Very-Large Scale Code Analysis and Visualization of Open Source Programs Using Distributed CCFinder: D-CCFinder

Simone Livieri
Yoshiki Higo
Makoto Matsushita
Katsuro Inoue

Software Engineering Laboratory
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Outline

- Background
  - Mega Software Engineering
  - Code-Clone Analysis
  - CCFinder
- D-CCFinder
- Experiments
- Discussion
- Conclusion

Background

Software Engineering

- Various technologies for software quality improvement and development efficiency
- Mostly focused on a single project or programmer
- Most technologies are optimized for the local benefit of a single user or project

Mega Software Engineering

- Software Engineering for Projects Collections
  - Huge data collection for a large number of projects
  - Intensive data analysis beyond a single project's boundary
  - Information feedback for organizational improvement

Mega Software Engineering

- Software Engineering for Projects Collections
  - Huge data collection for a large number of projects
  - Intensive data analysis beyond a single project's boundary
  - Information feedback for organizational improvement

Mega Software Engineering Framework

Intensive Data Analysis

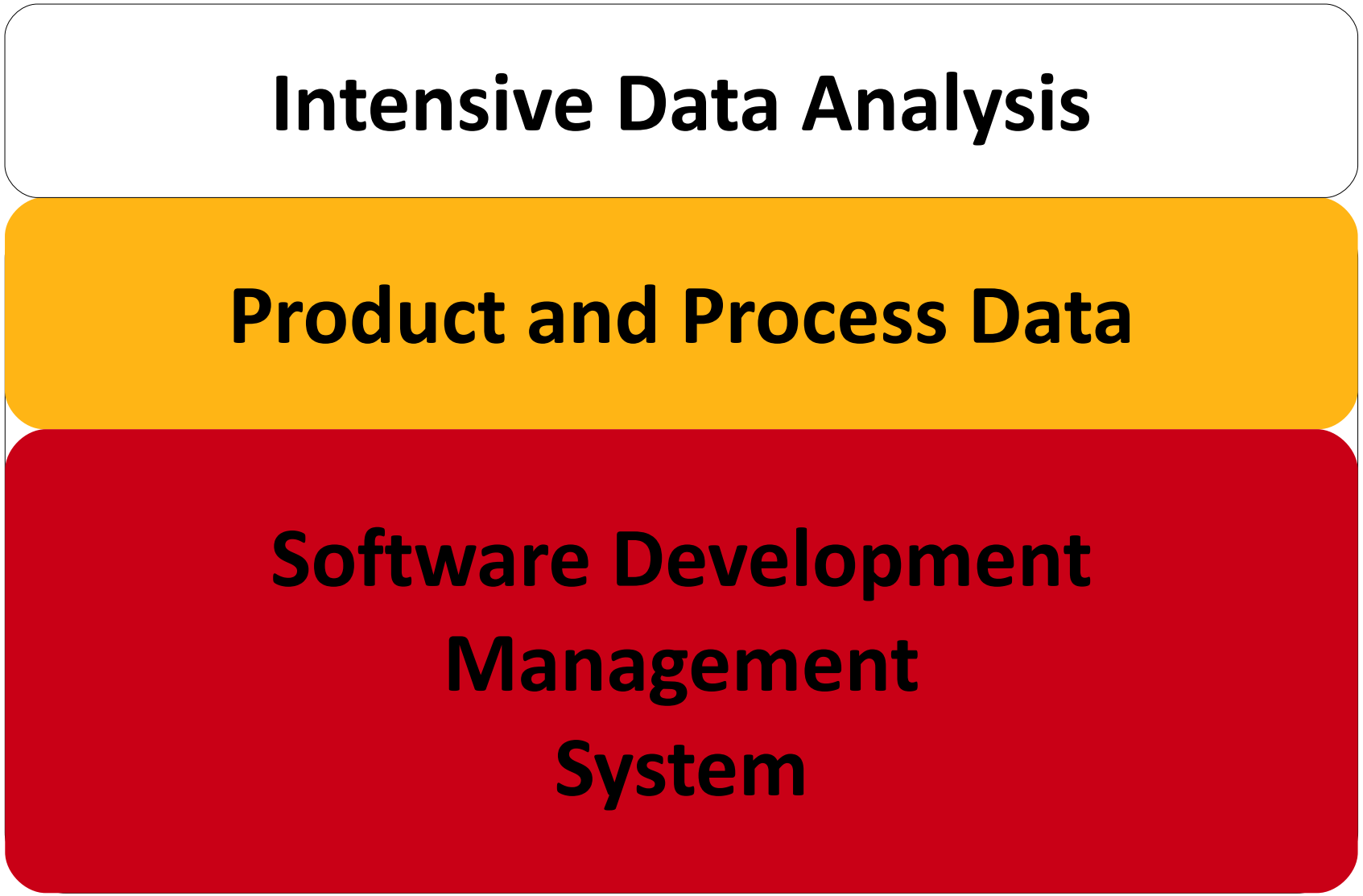
Huge Data Collection

Mega Software Engineering Framework

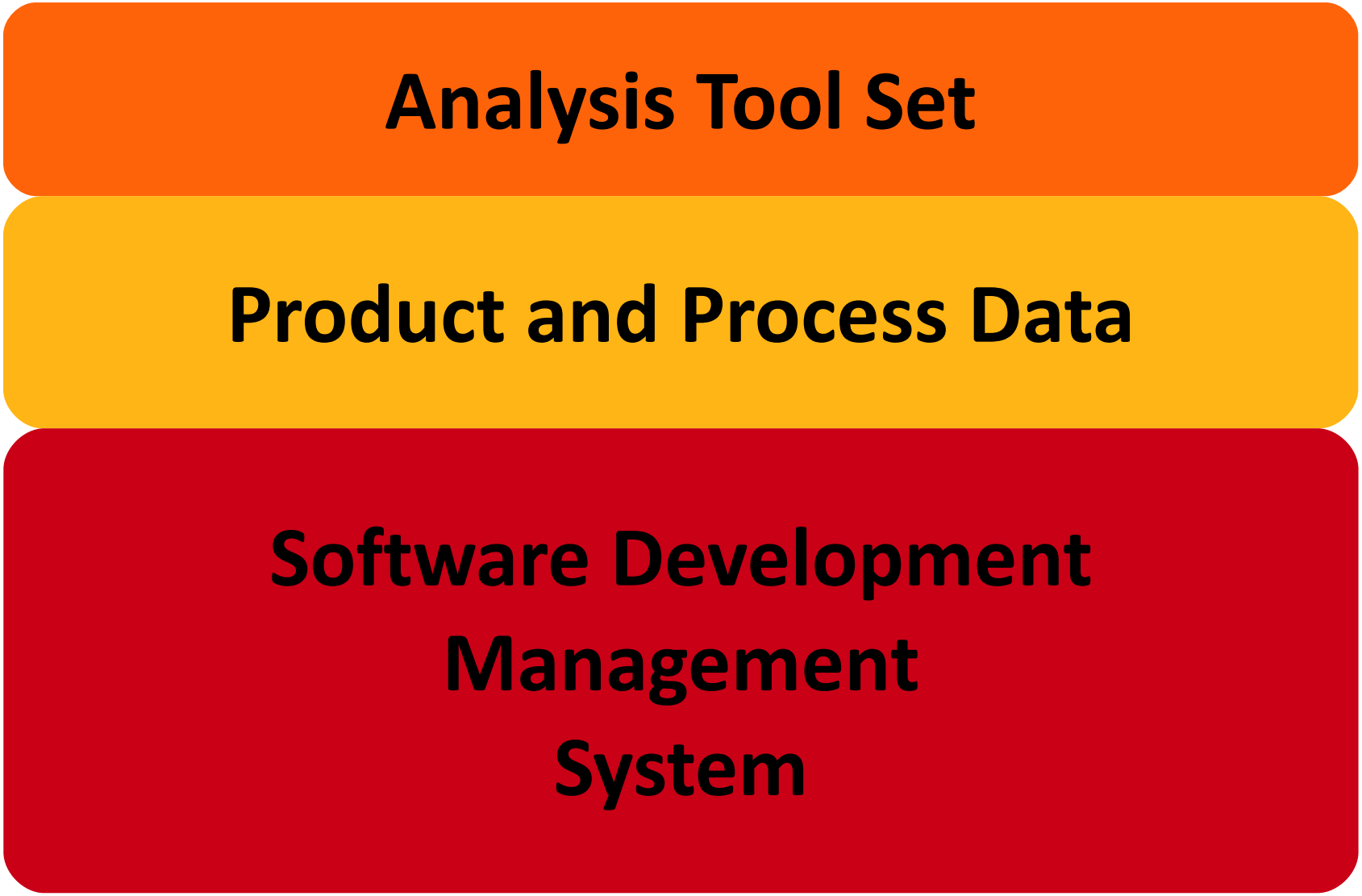
Intensive Data Analysis

Software Development Management System

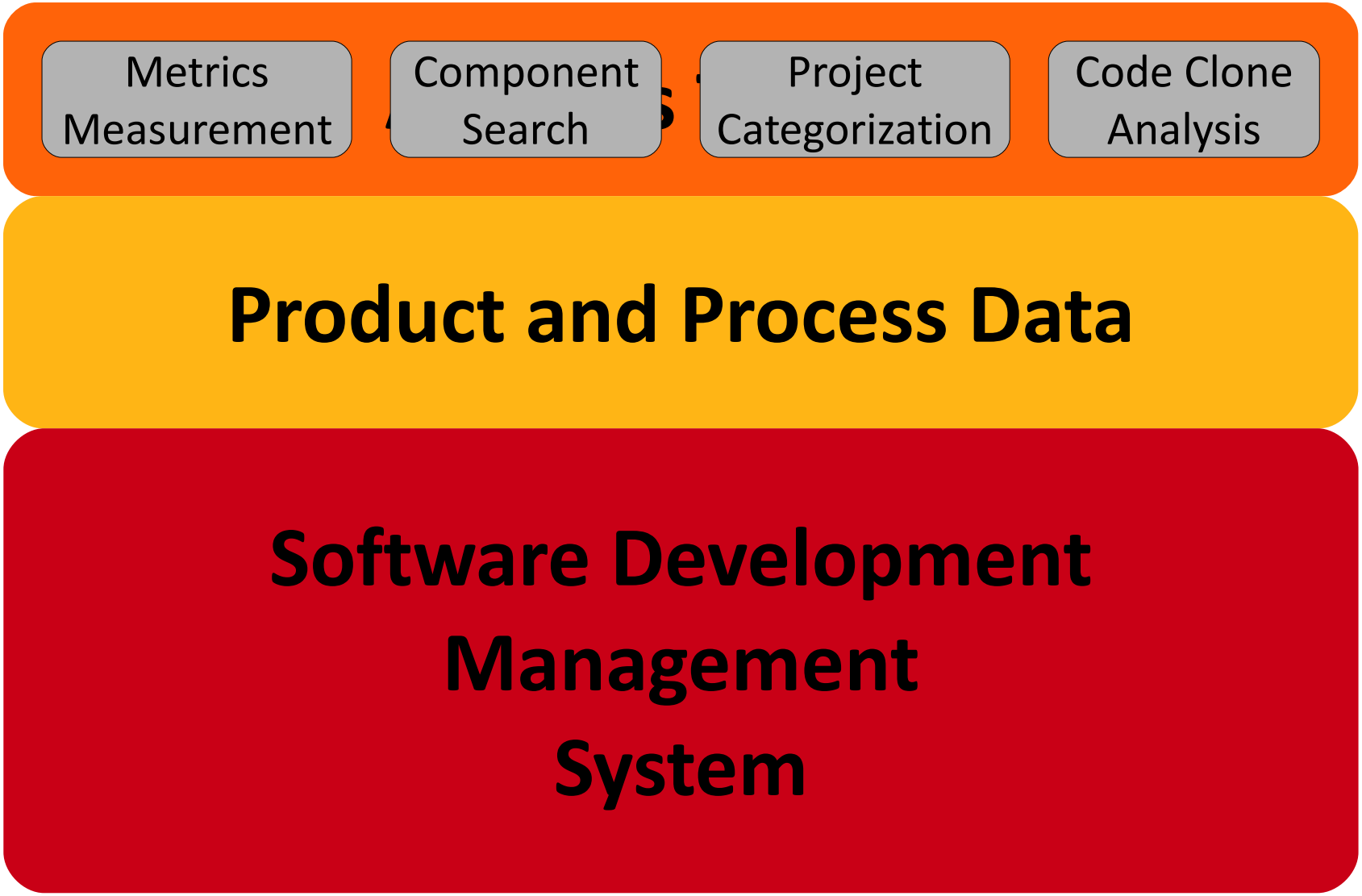
Mega Software Engineering Framework



Mega Software Engineering Framework

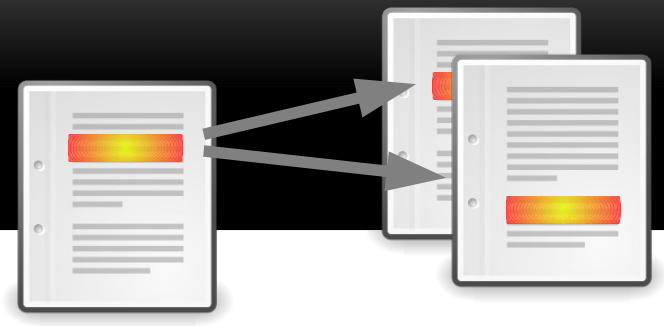


Mega Software Engineering Framework



Code Clone Analysis

Code Clone?



- A code clone is a set of identical or similar code fragments
- Generated by:
  - Code reuse by copy & paste
  - Stereotyped functions or tool generated code
  - Intentional iteration for performance enhancement
  - ...

Code Clone Detection

- Various detection methods with different complexity and performance
  - Line-by-line comparison
  - Suffix Tree
  - Abstract Syntax Tree
  - Program Dependence Graph
  - ...

CCFinder

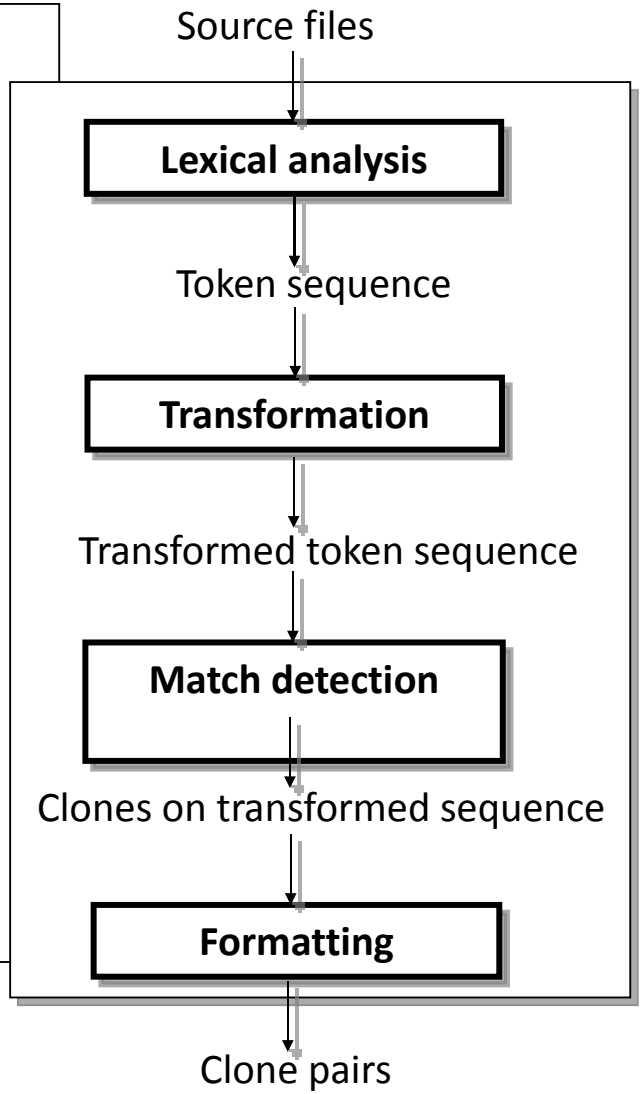
- Suffix tree based code clone detection tool
- Lexical analysis of the source code
- Insensitive to renamed variable and changed code layout
- Multi language support (C, C++, COBOL, Java, ...)
- Good scalability and speed (5MLOC/20m)
- Widely used by both researchers and practitioners

