Filebench Tutorial

Richard McDougall

Sun Microsystems

We make the net work.

Why did we invest in a File System Perf. Framework?

- We need complete test coverage for file level applications
  - Current test coverage is mostly via “micro benchmarks”: Bonnie, iozone, mongo
  - Test coverage was very limited (less than 10% of important cases covered)
  - The current approach is to use benchmark full application suites: e.g. Oracle using TPC-C: expensive, labor intensive
  - Up to 100 different benchmarks are required to accurately report on filesystem performance today
- SPECsfs is limited to NFS Version 3
  - And only represents “home directory servers”

Requirements for file-level benchmarking

- Represent Apps rather than I/Os
- Trace-derived synthesis
- Thread-level representation
- Inter-thread dependency/sync.
- Forward Path
- Extensible to new protocols
- Modular to include test of client: process/thread model, cpu efficiency etc...
- Pre-structuring/aging of file sets
- Scalable
  - Throughput, #Users
  - #Files/Directories
  - Working set size
  - #Clients
  - Client resources (mem/cpu)

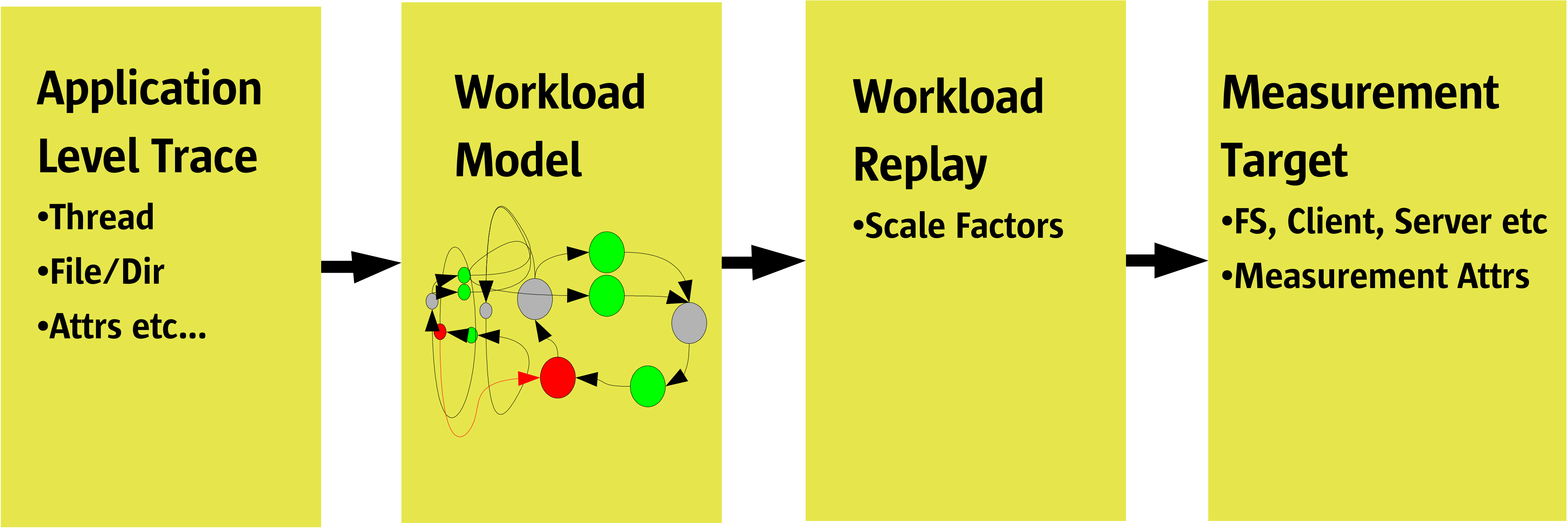
Characterization Strategies

- I/O Microbenchmarking
  - Pros: Easy to run
  - Cons: Small test coverage, Hard to correlate to real apps
- Trace Capture/Replay
  - I/O Trace, NFS Trace, Application Trace
  - Pros: Accurate reconstruction of real application I/O mix
  - Cons: Large traces, difficult to reconstruct I/O dependencies
- Model Based
  - Distillation of trace into representive model
  - Probability based, Simulation based
  - Pros: Easy to run, Scalable in multiple dimensions
  - Cons: Care required to ensure accurate real-world representation

FileBench: Application Level File System Measurement

- FileBench is a configurable file level workload synthesis and measurement framework
- FileBench is an application simulator
  - Facilitates easy reproduction of complex applications
  - Applications are pre-defined by “workload descriptions”
- Workloads closely mimic real applications
  - Unique model-based approach can emulate complex applications – for example Oracle RDBMS
  - Workloads are defined using a model-language “f”
- Framework is highly extensible

Model based methodology study



Application Level Trace

- Thread
- File/Dir
- Attrs etc...

