Tokyo Tyrantの設計と実装

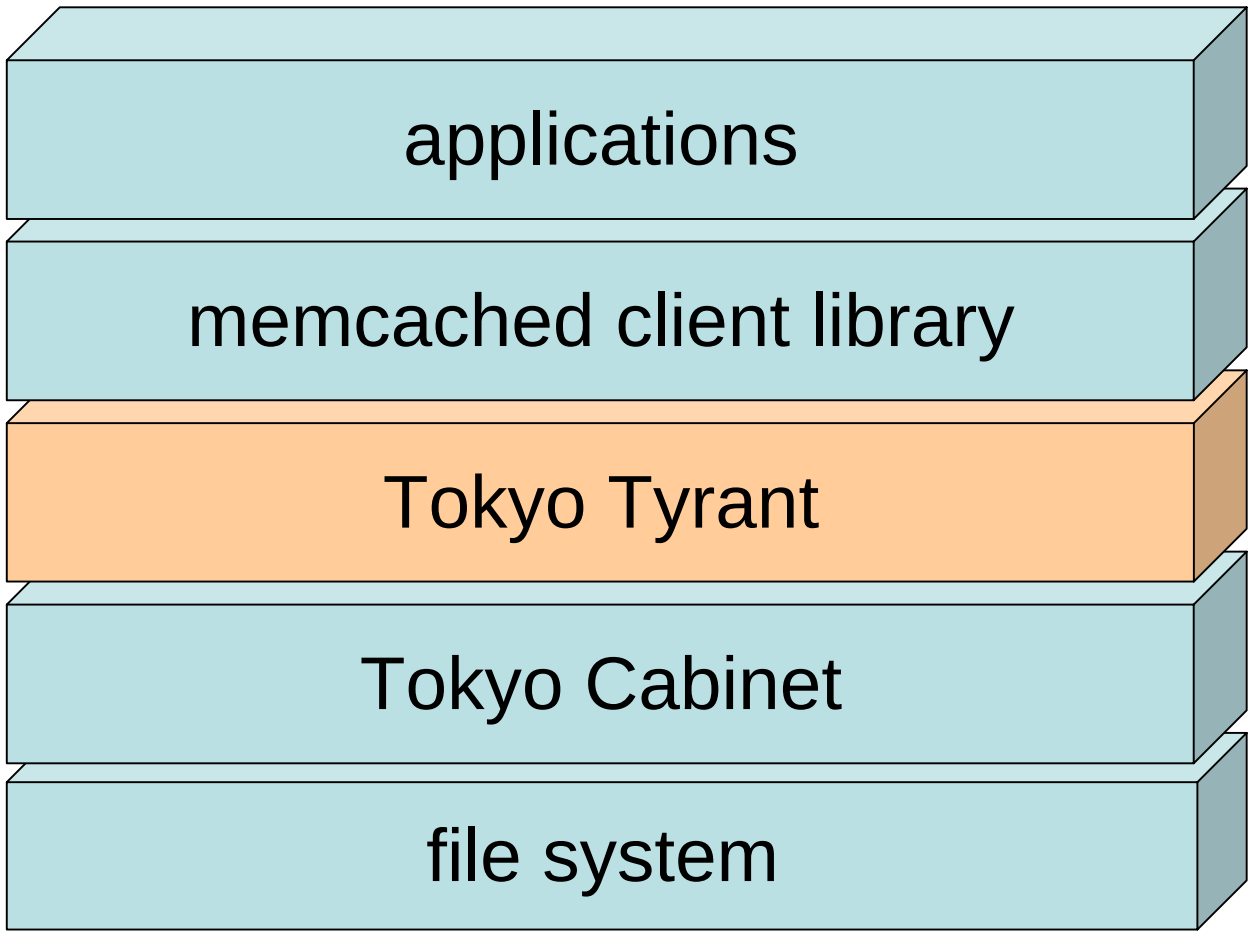
平林 幹雄
<mikio@users.sourceforge.net>


東京
タイラント
c10k

何が嬉しいの？

- memcachedの置き換え
  - memcached互換プロトコル
- データが永続化される
  - ファイル (DBM) にデータを保存
  - レプリケーション機能
- 高パフォーマンス、高スループット
  - 秒間20000クエリ以上
  - 同時接続10000クライアント以上