CCFinder – Detection Process

- Lexical analysis
- Transformation
- Detection
- Formatting

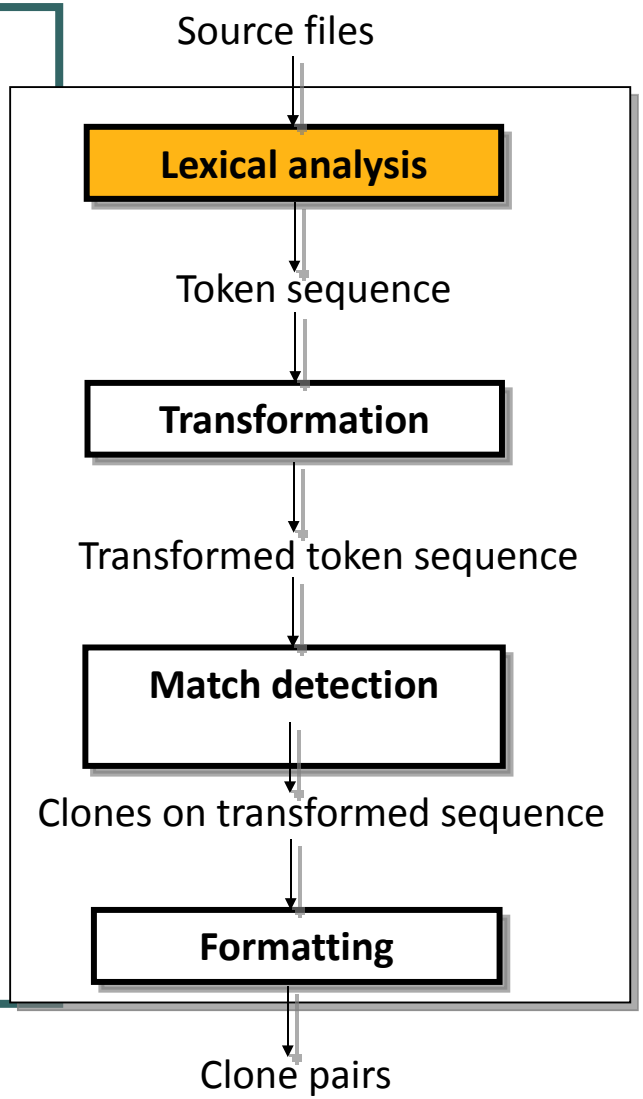
CCFinder - Detection

```
 1. static void foo() throws RESyntaxException {
 2.     String a[] = new String [] { "123,400", "abc",
"orange 100" };
 3.     org.apache.regexp.RE pat = new
org.apache.regexp.RE("[0-9,]+");
 4.     int sum = 0;
 5.     for (int i = 0; i < a.length; ++i)
 6.         if (pat.match(a[i]))
 7.             sum += Sample.parseNumber(pat.getParen(0));
 8.     System.out.println("sum = " + sum);
 9. }
10. static void goo(String [] a) throws RESyntaxException
11.     RE exp = new RE("[0-9,]+");
12.     int sum = 0;
13.     for (int i = 0; i < a.length; ++i)
14.         if (exp.match(a[i]))
15.             sum += parseNumber(exp.getParen(0));
16.     System.out.println("sum = " + sum);
17. }
```



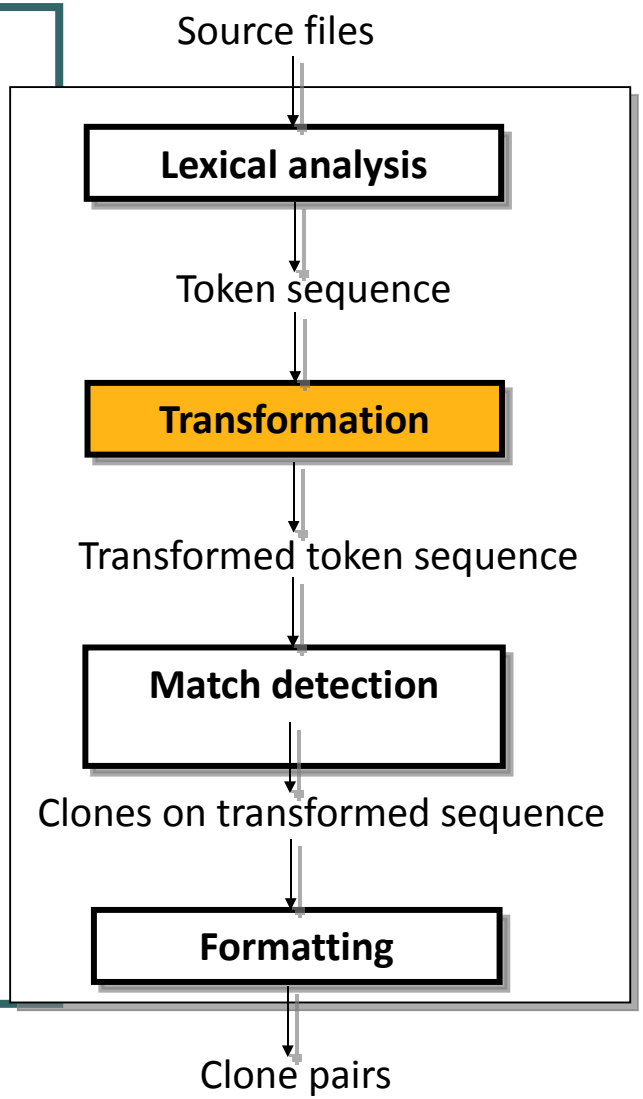
CCFinder - Detection

```
static void foo ( ) throws RESyntaxException { String a
[ ] = new String [ ] { "123,400" ,
"abc" , "orange 100" } ; org . apache . regexp
. RE pat = new org . apache . regexp
. RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( pat
. match ( a [ i ] ) ) sum
+= Sample . parseNumber ( pat . getParen ( 0
) ) ; System . out . println ( "sum = "
+ sum ) ; } static void goo ( String
a [ ] ) throws RESyntaxException { RE exp =
new RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( exp
. match ( a [ i ] ) ) sum
+= parseNumber ( exp . getParen ( 0 ) )
; System . out . println ( "sum = " + sum
) ; }
```



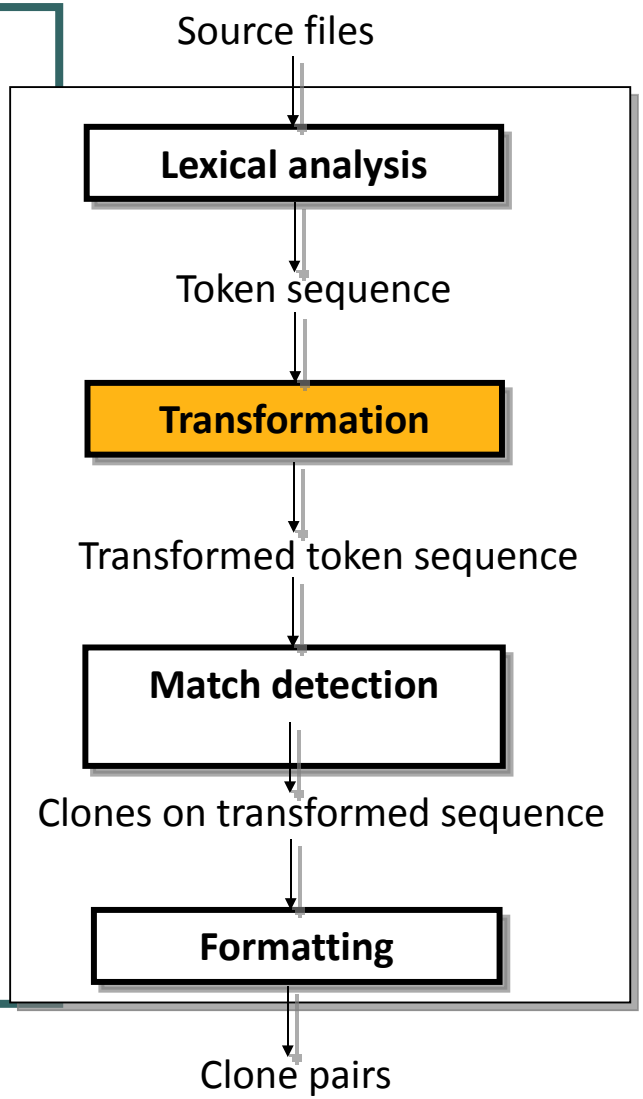
CCFinder-L detection

```
static void foo ( ) throws RESyntaxException { String a
[ ] = new String [ ] { $
                       } ;
    RE pat = new
    RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( pat
. match ( a [ i ] ) ) sum
+= Sample . parseNumber ( pat . getParen ( 0
) ) ; System . out . println ( "sum = "
+ sum ) ; } static void goo ( String
a [ ] ) throws RESyntaxException { RE exp =
new RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( exp
. match ( a [ i ] ) ) sum
+= $ . parseNumber ( exp . getParen ( 0 ) )
; System . out . println ( "sum = " + sum
) ; }
```

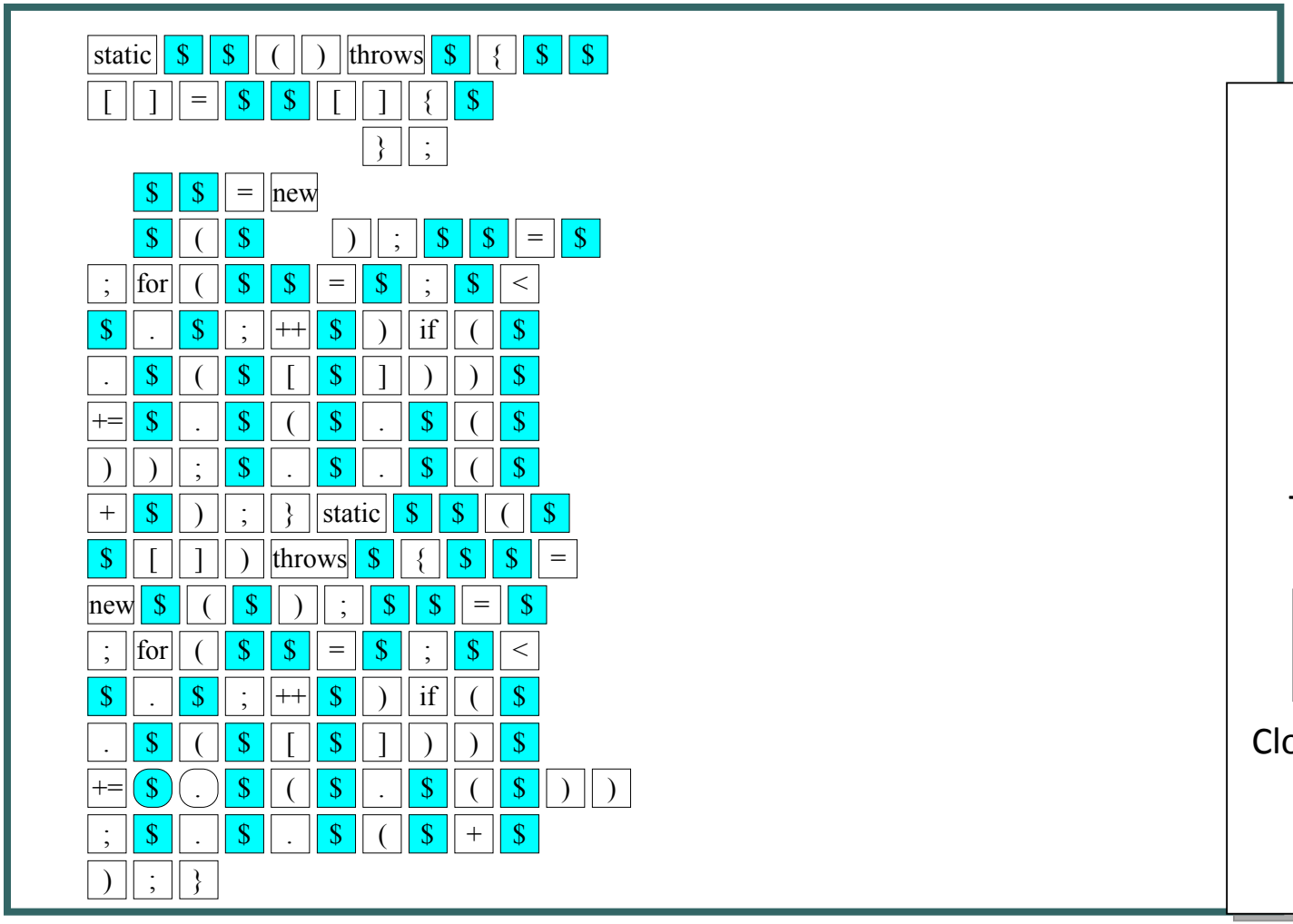


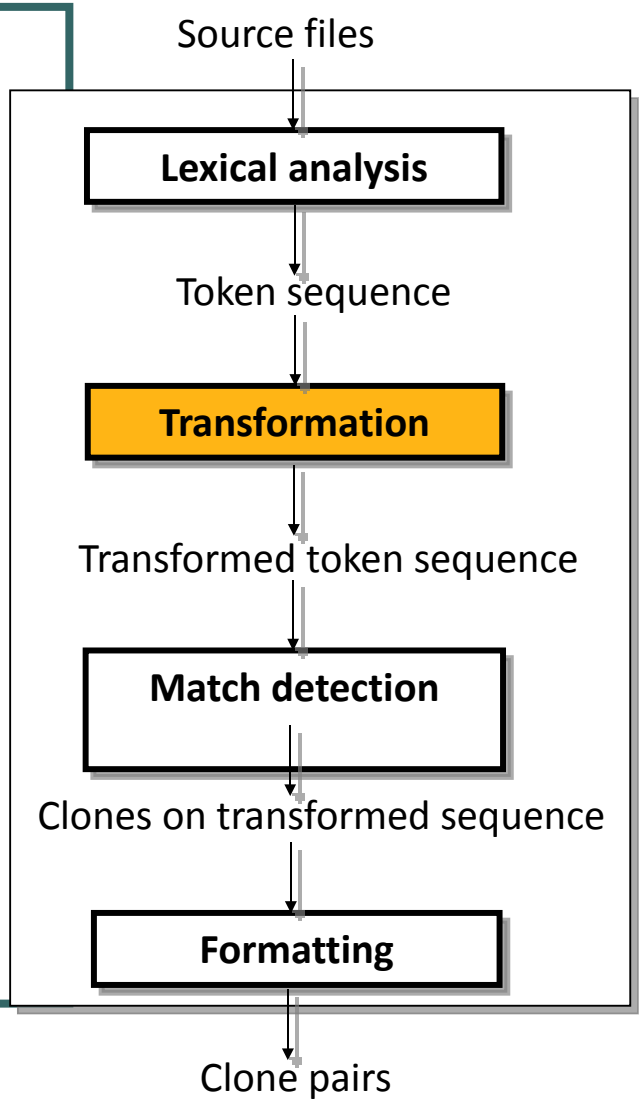
CCFinder - Detection

```
static void foo ( ) throws RESyntaxException { String a
[ ] = new String [ ] { $
                       } ;
    RE pat = new
    RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( pat
. match ( a [ i ] ) ) sum
+= Sample . parseNumber ( pat . getParen ( 0
) ) ; System . out . println ( "sum = "
+ sum ) ; } static void goo ( String
a [ ] ) throws RESyntaxException { RE exp =
new RE ( "[0-9,]+" ) ; int sum = 0
; for ( int i = 0 ; i <
a . length ; ++ i ) if ( exp
. match ( a [ i ] ) ) sum
+= $ . parseNumber ( exp . getParen ( 0 ) )
; System . out . println ( "sum = " + sum
) ; }
```