Workload Model

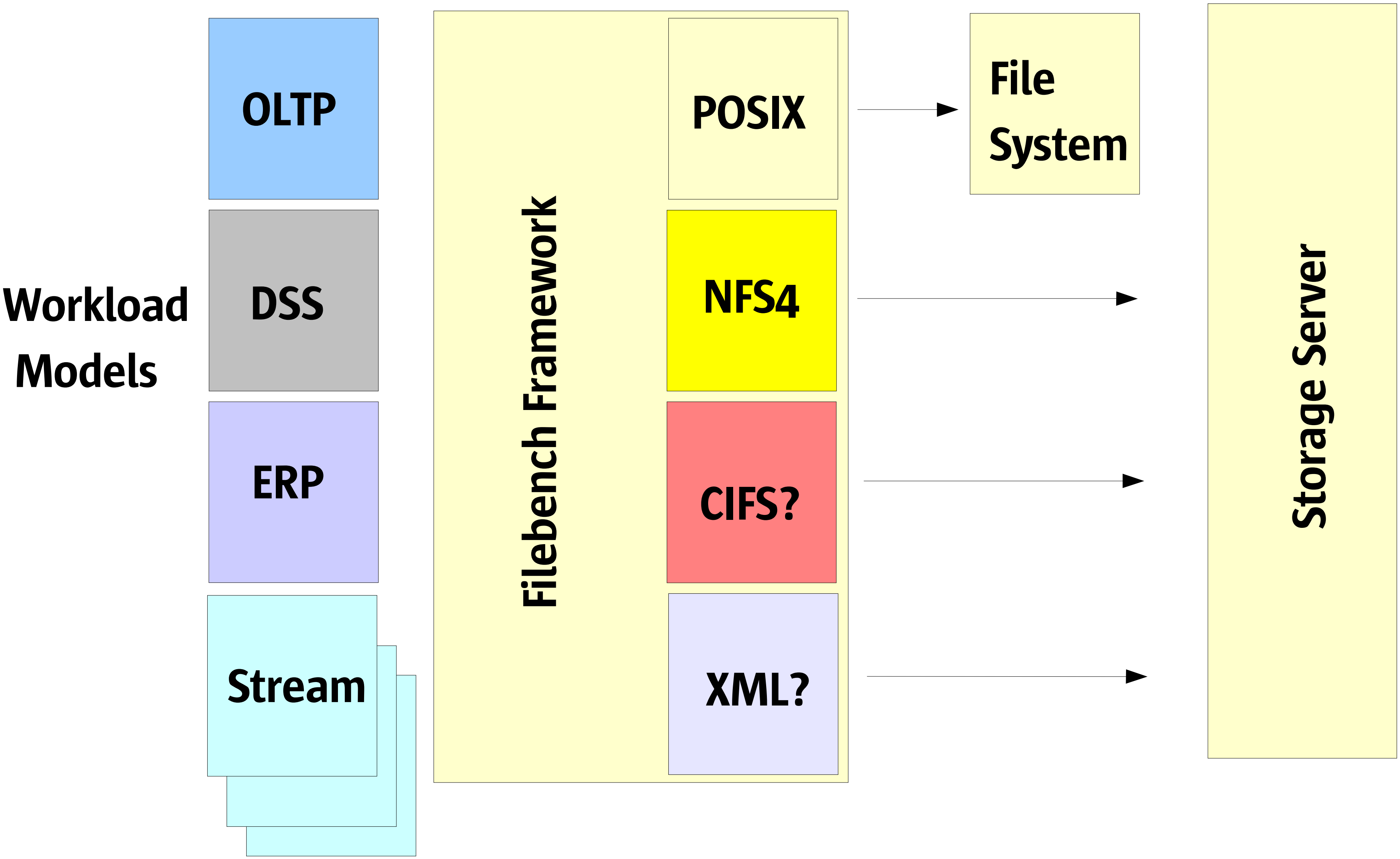
Workload Replay

- Scale Factors

Measurement Target

- FS, Client, Server etc
- Measurement Attrs

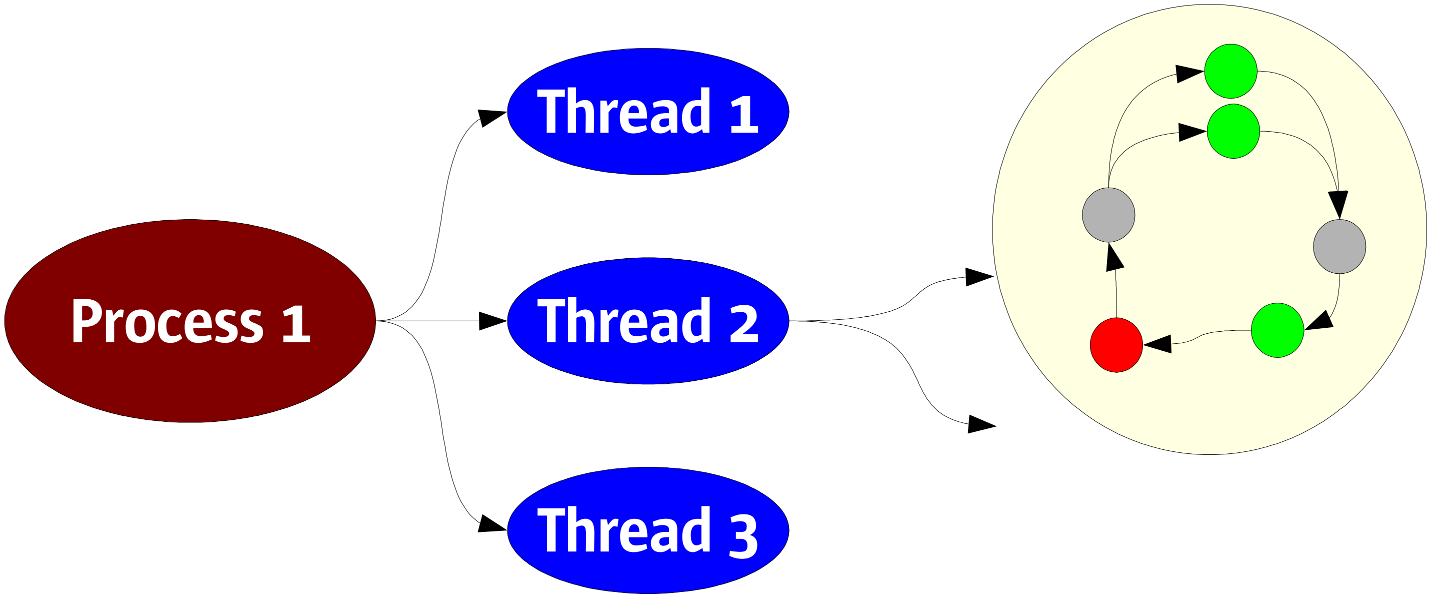
Filebench Achitecture



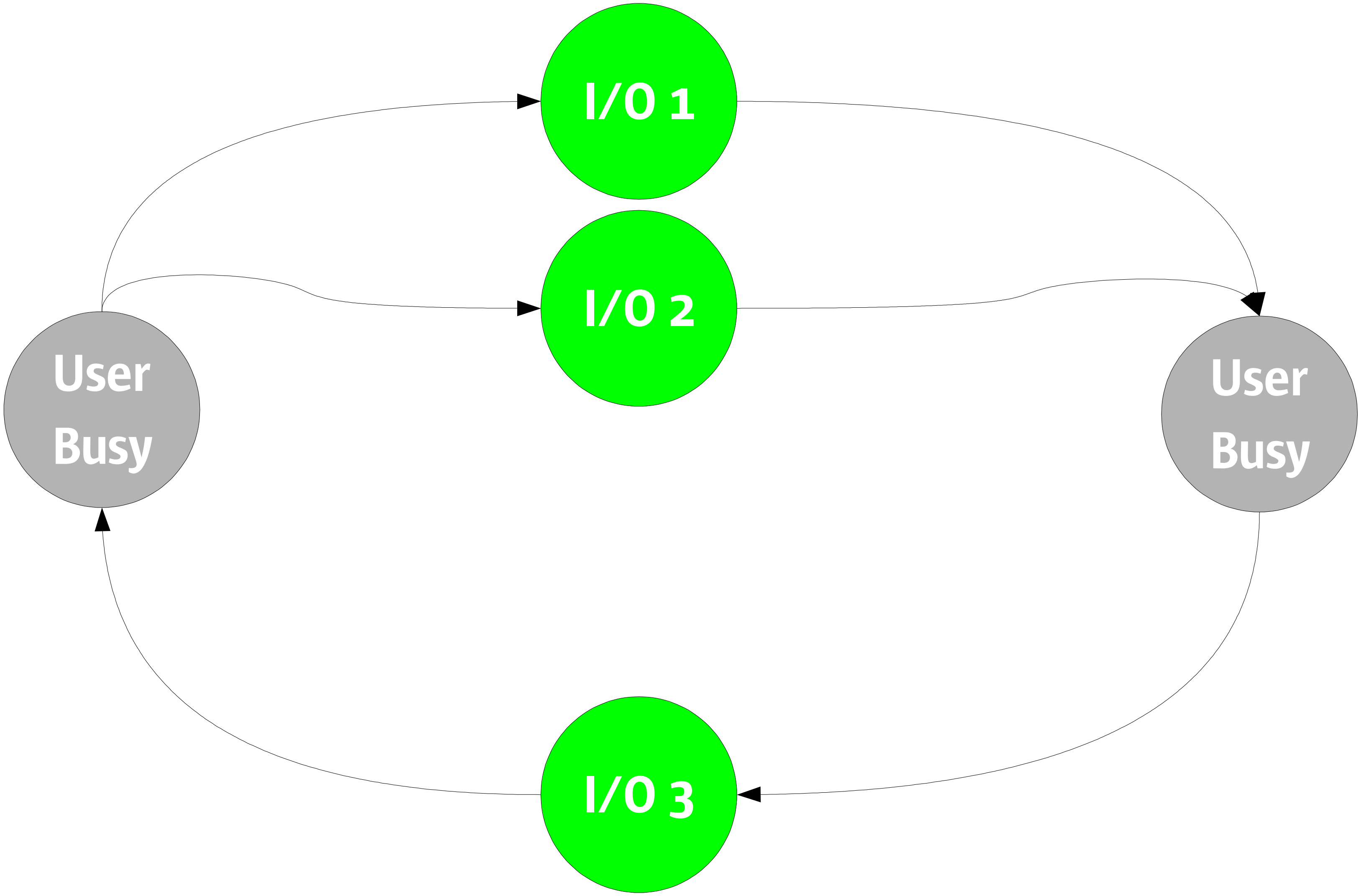
Model Allows Complex/Important Scaling Curves

