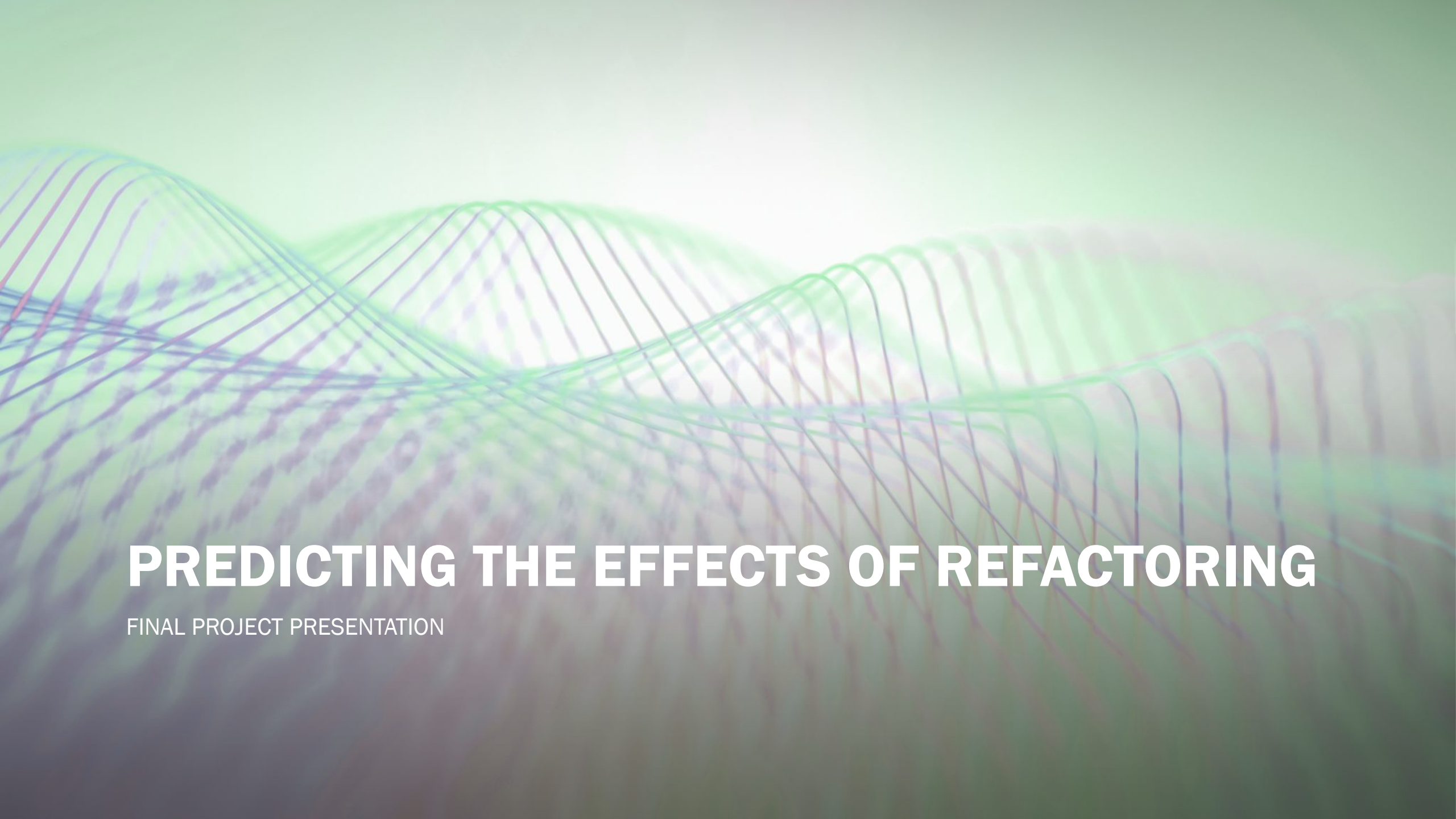


PREDICTING THE EFFECTS OF REFACTORING

FINAL PROJECT PRESENTATION

ORIGINAL OVERVIEW AND GOALS

- Predicting refactoring effects using machine learning
  - Through product metrics before and after a refactoring commit based on history git data
  - Approach based on the paper „On the Impact of Refactoring Operations on Code Quality Metrics“ [7]

