OpenGFS

Ben Cahill

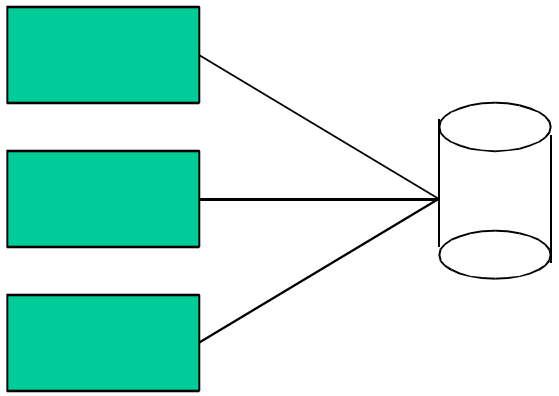
Ben.m.cahill@intel.com

Not a filesystem expert

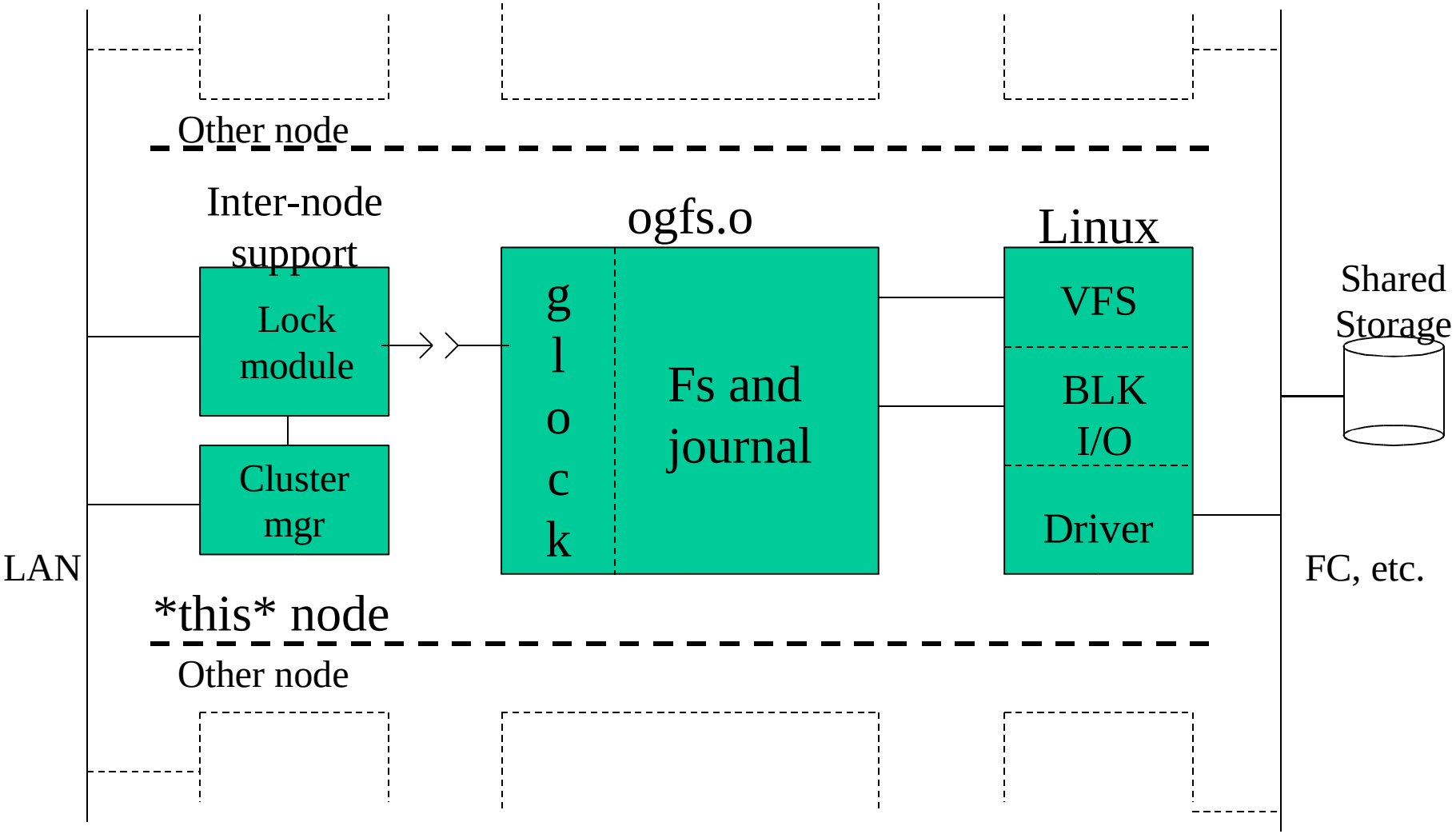
(but having fun learning)!