- e.g.
  - Thoughput/Latency vs. Working set size
  - Thoughput/Latency vs. #users
  - CPU Efficiency vs. Thoughput
  - Caching efficiency vs. Workingset size/Memsize

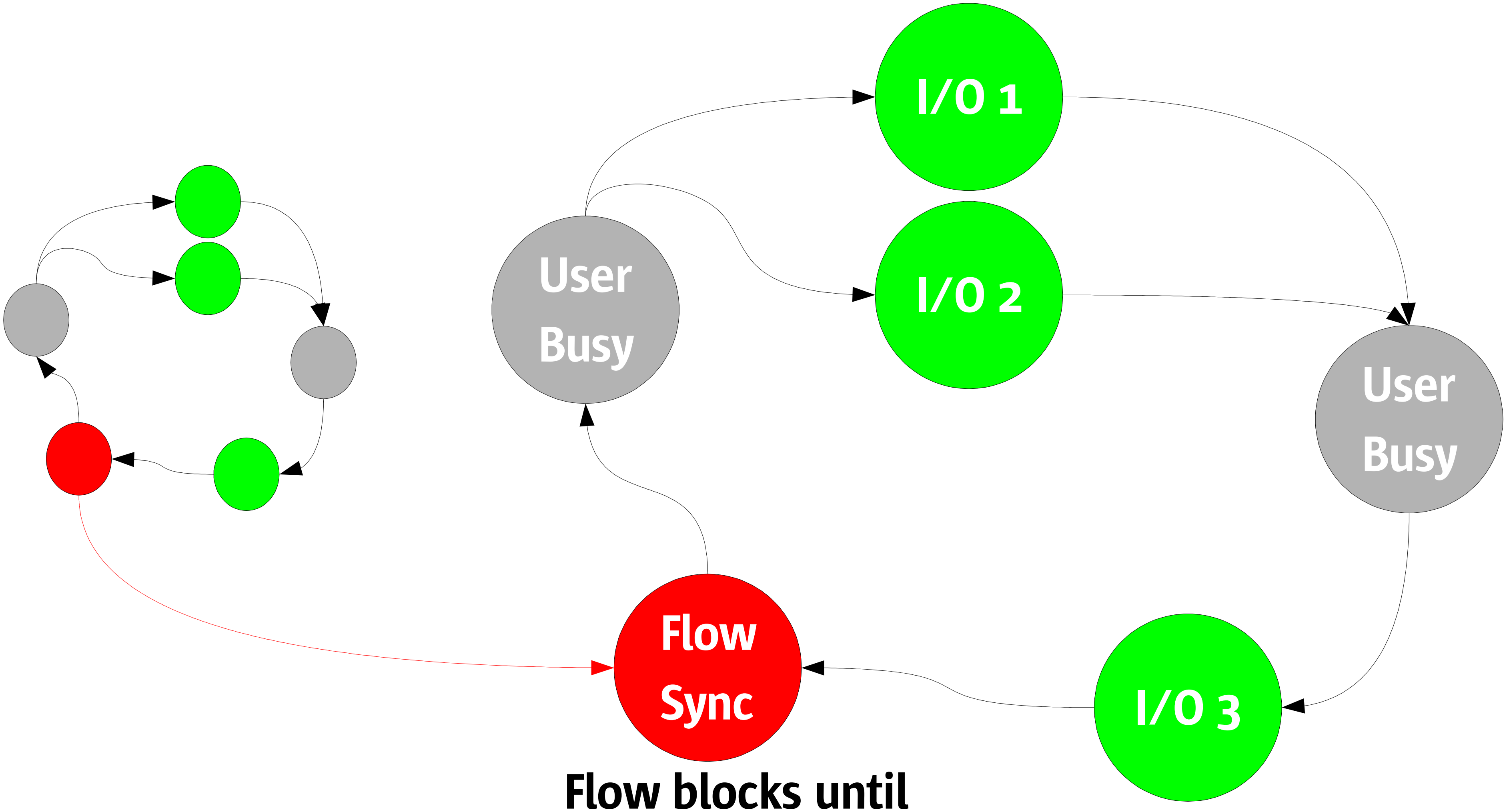
Characterize and Simulate via Cascades of Workload Flows:



Flow States: Open Ended Flow



Flow States: Synchronized Flow



Flow blocks until completion of other flow

Examples of Per-flow Operations

- Types
  - Read
  - Write
  - Create
  - Delete
  - Append
  - Getattr
  - Setattr
  - Readdir
  - Semaphore block/post
  - Rate limit
  - Throughput limit

- Attributes
  - Sync_Data
  - Sync_Metadata
  - IO Size
  - I/O Pattern, probabilities
  - Working set size
  - Etc...

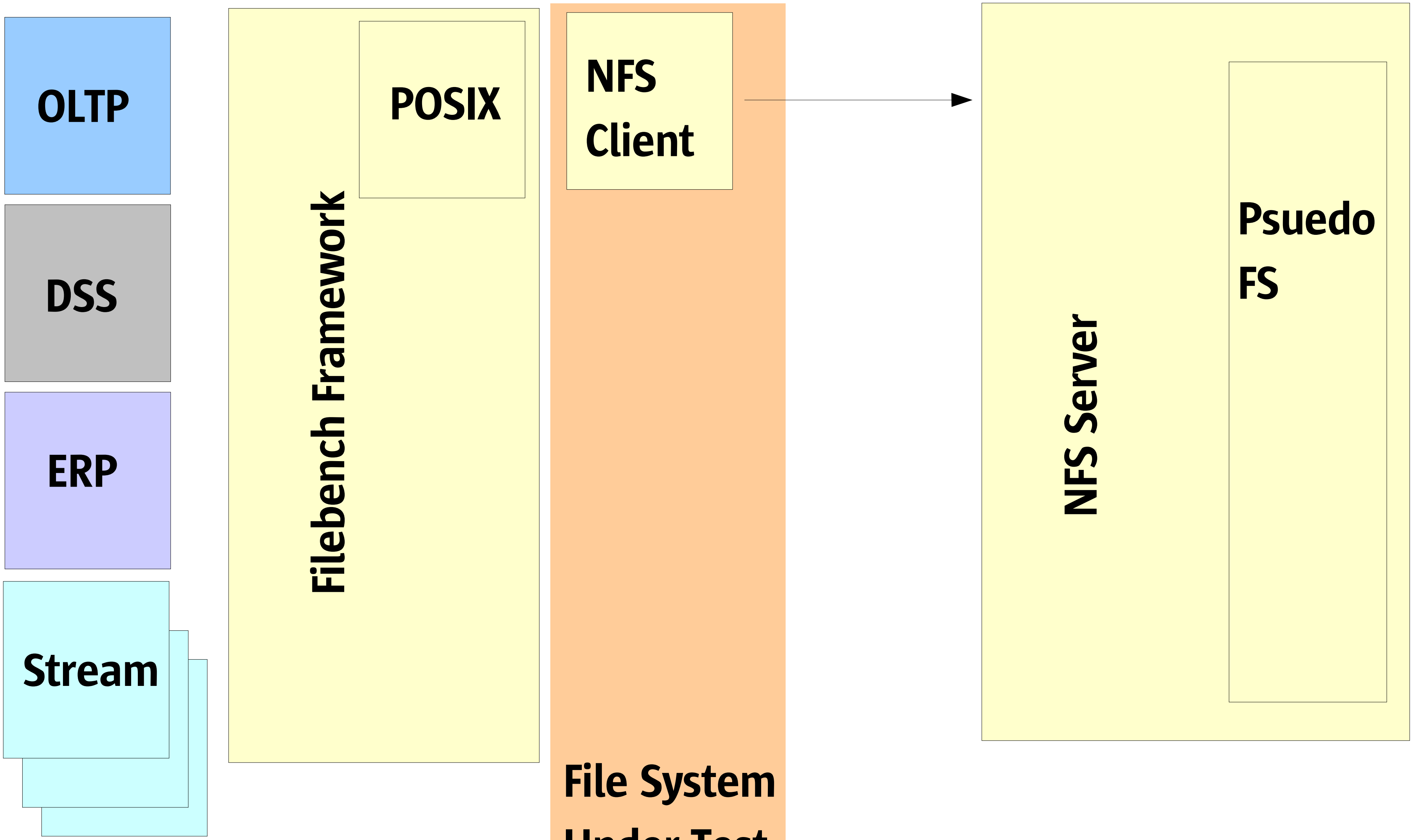
Simple Random I/O Workload Description

```
define file name=bigfile0,path=$dir,size=$filesize,prealloc,reuse,paralloc

define process name=rand-read,instances=1
{
  thread name=rand-thread,memsize=5m,instances=$nthreads
  {
    flowop read name=rand-read1,filename=bigfile0,iosize=$iosize,random
    flowop eventlimit name=rand-rate
  }
}
```

