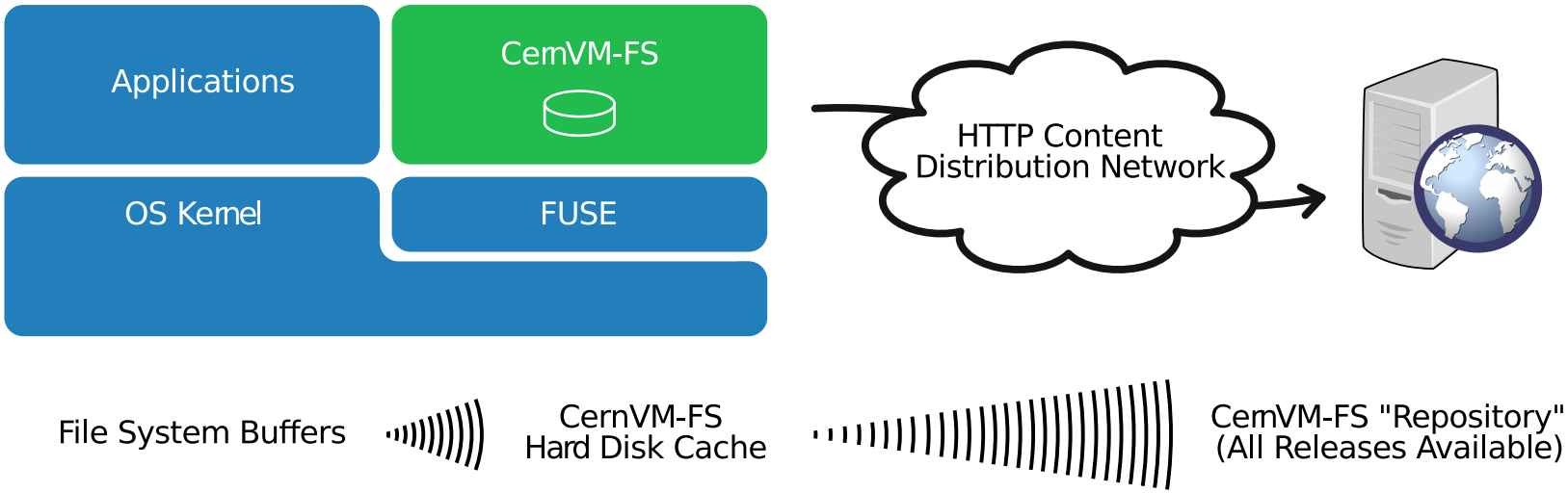
HEP Application Delivery on HPC Resources

Tim Shaffer[1]
Supervisors: Jakob Blomer[2], Gerardo Ganis[2]