- Goals
  - Create a large dataset containing enough refactoring operations
  - Determine which features are suitable to predict how well an refactoring operation is performing
  - Generate a prediction model based those features and evaluate it

- Main research questions to be answered
  - **Which features of a commit-refactoring combination are suitable to predict the effect of the refactoring method?**
  - **How well does the trained machine learning model perform?**

ORIGINAL OVERVIEW AND GOALS

- Predicting refactoring effects using machine learning
  - Through product metrics before and after a refactoring commit based on history git data
  - Approach based on the paper „On the Impact of Refactoring Operations on Code Quality Metrics“ [7]

- Goals
  - Create a large dataset containing enough refactoring operations
- Why?

CHALLENGES IN MINING SOFTWARE REPOSITORIES

REPRODUCIBILITY CHALLENGE

- Most projects do not provide a pipeline or even the code to generate the used data set at all
- GHTorrent or Sourced use hosted datasets
  - Case Sourced shows that this can get critical when site and dataset is not available anymore
- Therefore, a pipeline with well defined in- and outputs will provide better reproducibility and result validation

MULTI-DIMENSIONALITY OF GIT DATA

- Git is file-versioning built upon a tree-like structure
- When looking at only one branch, it can be seen like a linear timeline
- When mining, several new dimensions occur (introduced by a loop for reaching point in the file once) and each is impacting the computation time

Dimensions:

- **Time dimension:** Total size of the analyzed commits
- **File dimension:** Each commit contains multiple files, and they can even vary by adding and removing files.
- **Structural node dimension:** Depending on the analysis problem, each class, method or class member must be visited. Could be compared to an abstract syntax tree (AST).
- **Metric dimension:** Also dependent on what should be calculated. RefactoringMiner uses UML to identify changes, CK uses an AST. This also adds a potential loop or at least computation time.
- Depending on the problem, structural node dimension and and metric dimension is tied together very strongly.

PARALLELIZATION CHALLENGE

- Common libraries used are RepoDriller and PyDriller (based on jGit and GitPython)
- RepoDriller is not threaded at all, as PyDriller claims to be
  - Looking closer at the code unveils the following line:

```
157     def checkout(self, _hash: str) -> None:
158         """
159         Checkout the repo at the speficied commit.
160         BE CAREFUL: this will change the state of the repo, hence it should
161         *not* be used with more than 1 thread.
162 
163         :param _hash: commit hash to checkout
164         """
165         with self.lock:
166             self._delete_tmp_branch()
167             self.git.checkout('-f', _hash, b='_PD')
168 
```

PARALLELIZATION CHALLENGE

- Common libraries used are RepoDriller and PyDriller (based on jGit and GitPython)
- RepoDriller is not threaded at all, as PyDriller claims to be
  - PyDriller is only multithreaded on file dimension, not time dimension
  - And it can not be using parallelization, because the libraries built on are also not able to do this (neither jGit nor GitPython)
- Reason for this: Git writes lock-files to prevent inconsistencies when checking out multiple revisions at a time
- Only way to parallelize: Clone or copy the repository multiple times

GENERALIZATION CHALLENGE

- MSR is used for a variety of different research projects
  - e.g. developer behavior, social aspects, code smell detection or code evolution
- Since every project uses its own approach, single steps can not be reused
- By using frameworks like RepoDriller oder PyDriller, it is easier, but no complete pipelining framework does exist

