UNIVERSITY OF NOTRE DAME

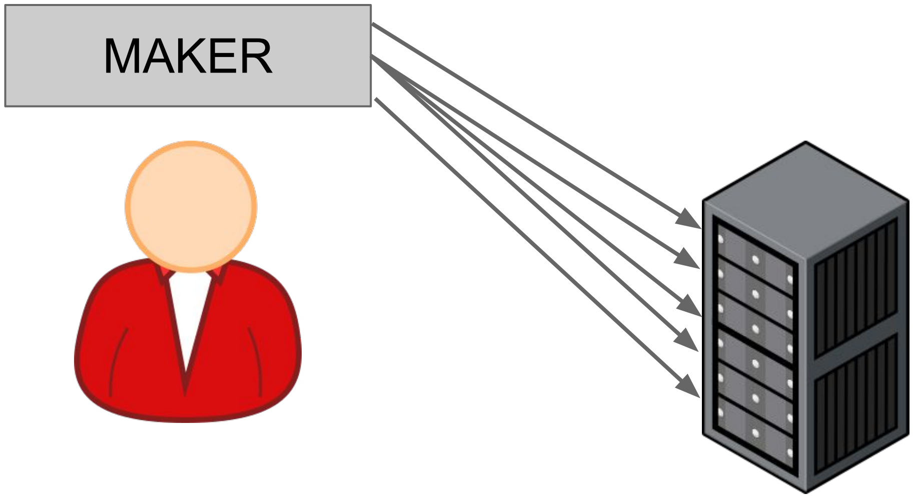
Taming Metadata Storms in Parallel Filesystems with MetaFS

CCTools

Tim Shaffer

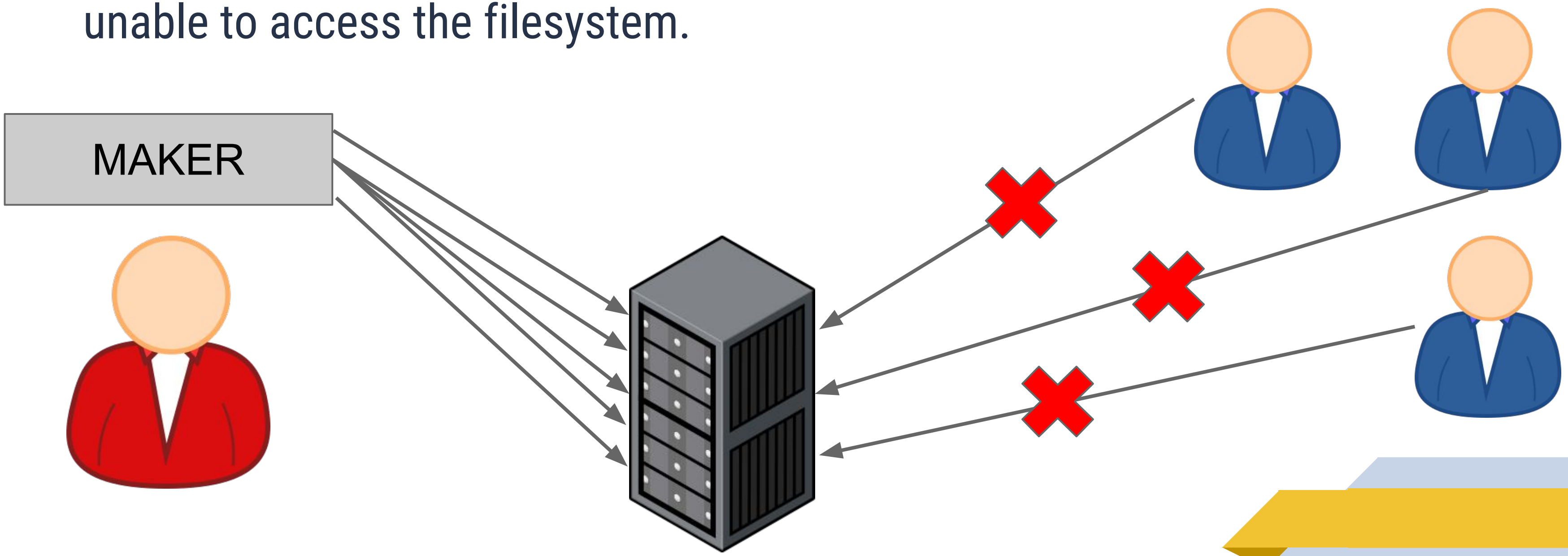
Motivation

A (well-meaning) user tried to run a bioinformatics pipeline to analyze a batch of genomic data.



