Christian-Albrechts-Universität zu Kiel
Technische Fakultät

A Pattern-based Transformation Approach to Parallelise Software Systems using a System Dependency Graph

Johanna E. Krause

27 January 2016

Outline

- Parallel programs are mostly more performant

Motivation for (Semi-)Automatic Parallelisation

- Parallel programs are mostly more performant
- Many legacy systems would benefit from parallelisation

Motivation for (Semi-)Automatic Parallelisation

- Parallel programs are mostly more performant
- Many legacy systems would benefit from parallelisation
- Manual adjustments are time and cost consuming

Christian-Albrechts-Universität zu Kiel
Technische Fakultät

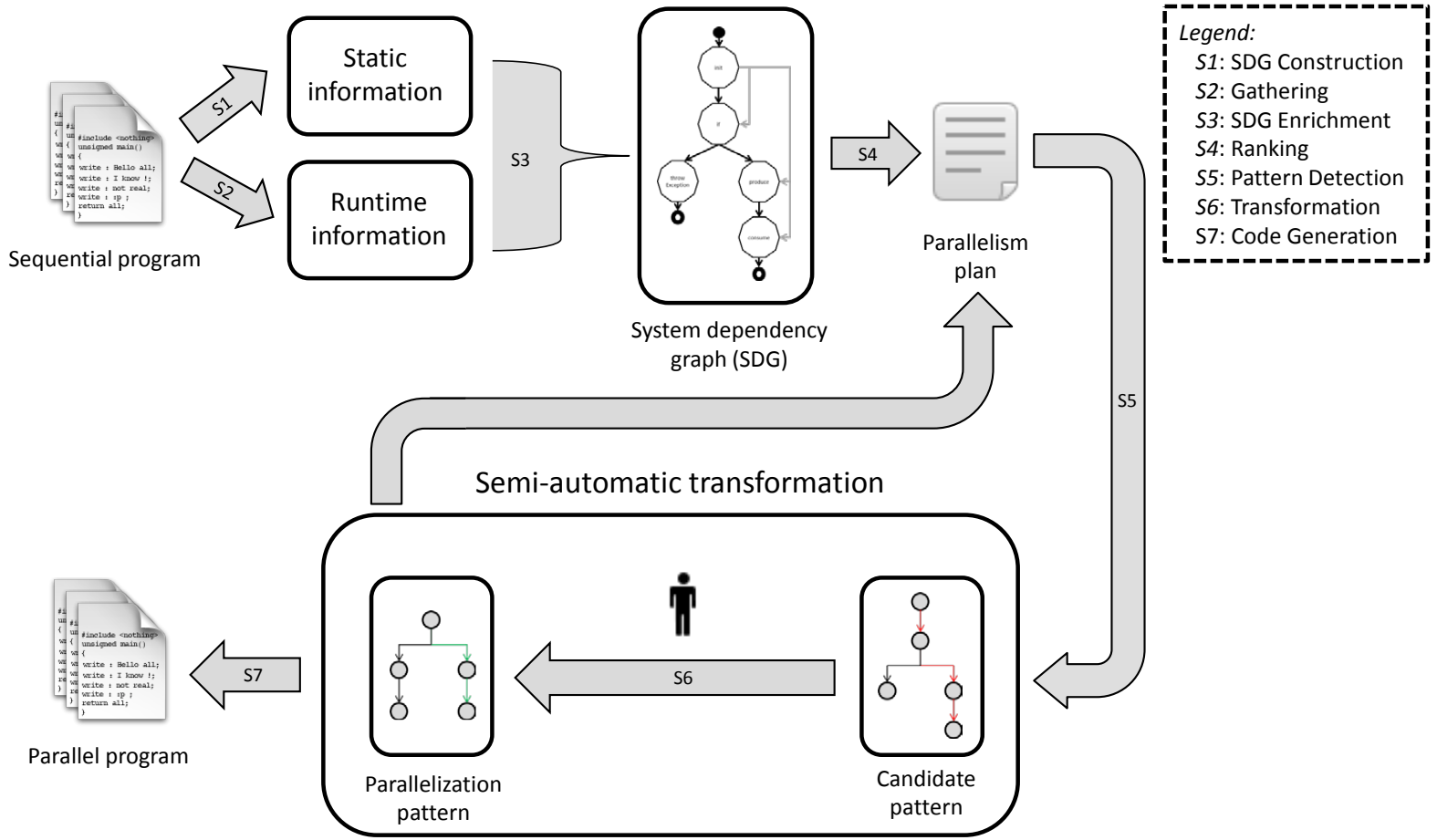


Figure 1 : Pattern-based detection and utilization of potential parallelism in software systems [Wulf14]