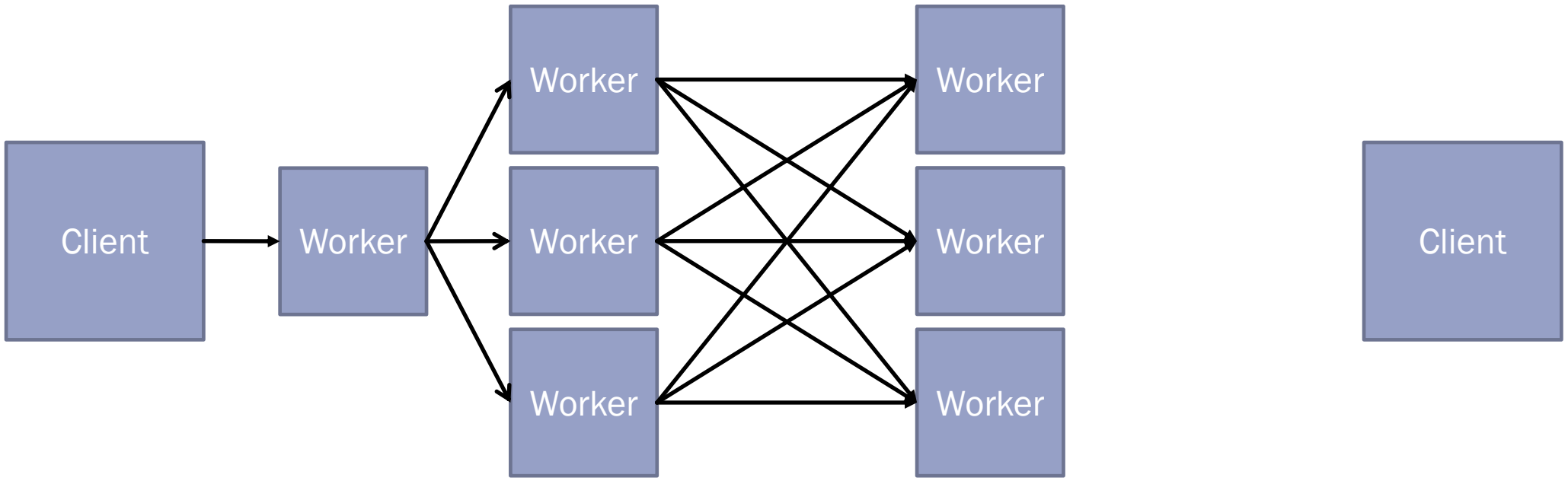
PIPELINE OVERVIEW

PIPELINE OVERVIEW

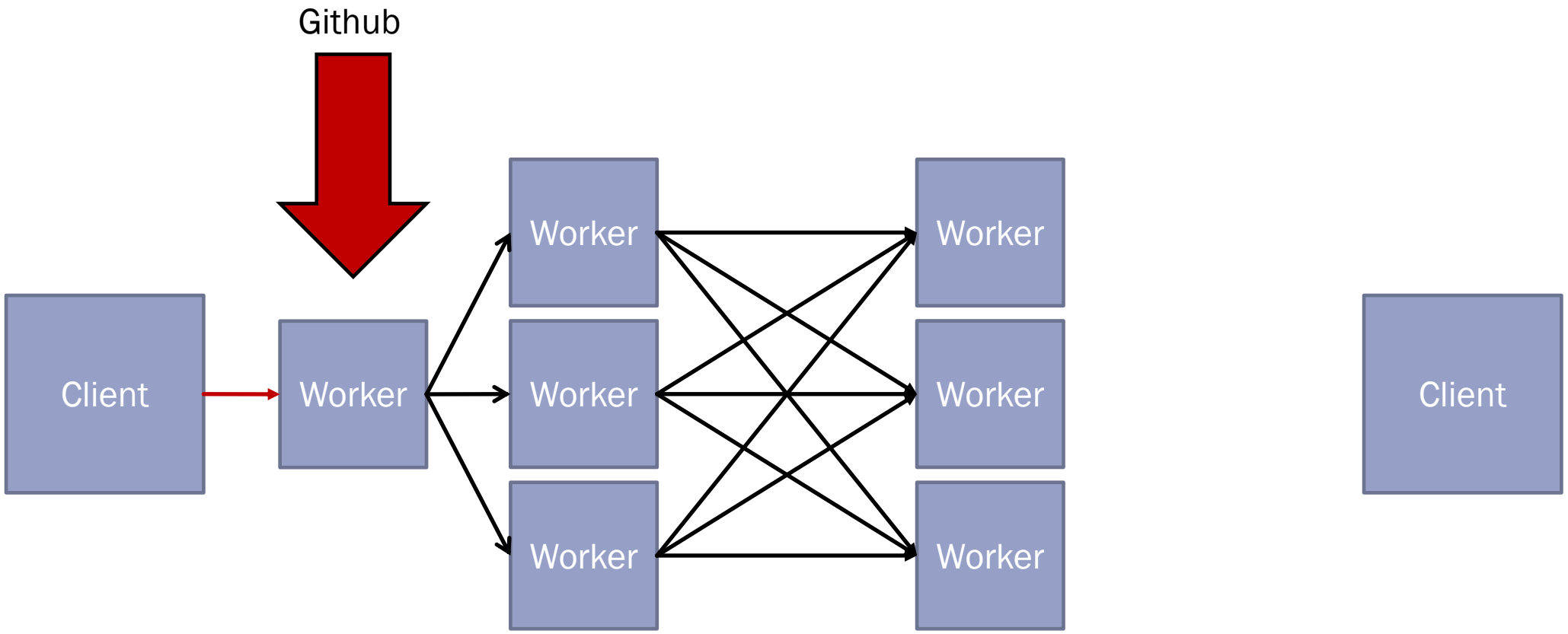
- Pipeline for detecting refactoring operations and calculate software quality metrics before and after the commit
- Fully automated pipeline which takes as input a list of repository urls
- And outputs a list of commits with information about the **refactoring operation**, the **affected files**, and different **types of metrics** (provided by the CK library)

