CScript: Supporting Consistent and Available Replication

(Deliverables: D2.1.2, D2.2.1)

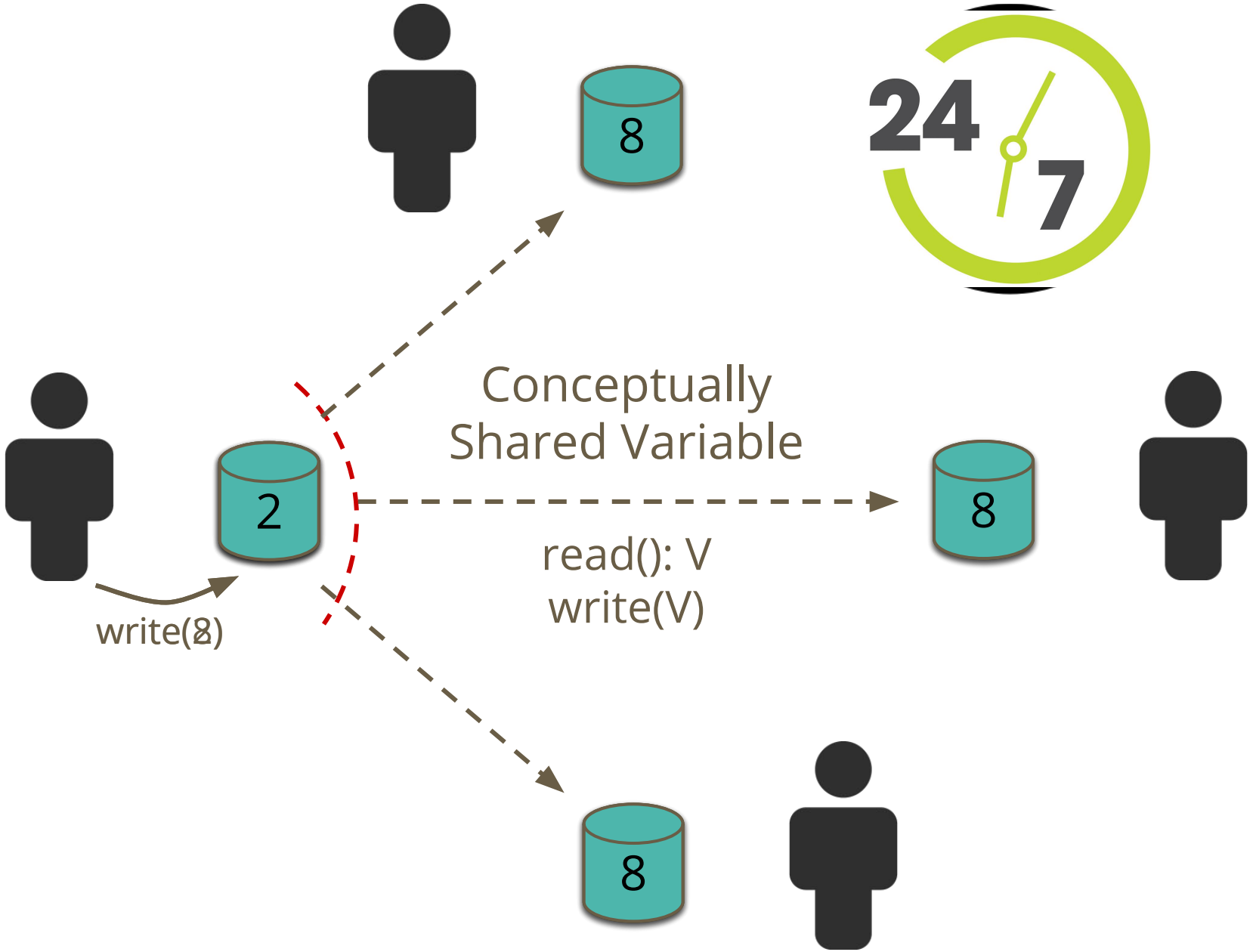
Kevin De Porre
kdeporre@vub.be

VRIJE UNIVERSITEIT BRUSSEL

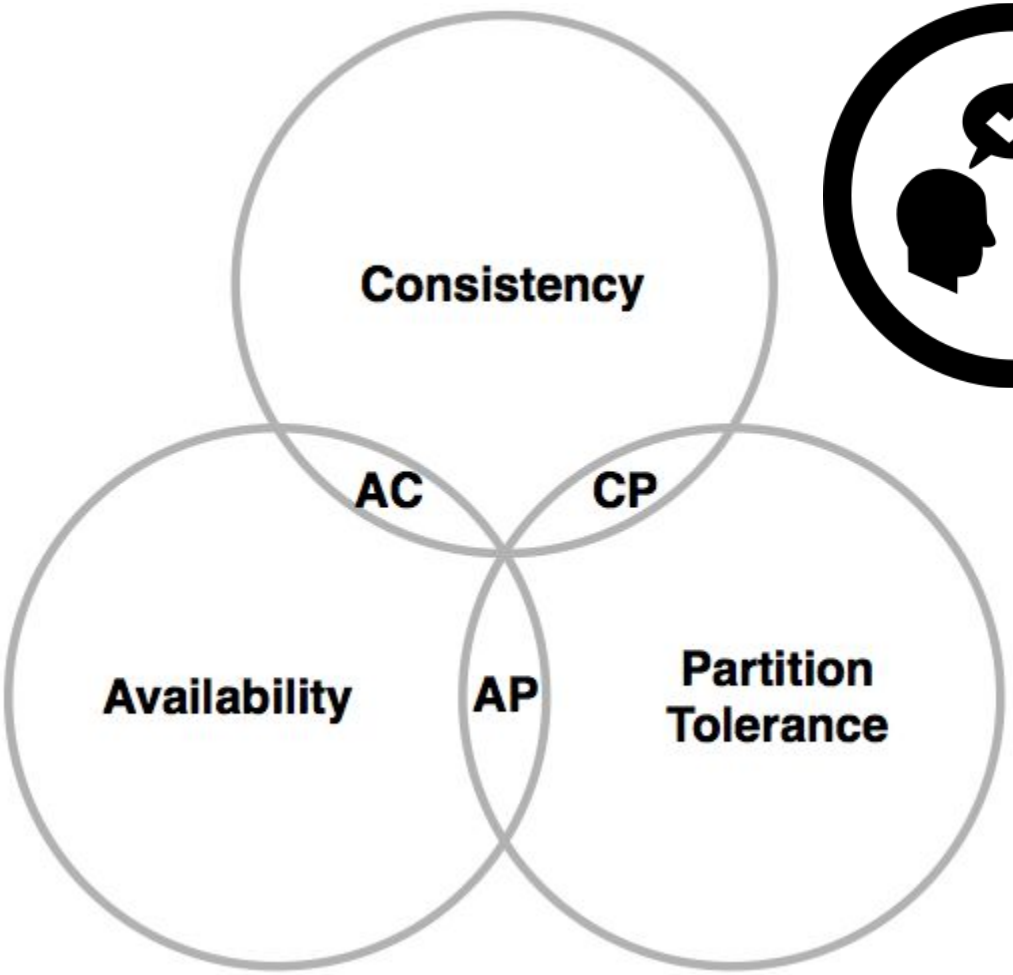
KU LEUVEN


Microsoft
Microsoft
Windows 95


20 35

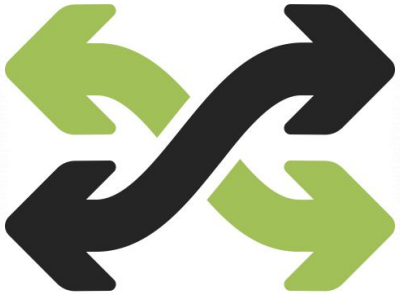

8
24
7
Conceptually
Shared Variable
2
8
read(): V
write(V)
write(8)
8

CAP Theorem

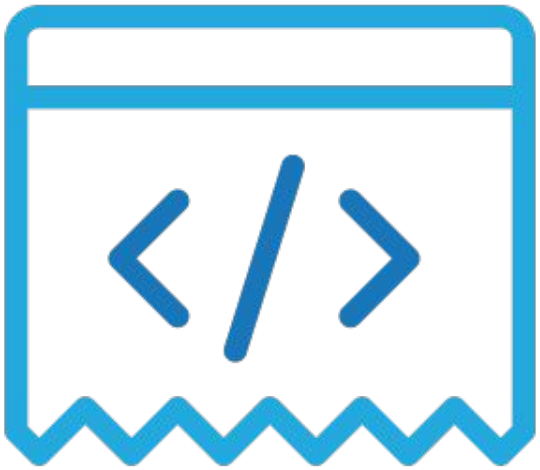


Brewer, E. A. (2000). Towards Robust Distributed Systems. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing* (p. 7). New York, NY, USA. doi: 10.1145/343477.343502