- **Goal 1: Mining of Candidate and Parallelisation Patterns**

Christian-Albrechts-Universität zu Kiel
Technische Fakultät

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)
  - G2.2: Formalising as Cypher Match Query (CMQ)

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)
  - G2.2: Formalising as Cypher Match Query (CMQ)
- **Goal 3: Transforming Candidate Patterns to Parallelisation Patterns**

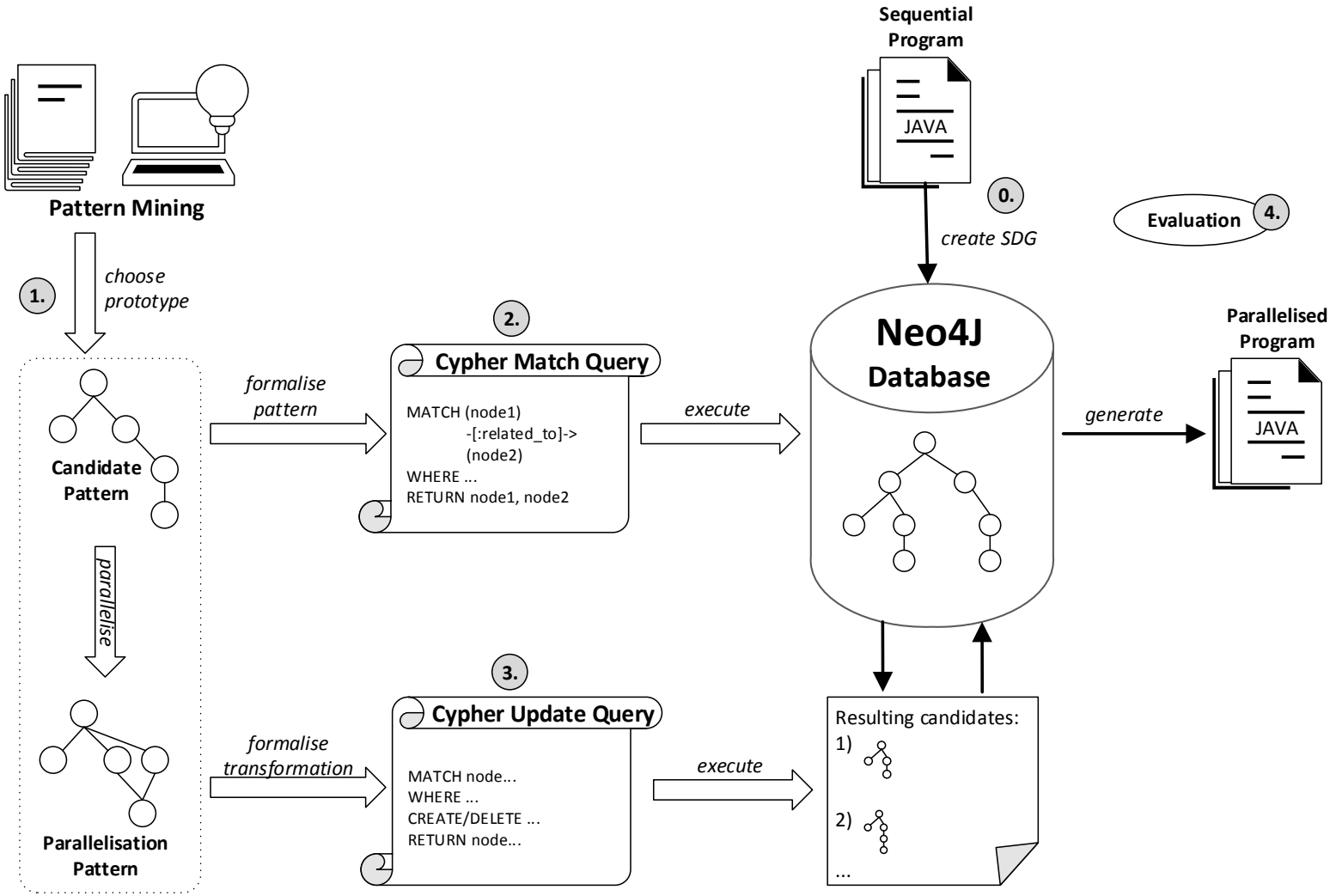
- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)
  - G2.2: Formalising as Cypher Match Query (CMQ)
- **Goal 3: Transforming Candidate Patterns to Parallelisation Patterns**
  - G3.1: Formalising as SDG

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)
  - G2.2: Formalising as Cypher Match Query (CMQ)
- **Goal 3: Transforming Candidate Patterns to Parallelisation Patterns**
  - G3.1: Formalising as SDG
  - G3.2: Formalising as Cypher Update Query (CUQ)

