Index-Based Code Clone Detection: Incremental, Distributed, Scalable

Benjamin Hummel
Elmar Juergens
Lars Heinemann

Technische Universität München

Michael Conradt

Google Germany GmbH

```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
   boolean found = false;
   StringBuffer res = new StringBuffer();
   if (elements != null) {
      Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
      for (int i = 0; i < elements.length; i++) {
         ModelElement el = elements[i];
         res.append(showElementLink(el)).append(HTML.LINE_BREAK);
         found = true;
      }
      Index.getInstance().resetCurrentRenderer();
   }
   if (!found && nomsg != null && nomsg.length() > 0) {
      res.append(HTML.italics(nomsg));
   }
   return res.toString();
}
```

```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
   boolean found = false;
   StringBuffer res = new StringBuffer();
   if (elements != null) {
      Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
      for (int i = 0; i < elements.length; i++) {
         ModelElement el = elements[i];
         res.append(showElementLink(el)).append(HTML.LINE_BREAK);
         found = true;
      }
      Index.getInstance().resetCurrentRenderer();
   }
   if (!found && nomsg.length() > 0) {
      res.append(HTML.italics(nomsg));
   }
   return res.toString();
}
```

Code Cloning

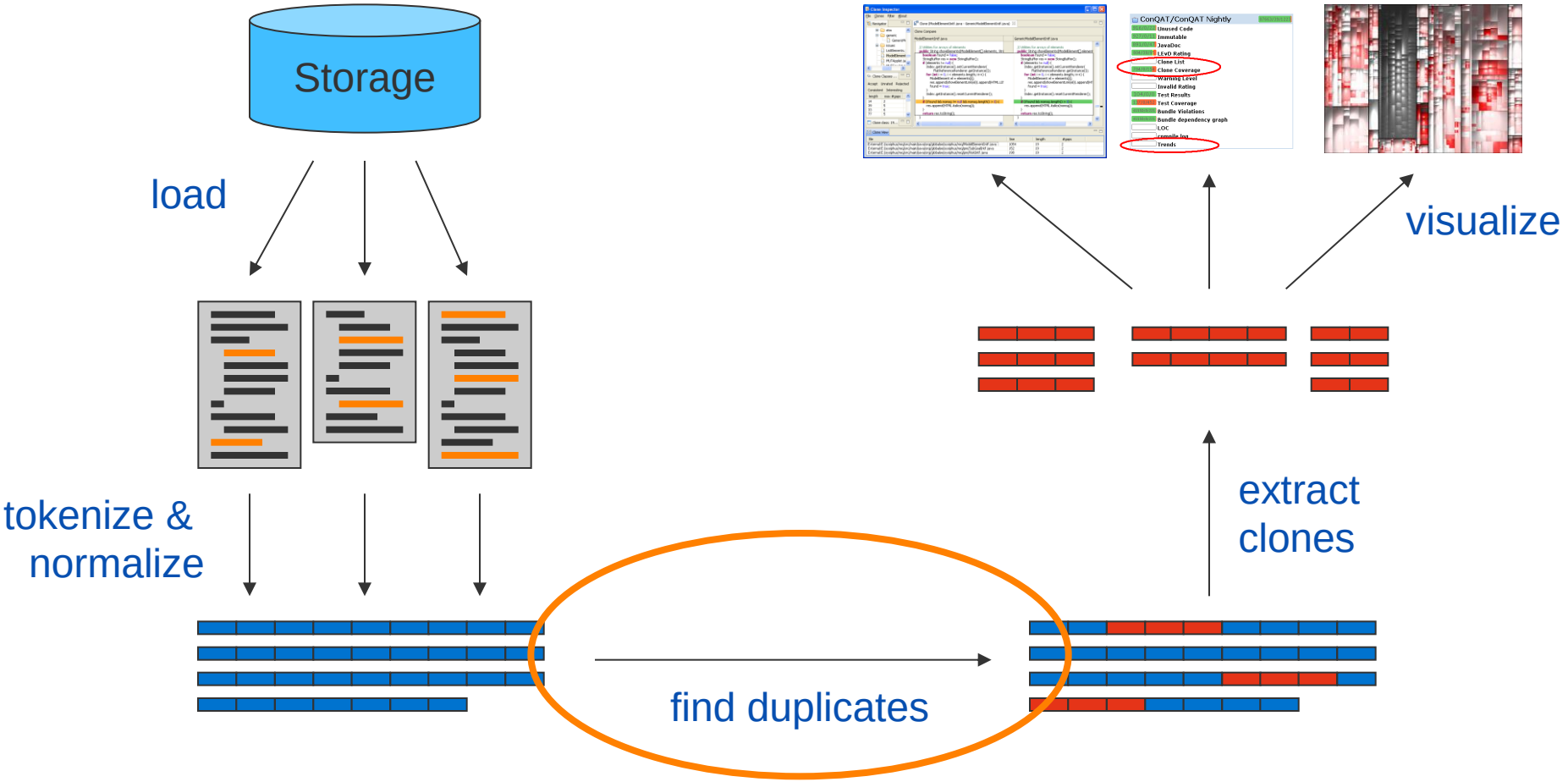
Reasons

- Time pressure
- Code Ownership
- Unsuited abstraction mechanisms

Consequences

- Increased code size
- Modification of one clone probably affects other clones
  => Increased modification effort
- Inconsistent changes can cause bugs

