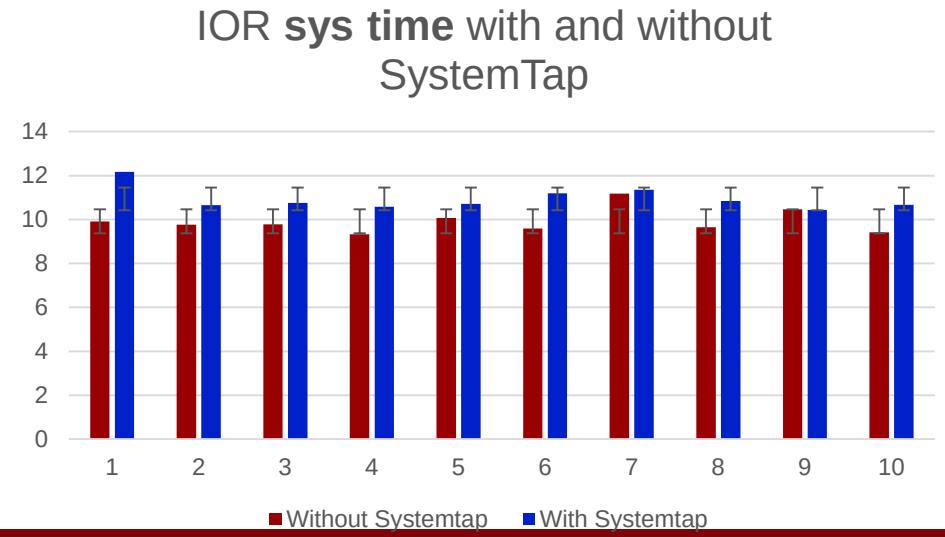


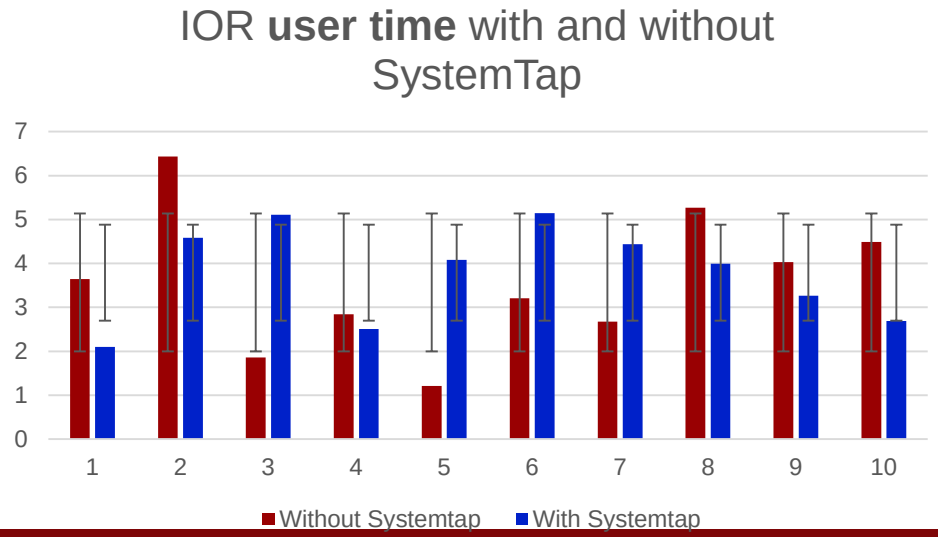


Let me prove it! (on a small example)

IOR runtime







Strace – Really?!

When you make the comparison: SystemTap compiles the stap script to a Kernel module and loads it → Adds ~5 seconds delay to the startup. This can be consider constant and does not change with the runtime of the application to trace