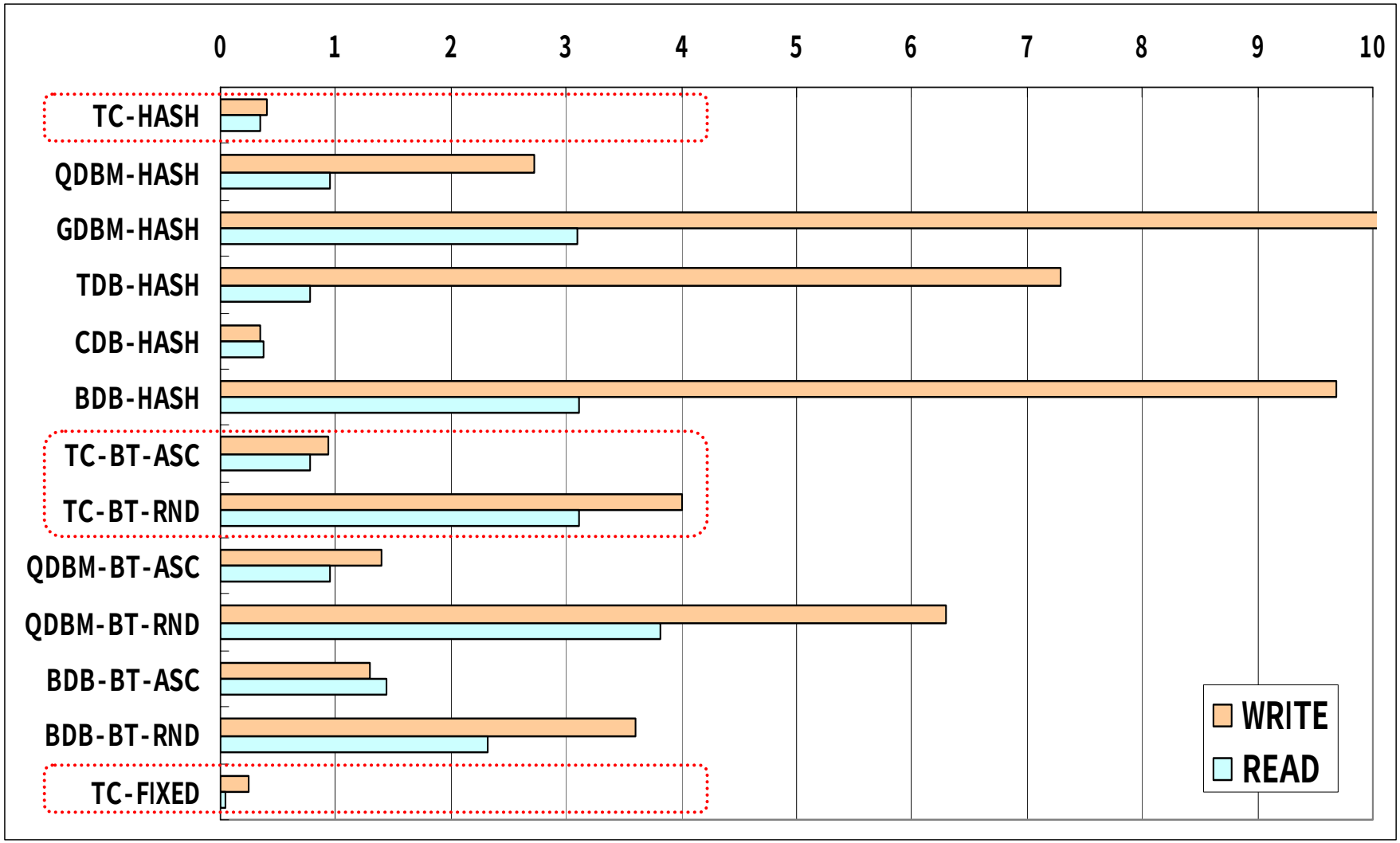
コンポーネント構成


applications
memcached client library
Tokyo Tyrant
Tokyo Cabinet
file system

Tokyo Cabinet

- DBMのモダンな実装
  - key/value型データベース
  - Win32を捨ててPOSIX系に特化
    - QDBMの後継
    - C99、Pthread、mmap、pread/pwrite、etc...
- 高パフォーマンス、高スループット
  - 100万レコードのinsertが0.4秒（2,500,000 qps）
    - searchは0.33秒（3,000,000 qps）
  - レコード単位でリードライトロック
- バインディング各種
  - Perl、Ruby、Java、Lua、Python、PHP、Scheme、etc...

