Linux Kernel Encryption Support for File system

Kyungsik Lee
SW Platform Lab., Corporate R&D
LG Electronics, Inc.

2016/10/20

Mobile Security

- Mobile Security is an important issue
  - More data could be more danger with mobile devices

- Android 6.0 FDE(full-disk encryption)
  - User data protected against offline attacks
  - Plaintext -> ciphertext
  - Based on a **Linux Kernel Encryption feature** that works at the block device layer

Performance Issue (1/2)

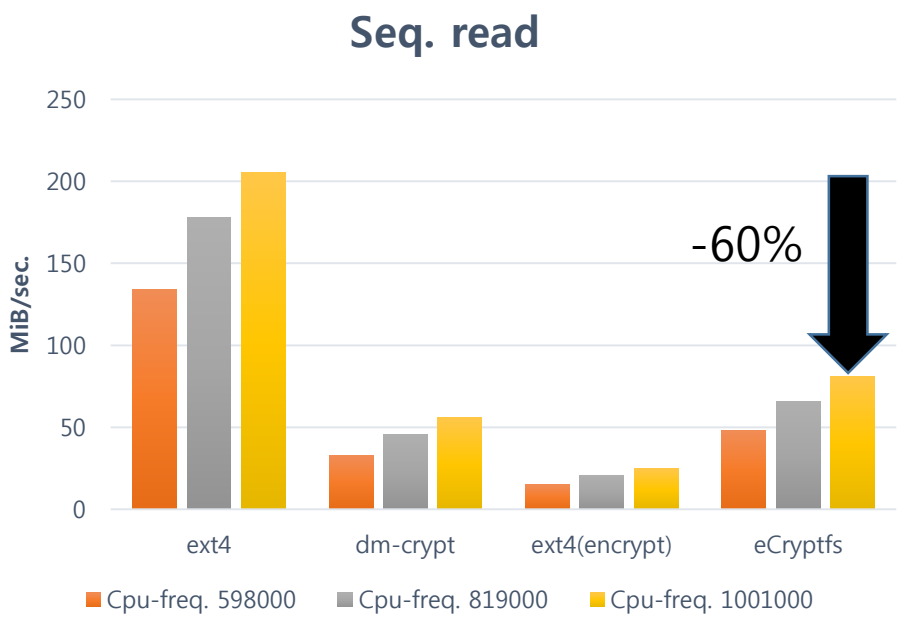
- Android 5.0(Lollipop) was to have device encryption enabled by default but …

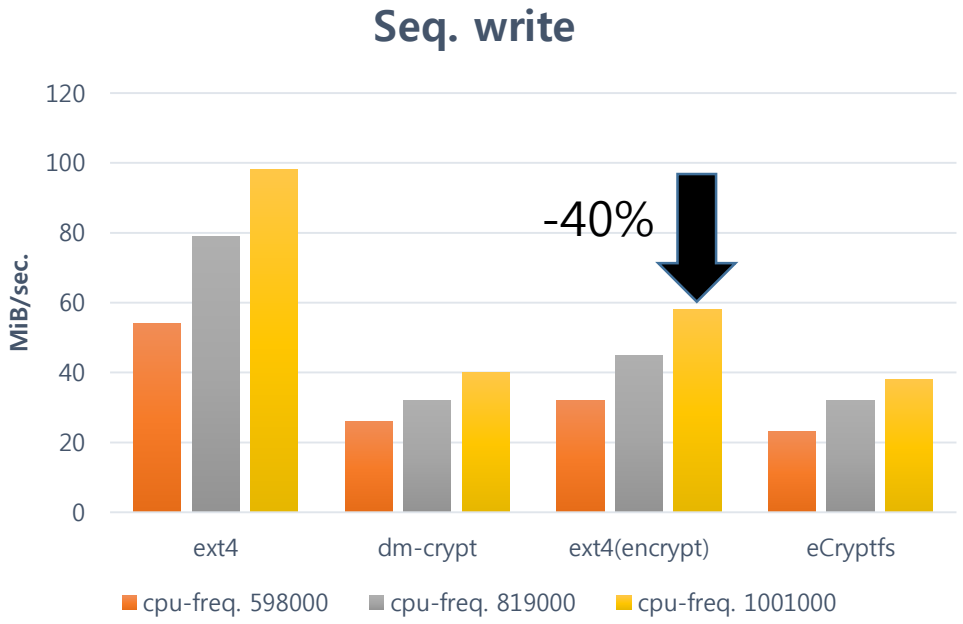
- According to Android 6.0 CDD

> **For device implementations supporting full-disk encryption and with Advanced Encryption Standard (AES) crypto performance above 50MiB/sec**, **the full-disk encryption MUST be enabled by default** at the time the user has completed the out-of-box setup experience
>
> Excerpted from Android 6.0 Compatibility Definition Document

Performance Issue (2/2)

- Sequential IO Read/Write
  - 1 CPU core, freq.(0.6~1 GHz)





Linux Kernel Encryption (1/2)