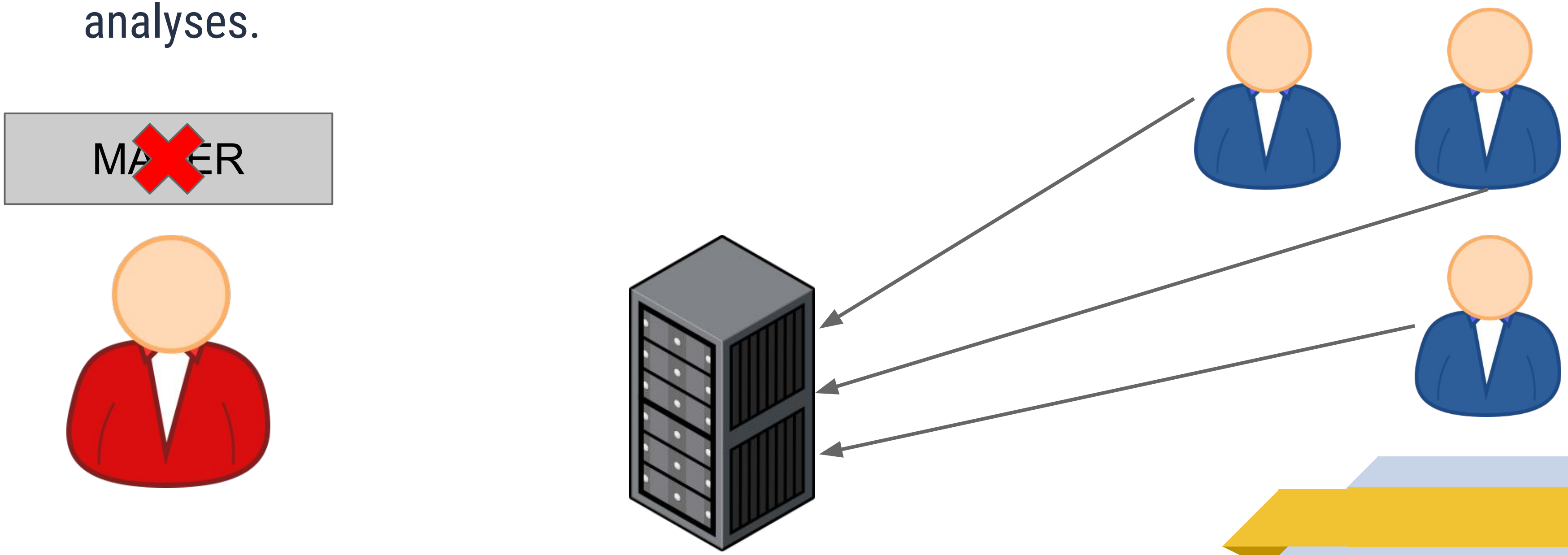
Motivation

Shared filesystem performance became degraded, with other users unable to access the filesystem.



Motivation

That user got a strongly worded email and had to stop their analyses.



Metadata Storm

Certain program behaviors produce **large bursts** of metadata I/O activity (e.g. library search).

These behaviors can occur at the **same time across multiple workers** (e.g. startup, new analysis phase).

With a large number of nodes, the timing and intensity of metadata activity align to **overwhelm the shared FS**.

Existing Approaches/Related Work

Shared filesystems can scale up their metadata capacity.

Panasas, Ceph, etc. use multiple metadata servers to better distribute the load.

General purpose solution

Existing Approaches/Related Work

Applications can use a metadata service layered on top of the shared filesystem (e.g. BatchFS, IndexFS).

More efficient metadata management than the native filesystem.

Allows for client-side caching and batch updates.

Existing Approaches/Related Work

Changes to the filesystem interface that allow weaker consistency or bulk operations

`statlite` and `getlongdir` system calls are examples.

This approach is not widely implemented.

Existing Approaches/Related Work

Spindle provides library loading as a service.

Hooks into the dynamic loader on each node and builds an overlay network.

Nodes load shared objects by contacting each other rather than reading from the shared FS every time.

Case Study: MAKER

MAKER is a bioinformatics pipeline for analyzing raw gene sequence data.

It builds an annotated genome database with information on sequence repeats, proteins, etc.

http://www.yandell-lab.org/software/maker.html

Case Study: MAKER

MAKER presents a number of challenges at scale

- Large number of software dependencies (OpenMPI, Perl 5, Python 2.7, RepeatMasker, BLAST, several Perl modules)
- Composed of many sub-programs written in different languages (Perl, Python, C/C++)
- Installation consists of 21,918 files in 1,757 directories
- Unusual metadata load on shared filesystems
- Prone to causing a metadata storm

Profiling MAKER's I/O Behavior

CCTools

To help identify the causes of MAKER's performance issues, we used `strace` to record syscalls made during an analysis.

For each syscall, we captured the type, timestamp, and paths/file descriptors used.

We also `strace`d all children to capture sub-programs.

Profiling MAKER's I/O Behavior

```
18212 1503501245.079960 read(3</lib64/libpthread-2.12.so>,
"\x7f\x45\x4c\x46\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x03\x00\x3e\x00\x01\x00\x00\x00"..., 832) = 832
```

Profiling MAKER's I/O Behavior

Grouped relevant syscalls as

- data (`read`, `readv`, `write`, ...)
- metadata (`stat`, `readdir`, `readlink`, `open`, ...)