他のDBMとの性能比較



6種類のデータベース型

- TCHDB: ファイル上のハッシュデータベース
- TCBDB: ファイル上のB+木データベース
- TCFDB: ファイル上の配列データベース
- TCMDB: メモリ上のハッシュデータベース
- TCNDB: メモリ上のツリーデータベース
- TCADB: 上記5種の抽象インターフェイス

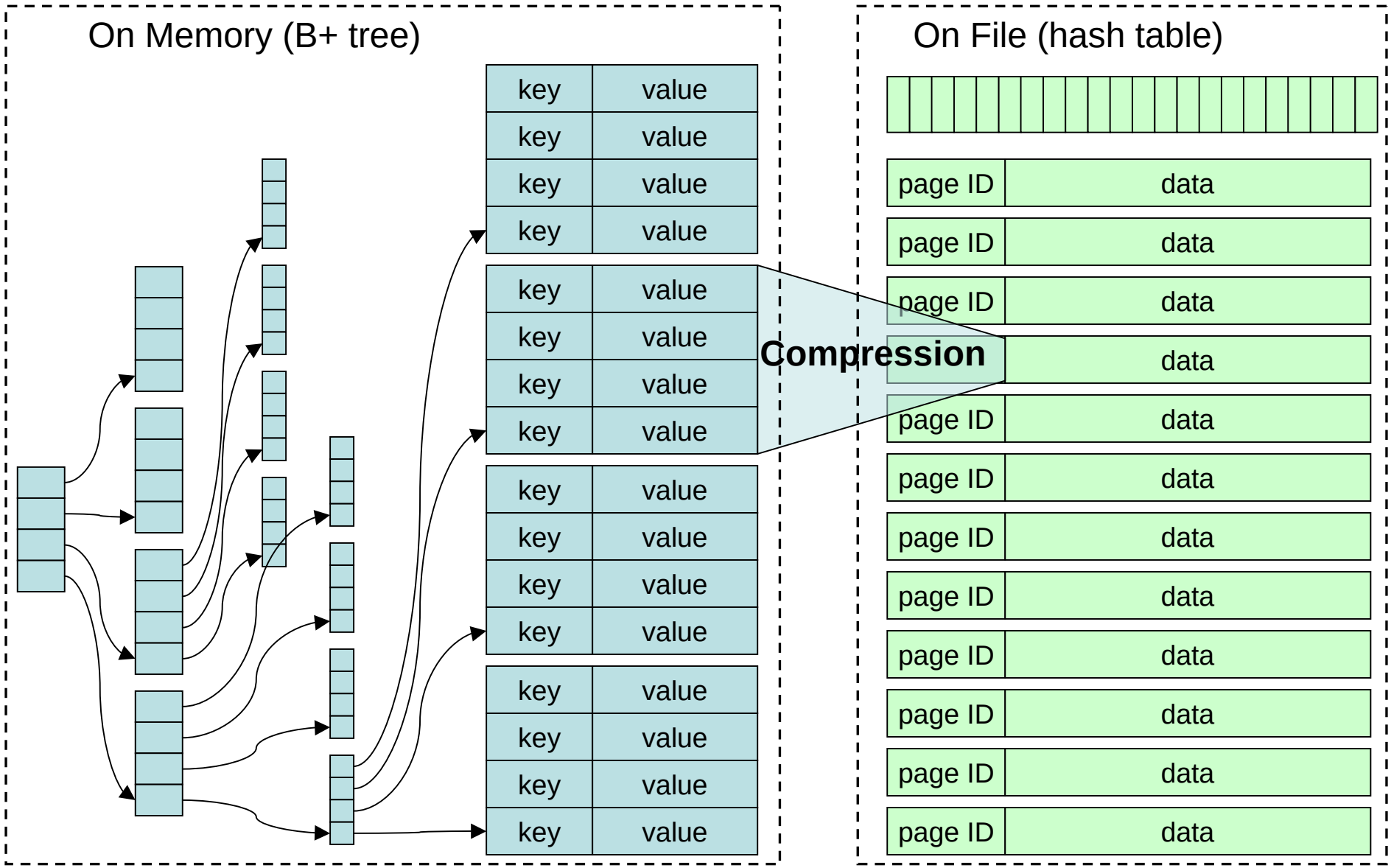
TCHDB: ファイル上のハッシュデータベース

- ハッシュ表と二分探索木
  - 完全一致検索のみ
- 高速
  - O(1)
  - ランダムアクセスに強い
- 省メモリ
  - バケットのみオンメモリ
- mmapとpread/pwriteの併用

TCBDB: ファイル上のB+木データベース

- B+木 (B木とページング)
  - 比較関数による範囲検索
- そこそこ高速
  - O(log N)
  - シーケンシャルアクセスに強い
- メモリ食い
  - LRU消去のページキャッシュ
- TCHDB上でページ管理

