Lustre Client Encryption

Lustre User Group 2023

sbuisson@whamcloud.com

Lustre Client Encryption

- Lustre Client Encryption features wrapped-up in 2.15

- Limitations with client-side encryption
  - `fid2path`
  - access to raw encrypted information

- How to address these limitations

Recently added capabilities

Whamcloud

- ▶ Compat with in-kernel fscrypt API
  - Align “no key” filename presentation with RFC 4648 base64url – LU-16374
  - `mgs# lctl set_param -P llite.*.`**`filename_enc_use_old_base64`**`={`**`0`**`,1}`
- ▶ Encrypted objects consistency
  - S_ENCRYPTED flag on OST objects for enc files – LU-16091
  - Better support for e2fsprogs, lfsck

⇒ Available in future 2.15.3 maintenance release

- ▶ Lustre/HSM on enc files with enc keys
  - Internally similar to file migration – LU-16310

⇒ Now merged in master branch

Limitations with encryption – `fid2path`

- `lfs fid2path` maps a numeric Lustre File IDentifier (FID) to one or more pathnames

- fid-to-path resolution carried out on **server** side
  - Full path built on server side, then returned to client
- Name encryption/decryption carried out on **client** side
  - Server side almost not aware of encryption
- Name encrypted with **parent’s** key
  - All entries in a directory encrypted with same key

Limitations with encryption – `fid2path`

Solution: `LU-16205 sec: fid2path for encrypted files`

- Server returns raw encrypted names, encoded

```
vault/sqS08A2BqseOU4aZ/Ms5q5BN29tREEpO1
```

- Client parses string, isolates components
- From top to bottom, recursively with parent inode
  - Client decrypts name
  - If directory, client lookups name, gets inode
    - Lookups not required **without** the encryption key, or when names are **not** encrypted

- Now merged in master branch

Limitations with encryption – access to raw enc info

- Use cases for access to raw encrypted information
  - Backup/restore
    - at backend file system level (ldiskfs)
    - at Lustre client level
  - Lustre/HSM
  - Moving encrypted files between file systems
- Without the encryption key
  - to avoid making clear text copies
  - to be done by admins, without asking users for their keys
  - to avoid storing users’ keys
  - to be faster than decrypting/re-encrypting

Limitations with encryption – access to raw enc info

- *fscrypt* forbids access to raw encrypted info

  Open encrypted files without the encryption key

  Read and write without the encryption key

- But there are no associated security risks
  - Encrypted info is useless without the key
    - This is why we encrypt
  - Encryption context does not contain per-file key
    - Just a 16-byte nonce
  - But the risk is to corrupt files
    - Write one byte, and decryption reads garbage

Limitations with encryption – access to raw enc info

- Encryption context is not exposed
  - Needs to be saved and restored
- Raw encrypted name is not exposed
  - And cannot be “rebuilt” from presented name without enc key
    - Long names are digested, contain only portion of raw enc name
- Without key, file size rounded up to next encryption block boundary
  - Required to be able to read whole raw content
  - But need to keep track of clear text file size
    - Cannot be inferred from raw content
    - Restore must set back correct file size

Limitations with encryption – access to raw enc info

Solution proposal sent to `linux-fscrypt` mailing-list

https://lore.kernel.org/linux-fscrypt/03a87391-1b19-de2d-5c18-581c1d0c47ca@gmail.com/T/#rcde55362dd39c2a5d130d6eb3495b3dde106c384

- Virtual xattr `security.encdata`, exposing:
  - clear text file data length
  - encryption context
  - raw encrypted name

```
{ encoding: base64url, size: 3012,
  enc_ctx: YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXphYmNkZWZnaGlqa2xtbg,
  enc_name: ZmlsZXdpdGh2ZXJ5bG9uZ25hbWVmaWxld2l0aHZlcnlsb25nbmF...
}
```

Limitations with encryption – access to raw enc info

Solution proposal sent to `linux-fscrypt` mailing-list

- For backup
  - modify tar utility
    - same would apply to other tools
- Explicitly fecth `security.encdata` xattr
- Store it along with backed-up file
  - Content not interpreted by tools
- Open file with special flag O_FILE_ENC + O_DIRECT and read content
- Name of backed-up file: no-key name returned by fscrypt

