Whamcloud

Lustre Security

08/2020

sbuisson@whamcloud.com

Lustre security

- Security fixes
  - Stabilization & improvements
  - Security related testing
- Client-side encryption
  - Development areas
  - Compatibility with future releases
  - Roadmap

Security related stabilization & improvements

Whamcloud

- LU-12266 mdd: fix up non-dir creation in SGID dirs
- LU-12604 mdt: check field size of sec context name
- LU-12895 mdt: check if object exists first
- LU-12944 mdd: pass correct xattr size to lower layers
- LU-12469 mdd: handle migrate case with SELinux
- LU-13077 pfl: cleanup xattr checking
- LU-13152 llapi: llapi_layout_get_by_xattr groks DoM
- LU-13142 lod: cleanup layout checking
- LU-13116 mgc: do not lose sptlrpc config lock
- LU-12401 gss: fix checksum for Kerberos and SSK
- LU-12894 sec: fix checksum for skpi
- LU-13216 ptlrpc: sptlrpc_req_refresh_ctx's timeout semantic
- LU-13466 mgc: protect from NULL exp in mgc_enqueue()
- LU-12992 gss: retry in case of short computed shared key
- LU-12854 nodemap: allow boolean value for audit_mode
- LU-13754 gss: open sptlrpc init channel in R+W mode
- LU-13474 gss: do not return -ERESTART when gss rpc times out *(not landed yet)*
- LU-13355 crypto: adler32 wrapper in libcfs
- LU-13064 sec: check permissions for changelogs access

Security related testing

Whamcloud

- To avoid regressions: new test groups
  - review-dne-selinux
    - sanity, recovery-small, sanity-selinux **with SELinux enabled**
  - review-dne-ssk
    - sanity, recovery-small, sanity-sec with **SSK enabled**
  - review-dne-selinux-ssk
    - sanity, recovery-small, sanity-selinux, sanity-sec **with both SELinux and SSK**

- review-dne-selinux enforced for 2.14
  - once other test sessions are passing they will also be enforced
    - any help would be welcome

Security related testing

Whamcloud

- Test fixes, landed
  - LU-12039 tests: fix 'lfs mkdir' in sanity-selinux
  - LU-12131 tests: only create lgssc.conf file if necessary
  - LU-12131 tests: fix SSK handling in tests
  - LU-12267 tests: update filter in acl for SElinux case
  - LU-12131 tests: properly handle GSS in server failover
  - LU-12131 tests: fix test_802a for GSS
  - LU-12472 tests: update sanity-krb5.sh
  - LU-12428 tests: wait for nodemaps to be synchronized
  - LU-12428 tests: fix sanity-sec wait_nm_sync
  - LU-12778 tests: give time to apply nodemap
  - LU-13042 tests: give more time in sanity-selinux test_21b
  - LU-13097 tests: set fail_loc on all MDS nodes for pdir tests
  - LU-13133 tests: sanity-selinux test_21{a,b} sepol update
  - LU-13082 tests: enable lgss_keyring debug traces
  - LU-13116 tests: properly clean keyring in sanity-sec test_30
  - LU-13156 tests: wait for nodemap update in sanity-selinux
  - LU-12730 tests: sync file before checking LSOM
  - LU-13580 tests: fix retrieval of SELinux context

Lustre Client Encryption – solution proposal

Whamcloud

- Conform to fscrypt kernel API
  - Current users are ext4, F2FS, and UBIFS
  - Need for encryption policies v2 from 5.4 kernel
- Retain ext4 encryption principles in place in Ubuntu/RHEL8
  - Encrypted page size = clear text page size
  - Encryption chunks are independent from each other
  - Pages in the page cache always contain clear text data
- Make use of fscrypt userspace tool
  - Manage encryption policies
    - Tell which directories to encrypt, and how

Lustre Client-side Encryption – workflow

Whamcloud

- Applications see clear text
- Information is encrypted before being sent to servers
  - Then remains untouched
- Information is decrypted upon receipt from servers
  - Untouched before that
- Servers only see encrypted information
  - But (mostly) do not need to be aware of it
- Only client nodes have access to encryption keys