time ./create-file.py

Without tracing: ~ 5 seconds*

time strace -o strace.log -T -ttt -e trace=write,read,lseek ./create-file.py

With **strace**: ~ 11 seconds*

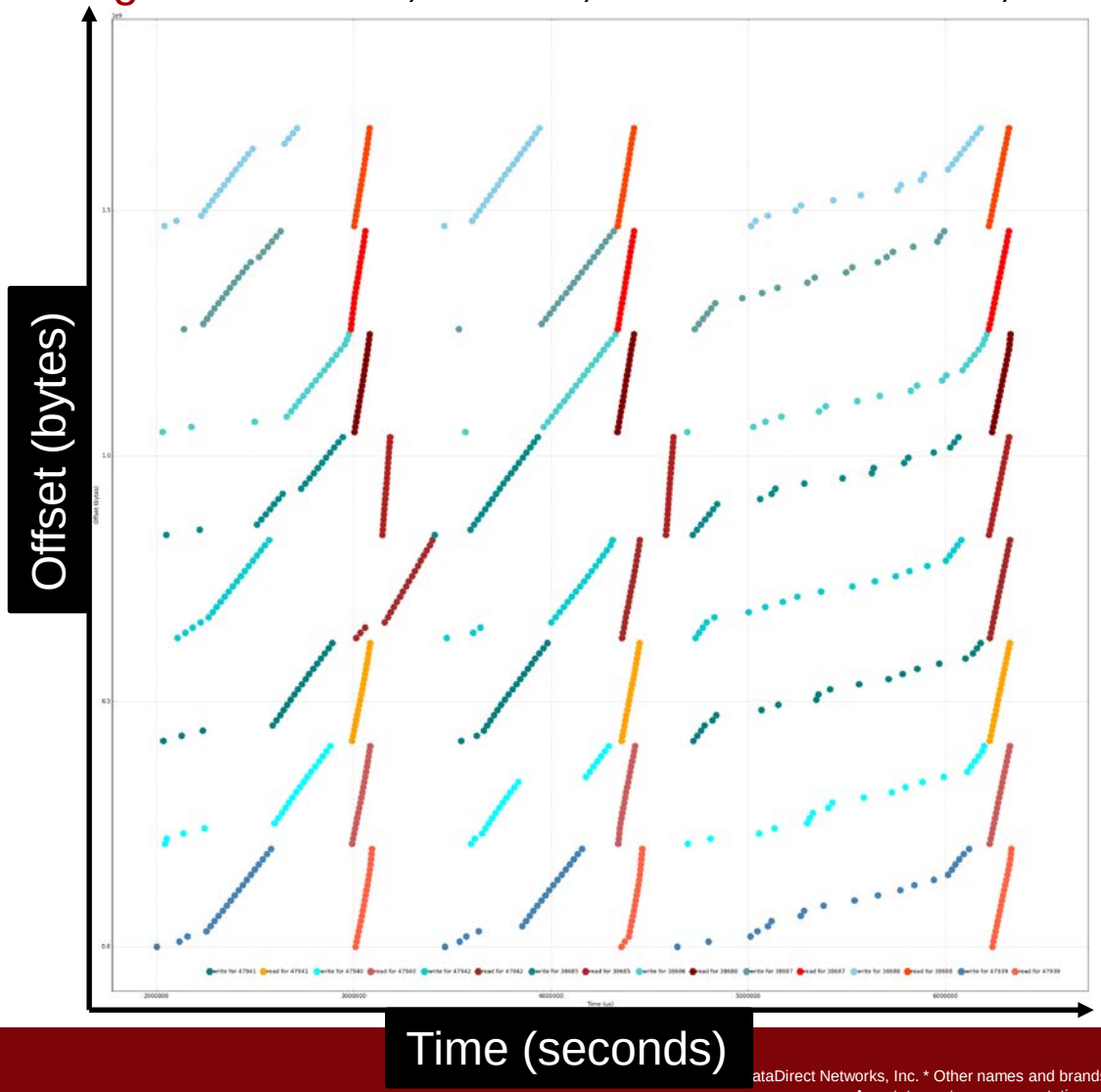
time stap my_basic_iomon.stp -o stap.log -c ./create-file.py