Files and Filesets

- Files: a definition of a single file
  - Soon to be deprecated
- Filesets: a definition of a set of files
  - A fractal tree of files
  - A fileset has a depth and size, width of directories is computed from these
  - Can also have a depth of 1 to make one large directory
  - Can have uniform sizes, depths, widths or configured as a [gamma] distribution
  - Filesets that mimic file servers typically use gamma distribution for size and depth.

NFS Client Testing: POSIX level workload + NFS server

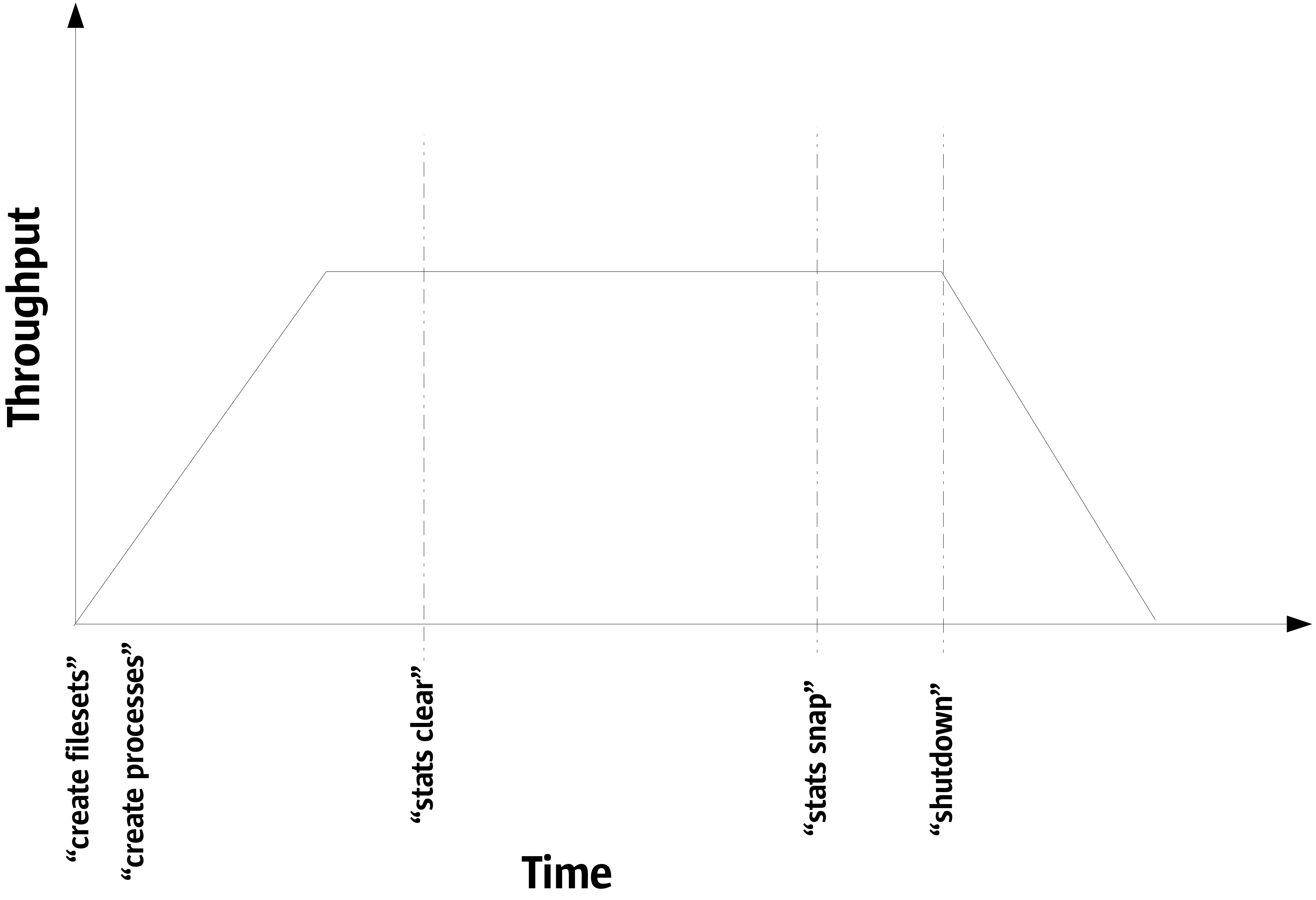


Running a single FileBench workload...

Example varmail run:

```
filebench> load varmail

 Varmail personality successfully loaded
 Usage: set $dir=<dir>
        set $filesize=<size>    defaults to 16384
        set $nfiles=<value>     defaults to 1000
        set $dirwidth=<value>   defaults to 20
        set $nthreads=<value>   defaults to 1
        set $meaniosize=<value> defaults to 16384
        run <runtime>

filebench> set $dir=/tmp

filebench> run 10

 Fileset mailset: 1000 files, avg dir = 20, avg depth = 2.3,mbytes=15
 Preallocated fileset mailset in 1 seconds
 Starting 1 filereader instances
 Starting 1 filereaderthread threads
 Running for 10 seconds...
 IO Summary: 21272 iops 2126.0 iops/s, (1063/1063 r/w) 32.1mb/s,338us cpu/op, 0.3ms
   latency
```

The steps behind the “run” command



Running a single filebench workload...

```
Example varmail run:

filebench> load varmail

 Varmail personality successfully loaded
 Usage: set $dir=<dir>
        set $filesize=<size>    defaults to 16384
        set $nfiles=<value>     defaults to 1000
        set $dirwidth=<value>   defaults to 20
        set $nthreads=<value>   defaults to 1
        set $meaniosize=<value> defaults to 16384
        run <runtime>

filebench> set $dir=/tmp
filebench> create filesets
 Fileset mailset: 1000 files, avg dir = 20, avg depth = 2.3,mbytes=15
 Preallocated fileset mailset in 1 seconds
filebench> create processes
 Starting 1 filereader instances
 Starting 1 filereaderthread threads
filebench> stats clear
filebench> sleep 10
 Running for 10 seconds...
filebench> stats snap
filebench> stats dump “mystats.out”
 IO Summary: 21272 iops 2126.0 iops/s, (1063/1063 r/w) 32.1mb/s,338us cpu/op, 0.3ms
   latency
filebench> shutdown
```

Listing available workloads...

