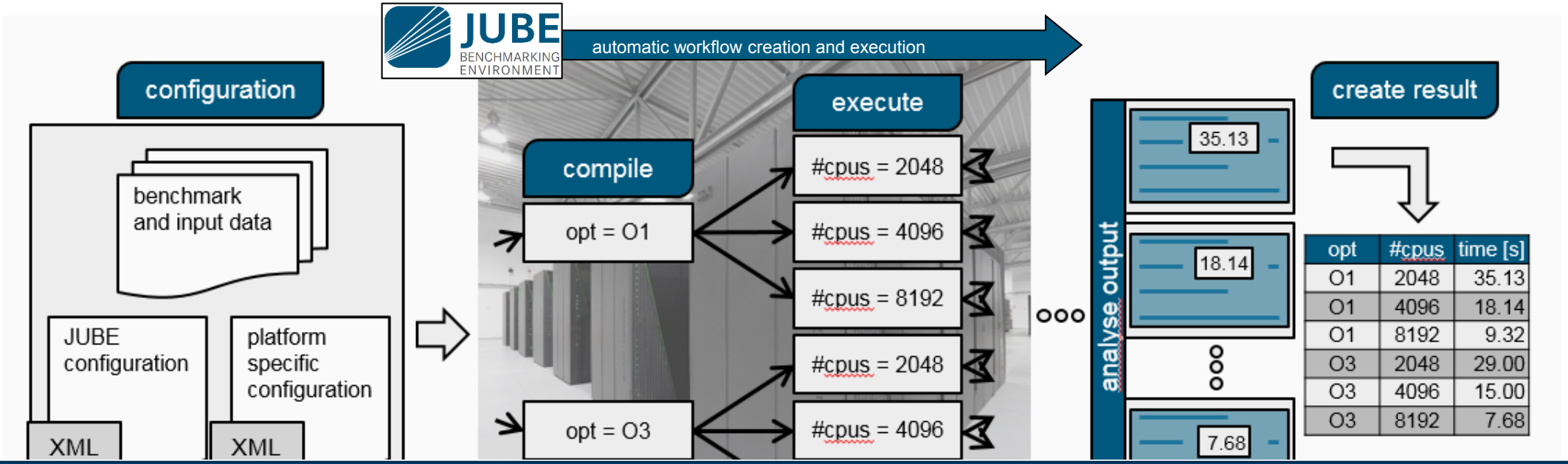



# BENCHMARKING AND PARAMETER STUDIES WITH JUBE

25 JUNE 2018 I SEBASTIAN LÜHRS

nest:: Conference 2018

# PARAMETER STUDIES ARE A LOT OF MANUAL WORK

I have to **touch multiple files** to change all my parameters.

A longer **workflow** is needed to perform my run properly.


Each run should be executed **independently** and in a **reproducible** way.

I can't **remember the parametrization** I used in one of my runs.

I want to use a **HPC** system.

I want to **reuse as much as possible** when switching to a different platform or to a different application.

I need an **overview** about my parametrization and the results.

# WHAT IS JUBE?

Generic, lightweight, configurable environment to run, monitor and analyse application execution in a systematic way

Can be used for:

- Benchmarking,
- Parameter studies,
- Testing,
- Profiling,
- Production scenarios

Alternatives

*- or -*

*Are you a potential JUBE candidate?*

- Manual execution
  - *Easy to use*
  - *Time-consuming*
  - *Very error-prone*
- Application specific script solution
  - *Optimized*
  - *Changes can be time-consuming*
  - *Portability problems*

# WORKSHOP ON BENCHMARKING AND PARAMETER STUDIES

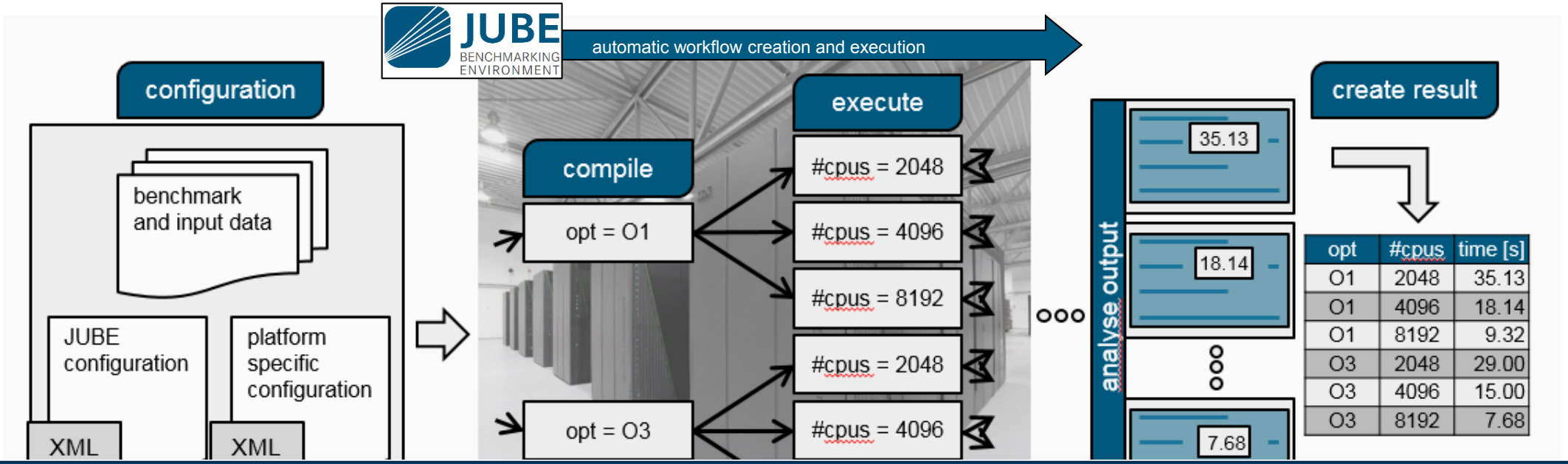
**Introduction and tutorial to the JUBE environment**

- *Try JUBE by yourself and learn how to use JUBE for your use case.*

nest:: **and** JUBE BENCHMARKING ENVIRONMENT

- Discussion on how JUBE can be used to perform NEST parameter studies, benchmarking scenarios and future use cases
- *Bring in your requirements.*

