Analysis of the Linux Kernel Evolution Using Code Clone Coverage

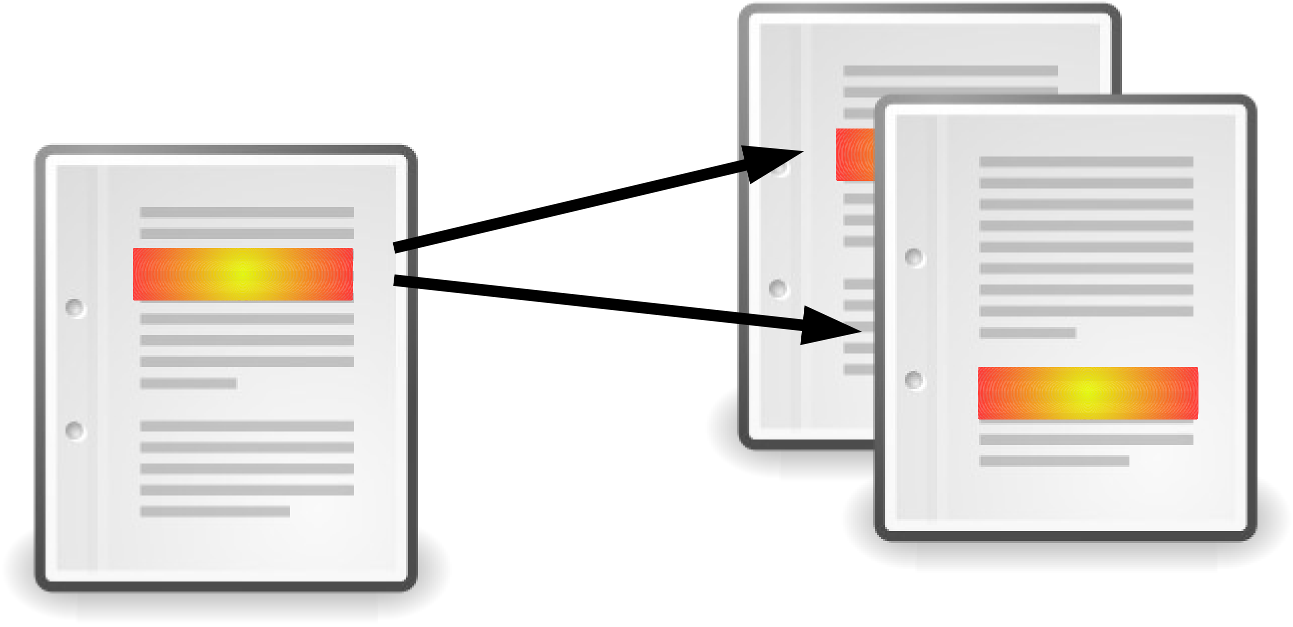
Simone Livieri
Yoshiki Higo
Makoto Matsushita
Katsuro Inoue

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

Code clone?

Code Clone?

- A code clone is a set of identical or similar fragments of code



Code Clone Detection

- Various detection methods
  - Token based
  - Abstract Syntax Tree based
  - Program Dependence Graph based
  - …

CCFinder

- Token-based code-clone detection tool
  - Insensitive to renamed variable and code layout
  - Multi language support (C, C++, COBOL, Java, ...)
- Very good scalability and speed...

CCFinder

- Token-based code-clone detection tool
  - Insensitive to renamed variable and code layout
  - Multi language support (C, C++, COBOL, Java, ...)
- Very good scalability and speed...
- ...but scalability is limited by the hardware used

D-CCFinder

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

What?

What

Start a large scale study of a software system's evolution using code clone analysis

Why?

