XtreemFS: high-performance network file system clients and servers in userspace

Minor Gordon, NEC Deutschland GmbH

mgordon@hpce.nec.com

Why userspace?

- File systems traditionally implemented in the kernel for performance, control
- Some advantages doing things in userspace:
  - High-level languages: Python, Ruby, et al. for prototyping, then C++ (→ tool support, reduced code footprint, etc.)
  - Protection: kernel-userspace bridges (Dokan, FUSE) are fairly stable, file system can crash without requiring a reboot
  - Porting: one common kernel->userspace upcall interface (FUSE) on Linux, OS X, Solaris
- Acceptable performance for network file systems
  - Often bound to disk anyway

Overview

- Implementing file systems in userspace
- Handling concurrency
- XtreemFS: an object-based distributed file system

Implementing file systems in userspace


```
static int
mkdir(
  const char* path,
  mode_t mode
);
```

```
static int
DOKAN_CALLBACK
CreateDirectory(
  LPCWSTR FileName,
  PDOKAN_FILE_INFO
);
```

- ~ VFS functions
- FUSE kernel module translates operations to messages, writes them on an FD
- FUSE userspace library reads the messages, calls the appropriate function, returns the result as a message
  - Callbacks must be thread-safe, completely synchronously.
- Dokan (Win32) calls can be translated, sans sharing modes.

Abstract away

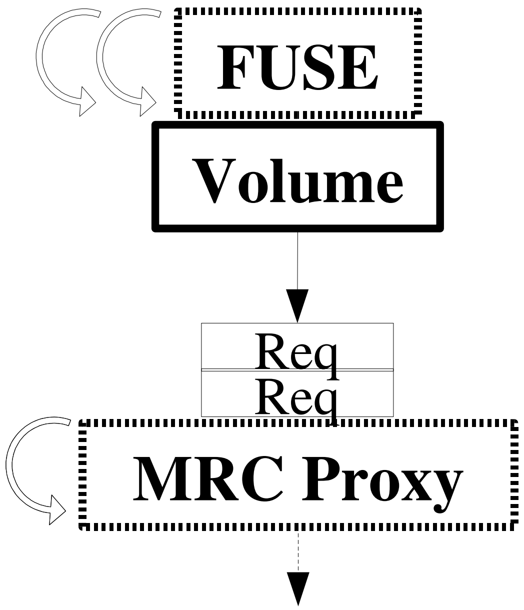
```
bool Volume::mkdir(
  const YIELD::Path& path,
  mode_t mode
)
{
  mrc_proxy.mkdir(
    Path( this->name,
          path ), mode );
  return true;
}
```

- **Yield** C++ library for minimalist platform primitives, concurrency (next section), IPC
- Auto-generate client-server interfaces from IDL; make synchronous proxy calls that do message passing under the hood.

Handling concurrency

- Possible approaches:
  1) Let the (multiple) FUSE threads execute all of the logic of the system
     - Advantages: simple at the outset
     - Disadvantages: have to lock around shared data structures, error prone and code becomes a mess
  2) Have some sort of event loop
     - Advantages: obviates need for locks
     - Disadvantages: code becomes even uglier, even faster; hard to parallelize

Stages



- Decompose file system logic into **stages** that pass messages via queues.
- A stage is a *unit of concurrency*: two stages can always run concurrently on two different physical processors.
  - Single-threaded stages: shared data structures encapsulated by a single serializing stage – no locking
  - Most stages should be thread-safe (otherwise Amdahl's law comes into play).
  - A stage-aware scheduler can exploit the nature of stages as well as their communications pattern (the *stage graph*, similar to a process interaction graph).