- Pipeline uses **Apache Spark** to **parallelize** the time dimension in a **distributed and scalable** way

MSR MINING PIPELINE

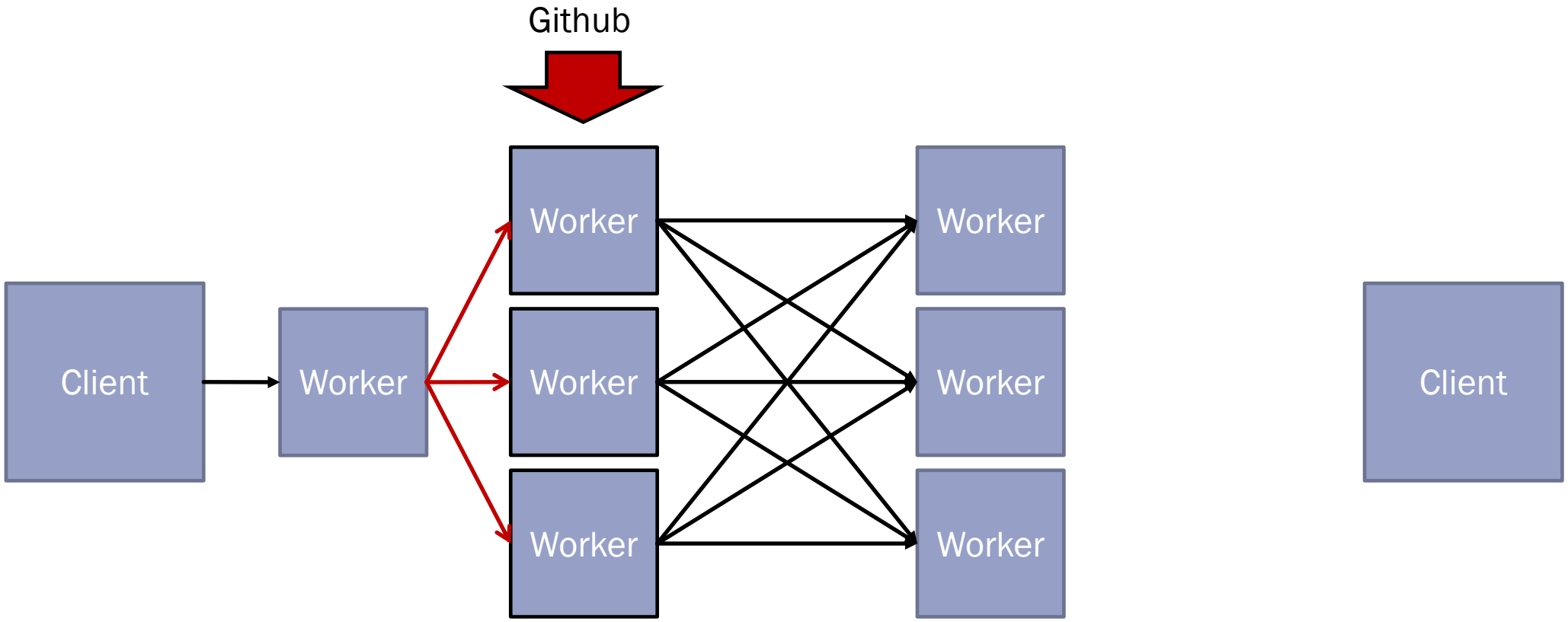


1. CLONE REPOSITORY + SPLIT COMMITS IN K-SIZE BATCHES



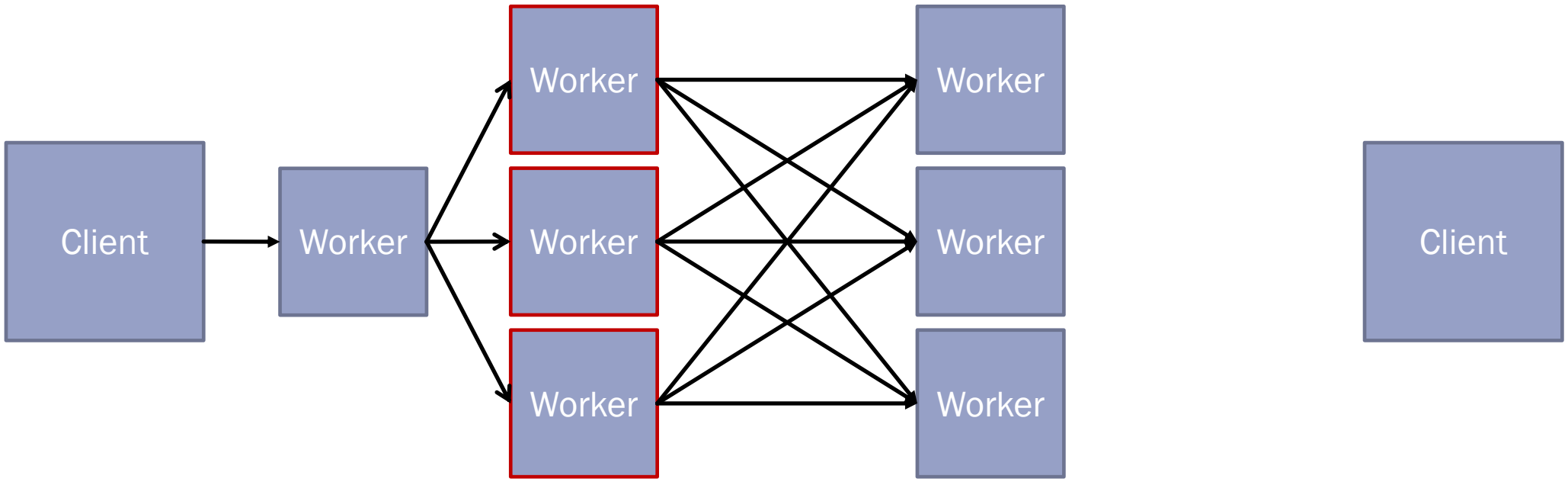
1. Clone repository from Github once
2. Split into batches of size k (configurable, 100-1000 commits per batch)

2. SHUFFLE COMMIT BATCHES + CLONE REPOSITORY



1. Shuffle and sort phase to redistribute commit batches to all available workers
2. Each used worker will clone the repository (by if-not-exist style to avoid multiple cloning)