Why

- The evolution of a software system can be reconstructed with code clone analysis
- No large scale study have been performed yet
- D-CCFinder permits large scale code clone analysis

Why

- The evolution of a software system can be reconstructed with code clone analysis
- No large scale study have been performed yet
- D-CCFinder permits large scale code clone analysis
- There were two weeks left before the deadline and we had nothing to do

The guinea pig

The Linux Kernel

- 15 years long development effort involving hundreds of developers
- Two development branches: **stable** and **unstable** (prior to version 2.6)
- The source code size grew from 3.8Mbytes (version 1.0) to 157Mbytes (2.6.18.3)

The Linux Kernel

Version	LOC	Size (Kbytes)	# of versions
1.0	141K	3,926	1
1.2.0~ 1.2.13	234K 238K	6,534 6,596	14
2.0.0~ 2.0.40	563K 768K	16,076 21,952	41
2.2.0~ 2.2.26	1,310K 1,970K	37,056 58,812	27
2.4.0~ 2.4.33.4	2,366K 3,865K	69,200 112,148	34
2.6.0~ 2.6.18.3	4,120K 5,476K	120,030 157,290	19

Total number of versions	136
Number of .c files	376,596
Total lines of code	266,943,565
Total size	7.4 Gbytes

- 136 kernel versions from the stable branches
- Considered only .c files
- Size measured with `du`
- LOC counted with `wc`

How?

How

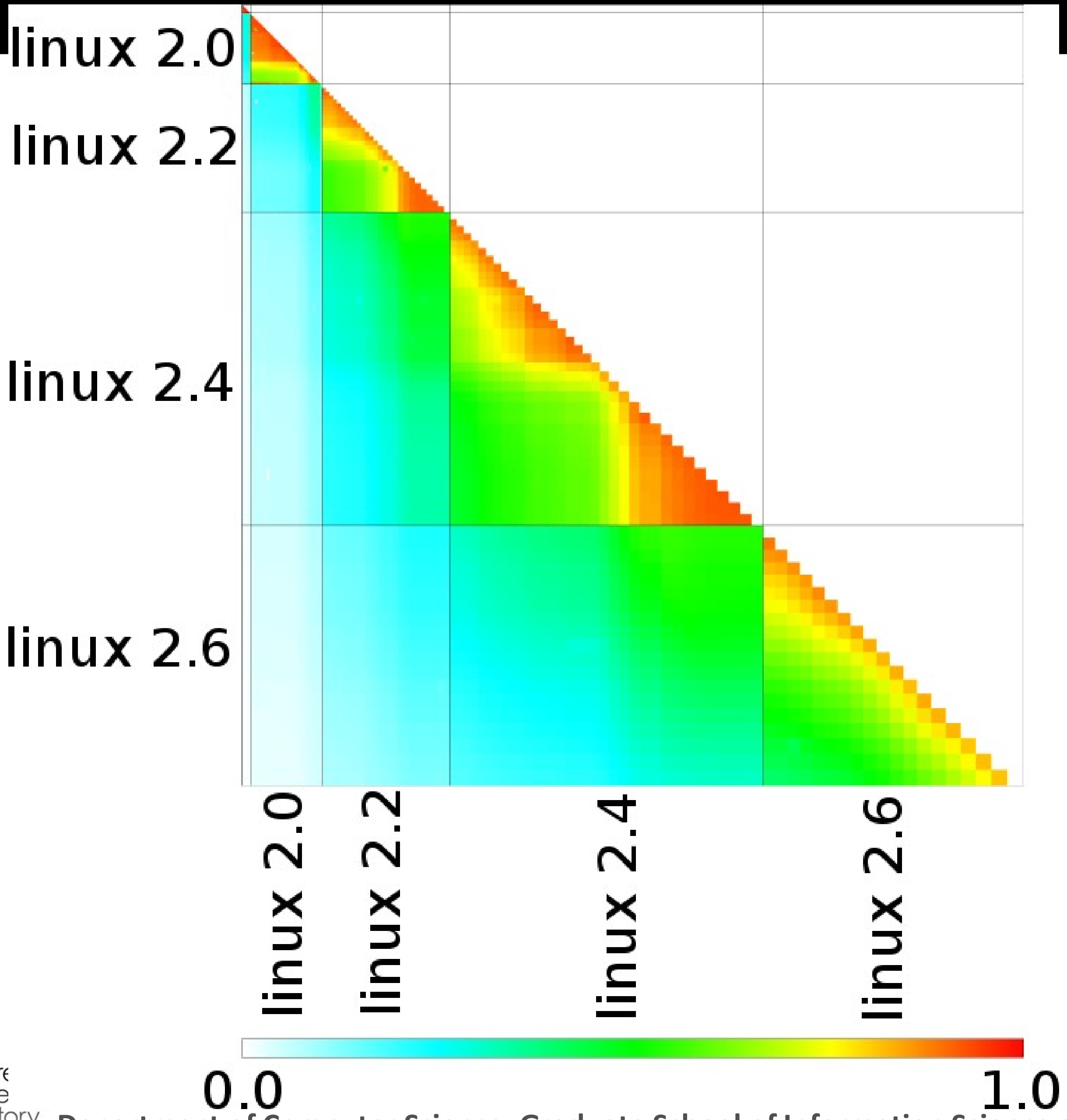
- For each pair (A, B) of kernel versions we computed and plotted the code clone coverage