With **SystemTap**: ~ 5 seconds*

* average over 5 consecutive runs

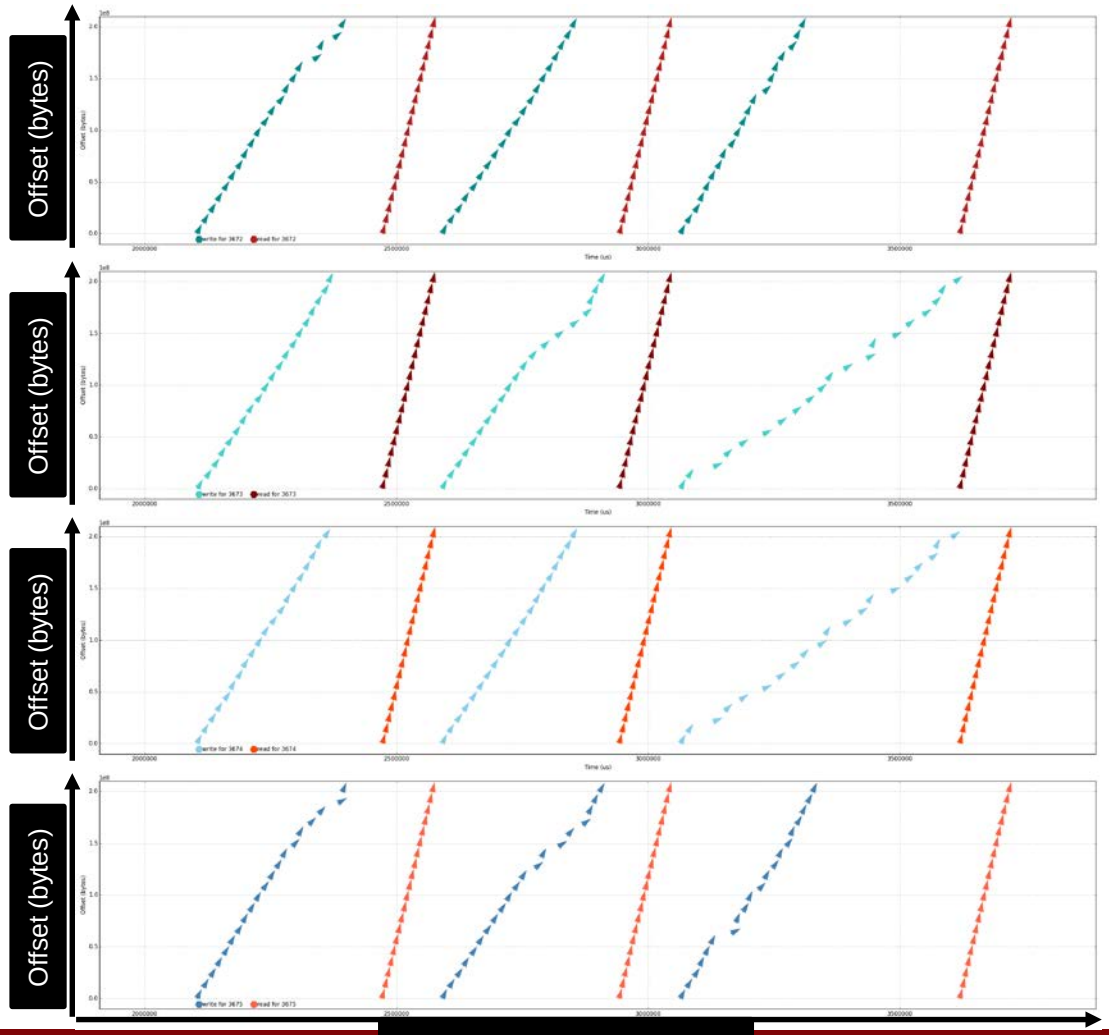
Some vfs.read & vfs.write plots