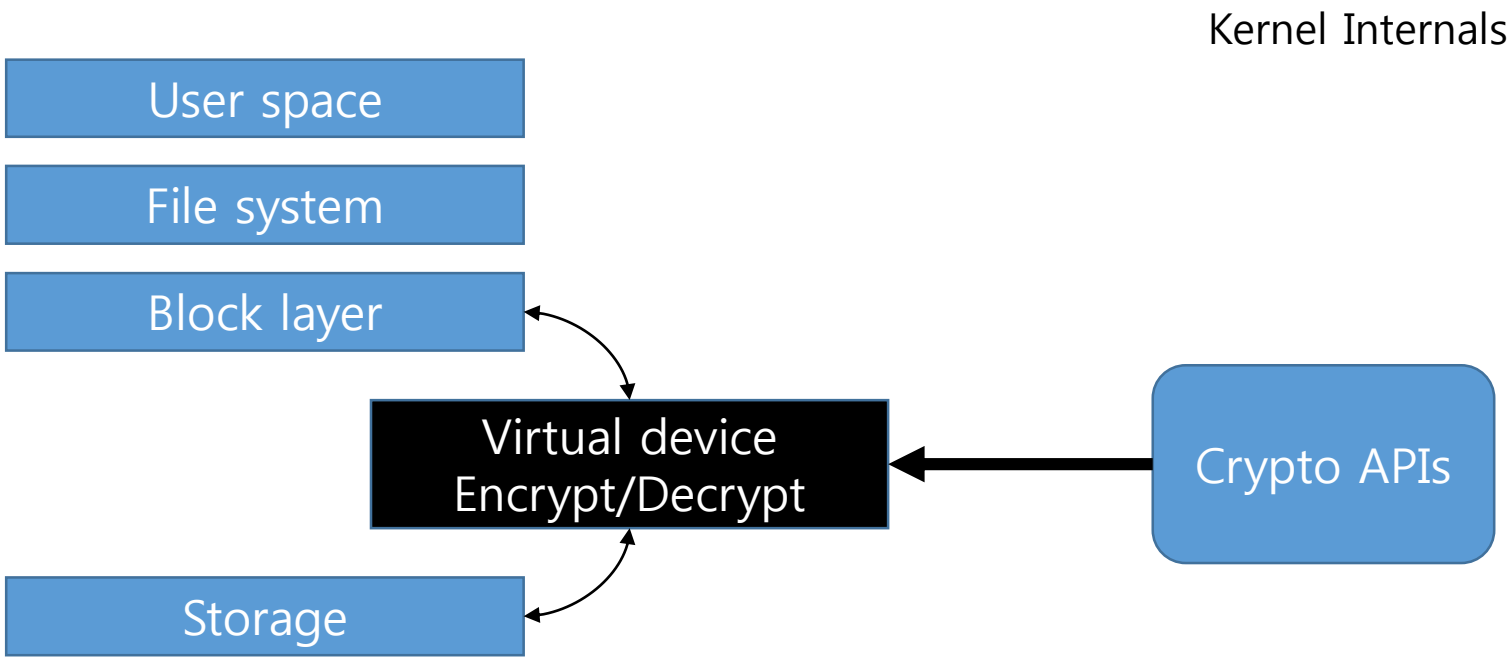
- History
  - dm-crypt, merged into 2.6.4 kernel(March, 2004)
  - eCryptfs, 2.6.19 kernel(November, 2006)
  - Ext4 encryption, 4.1 kernel(Jun, 2015)
  - VFS Crypto engine, 4.6 kernel
    => **Generic File system Encryption Support**

Linux Kernel Encryption (2/2)

- File system-level encryption, FBE
  - File-based encryption allows different files to be encrypted with **different keys** that can be unlocked independently.
  - File system-level encryption does not typically encrypt filesystem metadata
  - eCryptfs, ext4 encryption …
- Disk encryption, FDE
  - Disk encryption generally uses **the same key** for encrypting the whole volume, disk partition
  - dm-crypt …

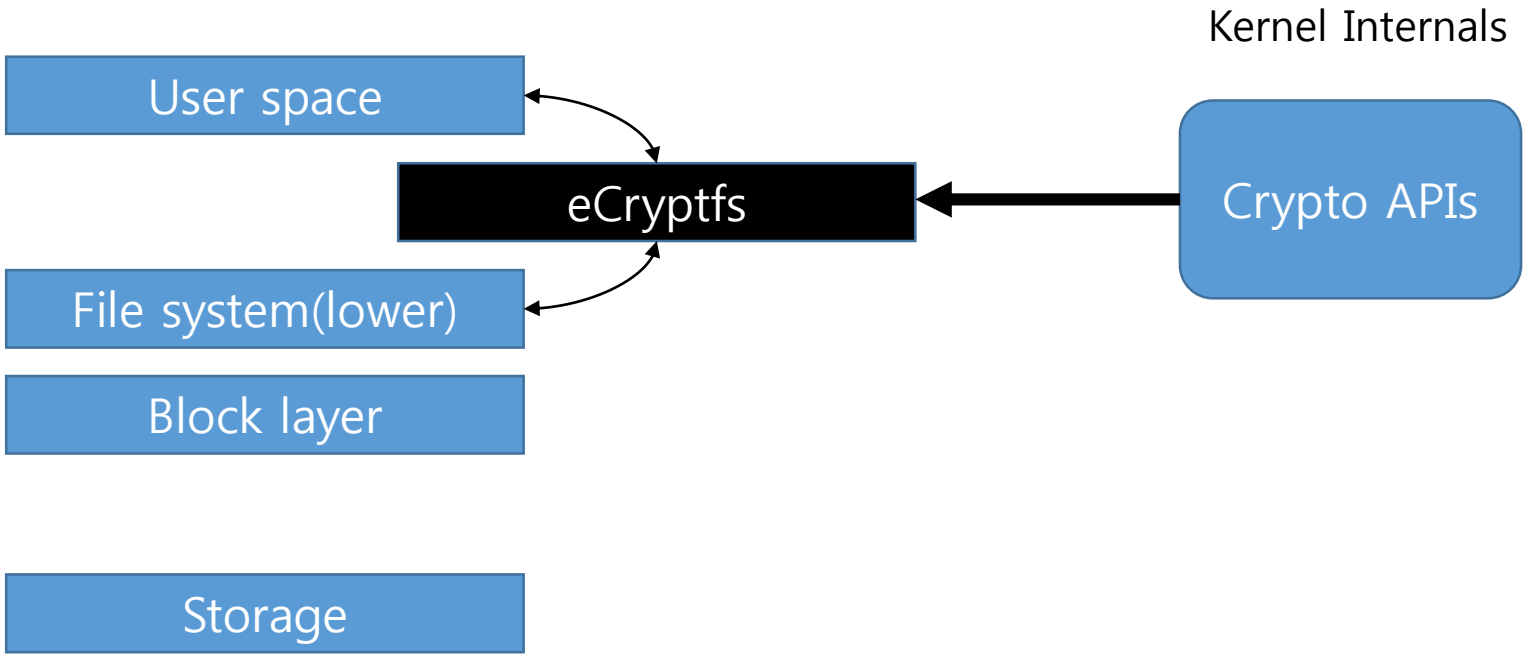
dm-crypt

- Part of the device mapper infrastructure, and uses cryptographic routines
- Encrypt whole disks (including removable media), partitions