圧縮B+木


On Memory (B+ tree)
key
value
Compression
On File (hash table)
page ID
data

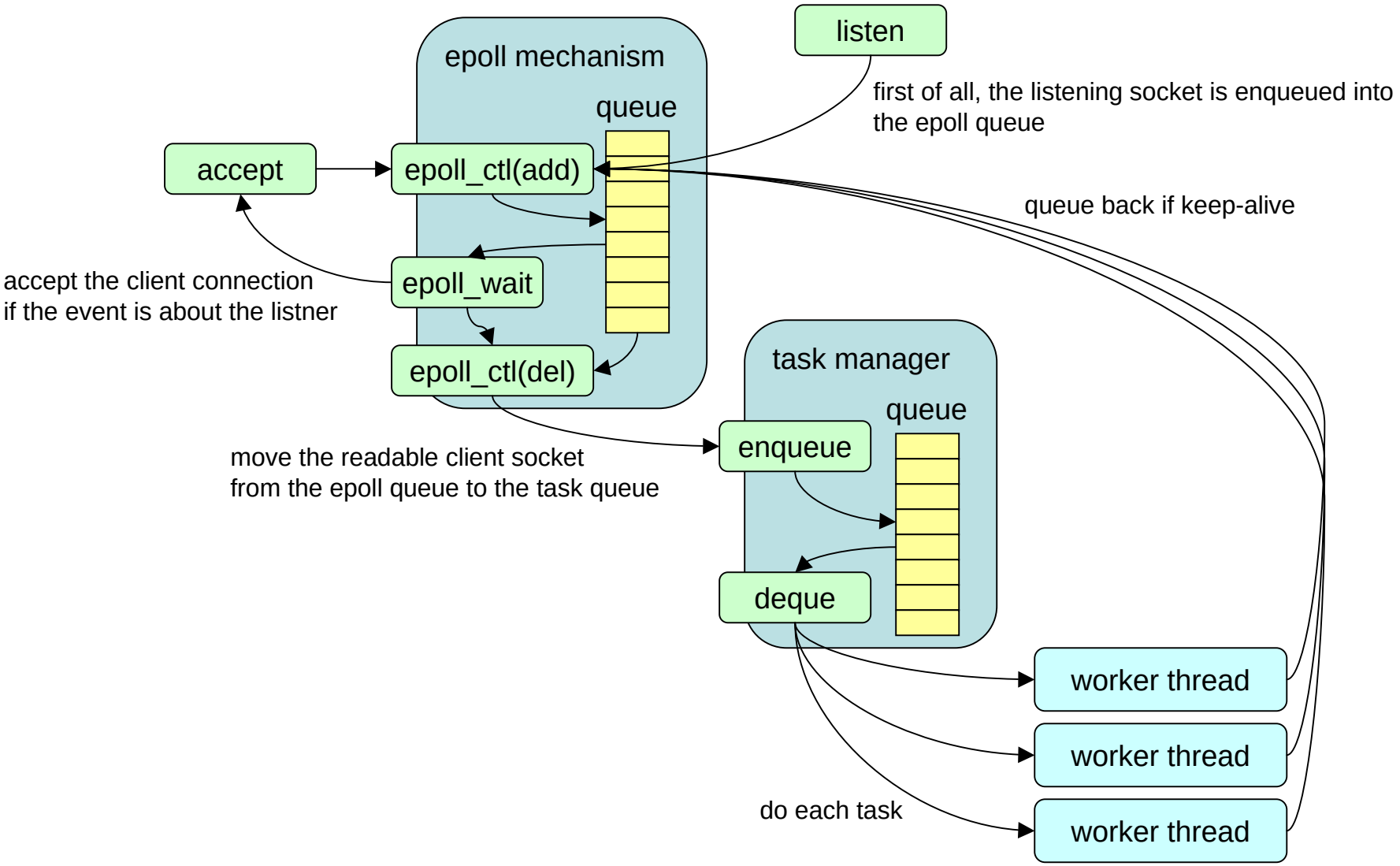
その他

- TCMDB: メモリ上のハッシュデータベース
  - ハッシュ表と二分探索木
  - 格納順序をリンクリストで保存
- TCNDB: メモリ上のツリーデータベース
  - スプレー木
  - 参照したノードを根に移動
- TCFDB: ファイル上の配列データベース
  - キーは自然数のみで、キーの倍数でアドレッシング
- TCADB: 上記5種の抽象インターフェイス
  - DB接続 (open) 時にDBの型を決定
  - Tokyo Tyrantで利用