Lustre Client-side Encryption – development areas

Whamcloud

A. Ability to encrypt file content
   - Encrypt on write, decrypt on read
   - Bandwidth performance impact

B. Ability to set encryption policies on directories
   - Support new IOCTLs from fscrypt userspace tool
   - Set encryption context directly with create requests
   - Fetch encryption context directly with open/lookup requests

C. Ability to encrypt file, symlink and directory names
   - Call fscrypt primitives from Lustre code
   - Metadata performance impact

Area A - encrypt file content

Whamcloud

- Main patches
  - LU-12275 sec: reserve flags for client side encryption
  - LU-12275 osd: make osd layer always send complete pages
  - LU-12275 sec: documentation for client-side encryption
  - LU-12275 sec: enable client side encryption
  - LU-12275 sec: control client side encryption
  - LU-12275 sec: encryption for write path
  - LU-12275 sec: decryption for read path
  - LU-12275 sec: deal with encrypted object size
  - LU-12275 sec: support truncate for encrypted files
  - LU-12275 tests: exercise file content encryption/decryption
  - LU-12275 sec: encryption support for DoM files

Lustre Client-side Encryption – truncate case

Whamcloud

- Truncate size aligned with PAGE_SIZE: fine
- Otherwise: truncating would corrupt encrypted page
  - because encryption block size is PAGE_SIZE
  - write 0s inside an encrypted page and it becomes un-decryptable

- Need to:
  - read whole page in which truncation happens
  - decrypt content
  - do truncate
  - re-encrypt and rewrite

Lustre Client-side Encryption – truncate case

Whamcloud

- Not easy to implement in Lustre
  - truncate = change size attribute
  - in *ll_setattr* (no file handle available)
    - take cl_lock on range corresponding to concerned page
    - grab vm page
    - associate cl_page
    - proceed to clio read
    - zero range in page
    - proceed to cl_page flush
    - release cl_lock
    - finally change size attribute

→ see *ll_io_zero_page* (LU-12275 sec: support truncate for encrypted files)