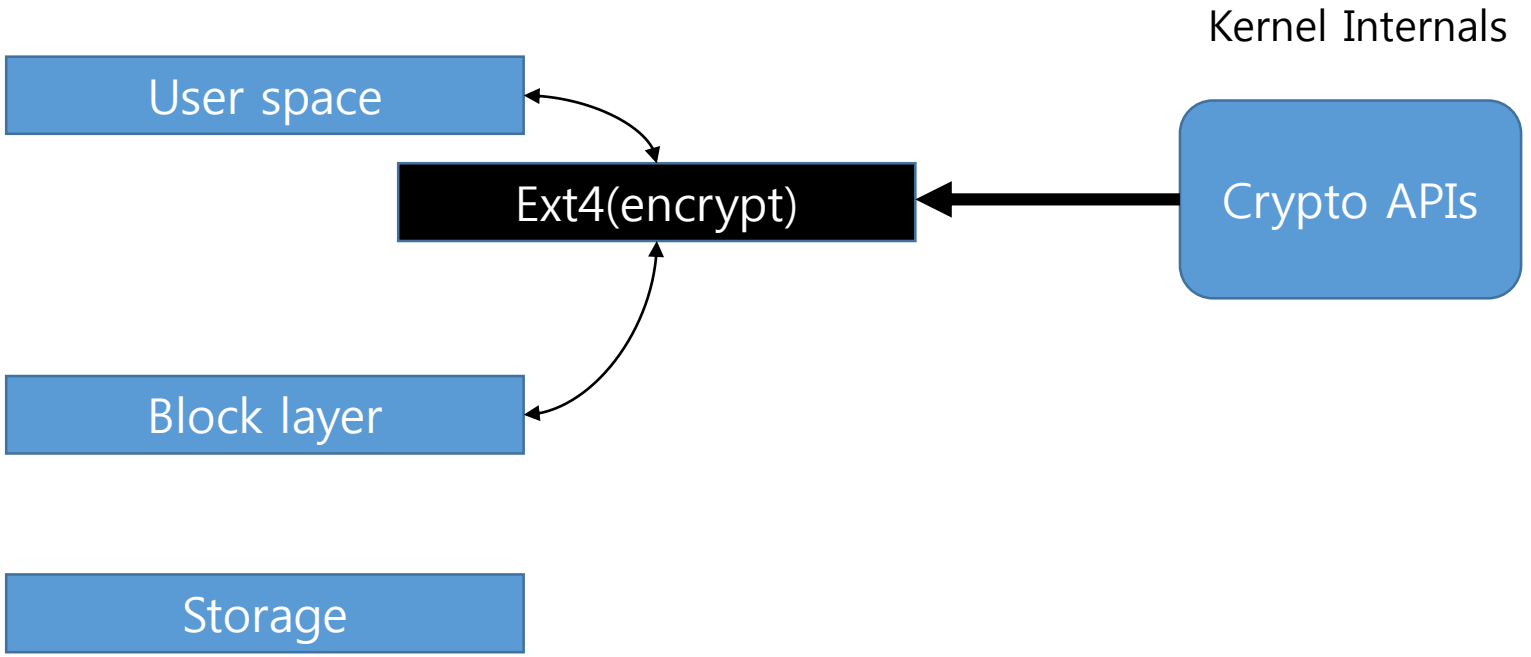
eCryptfs

- Stacked cryptographic file system
- Mount eCryptfs on top of any single directory to protect it



Ext4 Encryption

- In a directory tree marked for encryption, file contents, filenames, and symbolic link targets are all encrypted



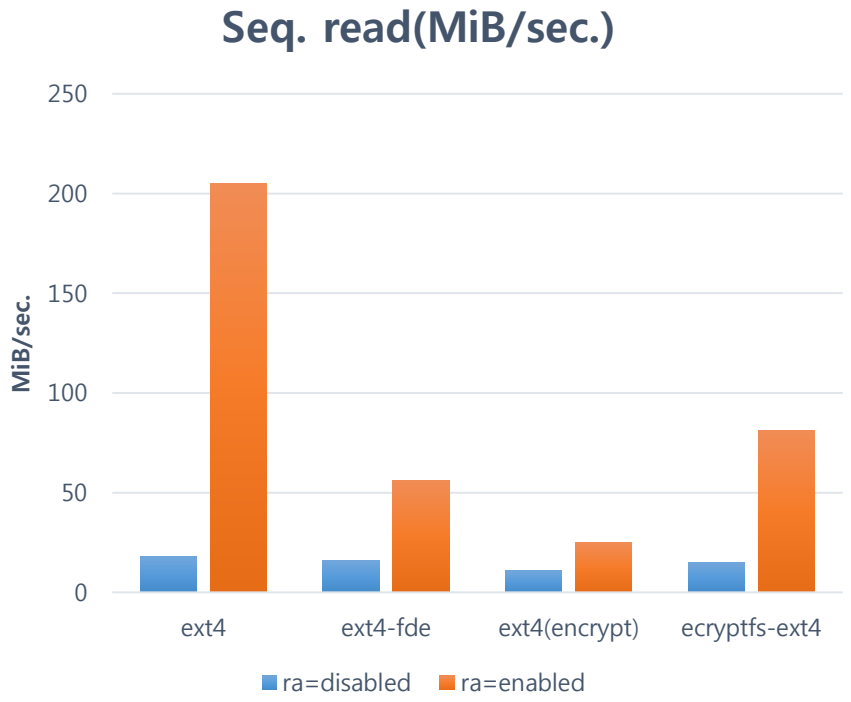
Case Study

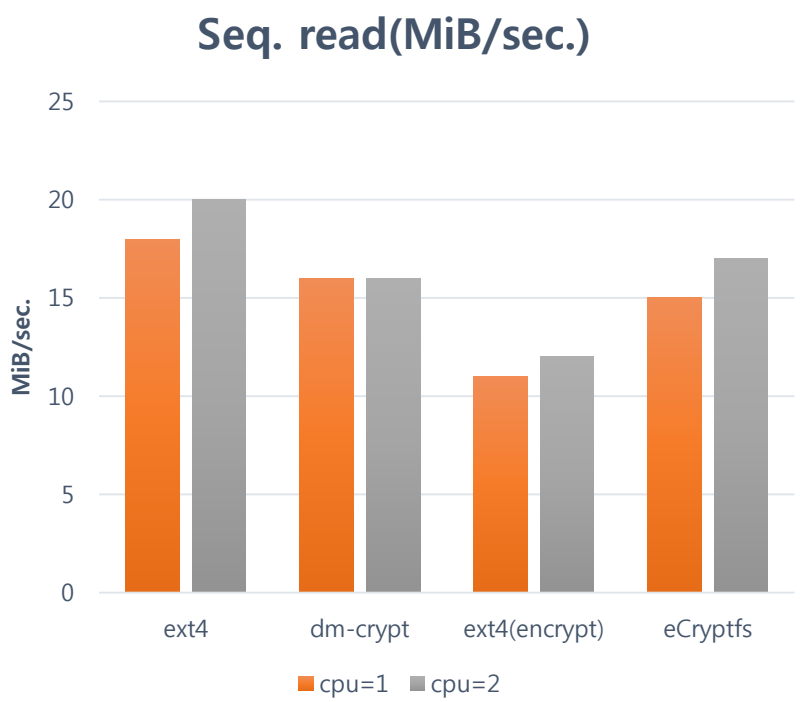
- Linux Kernel Encryption Scalability on multi-core system