*See you at the JUBE workshop session*

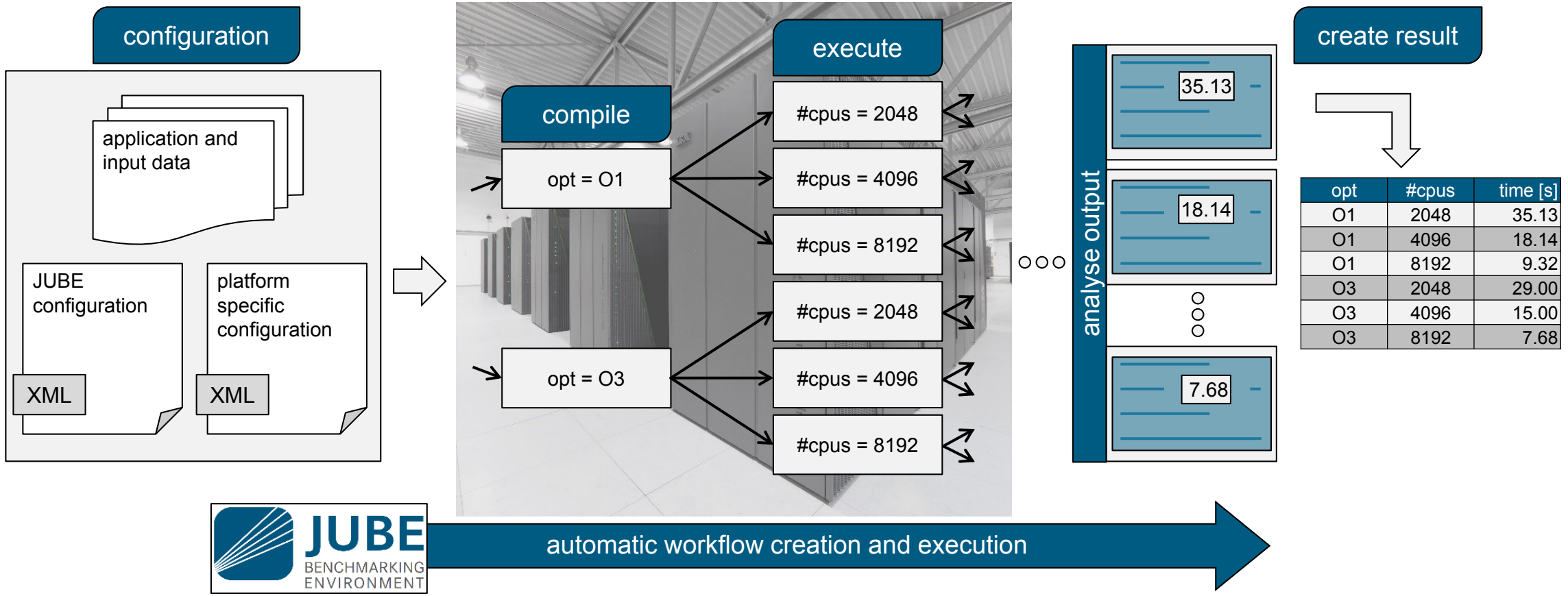



# BENCHMARKING AND PARAMETER STUDIES WITH JUBE

25 JUNE 2018 I SEBASTIAN LÜHRS

nest:: Conference 2018

# WHAT IS JUBE?


configuration
application and input data
JUBE configuration
XML
platform specific configuration
XML
compile
opt = O1
opt = O3
execute
#cpus = 2048
#cpus = 4096
#cpus = 8192
#cpus = 2048
#cpus = 4096
#cpus = 8192
analyse output
35.13
18.14
7.68
create result
opt | #cpus | time [s]
O1 | 2048 | 35.13
O1 | 4096 | 18.14
O1 | 8192 | 9.32
O3 | 2048 | 29.00
O3 | 4096 | 15.00
O3 | 8192 | 7.68
JUBE BENCHMARKING ENVIRONMENT
automatic workflow creation and execution

# JUBE AVAILABILITY

Download, Tutorials and Documentation:

- **www.fz-juelich.de/jsc/jube**

Prerequisites:

- OS: Linux
- Python 2.6, Python 2.7, Python 3.2 (or a more recent version)

Contact:

- jube.jsc@fz-juelich.de

# HOW TO BECOME A JUBE EXPERT

# COMMAND LINE ACCESS

Start a new benchmark run

- `jube run benchmark.xml`

Continue an existing benchmark run

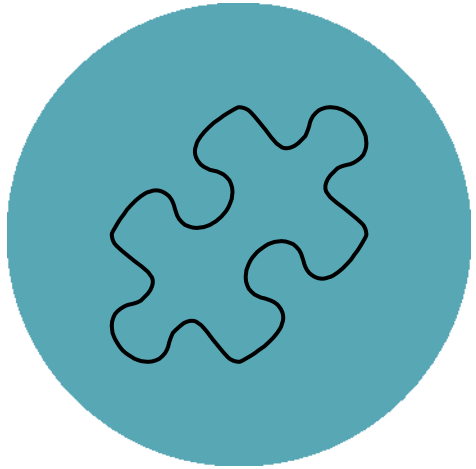
- `jube continue benchmark_dir [--id <id>]`

Analyse the benchmark data

- `jube analyse benchmark_dir [--id <id>]`

Create and show result representation

- `jube result benchmark_dir [--id <id>]`



# DEMO - HELLO_WORLD

# HELP?!


## Online documentation and tutorial

- `www.fz-juelich.de/jsc/jube`

## Info mode

- `jube info benchmark_dir [--id <id>] [--step <stepname>]`

## Command line accessible glossary

- `jube help <keyword>`

## Logs

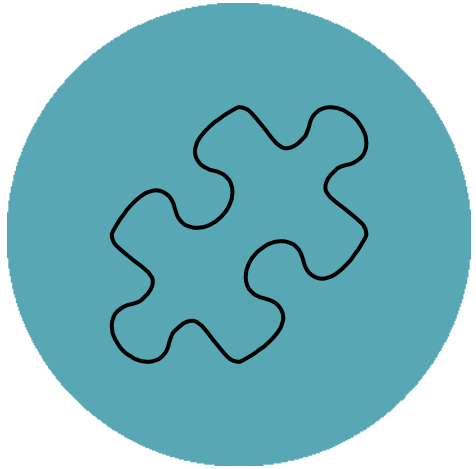
- `jube log benchmark_dir [--id <id>] [--command <cmd>]`