Area A - encrypt file content

Whamcloud

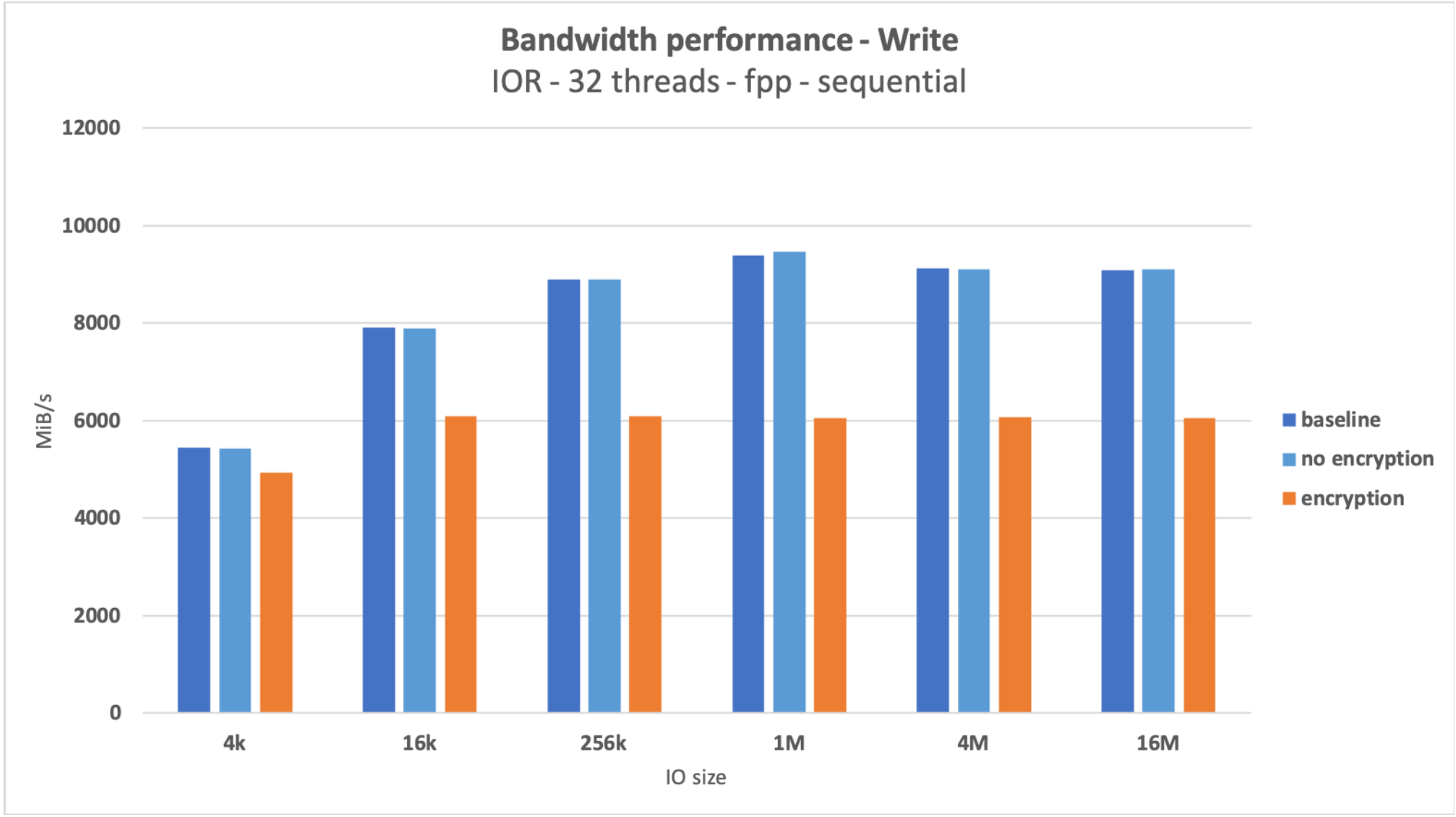
- Complementary patches
  - LU-12275 sec: O_DIRECT for encrypted file
  - LU-12275 sec: restrict fallocate on encrypted files
  - LU-12275 sec: ldiskfs not aware of client-side encryption
  - LU-12275 sec: encryption with different client PAGE_SIZE