$$Coverage(A,B)=\frac{Loc(CC(A,B))}{Loc(A)+Loc(B)}$$

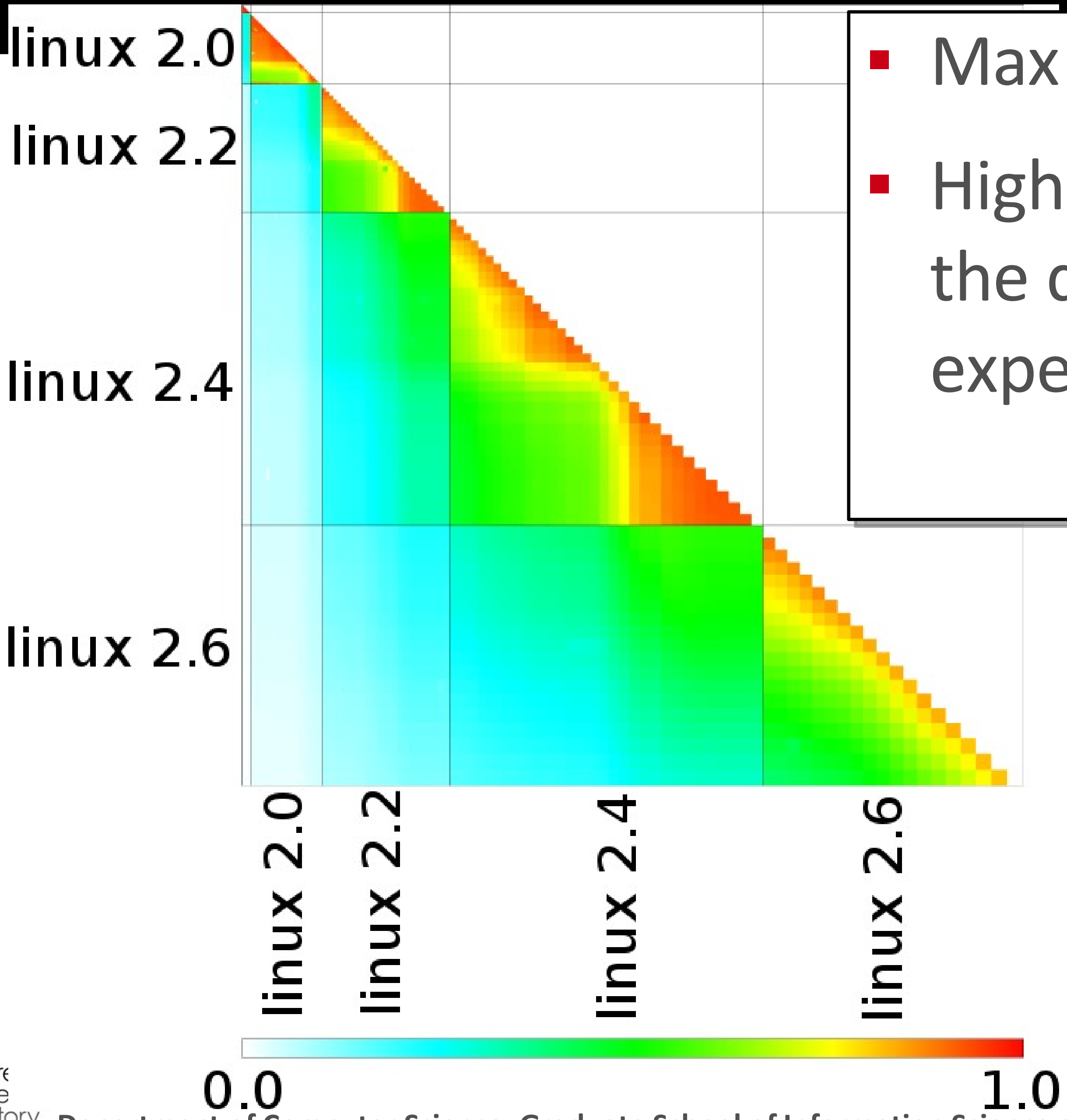
CC(A,B)= code clone fragments between A and B