## Debug mode

- `jube --debug run|continue|analyse|result …`

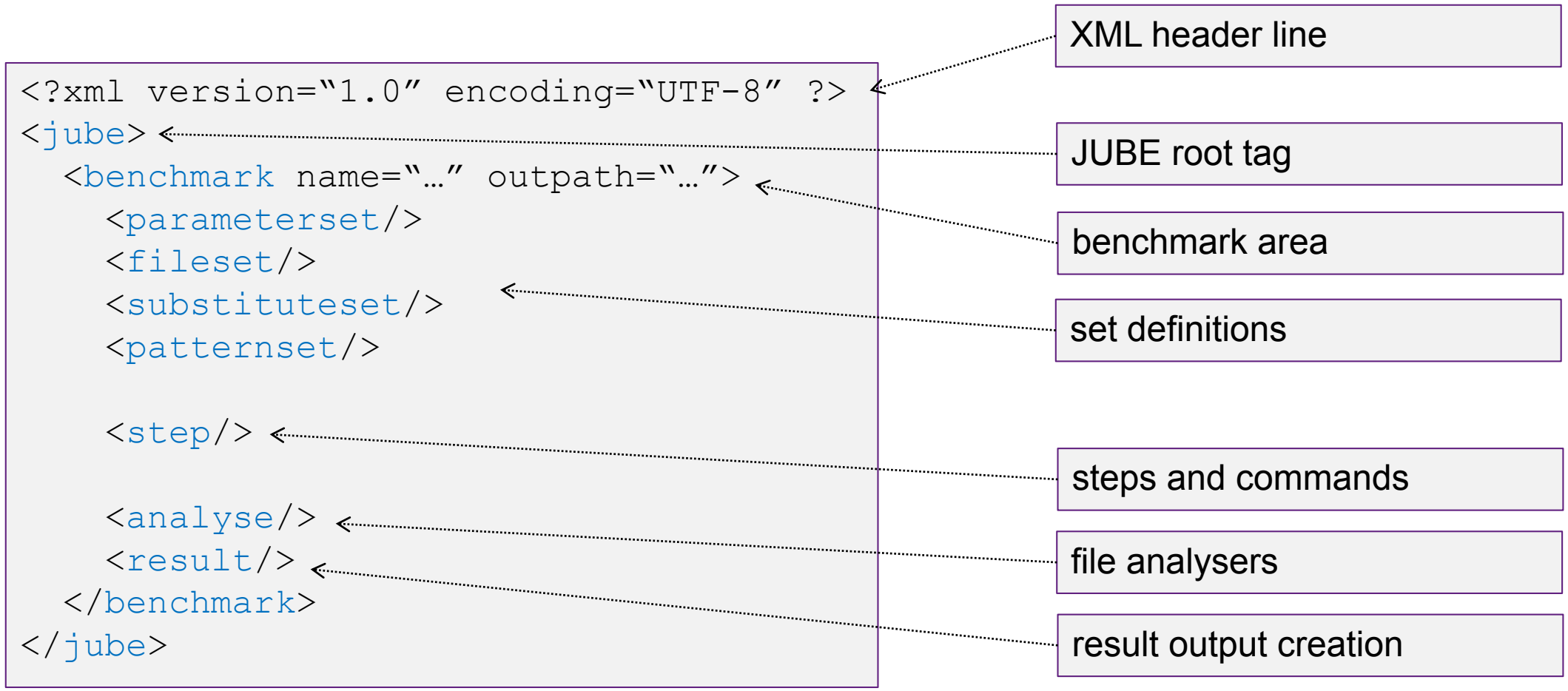
## Verbose mode

- `jube -v[vv] run …`



# DEMO - HELP

# HOWTO: GENERAL FILE LAYOUT




```
>> jube help general_structure
```

# XML INPUT FILE FORMAT

- XML: Extensible Markup Language
- XML must be well formed:
  - Only one root element: `<jube>`
  - `<a><b>…</a></b>` not allowed
  - Every tag must be closed (<a>…</a> or <a/>)
  - `<a attr1="…" attr1="…"/>` not allowed
  - `<a attr=… />` not allowed (missing "")
- Normal XML comment syntax can be used:
  - `<!-- … -->`
- JUBE tags can be validated using available DTD, schema, or RELAX NG file

# HOW TO: SETS

- Main JUBE information storage technique
- Four different types of sets are available
  - `<parameterset>` Parameter storage
  - `<fileset>` Define all available files
  - `<substituteset>` Define file substitution
  - `<patternset>` Define the analyse pattern
- Set names must be unique
- Can be initialised by using an additional configuration file
- Available `<parameterset>, <fileset> and <substituteset>` are used and combined within a `<step>`
- Available `<patternset>` are used within `<analyse>`

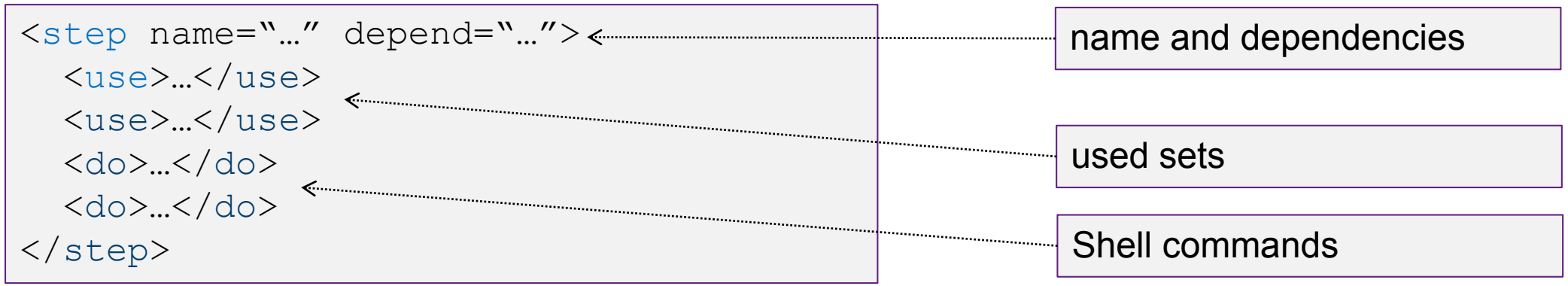
```
>> jube help <setname>_tag
```

# HOW TO: COMMAND EXECUTION

- `<do>...</do>` holds the executable commands
- All commands must use SHELL syntax (they will be executed by using `/bin/sh`)
- JUBE parameter can be used by using `$parametername`
- Parameter will be expanded in a pre-processing step
- Environment parameter can also be used
- JUBE stops execution if the command's return code fails
- Commands will only be executed once
- All `<do>` within the same `<step>` shares the same environment

```
>> jube help <setname>_tag
```

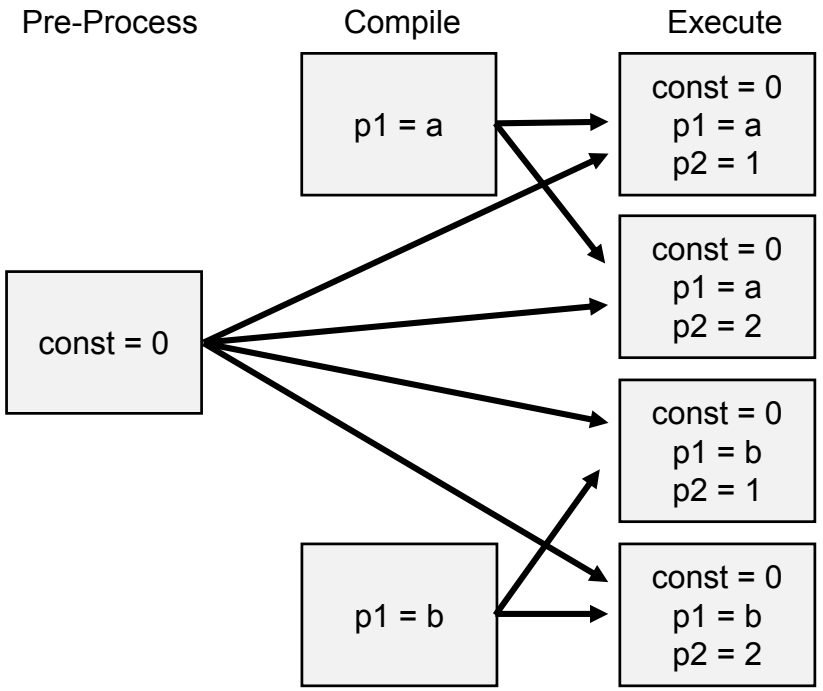
# HOW TO: COMMAND EXECUTION



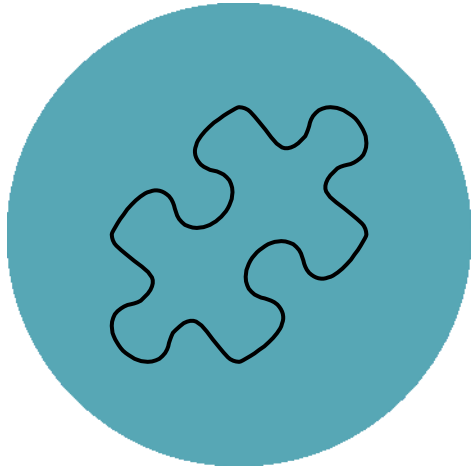

```
>> jube help step_tag
  >> jube help do_tag
```