- **Goal 1: Mining of Candidate and Parallelisation Patterns**
- **Goal 2: Formalising Candidate Patterns**
  - G2.1: Formalising as System Dependency Graph (SDG)
  - G2.2: Formalising as Cypher Match Query (CMQ)
- **Goal 3: Transforming Candidate Patterns to Parallelisation Patterns**
  - G3.1: Formalising as SDG
  - G3.2: Formalising as Cypher Update Query (CUQ)
- **Goal 4: Evaluating the Speed-Up of the Transformed Application**

Approach for the Thesis

Approach



Independent Successive Method Calls

```
dataserver.connect();
eventserver.connect();
```

Independent Successive Method Calls

```
dataserver.connect();
eventserver.connect();
```

Independent For-Each Loop

```
for (ImportantObject o : list) {
    result = calculateSomethingForQuiteAWhile(o);
    writeResultInDatabase(result);
}
```

Independent Successive Method Calls

```
dataserver.connect();
eventserver.connect();
```

Independent For-Each Loop

```
for (ImportantObject o : list) {
    result = calculateSomethingForQuiteAWhile(o);
    writeResultInDatabase(result);
}
```

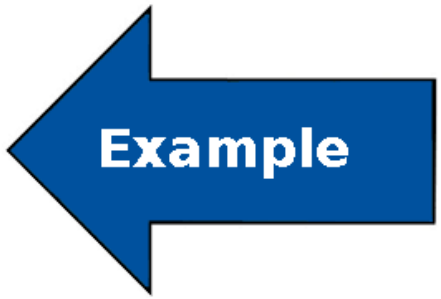
Array Reduction

```
int sum = 0;
for (int i = 0; i < array.length; i++) {
        sum = sum + array[i];
}
```

[Molitorisz12, Mattson04]

Independent Successive Method Calls

```
dataserver.connect();
eventserver.connect();
```



Independent For-Each Loop

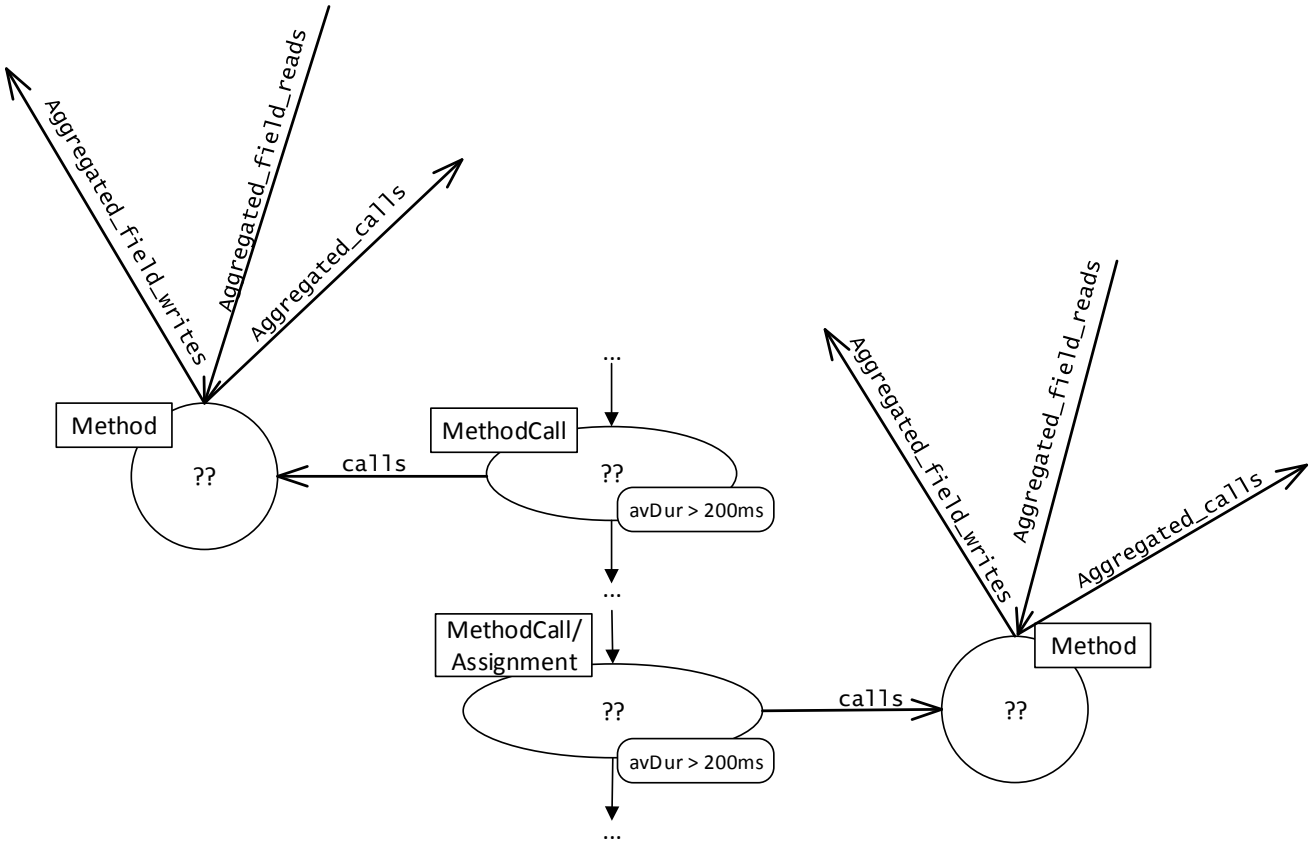
```
for (ImportantObject o : list) {
    result = calculateSomethingForQuiteAWhile(o);
    writeResultInDatabase(result);
}
```