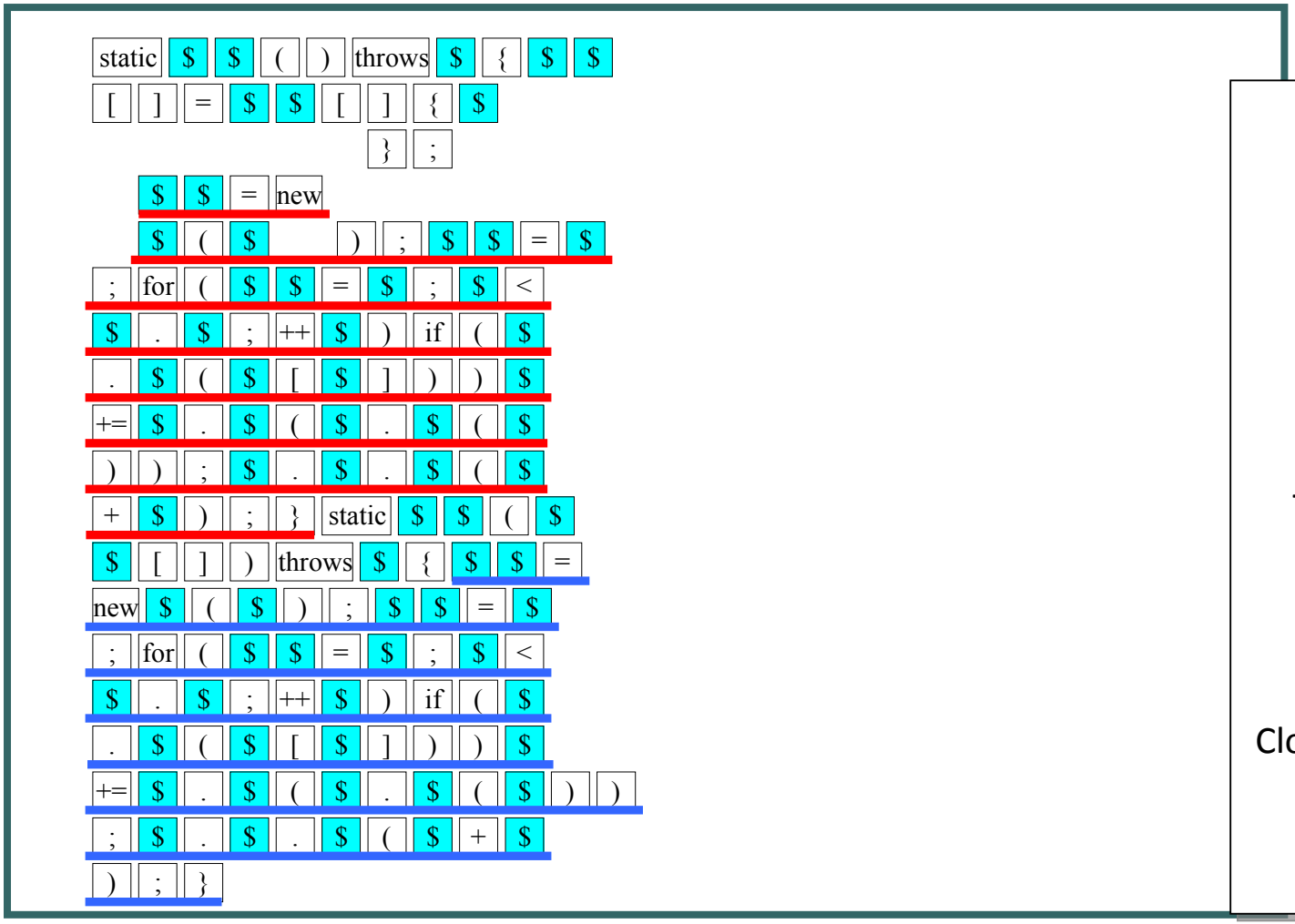


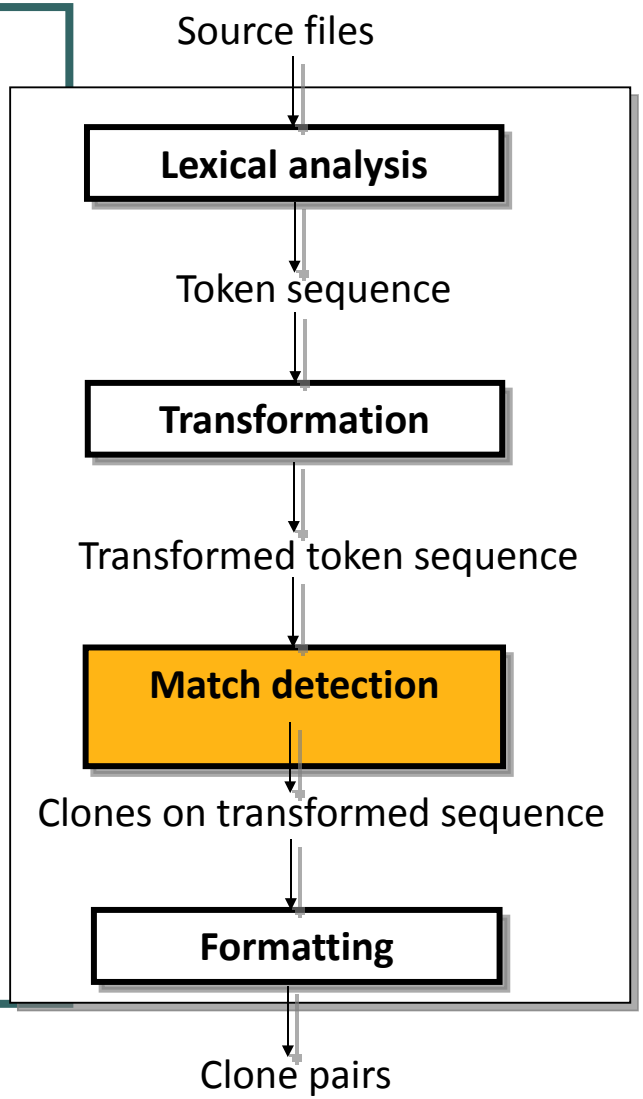
CCFinder - Detection





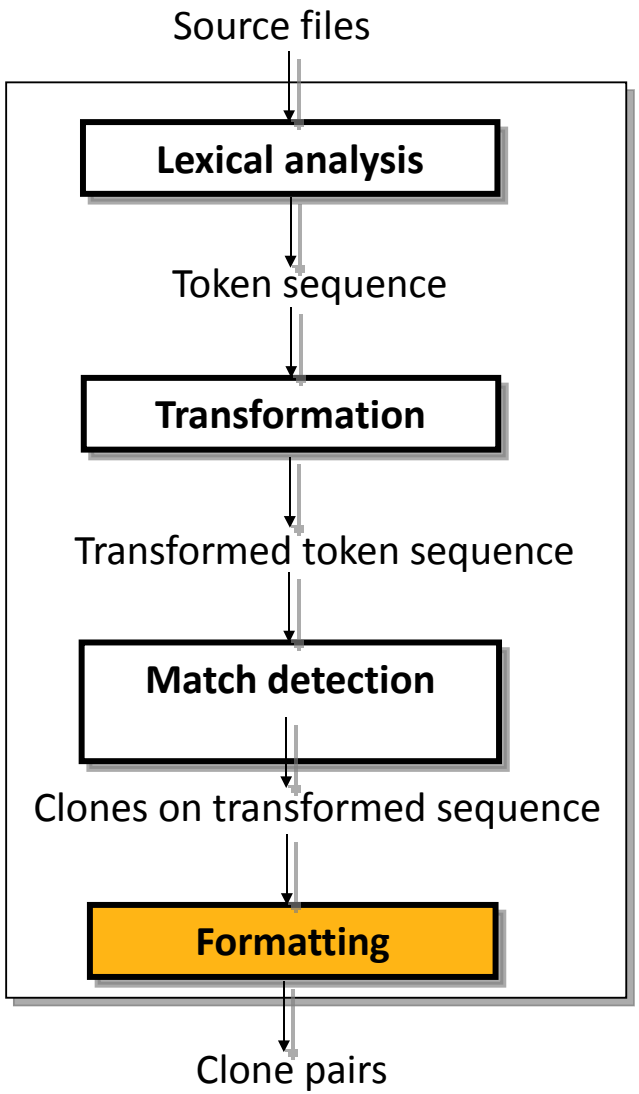
CCFinder - Detection





CCFinder-Detection

```
1. static void foo() throws RESyntaxException {
2.    String a[] = new String [] { "123,400", "abc", "orange 100" };
3.    org.apache.regexp.RE pat = new org.apache.regexp.RE("[0-9,]+");
4.    int sum = 0;
5.    for (int i = 0; i < a.length; ++i)
6.       if (pat.match(a[i]))
7.          sum += Sample.parseNumber(pat.getParen(0));
8.    System.out.println("sum = " + sum);
9. }
10. static void goo(String [] a) throws RESyntaxException {
11.    RE exp = new RE("[0-9,]+");
12.    int sum = 0;
13.    for (int i = 0; i < a.length; ++i)
14.       if (exp.match(a[i]))
15.          sum += parseNumber(exp.getParen(0));
16.    System.out.println("sum = " + sum);
17. }
```



Mega Software Engineering + Code Clone Detection

- Can code-clone detection be scaled to Mega Software Engineering level?
- Current code clone detection tools' scalability is good but limited by the underlying hardware

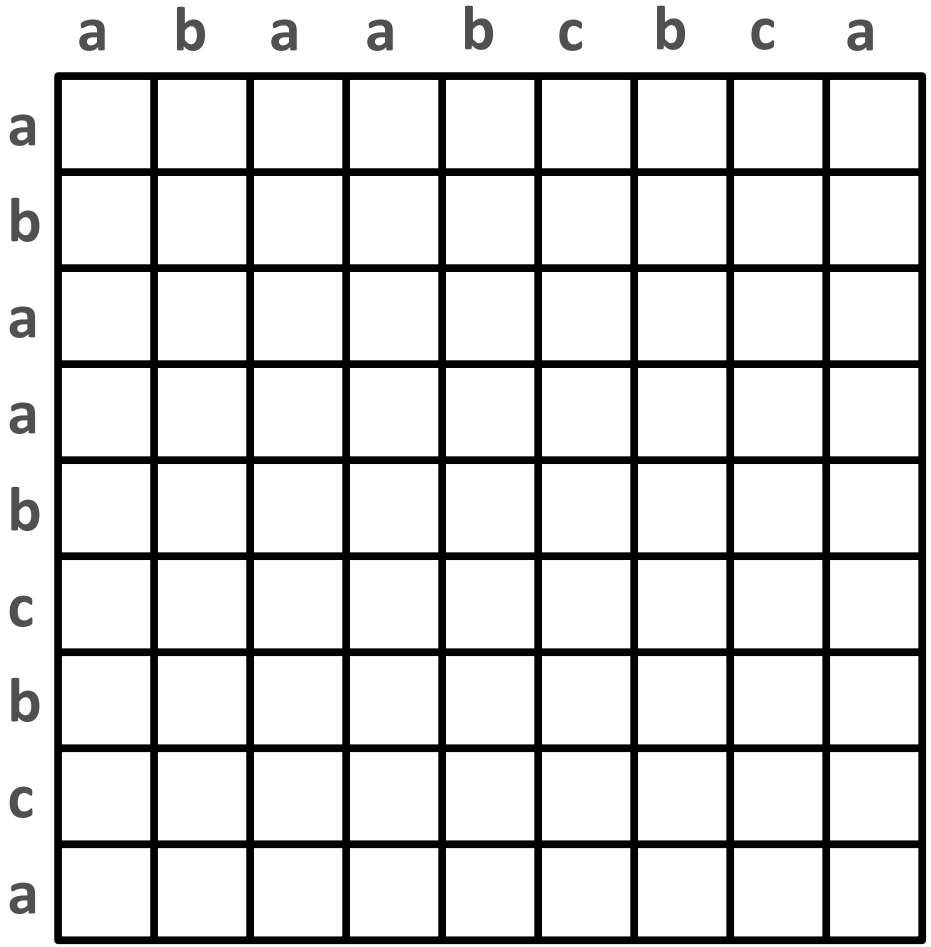
Mega Software Engineering + Code Clone Detection

- Can code-clone detection be scaled to Mega Software Engineering level?
- Current code clone detection tools' scalability is good but limited by the underlying hardware
- Code clone detection can be an **embarrassingly parallel problem**...
- ...and then easily adapted to a distributed computational model

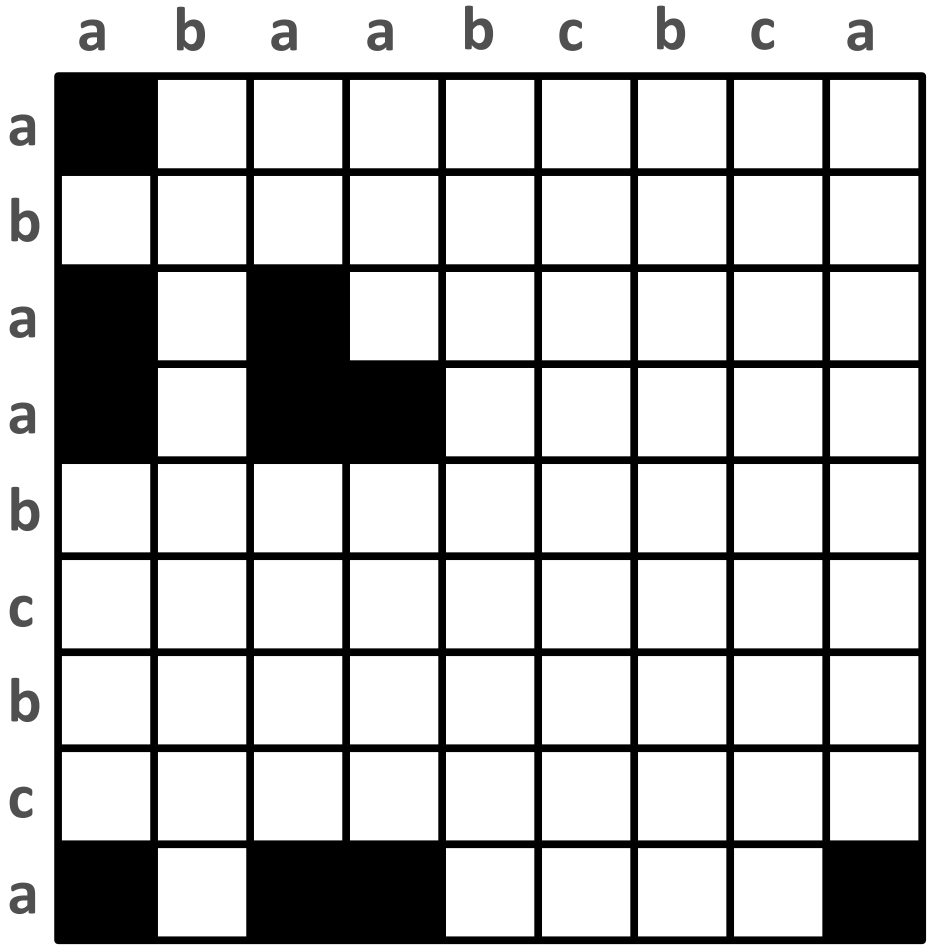
Embarrassingly Parallel Problem

A problem which can be segmented into a large number of parallel and independent tasks without any particular effort