Results

Results

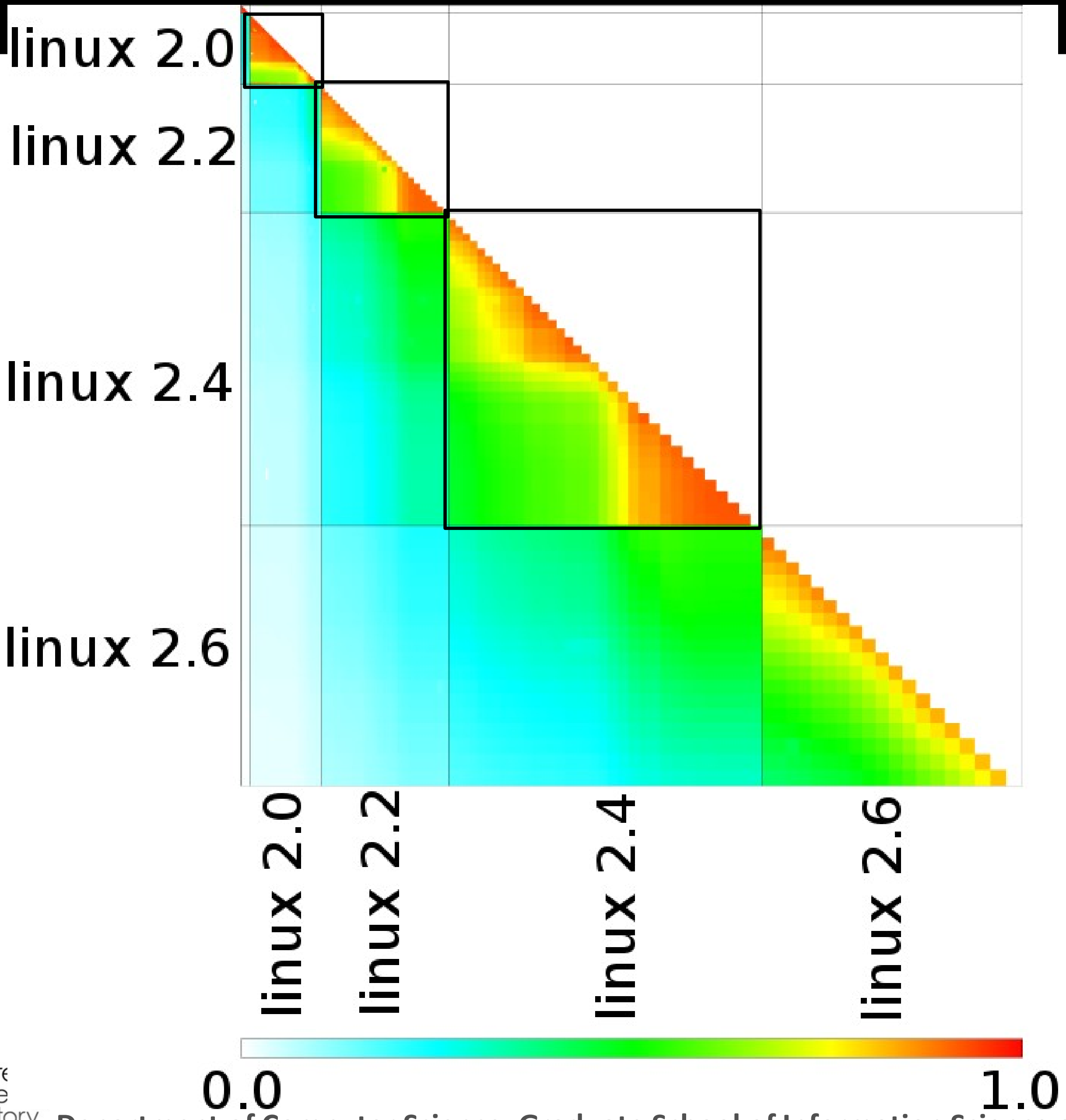

linux 2.0
linux 2.2
linux 2.4
linux 2.6
linux 2.0
linux 2.2
linux 2.4
linux 2.6
0.0
1.0