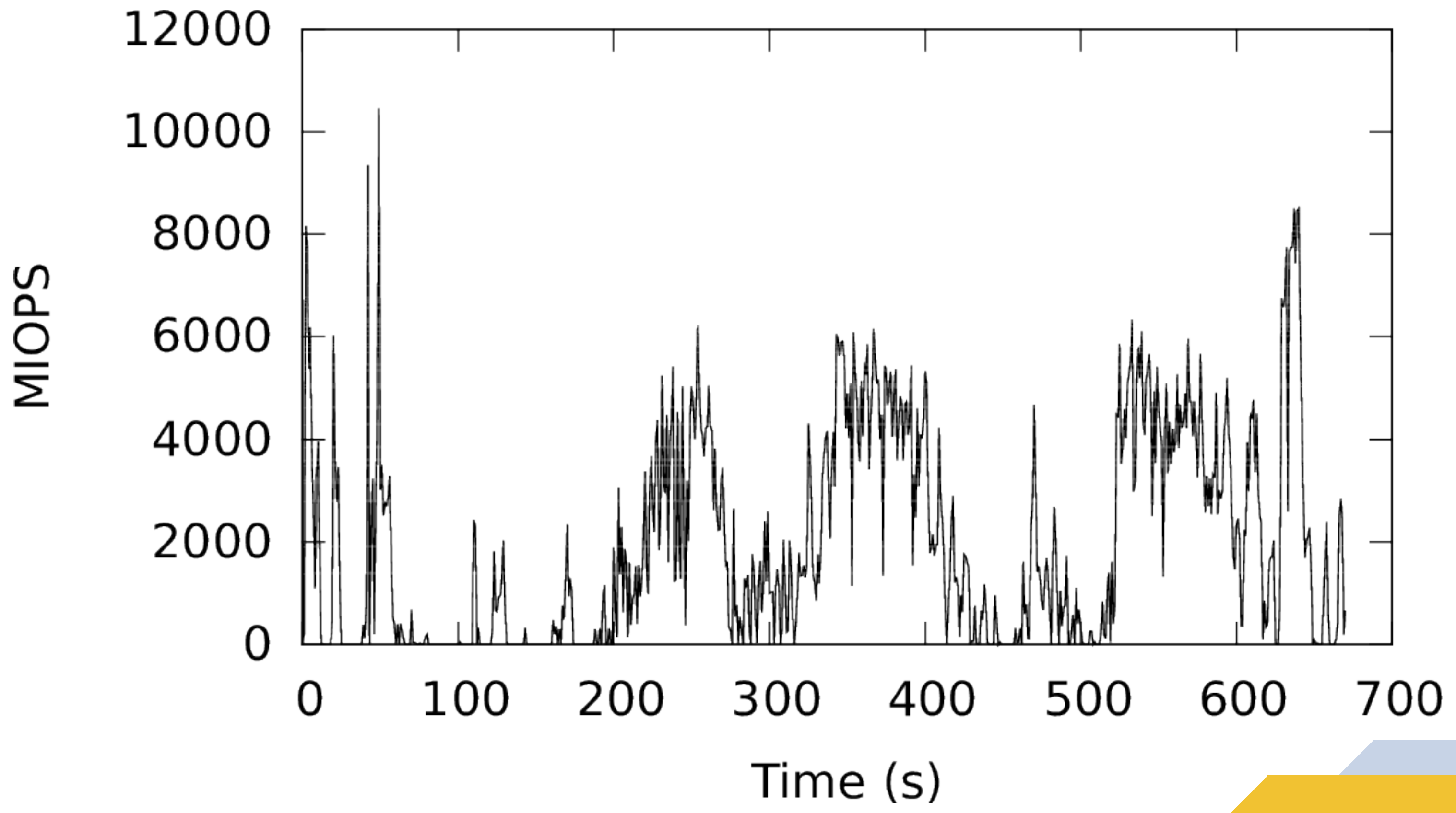
and by location

- Working directory (CWD)
- `/tmp`
- Shared FS
- Local system (`/bin`, `/usr/...`)

I/O Activity by Filesystem Location

	Access Mode	I/O Ops	Bandwidth (B)
CWD	RW	257,060	1,435,228,808
`/tmp`	RW	1,163,711	2,463,335,142
Shared FS	RO	1,512,545	2,807,495,139
Local System	RO	906,327	68,929,672

Single-instance Metadata I/O



Metadata Performance

As suspected, MAKER causes large bursts of metadata activity.

Intermediate and output data contribute relatively little to metadata activity over the course of an analysis.

Largest contributor is **subprogram startup/library loading**.

Shared Filesystem Performance

Panasas ActiveStor 16 filesystem

- 7 Director Blades + 70 Storage Blades
- Up to 84 Gb/s read bandwidth
- Up to 94,000 IOPS while reading data

We used a synthetic benchmark (`ls -r` in a directory tree with 74,256 files and 4,368 directories) to measure pure metadata performance.

Running Times for Parallel Benchmark Instances

Parallel Instances	Instance Running Time (s)	Total Metadata I/O Operations	Average FS MIOPS
1	13.7	179,091	13,038
4	22.6	716,364	31,664
8	41.9	1,432,728	31,194
16	86.1	2,865,456	33,262
24	130.6	4,298,184	32,916

Possible Solutions

To reduce shared FS load, we considered

- Local installation
- Disk image
- Containers (Docker, Singularity, ...)
- Filesystem overlay

These depend on availability at the site.