WP 2: End-to-end Eventual Consistency



Problem Statement



Lack of language support

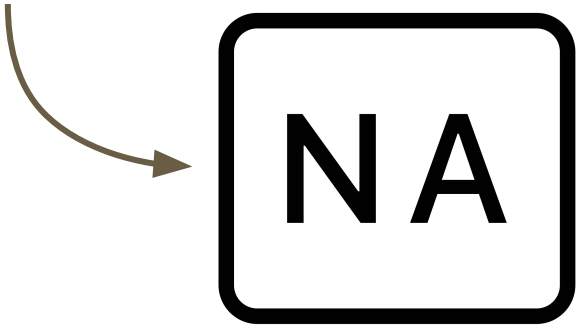
Language	AP	CP
Argus	✗	Actions
Distributed Smalltalk	Copy	RMI
Emerald	Move	RMI
AmbientTalk	Isolates	Far References
Geo	Caching	Linearizability & Cache Coherence Protocol
Lasp	CRDTs	✗

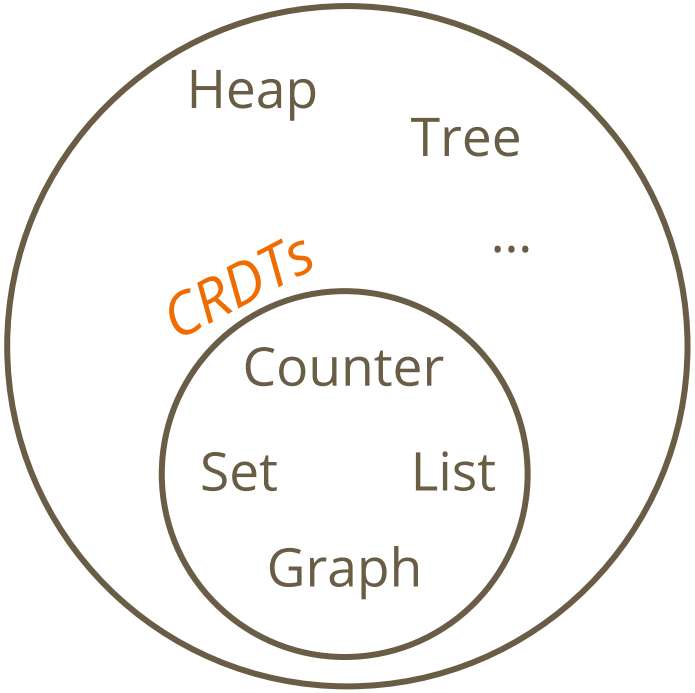
Conflict-free Replicated Data Types (CRDTs)

- Commutativity design
  - Avoids conflicts
  - But.. strong restriction

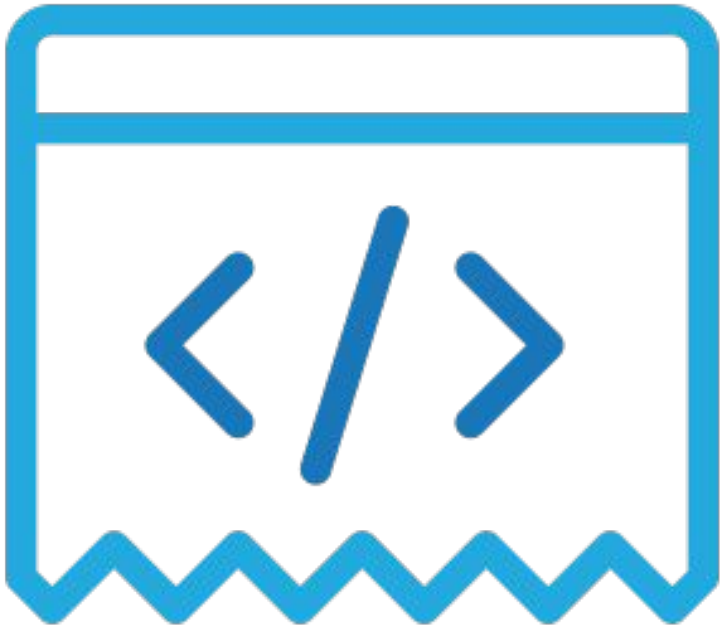
✗ Limited set of CRDTs

✗ Not generally applicable





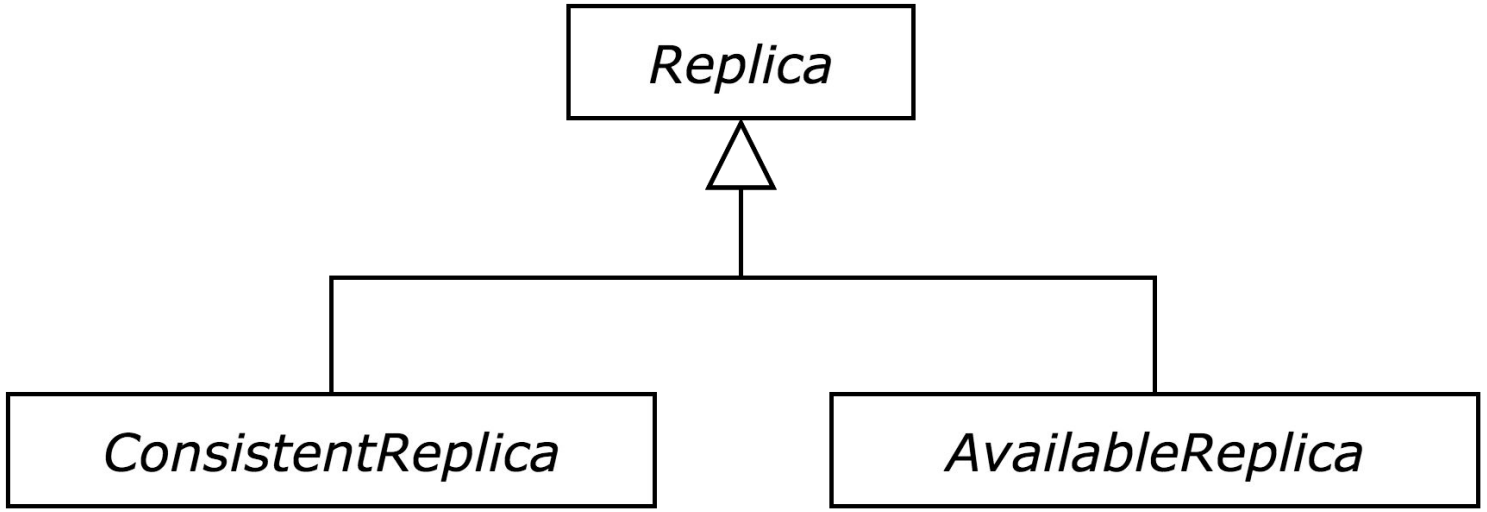
Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict- free Replicated Data Types. In X. Défago, F. Petit, & V. Villain (Eds.), *Symposium on Self-Stabilizing Systems* (pp. 386–400). Grenoble, France.



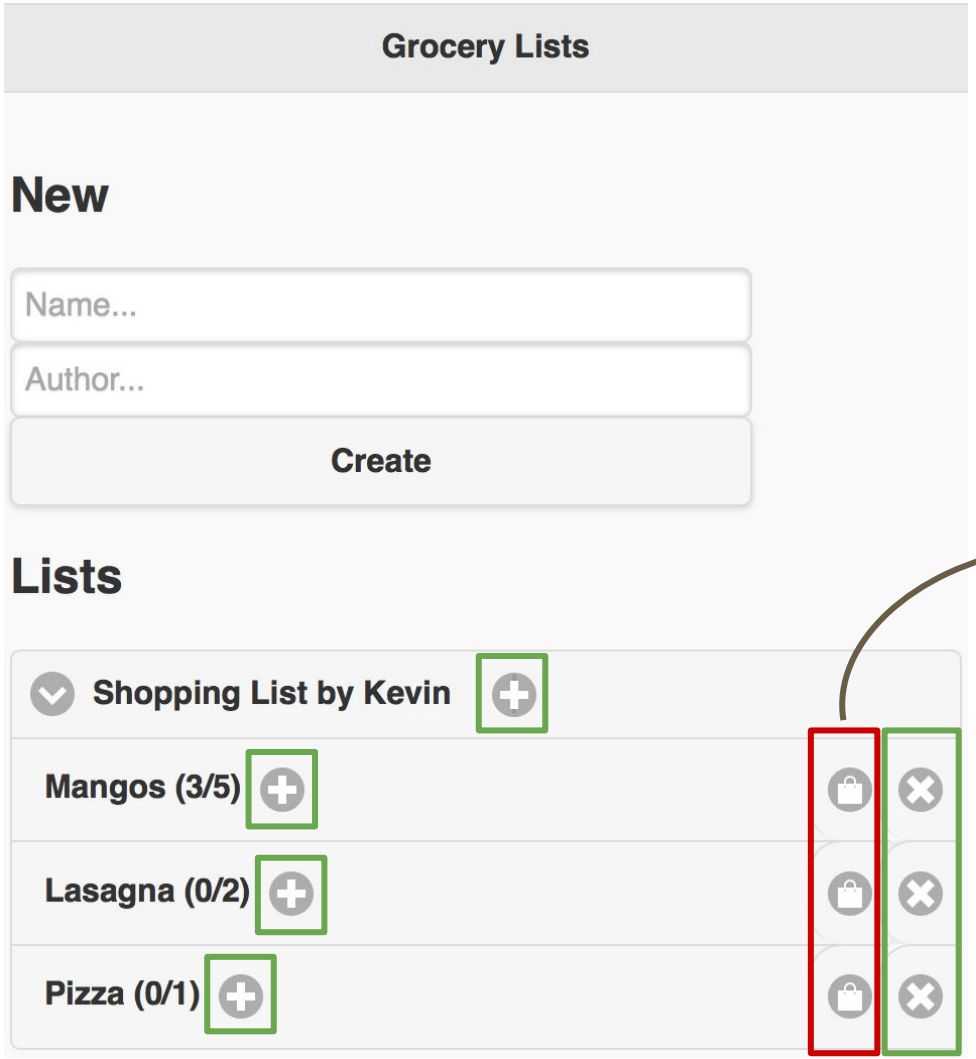
Lack of language support

CScript

- Extends JS
- Replicas
  - Consistent
  - Available



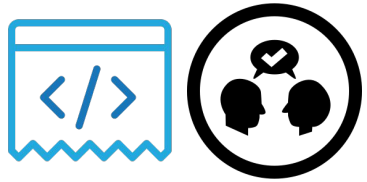
Use Case: Grocery App



Grocery App

```
service GroceryService {
    rep list = new GroceryList();
    rep inventory = new Inventory();

    constructor(name, author)
        this.name = name;
        this.author = author;
    }

    add(item, qty) {
        return this.list.add(item, qty);
    }

    delete (itemName) {
        return this.list.delete(itemName);
    }

    buy(itemName, qty) { /* ... */ }
}
```