Clone Detection



Challenge 1: Large Code Bases

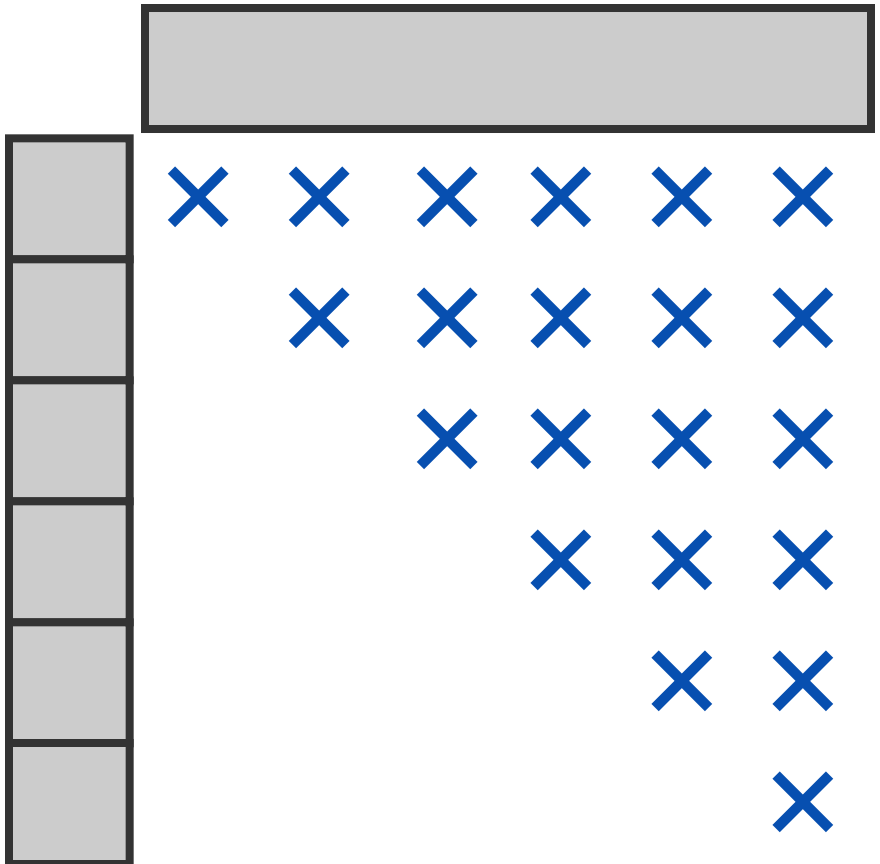
What if we want to run clone detection on all source code of a company like Google?

Or Microsoft, Oracle, SAP, …?

→ Distributed Clone Detection

Existing Work

D-CCFinder [Livieri et al., ICSE'07]



O(n²) jobs

Challenge 2: Frequent Updates

How are the clones affected if a developer changes a couple of files?

Potentially a copy to any other source file!

→ Incremental Clone Detection

Existing Work

Use generalized suffix-tree that supports updating of single substrings (= files)

[Göde, Koschke, CSMR'09]

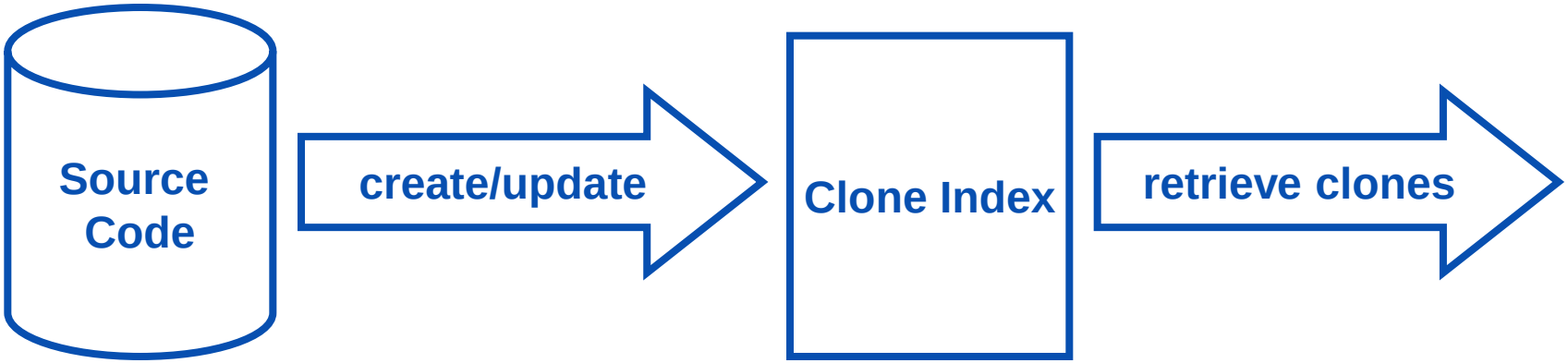
- **Complicated data structure (hard to implement)**
- **No obvious way to distribute on several machines**