Single shared file, 200MB, 10MB transfersize , POSIX



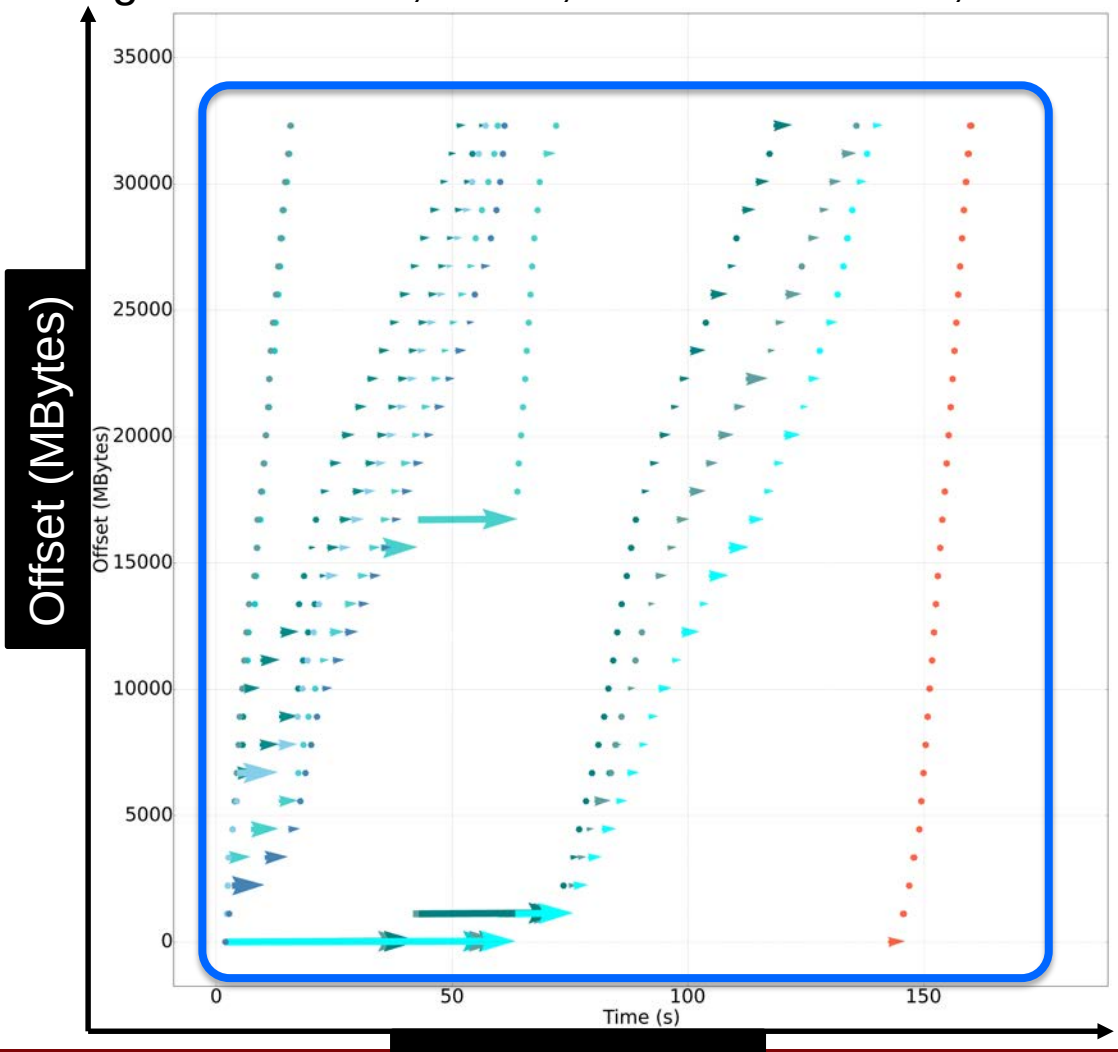
Some vfs.read & vfs.write plots

File per process, 200MB, 10MB transfersize , POSIX