Array Reduction

```
int sum = 0;
for (int i = 0; i < array.length; i++) {
        sum = sum + array[i];
}
```

[Molitorisz12, Mattson04]

C | A | U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät



C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
RETURN collect(DISTINCT id(m1))
```

C | A | U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
RETURN collect(DISTINCT id(m1))
```

C | A | U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
RETURN collect(DISTINCT id(m1))
```

runtime information configurable

C | A | U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[cfs:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE has(cf.case))
RETURN collect(DISTINCT id(m1))
```

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[cfs:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE has(cf.case))
RETURN collect(DISTINCT id(m1))
```

```
boolean isAvailable = isProductAvailable();
if(isAvailable){
    processOrder();
}
```

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (m1:MethodCall)
      -[cfs:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE has(cf.case))
AND NOT (m1) -[:DATA_FLOW*1..5]-> (m2)
RETURN collect(DISTINCT id(m1))
```

```
MATCH (m1:MethodCall)
      -[cfs:CONTROL_FLOW*1..5]->
      (m2:MethodCall)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE has(cf.case))
AND NOT (m1) -[:DATA_FLOW*1..5]-> (m2)
RETURN collect(DISTINCT id(m1))
```

```
int stock = materialInStock();
boolean enough = isEnoughInStock(stock);
makeOrders(enough);
```

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (d1:Method) <-[:CALLS]-
(m1:MethodCall) -[cfs:CONTROL_FLOW*1..5]-> (m2:MethodCall)
-[:CALLS]-> (d2:Method)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE exists(cf.case))
AND NOT (m1) -[:DATA_FLOW*1..5]-> (m2)
AND NOT (m1) -[:DATA_FLOW]-> (:Field) <-[:DATA_FLOW]- (m2)
AND d1.isParallelisable=true
AND d2.isParallelisable=true
RETURN collect(DISTINCT id(m1))
```

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
MATCH (d1:Method) <-[:CALLS]-
(m1:MethodCall) -[cfs:CONTROL_FLOW*1..5]-> (m2:MethodCall)
-[:CALLS]-> (d2:Method)
WHERE m1.avgDurInMs > 200 AND m2.avgDurInMs > 200
AND none(cf IN cfs WHERE exists(cf.case))
AND NOT (m1) -[:DATA_FLOW*1..5]-> (m2)
AND NOT (m1) -[:DATA_FLOW]-> (:Field) <-[:DATA_FLOW]- (m2)
AND d1.isParallelisable=true
AND d2.isParallelisable=true
RETURN collect(DISTINCT id(m1))
```

```
readField();
writeField();
```

```
MATCH (d1:Method) <-[:CALLS]-
(m1:MethodCall) -[cfs:CONTROL_FLOW*1..5]-> (m2:MethodCall)
-[:CALLS]-> (d2:Method)
WHERE m1.avgDurationInMs > 200 AND m2.avgDurationInMs > 200
WITH m1, m2, d1, d2, cfs
MATCH path = (m1) -[:CONTROL_FLOW*1..5]-> (m2)
WITH m1, m2, d1, d2, cfs, filter (intermediateNode IN nodes(path)
                                  WHERE intermediateNode <> m1
                                  AND intermediateNode <> m2)
                          AS intermediateNodes
WHERE
  NOT (m1) -[:DATA_FLOW*1..5]-> (m2)
  AND none(cf IN cfs WHERE exists(cf.case))
  AND NOT (m1) -[:DATA_FLOW]-> (:Field) <-[:DATA_FLOW]- (m2)
  AND d1.isParallelisable=true
  AND d2.isParallelisable=true

  AND all(node IN intermediateNodes
          WHERE
              NOT (m1) -[:DATA_FLOW]-> (node)
              AND (NOT node:MethodCall
                  OR all(pathcall IN ((node) -[:CALLS]-> ())
                      WHERE all(call IN rels(pathcall)
                          WHERE endNode(call).isParallelisable=true
                              OR endNode(call):Constructor))))

RETURN collect(DISTINCT id(m1))
```