# KEY CONCEPT: WORKFLOW CREATION

- Dependency driven step structure
- Parameter based expansion of steps



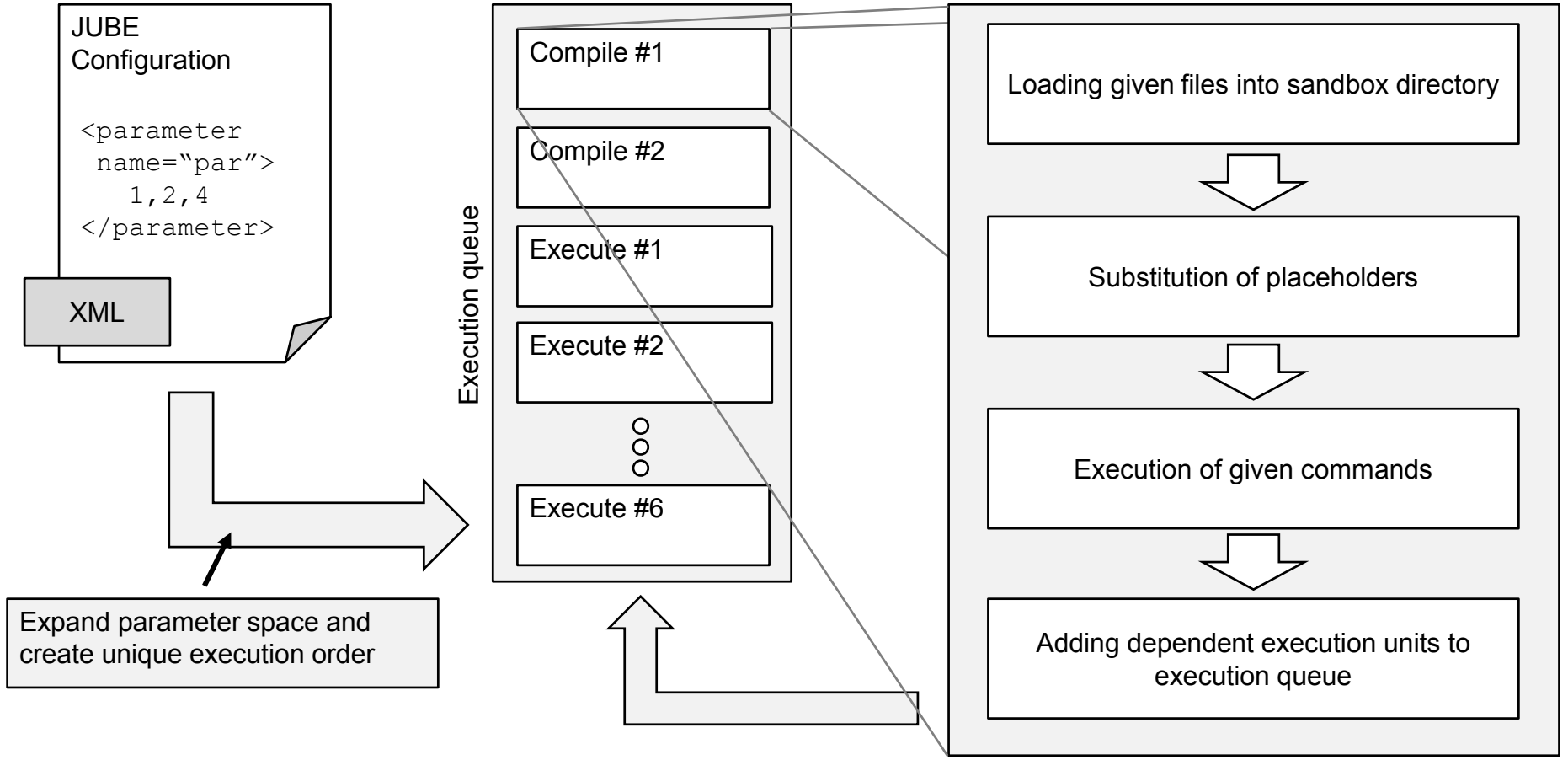

```
<parameterset name="preset">
  <parameter name="const">0</parameter>
</parameterset>
<parameterset name="compset">
  <parameter name="p1">a,b</parameter>
</parameterset>
<parameterset name="execset">
  <parameter name="p2">1,2</parameter>
</parameterset>

<step name="preprocess">
  <use>preset</use>
</step>
<step name="compile">
  <use>compset</use>
</step>
<step name="execute"
 depend="preprocess,combile">
  <use>execset</use>
</step>
```



# DEMO - DEPENDENCIES

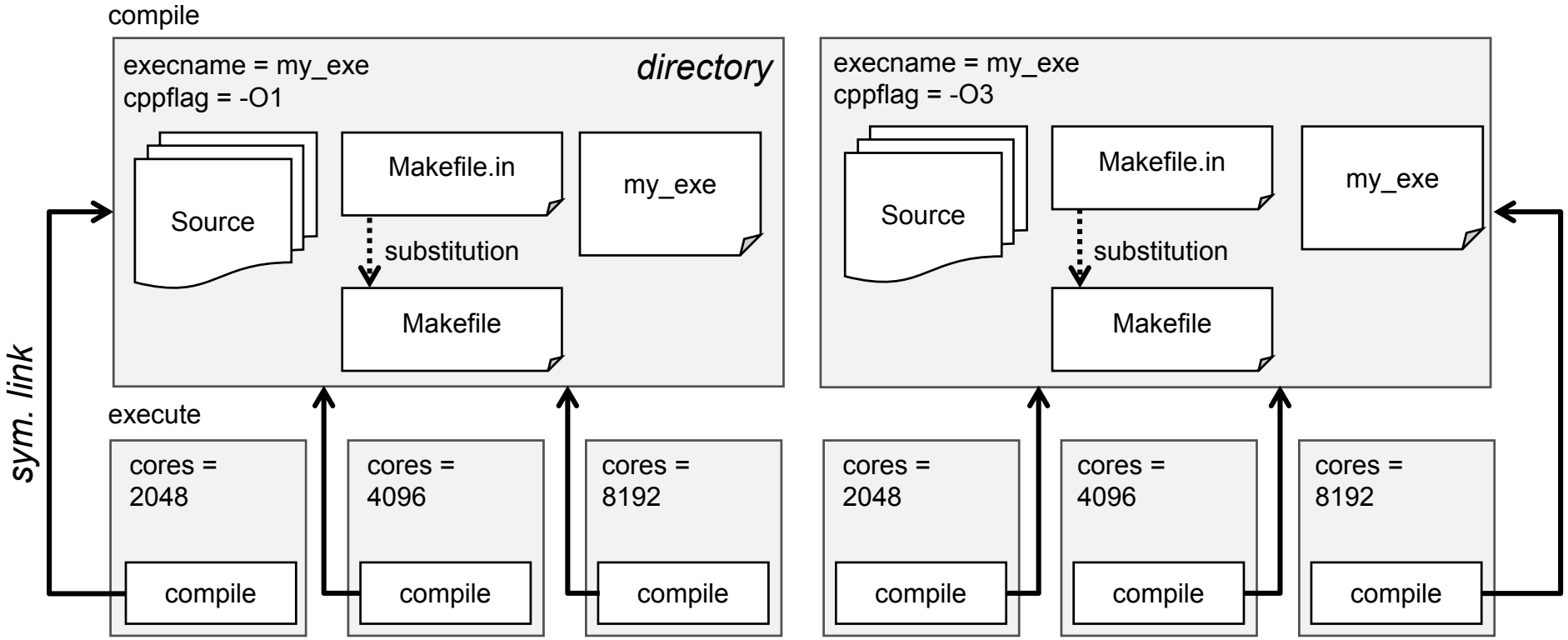
# EXECUTION ORDER

- Internal execution queue creates unique execution order



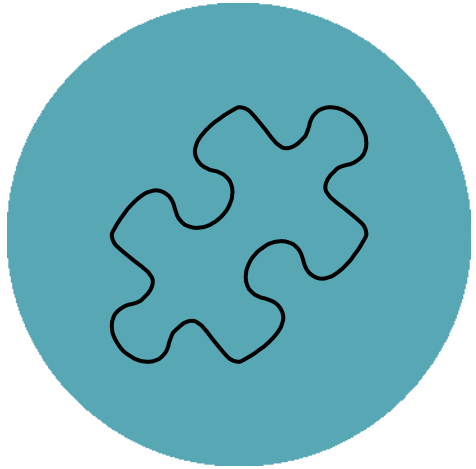
# KEY CONCEPT: DIRECTORY AND DATA HANDLING

- Each parameter/step combination runs in a separate sandbox directory
- Dependent steps can be accessed using sym. links