Some vfs.read & vfs.write plots

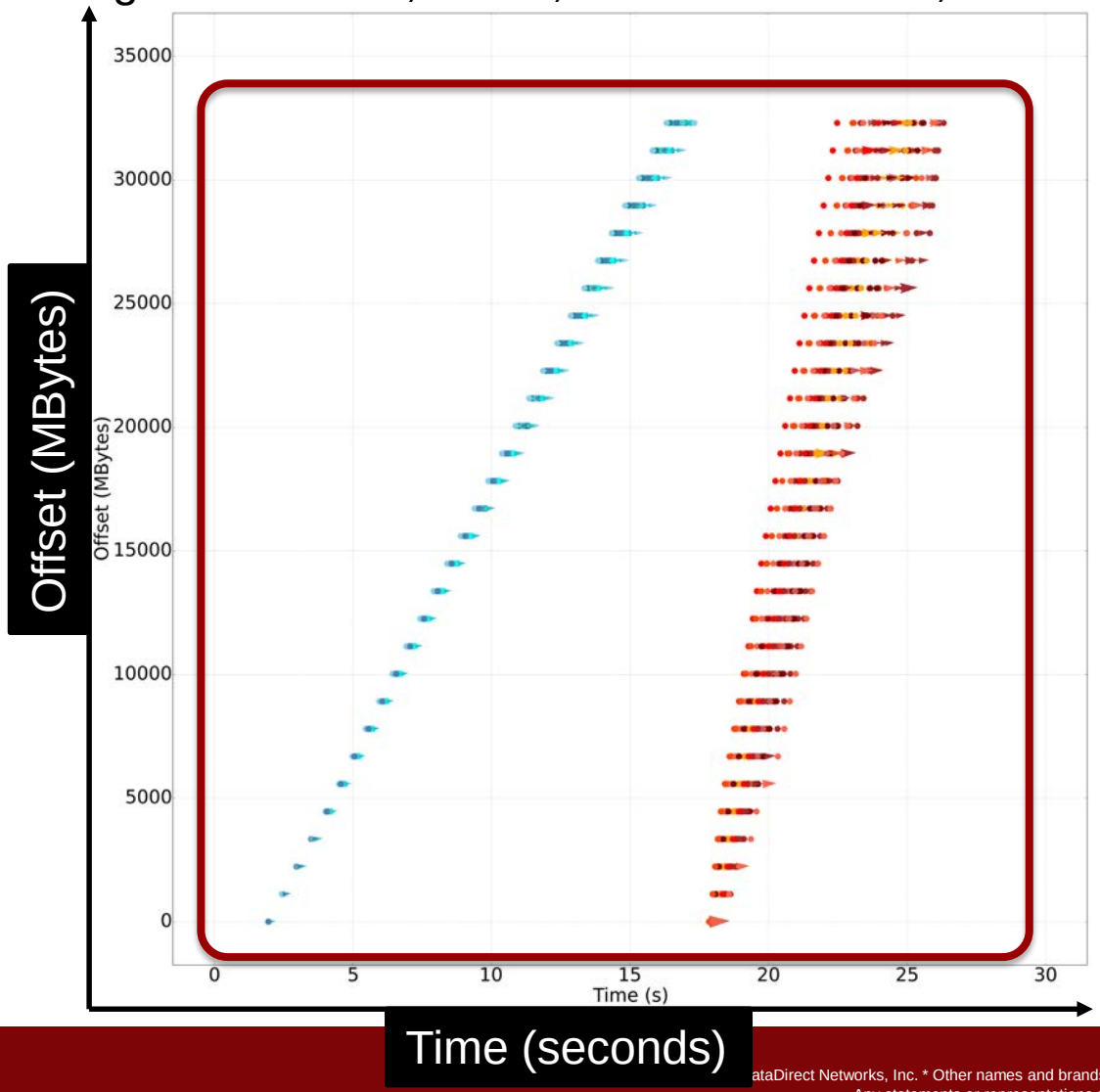
Single shared file, 32GB, 17kb transfersize, POSIX, 8 Threads, 4 Nodes



- GPFS (150s)