Lustre Client Encryption – bandwidth performance

Whamcloud

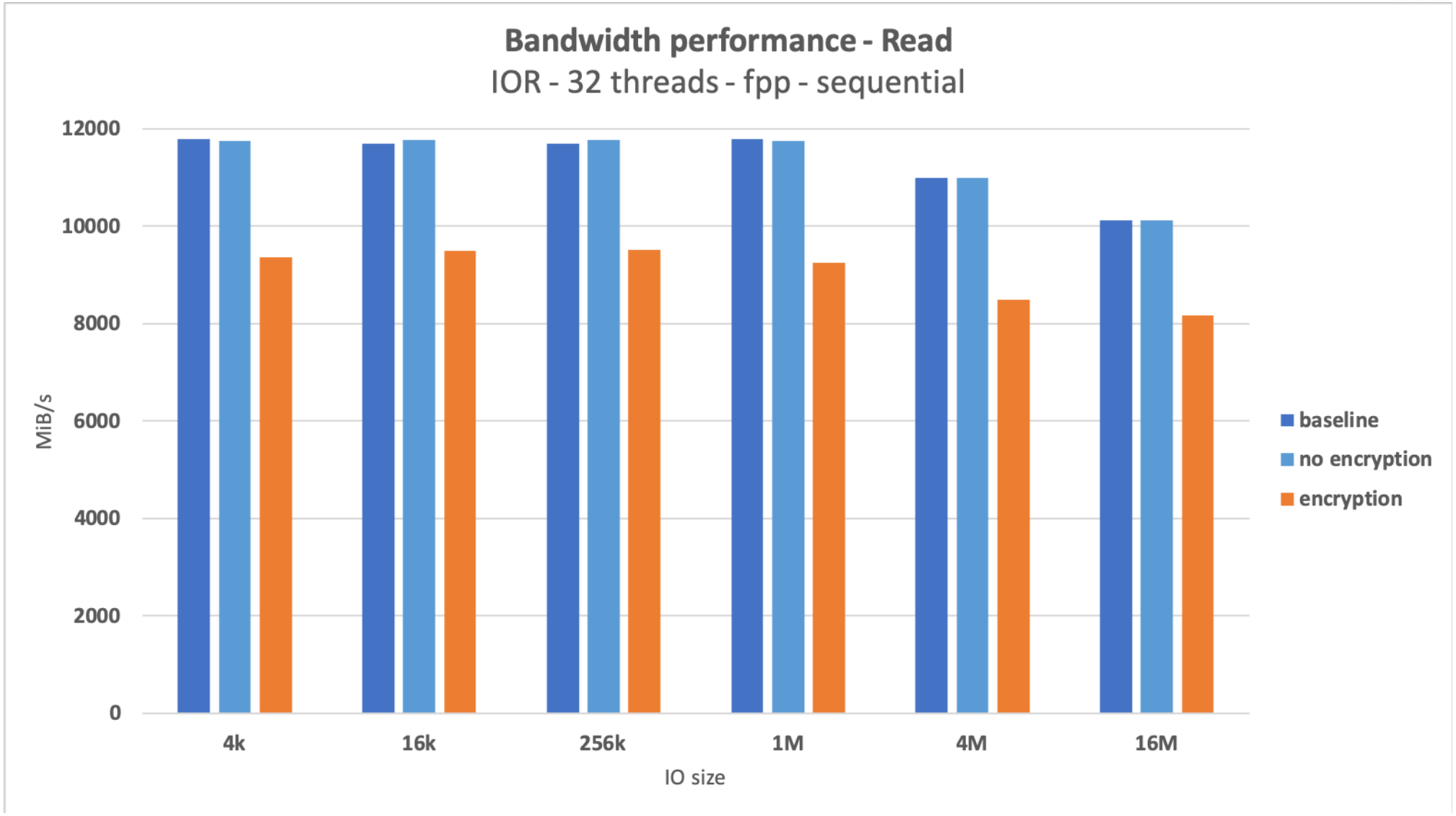
- POC code on top of `master`, **dummy encryption mode** (AES-256-XTS)
- Testbed
  - Client
    - Skylake 48 cores, Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz
    - 96 GB RAM
    - ConnectX-4 Infiniband adapter, EDR network
  - Storage
    - DDN ES200NV, 20 x NVMe HGST 1,7TB, 1 DCR pool
    - 4 OSTs, each 1/10th of pool
- Methodology
  - IOR, file per process, sequential IO