C | A | U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät


```
...
AND NOT d1.overridden=true AND NOT d2.overridden=true
AND (d1.isParallelisable=true
     OR NOT (d1) -[:AGGREGATED_FIELD_WRITE
                   |AGGREGATED_CALLS* ]-> (:Field) <-- (d2))
AND (d2.isParallelisable=true
    OR NOT (d2) -[:AGGREGATED_FIELD_WRITE
                   |AGGREGATED_CALLS* ]-> (:Field) <-- (d1))
...
```

- less restrictive: allow modification of fields except concurrently accessed ones
- Attention: handle overridden methods separately!

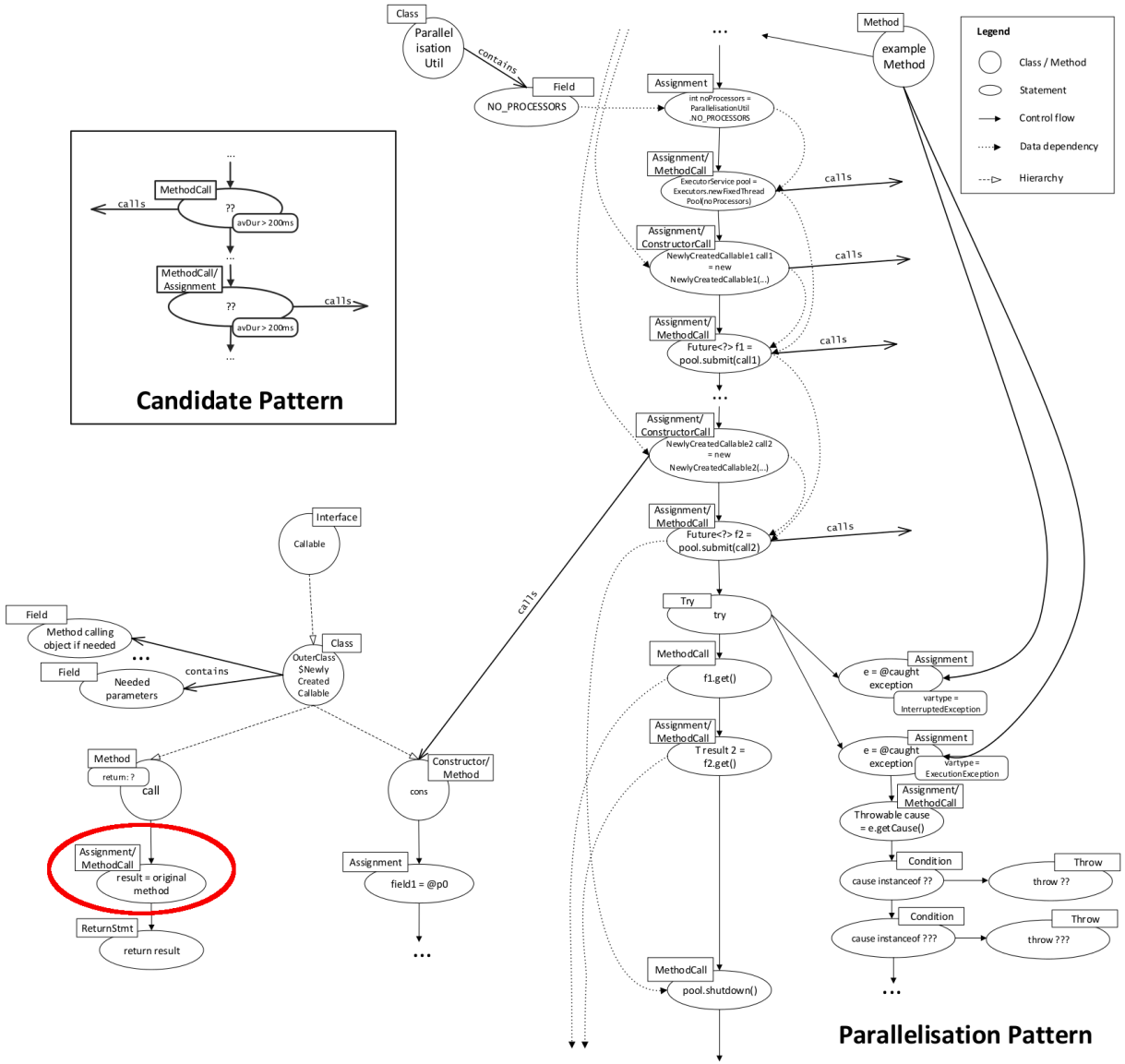
$\Rightarrow$ see details in thesis

Master Worker Pattern:

- Usage of nested Callables and Futures
  - Enables return value
  - Enables exception handling
- Organisation with Java's ExecutorsService (thread pool)

no Java 8 support (because of Soot)

Source: https://github.com/Sable/soot/issues/394


Candidate Pattern
MethodCall
calls
??
avDur > 200ms
MethodCall/ Assignment
Parallelisation Pattern
Class
Parallelisation Util
contains
Field
NO_PROCESSORS
Method
example Method
Legend
Class / Method
Statement
Control flow
Data dependency
Hierarchy
Assignment
int noProcessors = ParallelisationUtil .NO_PROCESSORS
Assignment/ MethodCall
ExecutorService pool = Executors.newFixedThreadPool(noProcessors)
Assignment/ ConstructorCall
NewlyCreatedCallable1 call1 = new NewlyCreatedCallable1(...)
Future<?> f1 = pool.submit(call1)
NewlyCreatedCallable2 call2 = new NewlyCreatedCallable2(...)
Future<?> f2 = pool.submit(call2)
Try
try
MethodCall
f1.get()
Assignment/ MethodCall
T result 2 = f2.get()
MethodCall
pool.shutdown()
Interface
Callable
Class
OuterClass $Newly Created Callable
Field
Method calling object if needed
Needed parameters
Method
return: ?
call
Assignment/ MethodCall
result = original method
ReturnStmt
return result
Constructor/ Method
cons
Assignment
field1 = @p0
Assignment
e = @caught exception
vartype = InterruptedException
Assignment
e = @caught exception
vartype = ExecutionException
Assignment/ MethodCall
Throwable cause = e.getCause()
Condition
cause instanceof ??
Throw
throw ??
Condition
cause instanceof ???
Throw
throw ???

- Add new nodes according to Soot representation
- Add new control flows and hierarchy
- Remove unused control flows
- New variable/class names (variable scope)
- Exception handling

- Add new nodes according to Soot representation
- Add new control flows and hierarchy
- Remove unused control flows
- New variable/class names (variable scope)
- Exception handling

Simplification/Optimisation:
no attention to data flows, instead new Soot run

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät

- Implementation in Java
- Cypher queries (from String)
- Neo4J Java API

- Implementation in Java
- Cypher queries (from String)
- Neo4J Java API

Advantages:

- Reusability of the queries
- Dynamic build of queries
- Temporary storing of nodes
  - ⇒ Comfortable handling of relationships

Live Demonstration

Solving Goal 4: Evaluating the Speed Up

Evaluation of the speed up not yet possible:

Solving Goal 4: Evaluating the Speed Up

Evaluation of the speed up not yet possible:

- Generation of Java source code from the Neo4J SDG is very complex

Evaluation of the speed up not yet possible:

- Generation of Java source code from the Neo4J SDG is very complex
  - Try-catch blocks
  - Throw exceptions
  - Loop

Evaluation of the speed up not yet possible:

- Generation of Java source code from the Neo4J SDG is very complex
  - Try-catch blocks
  - Throw exceptions
  - Loop

$\Rightarrow$ Approach differently evaluated

- Feasibility of the approach
- Quantitative occurrence of the candidate patterns
- Extendibility of the approach

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät

Yes, we can!
We successfully transformed three prototypes :)

Checkstyle and Findbugs

#	Checkstyle	Findbugs
Overall nodes in SDG	83619	140875
All classes in SDG	942	1357
All Methods in SDG (without constructors)	8759	11983
Classes in app	762	1160
Methods in app (without constructors)	6772	9392
Overridden methods in app	754	505

Independent Successive Method Calls Pattern

♯	Checkstyle	Findbugs
Classes in app	762	1160
Methods in app (without constructors)	6772	9392
Read-only methods	963	1953
Method calls	20664	39699
Read-only-method calls	1208	4078
Successive method calls	12870	29595
Successive read-only method calls	26	352
Independent successive method calls	551	–

Independent Successive Method Calls Pattern

♯	Checkstyle	Findbugs
Classes in app	762	1160
Methods in app (without constructors)	6772	9392
Read-only methods	963	1953
Method calls	20664	39699
Read-only-method calls	1208	4078
Successive method calls	12870	29595
Successive read-only method calls	26	352
Independent successive method calls	551	–