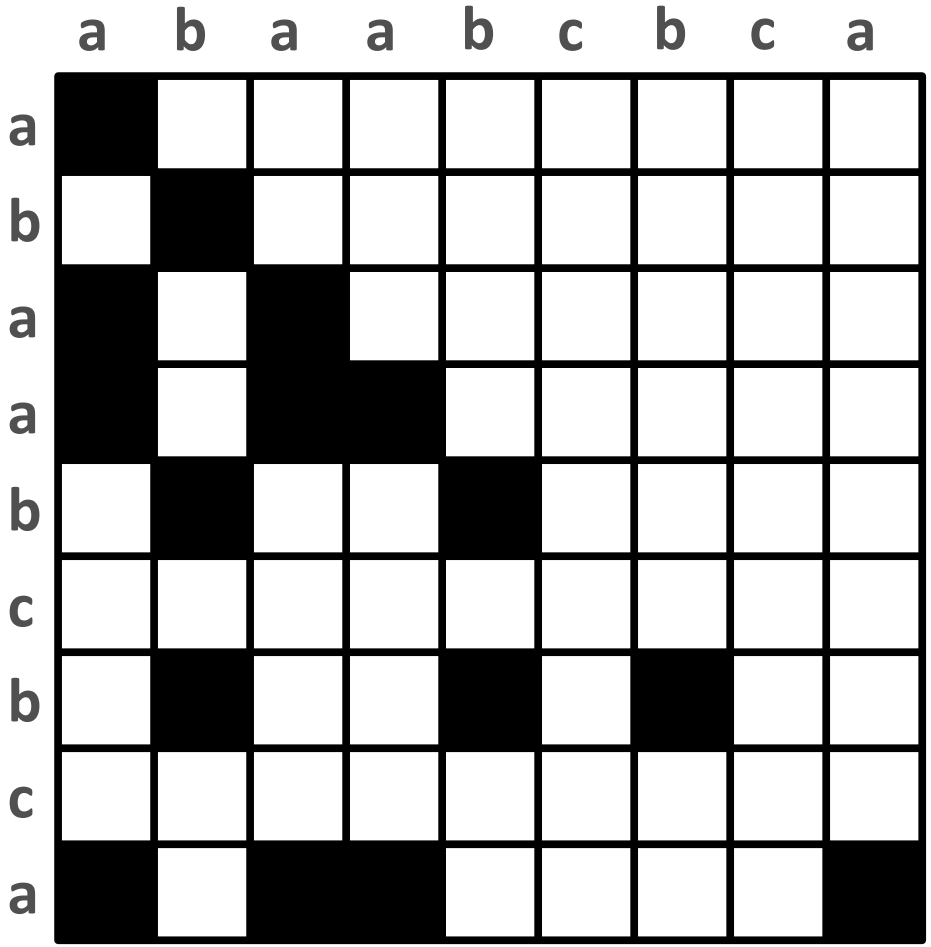
Embarrassingly Parallel Problem



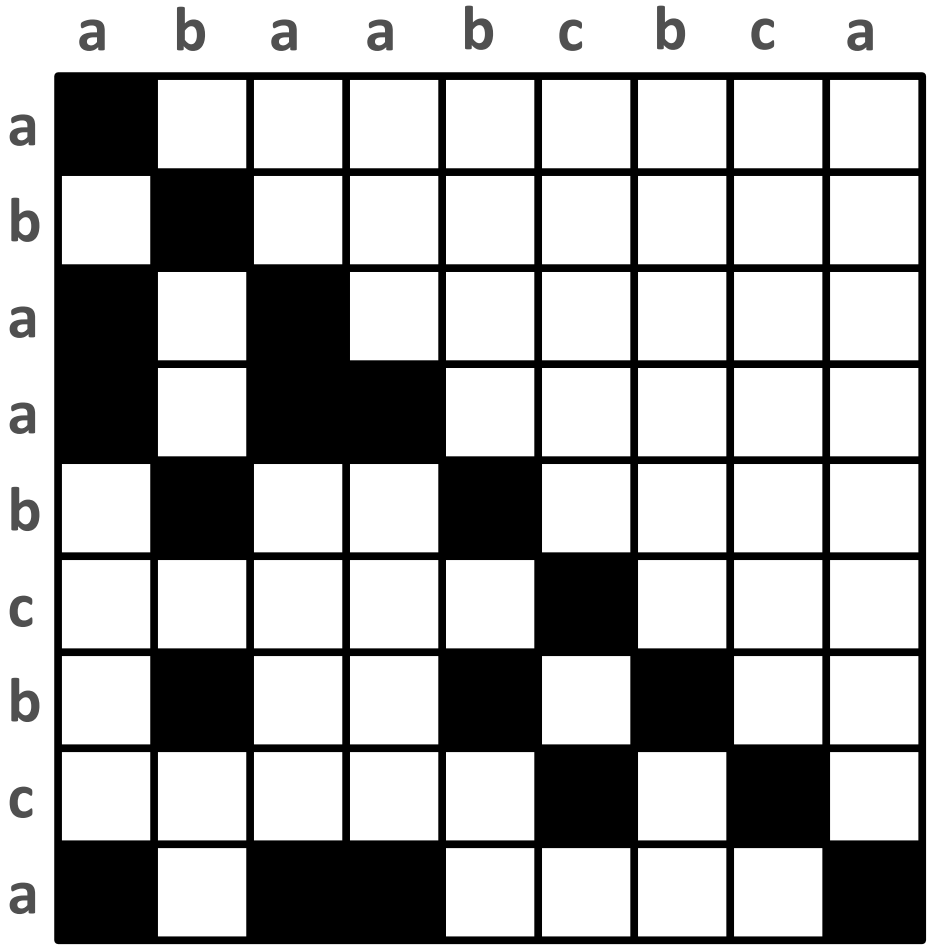
Embarrassingly Parallel Problem



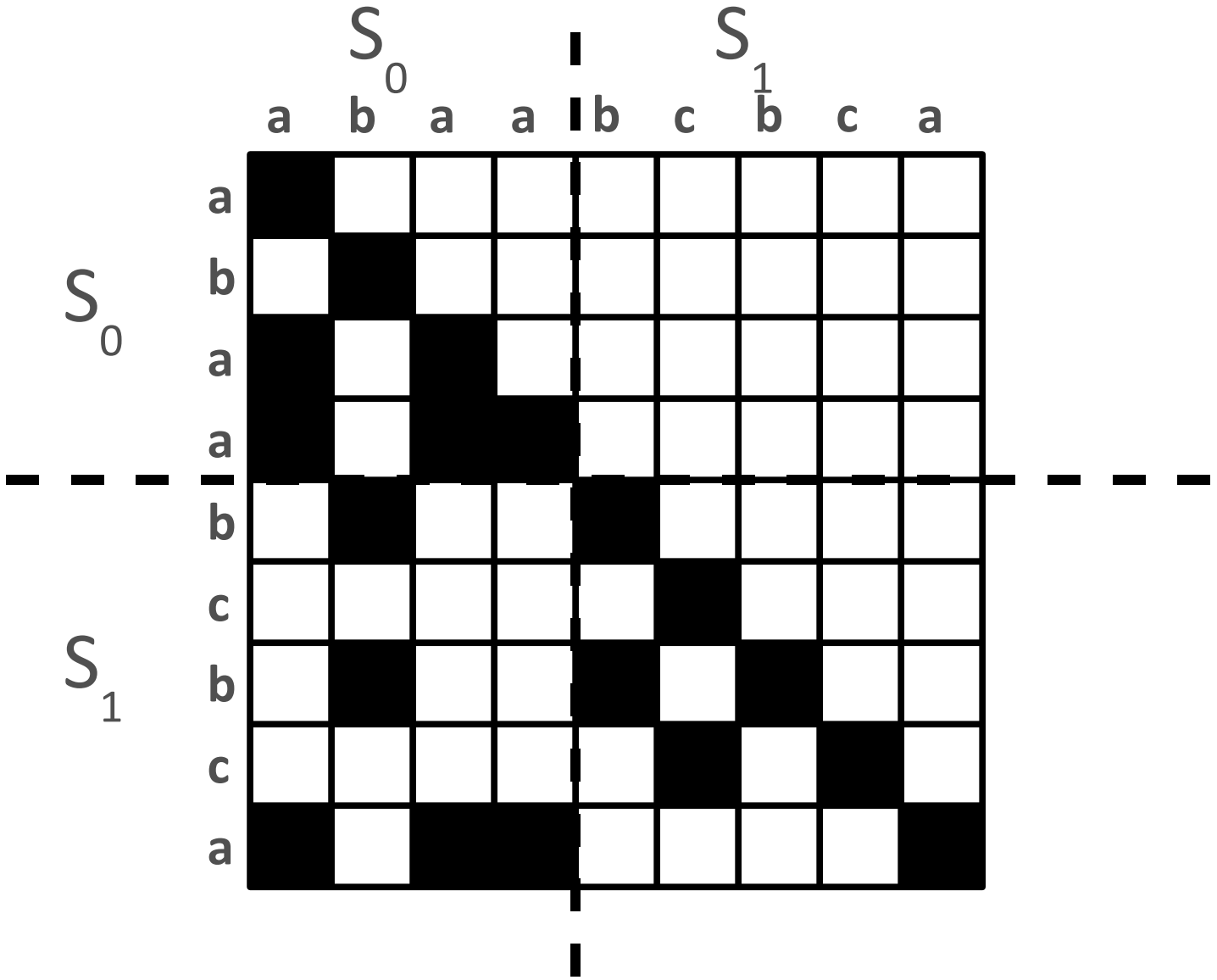
Embarrassingly Parallel Problem



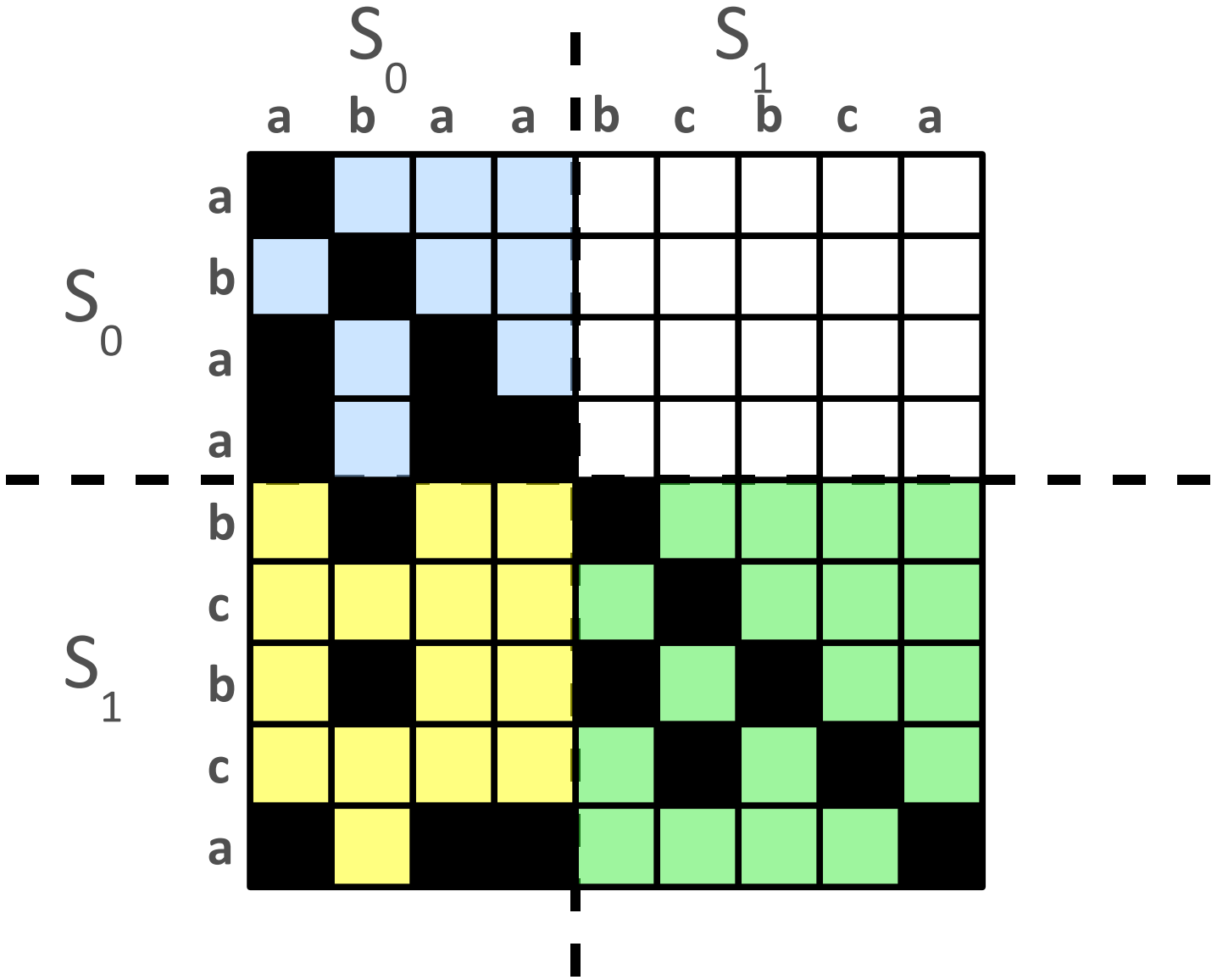
Embarrassingly Parallel Problem



Embarrassingly Parallel Problem



Embarrassingly Parallel Problem



Mega Software Engineering + Code Clone Detection

- Can code-clone detection be scaled to Mega Software Engineering level?
- Current code clone detection tools' scalability is good but limited by the underlying hardware
- Code clone detection can be an **embarrassingly parallel problem**...
- ...and then easily adapted to a distributed computational model

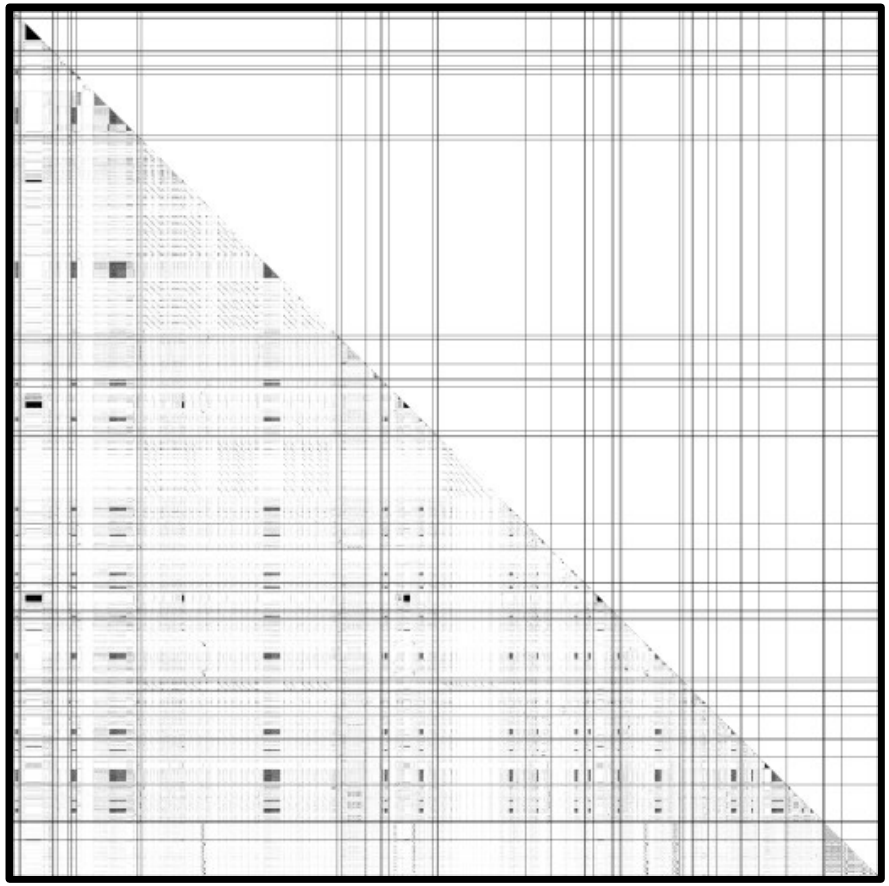
Distributed CCFinder

D-CCFinder

- A system for distributed code clone analysis
- Uses CCFinder as code clone detector