What Is OpenGFS?

- Clustered Filesystem – What does *that* mean?
  …. Many different things to different people.
- OpenGFS provides direct access to shared storage media. No “middle man”.



OpenGFS Architecture



What's OpenGFS Good For?

- Probably best in read-intensive applications.
- Should also be good for different computers writing into different directories.
- Might be bad for write-intensive apps sharing a single file or directory among several computers.
- All of the above are untested hypotheses, based on evolving understanding of architecture and internals.

What enables shared access??

- #1 is Locking!!
  - Resource groups for block allocation (exclusive).
  - Inodes for reads (shared) / writes (exclusive).
  - Several others for journal recovery, mount, flock, etc.
- Individual journals for each compute node.
  - no interleaving of different nodes' activities.
  - easy contention-free write/read.
  - easy node-specific recovery.
- Consistent representation of filesystem/journal devices across all compute nodes.

What enhances sharing?

- Resource groups (block alloc/unalloc)
  - block usage stats are distributed throughout the filesystem; each resource group (not superblock) stores its own block stats, along with a block usage bitmap.
  - each computer node uses a unique resource group within the filesystem for allocating blocks.
  - 3 methods of changing resource group:
    - Single – use it until it is full
    - Round robin – use new one for each new directory
    - Random – use new one for each new directory
  - Reads and non-size-changing writes do not use/access/lock resource group stats or bitmap.

What enhances sharing (cont)?

- One inode per block
  - Filesystem Block (typ. 4k) is the finest grain locking available (KISS)*.
  - No lock contention among multiple inodes (only one inode per block/lock).
  - Locks use block # as ID, along with lock type.

* Keep it simple, stupid!

What has very little to do with sharing, from FS perspective?

- Volume management
  - OGFS (now) really doesn't care how the filesystem is comprised of various disks (external journals allow OGFS to create journals on specific devices).
  - OGFS self-detects filesystem expansion (not volume expansion) or journal additions made by another node.
  - No need for special hooks (ioctls, etc.) between Volume Manager and OpenGFS.
- Cluster management
  - Normal OGFS filesystem operations don't care how many, or which, nodes are in a cluster.
  - The fs just needs to obtain a lock, it doesn't care where from (it relies on the lock manager for that).

What Was OpenGFS??

- *Everything* you might need to make a Clustered filesystem:
  - Filesystem code
  - Journaling code
  - Intra/Inter-node Locking code (g-lock/memexp)
  - Cluster management code (memexp)
  - Volume management code (pool)
- Much of this did not exist, or was not very mature, when OGFS was written.

History

- Project at U. Minnessota 1995-1999
- Sistina open source 2000-2001
- Sourceforge project 2001-present
  - Alan Cox, hch, et al set up Sourceforge site, grabbing source from Compaq SSIC-Linux 0.5.1, August 2001.
  - Ran out of time (weren't able to "solve the pool mess"). Project languished.
  - Mid 2002, Brian Jackson and Dominik Vogt started work. Solidified build system. Began documentation.
  - 2003 has seen activity pick up a lot.