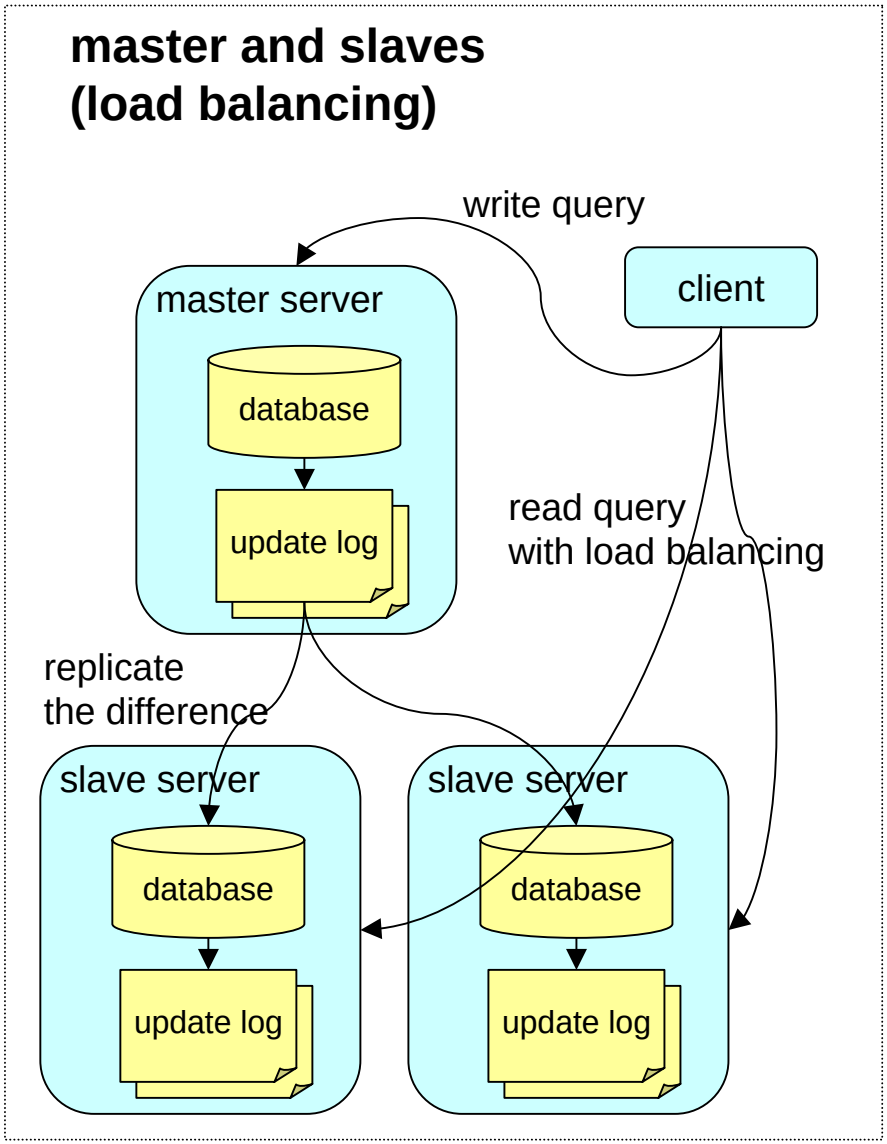
Tokyo Tyrant

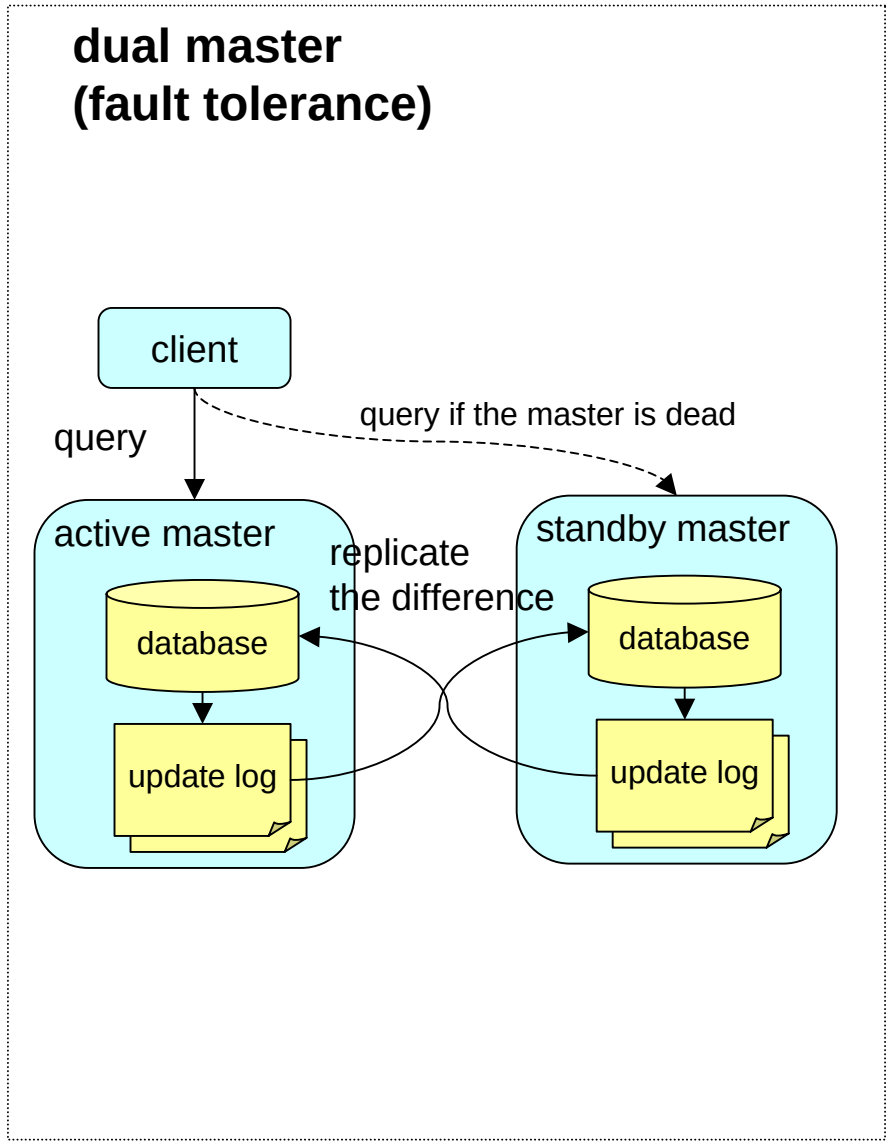
- TCのネットワークインターフェイス
  - 独自バイナリプロトコル、memcached互換、HTTP互換
- epoll/kqueueによるイベント補足
  - 10000クライアント以上の同時接続
- スレッドプール型並列処理
  - ボス (ネットワーク監視) +8個のワーカスレッド (DB操作)
  - 20000qps以上のスループット
- 高可用性
  - ホットバックアップ、更新ログ
  - 非同期レプリケーション