- Testing Environment
  - CPU core(x4), freq.(0.6 ~ 1 GHz)
  - CPU based encryption
  - Cipher type
    - ✓ eCryptfs, aes-cbc
    - ✓ Ext4-encrypt, aes-xts
    - ✓ dm-crypt, aes-cbc-essiv:sha256

Sequential Read Prefetching

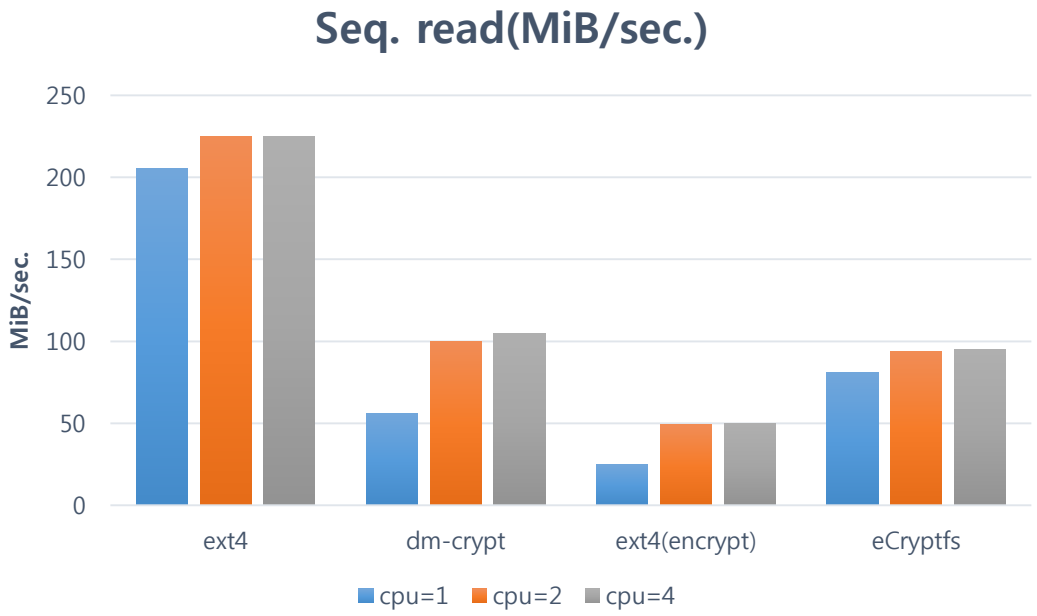
- Readahead





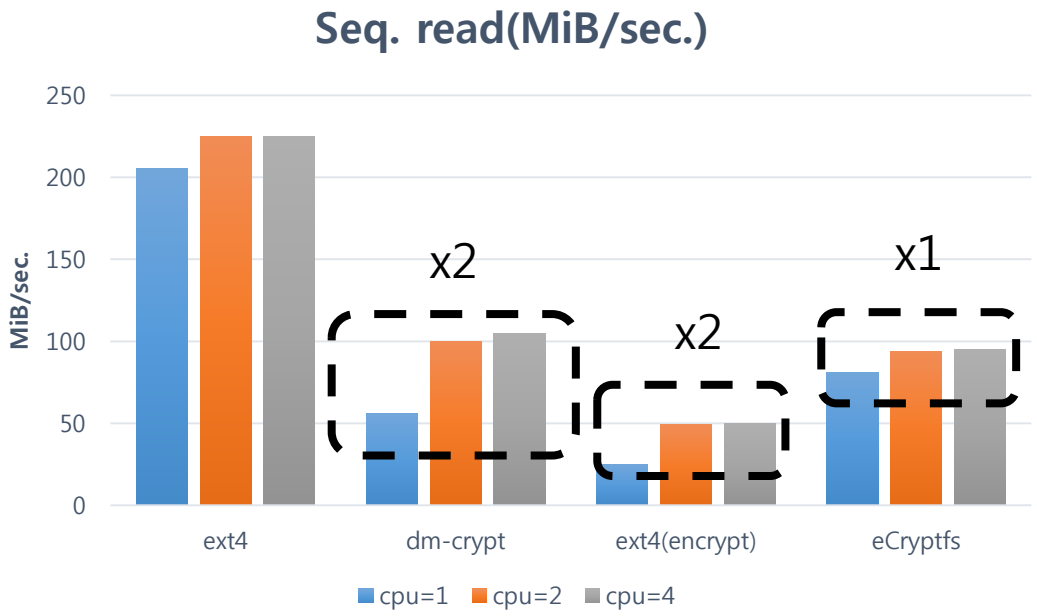
Read throughput

- CPU-cores(1/2/4)