Grocery App: Inventory

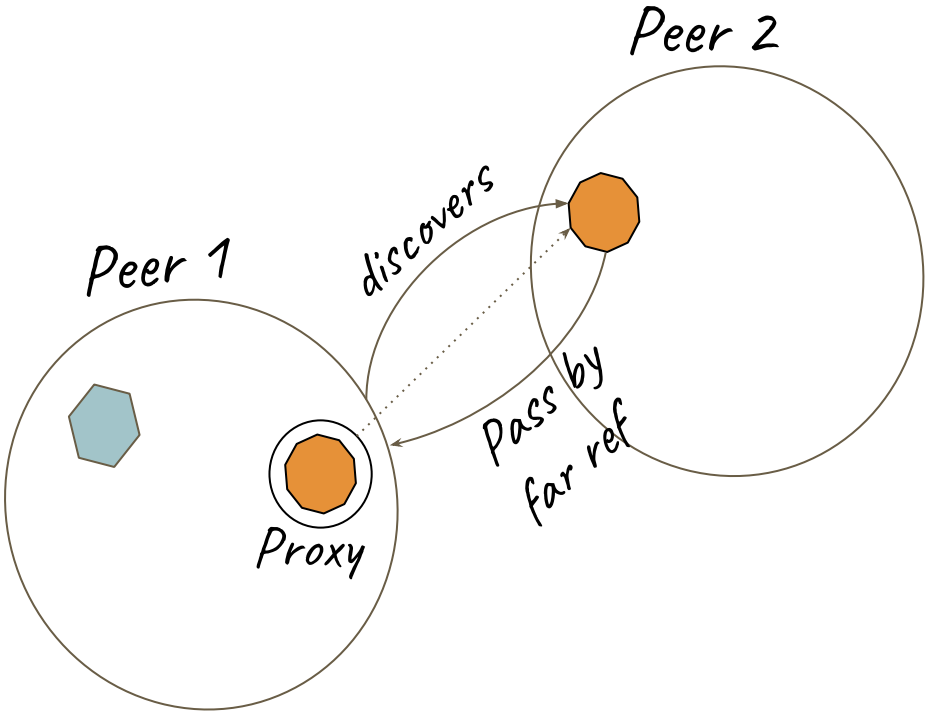


```
class Inventory {
    constructor() {
        this.stock = new Map();
    }

    approve(itemName, stockQuantity, buyingQuantity) {
        if (buyingQuantity <= 0)
            return false;

        const trueStock = this.stock.getOrElse(itemName, 0);
        if (trueStock === stockQuantity) {
            this.stock.set(itemName, trueStock + buyingQuantity);
            return true;
        }
        else {
            return false;
        }
    }
}
```

Consistent Replicas

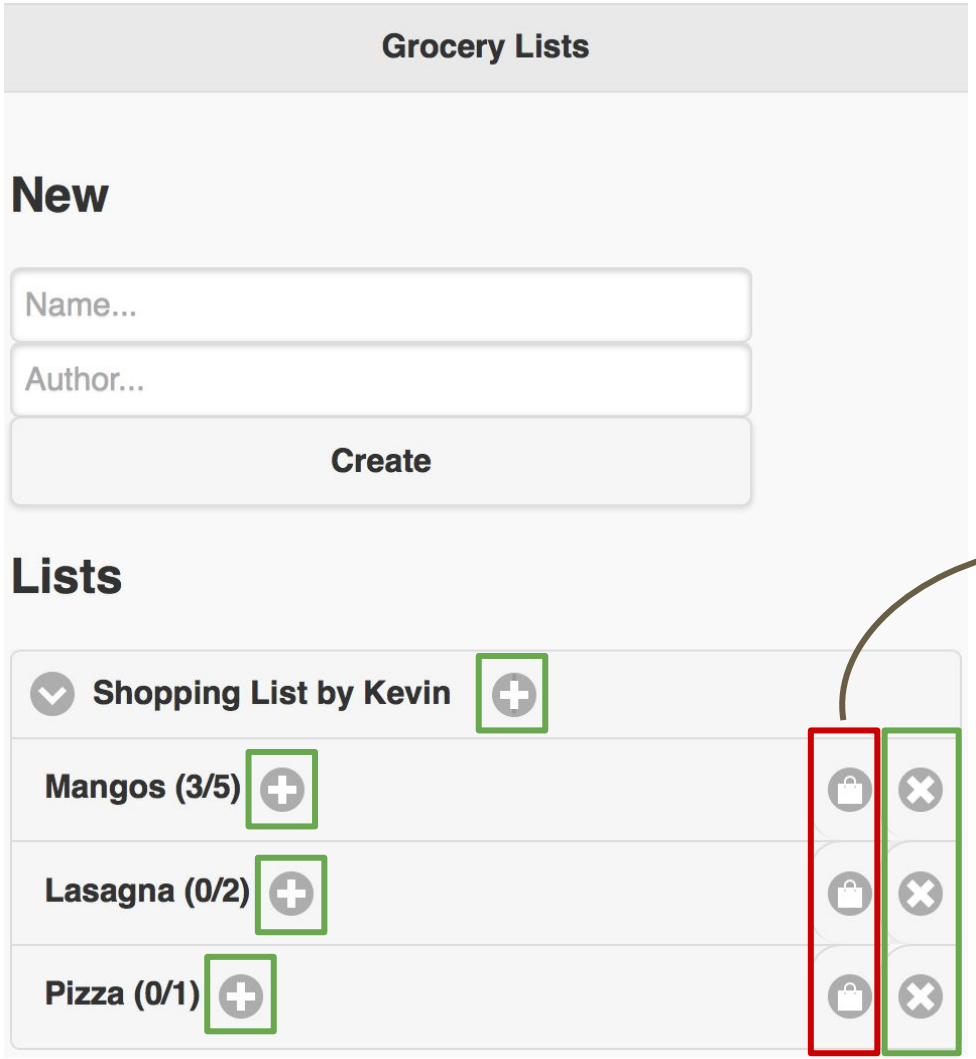
- Pass by far reference
  - Single copy
  - Serializes operations

