# Improve the Bw-tree

Hyeon Cheol, Kim

HYU 한양대학교 HANYANG UNIVERSITY

# Motivation

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
```

After finding the node that
delta node should be attached

# Motivation

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
```

Steps of attaching leaf Insert delta node

1. Allocate delta node
2. Try to attach with CAS
3. If CAS failed, deallocate the delta node

# Motivation

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
```

CAS fail will result useless operations

- Alloc delta node
- Dealloc delta node
- (CAS itself)

# Motivation

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
```

CAS fail will result useless operations
- Alloc delta node
- Dealloc delta node
- (CAS itself)

Why should we alloc delta node before CAS
even for the failed thread?
-> don't know the following CAS will succeed or not.

# Motivation

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
```

CAS fail will result useless operations
- Alloc delta node
- Dealloc delta node
- (CAS itself)

Why should we alloc delta node before CAS
even for the failed thread?
-> don't know the following CAS will succeed or not.

If we know CAS will fail without actually doing CAS,
Useless operations can be skipped

HYU 한양대학교 HANYANG UNIVERSITY

# Design

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    #if defined(MY_OPT)
    if (GetNode(node_id) != node_p) goto skip;
    #endif // MY_OPT
    {

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
    #if defined(MY_OPT)
    skip:
    (void)0;
    #endif // MY_OPT
```

Like the TTAS lock try TAS only when it is likely to succeed by checking it's status,

# Design

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    #if defined(MY_OPT)
    if (GetNode(node_id) != node_p) goto skip;
    #endif // MY_OPT
    {

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
    #if defined(MY_OPT)
    skip:
    (void)0;
    #endif // MY_OPT
```

Like the TTAS lock try TAS only when it is likely to succeed by checking it's status,

try CAS only when the node_p of mapping_table is unchanged with last snapshot's node_p

# Design

```
    NodeSnapshot *snapshot_p = GetLatestNodeSnapshot(&context);

    // We will CAS on top of this
    const BaseNode *node_p = snapshot_p->node_p;
    NodeID node_id = snapshot_p->node_id;

    #if defined(MY_OPT)
    if (GetNode(node_id) != node_p) goto skip;
    #endif // MY_OPT
    {

    const LeafInsertNode *insert_node_p =
        LeafInlineAllocateOfType(LeafInsertNode, node_p, key, value, node_p, index_pair);

    bool ret = InstallNodeToReplace(node_id, insert_node_p, node_p);
    if (ret) {
      INDEX_LOG_TRACE("Leaf Insert delta CAS succeed");

      // If install is a success then just break from the loop
      // and return
      break;
    }

    INDEX_LOG_TRACE("Leaf insert delta CAS failed");

#ifdef BWTREE_DEBUG

    context.abort_counter++;