Idea: Metadata Index

Software installation does not change during an analysis.

We can index the software installation metadata.

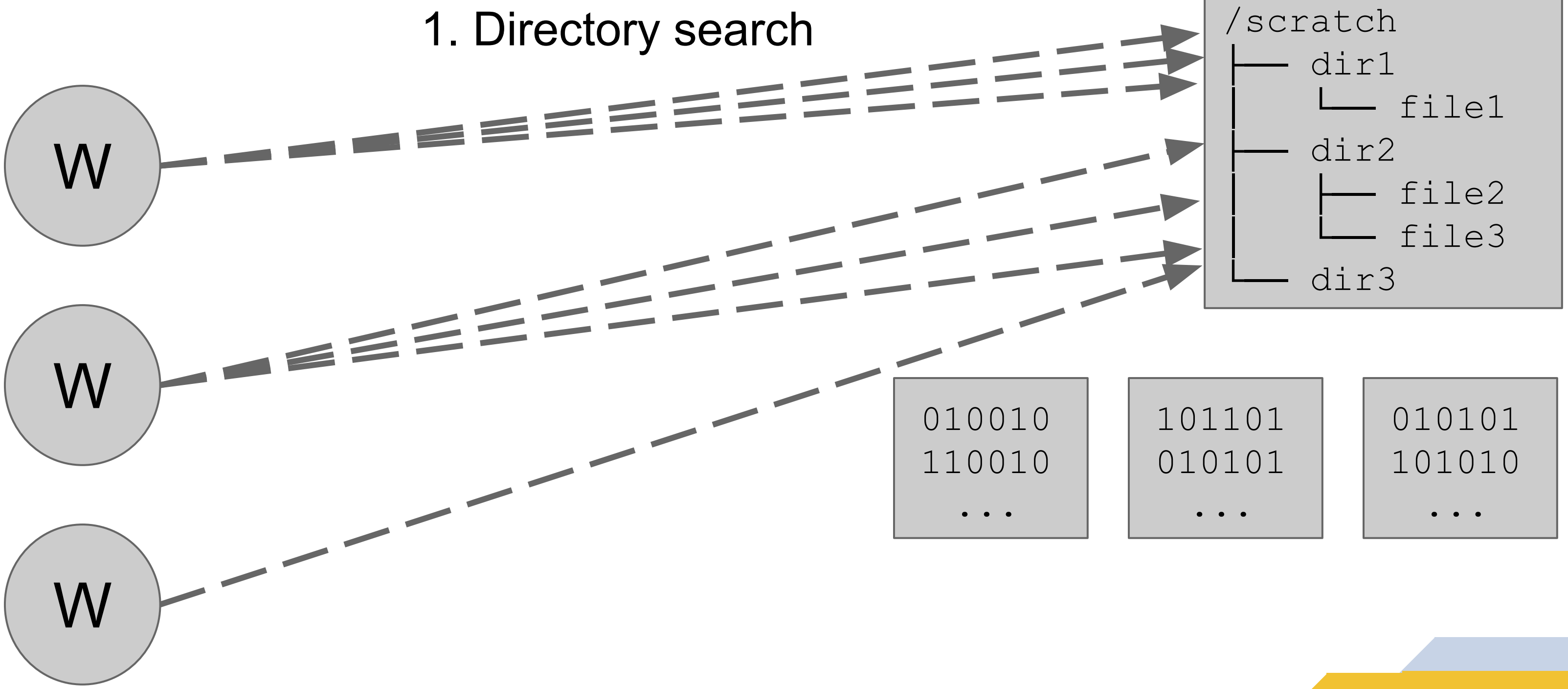
- Trade **numerous metadata operations** for a **single file read**
- Library is search handled locally

