LAD22
LUSTRE ADMIN & DEV WORKSHOP
Paris 26-28.09.2022

Whamcloud

Lustre Client Encryption

LAD 2022

sbuisson@whamcloud.com

ddn

Lustre Client Encryption

- Before we start…
  - This presentation includes a quiz!
    - online
    - live
    - competitive
    - win a free drink at tonight's dinner party! (or simply self-satisfaction and pride)

Kahoot!

  - With your laptop or smartphone => go to kahoot.it

Lustre Client Encryption

- What is encryption for Lustre and features wrapped-up in Lustre 2.15

- Current limitations with client-side encryption
  - `fid2path`
  - access to raw encrypted information
- How to address these limitations

What is Lustre Client Encryption?

- Kernel side
  - in-kernel fscrypt (5.4)
  - embedded *llcrypt* (CentOS/RHEL 8.1+, Ubuntu 18.04+, SLES 15 SP2+)

- User-space side
  - fscrypt userspace tool: works out of the box, thanks to fscrypt API support

- With Lustre 2.15: full encryption support
  - Content encryption
  - Name encryption

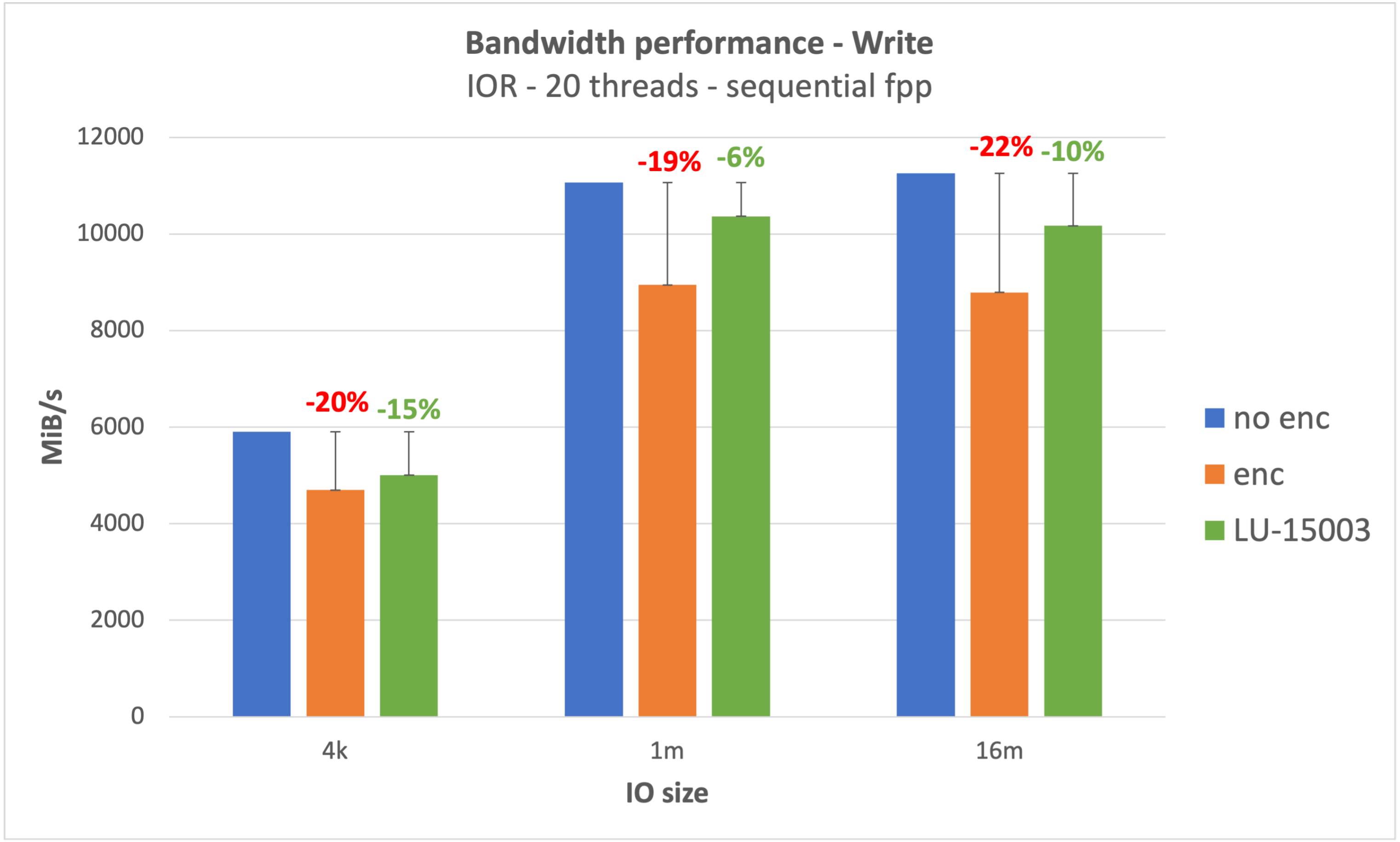
Recently added capabilities

- Access MDT target as ldiskfs
  - Escape encrypted names to avoid breaking the shell – LU-15848
- Reinstate null encryption for file name
  - name encryption disabled by default – LU-15858
  - `mgs# lctl set_param -P llite.*.`**`enable_filename_encryption`**`={`**`0`**`,1}`
- Compat between 2.15 client and 2.14 server
  - Client forced to null encryption for file name – LU-15922

⇒ Available in 2.15.2 maintenance release

Recently added capabilities - continued

- Encryption performance penalty with LU-15003: 5-10% (>= 1MB IO)



⇒Available in 2.15.2 maintenance release

Current limitations with client-side encryption

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

- Take the quiz!

Current limitations with encryption – `fid2path`

- Name encryption/decryption carried out on **client** side
  - Server side almost not aware of encryption
- fid-to-path resolution carried out on **server** side
  - Full path built on server side, then returned to client
- Name encrypted with **parent's** key
  - All entries in a directory encrypted with same key
- **If you try `lfs fid2path` on encrypted file with current code, with or without the encryption key**