Contribution

- Token-based clone detection algorithm based on index structure
- Supports both distributed and incremental detection
- Evaluation of performance in several scenarios

Algorithm

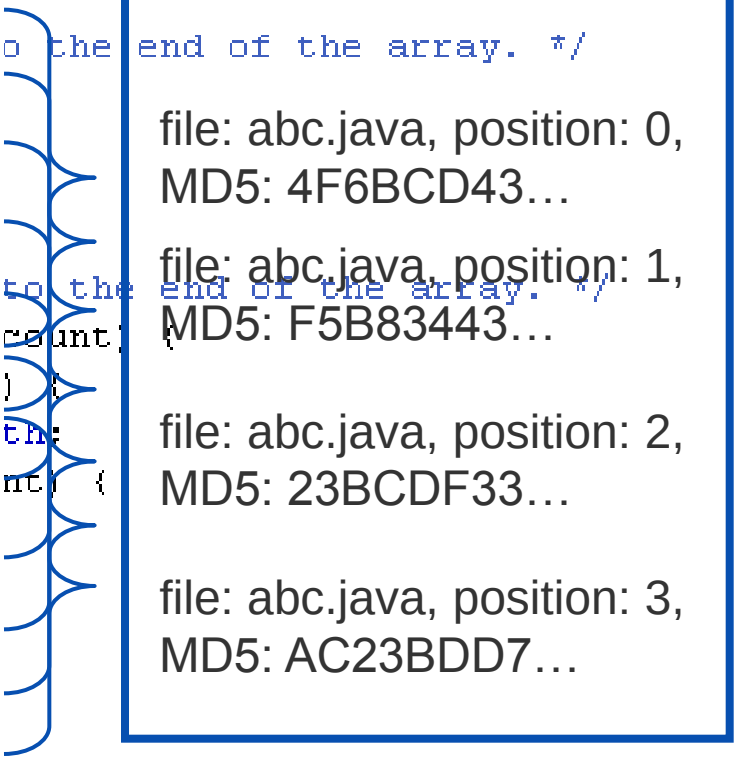
Big Picture



Clone Index

```
protected void i0()
i0(l0);
protected void i0(int i1)
if (i0 + i1 >= i2.i3)
int i0 = l0 * i1.i2;
while (i0 <= i1 + i2)
i0 *= l0;
int[] i0 = i1;
i0 = new int[i1];
i0.i1(i2, l0, i3, l0, i4);
i0 += i1;
```

```
o the end of the array. */
to the end of the array. */
count) {
) {
th;
nt) {
        System.arraycopy(oldArray, 0, array, 0, size);
    }
    size += count;
}
```



Clone Retrieval

Read chunks by file

file: abc.java, position: 0,
MD5: 4F6BCD43…

file: abc.java, position: 1,
MD5: F5B83443…

file: abc.java, position: 2,
MD5: 23BCDF33…

file: abc.java, position: 3,
MD5: AC23BDD7…

….

Read chunks by MD5 hash

file: efg.java, position: 7,
MD5: F5B83443…

file: hij.java, position: 4,
MD5: F5B83443…

file: efg.java, position: 8,
MD5: 23BCDF33…

Clone of length 6

Index Implementation

Index has to

- **store and update tuples**
- **support tuple lookup by file and hash**

Many possible implementations:

- **balanced tree, B-tree**
- **(distributed) hash table**

Case Studies

using ConQAT (http://www.conqat.org/)

Batch Detection

Computer: 2.4 GHz CPU / 4 GB RAM

	JabRef	Commercial	Linux
Language	Java	APAB	C
kLOC	115	461	11,250
suffix-tree	7.3 sec	28.7 sec	166 min
index-based	6.7 sec	28.7 sec	47 min