MetaFS

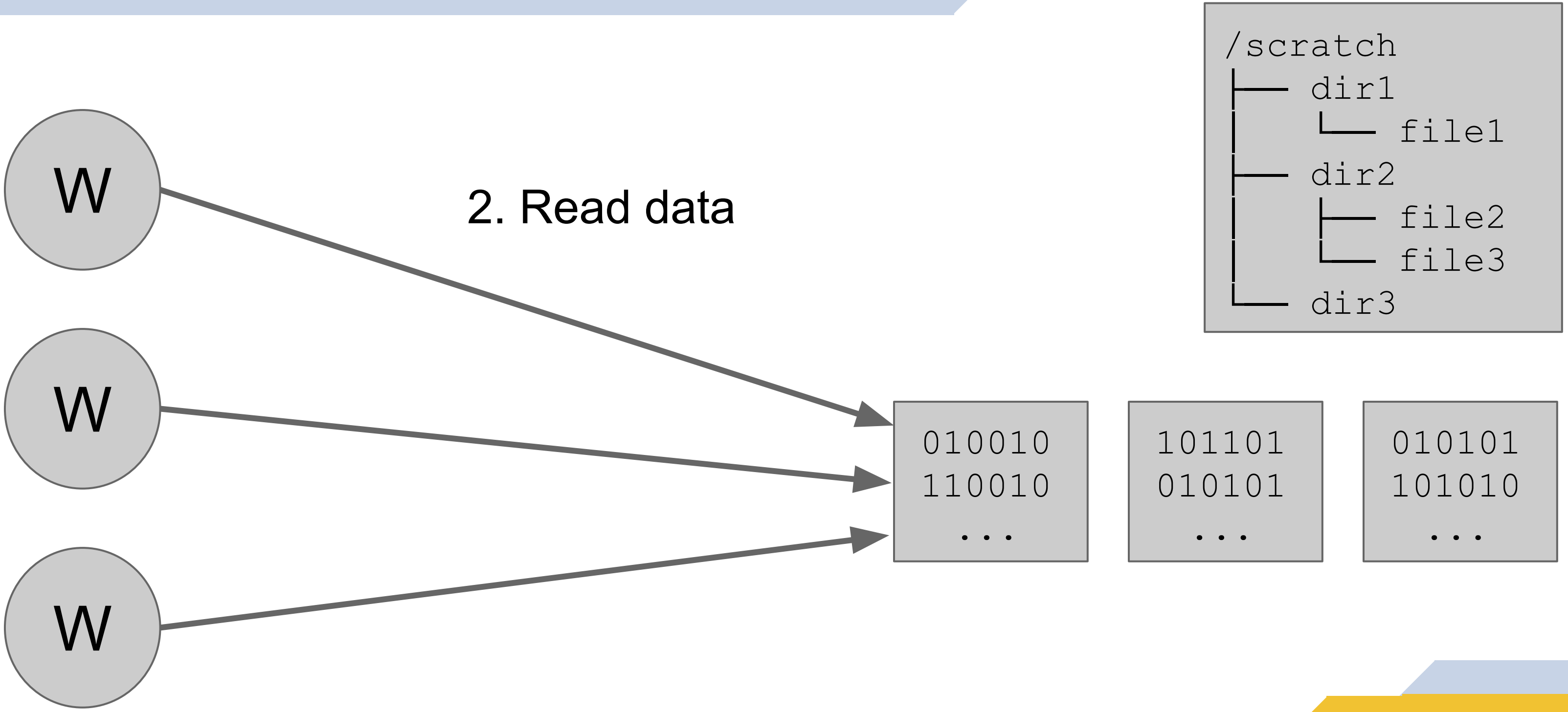
We implemented MetaFS as a FUSE module for evaluating this approach.

- Transparent overlay applied to an existing directory
- Easy to add/remove without modifying your scientific app
- Reads metadata index at startup and presents a read-only view of the software installation