  - would not make sense to present raw encrypted names

$\Rightarrow$ -ENODATA

Current limitations with encryption – `fid2path`

Solution proposal: with the encryption key:

- Let server return raw encrypted names, encoded

  `vault/sqS08A2BqseOU4aZ/Ms5q5BN29tREEpO1`
- client to parse string, isolate components
- from top to bottom, recursively with parent inode
  - client to decrypt name
  - if directory, client to lookup name, get inode

- Turns a single RPC action into a multiple lookups operation

Current limitations with client-side encryption

- Access to raw encrypted information
  - Open encrypted files without the encryption key
  - Read and write without the encryption key
  - Get raw encrypted name
  - Fetch encryption context

$\Rightarrow$ Forbidden by *fscrypt*

- Take the quiz!

Current limitations with encryption – access to raw enc info

- Use cases for access to raw encrypted information
  - Move encrypted files between file systems **without decrypt/re-encrypt**
  - Backup/restore without encryption key, **to avoid making a clear text copy**
  - Lustre/HSM without encryption key, **to avoid making a clear text copy**
- *fscrypt* forbids it, but there are no associated security risks
  - Raw 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
- Take the quiz!

Current limitations with encryption – access to raw enc info

- 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
- Encryption context is not exposed
  - needs to be saved and restored

Current limitations with encryption – access to raw enc info

Solution proposal:

- Virtual xattr `security.encdata`, exposing:
  - clear text file size
  - encryption context
  - raw encrypted name
- For backup/restore
  - Modify tar utility
- For Lustre/HSM
  - Modify POSIX copytool

⇒ Fetch this xattr for encrypted files

⇒Use O_FILE_ENC | O_DIRECT flags to read raw data

Lustre Client Encryption – wrap-up

- Lustre 2.15.2 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%

- Identified limitations
  - fid2path
  - access to raw encrypted information
  - design discussion with the other Lustre developers in the Community

Whamcloud

Thank you!