イベント処理


epoll mechanism
queue
listen
first of all, the listening socket is enqueued into
the epoll queue
accept
epoll_ctl(add)
queue back if keep-alive
accept the client connection
if the event is about the listner
epoll_wait
epoll_ctl(del)
task manager
queue
move the readable client socket
from the epoll queue to the task queue
enqueue
deque
worker thread
worker thread
worker thread
do each task

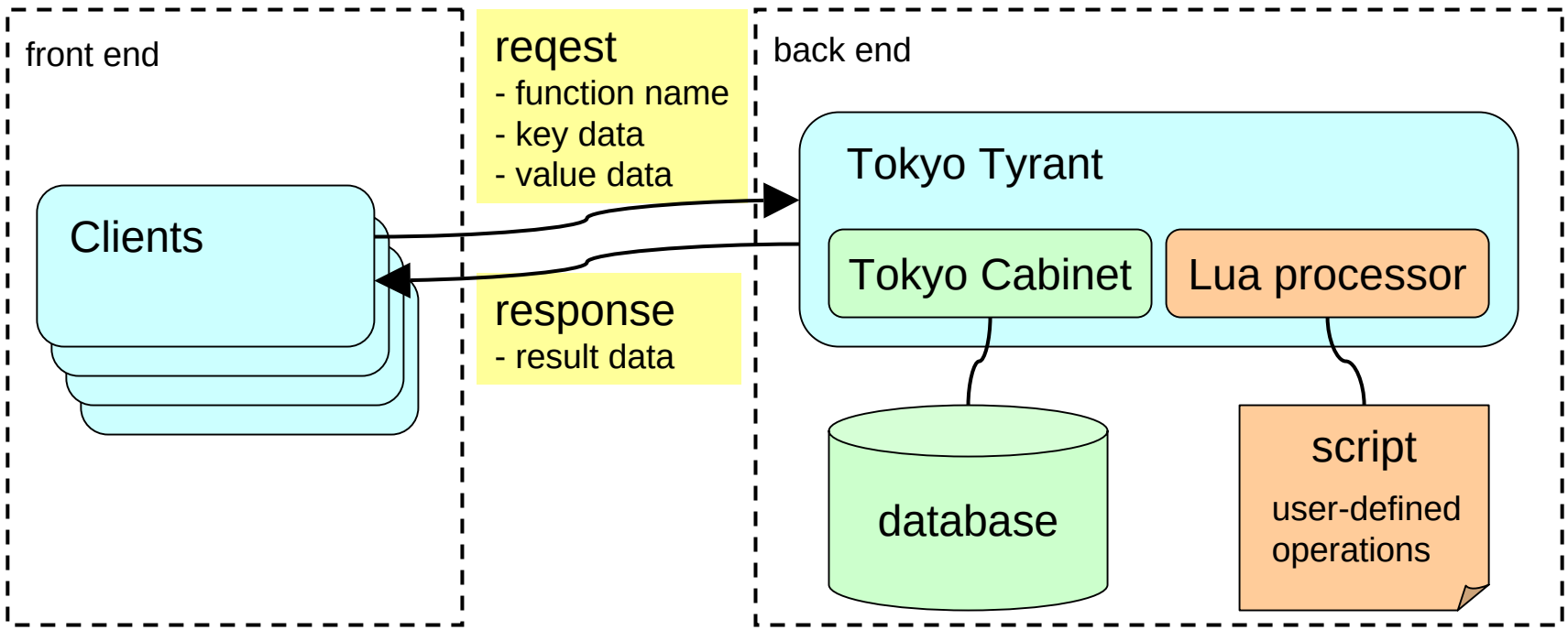
レプリケーション


master and slaves
(load balancing)
write query
client
master server
database
update log
read query
with load balancing
replicate
the difference
slave server
database
update log
slave server
database
update log