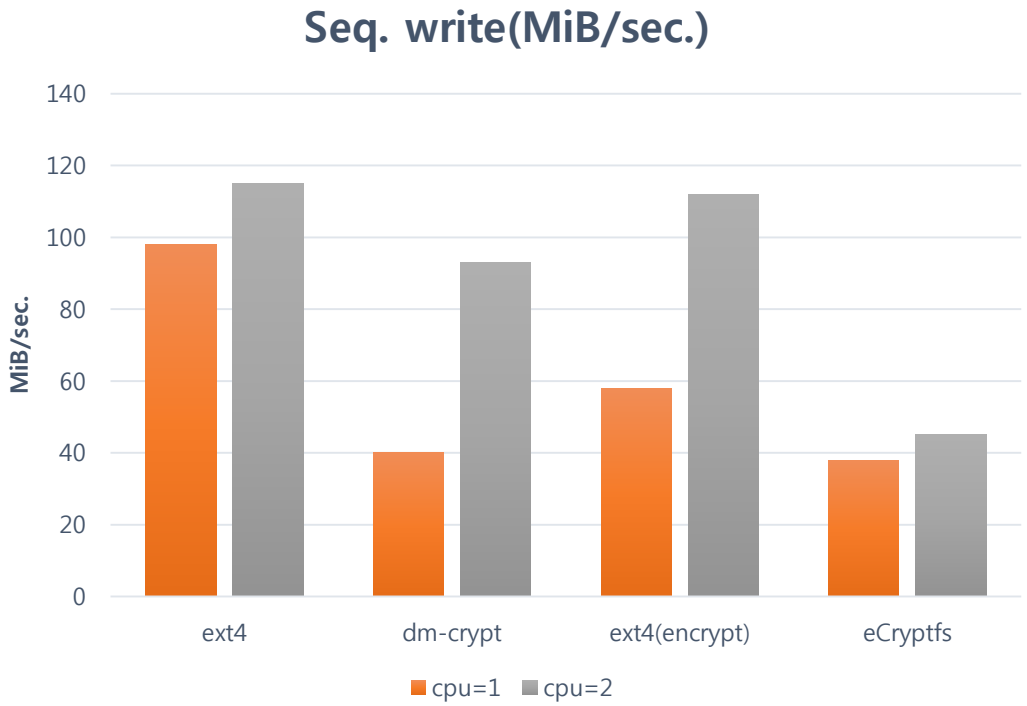
Read throughput

- CPU-cores(1/2/4)



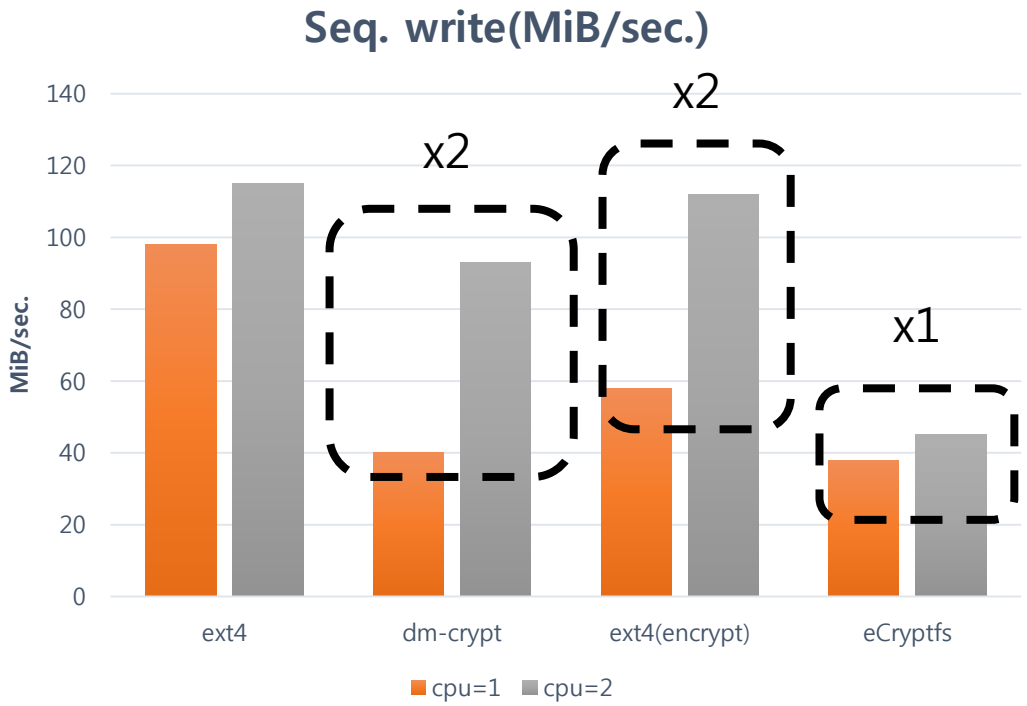
Write throughput

- CPU-cores(1/2)



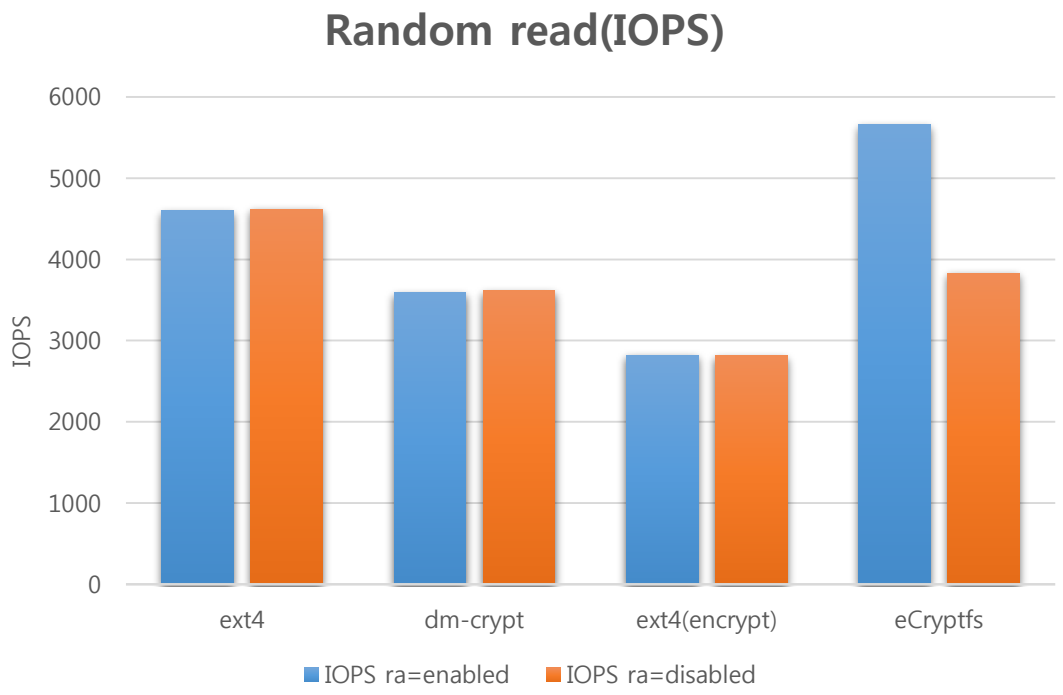
Write throughput

- CPU-cores(1/2)