CERN openlab Summer Student Program
15 August 2016

1. University of Notre Dame
2. CERN EP-SFT

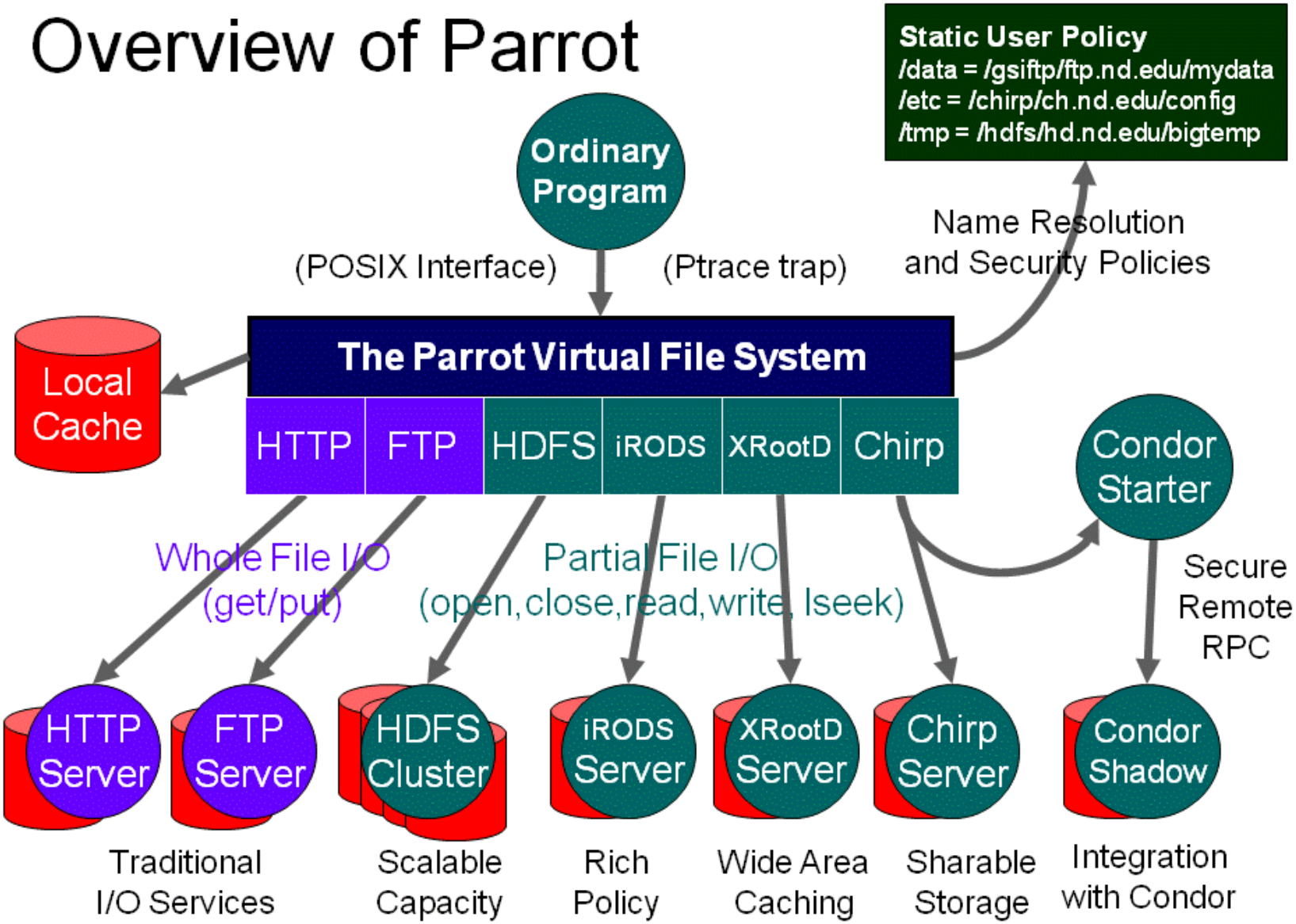

Applications
CernVM-FS
OS Kernel
FUSE
HTTP Content
Distribution Network
File System Buffers
CernVM-FS
Hard Disk Cache
CernVM-FS "Repository"
(All Releases Available)

On high-performance computing (HPC) resources,

- workers run with restricted permissions
- FUSE is not allowed
- worker nodes have limited/no internet connectivity
- filesystem access is read-only