- without runtime information constraints
- branches excluded
- no allowance of ’isParallelisable=true’ for more flexibility

Independent For-Each Loop

♯	Checkstyle	Findbugs
Classes in app	762	1160
Methods in app (without constructors)	6772	9392
For-Each Loops	97	234*
Read-only For-Each Loops	16	75*
Independent For-Each Loops	39	103*

** loop body size restricted to a maximum of 30 statements*

Array Reduction Pattern

♯	Checkstyle	Findbugs
Classes in app	762	1160
Methods in app (without constructors)	6772	9392
Array length operation	717	636
Array access operation	614	982
Array assignments	2424	536
Array reduction	0	0

Pros

- Modular because of Java
- Reusability of queries
- Semi-automatism easily extendible
- Configurable, e.g. runtime constraints
- New candidate and parallelisation patterns can be designed with the help of existing utility classes

Cons

- When SDG changes, CMQs and transformation have to be adjusted

$\Rightarrow$ Maintenance effort, e.g. for Java 8 support

Conclusion:

- Successful implementation of 3 patterns
- Powerful, but complex approach

Conclusion:

- Successful implementation of 3 patterns
- Powerful, but complex approach

Future Work:

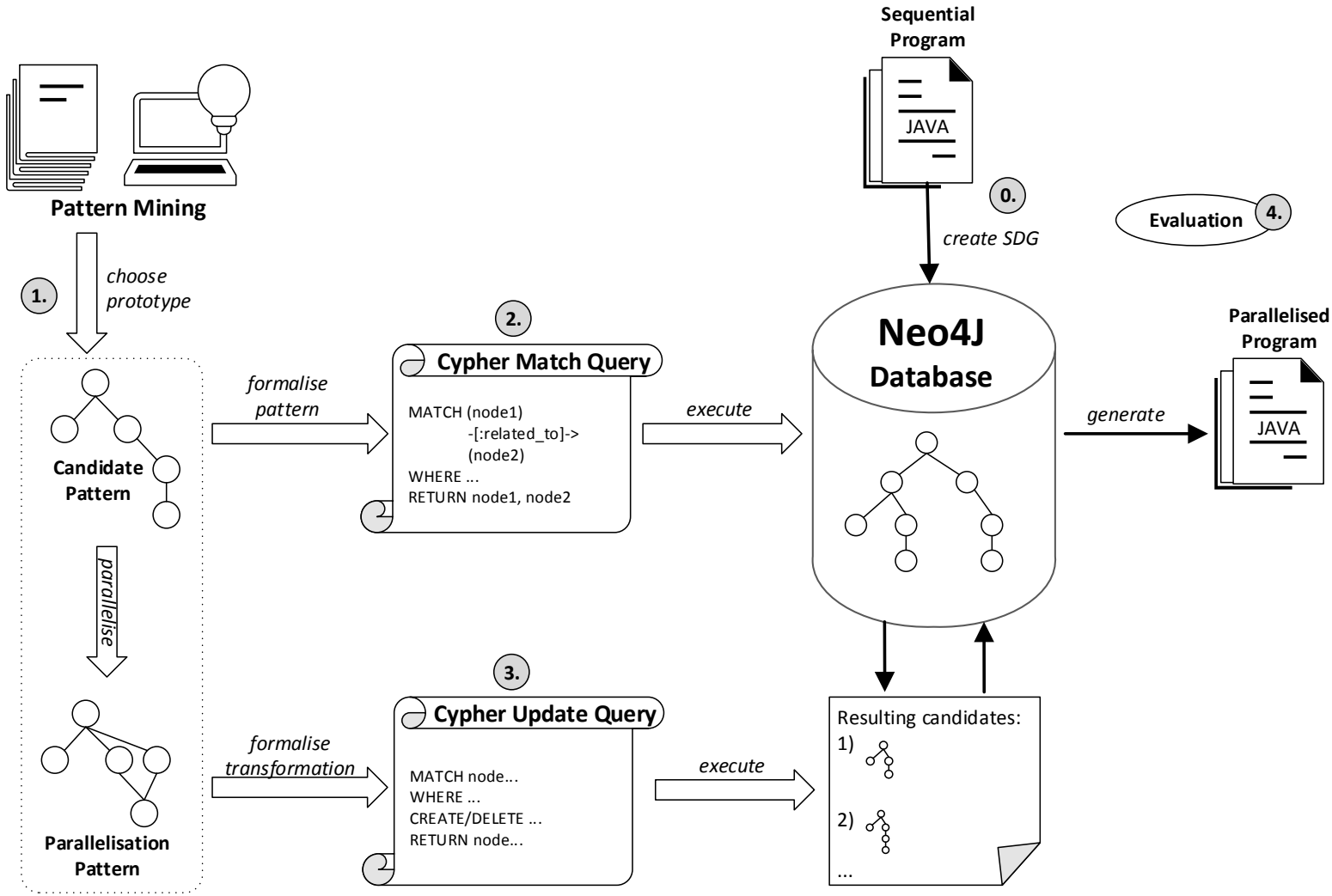
- Generate Java code from SDG for performance evaluation
- Add runtime information
- Implement additional candidate and parallelisation patterns