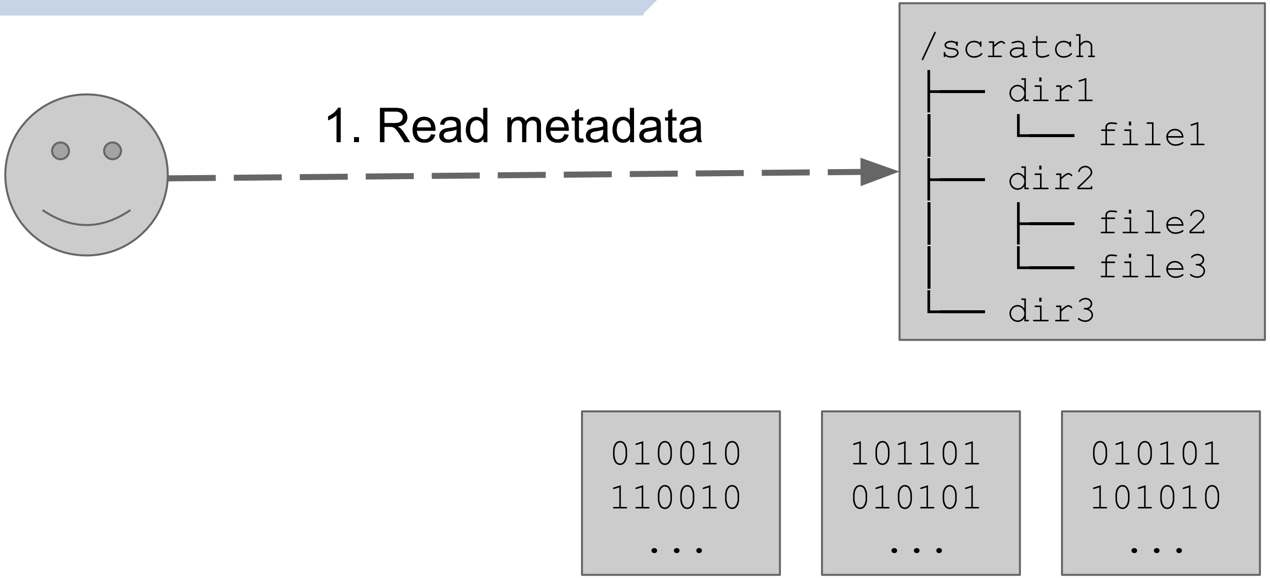
Normal Access



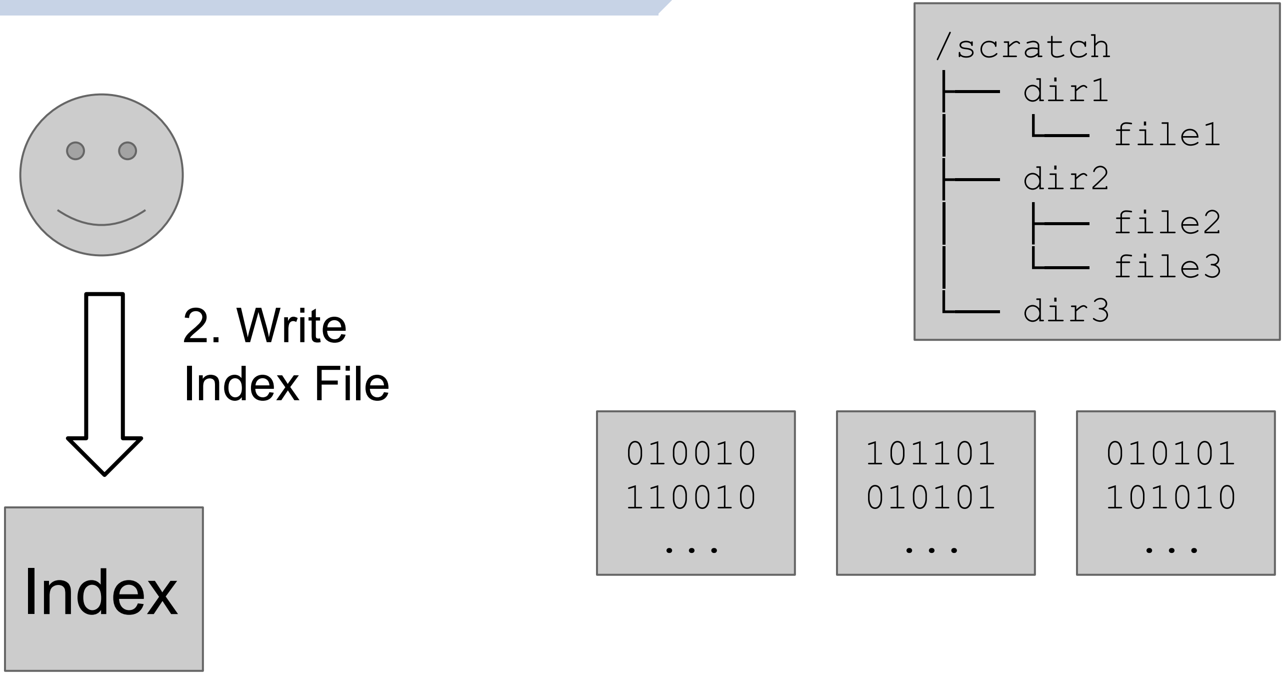
Normal Access



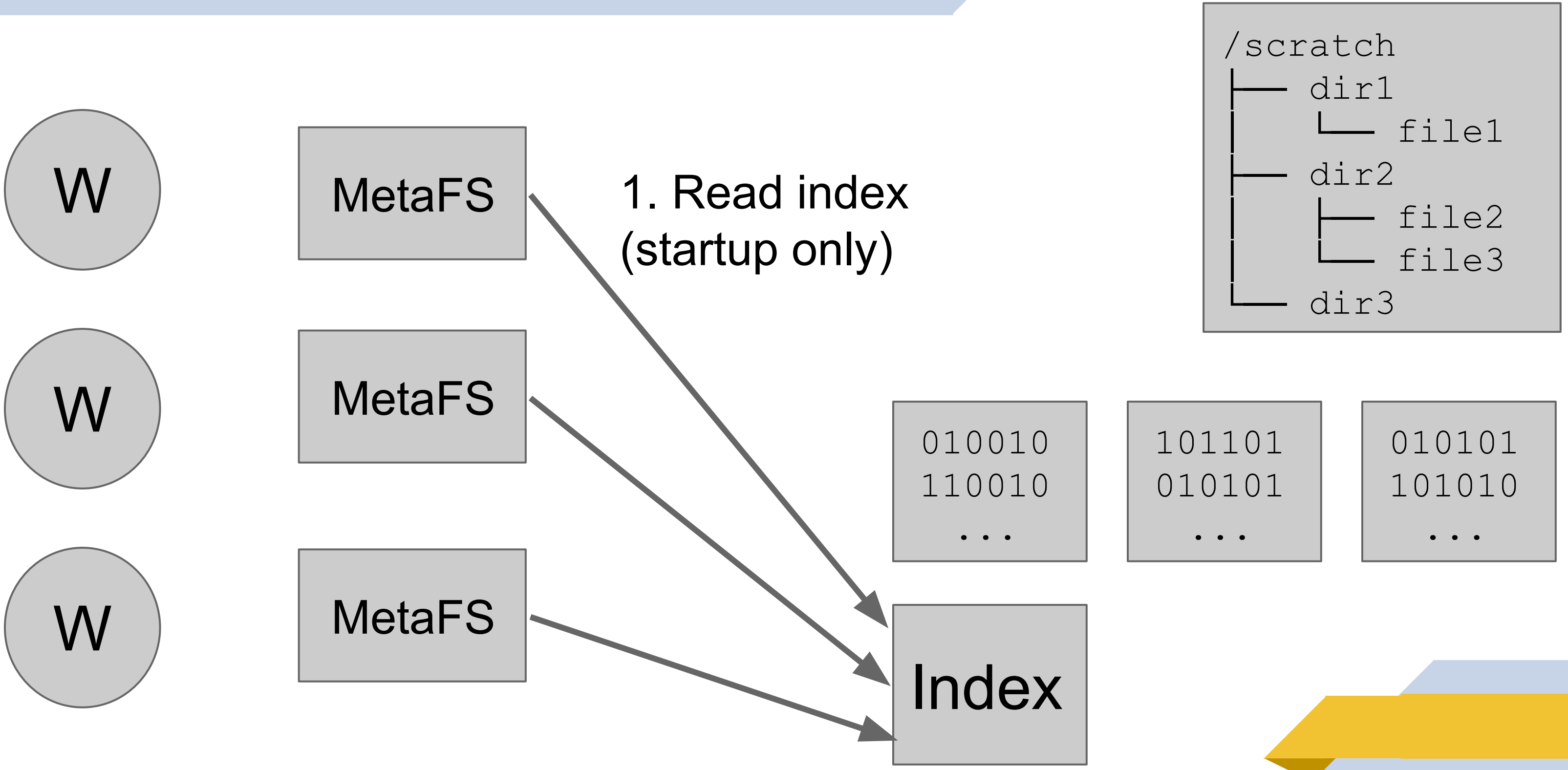
Create Index



Create Index

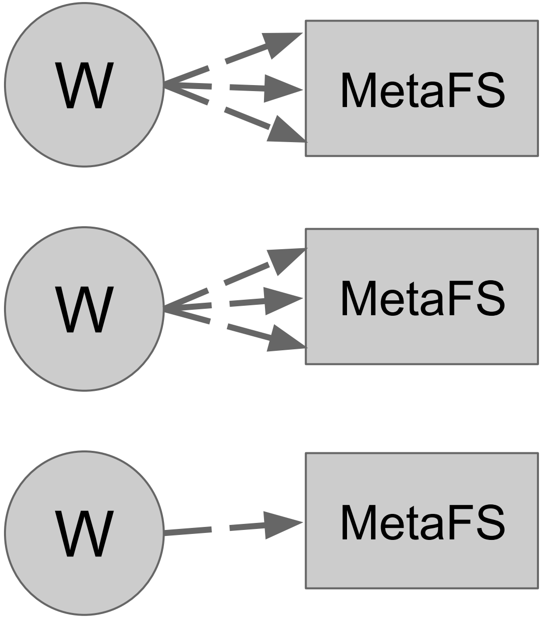


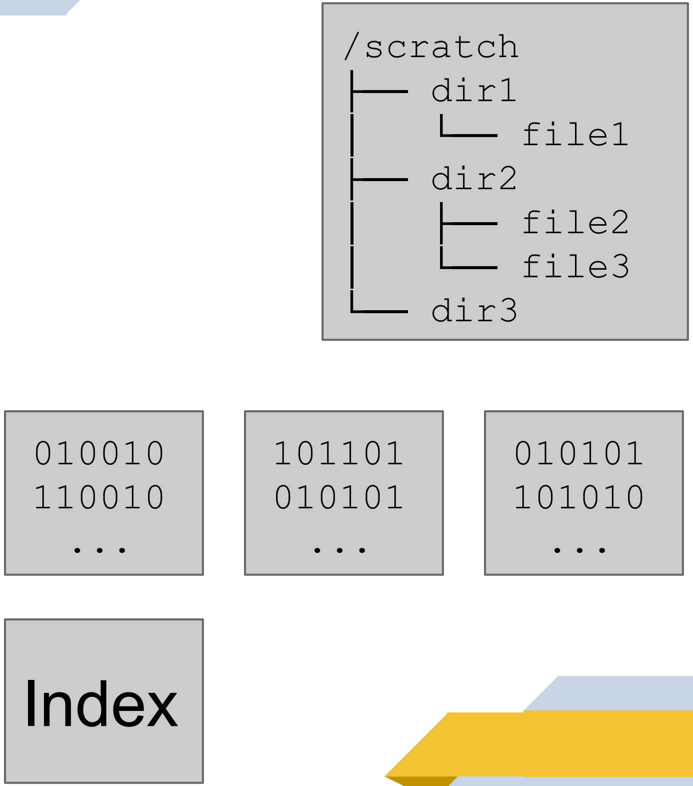
Using MetaFS



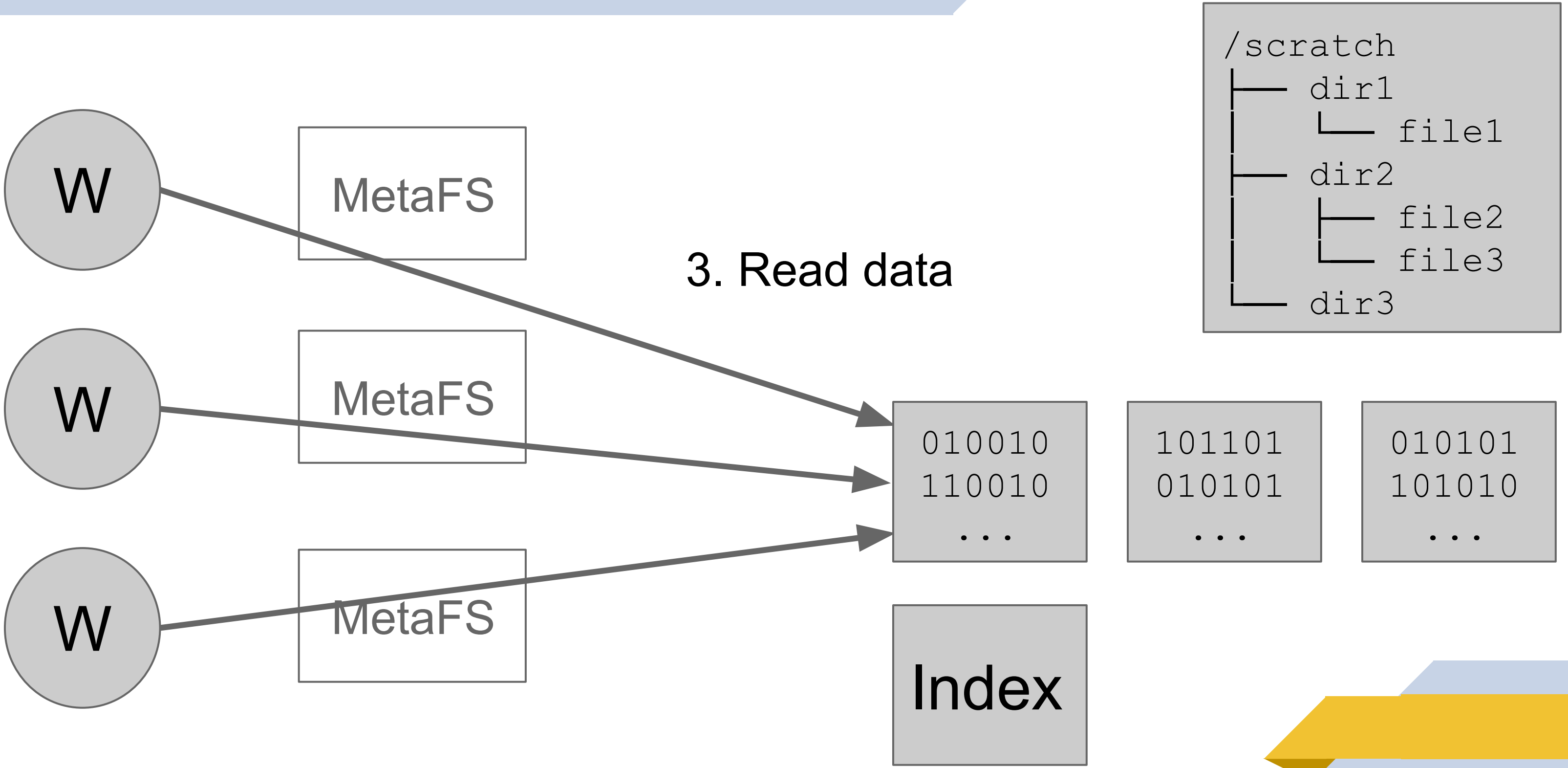
Using MetaFS

2. Directory search





Using MetaFS



Evaluation

For the `ls` benchmark with MetaFS in place, running time was on par with single-instance performance regardless of the number of parallel instances.

We also ran MAKER with MetaFS in place over the software installation directory.