Limitations with encryption – access to raw enc info

Solution proposal sent to `linux-fscrypt` mailing-list

- For restore
  - modify tar utility
    - same would apply to other tools
- Open file with special flag O_FILE_ENC + O_DIRECT and write content
- Restore `security.encdata` xattr if present
  - Content not interpreted by tools
  - Ldiskfs does not add this xattr to the file, but triggers internal processing
- O_TMPFILE flag also used to create unlinked file
  - Then atomically link with encrypted name

Limitations with encryption – access to raw enc info

Feedback from `linux-fscrypt` mailing-list

- ▶ Their main focus is Android and ChromeOS devices
  - Backups are done in clear text!
  - Or re-encrypted with a key derived from user's password
- ▶ They would want to support all cases at once
  - All encryption modes currently supported by fscrypt
  - All types of special files
- ▶ But we want to go by baby steps
  - First, simple encryption mode and regular files and directories
  - Then enrich capabilities

Limitations with encryption – access to raw enc info

POC – Work In Progress – LU-16374

- ▶ According to public HLD as linked from LU-16259

- ▶ 3 patches so far
  - LU-16374 ldiskfs: round-up enc file size
  - LU-16374 ldiskfs: implement security.encdata xattr
  - LU-16374 ldiskfs: implement backup/restore of enc files

- ▶ Changes to ldiskfs + new lctl command

```
lctl fscrypt read <path to Lustre file> -d <external dir>
lctl fscrypt write <path to backed up file> -d <dir>
```

Limitations with encryption – access to raw enc info

POC – Work In Progress – LU-16374

Limitations with encryption – access to raw enc info

- Next steps
  - Special files
    - symlinks
      - named pipes, device nodes, and sockets: not encrypted
  - Support more kernels
  - Modify tar
  - Client-level backup/restore
    - Leverage what exists for ldiskfs
  - Lustre/HSM without the encryption key

Lustre Client Encryption – wrap-up

Whamcloud

▶ Lustre 2.15 LTS has full encryption support

- encryption of file content
- encryption of file name
- good performance level

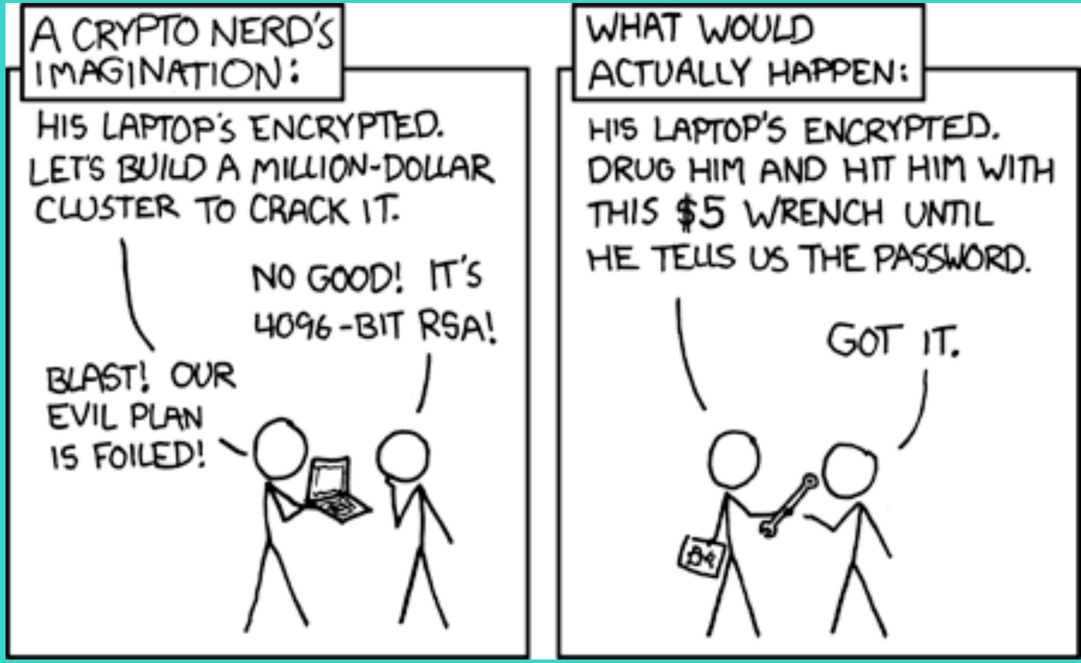
	Performance penalty
Bandwidth – write	5%-10% for large IOs, 15% for small IOs
Bandwidth – read	less than 10%
Metadata – create, stat, remove	5%