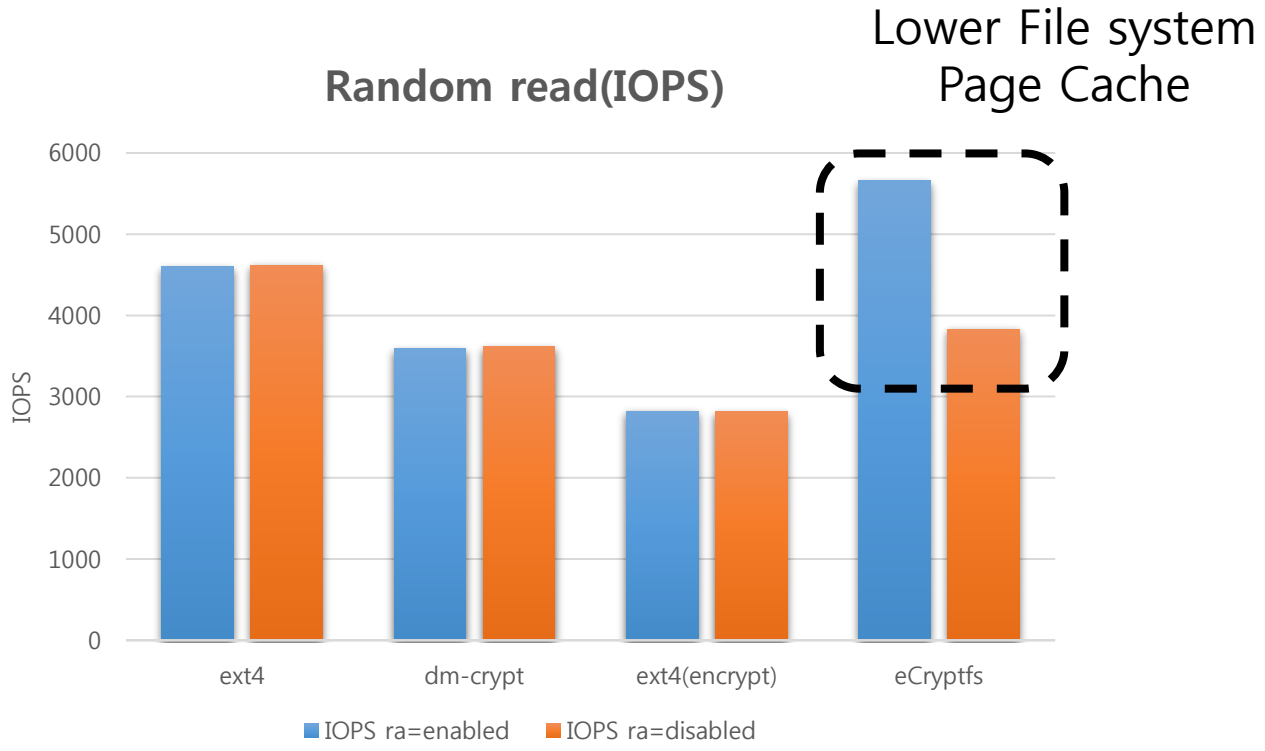
Random Read throughput

- Random read(IOPS)



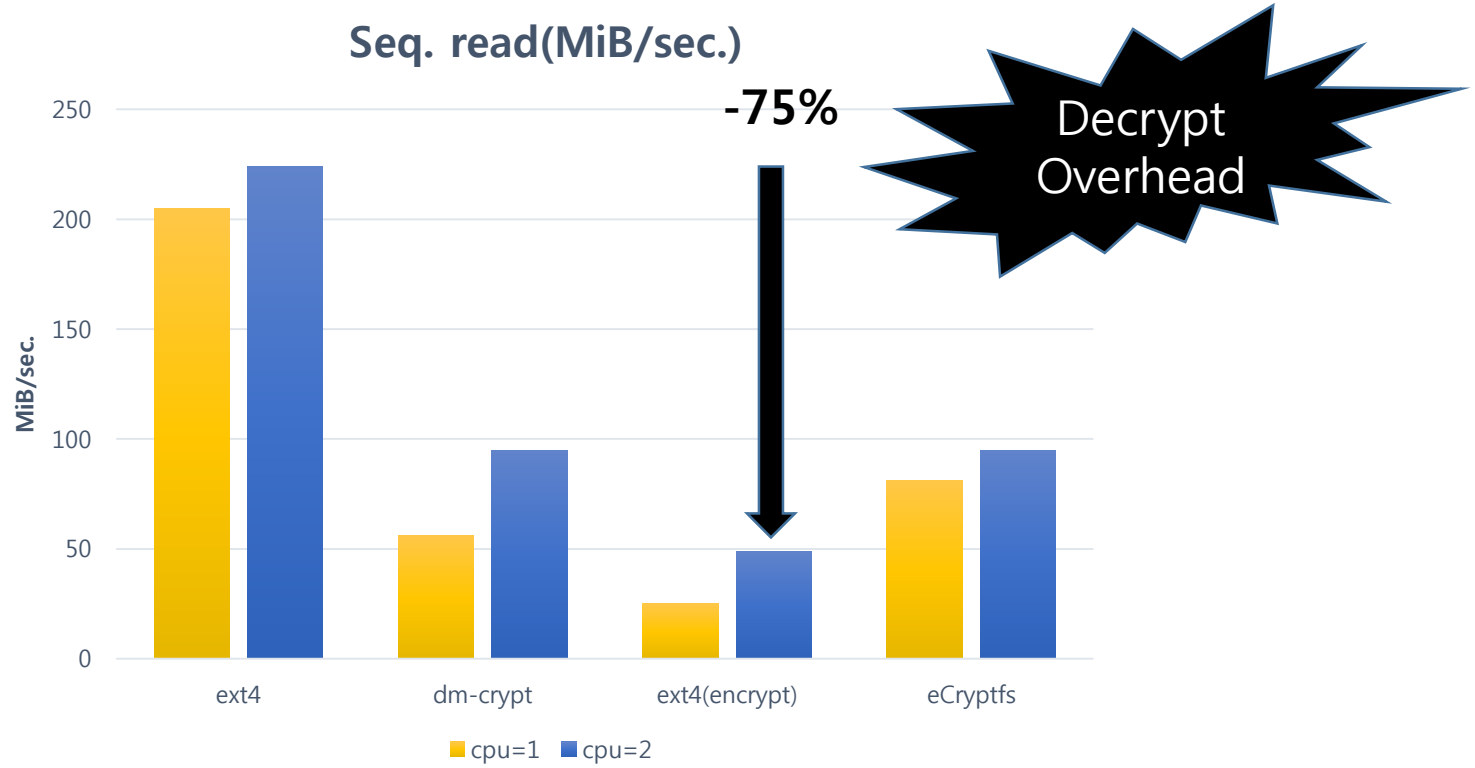
Random Read throughput

- Random read(IOPS)



