Horus: Fine-Grained Encryption-Based Security for High Performance Petascale Storage

Yan Li, Nakul Dhotre, Yasuhiro Ohara,
Ethan L. Miller, Darrell D. E. Long
{yanli,nakul,yasu,elm,darrell}@cs.ucsc.edu

The Problem

- Large files contain potentially sensitive data
- File data can be leaked by many HPC elements (disk, client, metadata server)
- Ensure data confidentiality in the face of physical and software attacks

Design Principles

Prevent compromise by metadata server and storage nodes

- Encrypt / decrypt all data at the client

Restrict client leaks to only parts of the file to which the client has access - Most clients don't need access to the whole file

Provide a small, stateless trusted computing base

Hierarchical Keyed Hash Tree (KHT)

Single file root key can encrypt / decrypt the entire file

Successively lower keys in the tree are based on a keyed hash depending on

- Parent key
- Level in the tree
- Position in the level

Deriving keys lower in the tree is fast and simple

Deriving keys higher in the key or at the same level is "difficult"

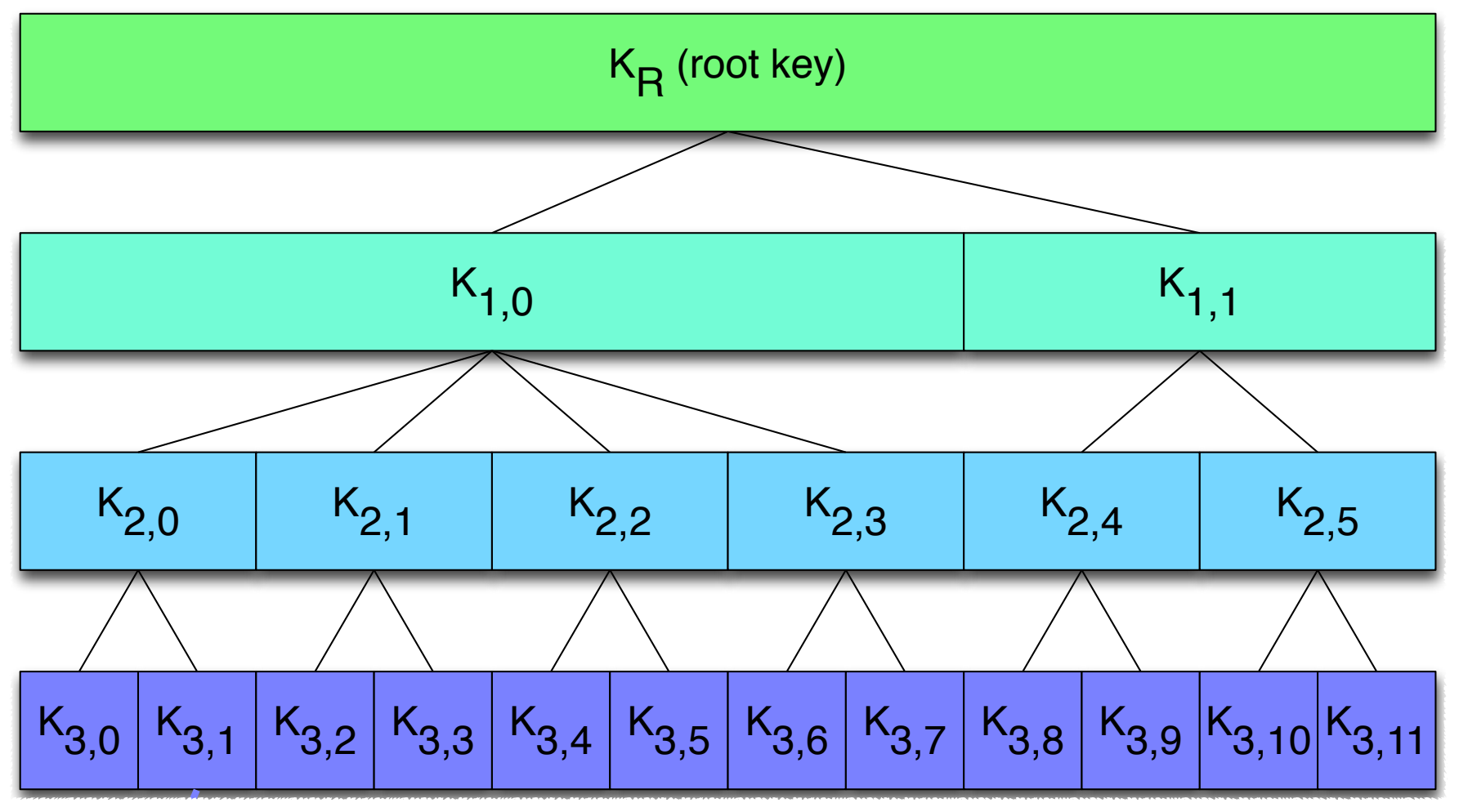


Figure 1: KHT

Evaluation

Machine setup: Intel(R) Xeon(R) CPU E5620 2.40GHz, Mem 24GB, Seagate® Constellation.2™ SATA. Running Fedora 16 Linux in x86-64 mode. Implemented as user-space library. Using Intel AESNI acceleration instructions.