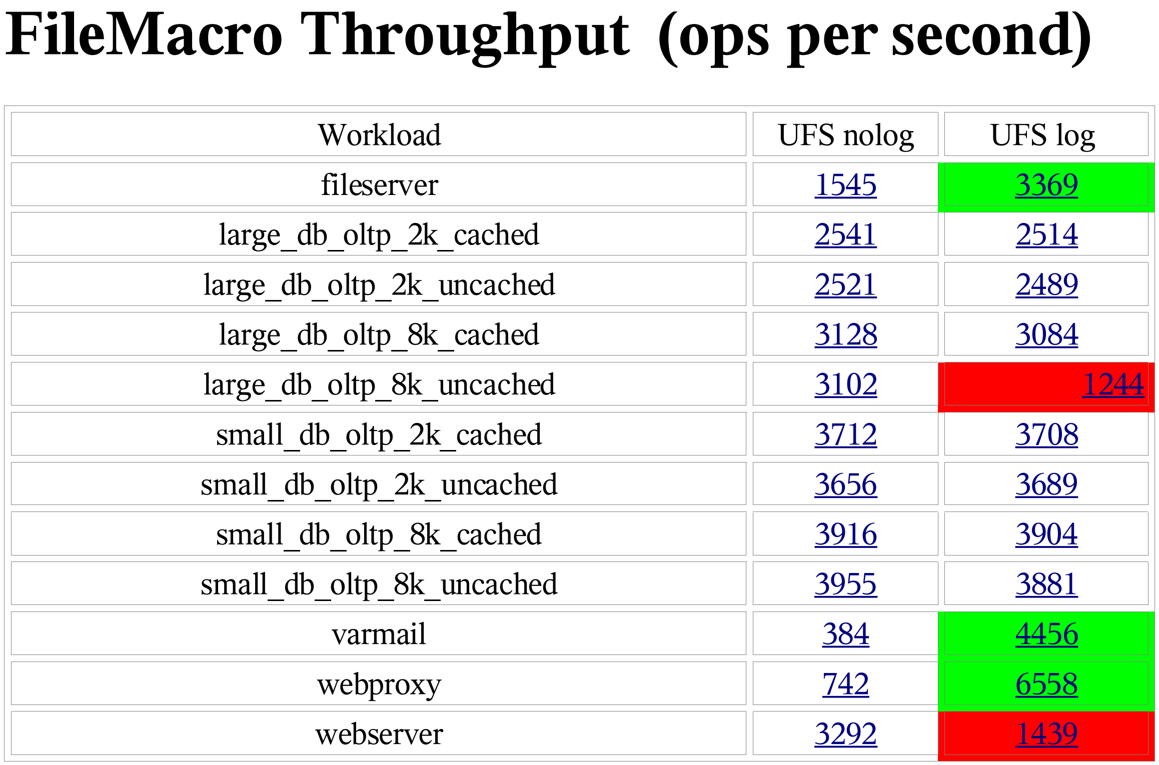
```
$  ls /opt/filebench/workloads
bringover.f                  filemicro_rwritefsync.f      postmark.f
copyfiles.f                  filemicro_seqread.f          randomread.f
createfiles.f                filemicro_seqwrite.f         randomwrite.f
deletefiles.f                filemicro_seqwriterand.f     singlestreamread.f
filemicro_create.f           filemicro_writefsync.f       singlestreamreaddirect.f
filemicro_createfiles.f      fileserver.f                 singlestreamwrite.f
filemicro_createfsyncrand.f  mongo.f                      singlestreamwritedirect.f
filemicro_createrand.f       multistreamread.f            tpcso.f
filemicro_delete.f           multistreamreaddirect.f      varmail.f
filemicro_rread.f            multistreamwrite.f           webproxy.f
filemicro_rwrite.f           multistreamwritedirect.f     webserver.f
filemicro_rwritedsync.f      oltp.f
```

“Benchpoint” Run Generation Wrapper

- A perl-based run environment
- Allows simple template-driven runs
- Can drive multiple configurations back to back from a single template
- Generates statistics, tabulates statistics

Running benchpoint...

```
Example filemacro run:

$ cp /opt/filebench/config/filemacro.prof myworkload.prof
$ vi myworkload.prof

    <edit directory, params etc...>

$ /opt/filebench/bin/benchpoint myworkload

.
.
.
.
.
$ browse stats/index.html
```



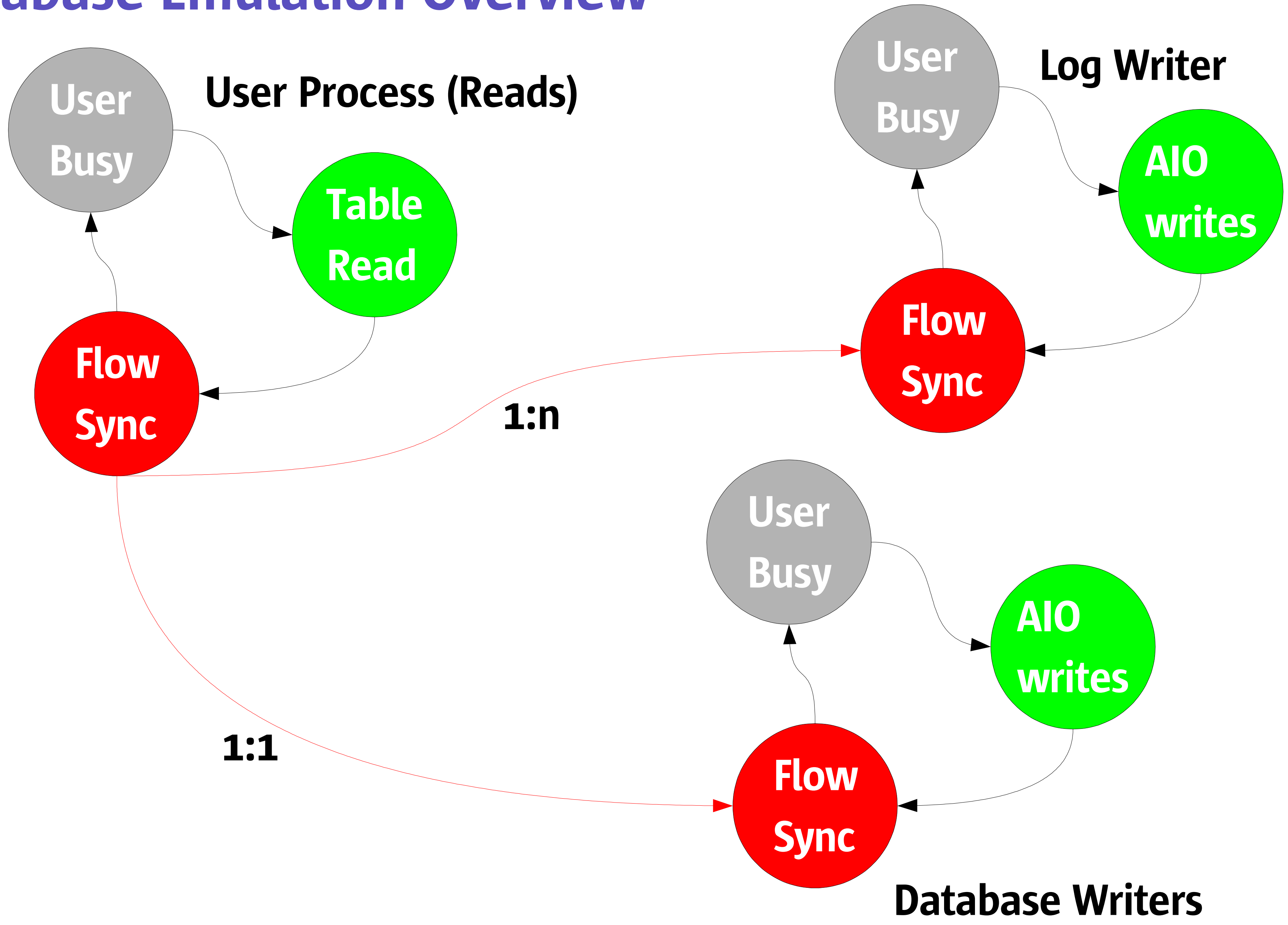
FileMacro Throughput (ops per second)