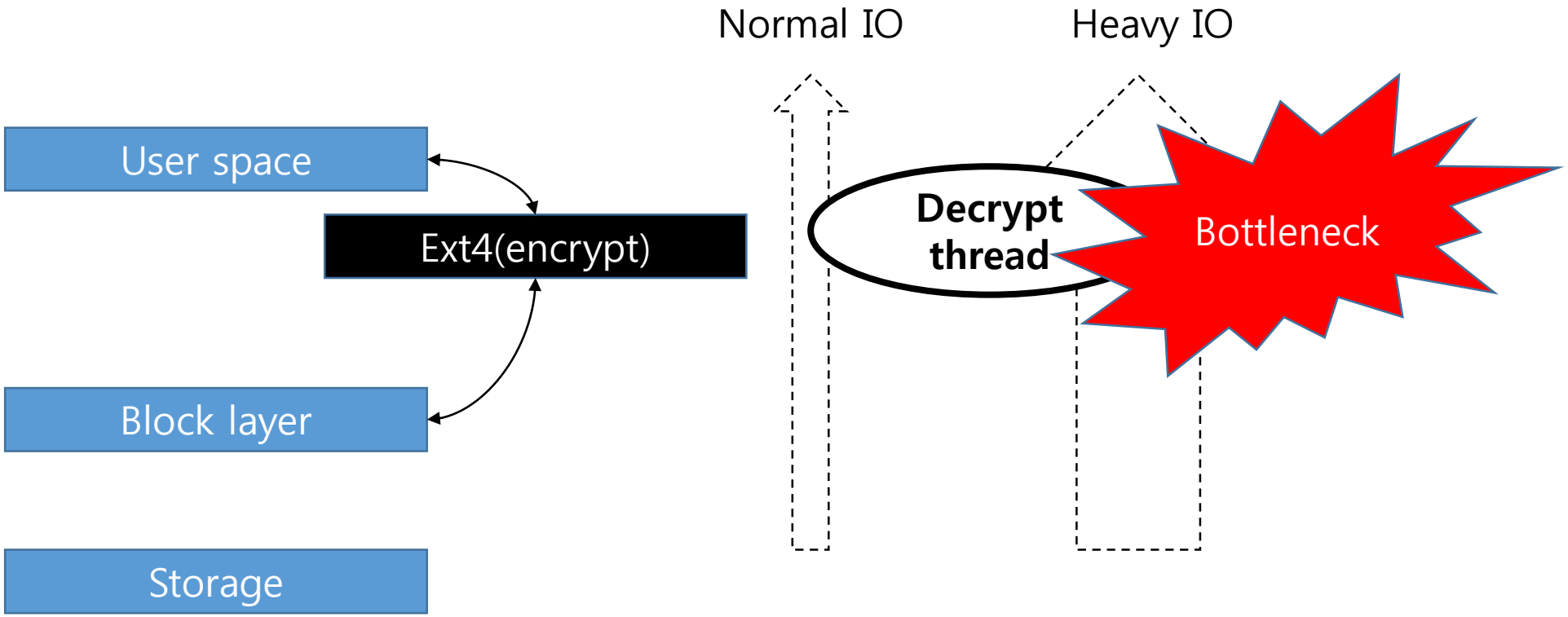
Improving Read performance (1/4)

- Ext4(encrypt) seq. read throughput



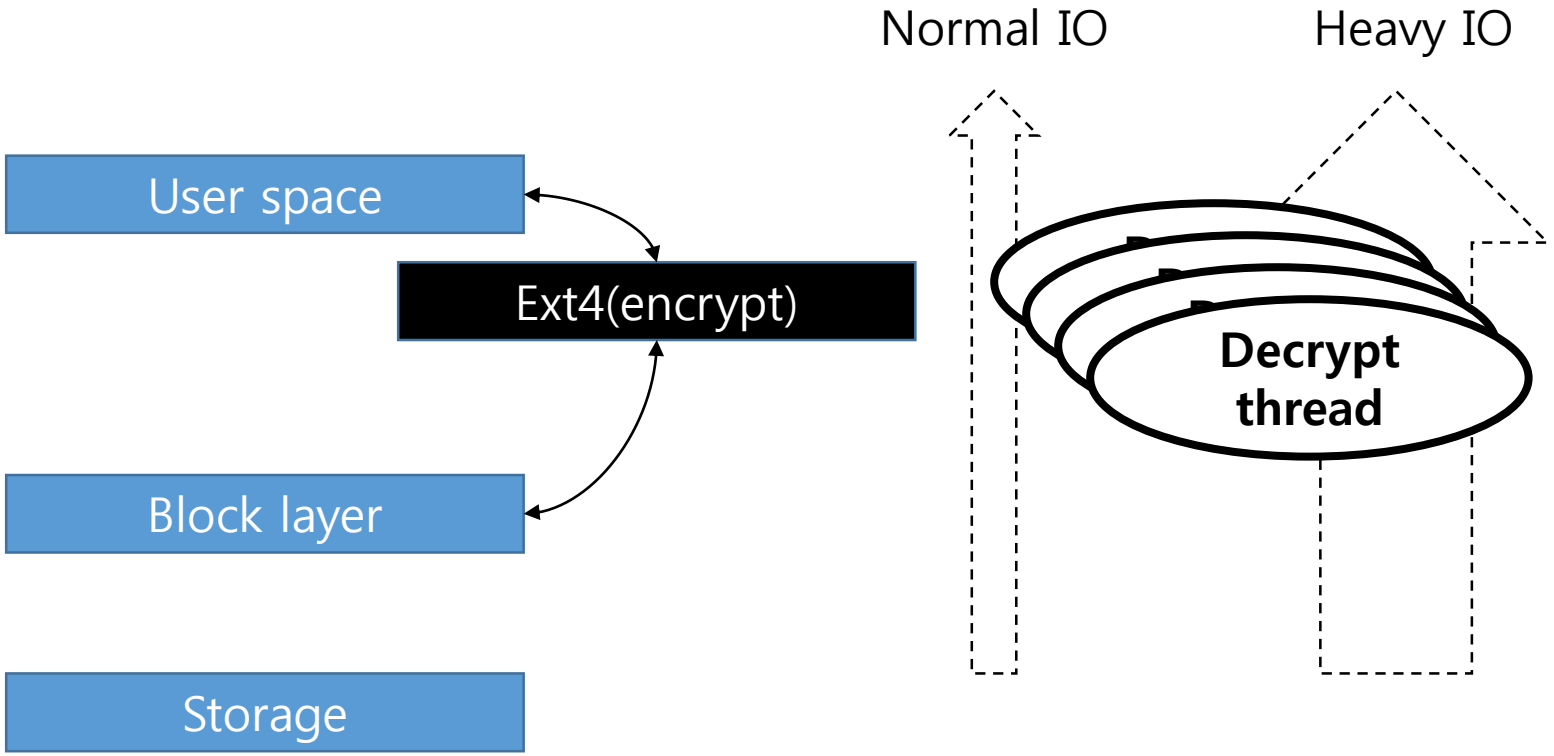
Improving Read performance (2/4)