3. RUN REFACTORING MINER



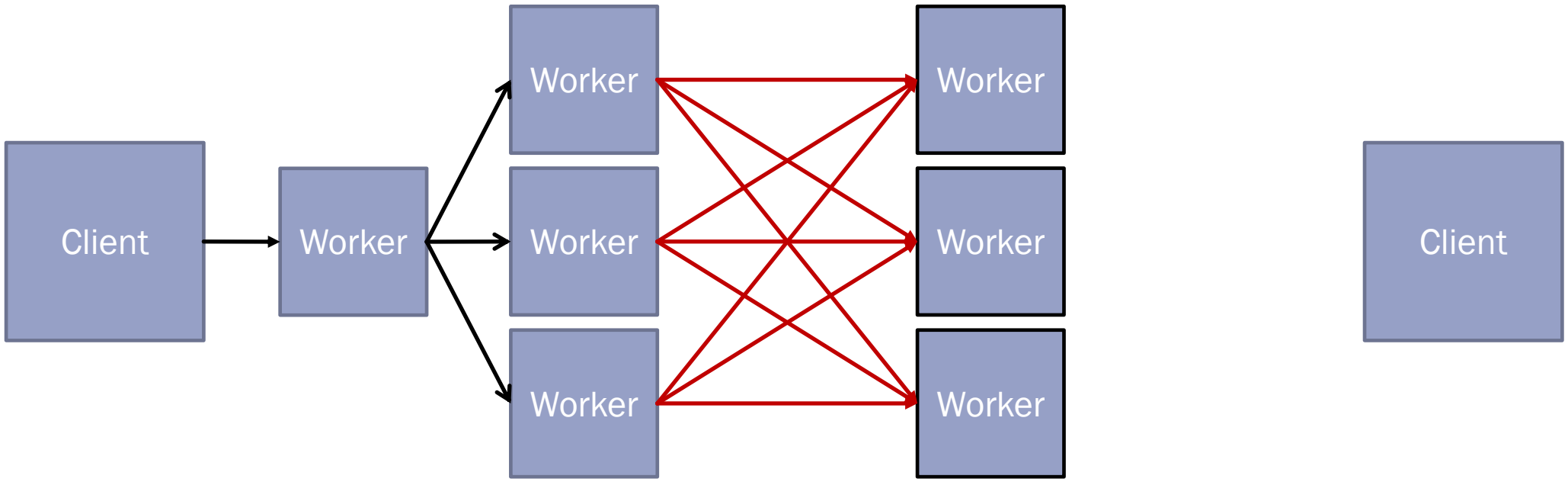
1. RefactoringMiner to analyze and identify interesting commits
2. Write intermediate results for fault tolerance

REPOSITOTY MINER INTERMEDIATE RESULTS

repository	commit_id	type
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	MOVE_CLASS
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	MOVE_CLASS
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	CHANGE_ATTRIBUTE_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_ATTRIBUTE_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	RENAME_PARAMETER
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	RENAME_PARAMETER
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE

Showing 1 to 10 of 29,006 entries

4. COMBINING INTERMEDIATE RESULTS



1. Combining intermediate results as every row contains one refactoring operation
2. Refactoring types are combined or “flattened” to only have one commit id with multiple refactoring operations