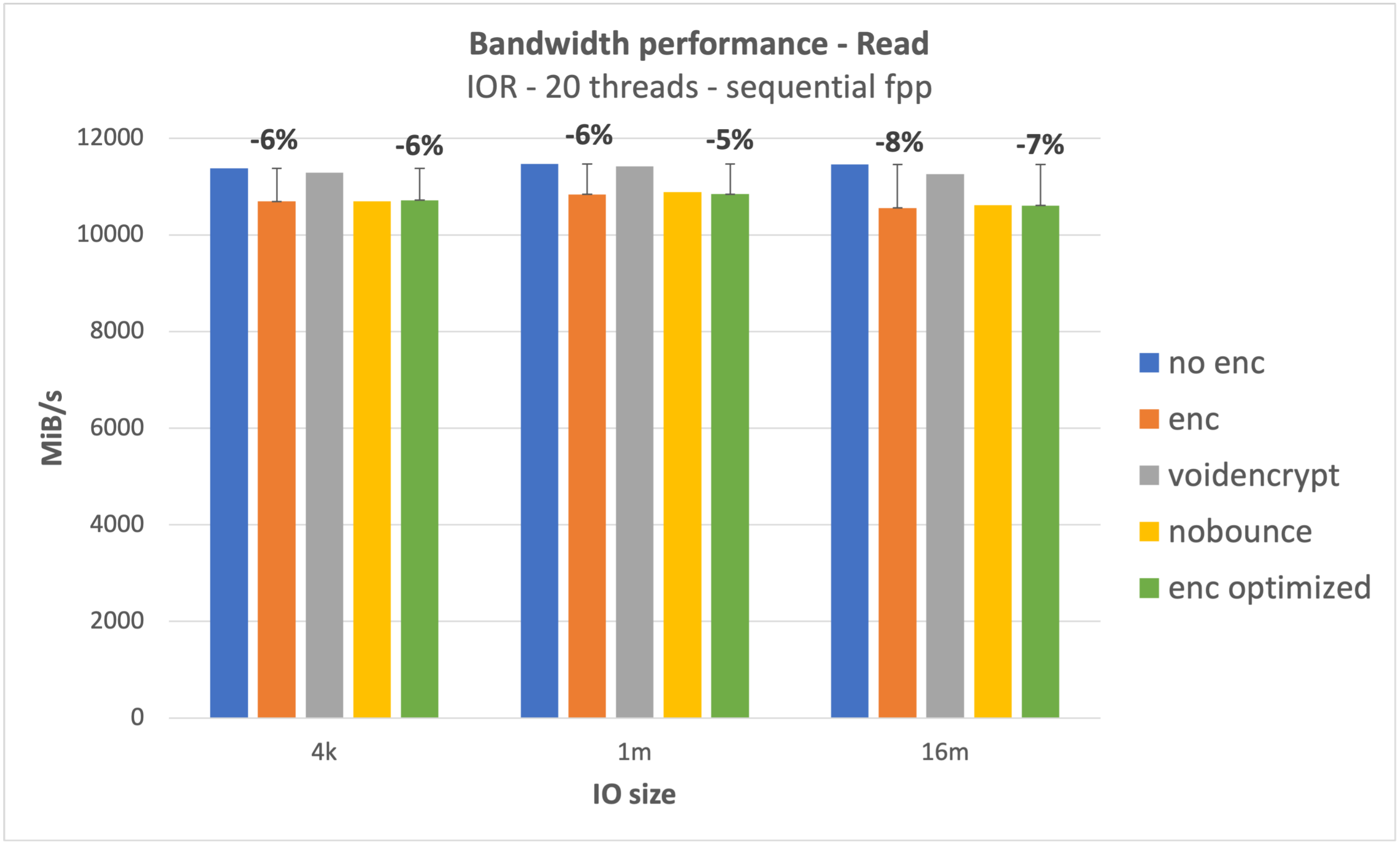
sbuisson@whamcloud.com

ddn

Lustre Client Encryption – performance

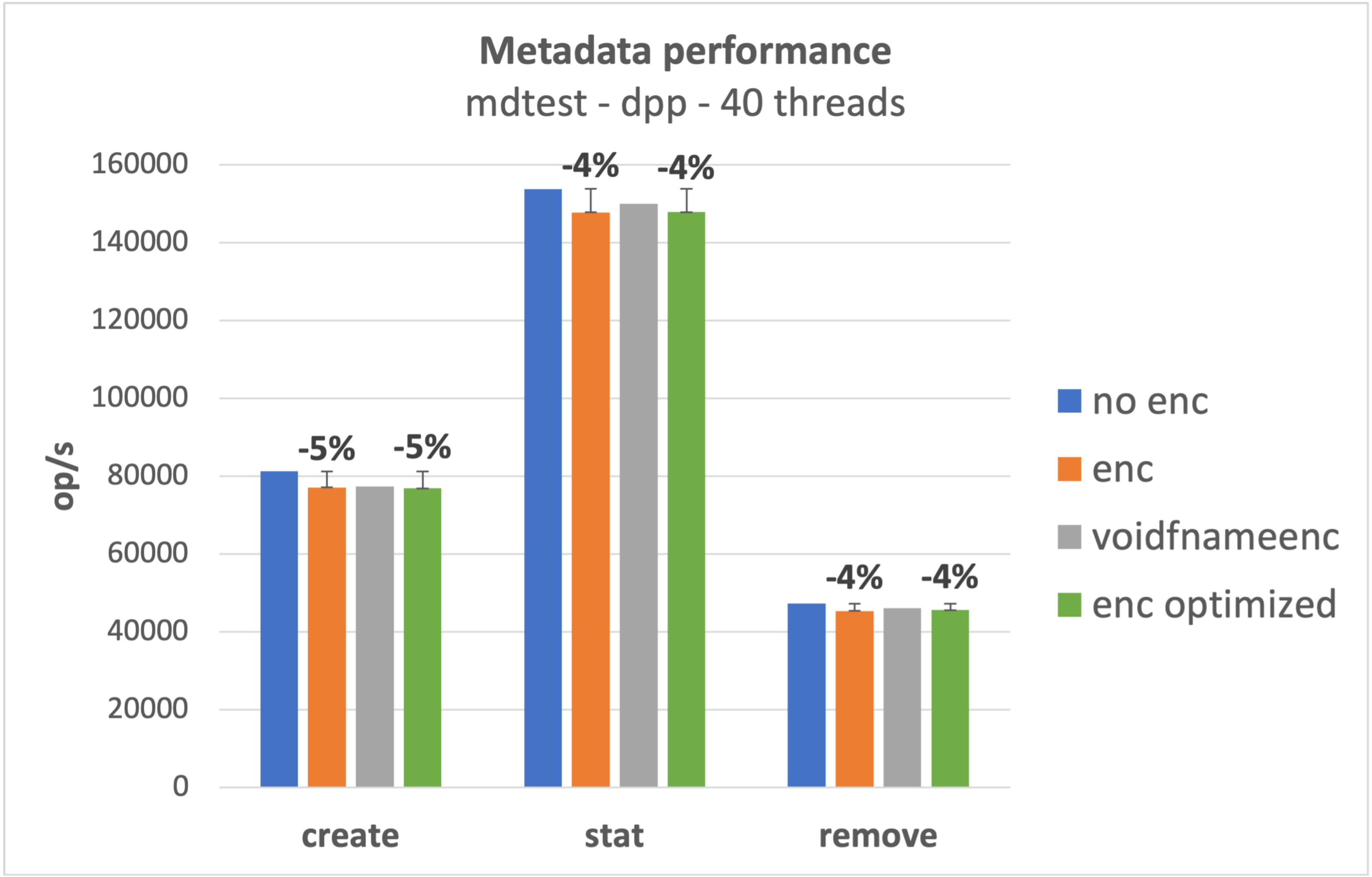
- **Initial benchmarks**
  - 30-35% drop in sequential write, 20-25% drop in sequential read
- **Testbed**
  - Client
    - Cascade Lake 20 cores, 6230 CPU @ 2.10GHz
    - 192 GB RAM
    - Infiniband adapter, EDR network
    - Ubuntu 20.04 kernel 5.4.0-107-generic
    - Lustre 2.15.0-RC3
  - Storage
    - 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%