Results



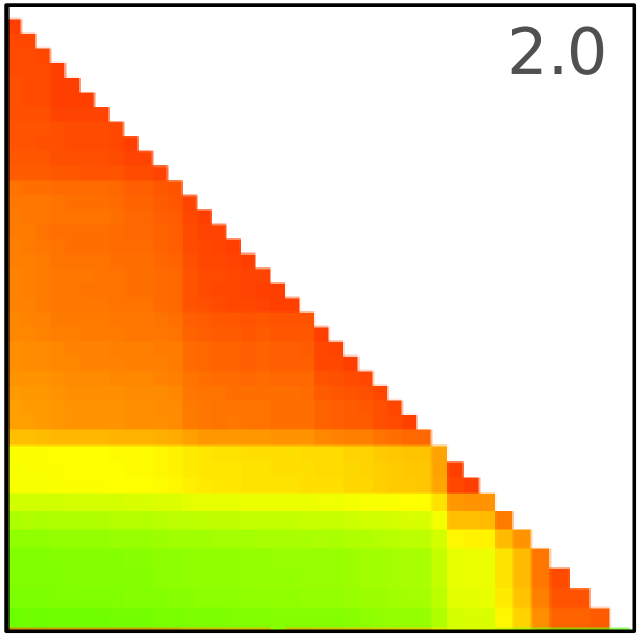
- Max coverage: 67%
- Highest coverage on the diagonal (as expected)

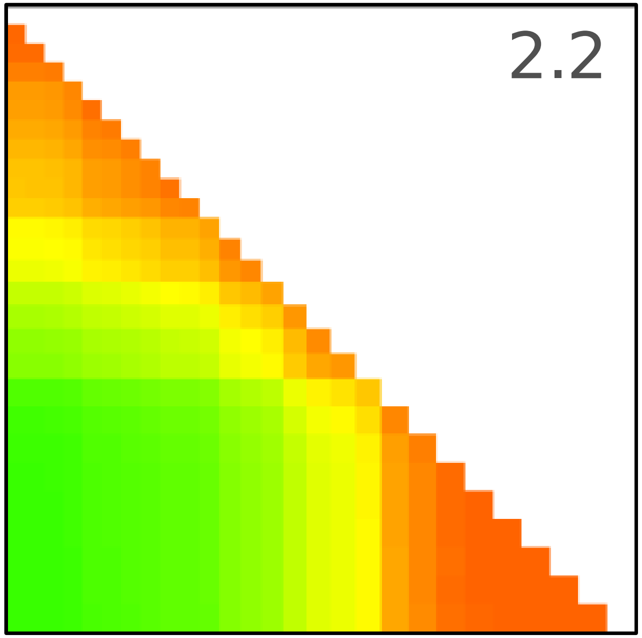
Results

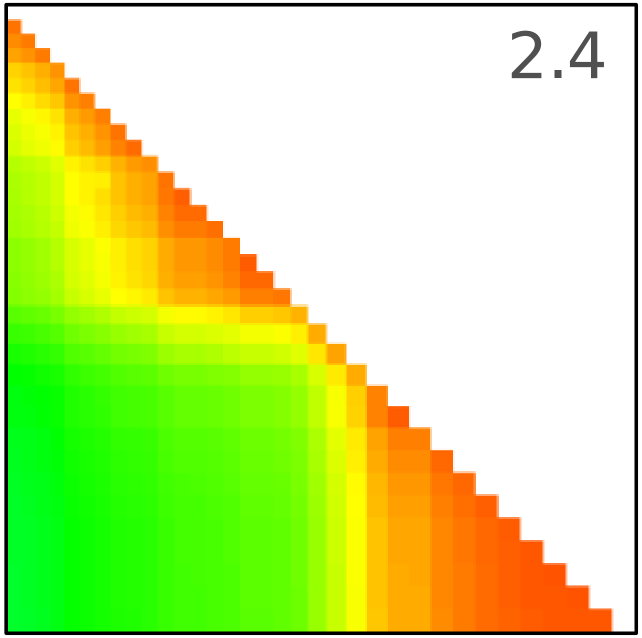

linux 2.0
linux 2.2
linux 2.4
linux 2.6
linux 2.0
linux 2.2
linux 2.4
linux 2.6
0.0
1.0