▶ Limitations being addressed

- fid2path
- access to raw encrypted information
  - discussions with Linux & Lustre developers in the Community

Thank you!

sbuisson@whamcloud.com


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Lustre Client Encryption – performance

Initial benchmarks

- 30-35% drop in sequential write, 20-25% drop in sequential read

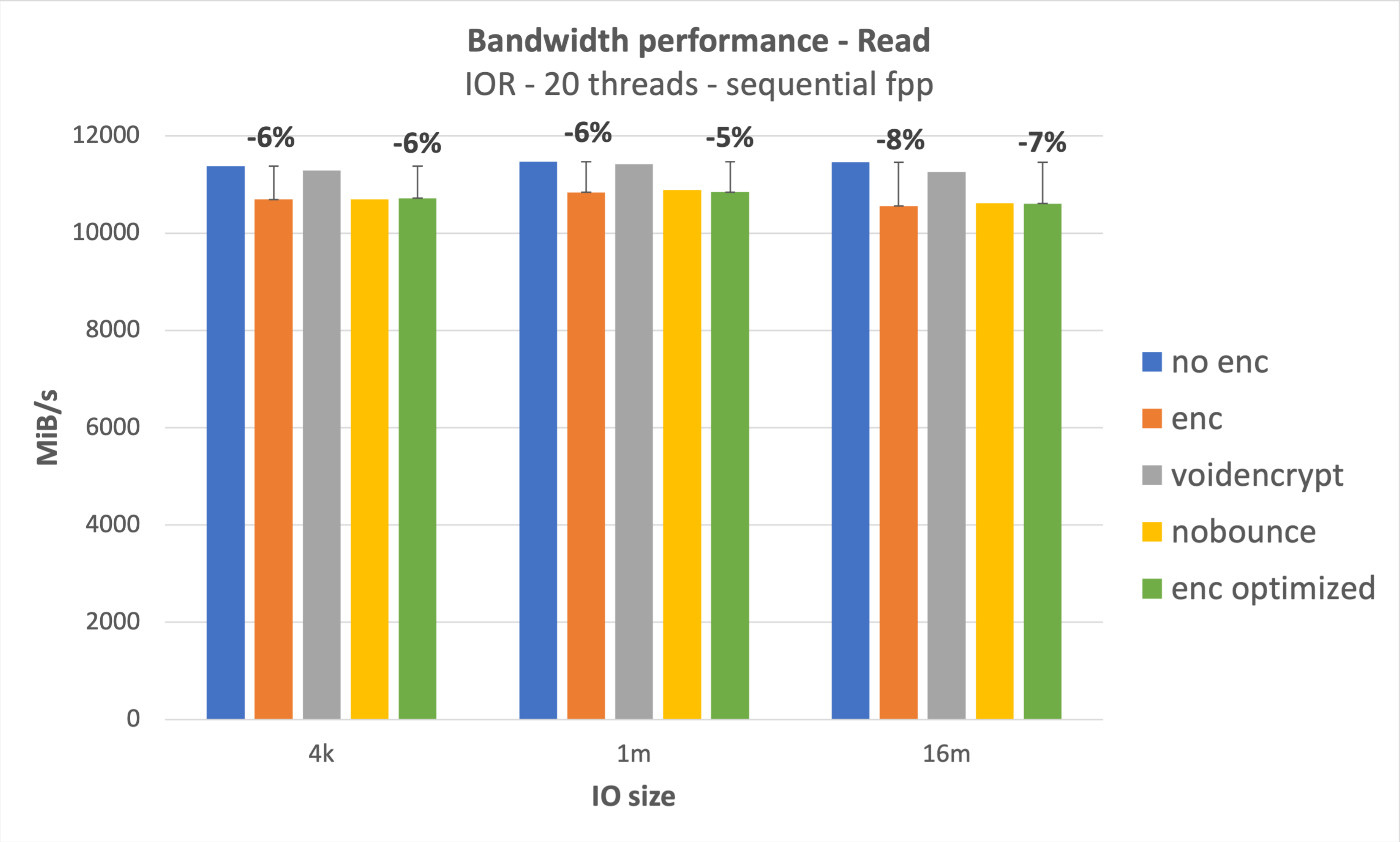
Testbed

- Client
  - Cascade Lake 20 cores, 6230 CPU @ 2.10GHz
  - 192 GB RAM
  - Infiniband adapter, EDR network
  - Ubuntu 20.04 kernel 5.4.0-107-generic
  - Lustre 2.15.0-RC3
- Storage
  - ES400NVX
  - 20 x NVMe, 2 DCR 10 disks
  - 8 OSTs, 4 MDTs
  - CentOS 7.9 kernel 3.10.0-1160
  - Lustre 2.15.0-RC3