Some vfs.read & vfs.write plots

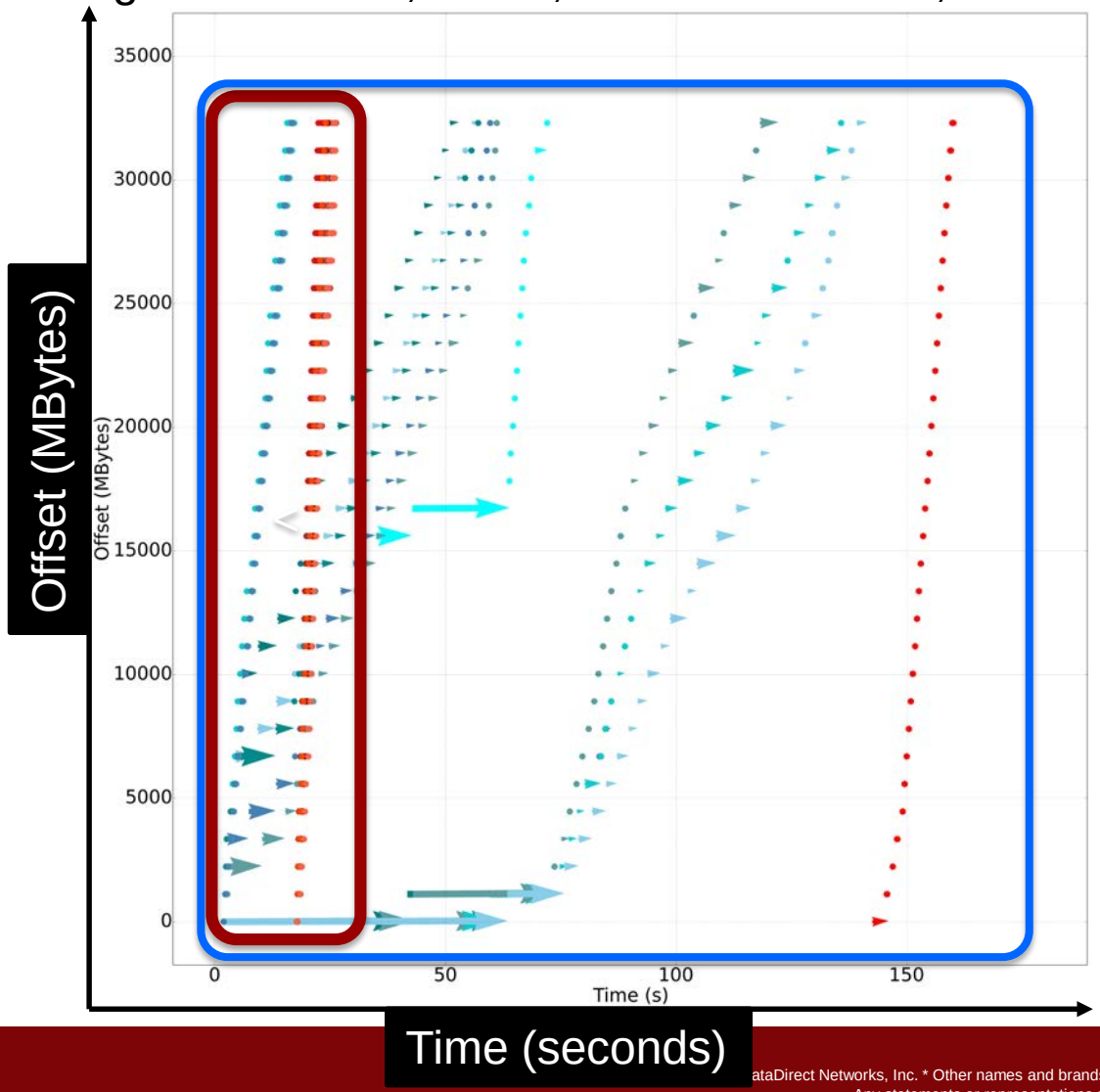
Single shared file, 32GB, 17kb transfersize, POSIX, 8 Threads, 4 Nodes



- IME (22s)

Some vfs.read & vfs.write plots

Single shared file, 32GB, 17kb transfersize, POSIX, 8 Threads, 4 Nodes



- GPFS (150s)
- IME (22s)

ToDo

1. Add more syscalls besides vfs.read, vfs.write, vfs.lseek
   - We can monitor:
     - Stat
     - Create
     - Open
     - Close
     - …
2. Add MPI-IO support
   - There is a plan…

```
probe process("/usr/mpi/gcc/mvapich2-2.1/lib/???.so")
.function("*io*") { }
```

Conclusion

1. Easy to use
2. Little overhead (reasonable for debugging and tracing applications)
3. Sophisticated monitoring possible