Requesting Keys from KDS via RPC

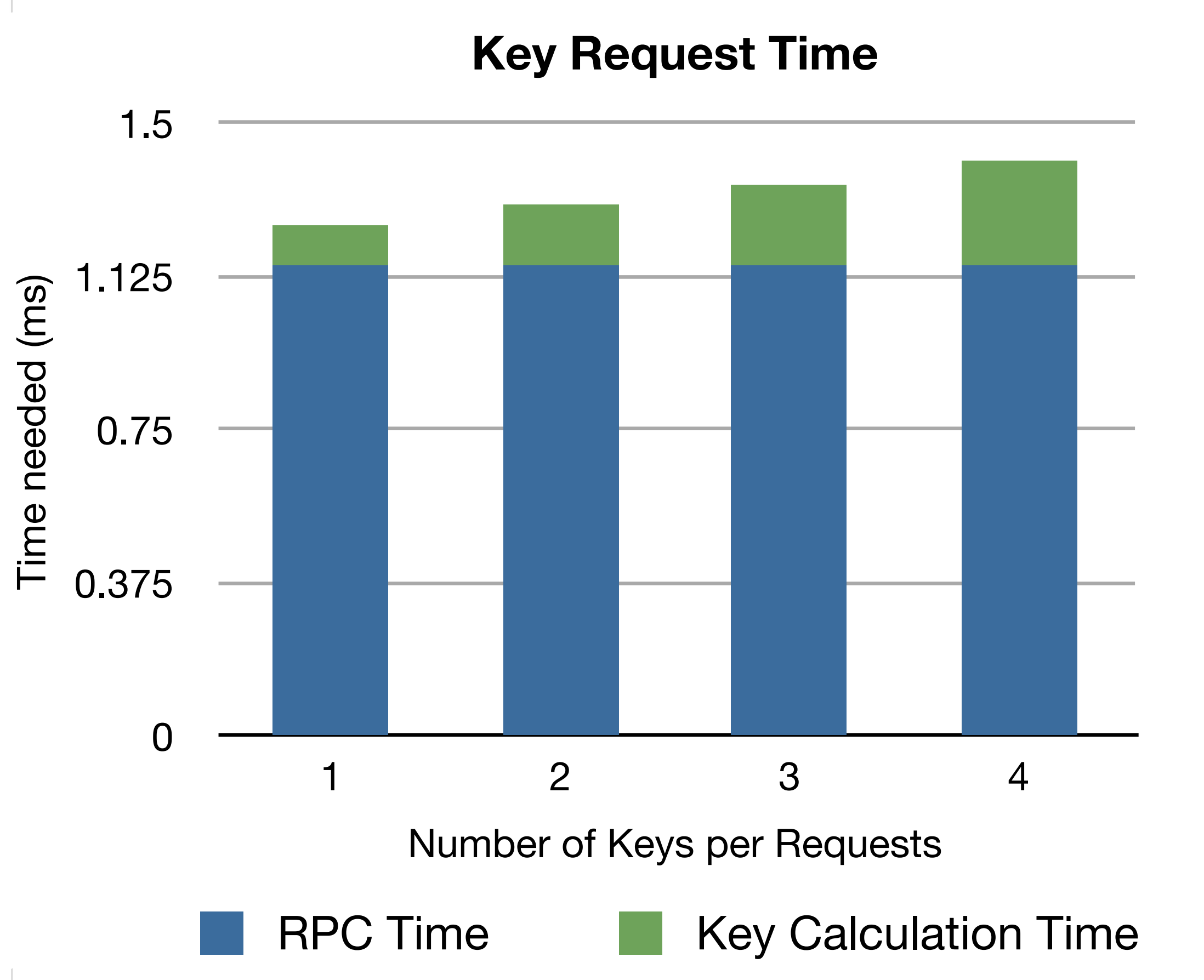


Figure 2

KHT Hashes need per MB