#endif

    insert_node_p->~LeafInsertNode();
    #if defined(MY_OPT)
    skip:
    (void)0;
    #endif // MY_OPT
```

Like the TTAS lock try TAS only when it is likely to succeed by checking it's status,

try CAS only when the node_p of mapping_table is unchanged with last snapshot's node_p

If changed, can safely assume following CAS will fail

Skip the useless operations

# Design

Additional overhead by checking status?

# Design

Additional overhead by checking status?

vanila

| Success |
| --- |
| Alloc |
| CAS |

| Fail |
| --- |
| Alloc |
| CAS |
| Dealloc |

optimized

| Success |
| --- |
| atomic::load() |
| Alloc |
| CAS |

| Fail |
| --- |
| atomic::load() |

# Design

Additional overhead by checking status?

vanila

| Success |
| --- |
| Alloc |
| CAS |

| Fail |
| --- |
| Alloc |
| CAS |
| Dealloc |

optimized

| Success |
| --- |
| atomic::load() |
| Alloc |
| CAS |

| Fail |
| --- |
| atomic::load() |

Difference = load * (#succ + #fail) - (alloc + CAS + dealloc) * #fail

= load * #succ - (alloc + dealloc) * #fail (Assume load , CAS has similar latency)

# Design

Additional overhead by checking status?

vanila

| Success |
| --- |
| Alloc |
| CAS |

| Fail |
| --- |
| Alloc |
| CAS |
| Dealloc |

optimized

| Success |
| --- |
| atomic::load() |
| Alloc |
| CAS |

| Fail |
| --- |
| atomic::load() |

Difference = load * (#succ + #fail) - (alloc + CAS + dealloc) * #fail

= load * #succ - (alloc + dealloc) * #fail

(Assume load , CAS has similar latency)

More fail -> improvement

More success -> slower

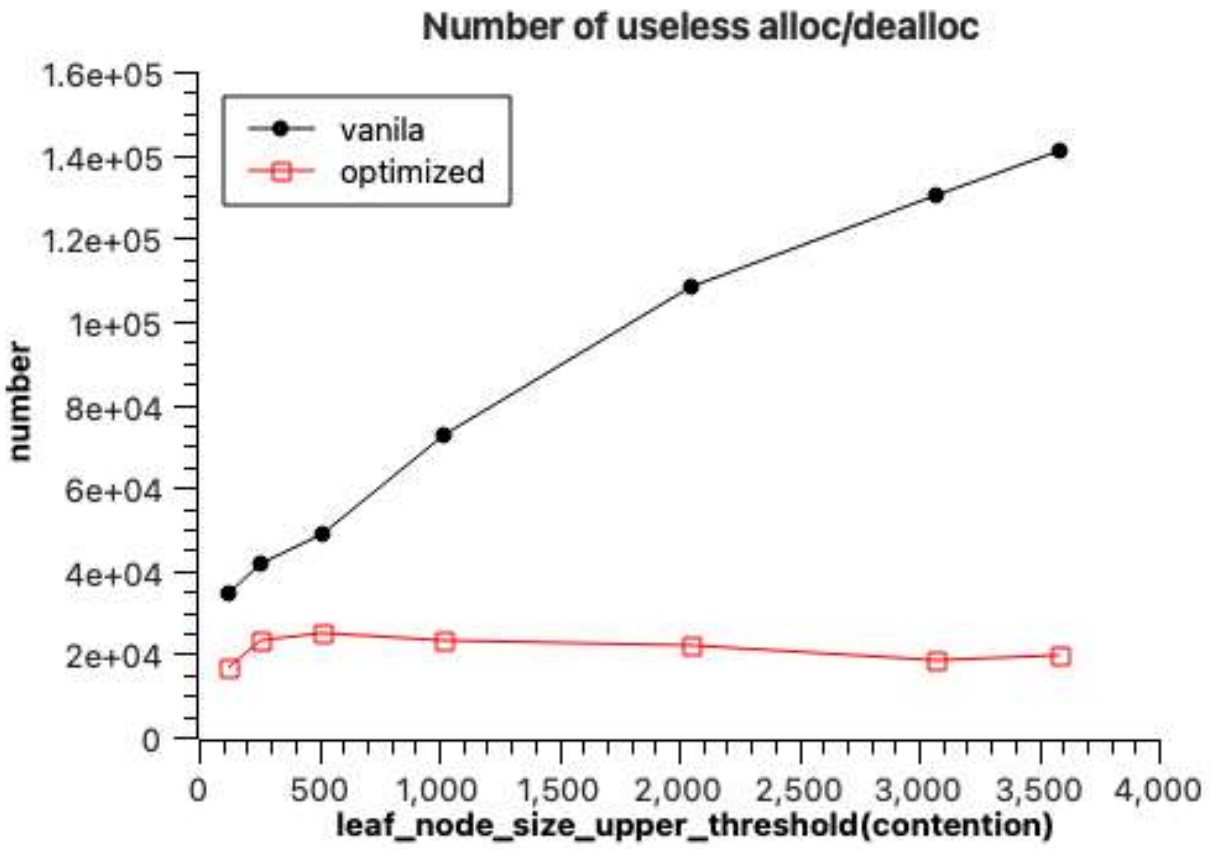
HYU 한양대학교 HANYANG UNIVERSITY

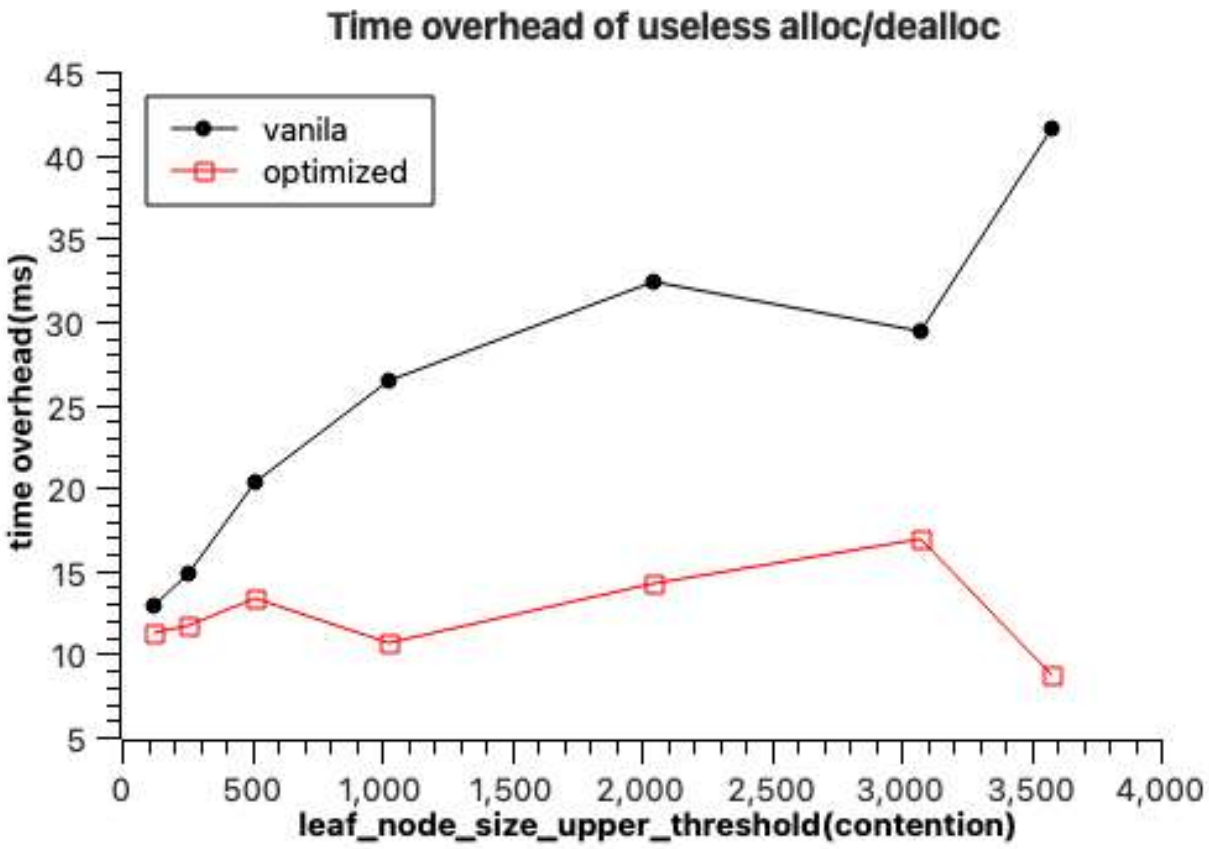
# Evaluation

Evaluation setting
- Processor : 12th Gen Intel(R) Core(TM) i7-12700
- Configuration : 20 Cores