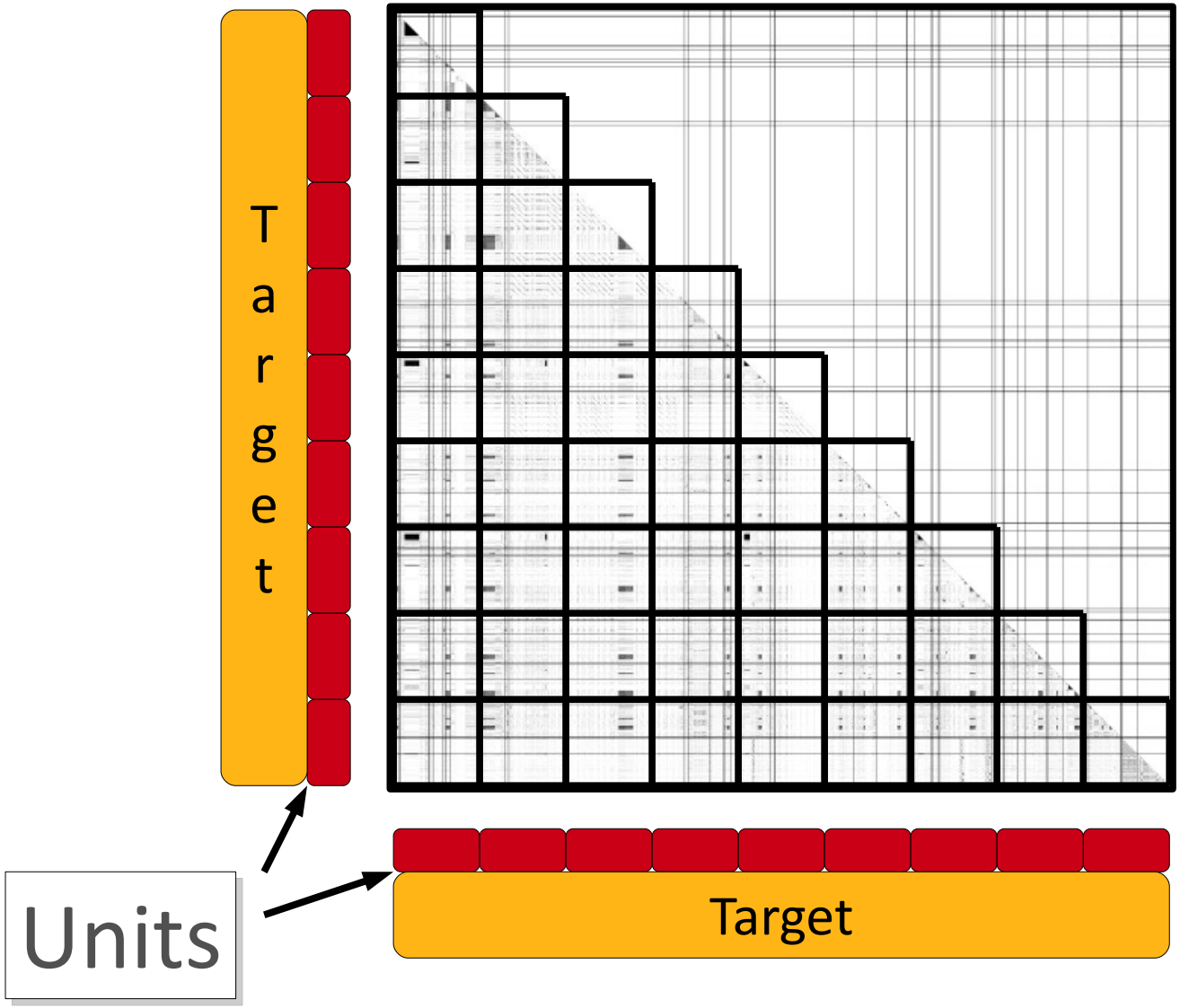
Getting Things Done

Target

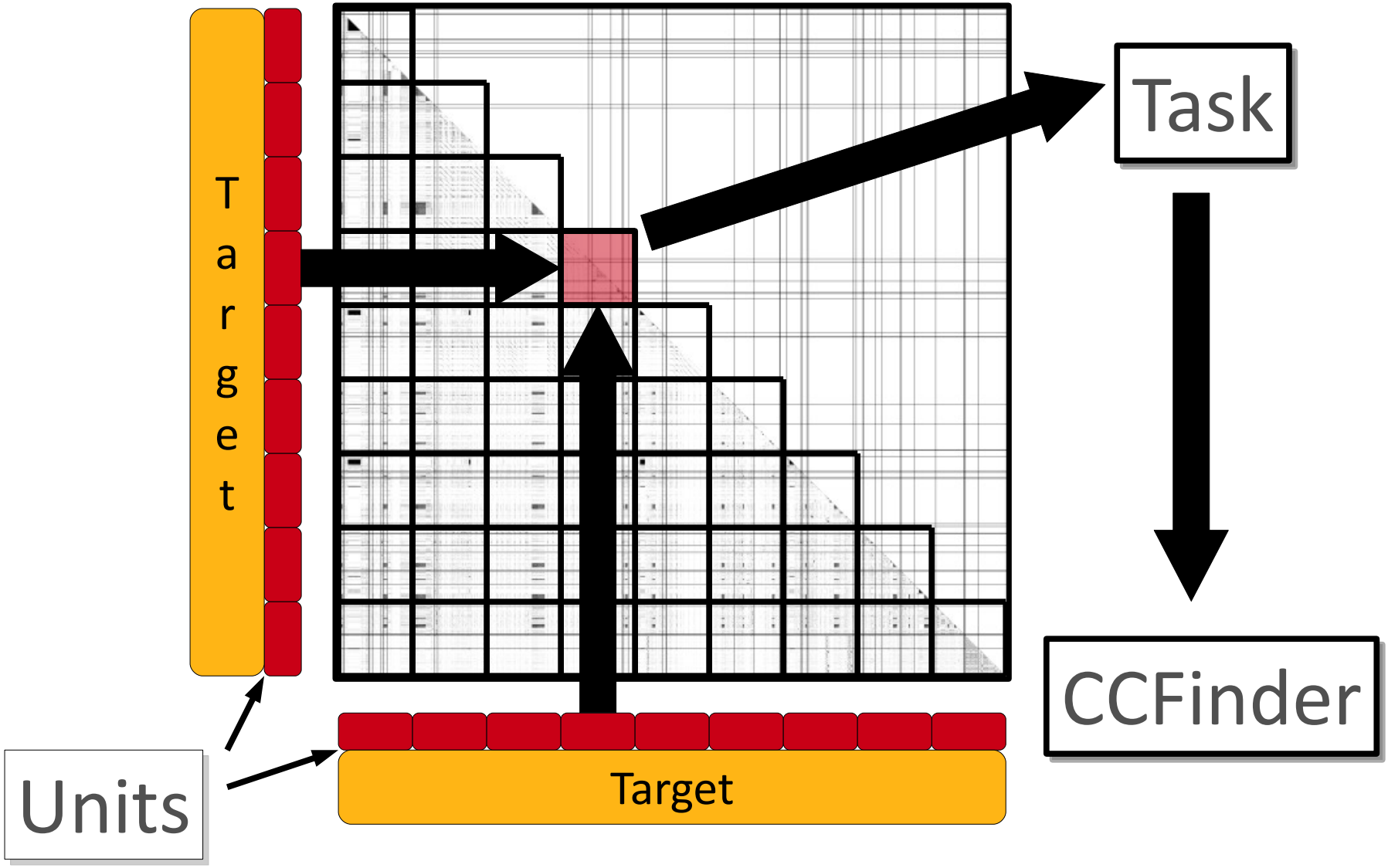


Target

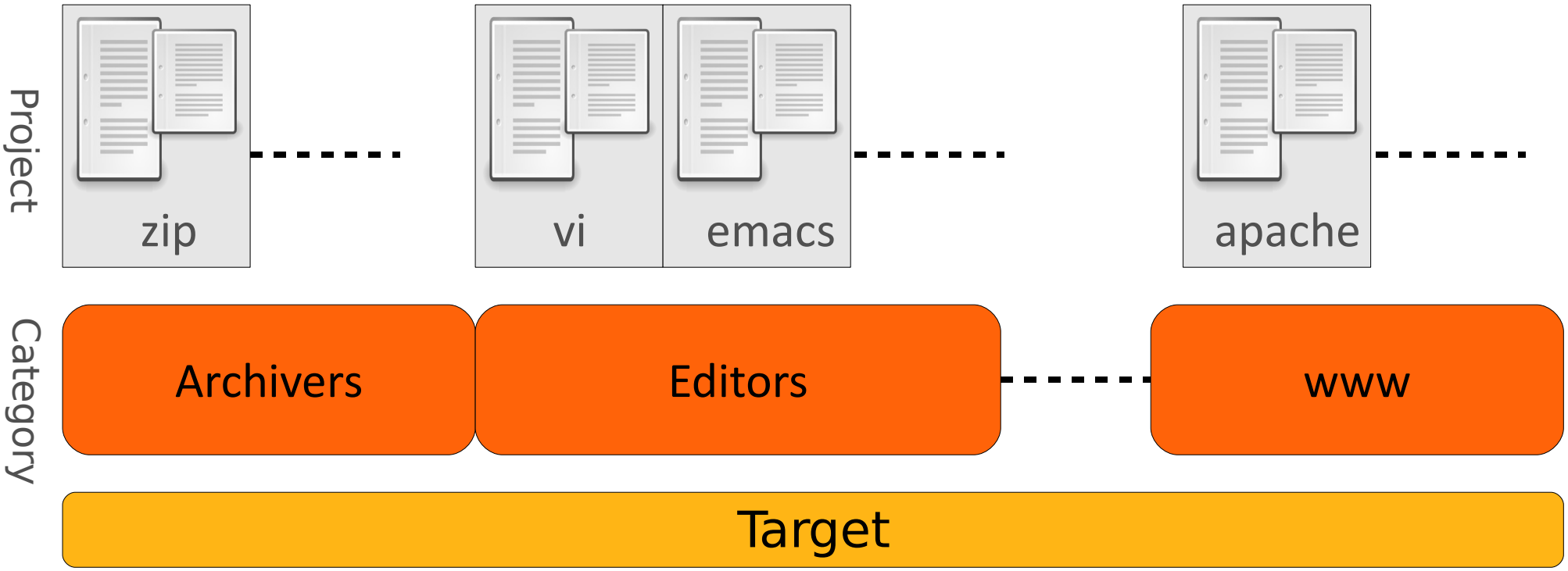
Getting Things Done



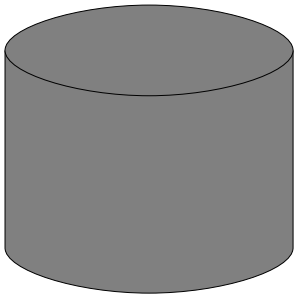
Getting Things Done



Target, Category and Project



Architecture



source code

Indexer

category

projects

units

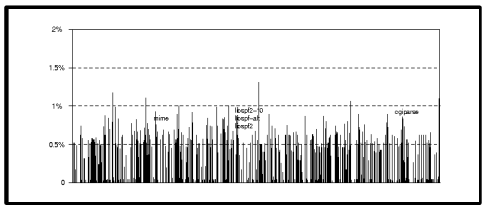
Code Clone Analysis

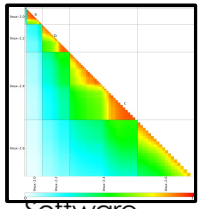
clone-pair data

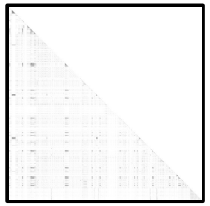
Clone Coverage Analyzer

clone coverage data

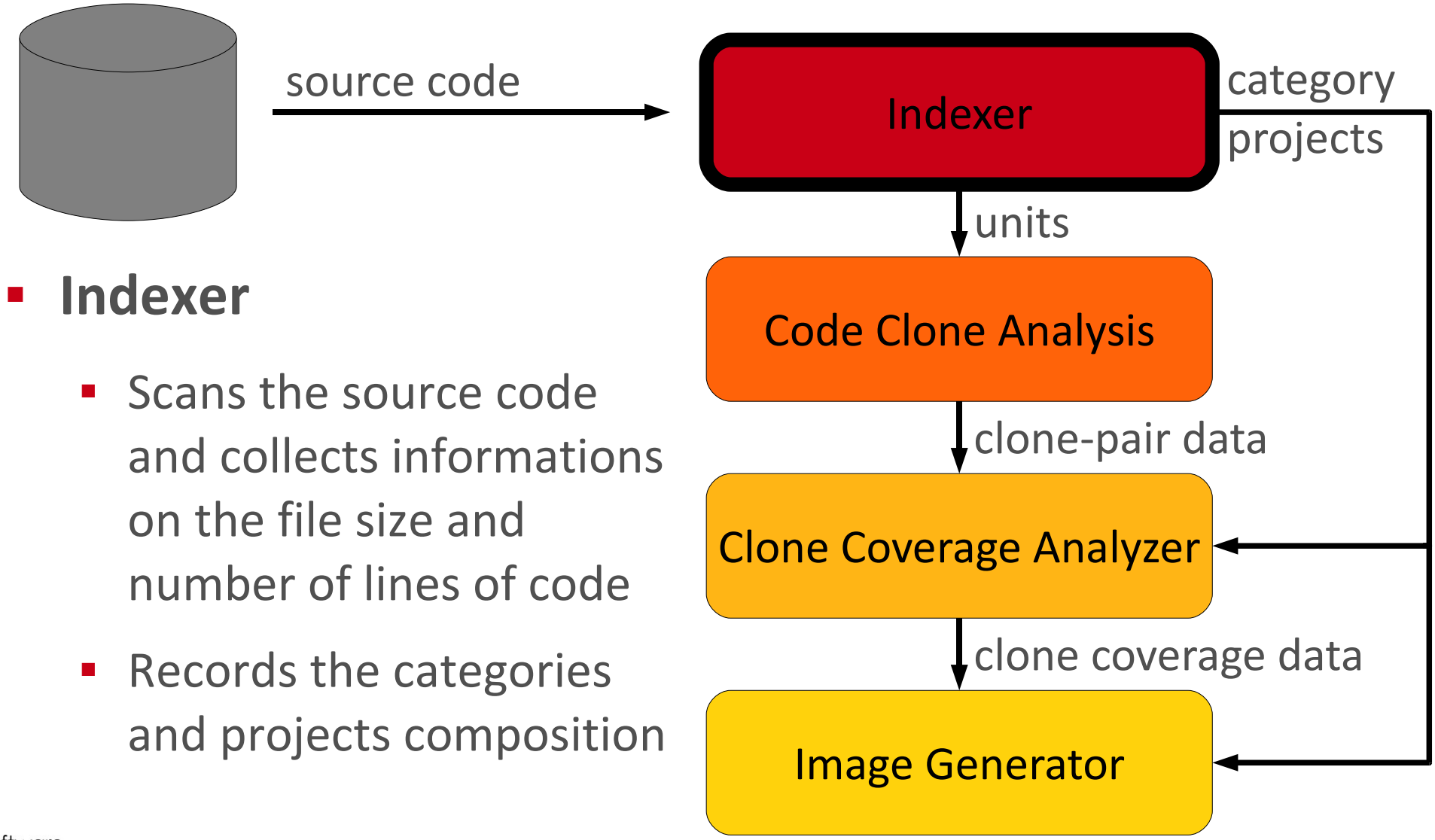
Image Generator





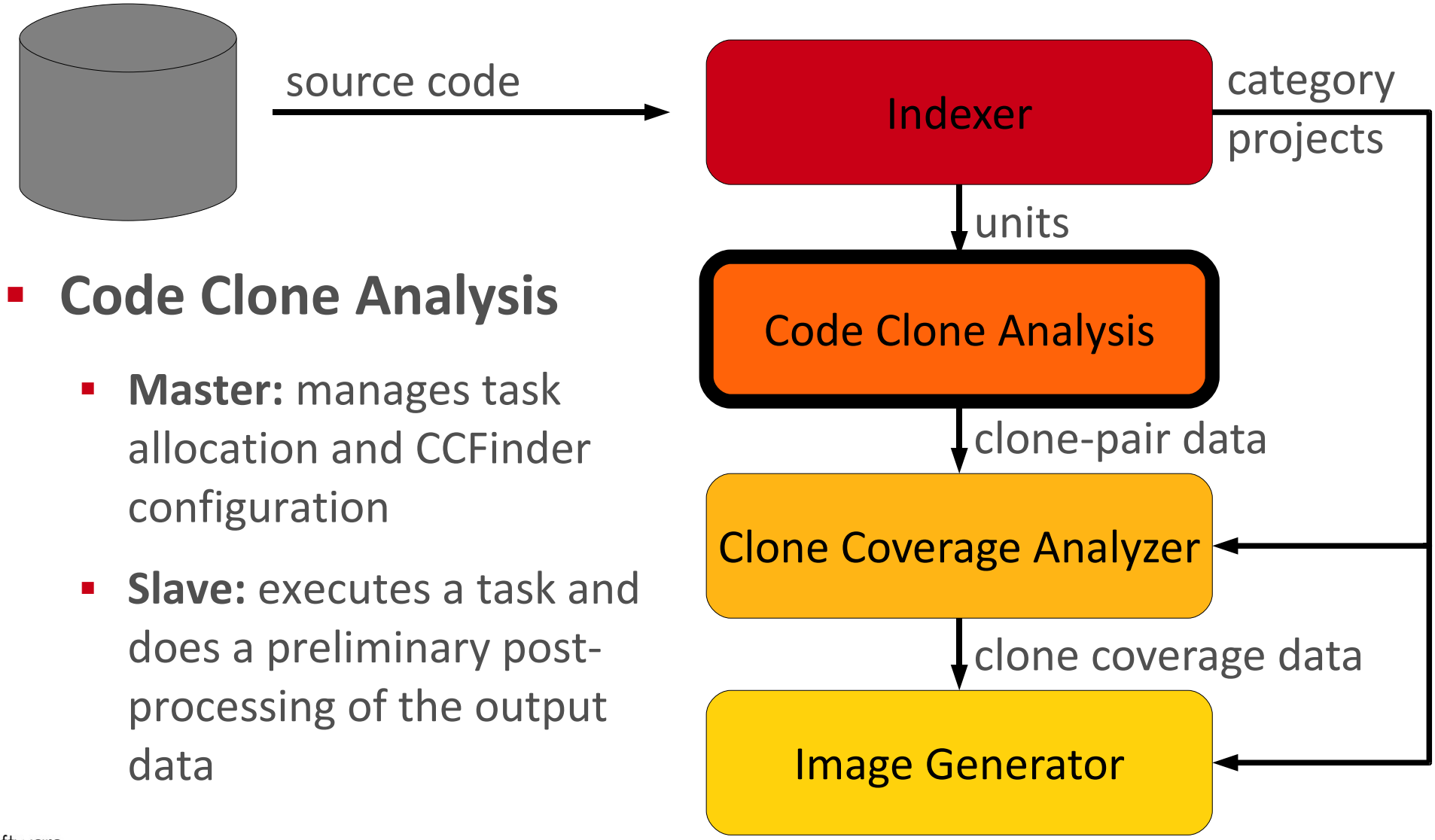


Architecture



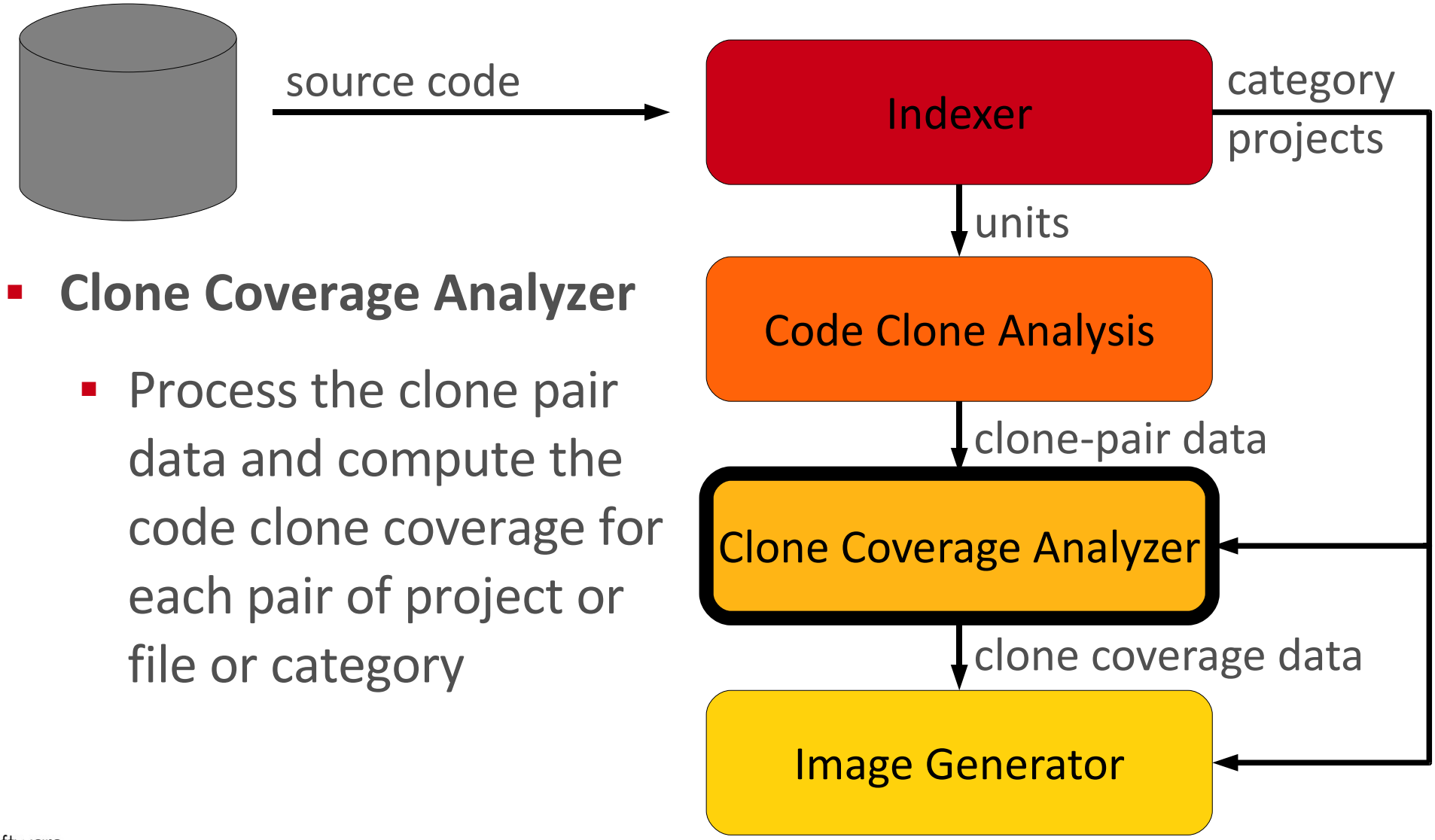
- **Indexer**
  - Scans the source code and collects informations on the file size and number of lines of code
  - Records the categories and projects composition

Architecture



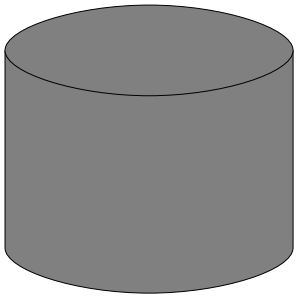
- **Code Clone Analysis**
  - **Master:** manages task allocation and CCFinder configuration
  - **Slave:** executes a task and does a preliminary post-processing of the output data

Architecture



- **Clone Coverage Analyzer**
  - Process the clone pair data and compute the code clone coverage for each pair of project or file or category