Same cloning results for both algorithms 

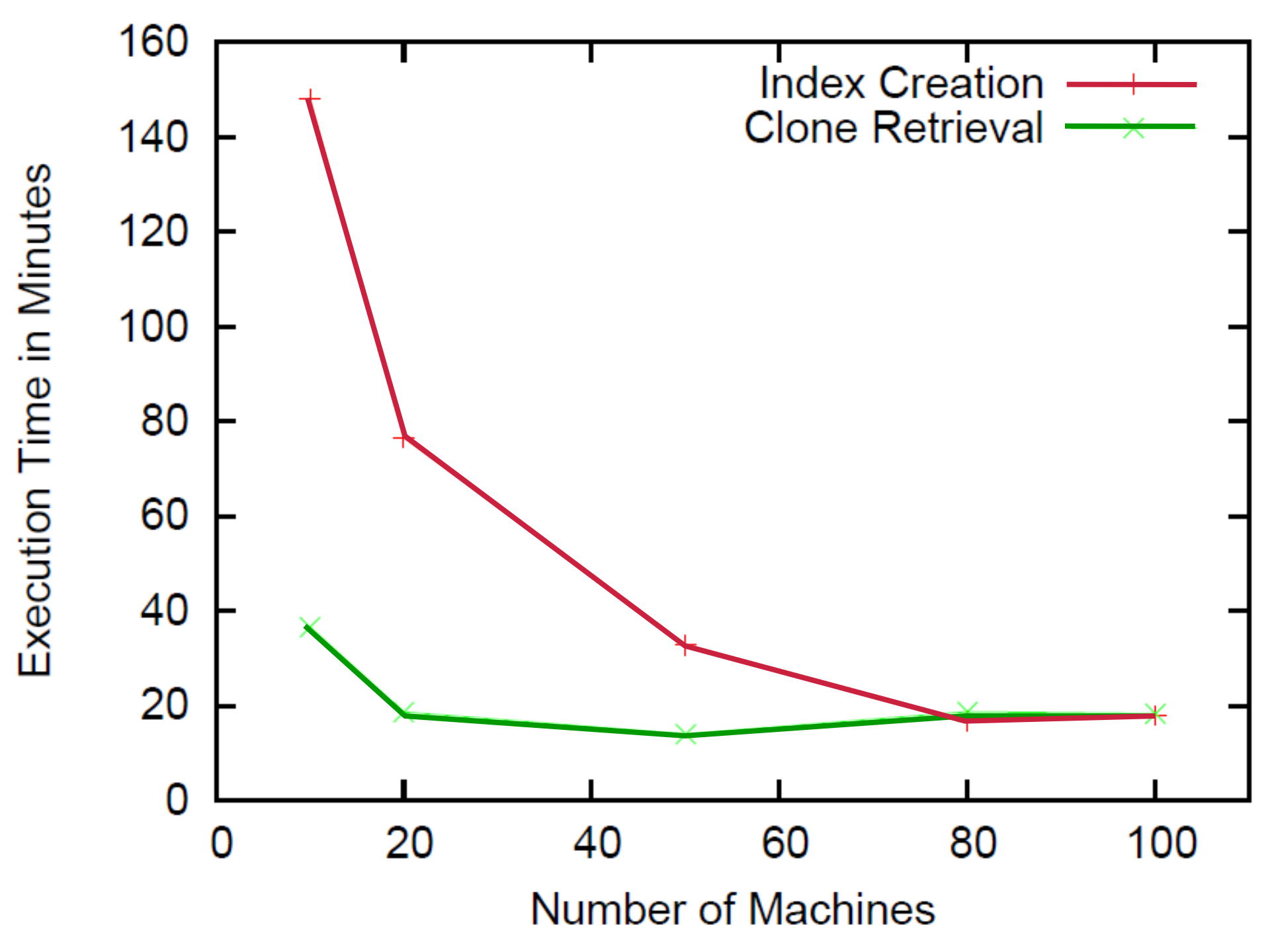
Distributed Detection (1)

Google Infrastructure:

- **MapReduce for distributing computing tasks**
- **Bigtable for storing index**
- **73.2 MLOC of Open Source code (C/C++/Java)**

Procedure:

- **Determine running time on 10, 20, 50, 80, 100 machines**


Index Creation
Clone Retrieval
Execution Time in Minutes
Number of Machines
0
20
40
60
80
100
120
140
160
0
20
40
60
80
100

Incremental Detection (1)

Study Object: Eclipse 3.3 (42 MLOC)

Index: disk-based (BerkeleyDB)

Procedure:

- Create full index and close all DB files
- Perform clone query on 1000 random files
- Remove and add 1000 random files

Incremental Detection (2)

Index creation:	**7 hours**
Index size:	**5.6 GB on hard disk**
Query time:	**0.91 seconds on average**
	0.21 seconds median
Update time:	**0.85 seconds on average**

Conclusion

Summary

- Simple algorithm that is both
  scalable/distributable and incremental
- Demonstrated performance in several case studies

Future Work: type-3 clones