✓ Strong consistency



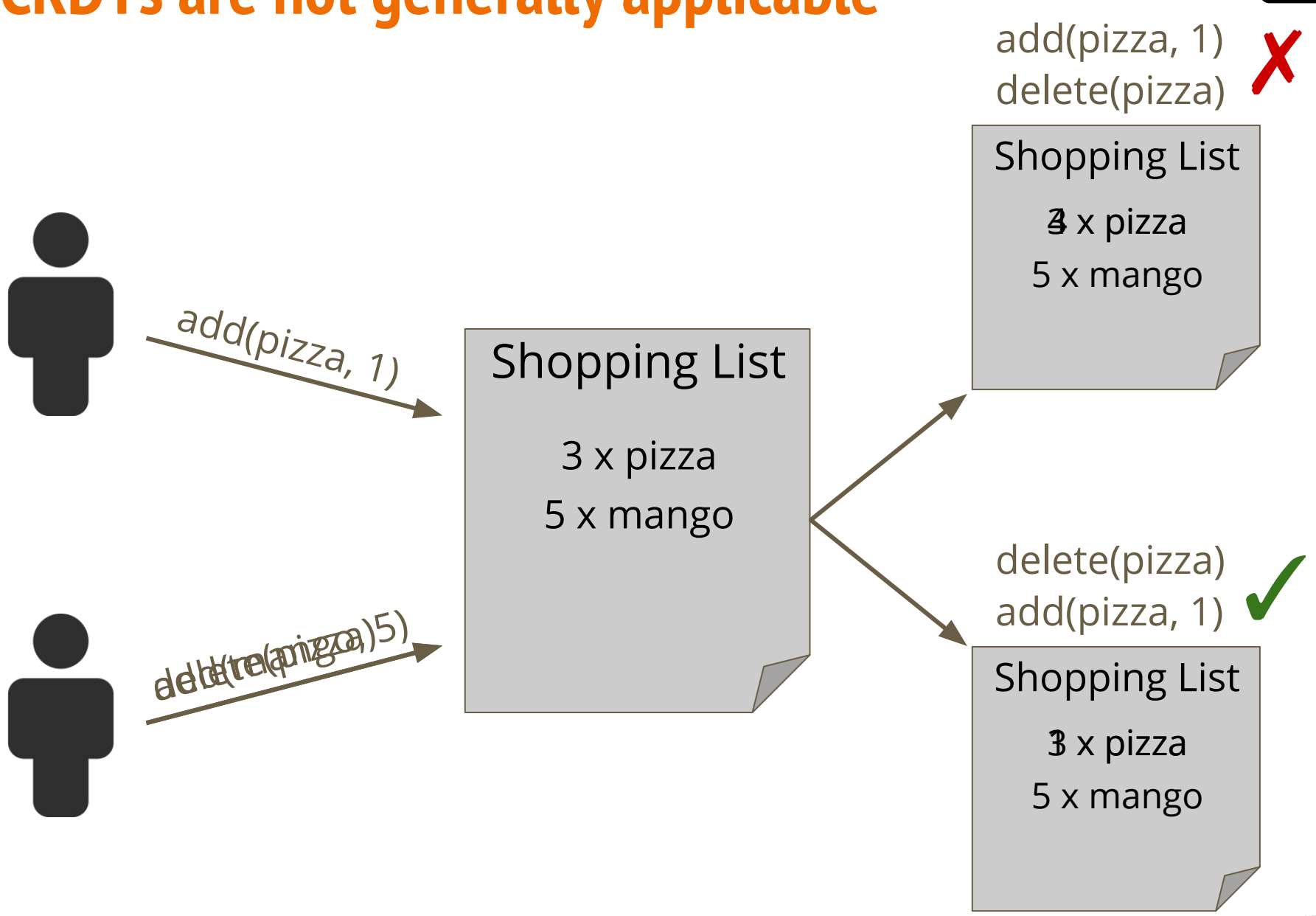
Overview


24
7


Grocery Lists
New
Name...
Author...
Create
Lists
Shopping List by Kevin
Mangos (3/5)
Lasagna (0/2)
Pizza (0/1)

CRDTs are not generally applicable

NA



NA

Strong Eventually Consistent Replicated Object

- General-purpose replicated data type
  - ✓ Availability
  - ✓ SEC
- No restrictions on operations
- Specify concurrent behavior
  - State validators

Grocery App: List

NA


```
class GroceryList extends SECRO {
    constructor() {
        super();
        this.items = new Map();
    }

    add(item, qty) {
        const description =
          this.items.getOrElse(item.name, {requested: 0, bought: 0});
        description.requested += qty;
        this.items.set(item.name, description);
    }

    post add(oState, state, args, res) {
        const [item]    = args,
              addedQty = item.requested,
              resQty    = state.items.getOrElse(item.name, 0).requested;
        return resQty >= addedQty;
    }

    delete(itemName) {
        this.items.delete(itemName);
    }
}
```

NA

SECRO: Algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```

add(pizza, 1)

delete(pizza)

Shopping List

4 x pizza

5 x mango

add(pizza, 1)

delete(pizza)

Shopping List

3 x pizza

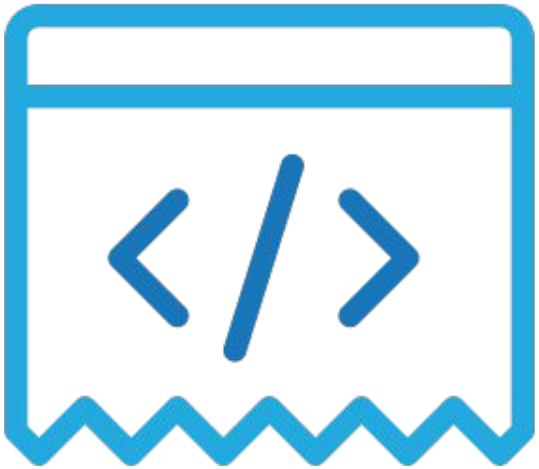
5 x mango

[add(pizza, 1), delete(pizza)]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

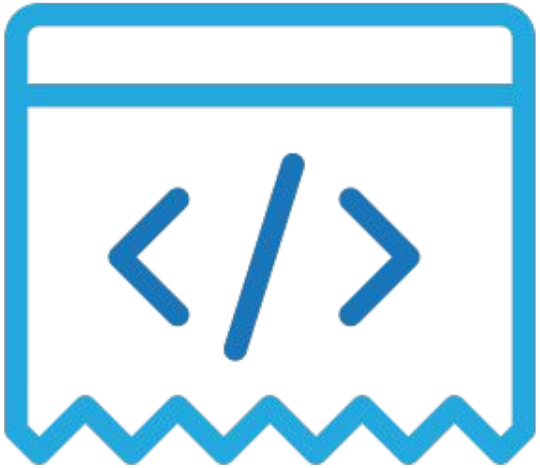
[add(pizza, 1), delete(pizza)]



Lack of language support

Limited applicability
of available solutions

Evaluation (I)



- Collaborative Text Editor
  - SECROs in CScript
  - JSON CRDT

Kleppmann, M., & Beresford, A. R. (2017). A Conflict-Free Replicated JSON Datatype. *IEEE Transactions on Parallel and Distributed Systems*, *28*(10), 2733–2746.

Code Comparison

- Similar implementations
  - API delegates to list
- SECROs: pre/post conditions
  - More LOC
  - Higher flexibility
- Tree-based text editor
  - ✓ SECROs: re-use AVL tree
  - ✗ JSON CRDT: not possible

```
insertAfter(charID, char) {
    const prev = charID === null ?
                 0 : this.indexOf(charID),
          newChar = new Character(char, generate_id());
    this._content.idx(prev).insertAfter(newChar);
    return newChar;
}

pre insertAfter(state, charID, char) {
    return charID === null ||
           state.indexOf(charID) !== -1;
}

post insertAfter(oState, state, args, newChar) {
    /*... */
}

delete(charID) {
    var idx = this.indexOf(charID);
    if (idx != -1)
        this._content.idx(idx).delete();
}

post delete(originalState, state, args, res) {
    /* ... */
}
```

```
insertAfter(charID, char) {
    const idx = charID === null ?
                0 : this.indexOf(charID),
          newChar = new Character(char, generate_id());
    if (idx !== -1) {
        cjrdt.doc.idx(idx).insertAfter(newChar);
        return newChar;
    }
}

delete(charID) {
    const idx = this.indexOf(charID);
    if (idx != -1)
        cjrdt.doc.idx(idx).delete();
}
```

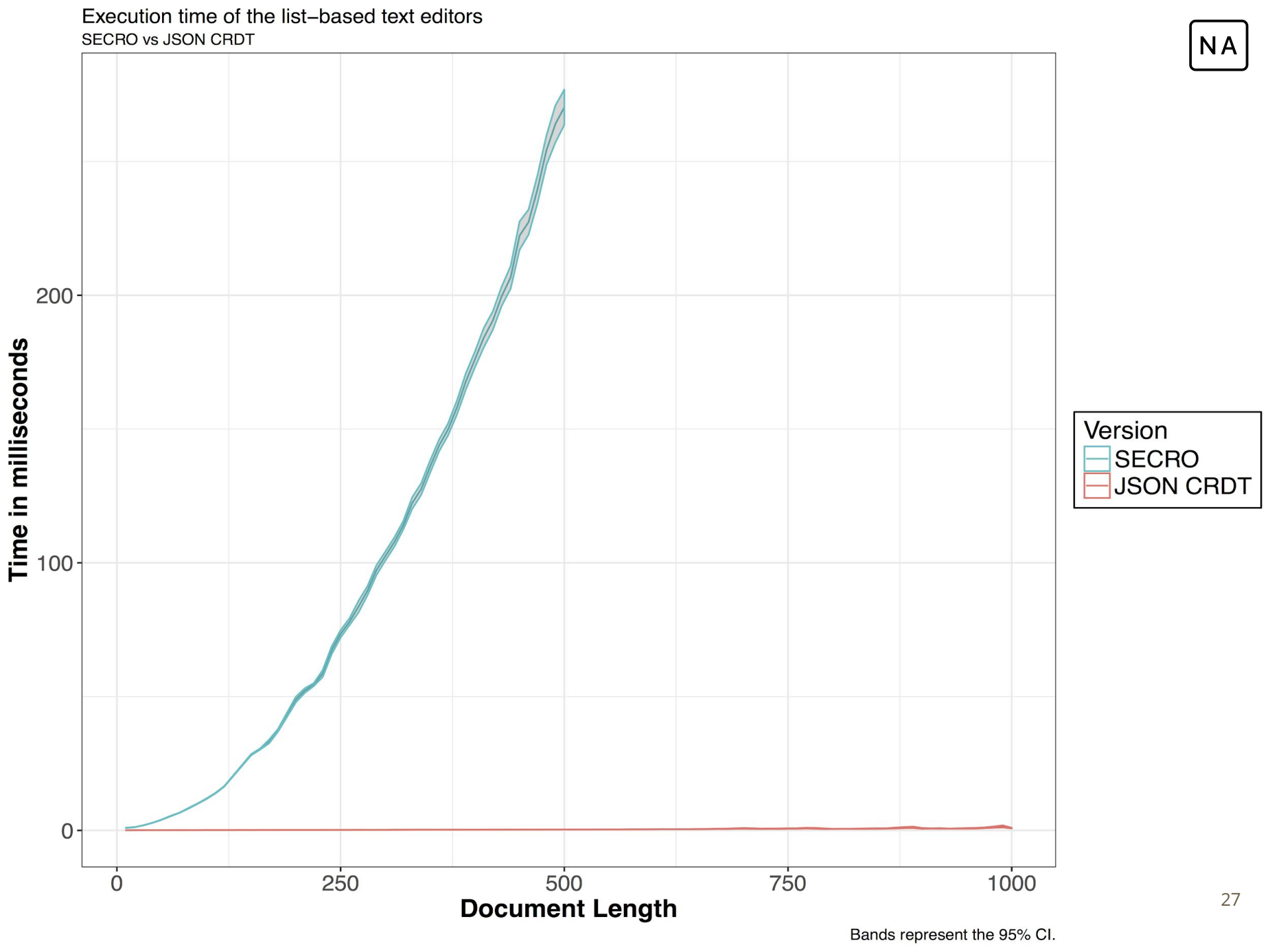
Evaluation (II)