“FLATTENING” INTERMEDIATE RESULTS

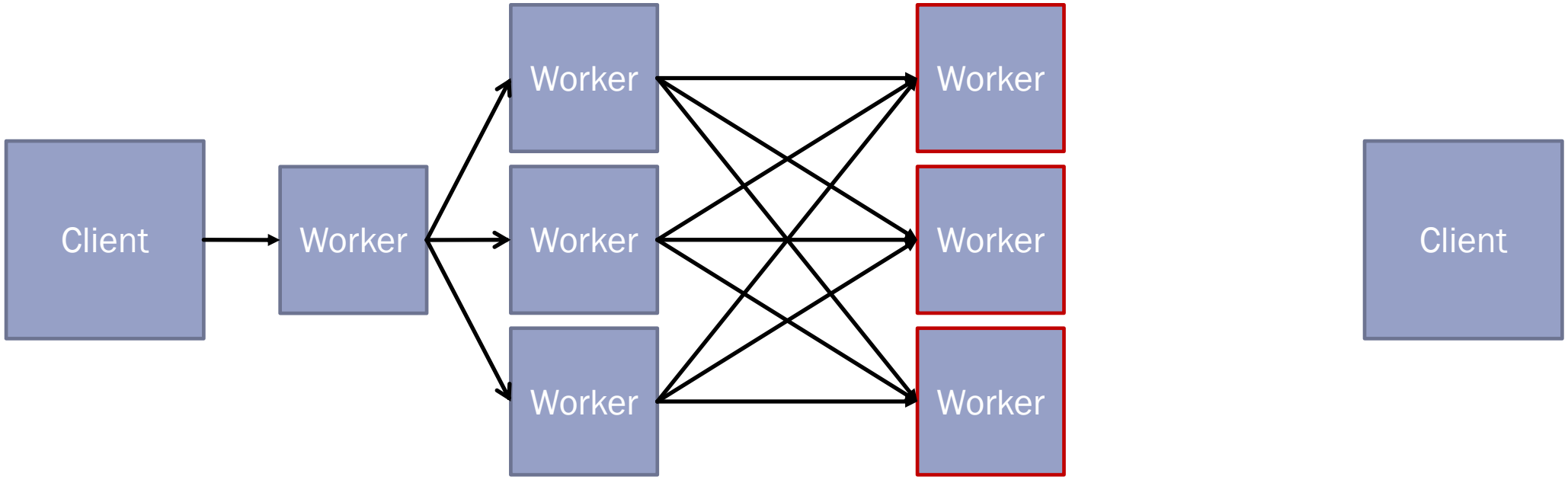
repository	commit_id	type
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	MOVE_CLASS
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	MOVE_CLASS
https://github.com/gradle/gradle	f666121b2aa03501e5758c09b97162592af103f2	CHANGE_ATTRIBUTE_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_ATTRIBUTE_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	RENAME_PARAMETER
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	RENAME_PARAMETER
https://github.com/gradle/gradle	c73bbb70cad1e4ae73ee6dfafdb48be8bfd701f5	CHANGE_PARAMETER_TYPE

MOVE_CLASS, MOVE_CLASS, CHANGE_ATTRIBUTE_TYPE

Showing 1 to 10 of 29,006 entries

Previous 1 2 3 4 5 ... 2901 Next

5. CALCULATING SOFTWARE QUALITY METRICS (CK)



1. Calculate software quality metrics using CK
2. Checkout commit and identify changed files between commit and parent commit (diff)
3. Calculate metrics for commit and parent commit and save results

METRIC RESULTS

repository	commit_id	parent_commit_id	file	refactoring_operation	modification_type	side	loc	dit	wmc	cbo	lcom	rfc	number_of_fields
https://github.com/eeayiaia/bobothepanda	059833ab69456340049b78b971c56334d0d43930	a928a72932169bb5e26fa2fa2a2ec34e4e24230b	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	294dda79e6d4fbe039756cb8cfb5f961c7943505	bda97024aed471d4f0baf1b8bbed7c09f067d4ab	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	859b97f424b8464a7daf7e3f88c0ffc704fccaae	5dfd4b133b410166d5890d6fdb028ee890a396cf	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	ba3942e8d504036640dfe375c2d7189ede8a9093	29cd137fdeef9cc22cbb5588598a68daf944d94e	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	bcb4fca62324b645931d57ceabebfb7af2193674	e8ecab33be489714985da818ad1f85f4c65205b4	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	5d3612638ad404d842ea8d3401c4712d65ae224b	e0fce606082fd8276a8fe162fac8907c8861a810	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	ADD_METHOD_ANNOTATION	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	294dda79e6d4fbe039756cb8cfb5f961c7943505	bda97024aed471d4f0baf1b8bbed7c09f067d4ab	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	EXTRACT_ATTRIBUTE	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	859b97f424b8464a7daf7e3f88c0ffc704fccaae	5dfd4b133b410166d5890d6fdb028ee890a396cf	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	EXTRACT_ATTRIBUTE	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	ba3942e8d504036640dfe375c2d7189ede8a9093	29cd137fdeef9cc22cbb5588598a68daf944d94e	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	EXTRACT_ATTRIBUTE	MODIFY	left	8	1	2	4	1	1	0
https://github.com/eeayiaia/bobothepanda	bcb4fca62324b645931d57ceabebfb7af2193674	e8ecab33be489714985da818ad1f85f4c65205b4	/tmp/repos/0/bobothepanda/project/bobothepanda/src/main/java/model/Door.java	EXTRACT_ATTRIBUTE	MODIFY	left	8	1	2	4	1	1	0