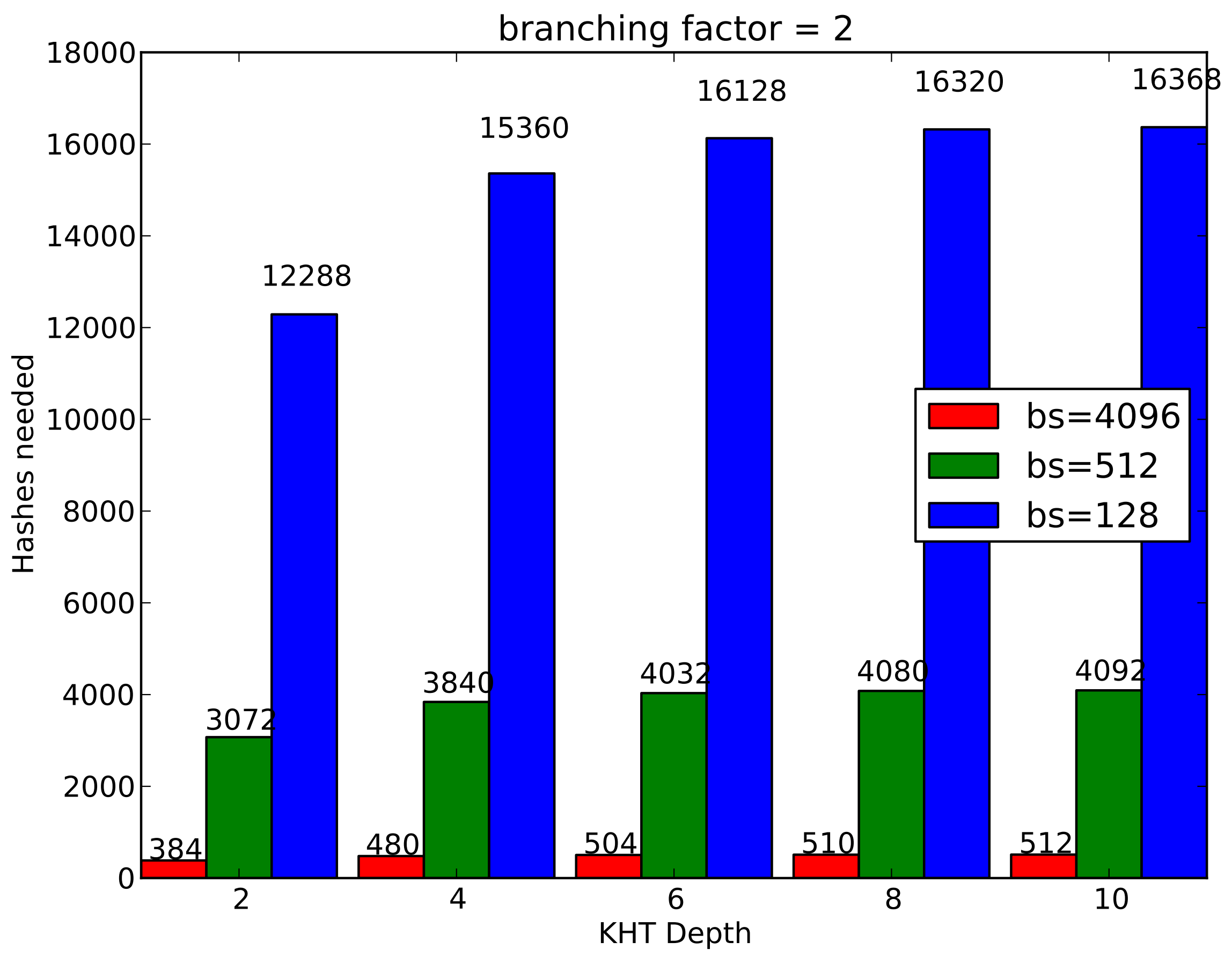


Figure 3

Key Calculation Overhead (for 2GB data, block size = 4096)