Architecture



source code

Indexer

category projects

units

Code Clone Analysis

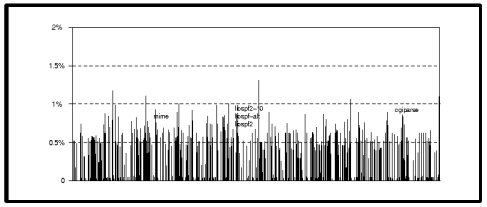
clone-pair data

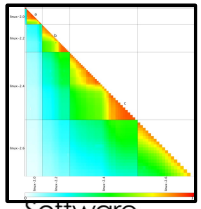
Clone Coverage Analyzer

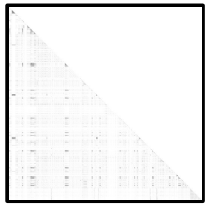
clone coverage data

Image Generator

- **Image Generator**
  - Generates dot plot or bar chart or heat map







Implementation/Setup

- **Implementation**
  - Written in Java
  - About 20kLOC
  - Network shared file system for data I/O
- **Setup**
  - 81 computers (FreeBSD, Java 1.5) in the department's student lab
    - Master:1 Slaves: 80
  - Pentium IV 3GHz with 1Gb RAM

Experiments

Experiment I
the FreeBSD's ports

FreeBSD's Ports

- Vast collection of Open-Source Software

# Categories	45
# Projects	6658
# .c files	~755K
Total LOC	~404M
Total Size	10.8 Gbytes

- Unit size: 15Mbytes (734 units, 267,745 tasks)
- Generated a dot-plot and an heatmap

FreeBSD' Ports - time

	Time Elapsed
Indexing	22 m
Code Clone Detection	51 h
Dot-plot	
Image Generation	4 h
Heatmap	
Clone Coverage	70 h
Image Generation	2 m

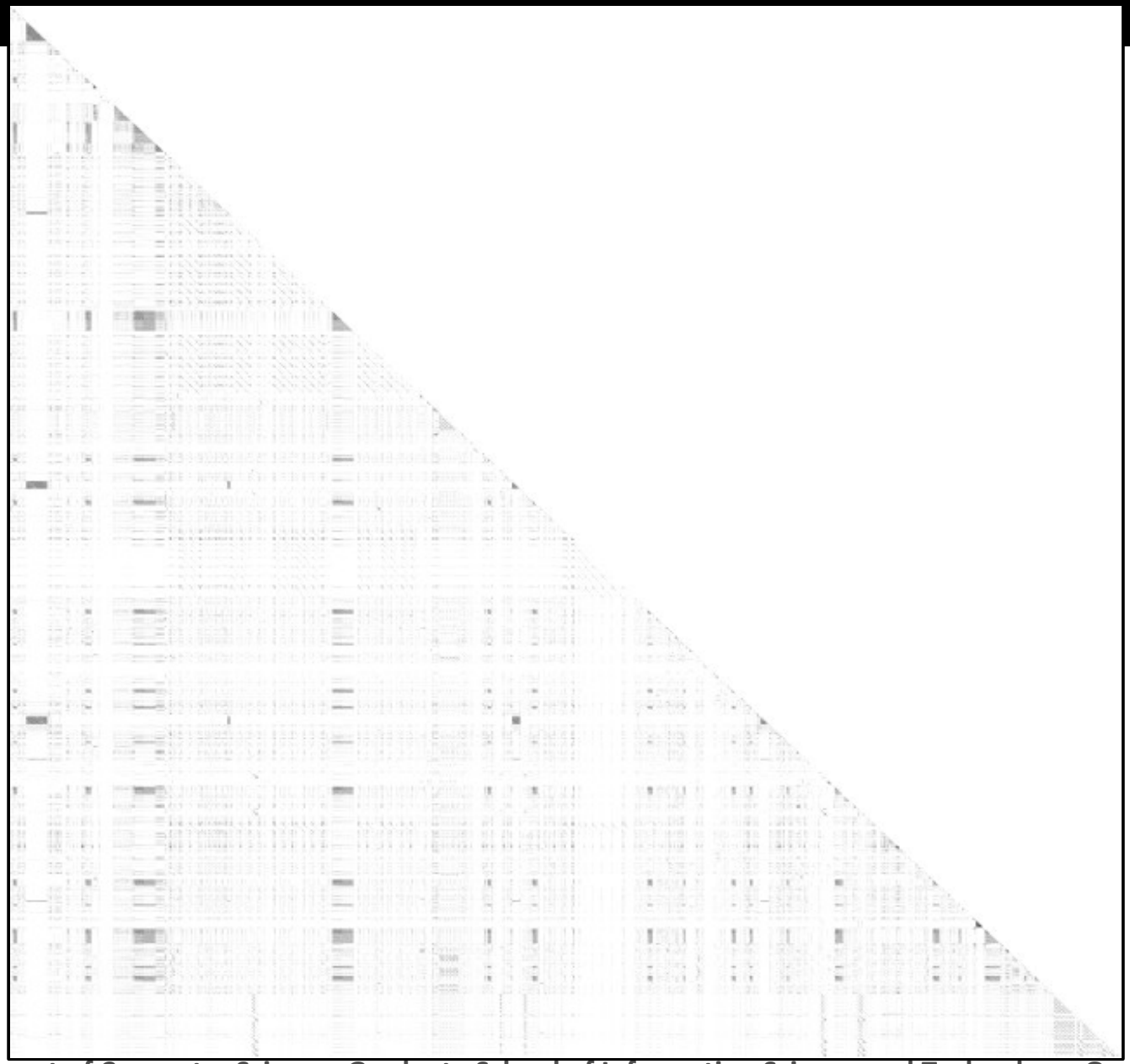
FreeBSD' Ports - time

	Time Elapsed
Indexing	22 m
Code Clone Detection	51 h
Dot-plot	
Image Generation	4 h
Heatmap	
Clone Coverage	70 h
Image Generation	2 m

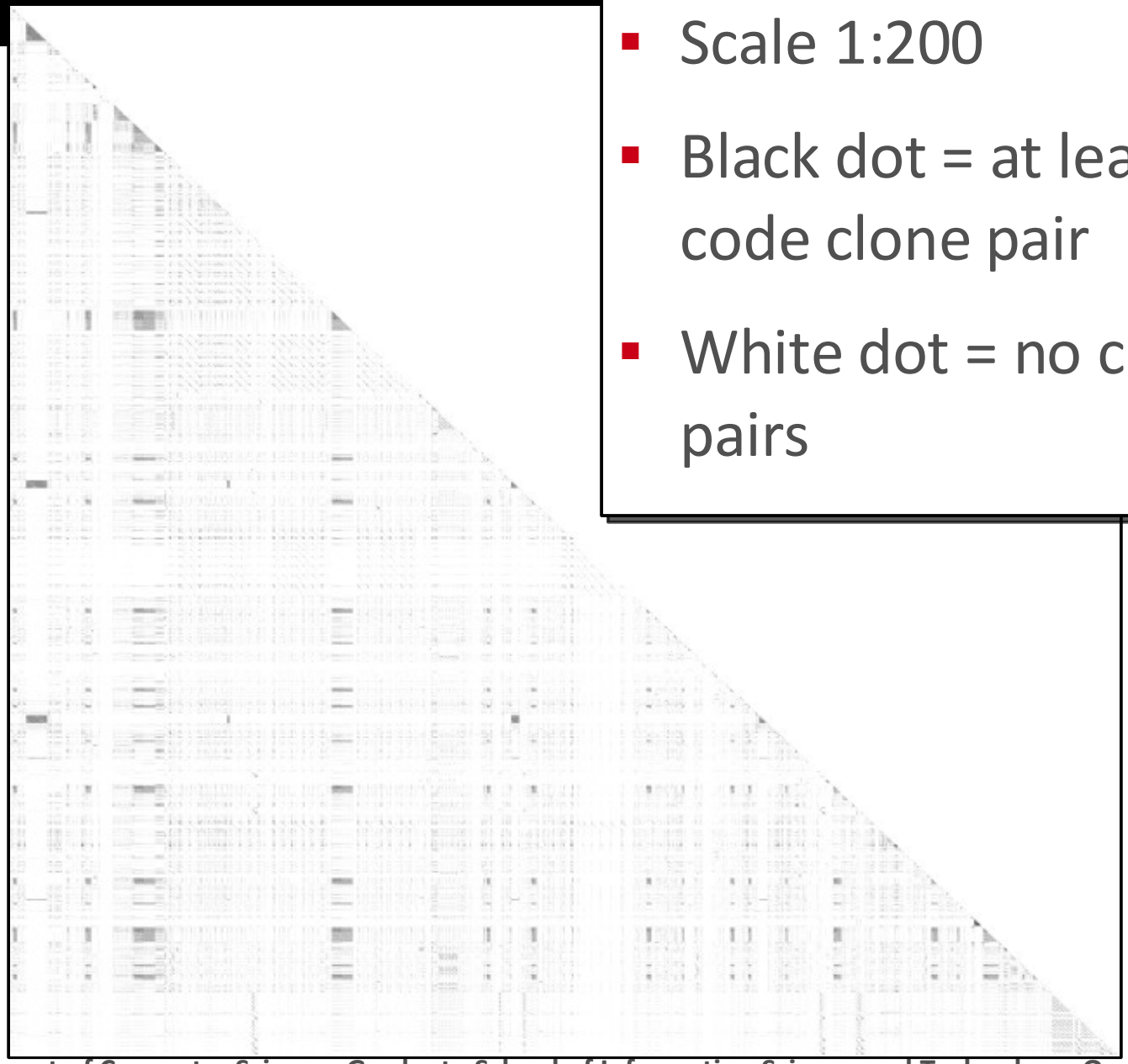
How much time would it take if...

- Using a **single computer**, processing 400MLOC requires to run CCFinder about **3200** times for a total of about **40** days
- In the **ideal case** running CCFinder 3200 times with 80 computers should require about **12** hours
- D-CCFinder took **51** hours
  - **20** times faster than using a single computer
  - **4** times slower than the ideal case with 80 computers

FreeBSD's ports - results

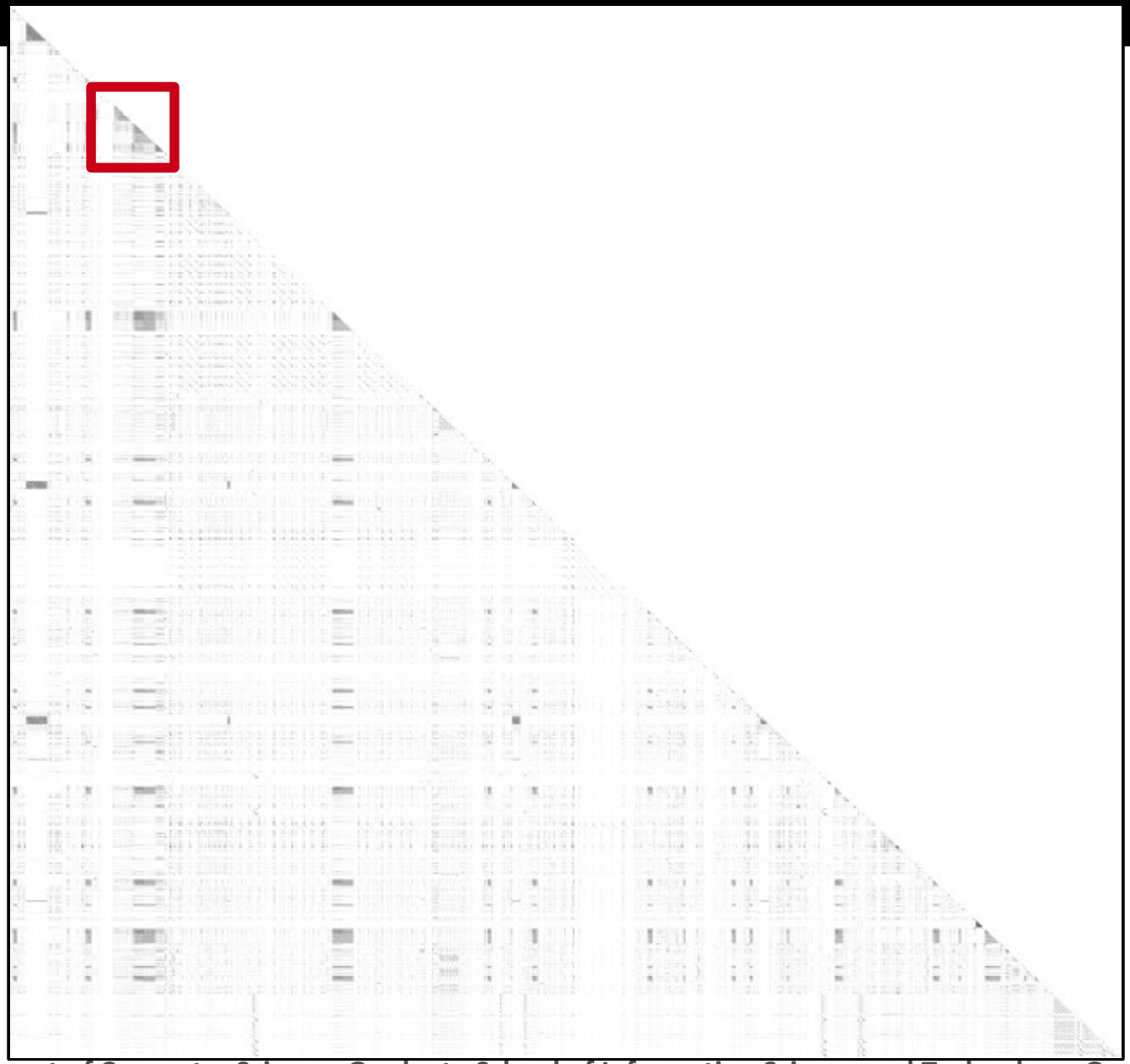


FreeBSD's ports - results



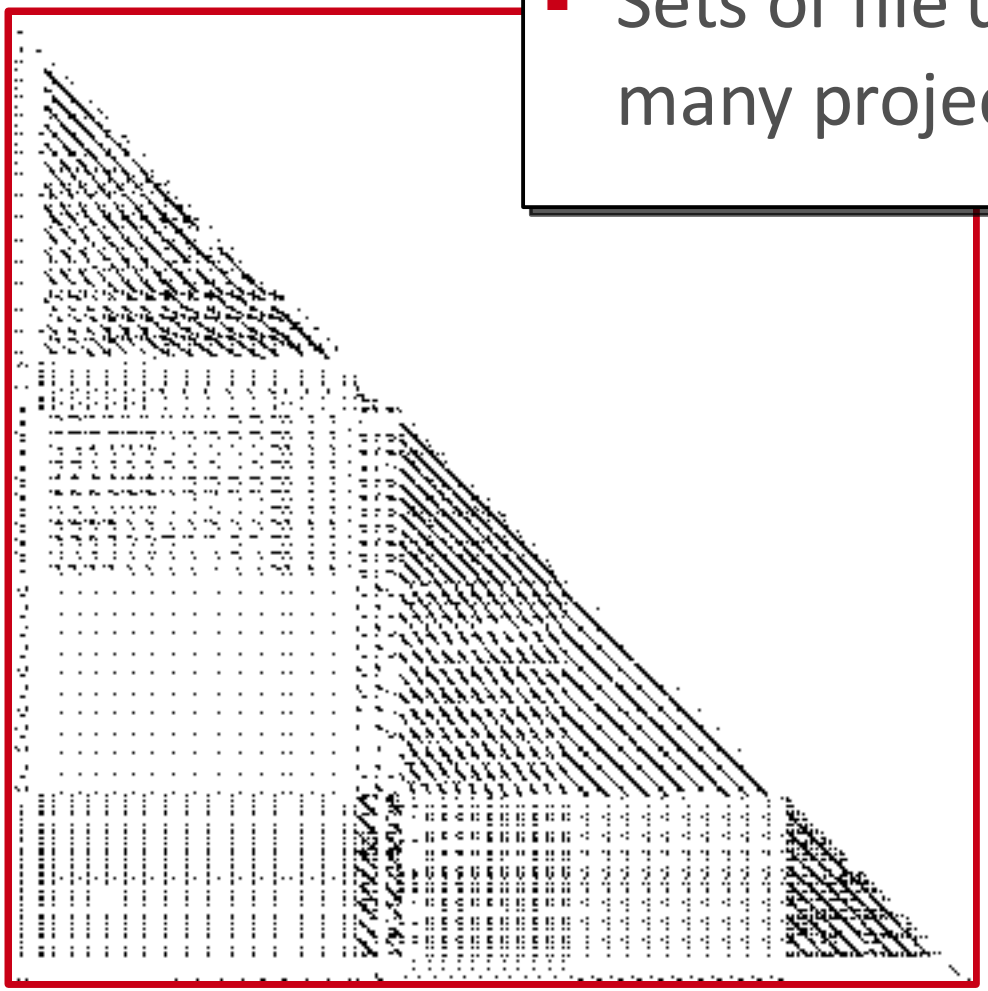
- Scale 1:200
- Black dot = at least one code clone pair
- White dot = no clone pairs