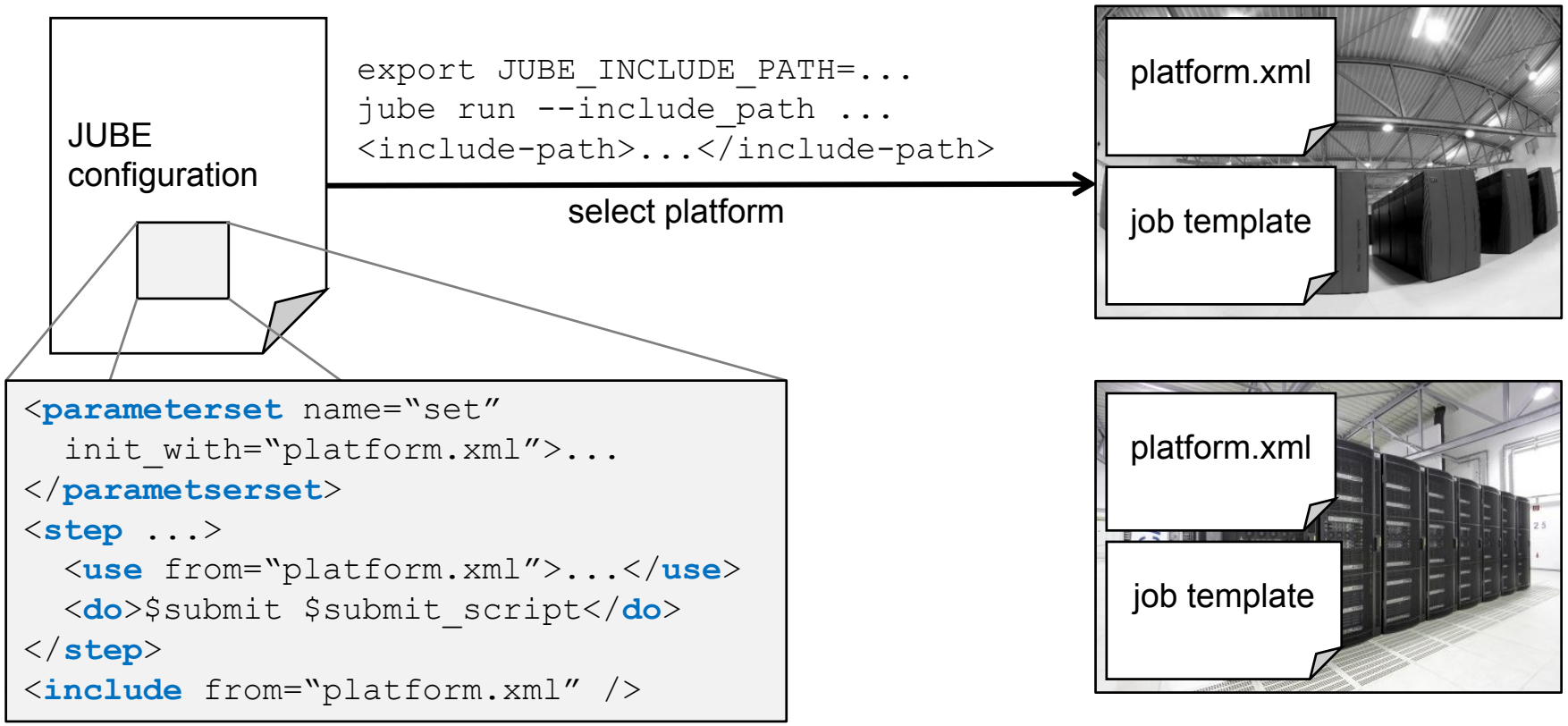

JÜLICH Forschungszentrum

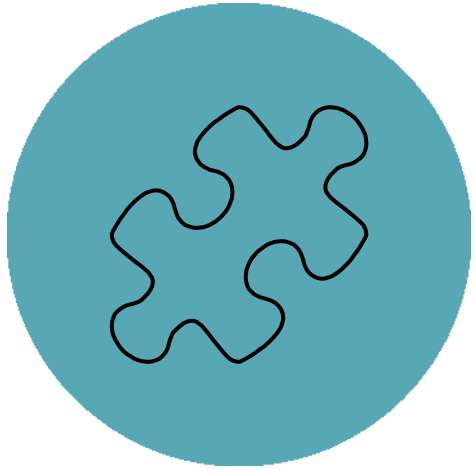


# DEMO - FILES_AND_SUB

# KEY CONCEPT: CONFIG REUSAGE

- e.g. separation of platform dependent and independent configuration options

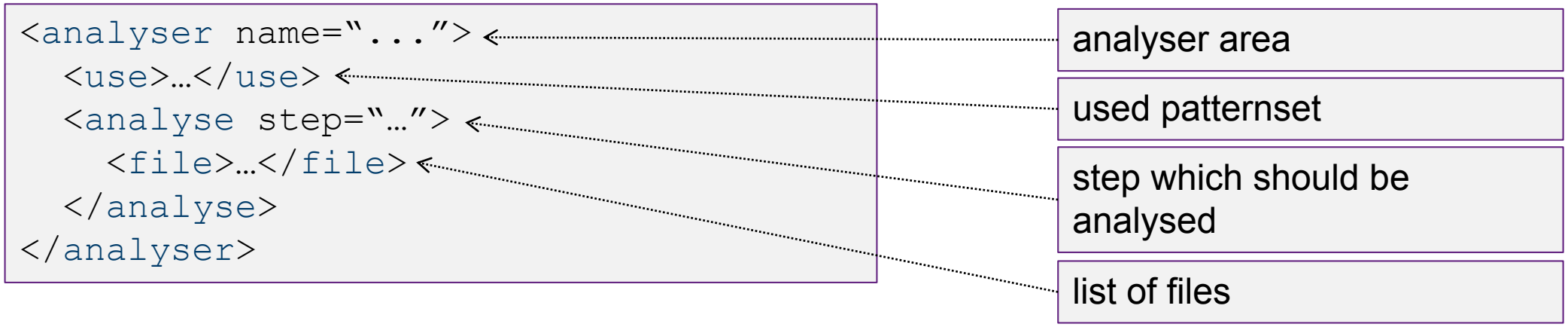




# DEMO - INCLUDE

# HOW TO: ANALYSE

- Files will be analysed by using regular expressions which are defined by the given patterns
- Multiple occurrences of the same pattern create statistical values (average, minimum, maximum etc.)




```
>> jube help analyser_tag
```

# HOW TO: RESULT CREATION

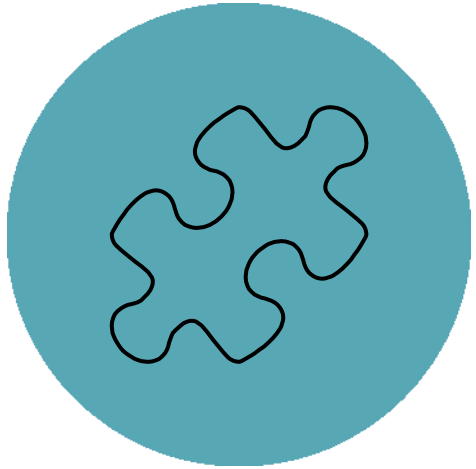
```
<result>
  <use>…</use>
  <table name="…">
    <column>…<column>
  </table>
</result>
```