35

DIO-pro

What is DIO-pro

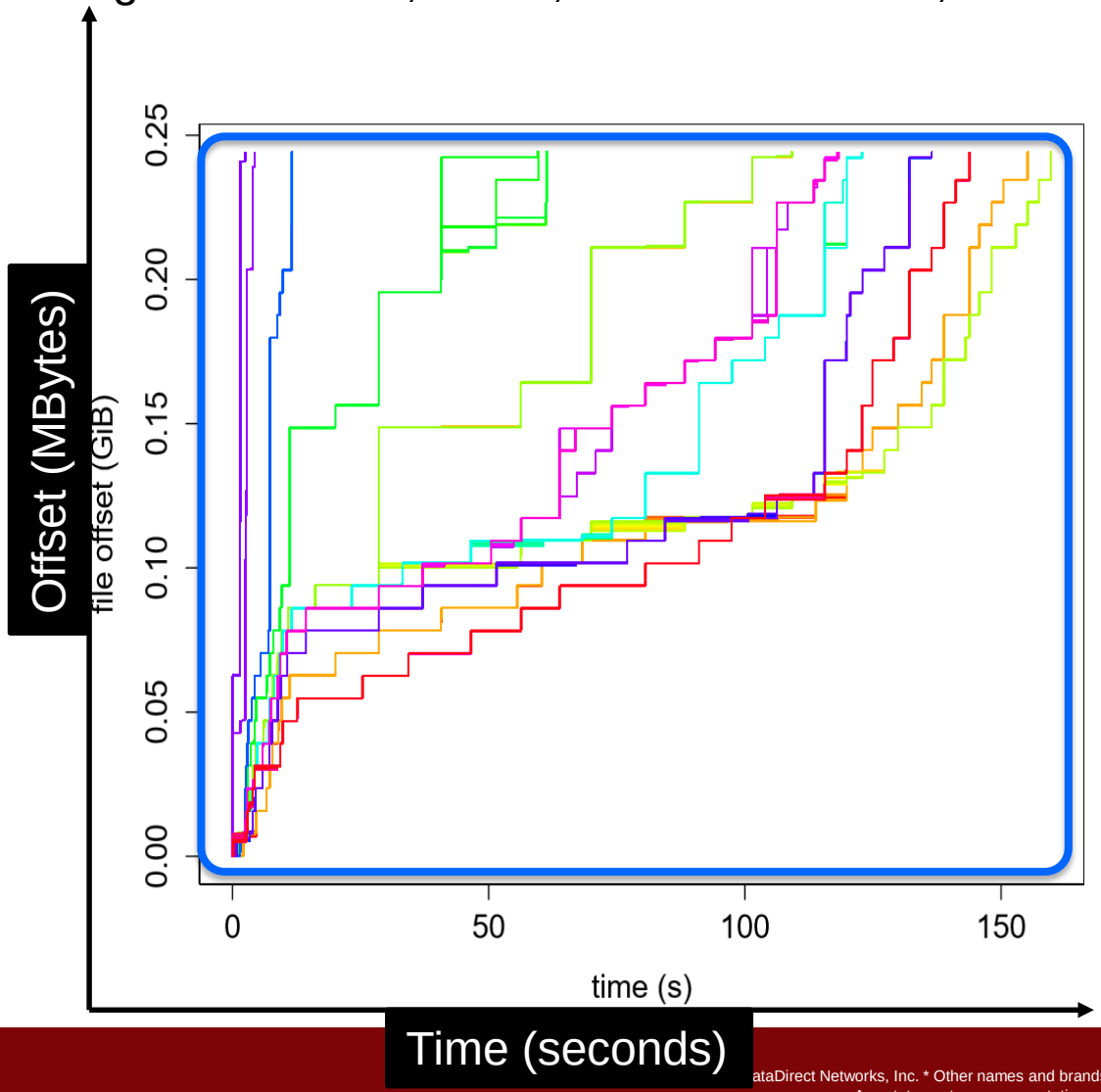
- DIO-pro is inspired on Darshan's technology
- DIO-pro offers extended features and higher accuracy
- Developed by DDN

DIO-pro in a nutshell

- **Profiles lightweight (overhead <1%)**
- **Includes tools to manipulate data logs**
- **Metrics per file, per process, and overall**
  - read/write speed: max, min, average, std. dev.
  - Meta data time
  - block sizes
  - POSIX/MPIIO function calls
- **File contention analysis**
- **Additional metrics continuously in development**
  - I/O arrival rate
  - spatial randomness
  - burst inter-arrival time
  - spatial-temporal correlation
  - …