XtreemFS

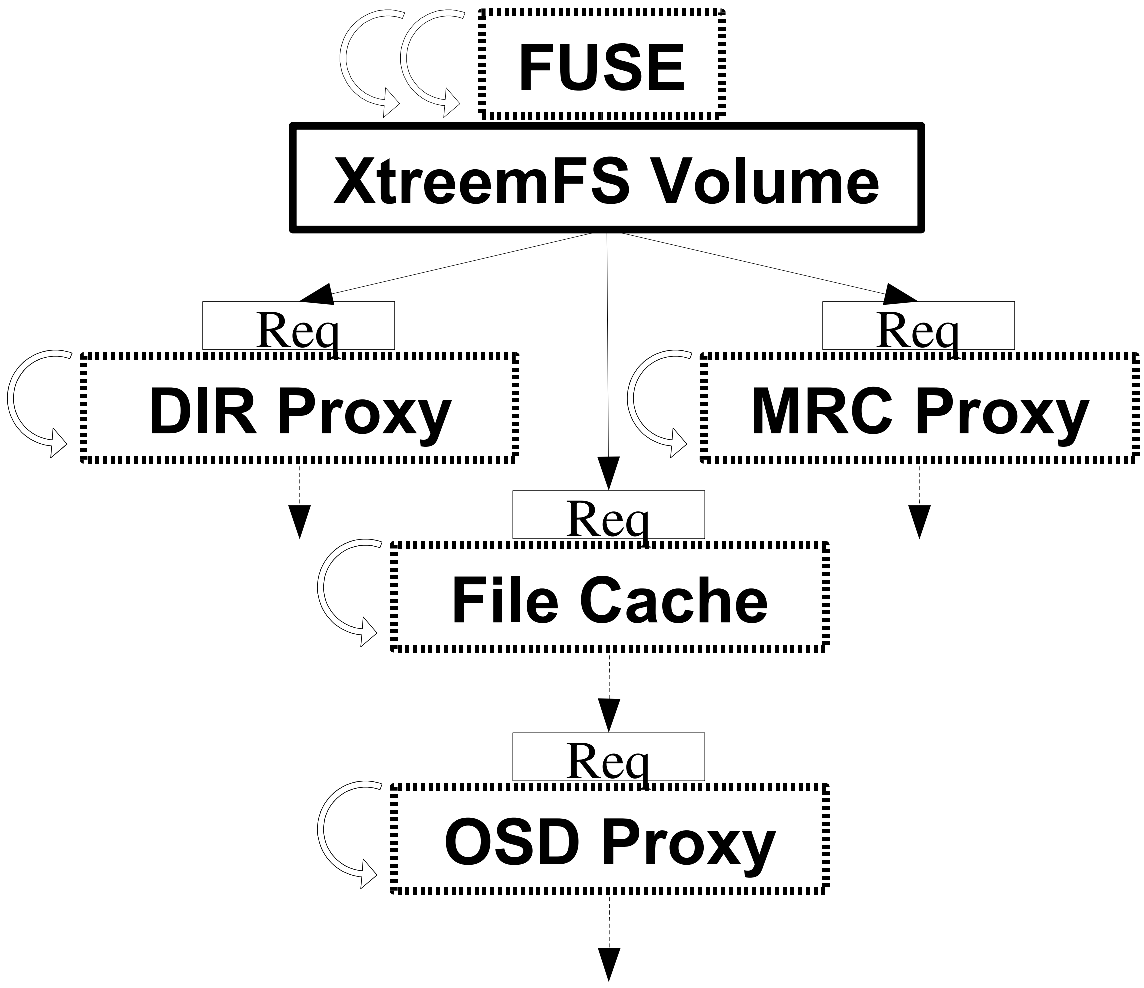
- EU research project
- Wide-area file system with RAID, replication
- Aim for POSIX semantics, allow per-volume relaxation
- Everything in userspace
  - Test new ideas with minimal implementation cost
- Goal: usable file system that performs within an order of magnitude of kernel-based network file systems

XtreemFS: Features

BABUDB

- Staged design
- Efficient key-value store for metadata
  - Based on Log-Structured Merge Trees
  - Simple implementation (~ 5k SLOC)
  - Snapshots
- Striping
- WAN operation
  - Distributed replicas held consistent
  - Automatic failover
  - Security with SSL, X.509