FreeBSD's ports - results

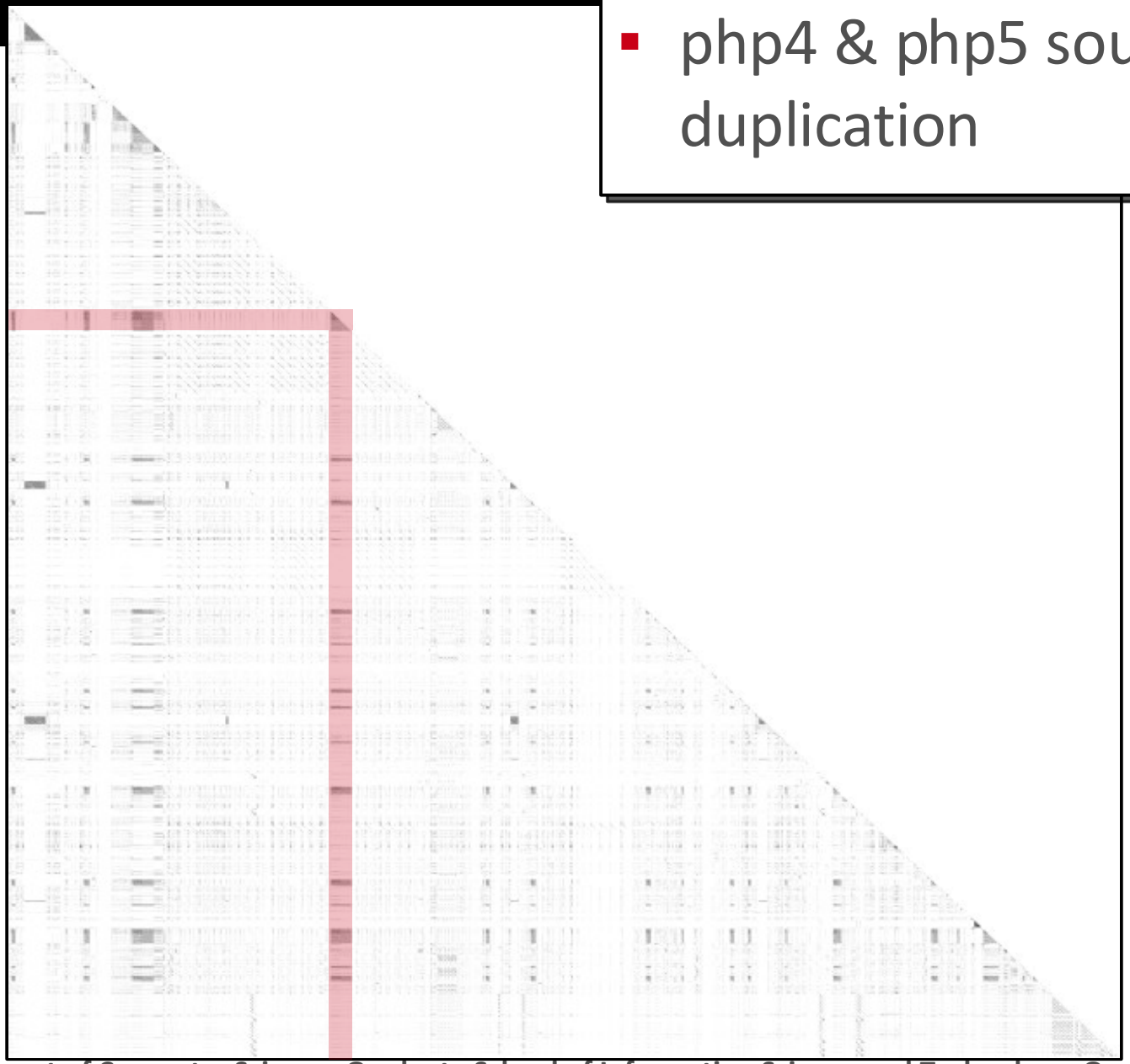


FreeBSD's ports - results

- Common pattern
- Sets of file unchanged in many projects



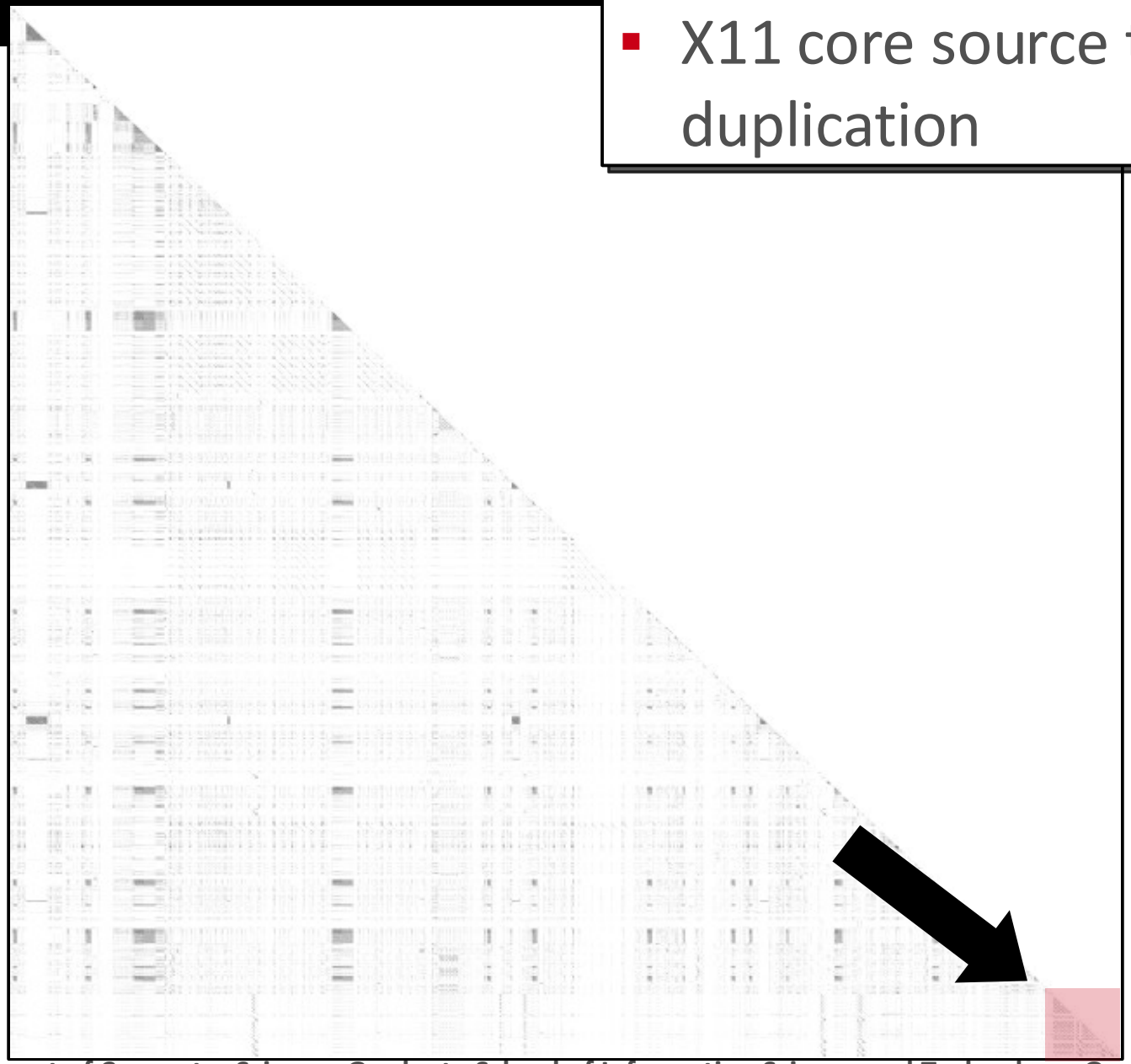
FreeBSD's ports - results



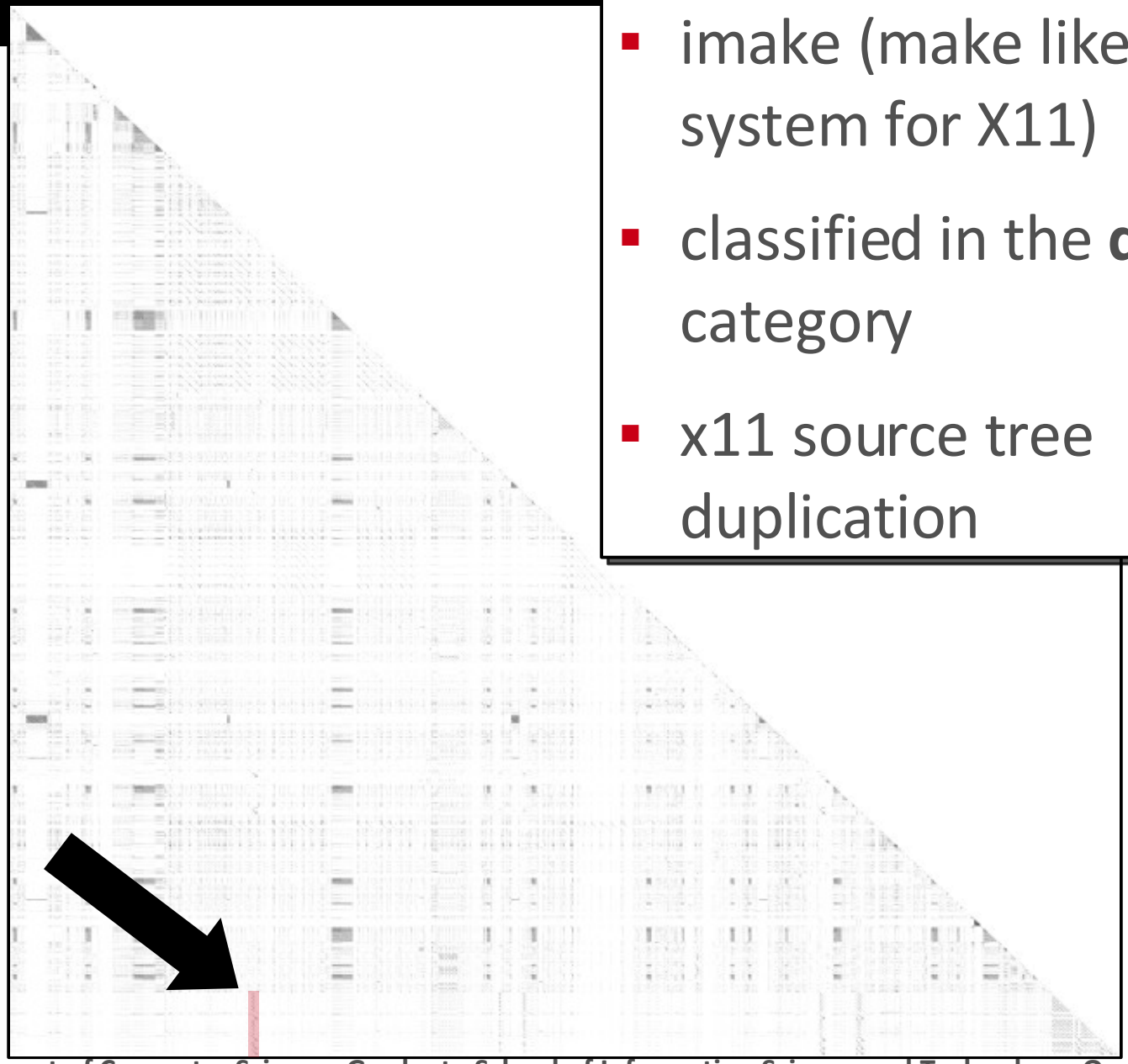
- php4 & php5 source tree duplication

FreeBSD's ports - results

- X11 core source tree duplication

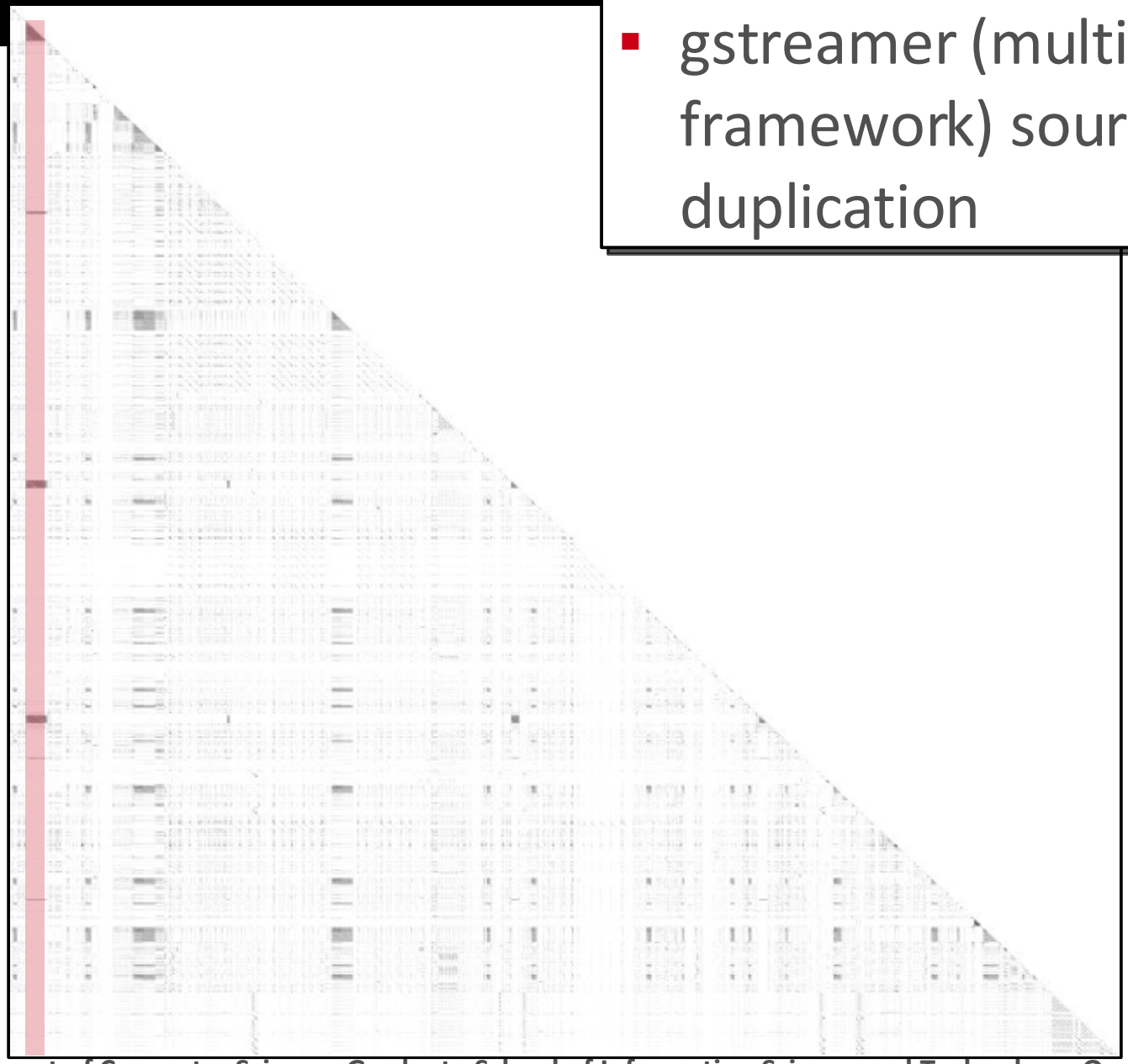


FreeBSD's ports - results



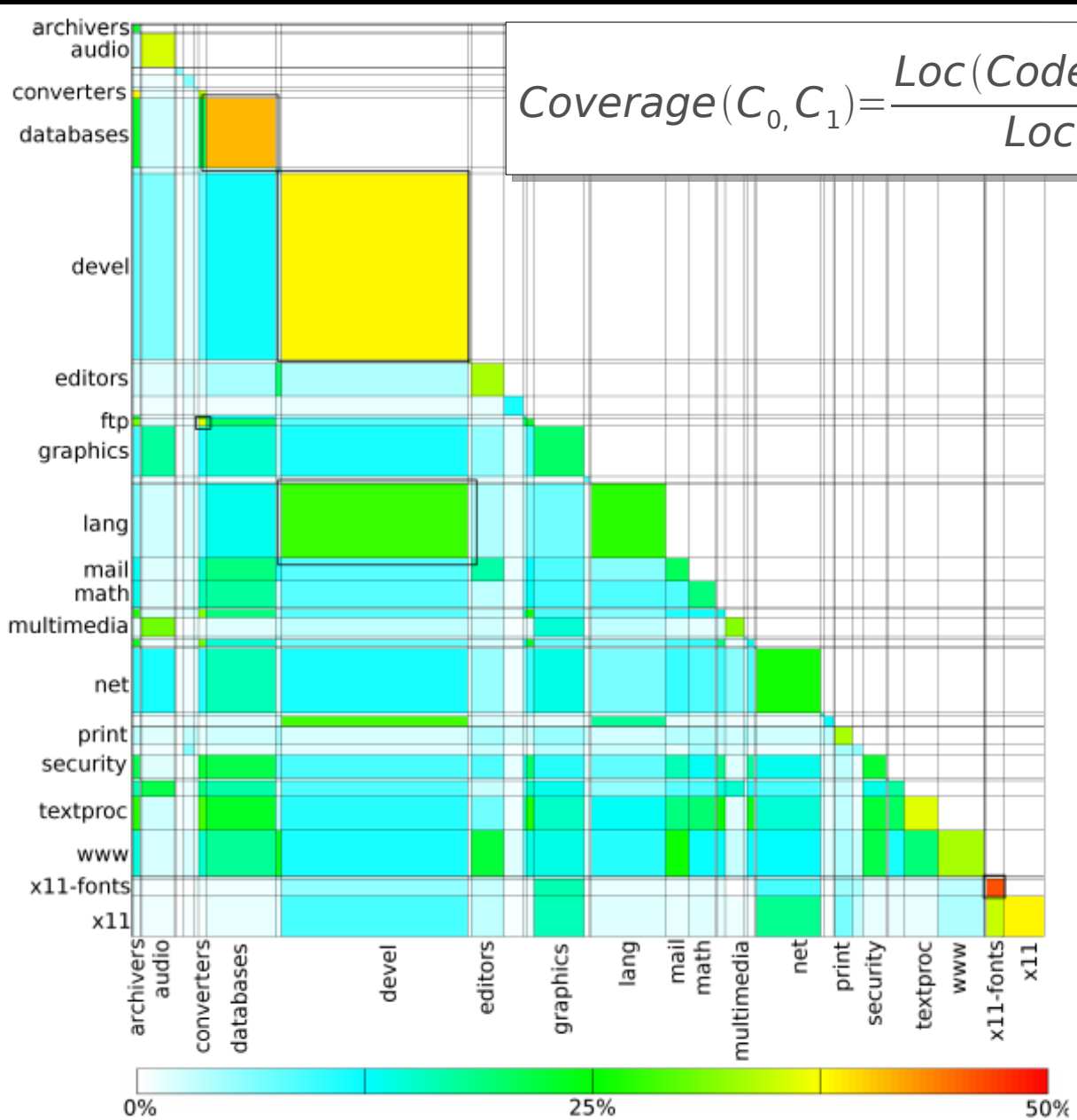
- imake (make like building system for X11)
- classified in the **devel** category
- x11 source tree duplication