- Benchmarks
  - Time complexity
  - Throughput
  - Memory

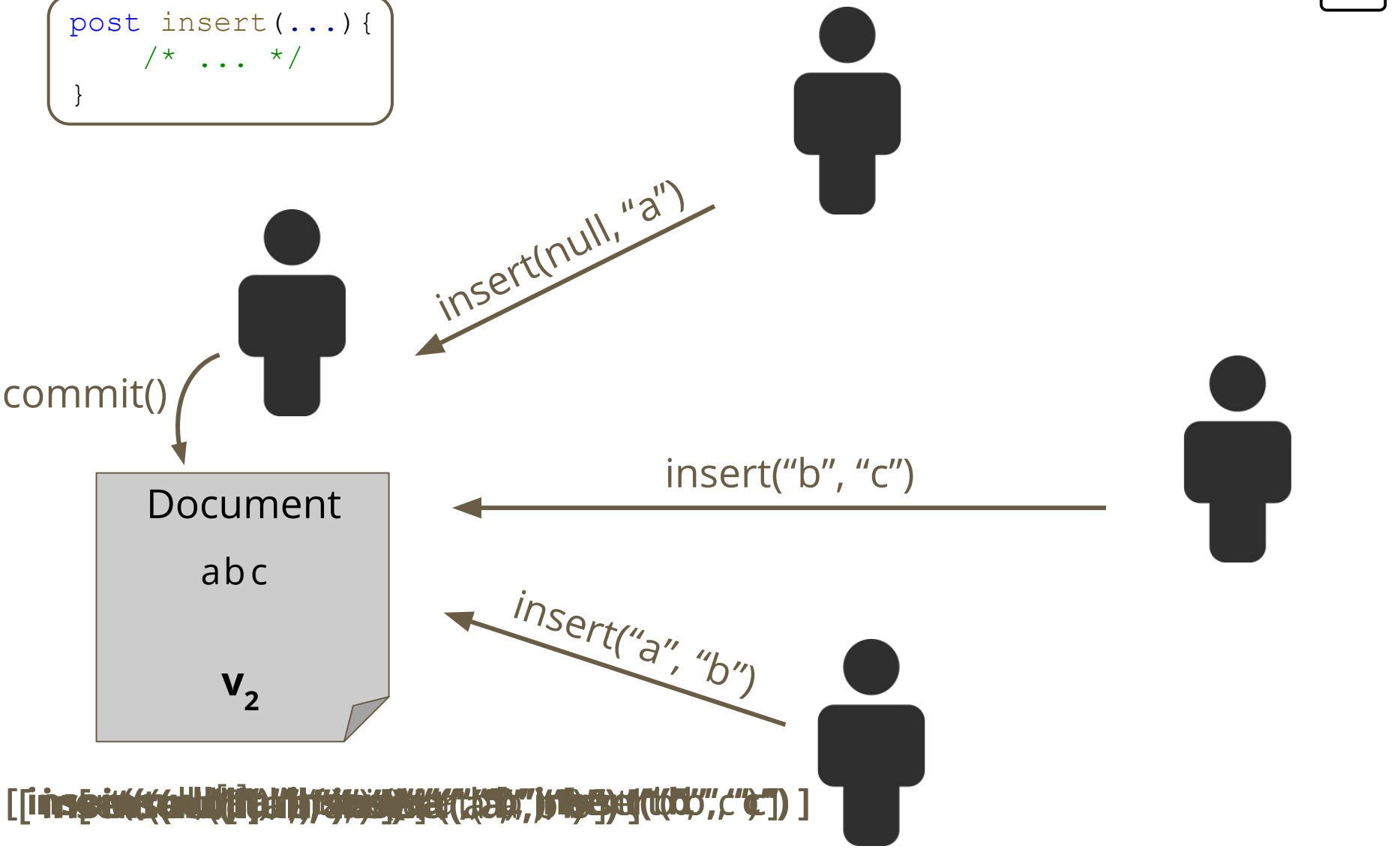
NA

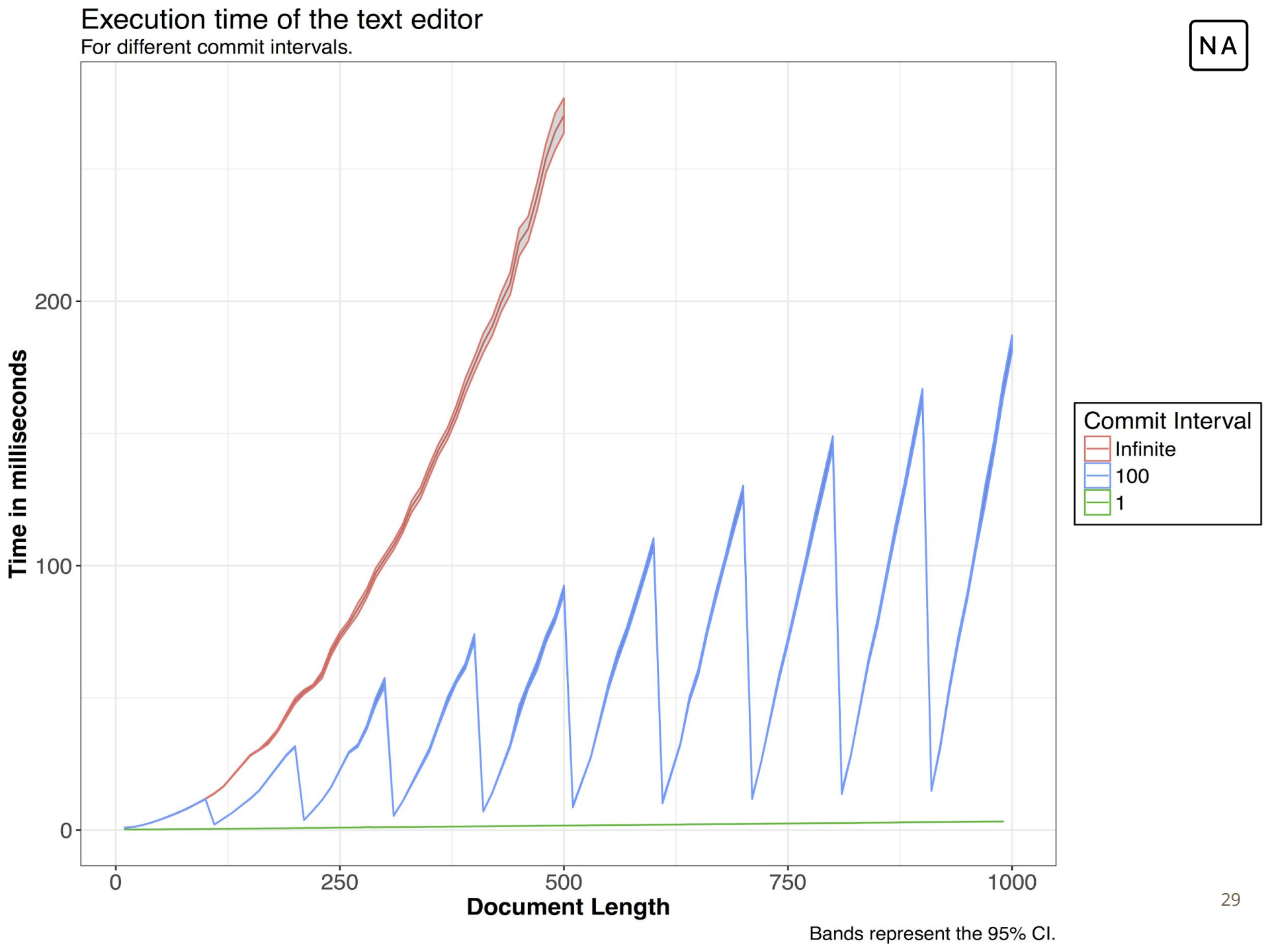
Methodology

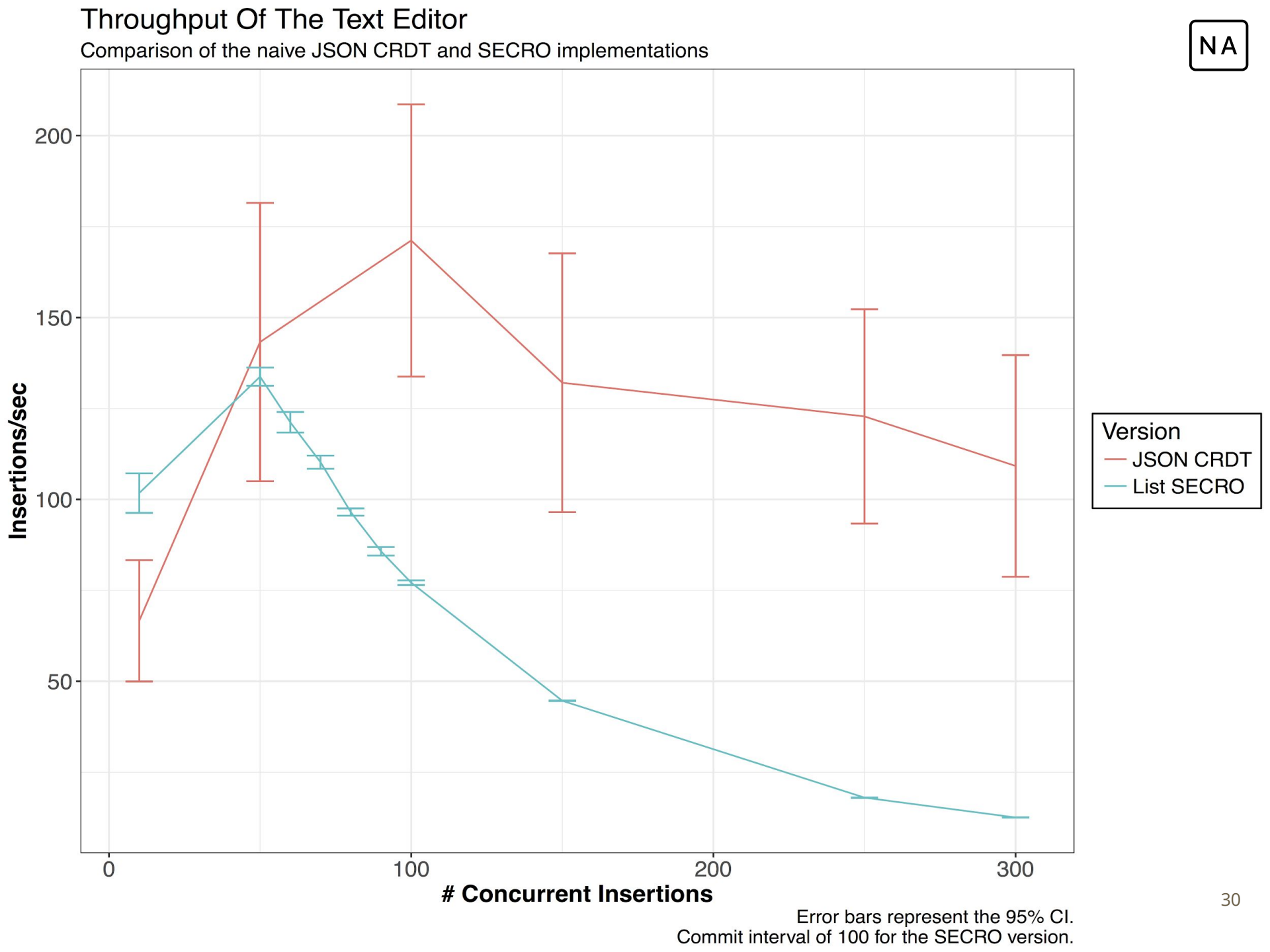
- Warm-up rounds
- Disabled JIT compiler
- At least 30 repetitions
- Discard samples affected by GC
- Compute avg and 95% CI

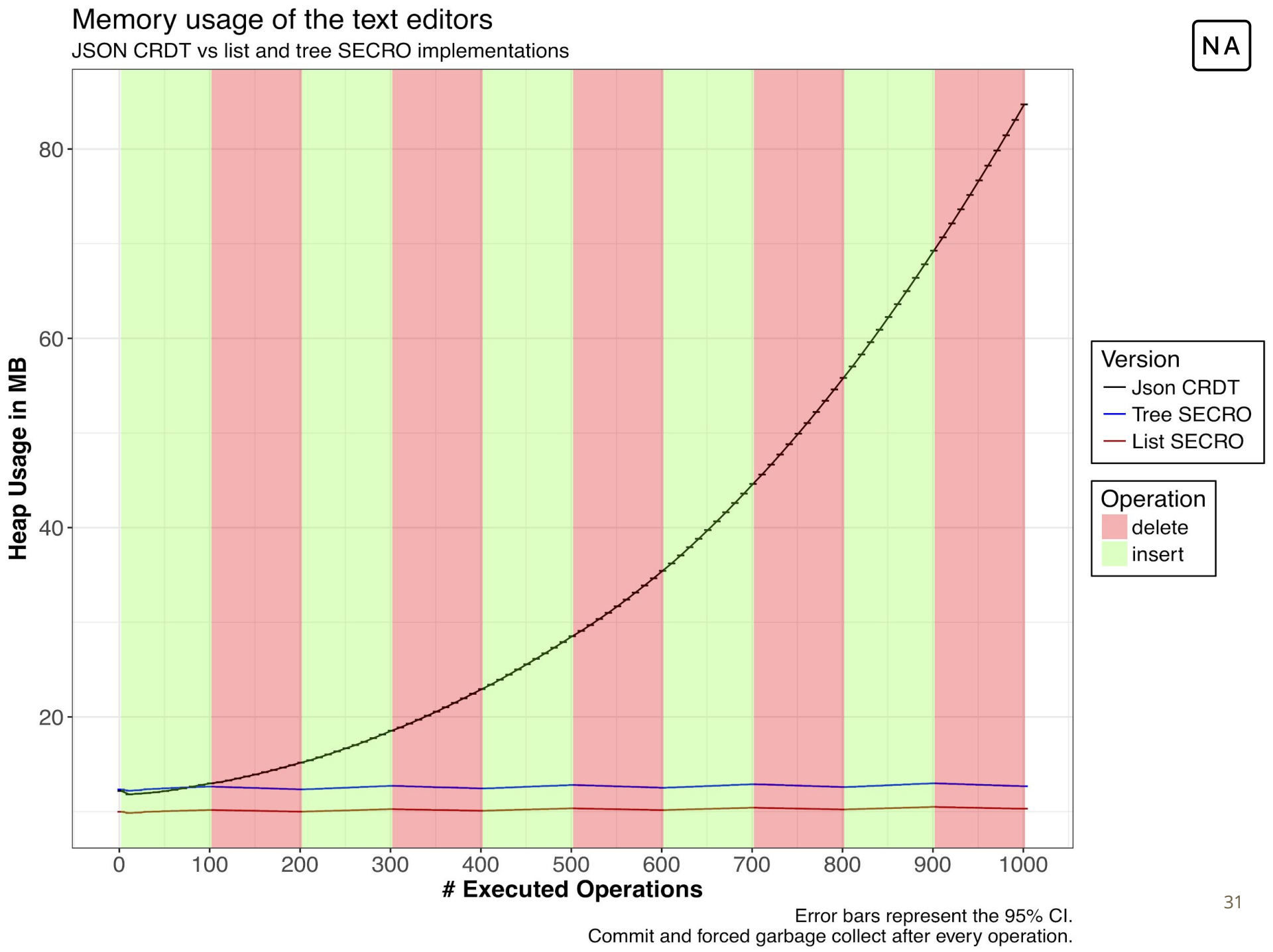

Execution time of the list–based text editors
SECRO vs JSON CRDT
NA
Time in milliseconds
200
100
0
0
250
500
750
1000
Document Length
Version
SECRO
JSON CRDT
Bands represent the 95% CI.

```
post insert(...){
    /* ... */
}
```



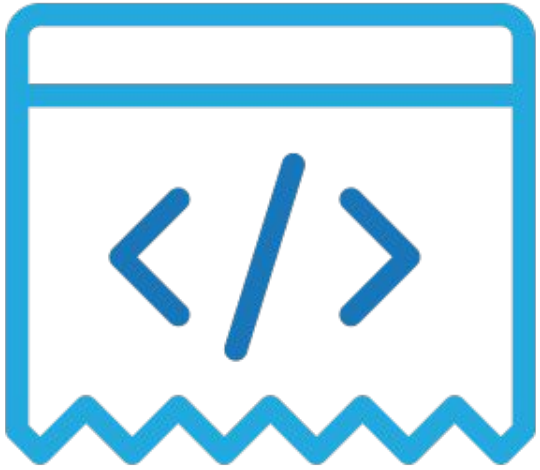

Execution time of the text editor
For different commit intervals.
NA
Time in milliseconds
200
100
0
0
250
500
750
1000
Document Length
Commit Interval
Infinite
100
1
Bands represent the 95% CI.


Throughput Of The Text Editor
Comparison of the naive JSON CRDT and SECRO implementations
NA
200
150
100
50
Insertions/sec
0
100
200
300
Concurrent Insertions
Version
JSON CRDT
List SECRO
Error bars represent the 95% CI.
Commit interval of 100 for the SECRO version.


Memory usage of the text editors
JSON CRDT vs list and tree SECRO implementations
NA
Heap Usage in MB
80
60
40
20
0
100
200
300
400
500
600
700
800
900
1000
Executed Operations
Version
Json CRDT
Tree SECRO
List SECRO
Operation
delete
insert
Error bars represent the 95% CI.
Commit and forced garbage collect after every operation.