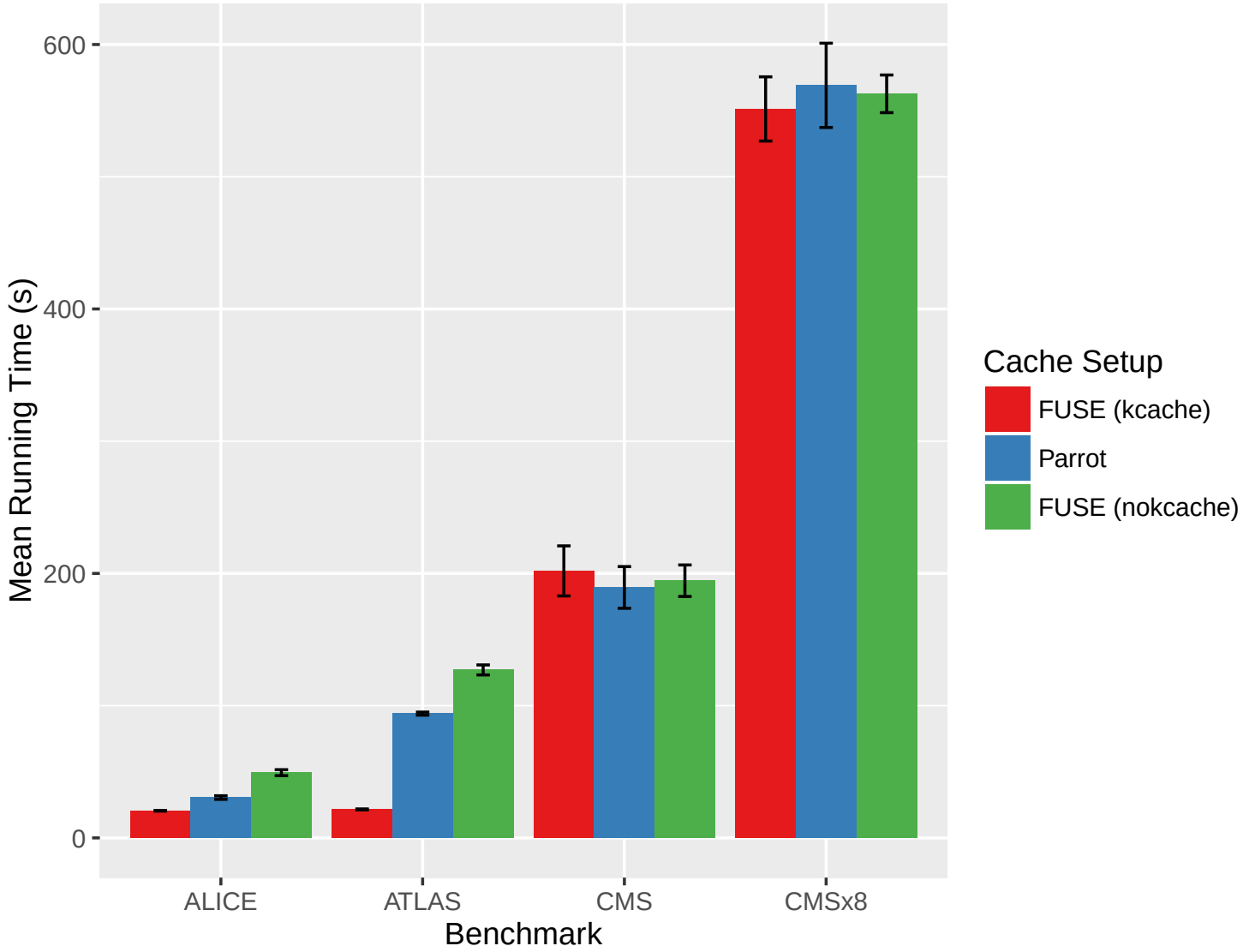
CVMFS now includes a RAM-backed cache manager:

- Adds in-memory caching for objects/metadata to CVMFS
- Designed to work with other storage backends (e.g. RamCloud)
- Customizable memory allocation
- Uses a reference counted key/value store internally
- Evicts entries in LRU order


Overview of Parrot
Static User Policy
/data = /gsiftp/ftp.nd.edu/mydata
/etc = /chirp/ch.nd.edu/config
/tmp = /hdfs/hd.nd.edu/bigtemp
Ordinary Program
(POSIX Interface)
(Ptrace trap)
Name Resolution and Security Policies
Local Cache
The Parrot Virtual File System
HTTP
FTP
HDFS
iRODS
XRootD
Chirp
Condor Starter
Whole File I/O (get/put)
Partial File I/O (open,close,read,write, lseek)
Secure Remote RPC
HTTP Server
FTP Server
HDFS Cluster
iRODS Server
XRootD Server
Chirp Server
Condor Shadow
Traditional I/O Services
Scalable Capacity
Rich Policy
Wide Area Caching
Sharable Storage
Integration with Condor

- Traps a program's syscalls via `ptrace` interface
- Runs most programs without modification
- Exposes remote I/O services (including CVMFS) in the filesystem
- No special privileges or kernel changes

Experiment benchmarks using RAM cache with arena allocator



Read a file of a given size from disk

	4 KB	128 KB
Native	4.902 GB/s	7.01772 GB/s
Parrot	203 MB/s	2.9 GB/s
Slowdown	$\sim 25x$	$\sim 2.4x$

Pass a shared memory object to another process via socket

	4 KB	128 KB	1 MB
Native	208 MB/s	1.81 GB/s	1.78 GB/s
Parrot	29.9 MB/s	582 MB/s	1.58 GB/s
Slowdown	$\sim 7x$	$\sim 3.2x$	$\sim 1.1x$

A RAM cache manager is implemented and merged into CVMFS

TODO:

- Shared local RAM cache
- RamCloud support
- More efficient memory allocation (possibly log structured)

Thank you

Currently,

1. Login node preloads CVMFS cache to a shared cluster filesystem
2. Nodes use local caches to avoid drowning the shared filesystem