[Mattson04] T. G. Mattson, B. A. Sanders, and B. L. Massingill. Patterns for Parallel Programming. Second. Addison Wesley, 2004.

[Molitorisz12] K. Molitorisz, J. Schimmel, and F. Otto. Automatic Parallelization Using AutoFutures. English. In: Multicore Software Engineering, Performance, and Tools. 2012, pages 78–81.

[Wulf14] C. Wulf. Pattern-based detection and utilization of potential parallelism in software systems. In: *Software Engineering 2014, Fachtagung des GI-Fachbereichs Soft- waretechnik, 25. Februar - 28. Februar 2014, Kiel, Deutschland.* 2014, pages 229–232.

Approach for the Thesis



```
MATCH (m:Method) <-[:CONTAINS_METHOD]- (classOrInterface)
      <-[:EXTENDS|IMPLEMENTS*1..]- (subclass) -[:CONTAINS_M
WHERE m.displayname = method.displayname
SET m.overridden=true
```

In theory:

```
MATCH (m:Method)
WHERE
   NOT (exists(m.overridden) OR m.overridden <> true)
   AND NOT (m) -[:AGGREGATED_FIELD_WRITE|AGGREGATED_CALLS*]-
WITH m
SET m.isReadOnly=true
```

In our 'capped' SDG:

1. Cypher-Query:

```
MATCH (m:Method)
WHERE
   m.origin = 'APP'
   AND (NOT exists(m.overridden) OR m.overridden <> true)
   AND NOT (m) -[:AGGREGATED_FIELD_WRITE]-> (:Field)
   AND (NOT (m) -[:AGGREGATED_CALLS]-> (:Method))
WITH m
SET m.isReadOnly=true
```

2.-x. Cypher query

```
MATCH (mRO:Method) <-[:AGGREGATED_CALLS]- (m:Method)
WHERE
  mRO.isReadOnly=true
  AND NOT EXISTS(m.isReadOnly)
  AND (NOT EXISTS(m.overridden) OR m.overridden <> true)
  AND NOT (m) -[:AGGREGATED_FIELD_WRITE]-> (:Field)
  AND (all(path IN ((m) -[:AGGREGATED_CALLS]-> (:Method))
      WHERE all(method IN nodes(path)
          WHERE m = method
              OR method.isReadOnly=true)))
  WITH m
SET m.isReadOnly=true
```

```
...

int nProcessors = ParallelisationUtil.NUMBER_OF_PROCESSORS;
java.util.concurrent.ExecutorService pool = java.util.concurrent.Executors
                    .newFixedThreadPool(nProcessors);

DataSCConnectCallable callable1 = new ConnectionSetup.DataSCConnectCallable(dataSC);
java.util.concurrent.Future<?> future1 = pool.submit(callable1);

EventSCConnectCallable callable2 = new ConnectionSetup.EventSCConnectCallable(eventSC);
java.util.concurrent.Future<?> future2 = pool.submit(callable2);

try {
   future1.get();
   future2.get();
} catch (InterruptedException e) {
} catch (java.util.concurrent.ExecutionException e) {
   Throwable cause = e.getCause();
   if (cause instanceof Error) {
       throw (Error) cause;
   }
   if (cause instanceof RuntimeException) {
         throw (RuntimeException) cause;
   }
}
pool.shutdown();

...
```

```
private static class DataSCConnectCallable implements Callable<Void> {

  private IDataServerConnection dataSC;

  public DataSCConnectCallable(IDataServerConnection dataSC) {
     super();
     this.dataSC = dataSC;
  }

  @Override
  public Void call() throws Exception {
     dataSC.connect();
     return null;
  }
}

private static class EventSCConnectCallable implements Callable<Void> {

  private IEventServerConnection eventSC;

  public EventSCConnectCallable(IEventServerConnection eventSC) {
     super();
     this.eventSC = eventSC;
  }

  @Override
  public Void call() {
     eventSC.connect();
     return null;
  }
}
```