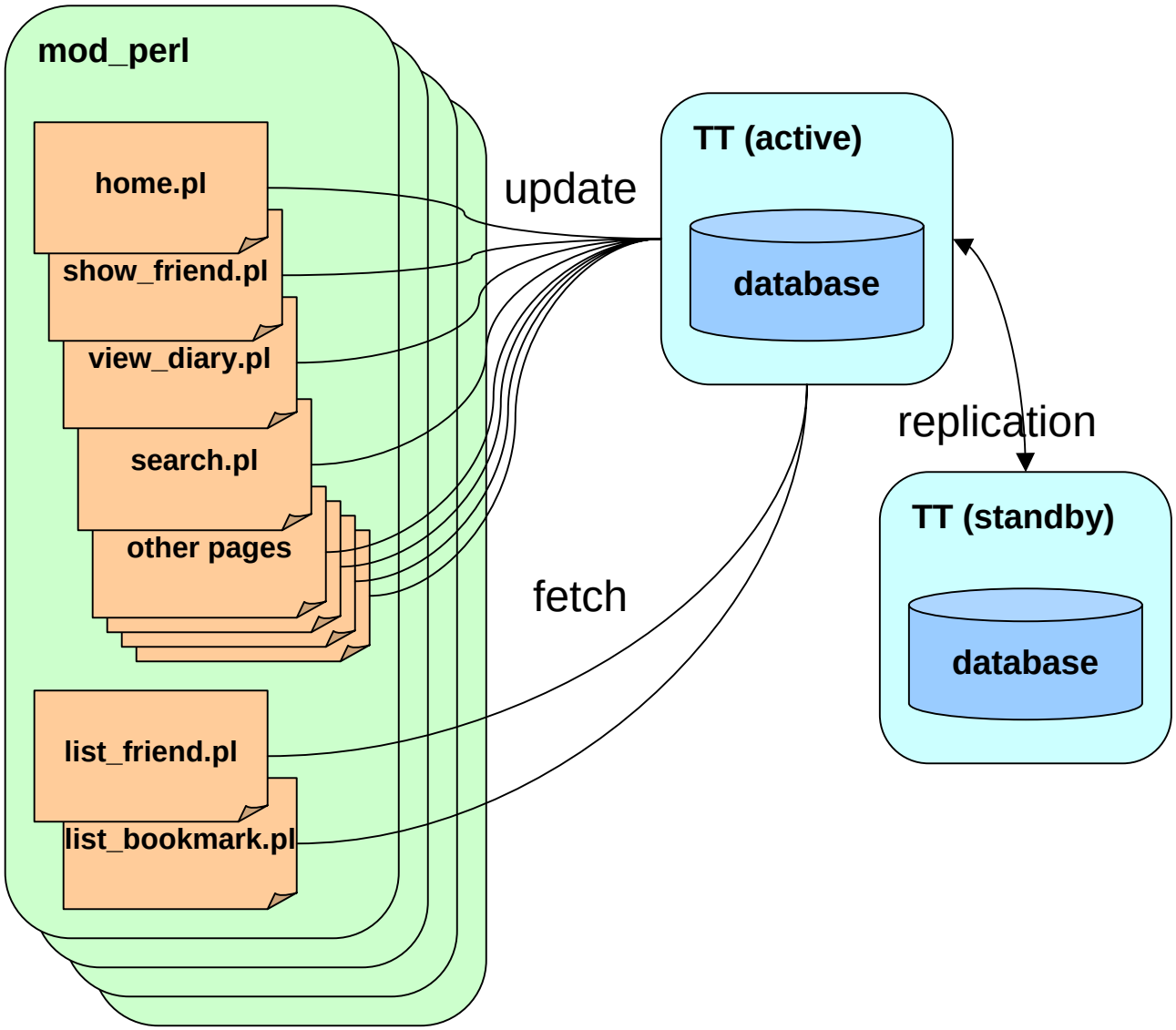

dual master
(fault tolerance)
client
query
query if the master is dead
active master
database
update log
replicate
the difference
standby master
database
update log