Results

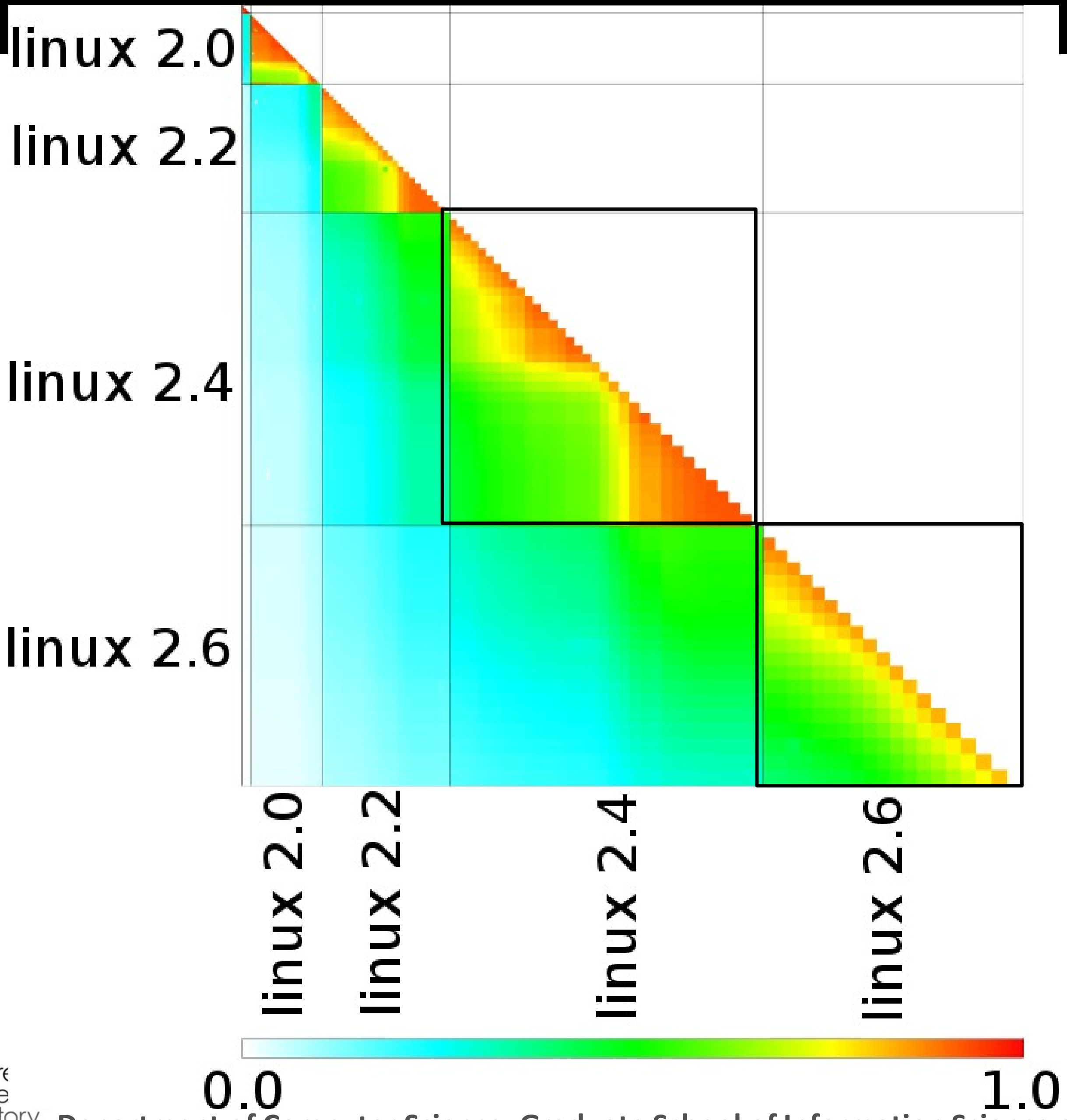
- Same pattern
- Code "back-ported" from the development branches