Memory usage of the text editors
List vs tree SECRO implementations
NA
Heap Usage in MB
13
12
11
10
0
100
200
300
400
500
600
700
800
900
1000
Executed Operations
Version
Tree
List
Operation
delete
insert
Data
measured
regression line
Error bars represent the 95% CI.
Commit and forced garbage collect after every operation.

Results

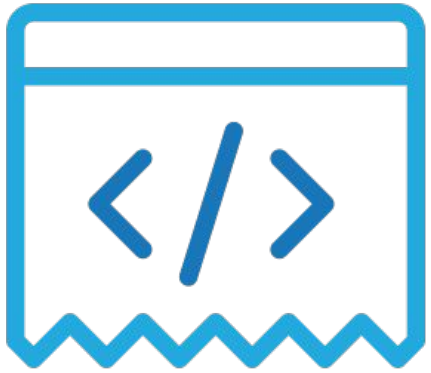


- More LOC
- But more flexible

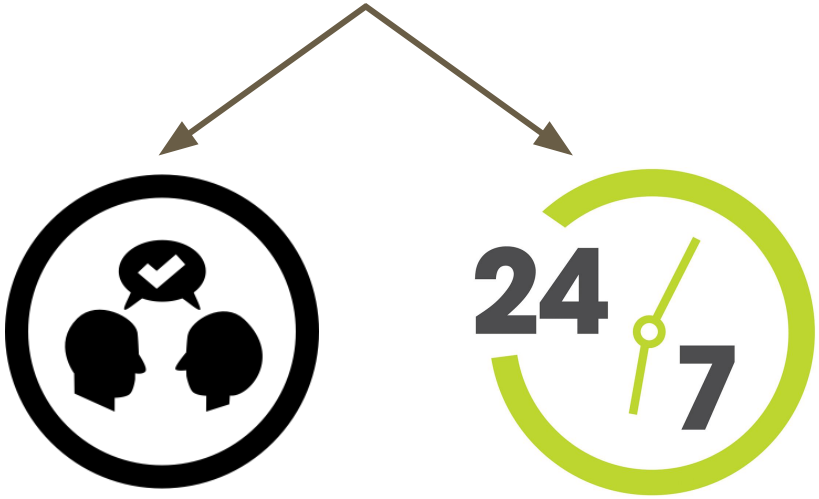
- Uses less memory
- But slower

  ⇒ commit

Conclusion

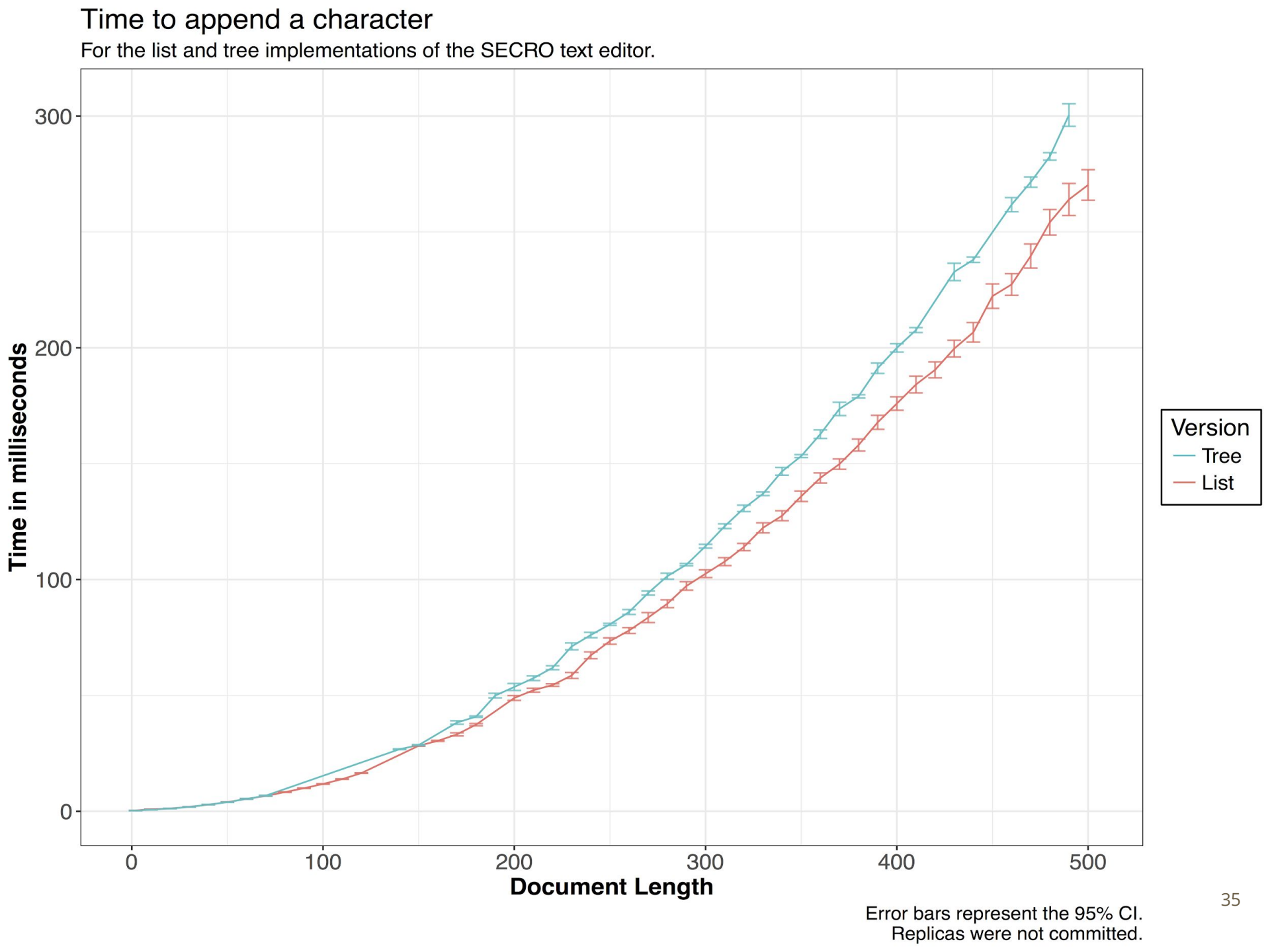


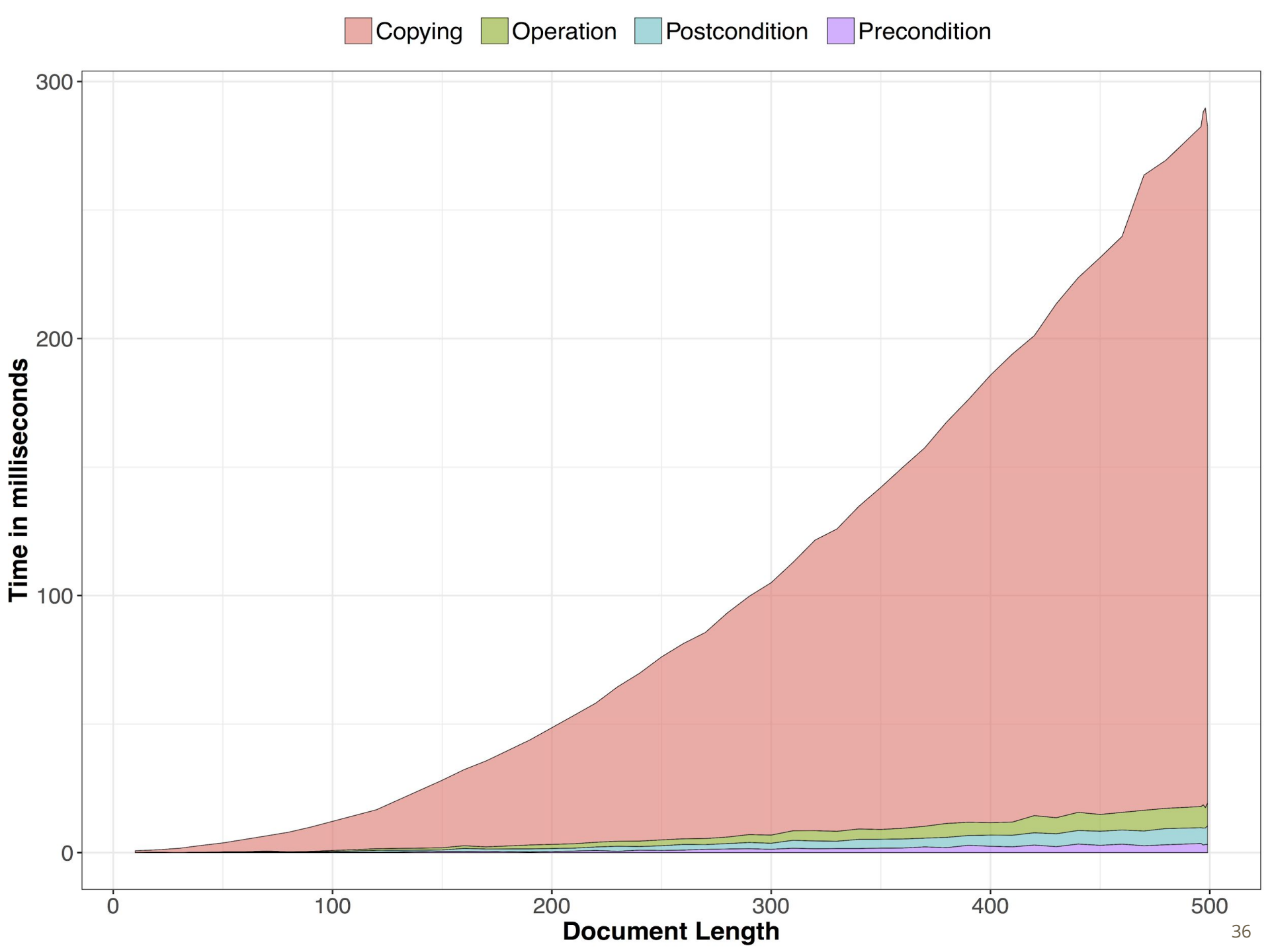
CScript: replicas

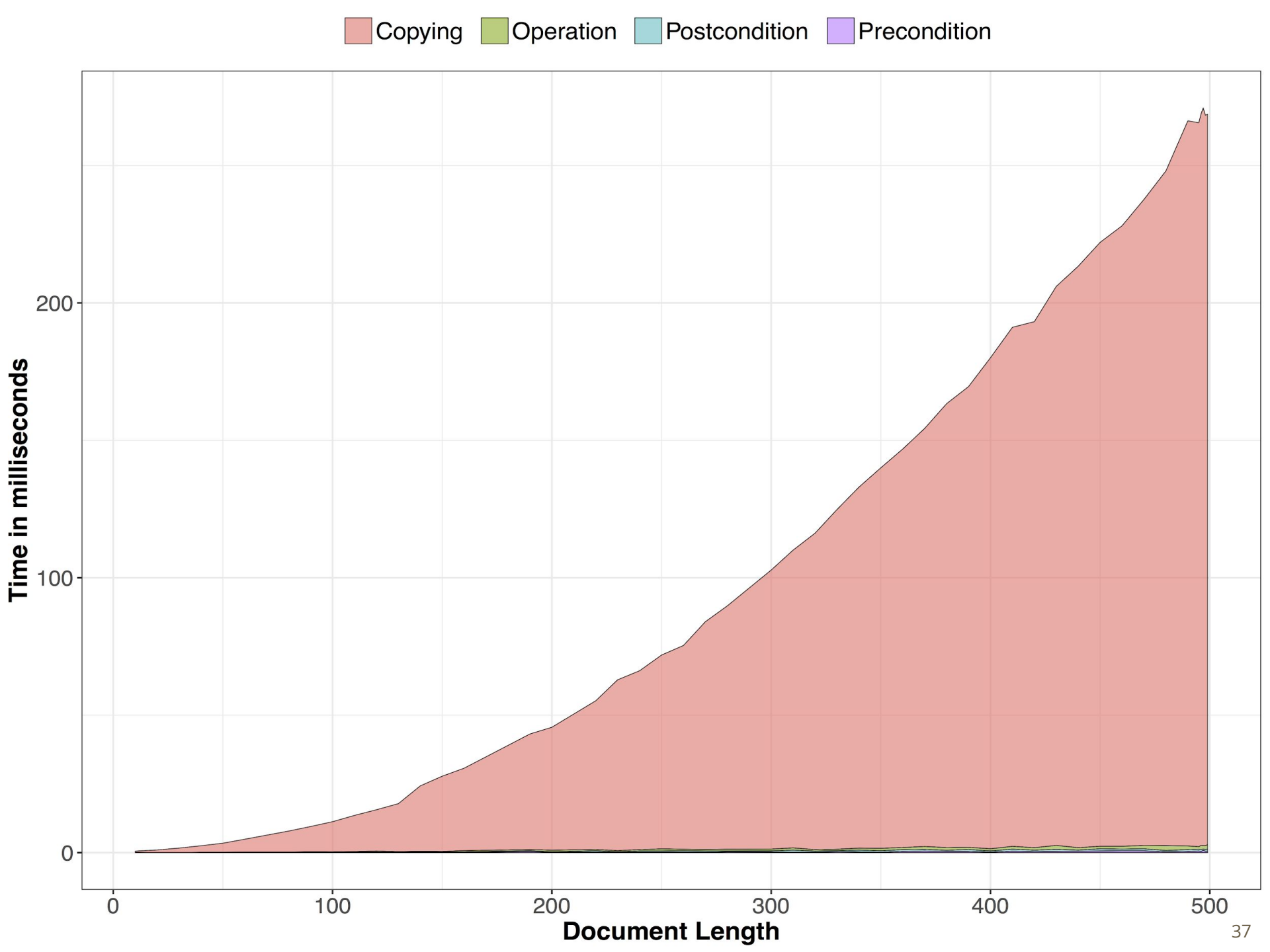


SECROs

- General-purpose
- Available