Methodology

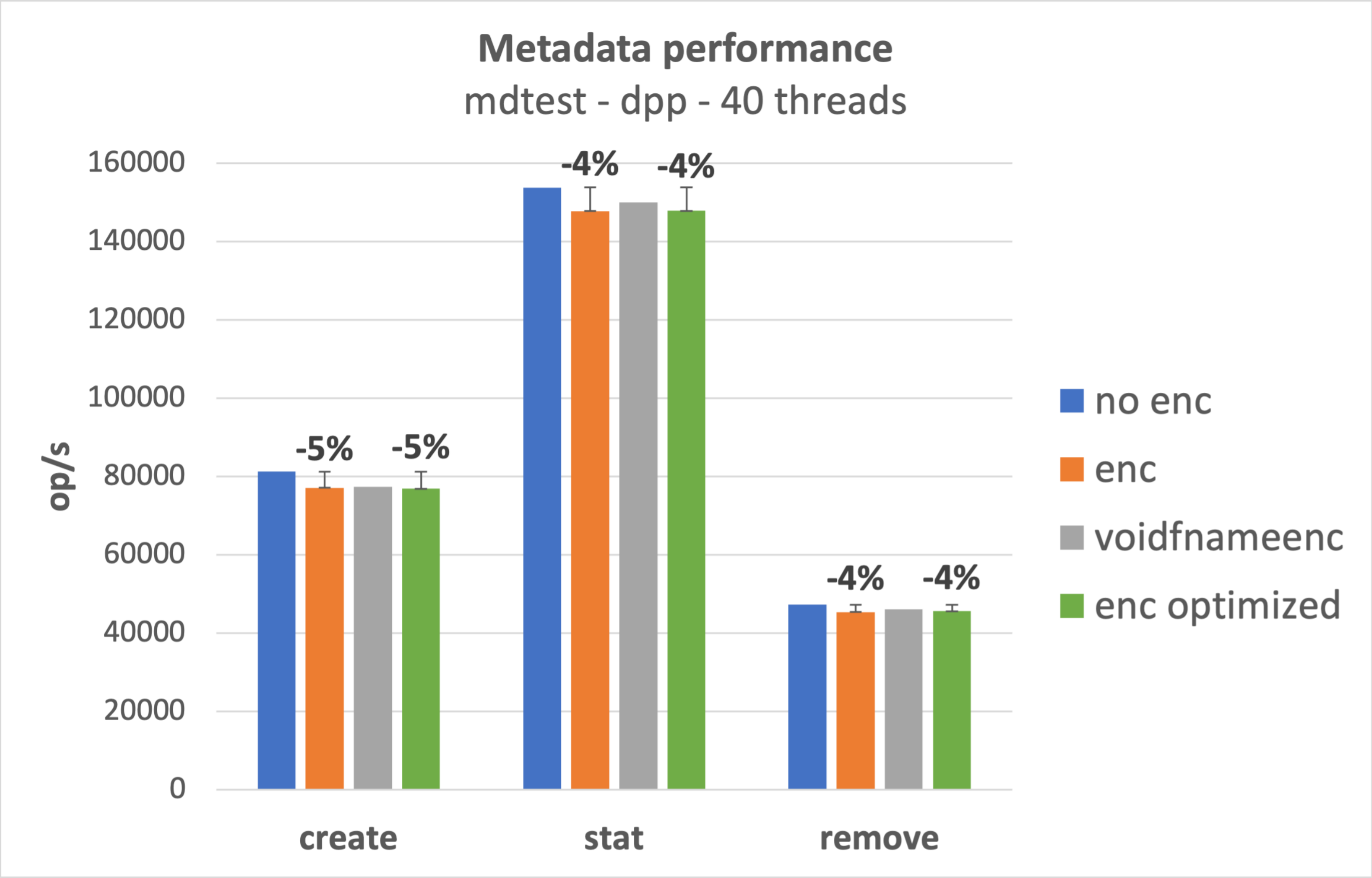
- fscrypt with AES-256-XTS for file content, AES-256-CTS for file names

Lustre Client Encryption – performance



Performance drop for all encryption versions: < 10%

Lustre Client Encryption – performance



Performance drop for all encryption versions: 5%