XtreemFS: Stages



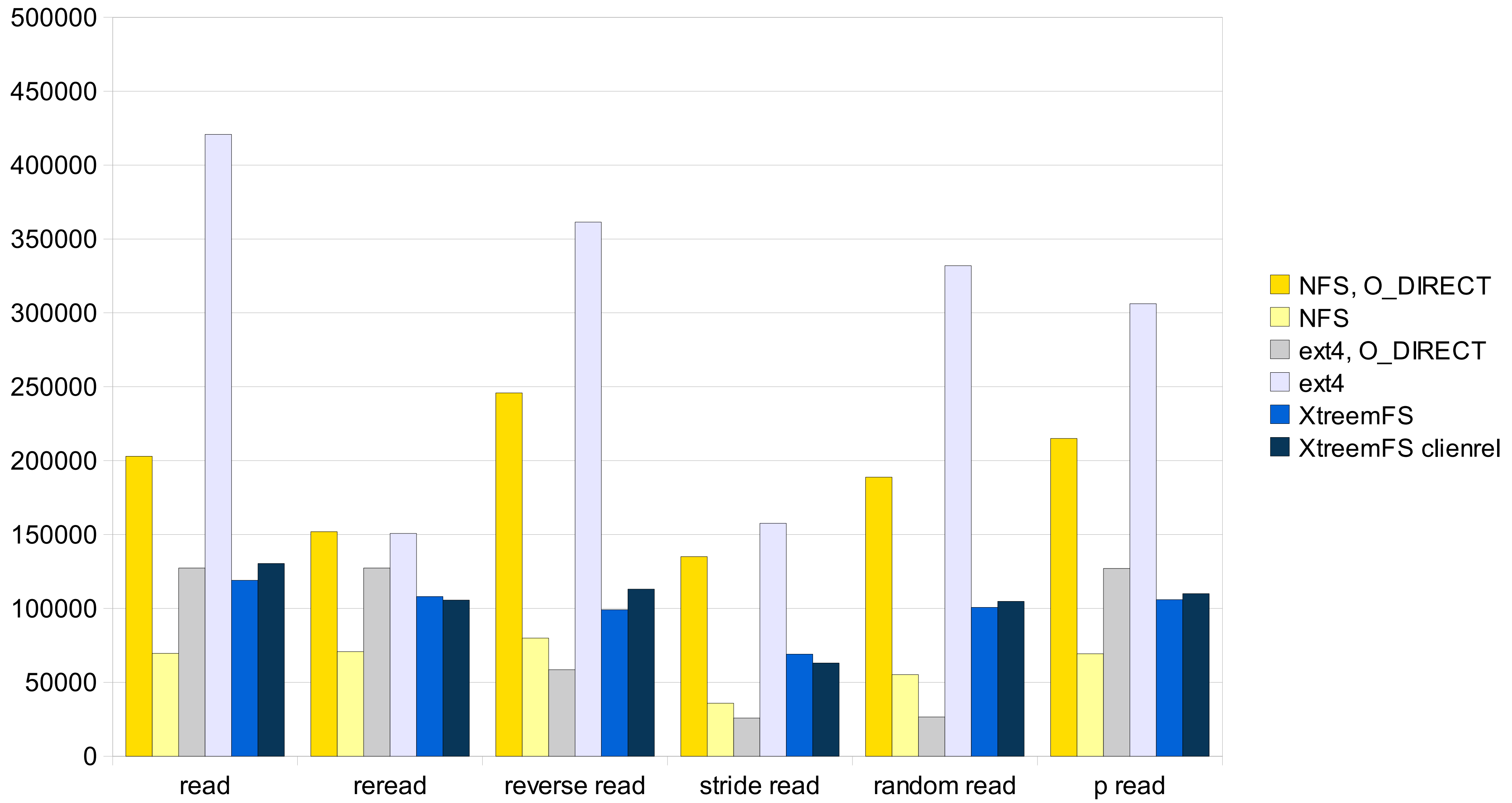
- Client
- Servers
  - Directory (DIR)
  - Metadata catalogue (MRC)
  - Object store (OSD)

XtreemFS: Stages cont'd

Advantages of staged design in XtreemFS:

- No locking around shared data structures like caches
  - Other stages can be multithreaded to increase concurrency or offset blocking
- Gracefully degrade under [over]load with queue backpressure (original raison d'etre of stages in servers)
- Userspace scheduling
  - Per-stage queue disciplines like SRPT
  - Stage selection (CPU scheduling)
    - Increase cache efficiency (Cohort scheduling, my research)