- SEC

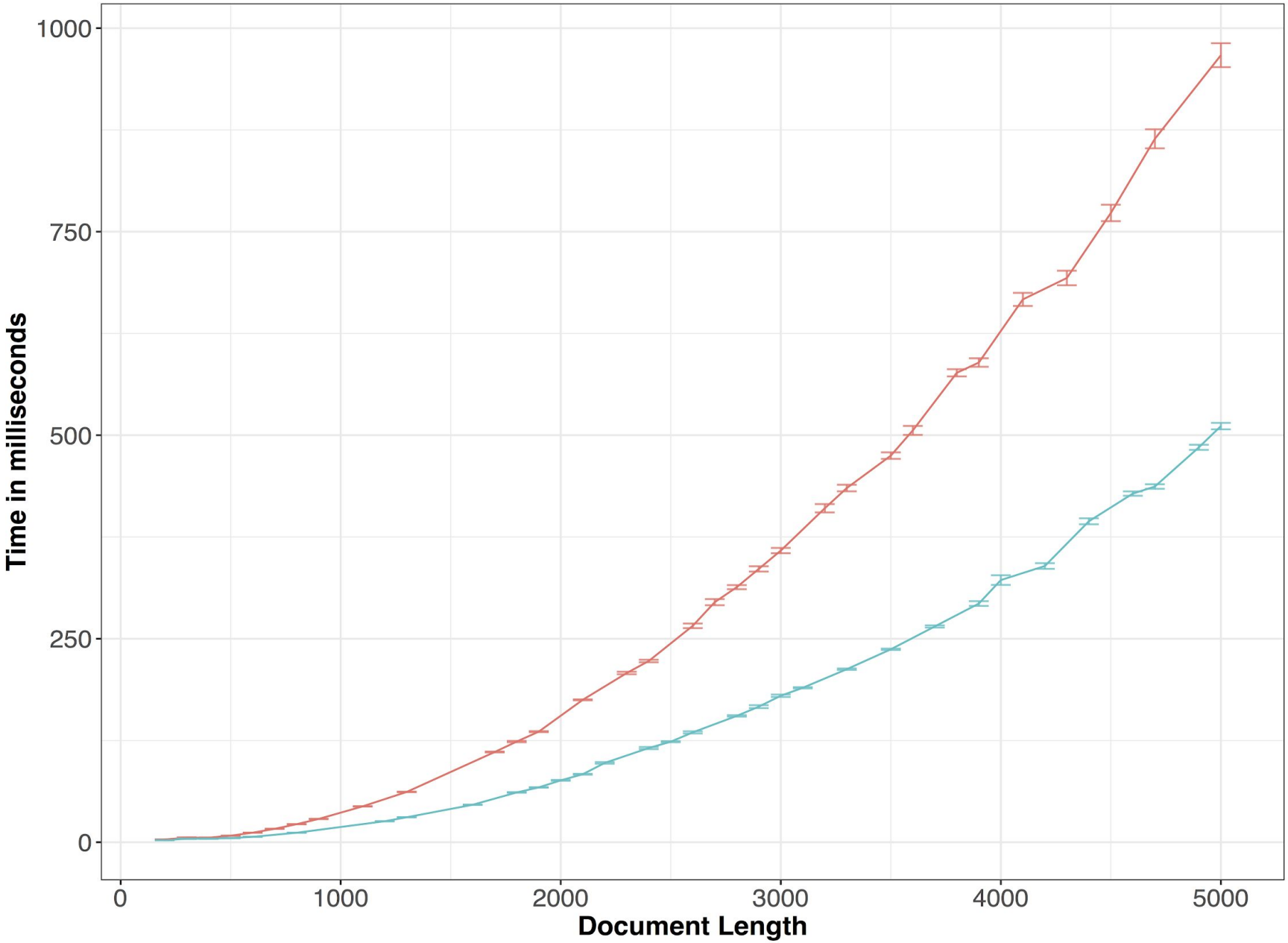

Time to append a character
For the list and tree implementations of the SECRO text editor.
Time in milliseconds
300
200
100
0
0
100
200
300
400
500
Document Length
Version
Tree
List
Error bars represent the 95% CI.
Replicas were not committed.


Copying
Operation
Postcondition
Precondition
300
200
100
0
Time in milliseconds
0
100
200
300
400
500
Document Length


Copying
Operation
Postcondition
Precondition
Time in milliseconds
0
100
200
0
100
200
300
400
500
Document Length

Time to append 100 characters

For the list and tree implementations of the SECRO text editor.



Error bars represent the 95% CI.
Replicas were not committed.