result area

table result type definition

column definition

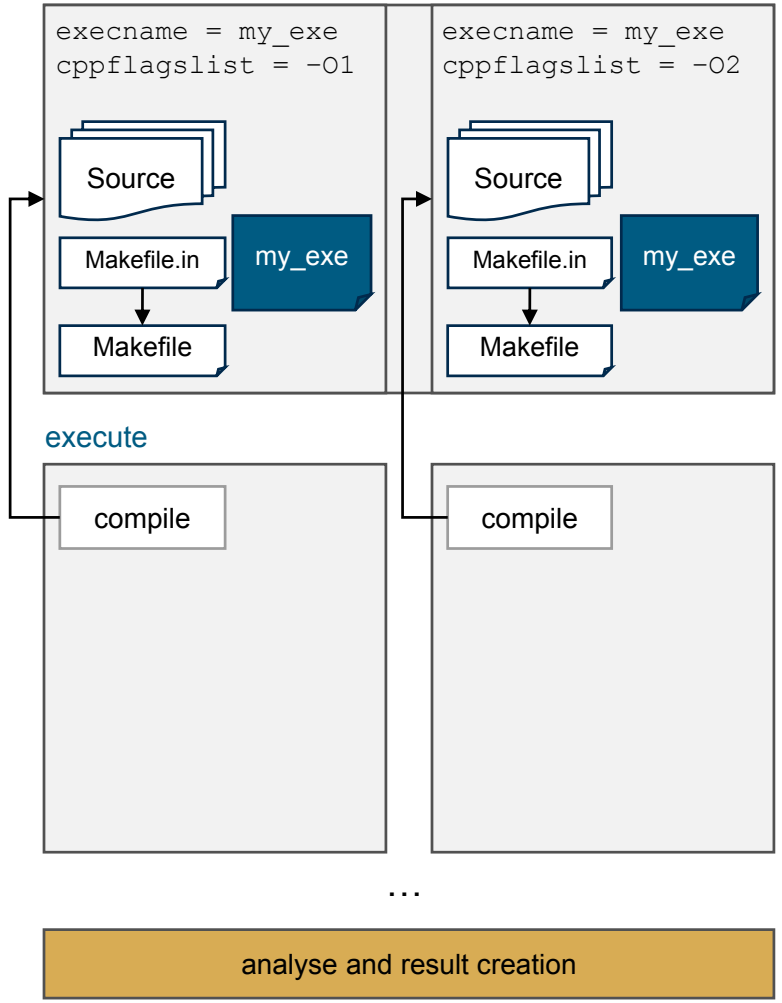
```
>> jube help result_tag
 >> jube help table_tag
```



# DEMO - RESULT_CREATION

# JUBE GENERAL WORKFLOW

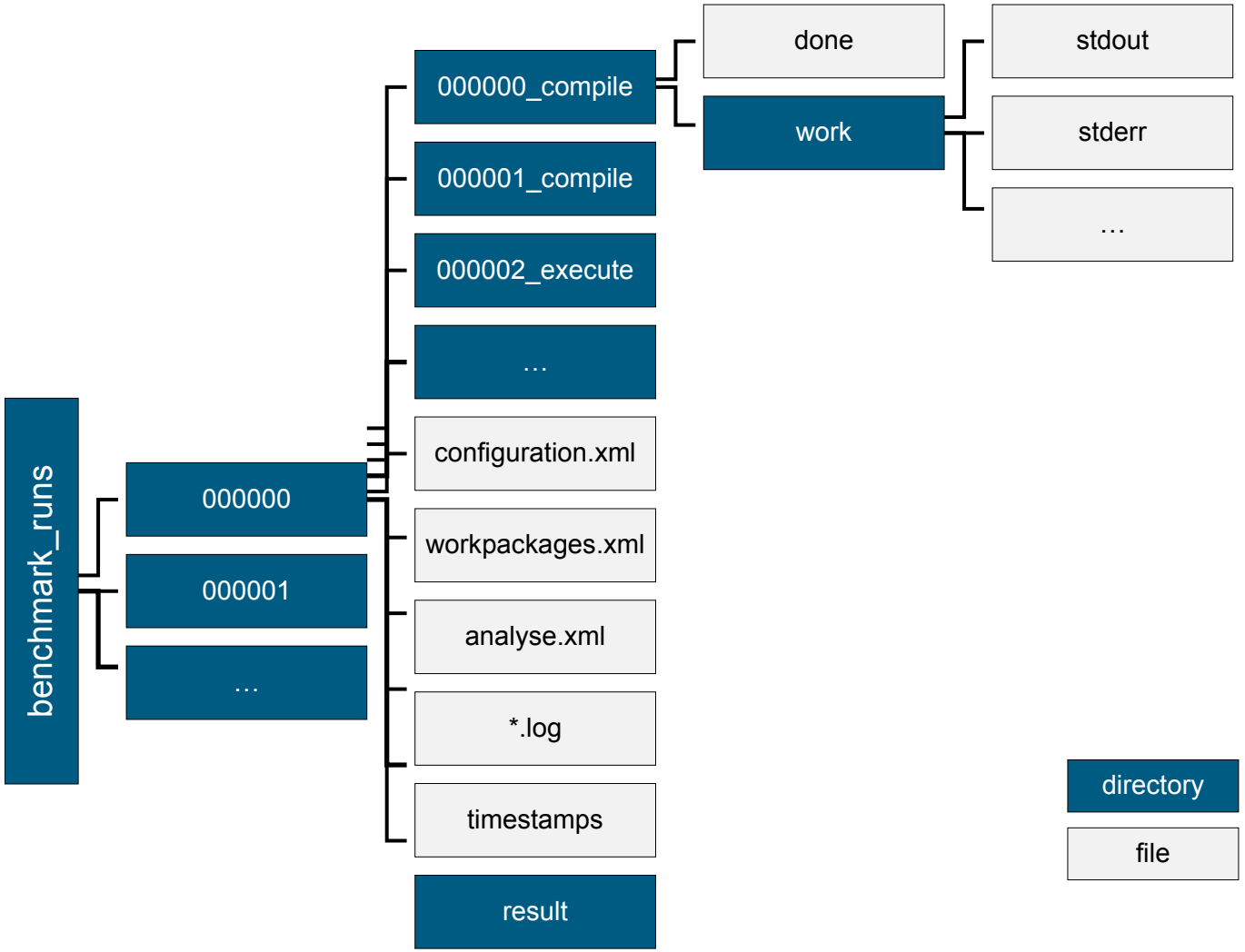
```
<jube>
  <benchmark name="bench" outpath="./benchmark_runs">
    <parameterset name="compileset">
      <parameter name="execname">my_exe</parameter>
      <parameter name="cppflagslist">
        -O1,-O2
      </parameter>
    </parameterset>

    <fileset name="sources">
      <copy>src/*</copy>
    </fileset>

    <substituteset name="compilesub">
      <iofile in="Makefile.in" out="Makefile" />
      <sub source="#PROGNAME#" dest="$execname" />
    </substituteset>

    <step name="compile">
      <use>compileset</use>
      <use>sources</use>
      <use>compilesub</use>
      <do>make OPT=$cppflagslist</do>
    </step>

    <step name="execute" depend="compile">
      …
```