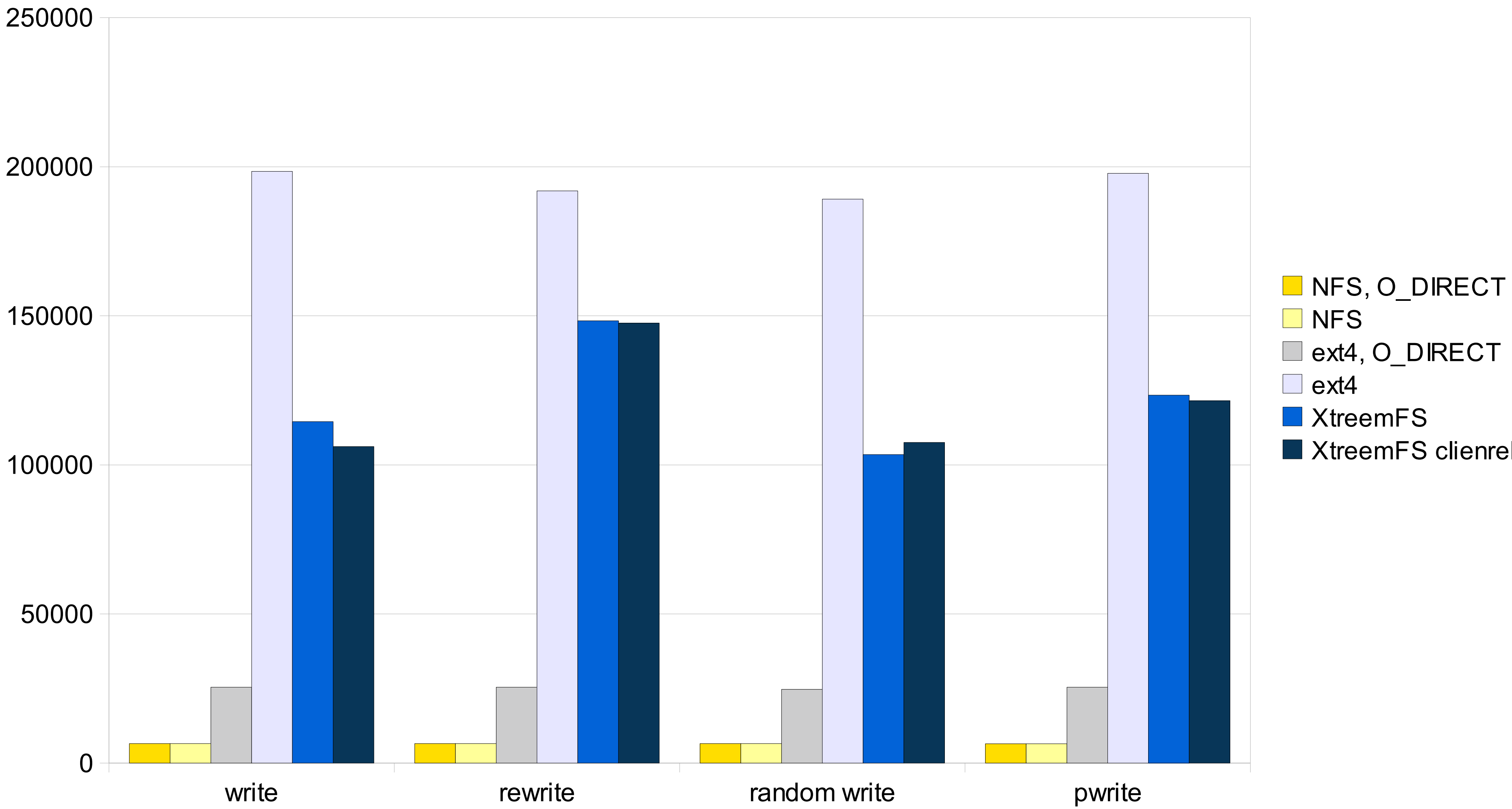
XtreemFS: local reads



XtreemFS: local writes

HIGH PERFORMANCE COMPUTING

05/04/09



Conclusion

- Project runs until June 2010
- Next release: beginning of May
  - Re-implemented client (Linux, Win, OS X)
  - Client-side metadata, data caching
  - New binary protocol (based on ONC-RPC)
  - Full SSL/X.509 support
  - Read-only WAN replication
  - Plugin policy modules for access control

http://www.xtreemfs.org/

Thank you for your attention.

Questions?