Lua拡張

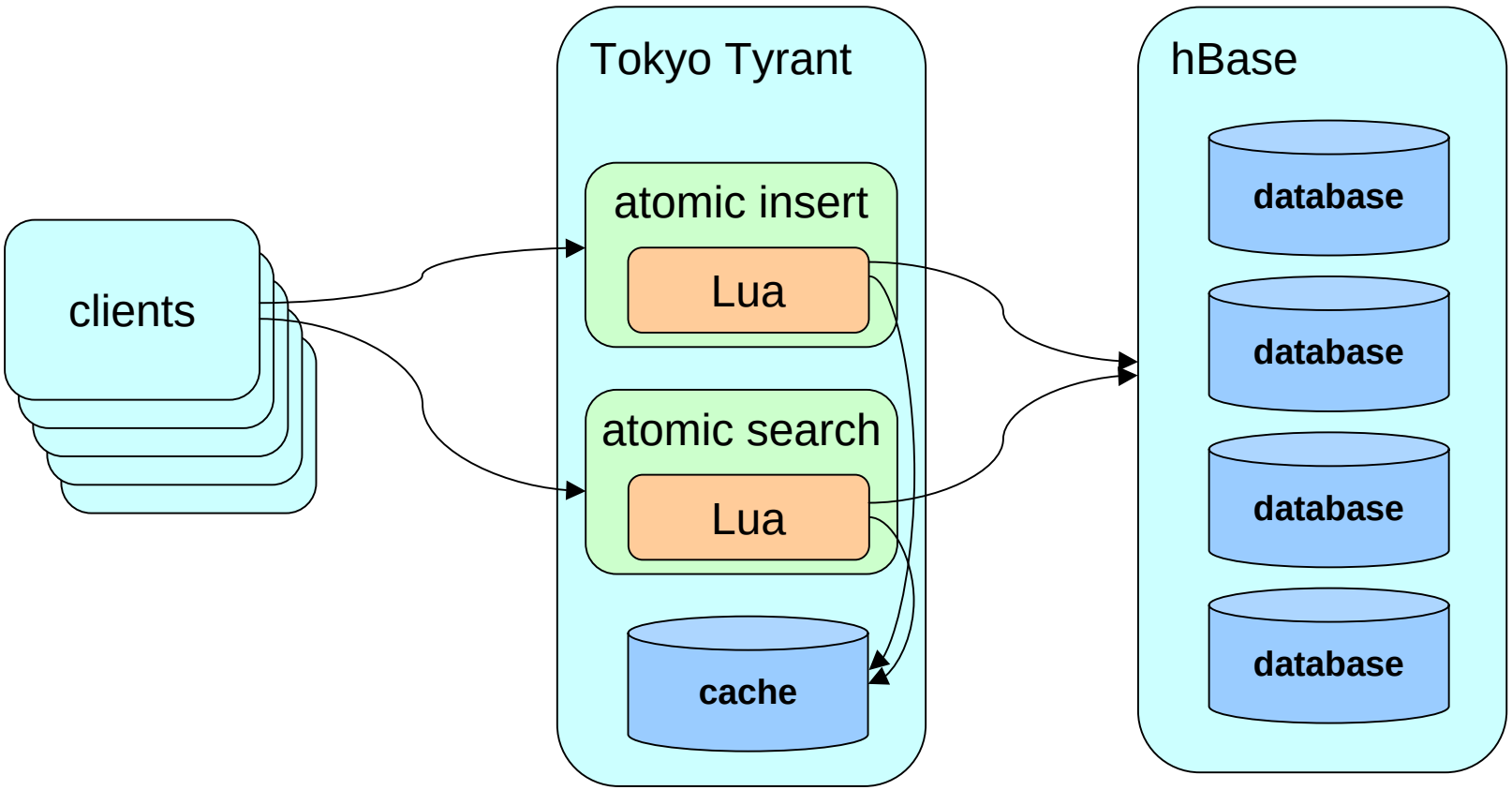
- TTにLuaインタープリタを内蔵
  - Luaの関数として任意のDB操作をサーバ側で記述
  - レコードロックによるアトミックな呼び出し



事例：mixiのタイムスタンプDB



事例：hBaseのキャッシュ



詳しくは...

- Tokyo Cabinetプロジェクトサイト
  - http://tokyocabinet.sourceforge.net/
- mixi engineers' blog
  - http://alpha.mixi.co.jp/blog/
    - Tokyo TyrantによるHAハッシュDBサーバの構築
      - http://alpha.mixi.co.jp/blog/?p=147
    - Tokyo Tyrantによる耐高負荷DBの構築
      - http://alpha.mixi.co.jp/blog/?p=166
    - Lua on Tyrant: DBサーバにLLを組み込む
      - http://alpha.mixi.co.jp/blog/?p=236