Workload	UFS nolog	UFS log
fileserver	1545	3369
large_db_oltp_2k_cached	2541	2514
large_db_oltp_2k_uncached	2521	2489
large_db_oltp_8k_cached	3128	3084
large_db_oltp_8k_uncached	3102	1244
small_db_oltp_2k_cached	3712	3708
small_db_oltp_2k_uncached	3656	3689
small_db_oltp_8k_cached	3916	3904
small_db_oltp_8k_uncached	3955	3881
varmail	384	4456
webproxy	742	6558
webserver	3292	1439

A sample profile

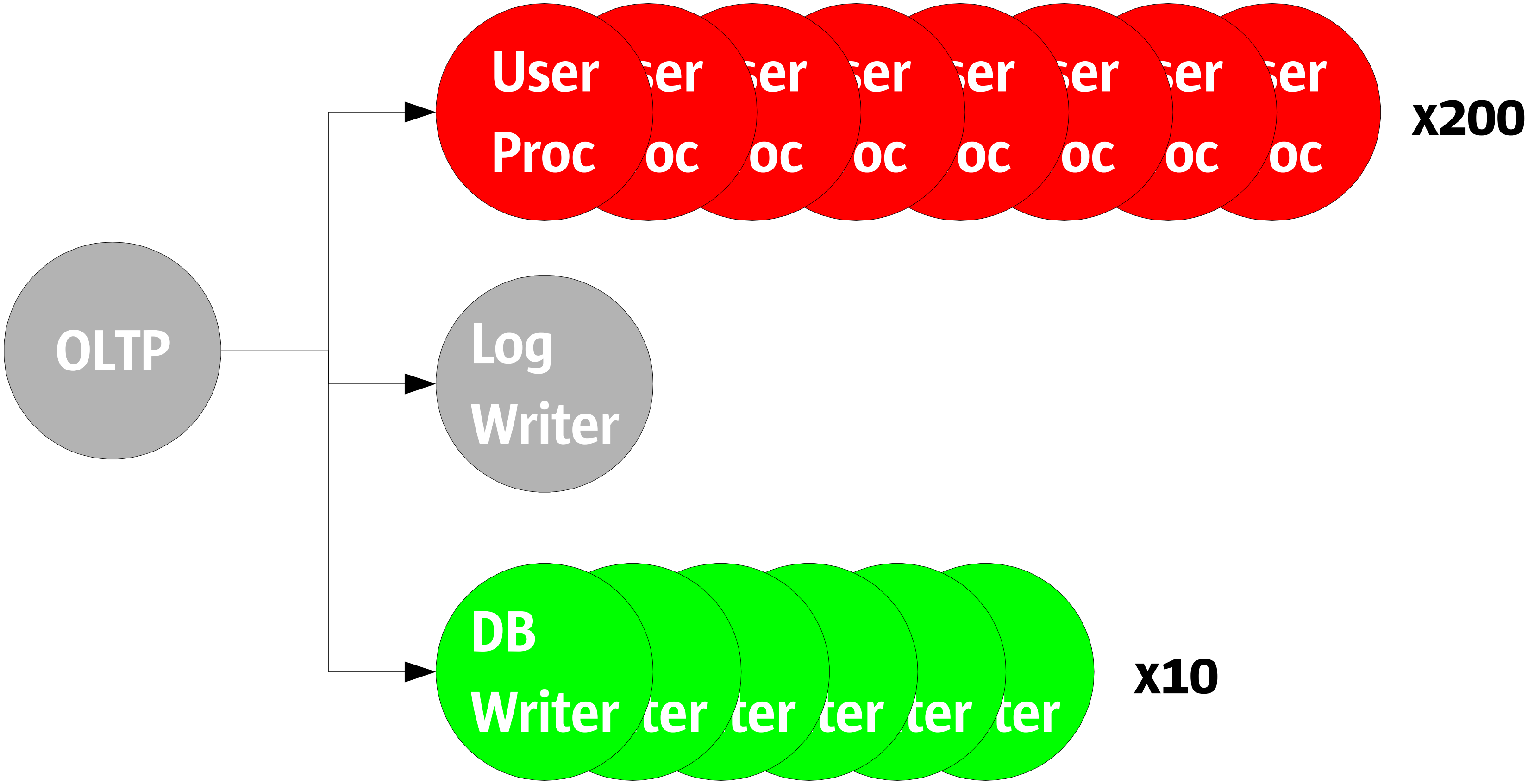
```
DEFAULTS {
        runtime = 120;
        dir = /filebench;
        stats = /home/rmc/filebench/stats;
        filesystem = zfs;
        description = "ZFS on Laptop";
}

CONFIG tiny_db {
        personality = oltp;
        function = generic;
        cached = 1;
        directio = 0;
        iosize = 8k;
        usermode = 20000;
        filesize = 10m;
        logfilesize = 10m;
        memperthread = 1m;
        workingset = 0;
}

CONFIG large_db {
        personality = oltp;
        function = generic;
        cached = 1;
        directio = 0;
        iosize = 8k;
        usermode = 20000;
        filesize = 10g;
        logfilesize = 1g;
        memperthread = 1m;
        workingset = 0;
}
```

Database Emulation Overview



Database Emulation Process Tree

User Process (Reads)