- Hyperthreading : disabled
- Main Memory : 32GB
- Operating System : Ubuntu 20.04.5 LTS

Evaluation protocol
- Each thread try to insert random (key, val) until 4M key value pair inserted
- Measured number and time of useless operations (when CAS failed) as increasing the leaf_node_size_upper_threshold. (which means contention on CAS of single leaf node, probability of failure of CAS)
- Repeat whole procedure 10 times to get average

# Evaluation

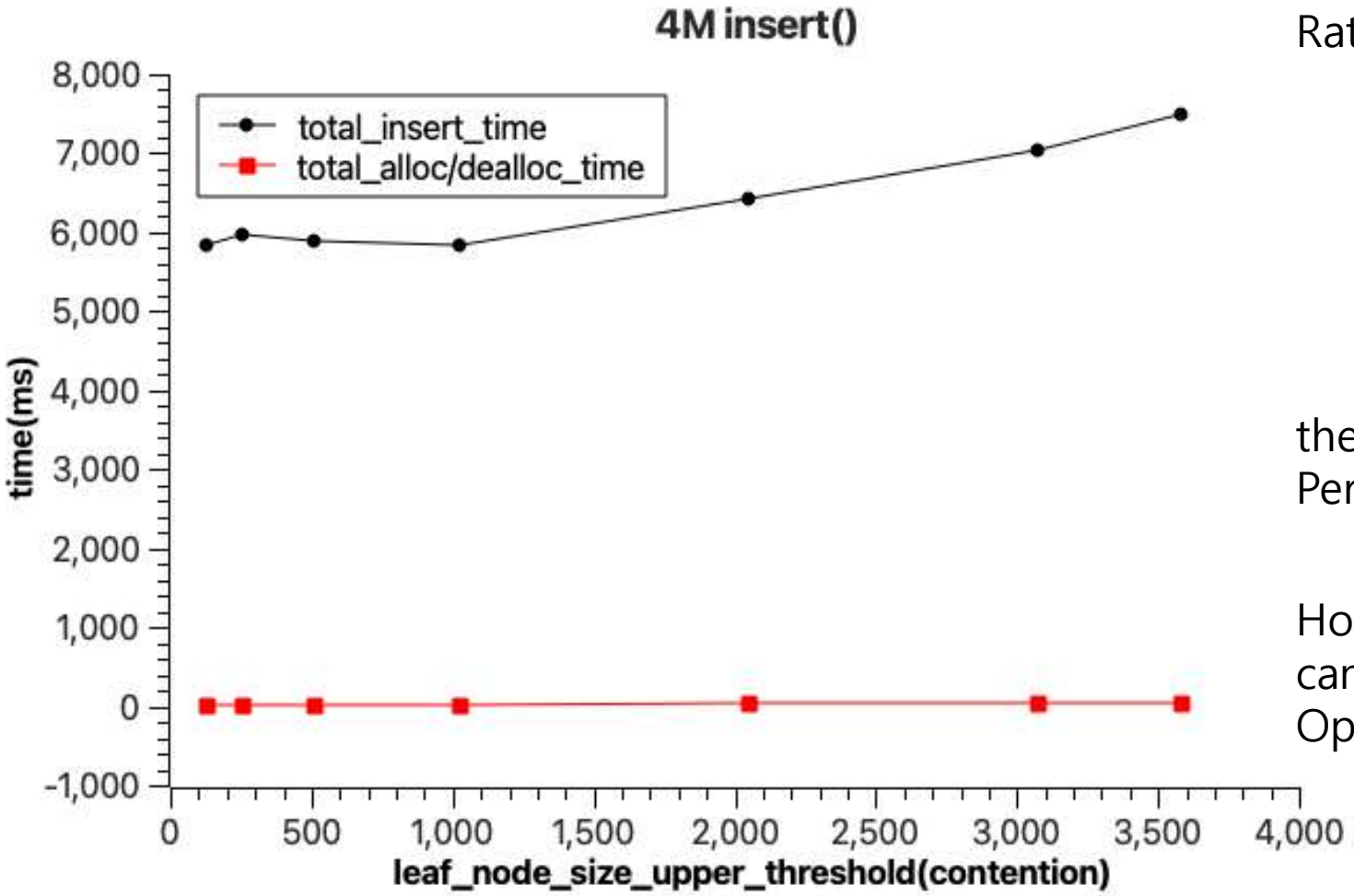



The optimization result seems to be fine

HYU 한양대학교 HANYANG UNIVERSITY

# Evaluation




Ratio of alloc, dealloc time

the ratio of optimization target is extremely low,
Performance improvement is almost zero

However, as the alloc-CAS-dealloc execution flow can be seen in many place(delete, consolidate..),
Optimizing entire can show more improvements..

HYU 한양대학교 HANYANG UNIVERSITY

# Thank you