MAKER requires **no modification** to run with MetaFS.

Evaluation

When starting, MetaFS reads the index file (~2 MB for MAKER's installation directory).

Metadata activity to the shared FS is **significantly reduced** at the cost of a **small increase in data transfer** (index file).

No observed performance decrease due to FUSE.

Reduction in Metadata Load on the Shared Filesystem with MetaFS

	Metadata Ops.	**Data Transfer (B)**
`ls`	179,091	0
`ls` **+ MetaFS**	8,738	4,900,655
MAKER	1,142,781	2,807,495,139
MAKER + MetaFS	14,726	2,809,472,114

Scalability of MAKER

Based on the number of I/O ops. and the measured capacity of the system, a single user would saturate the shared FS with an average of **66 instances of MAKER** running in parallel.

Bursty activity could reduce this limit further.

With MetaFS in place, we can remove this limit, allowing an estimated **5,000 parallel instances** (✱).

Conclusions

MetaFS significantly reduces the (often unnecessary) metadata I/O encountered during program startup.

Local indexing is a lightweight approach: no changes to application or infrastructure necessary.

A major challenge for users is identifying when to apply optimizations. This is easy for software installations.

CCTools

Tim Shaffer
tshaffe1@nd.edu
github.com/trshaffer

UNIVERSITY OF NOTRE DAME