Simplified OLTP Database Program

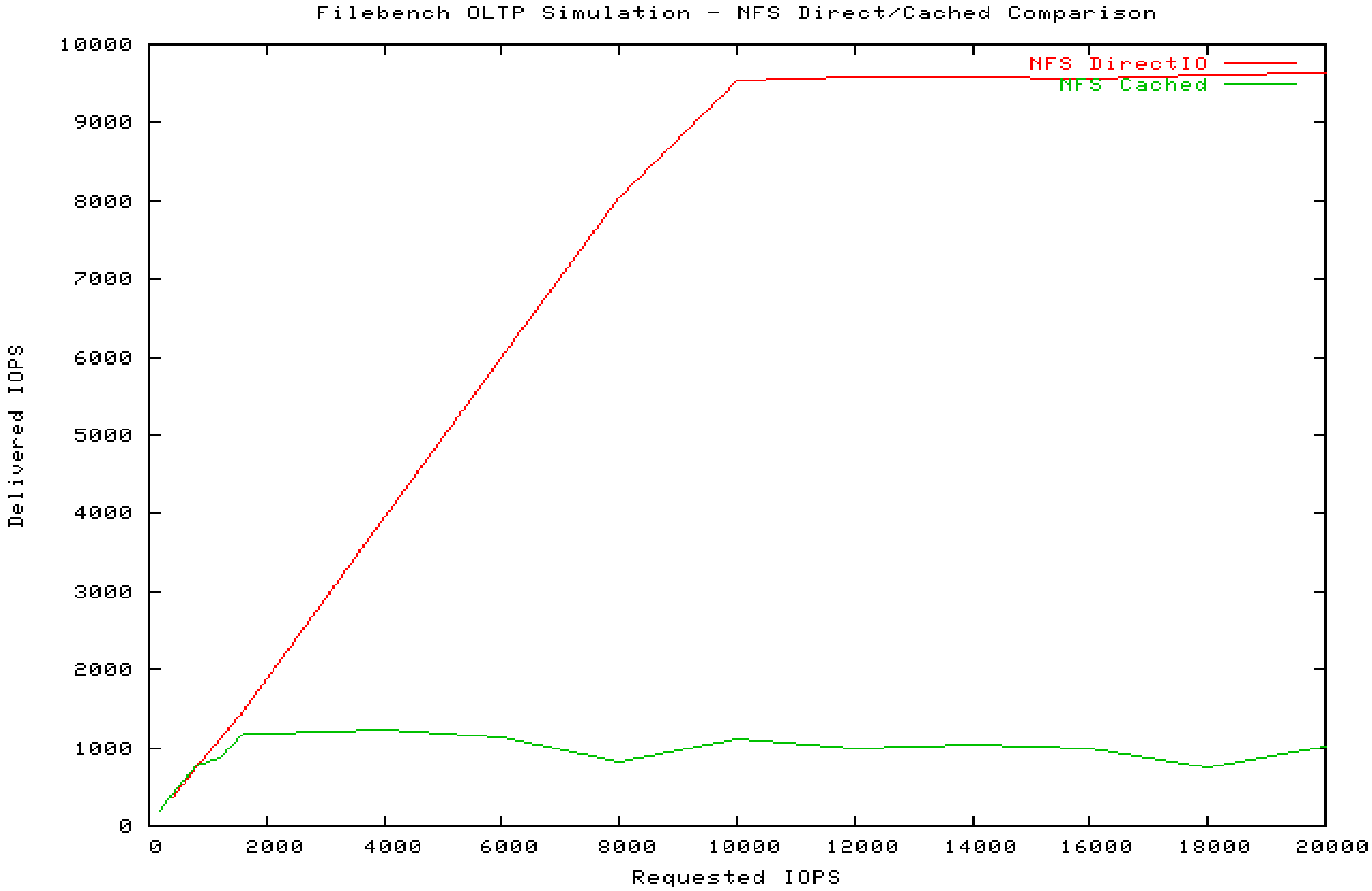
```
define file name=logfile,path=$dir,size=1g,reuse,prealloc,paralloc
define file name=datafilea,path=$dir,size=$filesize,reuse,prealloc,paralloc
define process name=dbwr,instances=$ndbwriters
{
 thread name=dbwr,memsize=$memperthread,useism
 {
  flowop aiowrite name=dbaiowrite-a,filename=datafilea,
    iosize=$iosize,workingset=10g,random,dsync,directio,iters=10
  flowop hog name=dbwr-hog,value=10000
  flowop semblock name=dbwr-block,value=100,highwater=10000
  flowop aiowait name=dbwr-aiowait
 }
}

define process name=lgwr,instances=1
{
 thread name=lgwr,memsize=$memperthread,useism
 {
  flowop write name=lg-write,filename=logfile,
    iosize=256k,workingset=1g,random,dsync,directio
  flowop semblock name=lg-block,value=320,highwater=1000
 }
}
define process name=shadow,instances=$nshadows
{
 thread name=shadow,memsize=$memperthread,useism
 {
  flowop read name=shadowread-a,filename=datafilea,
   iosize=$iosize,workingset=10g,random,dsync,directio
  flowop hog name=shadowhog,value=$usermode
  flowop sempost name=shadow-post-lg,value=1,target=lg-block,blocking
  flowop sempost name=shadow-post-dbwr,value=1,target=dbwr-block,blocking
  flowop eventlimit name=random-rate
 }
}
```

OLTP Program – Benchmark Result Detail

```
Flowop totals:

shadow-post-dbwr          4554ops/s   0.0mb/s    215.7ms/op       91us/op-cpu
shadow-post-lg            4555ops/s   0.0mb/s      0.7ms/op       21us/op-cpu
shadowhog                 4546ops/s   0.0mb/s      2.5ms/op      111us/op-cpu
shadowread                4455ops/s   0.9mb/s     23.2ms/op       89us/op-cpu
lg-block                   100ops/s   0.0mb/s    605.2ms/op      305us/op-cpu
lg-write                   100ops/s   0.4mb/s     96.2ms/op     1962us/op-cpu
dbwr-aiowait              4445ops/s   0.0mb/s    144.0ms/op      242us/op-cpu
dbwr-block                4445ops/s   0.0mb/s      9.6ms/op       44us/op-cpu
dbwr-hog                  4445ops/s   0.0mb/s      1.1ms/op       50us/op-cpu
dbaiowrite                4449ops/s   0.9mb/s      0.2ms/op       17us/op-cpu
```

IO Summary: 9087.7 ops/s, 4547/4496 r/w   18.0mb/s,   129uscpu/op

NFS OLTP – IOPS Scaling



Important Workloads

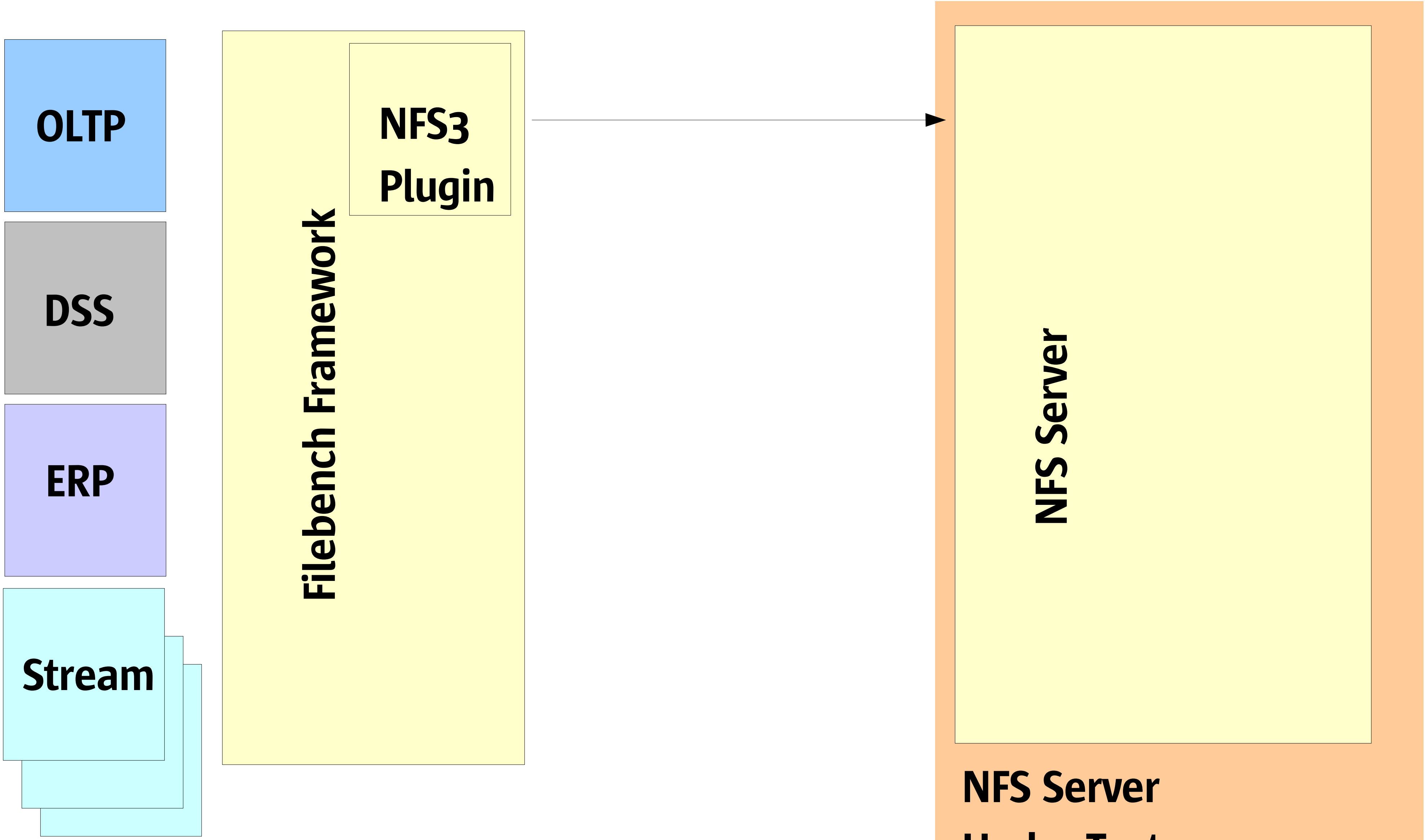
	File Access						
Workload	**File Size**	**# files**	**#Streams**	**Sharing**	**I/O Mix**	**Seek Mode**	**Access type mmap/posix**
Web Server	Small	Large	Large	Low	<5%	Random Read/10% Sequential Write	Both
Small DB	Large	Small	~100	High	50r/50w, 1% large sequential	99% Random	POSIX
Large DB	Large	Small	~1000	High	50r/50w, 1% large sequential	99% Random	POSIX
DB Mail Server	Large	Small	>1000	High	?		
NFS Mail Server	Moderate	Moderate	>10k	Low	?	Sequential	POSIX
HPTC	Huge	Small	Small	Low	50r/50w	Sequential	POSIX
SW Development	Small	Large	>1000	Low	5r/5w/90a	Sequential	POSIX
Video Streaming							