FreeBSD's ports - results



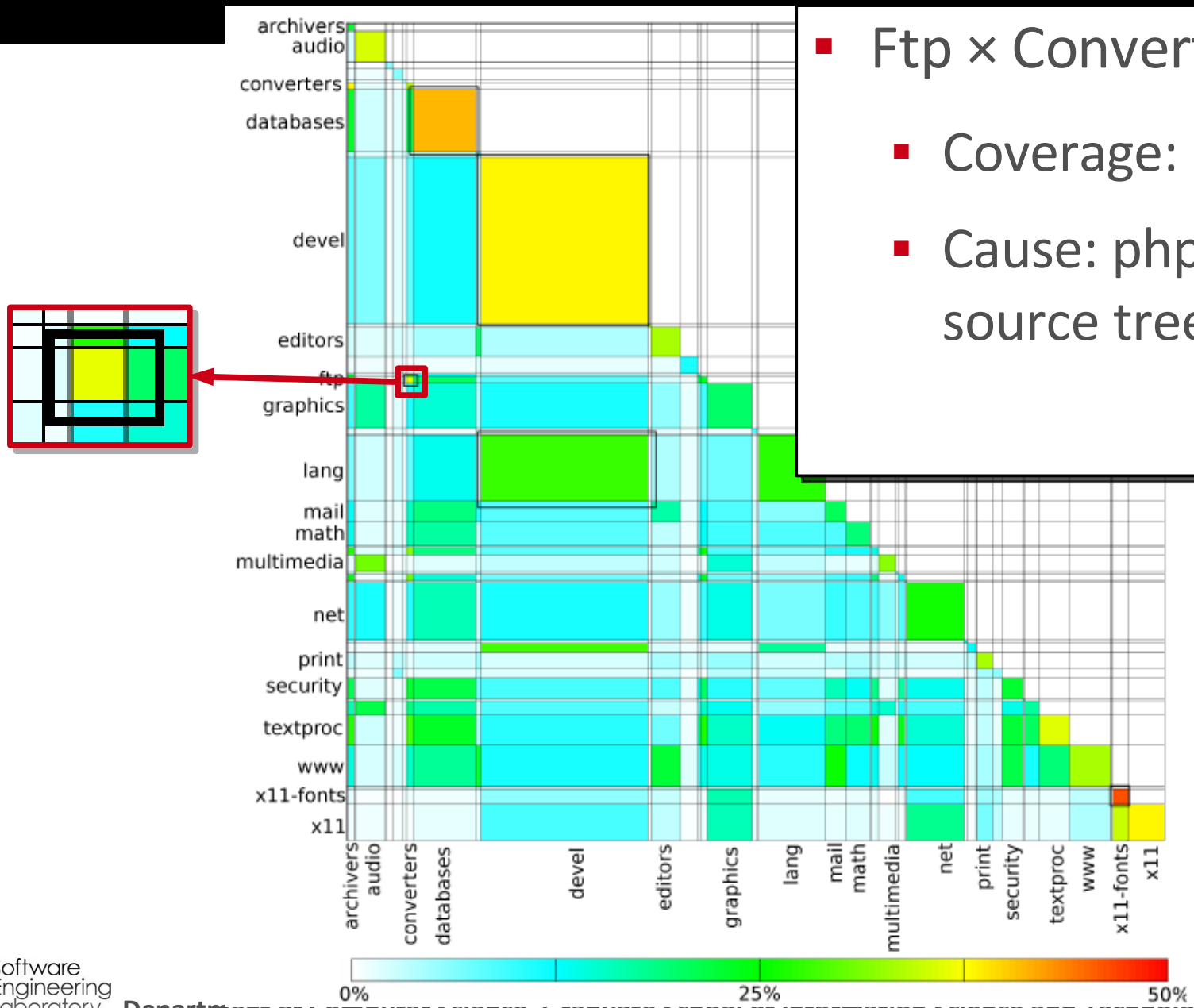
- gstreamer (multimedia framework) source tree duplication

FreeBSD's ports - results



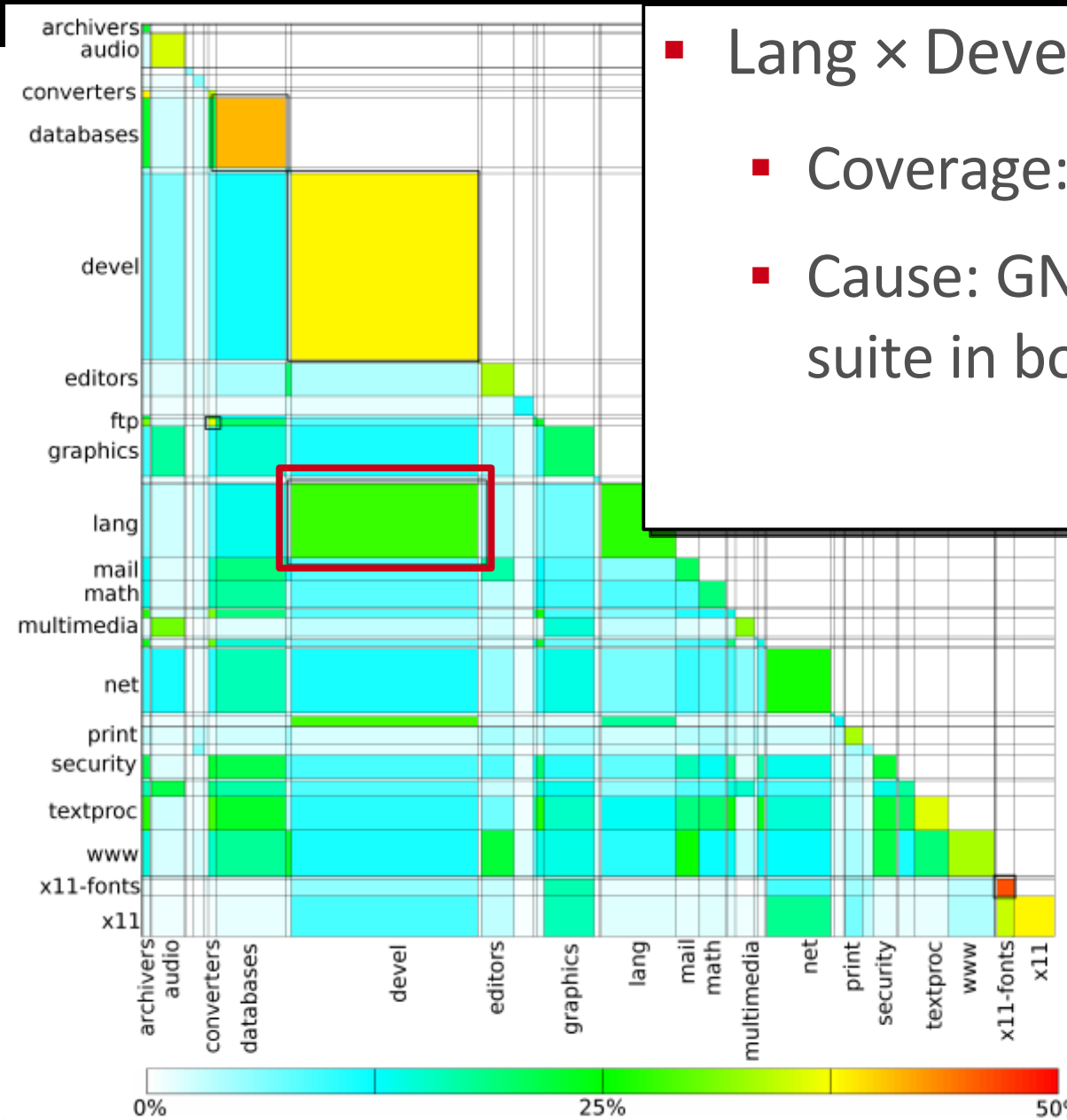
$$Coverage(C_{0,}C_1)=\frac{Loc(CodeSegments(C_{0,}C_1))}{Loc(C_0)+Loc(C_1)}$$

FreeBSD's ports - results



- Ftp × Converters
  - Coverage: 37%
  - Cause: php4 and php5 source trees

FreeBSD's ports - results



- Lang × Devel
  - Coverage: 28%
  - Cause: GNU compiler suite in both categories

Experiment II
SPARS-J & the FreeBSD's ports

SPARS-J

- A Java component analysis and search tools developed at the Software Engineering Laboratory of the Osaka University
- Written in C language
- About 47,000 lines of code

Experiment II - Rationale

- Increased availability of Open-Source Software for different purposes
- Increased probability of including Open-Source Software's source code into a new system
- Increased risk of violating the terms of Open-Source Licenses

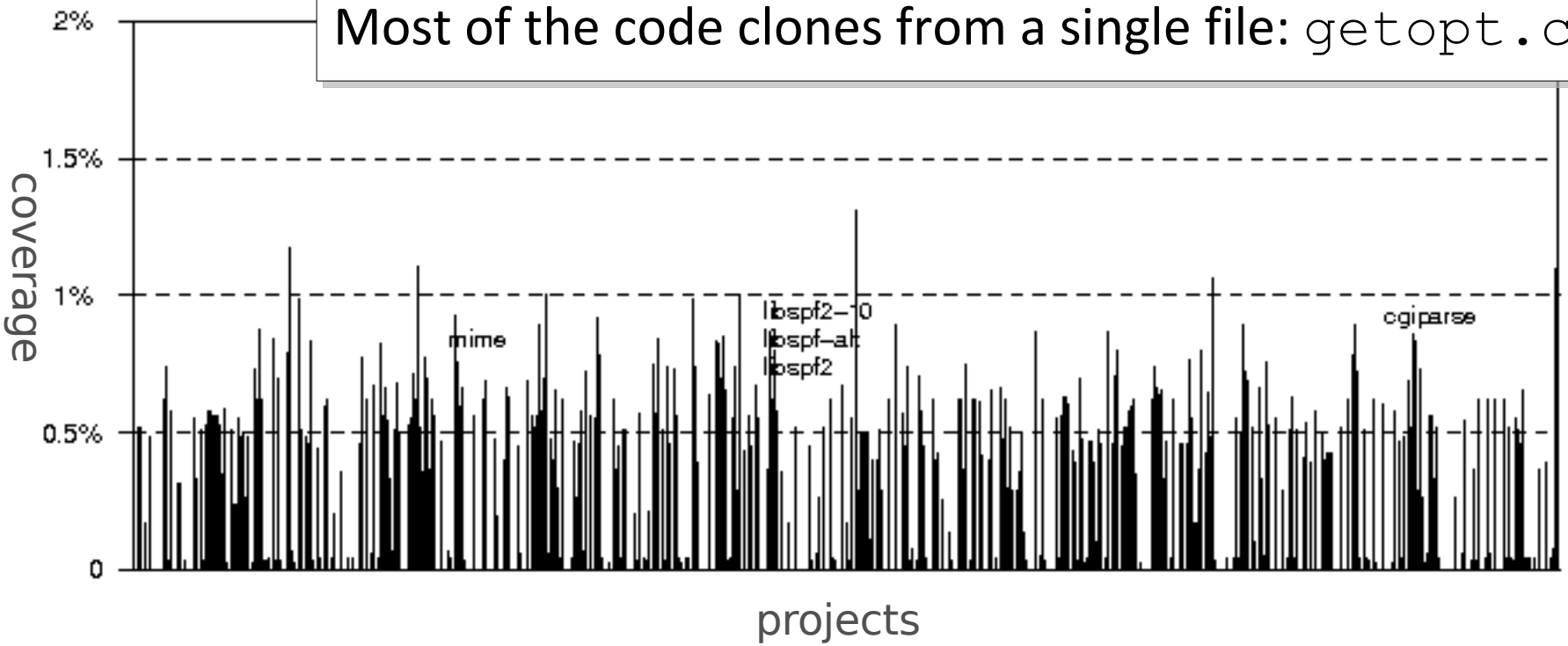
SPARS-J & FreeBSD's ports

- **Goal**
  - Investigate if SPARS-J contains source code in the FreeBSD's ports repository
- **Method**
  - Run D-CCFinder with each task made of the source code of SPARS-J and one unit of the FreeBSD's ports (734 tasks, 40 minutes)
  - Computed the coverage between SPARS-J and each single project in the FreeBSD's ports

SPARS-J & FreeBSD's ports Results

$$Coverage_{SPARSJ}(P) = \frac{Loc(CodeSegments(SPARSJ, P) \cap SPARSJ)}{Loc(SPARSJ)}$$

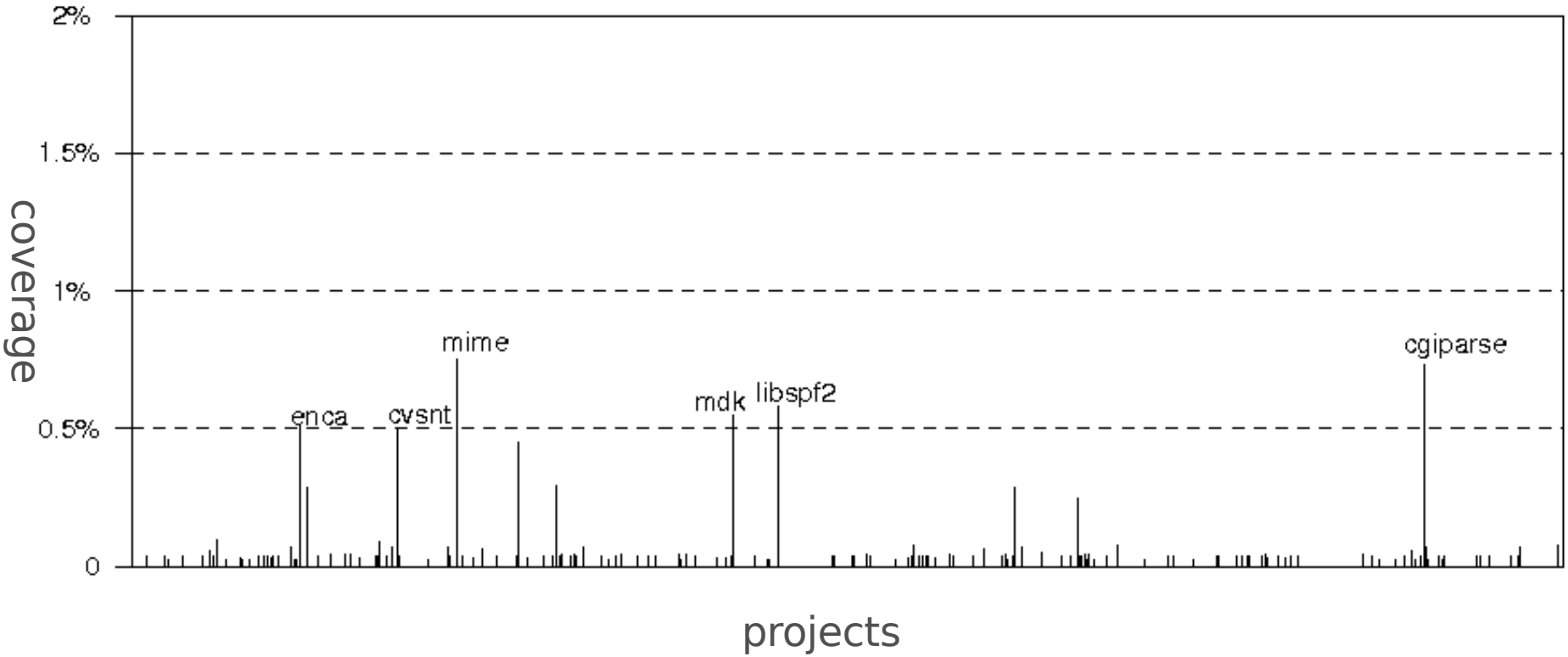
Most of the code clones from a single file: `getopt.c`



SPARS-J & FreeBSD's ports Results

Code for handling CGI requests

Code from a specialized version of `getopt.c`



Discussion

Speed

- Code clone detection was 4 times slower than the ideal case…
  - Data transfer from and to the shared network file system
  - Preprocessing/post-processing of files
- …but 20 times faster than performing the computation using the distributed model with a single computer

Dot plot

- Traded speed and size for accuracy
- Source code structure easily visible
- Exaggerated cluster of pixels and fine details lost
- Small projects not visible

The FreeBSD's ports

- A beforehand knowledge of the source code repository's structure would have probably led to more interesting results

Conclusions/Future

- Proposed a novel approach to distributed large scale code clone detection and implemented it as D-CCFinder
- Evaluated D-CCFinder with two case studies
  - Performed code clone detection over a source code repository of about 11Gbytes and visualized the results
  - Detected the use of source code from various open source projects within the SPARS-J system
- Remove the dependency on CCFinder
  - New code clone detection algorithm

Thanks for your attention

Questions?