Area A – bandwidth performance

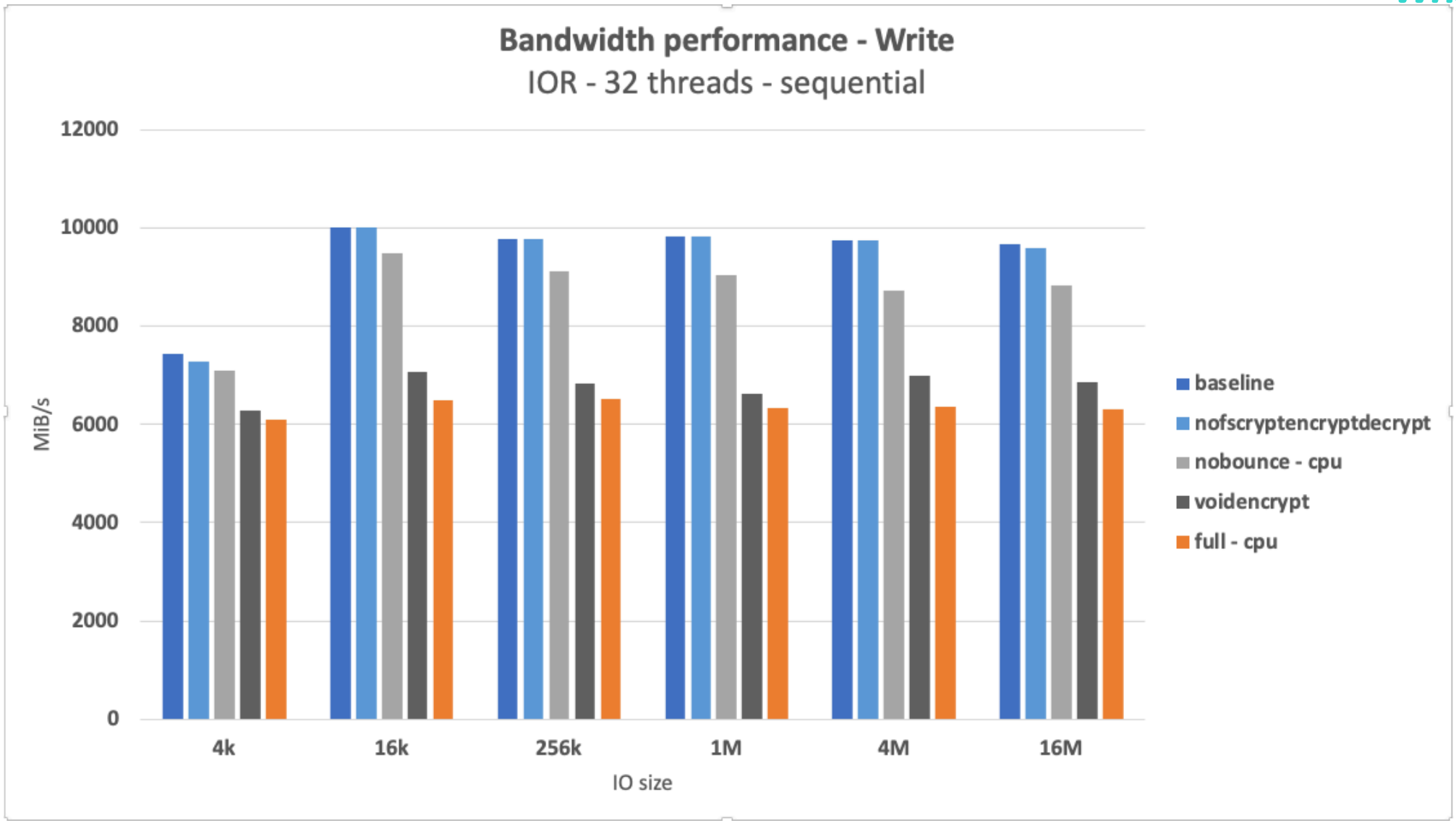


Area A – bandwidth performance

Whamcloud



Area A – performance investigations



Area A – performance investigations

Whamcloud

- **Compare `nobounce` and `voidencrypt`**
  - `nobounce`: encryption but no bounce page allocation: 10% drop
  - `voidencrypt`: no encryption but bounce page allocation: 30% drop

⇒ **bounce page allocation hurts**

- **Possible optimization path**
  - leverage Lustre's enc_pool mechanism
    - take bounce pages from this pool
    - do not allocate bounce page for every call to encryption primitive

Area B – new ioctls for policies

Whamcloud

- fscrypt API v2
  - Revokes pages from page cache when encryption key is removed
  - Landed in Linux 5.4
    - but we need to support CentOS 8 (4.18) / Ubuntu 18 (4.15) clients

- Solution retained
  - Include fscrypt API from Linux v5.4 inside Lustre tree

  → LU-12275 sec: add llcrypt as file encryption library
    - renamed to llcrypt to avoid name conflicts
    - available in libcfs module
  - Wire up new ioctls in llite