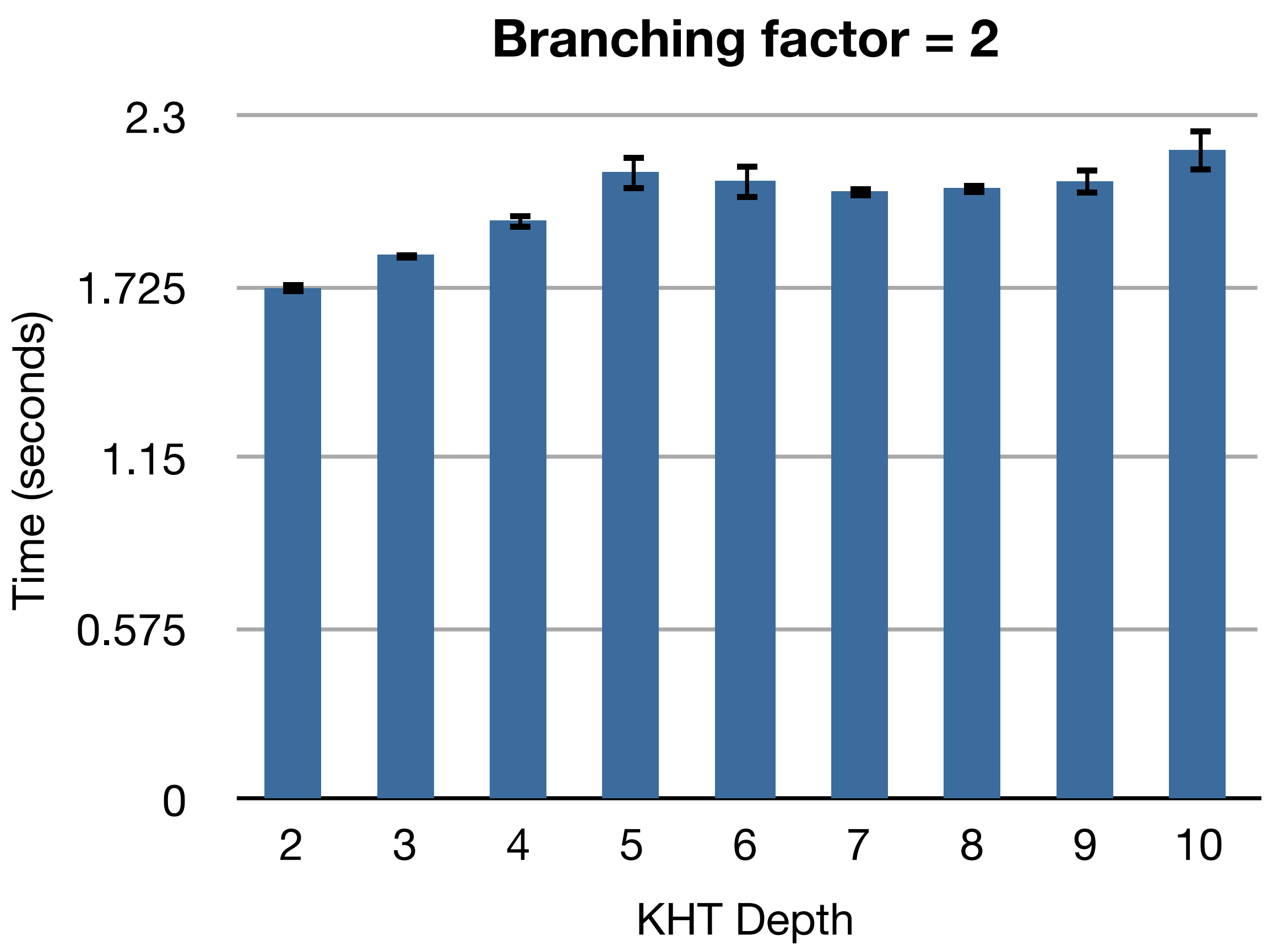


Figure 4

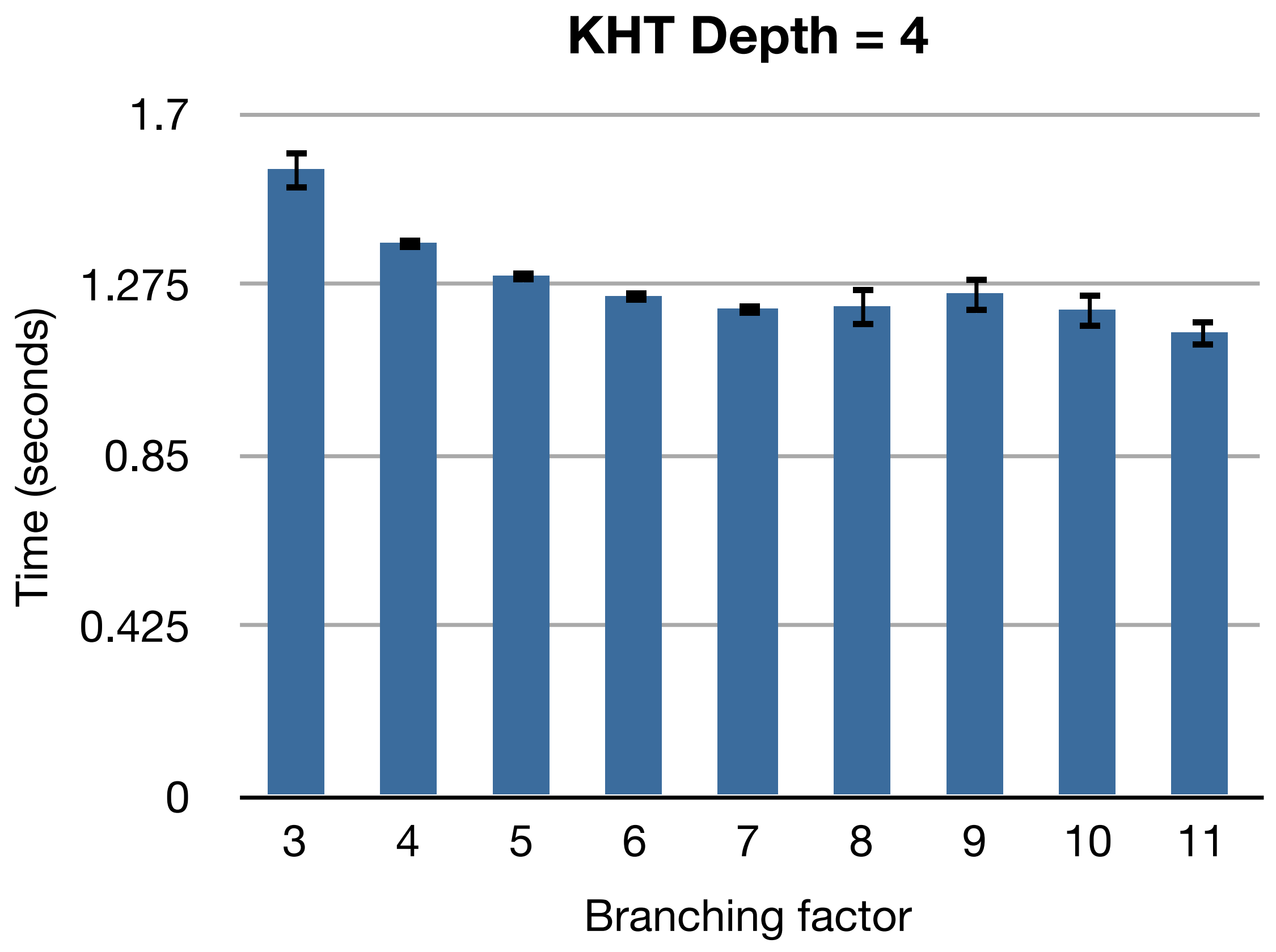


Figure 5

Read / Write Throughput (block size = 4096)