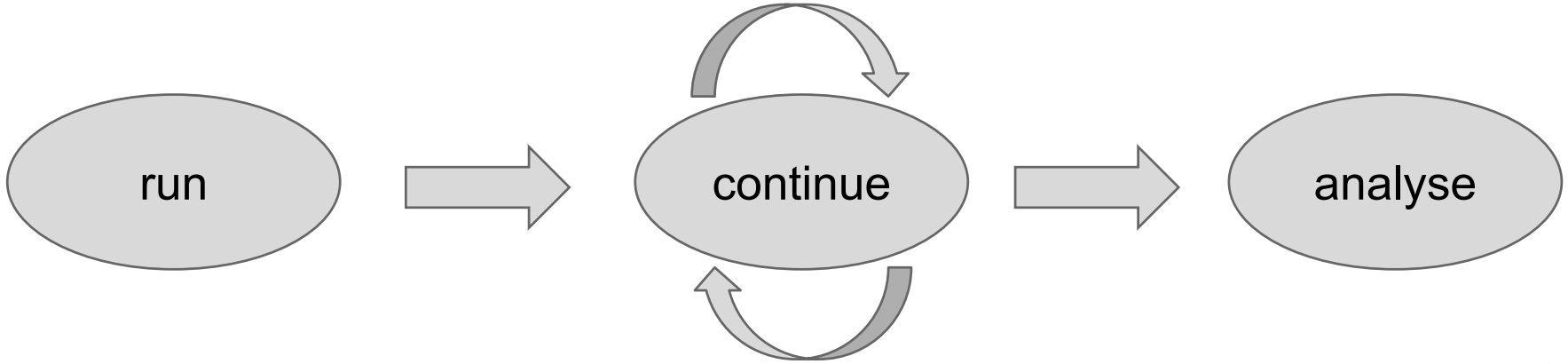

JÜLICH Forschungszentrum

# DIRECTORY STRUCTURE

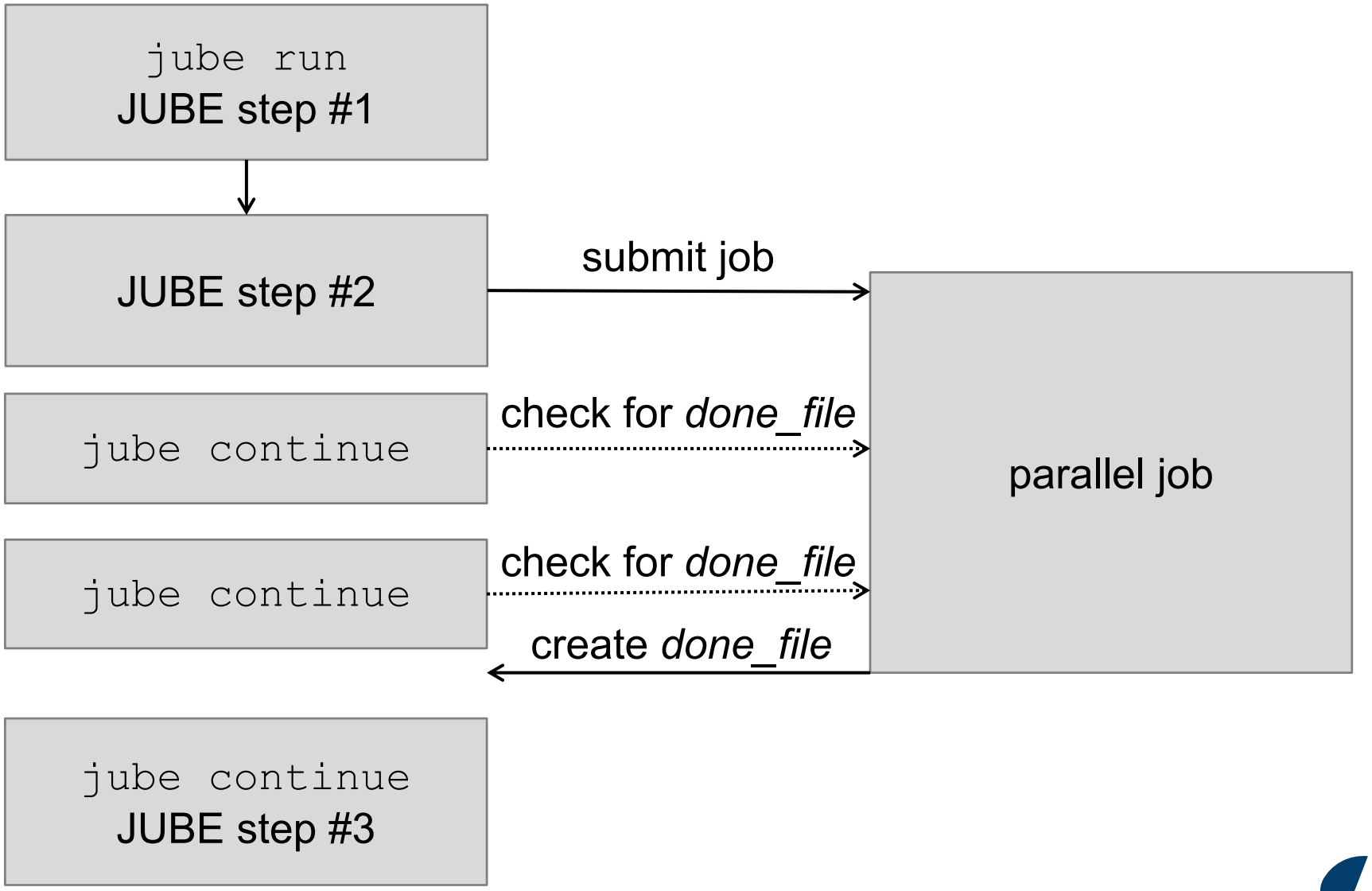


# HOW TO: JOB SUBMISSION

- A job template and substitution can be used to generalize the job submission process
- `<do>…</do>` is used to submit the job
- `<do>` returns immediately after the job was submitted. To wait for its execution use: `<do done_file=„…“>`
- The marker file, given by `done_file`, must be generated by the job script after the parallel part was executed
- `jube continue` triggers JUBE to check all available marker files



# HOW TO: JOB SUBMISSION



# JUBE IN CONTEXT OF BENCHMARKING

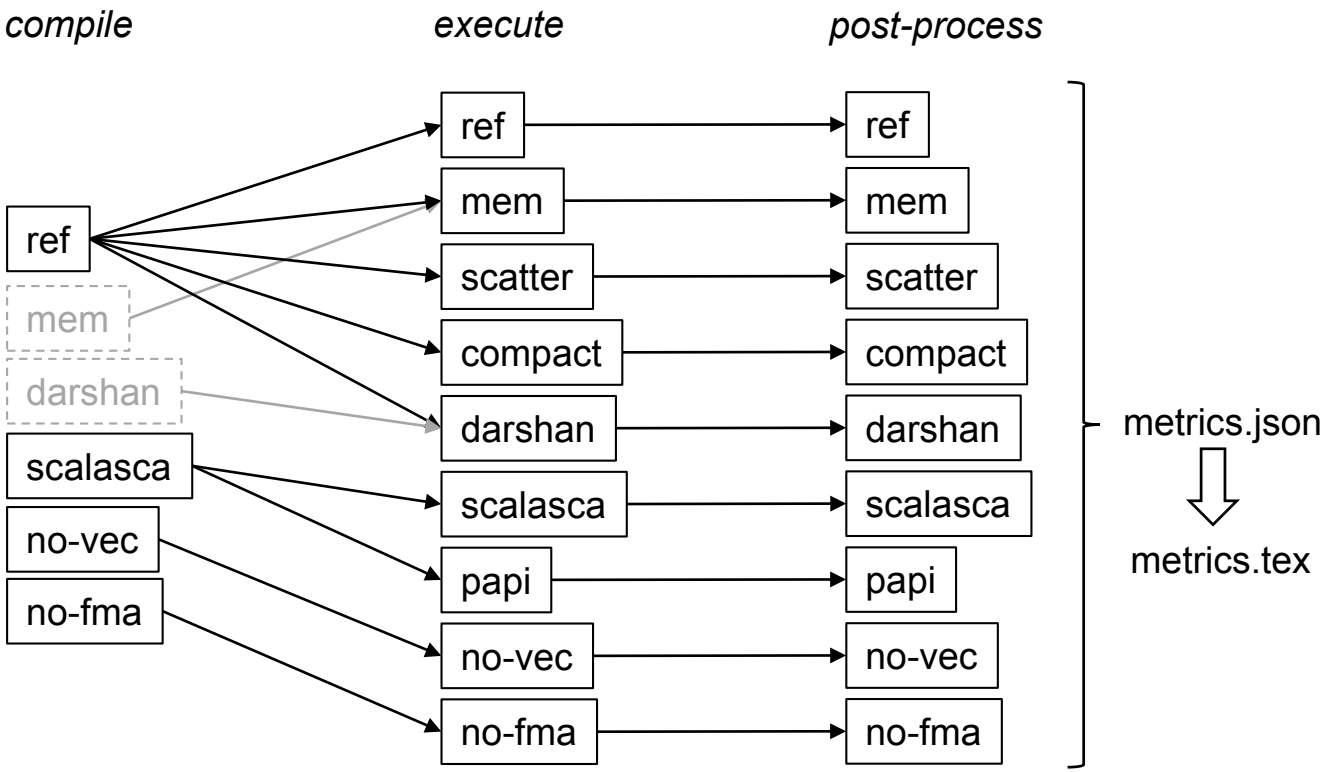
Example DEEP-Projects:

- Collecting benchmarking tests to compare:
  - The different I/O approaches used in DEEP-ER.
  - The I/O performance in DEEP-ER with respect to production systems.