	I/O Characteristics					
Workload	**App/IO CPU Content**	**Typical IOPS**	**Data Set Size**	**Working Set Size**	**Typical I/O Size**	**Typical Bandwidth**
Web Server	99/1	<1000 per cient			<64k	<1MB/s
Small DB	90/10	~1000	1-10GB	50.00%	Random 2-8k, 128k sequential	~10MB/s
Large DB	80/20	>10000	10GB-1TB	30.00%	Random 2-8k, 128k sequential	50MB/s
DB Mail Server	90/10?				Small?	?
NFS Mail Server	90/10?	Low			Large reads, small writes	1-10MB/s
HPTC	80/20?	~1000?			~1MB	>100MBs Client, 1GB/s Server
SW Development	95/5?	~1000			~32k	~100mb/s

FileBench Pre-defined Workloads

- “File Macro”
  - Small Database
  - Large Database
  - Multi-threaded web server
  - Multi-threaded proxy server
  - Home directory server
  - NFS Mail Server (postmark)
  - DB Mail Server
  - Video Server
- “File Micro”
  - Sequential Read/Write
  - Multistream Read/Write
  - Allocating Writes
  - Reallocating Writes
  - Random Read/Write
  - MT Random Read/Write
  - File Create/Delete
  - File meta-data ops
  - I/O Types: O_DSYNC etc
  - Directory size scaling

Future: NFS Plugin



FileBench Status

- Porting Status
  - Completed: S8, 10, x86, SPARC, Linux
  - Binary packages for Solaris 8/9/10 for x86/SPARC avail.
- FileBench is Open Source
  - See opensolaris.org performance community
- Future Activities
  - Complete linux + other ports
  - Add support for C based workload plugins
  - Refine, develop workloads
  - Add multiple-client support
  - Develop NFS plugin

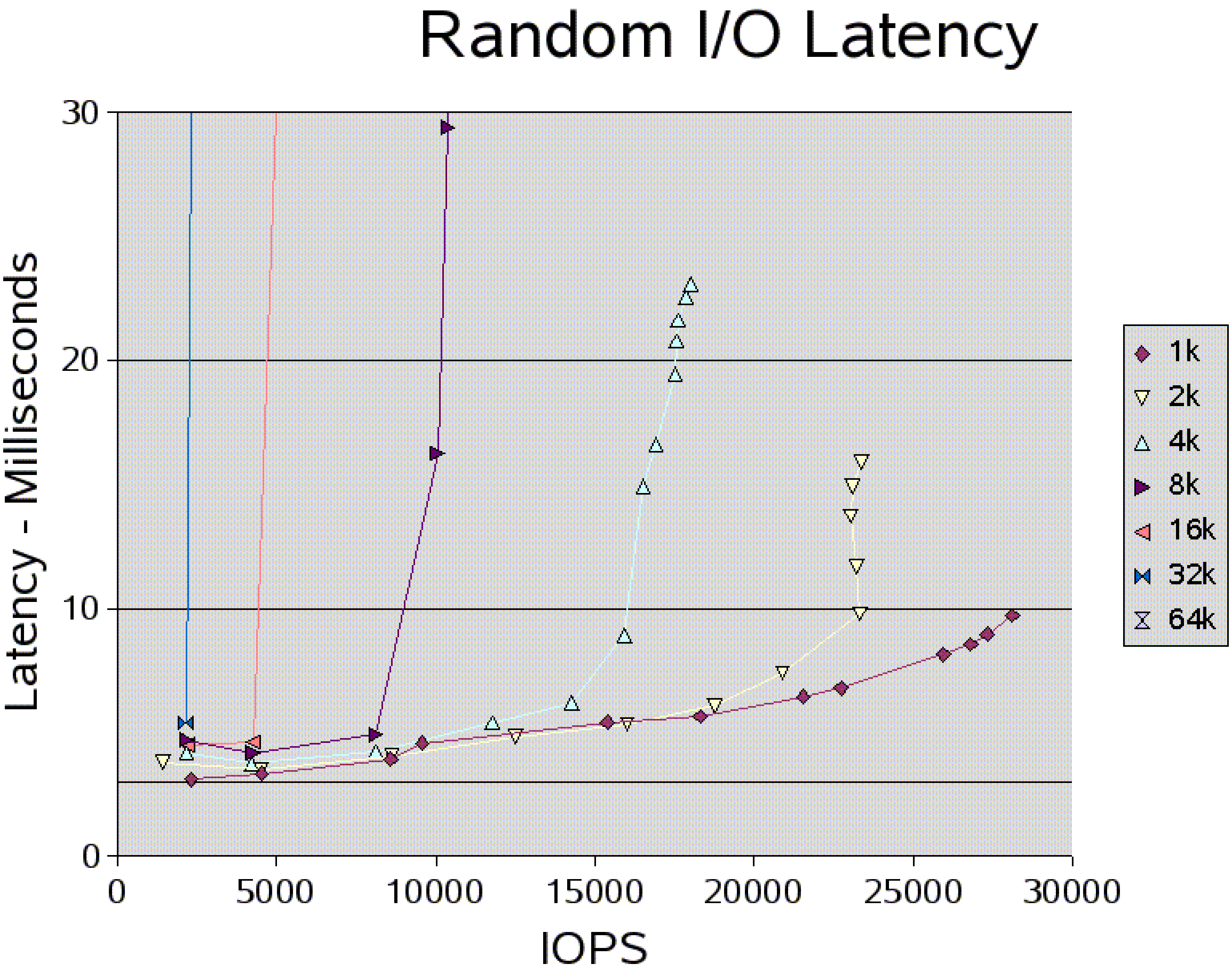
FileBench

r@sun.com

Sun microsystems

We make the net work.

Random I/O – NFS V3


Random I/O Latency
Latency - Milliseconds
IOPS
1k
2k
4k
8k
16k
32k
64k