  → LU-12275 sec: ioctls to handle encryption policies

Area B – new ioctls for policies

Whamcloud

- fscrypt userspace tool
  - Works with Lustre out of the box, thanks to fscrypt API support
  - Supports encryption policies v2
  - Associates protectors (passphrase, raw key, pam) to policies

```
# fscrypt setup /mnt/lustre
$ fscrypt encrypt /mnt/lustre/vault
$ fscrypt lock /mnt/lustre/vault
$ fscrypt unlock /mnt/lustre/vault
$ fscrypt metadata change-passphrase
    --protector=/mnt/lustre:7626382168311a9d
$ fscrypt metadata add-protector-to-policy
    --protector=/mnt/lustre:2c75f519b9c9959d
    --policy=/mnt/lustre:16382f282d7b29ee
```

Area B – encryption context handling

Whamcloud

- Encryption context handling
  - Per-file encryption context is stored in an xattr
    - Hopefully not changed after file creation
  - llcrypt sets encryption context on files and dirs at create
  - llcrypt fetches encryption context upon metadata ops
    - lookup
    - getattr
    - open
    - truncate
    - layout

Area B – encryption context handling

Whamcloud

- Encryption context atomicity

  → LU-12275 sec: atomicity of encryption context getting/setting

  - ‘setting’ case
    - send encryption context to the MDT along with create RPCs
      - closing insecure window between creation and setting of encryption context
      - saving a setxattr request for better performance
    - explicitly put xattr in cache (new *ll_xattr_cache_insert()* primitive)
      - saving a subsequent getxattr request
  - ‘getting’ case
    - server returns encryption context upon granted lock reply
      - making the encryption context retrieval atomic
      - saving a getxattr request for better performance
    - client inserts received context in xattr cache
      - saving a subsequent getxattr request

Area C – metadata encryption

Whamcloud

- POC just started: LU-13717
- Wire up llcrypt API in llite
  - *llcrypt_setup_filename*
  - *llcrypt_prepare_lookup*
  - *llcrypt_prepare_symlink*
- Convert between plain text and cipher text names
  - from plain to cipher before sending request to MDT
  - from cipher to plain upon reply
  - 2 cases to support
    - access with the key: present actual names
    - access without the key: base64 encoding of cipher text names

Area C – metadata encryption challenge

Whamcloud

- ‘name’ is no longer a valid path name, not even a well-formed string
  - binary ciphertext names just cannot be encoded (base64 or similar)
    - fscrypt API supports plain text file names of up to NAME_MAX length
    - NAME_MAX limit would be exceeded when encoding
- Impact of binary ‘names’
  - lu_name_is_valid* check meaningless
  - protocol: `RMF_NAME` no longer an `RMF_F_STRING`
  - OSD layer: convert `char*` to `struct lu_name`
  - `lfsck`…
  - hopefully, ldiskfs and ZFS backend file systems are capable of handling binary names

Area C – metadata encryption preliminary testing

Whamcloud

- Works fine with DNE
  - thanks to FID mapping, no particular aspect to take care of
- symlink a little bit tricky
  - needs inode to encrypt target name to send to MDT…
  - … but inode obtained only after create request sent to MDT
- fid2path broken
  - because path is built on server side
  - could proceed to open-by-fid if server says file is encrypted?
- Metadata performance benchmarks to be done

Client-side encryption – releases compatibility

Whamcloud

- Compatibility with future versions
  - Lustre 2.14 will have content encryption only
  - future versions will add name encryption
  - but fscrypt policies designed to handle both content and name encryption
- Problem when upgrading from 2.14
  - new code will try to decrypt clear text names created with 2.14
- Solution proposal
  - add Lustre specific LLCRYPT_MODE_NULL

→ LU-12275 sec: introduce null algo for filename encryption

→ LU-12275 sec: force file name encryption policy to null

Lustre Security – Client-side encryption

Whamcloud

- Projected roadmap
  - content encryption
  - fscrypt inclusion
  - encryption policies support

  } land in 2.14

  - metadata encryption
  - performance optimizations

  } target 2.15

Whamcloud

Thank you!

sbuisson@whamcloud.com