ForestDB: A Fast Key-Value Storage System for Variable-Length Key Strings

Chiyoung Seo
Software Architect
Couchbase Inc.

B+Tree Limitations

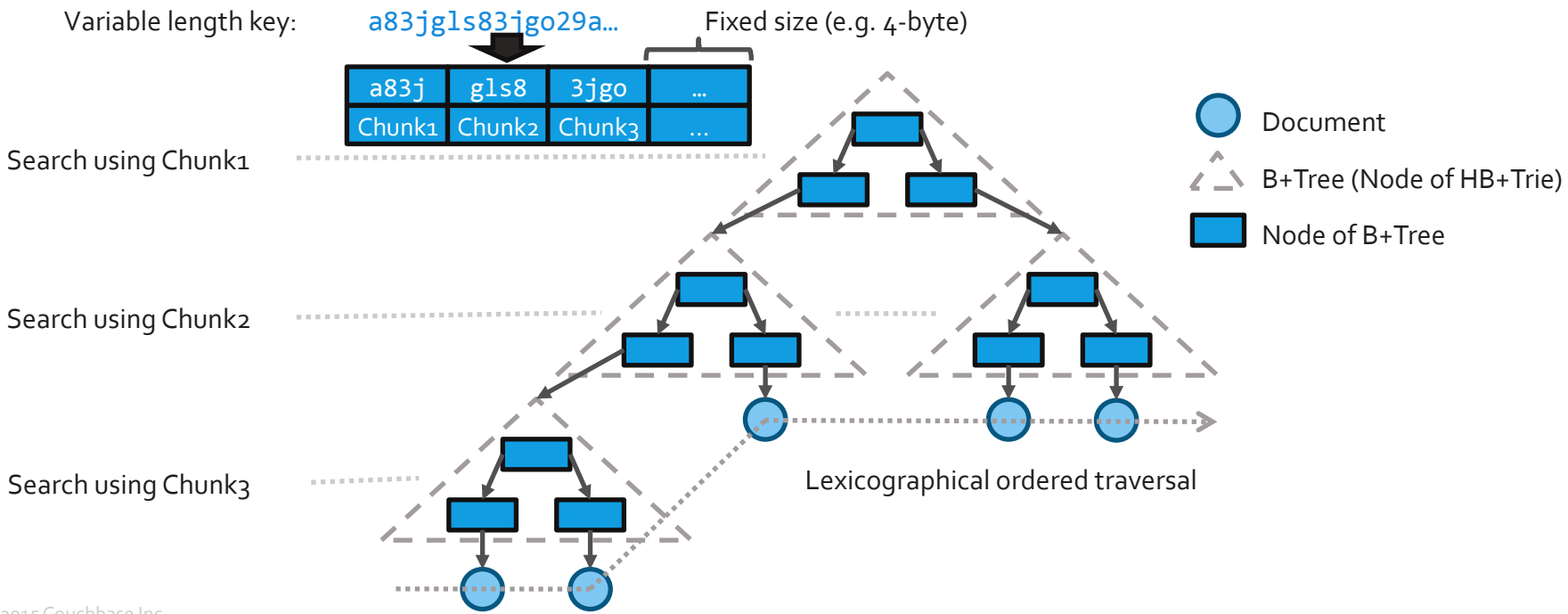
- Not suitable to index variable-length long keys
  - Significant space overhead as entire key strings are indexed in non-leaf nodes
- Tree depth grows quickly as more data is loaded
- I/O performance is degraded significantly as the data size gets bigger
- Several variants of B+Tree were proposed
  - LevelDB (Google)
  - RocksDB (Facebook)
  - TokuDB (Tokutek)
  - WiredTiger (MongoDB)

ForestDB

- Key-Value storage engine developed by Couchbase Caching / Storage team
- Its main index structure is built from *Hierarchical B+-Tree based Trie* or *HB+-Trie*
- Significantly better read and write performance with less storage overhead
- Support various server OSs (Centos, Ubuntu, Debian, Mac OS x, Windows) and mobile OSs (iOS, Android)
- ForestDB paper published in *IEEE Trans. On Computers*

HB+Trie (Hierarchical B+Tree based Trie)

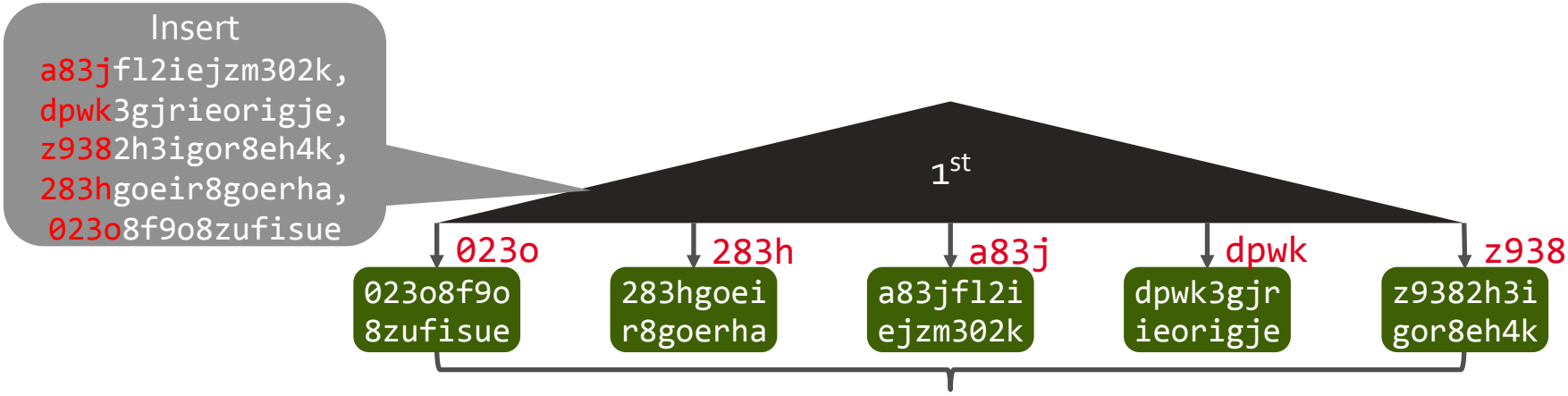
- Trie (prefix tree) whose node is B+Tree
  - A key is split into the list of fixed-size chunks (sub-string of the key)



Benefits

- When keys are sufficiently long & uniform random (e.g., UUID or hash value)
- When keys have common prefixes (e.g., secondary index keys)

Example: Chunk size = 4 bytes



Majority of keys can be indexed by first chunk

➔ There will be **only one B+Tree** on HB+Trie

➔ We don't need to store & compare entire key strings