IOR IO-pattern

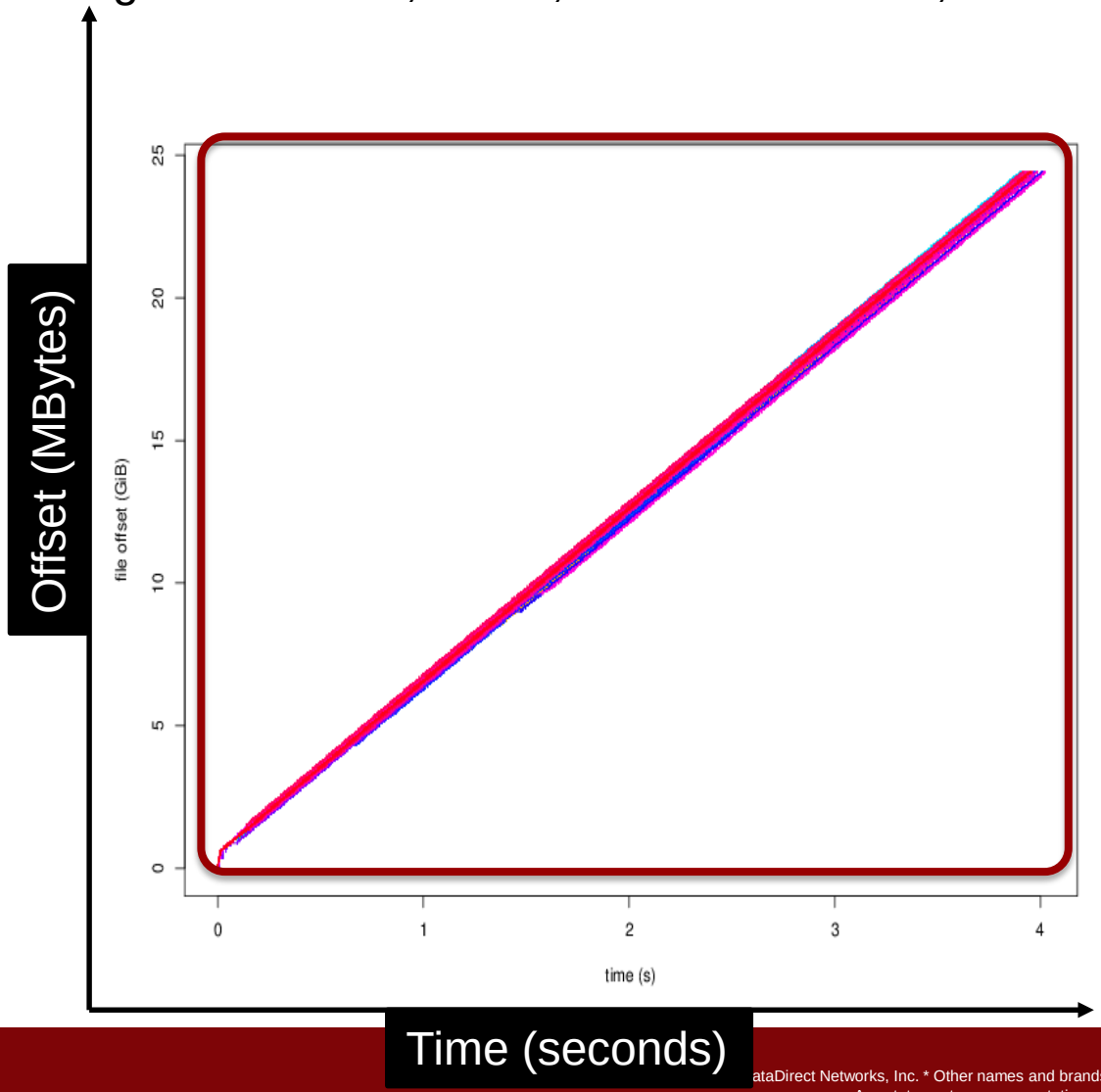
Single shared file, 25GB, 17kb transfersize, POSIX, 8 Threads, 4 Nodes



- GPFS (150s)

IOR IO-pattern

Single shared file, 25GB, 17kb transfersize, POSIX, 8 Threads, 4 Nodes



- IME (4s)

Thank You!

Keep in touch with us

sales@ddn.com

9351 Deering Avenue
Chatsworth, CA 91311

@ddn_limitless

1.800.837.2298
1.818.700.4000

company/datadirect-networks