Won't work without

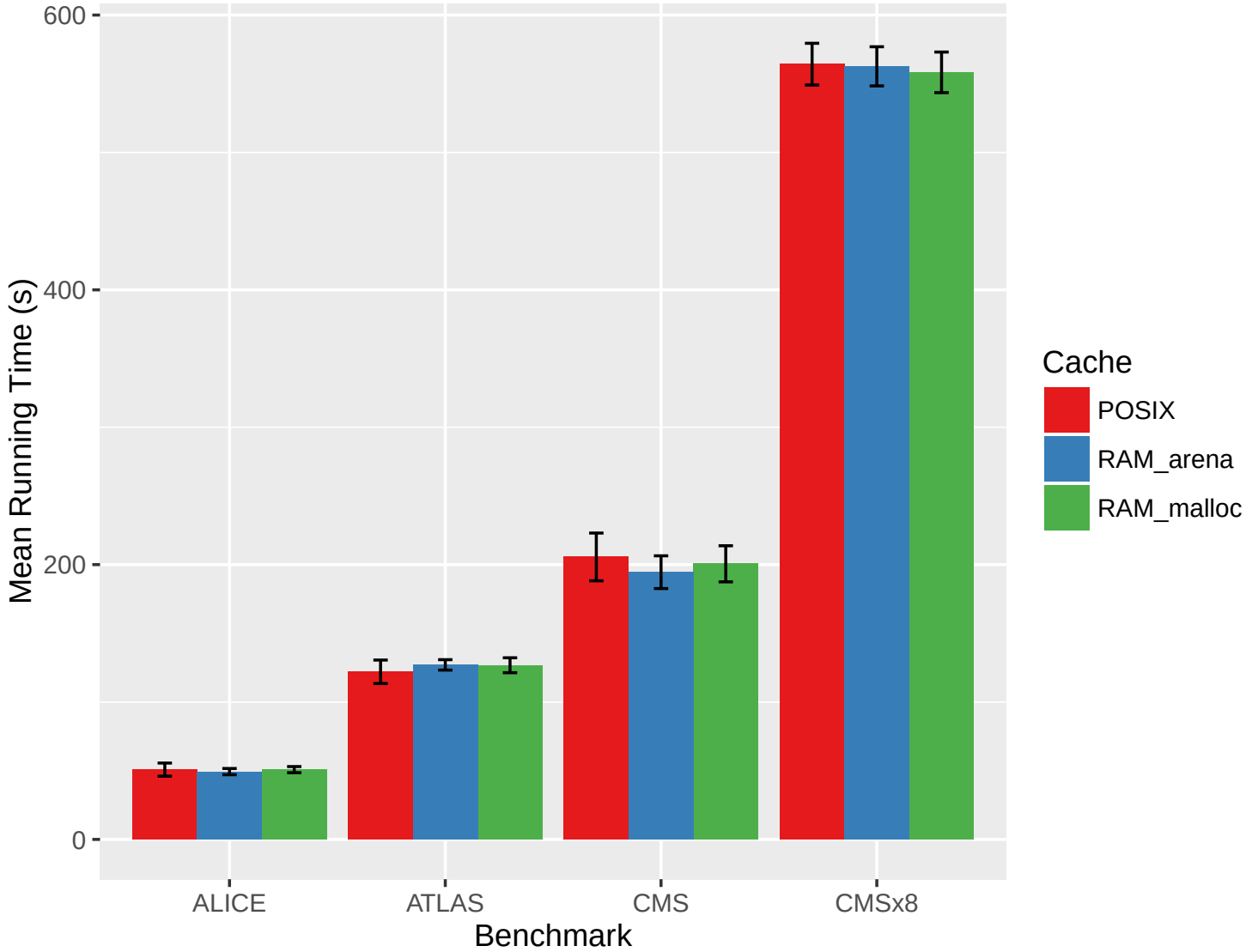
- an existing shared cluster filesystem
- writable storage attached to each node

Alternative Cluster Caching

Alternatively, serve preloaded cache data via an HTTP proxy

- Nodes use *in-memory* cache
- Minimal requirements on cluster infrastructure
- FUSE module requires privileges on worker nodes

Experiment benchmarks using the available cache implementations



Open & close or stat a file

	File open/close	Stat
Native	430k/s	1.221M/s
Parrot	13.2k/s	27.68k/s