2003 Activity

- Solve “the pool mess”. Liberate OGFS from pool VM, by adding external journals (Ben Cahill).
- OpenDLM lock module (Stan Wang).
- EVMS File System Interface Module (FSIM) (Luciano Chavez)
- Block alloc enhancement (Dominik Vogt).
- Mmap support (Daniel Phillips).
- Documentation!! (Ben Cahill and others)

2003: Pool

- Myth that OGFS depended strongly on pool.
- Actually, relatively weak dependence: only at filesystem creation time. Mkfs.ogfs needed to know virtual device makeup in order to:
  - Place specific journals on specific partitions. External journals solve this problem.
  - Support striping efficiently. We ignored this, but a command line option to mkfs.ogfs could provide info.
- Now we can use virtually any volume manager (or none at all if all cluster members expose devices in consistent way). Seems solid.

2003: OpenDLM

- Memexp has one central lock storage server. If it crashes, it's very bad. Also, has inefficient load/store protocol. Need a Distributed Lock Manager (DLM), with efficient messaging.
- G-lock supports plug-in locking protocol modules.
- Sourceforge OpenDLM was set up by OpenGFS guys (Brian and Dominik). Avoids single point of failure.
- We now have OpenDLM lock module for OpenGFS, not production-ready, but there!

2003: EVMS

- Enterprise Volume Management System (EVMS), central “control panel” for storage management.
  - Virtual device (volume) composition, expansion.
  - Filesystem creation, expansion, journal addition, etc.
- EVMS now cluster-aware, so propagates volume management info/changes to all nodes.
- File-System Interface Module (FSIM) provides the filesystem management interface, using (mostly) pre-existing OGFS user space utils.
  - OGFS doesn’t care about *volume* expansion
  - OGFS self detects *filesystem* expansion and new journals, no need for notification from volume mgr

2003: Block Alloc & mmap

- Block alloc enhancement:
  - AVL tree for resource group search.
  - Recovery of excess dinode and metadata blocks.
  - Code cleanup, comments.
- Clustered mmap:
  - OpenGFS provided a good test bed for kernel support of clustered mmap.
  - Small patches to OGFS and kernel bridged a file consistency gap (patch releases exclusive inode lock *after* page table is updated).

2003: Documentation

- WHATIS-ogfs: Introductory material
- HOWTO-generic, HOWTO-nopool: Installation with, and without, the (deprecated) “pool” volume manager.
- HOWTO-evms: Setup of EVMS to provide a filesystem device and an external journal device.
- HOWTO-debug: Use of OGFS internal debug facilities.
- Comments in code (pretty barren, more to do)

2003: Documentation (cont)

- ogfs-* internals docs:
  - On-disk layout: What's on disk, and more
  - Pool: volume manager, research for "liberating" OGFS from pool
  - Locking: G-lock system, research for using OpenDLM for OGFS, and using g-lock for ext3
  - Memexp: memexp lock module, research for using OpenDLM for OGFS, and using g-lock for ext3
  - Block alloc: research for enhancing performance and filespace usage, and fixing a panic.
  - Ops: OS interface and big view of OpenGFS operations
  - Still need: Journaling (research for performance)

Future activity

- Performance suspects:
  - Journaling
  - Locking
  - Internals
- Clusterize ext3?
  - Separate glock from ogfs module
- Port to 2.5/2.6
- More documentation (inc. comments)!

Future: Performance

- OpenGFS performance is a bit lacking at present, and we need to learn why.
- Recent (quick) experiment shows that journaling is a performance bottleneck:
  - rm takes as long, or longer, than untar of Linux tree
  - OGFS journaling serializes flushes to disk
  - Tweak OGFS journaling?
  - Try Journaling Block Device (JBD)?
- Other suspects:
  - Locking performance
  - Overuse (?) of BKL

Future: “Clusterizing” ext3

- Well known/exercised/maintained code.
- Add locking (via g-lock interface?).
- Tweak ext3 journaling (JBD) and flushing to provide good performance along with inter-node file consistency (Stephen Tweedie has some ideas).
- Individual journal for each node.
- Potential trouble spots:
  - All block alloc stats stored in superblock (bottleneck)
  - Multiple inodes per block (big problem, or not?)

More Info

- Opengfs.sourceforge.net
  - Start with WHATIS-ogfs
    - Overview
    - CVS tree tour
    - References (inc. other interesting filesystems)