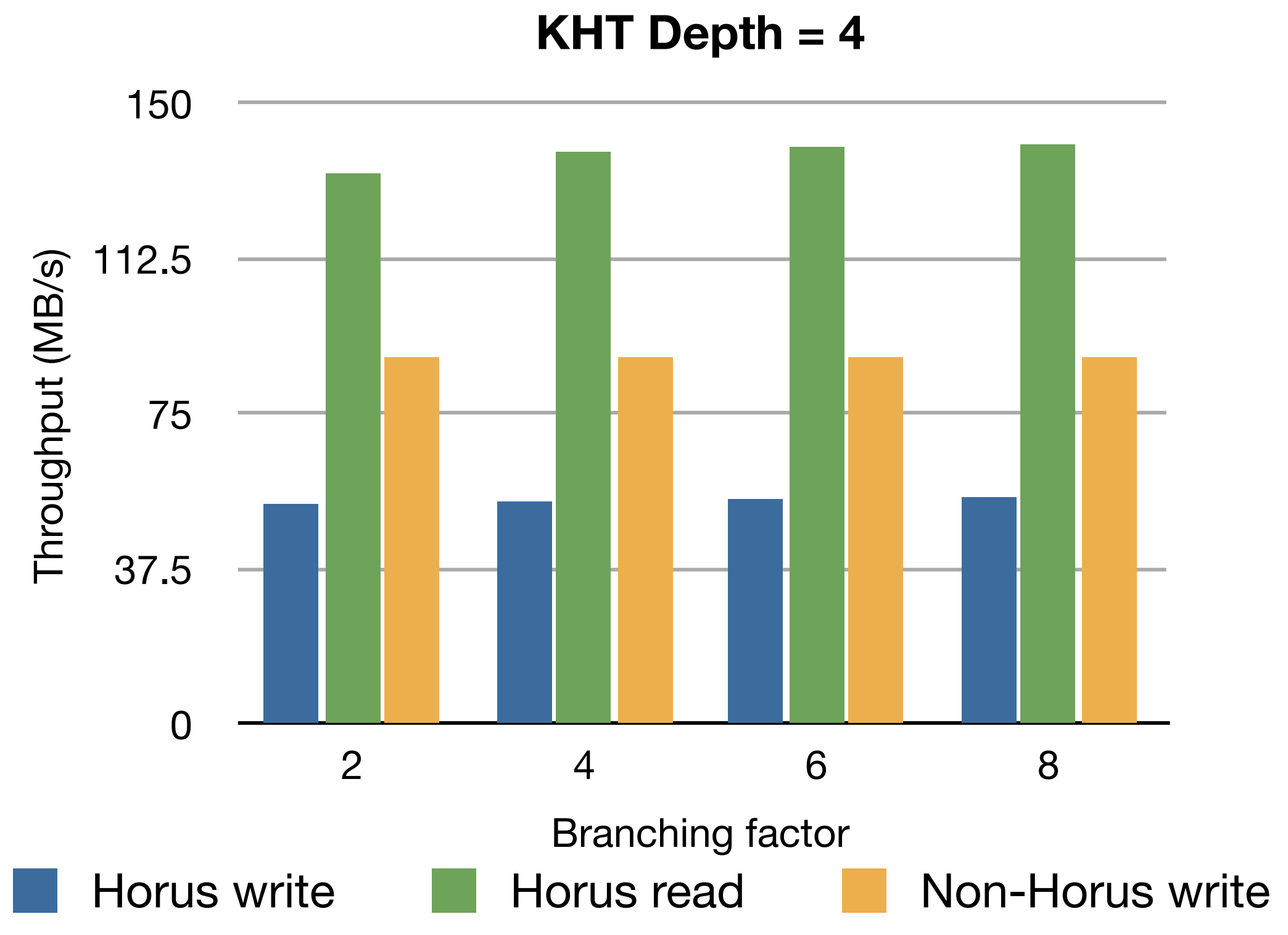


Figure 6

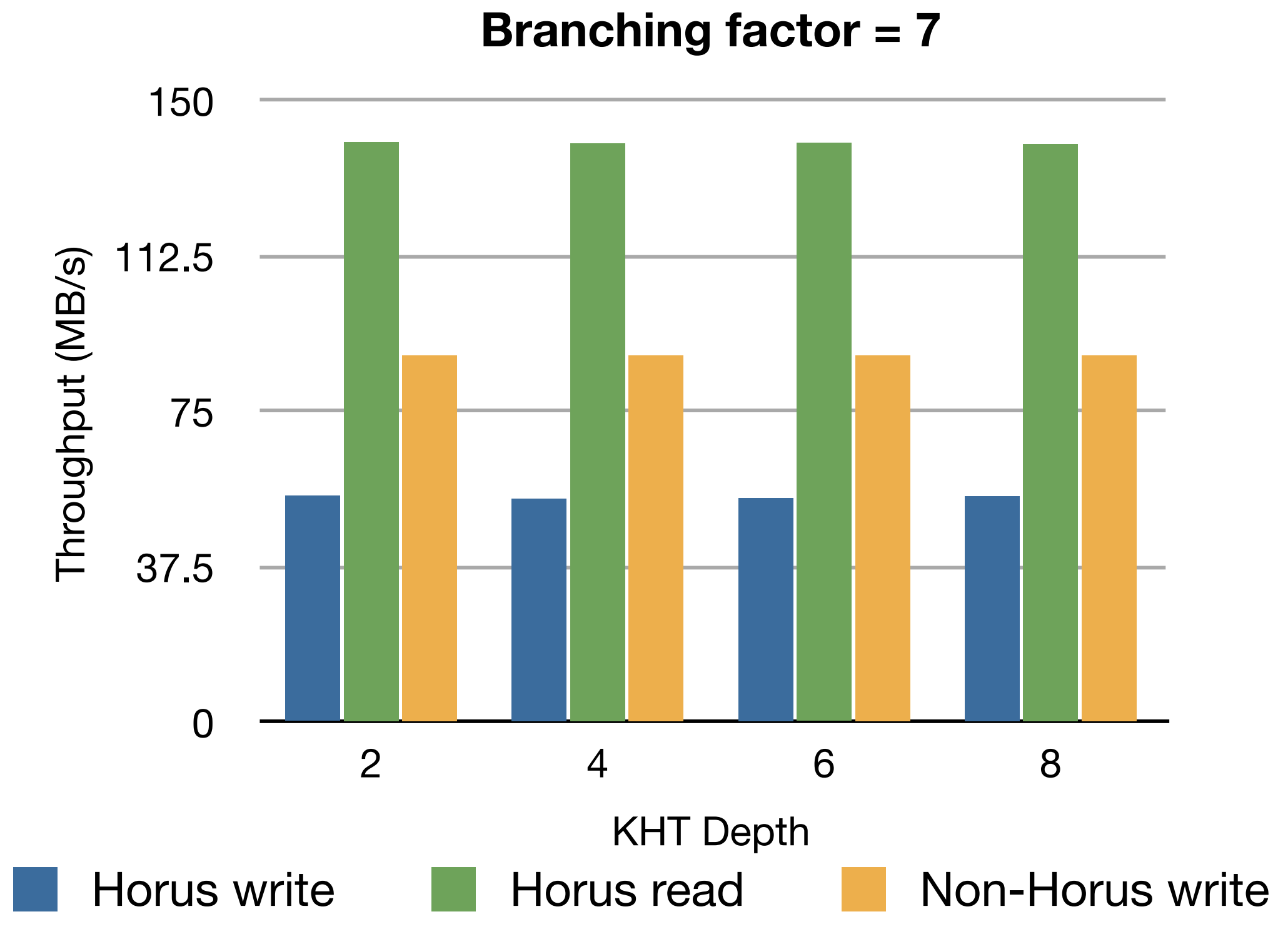


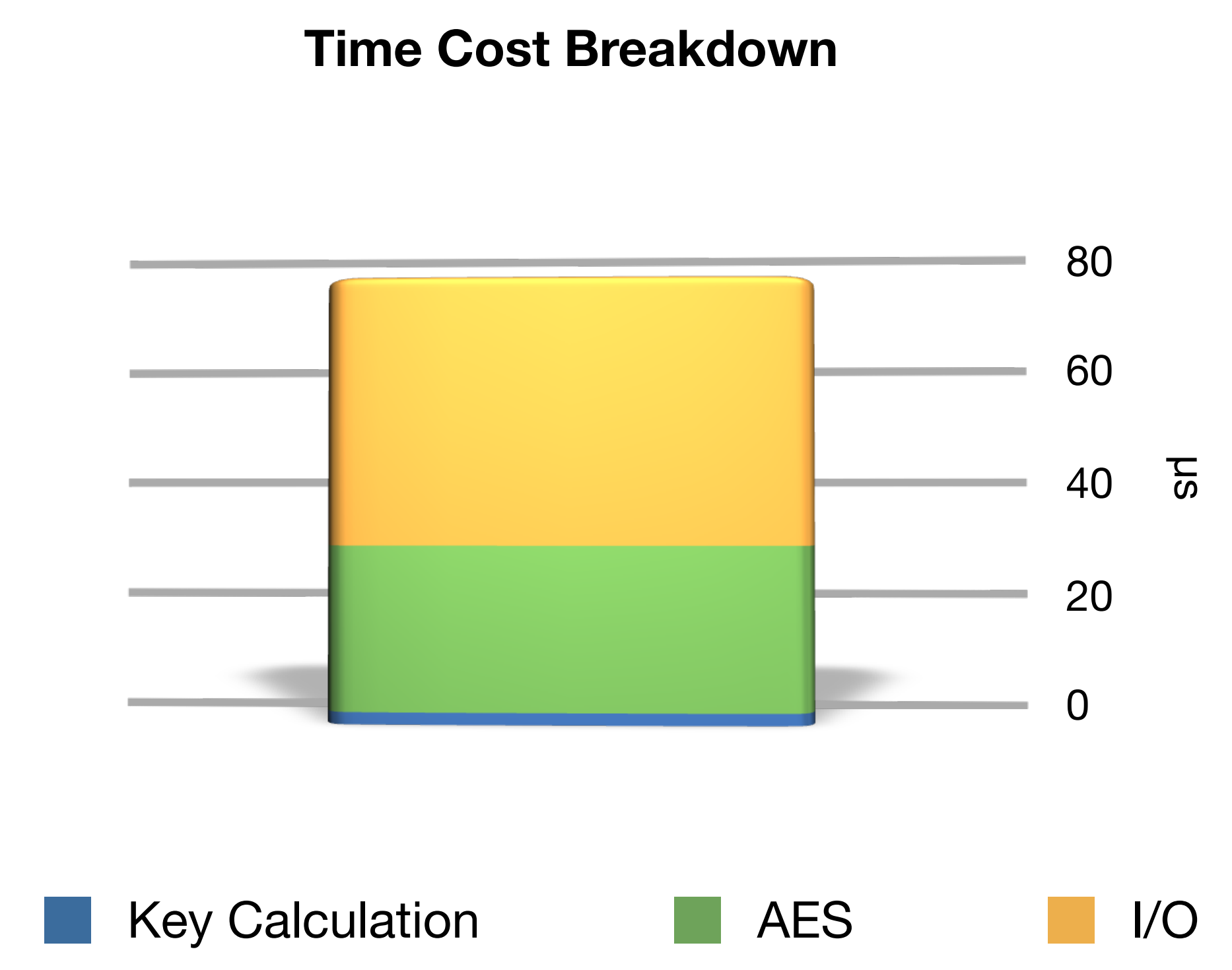
Figure 7

Time Cost Breakdown

Time for one keyed hash: 1.986 µs.

Time for one 4096-byte block AES (using Intel AESNI instructions): 27.265 µs.

Disk I/O: 44.047 µs



Ongoing Work

- Implementation in Linux file systems (using FUSE)
- Integration with Ceph
- Open source the prototype

Reference

Ranjana Rajendran, Ethan L. Miller, Darrell D. E. Long, Horus: Fine-Grained Encryption-Based Security for High Performance Petascale, PDSW'11

Conclusion

- With Horus, each client can only access the parts of large files that they are allowed to access
- Using KHT for key management is well understood and the performance penalty is reasonable

Acknowledgement

We would like to thank the faculty and students of the Storage Systems Research Center (SSRC), Center for Research in Intelligent Storage (CRIS), and Zhichao Hu from NLDS Lab, UCSC for their help and guidance. This research was supported in part by the NSF under awards IIP-0934401 and CCF-0937938, the Dept. of Energy under Award number DE-FC02-10ER26017/DE-SC0005417, and by the Dept. of Energy's Petascale Data Storage Institute under award DE- FC02-06ER25768. We also thank the industrial sponsors of the Storage Systems Research Center and the Center for Research in Intelligent Storage for their generous support.

Mar 14, 2012