Showing 1 to 10 of 104 entries

Previous 1 2 3 4 5 ... 11 Next

PIPELINE AUTOMATION

```
#!/usr/bin/env bash

rm -R /Users/martinsteinhauer/Desktop/commits
rm -R /Users/martinsteinhauer/Desktop/commitmetricresults

~/sdk/spark-2.4.5/bin/spark-submit --master "spark://192.168.0.100:7077" \
--class it.unisa.softwaredependability.Main \
--executor-memory 8g \
--conf "spark.serializer=org.apache.spark.serializer.KryoSerializer" \
target/dataset-generation-pipeline-1.0-SNAPSHOT.jar \
--input "datasets/small.csv" \
--commits-output "path/to/dir/commits" \
--metrics-output "path/to/dir/commitmetricresults" \
--refactoring-mining-only \
--parallel-jobs 1 \
--parallel-repos 1 \
--batch-size 100 \
--username "<yourGithubUsername>" \
--token "<yourGithubAccessToken>"
```

RUNTIME EVALUATION

EVALUATION ENVIRONMENT

- Running on smaller repositories
  - https://github.com/google/guava
  - https://github.com/google/gson
  - https://github.com/reactivex/rxjava
- Each project evaluated with 1, 2, 4 and 8 worker nodes
- Pipeline with 1 worker is the reference implementation for no parallelization
- Evaluation system
  - All nodes running on one machine with multiple worker instances (system with 4 Intel Xeon E-7 4870 deca-core CPUs and 64GB memory)
  - Each worker node with available memory of 8GB memory and 1 CPU core
- Restricted to only refactoring mining part of the pipeline because
  - No reference implementation to compare with
  - Long runtime when evaluating each project four times
  - Some repositories are still able to mine due to problems with *core.autodiff* and case-sensitive files from Windows systems (“merge” conflict)

RUNTIME RESULTS

TABLE I
COMPUTATION TIME

Repository	Executors	Runtime (min)
reactivex/rxjava	1	48
reactivex/rxjava	2	40
reactivex/rxjava	4	34
reactivex/rxjava	8	27
google/gson	1	1.5
google/gson	2	1.7
google/gson	4	1.9
google/gson	8	1.9
google/guava	1	35
google/guava	2	27
google/guava	4	21
google/guava	8	19

TABLE II
REPOSITORY DIMSIONALITY METRICS

Repository	File Min	File Max)	Commit Count
reactivex/rxjava	4	1862	5752
google/gson	134	207	1485
google/guava	65	3173	5282

LIMITATIONS

- Runtime could be reduced up to 45.7% in comparison to a single threaded approach
- Nevertheless, scaling/doubling worker nodes only reduces the runtime by up to 25%, not as expected by 50%

- Each step still uses a different content analysis input
  - RefactoringMiner uses UML extracted directly from the object store
  - CK uses a folder with source files as input (or a list of source files)
- Repositories are cloned multiple times
  - Depending on the internet connection, this can take a lot of time (in my case, biggest repo only took about 5 minutes)
- Spark can not optimize the data and calculations within and around each library computation step
  - Therefore, it is a blackbox and each step must wait until all partitions are done
  - No query optimization like selection and join operation push up and down are available which reduce shuffled data and increase performance

FURTHER WORK

- Replace the file-based Git archive with an alternative data storage like it has been done in Sourced or GHTorrent, but with additional file content
  - Avoid file system locks
  - Data available in structured format, query optimization can do its job
- Build cleaner data in- and output interfaces for each step to make the pipeline more generalizable
- Look at alternative storage formats like storing file content as an AST within a graph database
  - Could be a good option for code evolution research

REFERENCES

[1] N. Tsantalis, M. Mansouri, L. M. Eshkevari, D. Mazinanian, and D. Dig, “Accurate and efficient refactoring detection in commit history,” pp. 483–494, 2018.

[2] https://github.com/tsantalis/RefactoringMiner

[3] https://github.com/mauricioaniche/repodriller

[4] https://github.com/im-a-giraffe/RefactoringMiner

[5] https://github.com/mauricioaniche/ck

[6] https://www.eclipse.org/jgit/