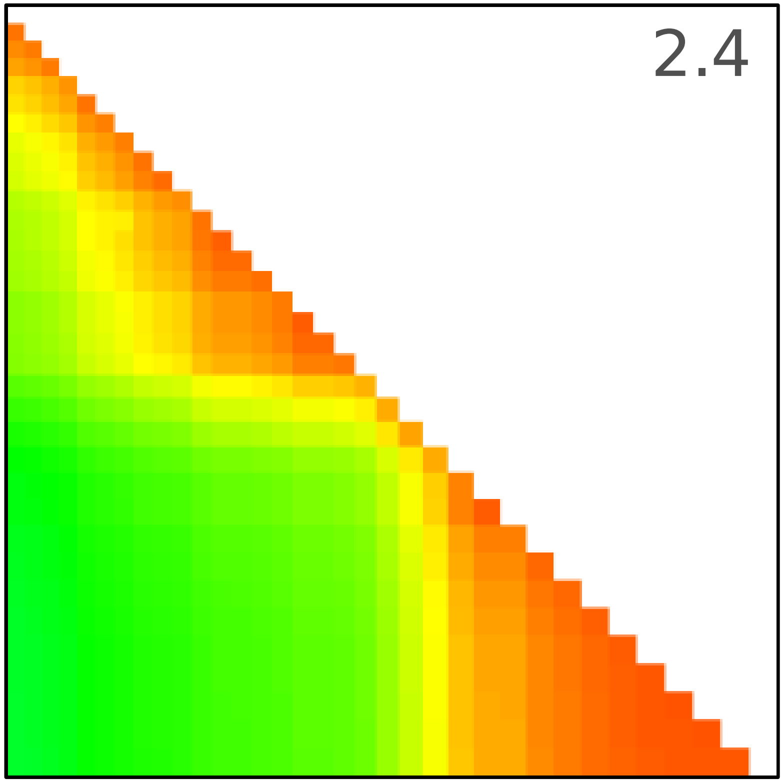


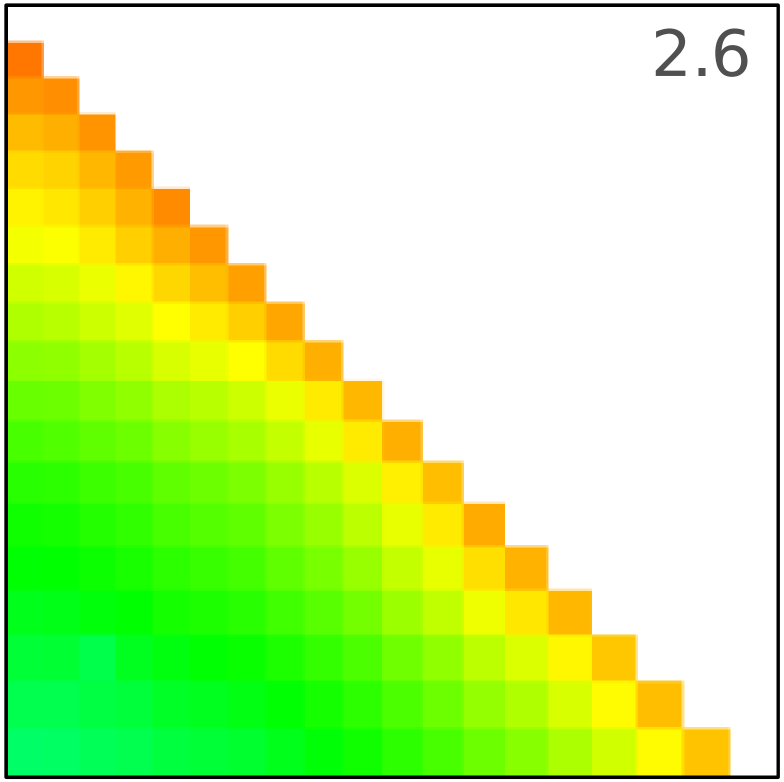
Results


linux 2.0
linux 2.2
linux 2.4
linux 2.6
linux 2.0
linux 2.2
linux 2.4
linux 2.6
0.0
1.0

Results

- Different patterns due to different development processes





Conclusion & Future

- **Conclusion**
  - Presented a tentative study of the evolution of the Linux kernel computing and visualizing the code-clone coverage metric
- **Future**
  - Elaborate and complete the analysis presented and produce a more detailed report of the code changes

以上