- Multi-threaded decryption(ext4)



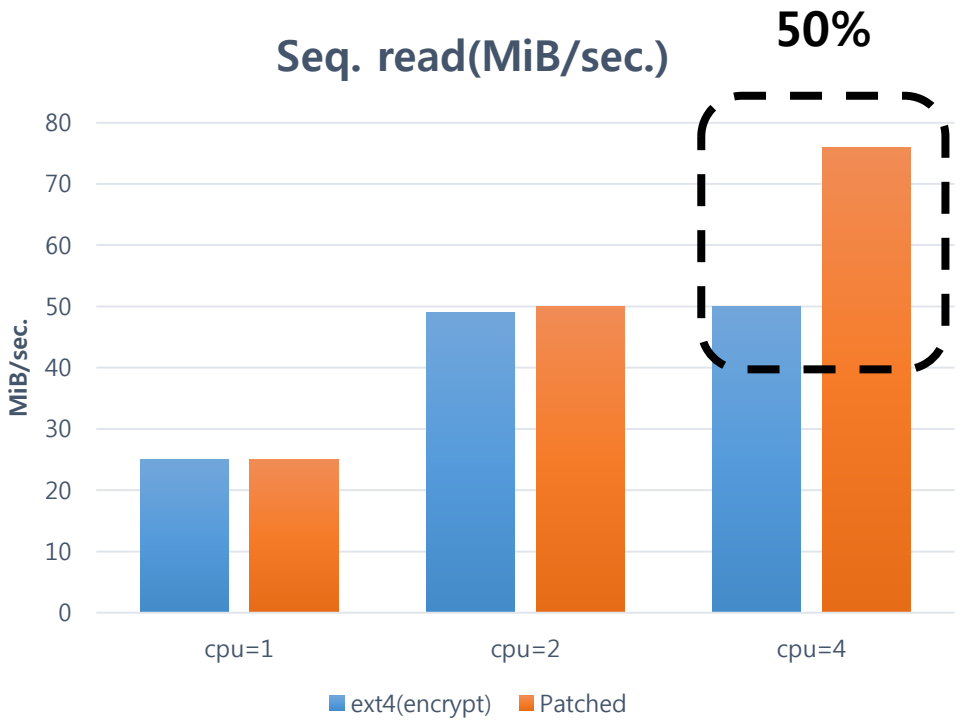
Improving Read performance (3/4)

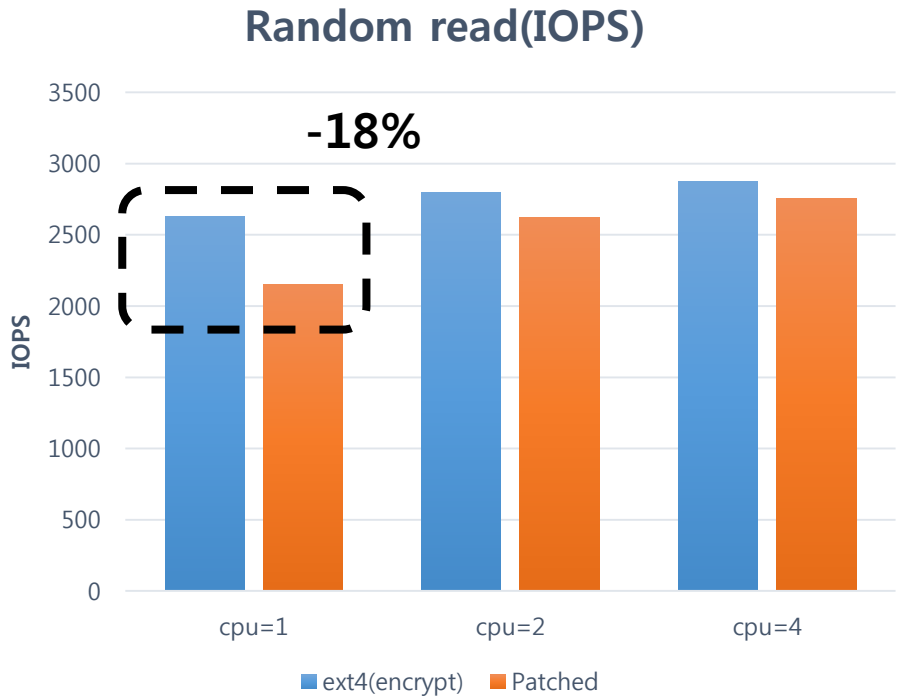
- Multi-threaded decryption(ext4)



Improving Read performance (4/4)

- Ext4(encrypt) seq. read throughput: +50%





Conclusion

- Seq. read throughput dropped significantly in CPU based encryption, leading to performance degradation
- Read(decrypt) overhead: seq. read >> random read
- Seq. write throughput falls slightly except eCryptfs
- IO throughput of eCryptfs is shown less scalable in multi-core system
- Seq. read performance can be improved by applying multi-threaded decryption

Q & A