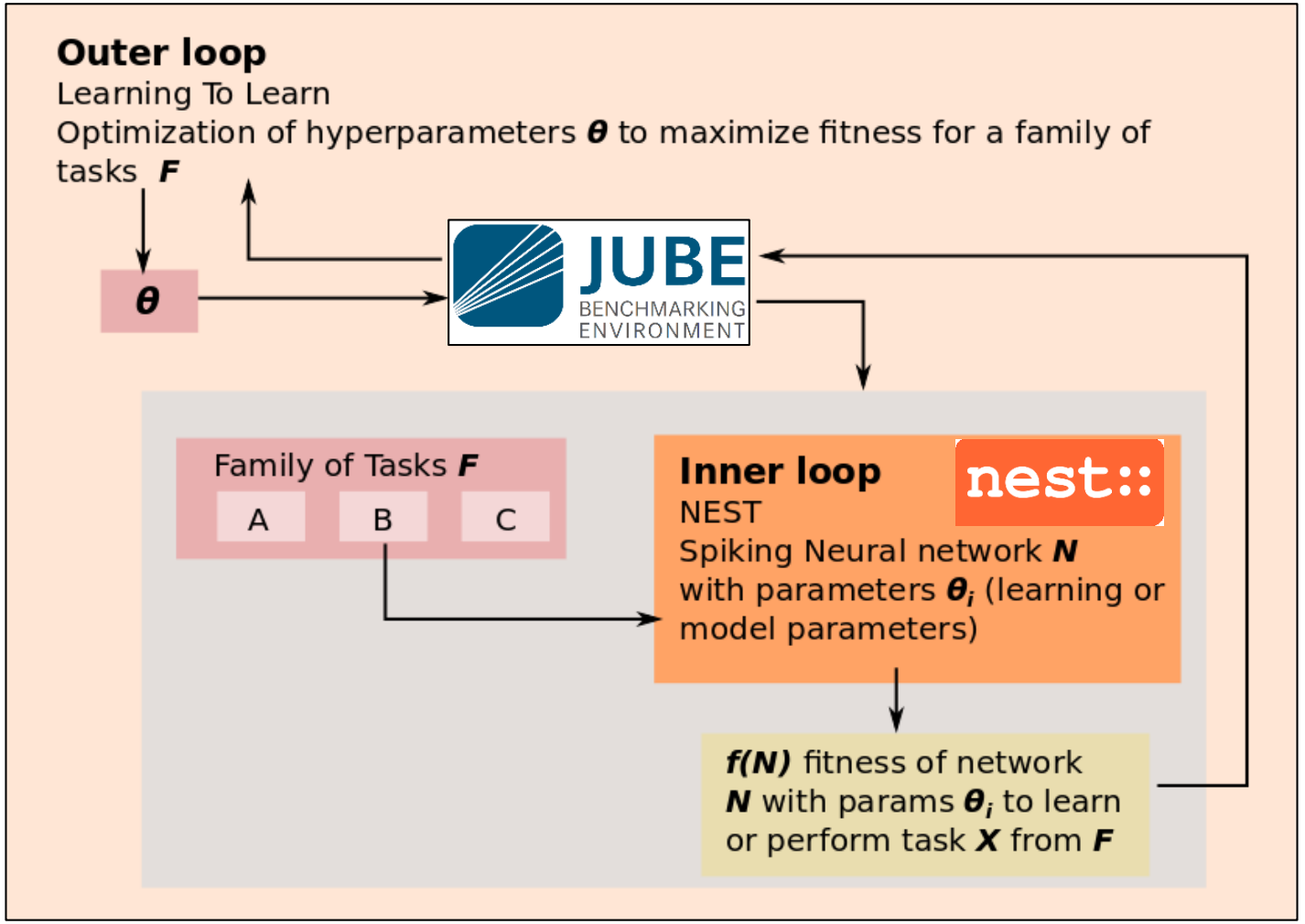
# JUBE IN CONTEXT OF PERFORMANCE ANALYSIS

Example EoCoE:

- Reproducible, comparable setup to extract performance metrics for different applications
- Orchestration of different performance evaluation tools and techniques



# JUBE IN CONTEXT OF PARAMETER STUDIES



# LEARNING TO LEARN WITH JUPEX

Learning to Learn concepts reapplied

- Optimizer: an optimization algorithm such as evolutionary strategies, gradient descent, simulated annealing, etc.
- Optimizee: a function or model to be optimized. The fitness of the execution of the function/model with specific parameters provided by the optimizer on each iteration is assessed and given as result to the optimizer. Could be also non neuroscientific models.

JUPeX joins the power of L2L with JUBE and Unicore

- With JUPeX the parameters to be explored by JUBE are set to the ones provided by the L2L optimizer
- Unicore is a framework for the deployment of workflows on HPC

JUPeX can be used to execute long training runs for L2L models

It can also be used to perform parameter and hyperparameter optimizations over large spaces

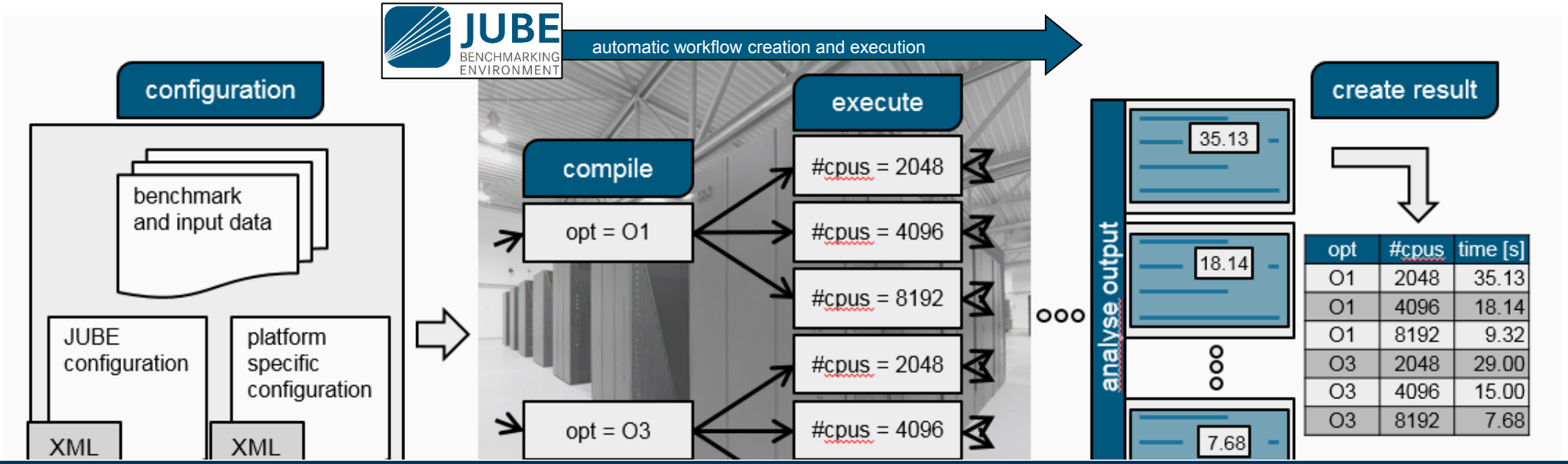
Questions: slns@fz-juelich.de

nest:: AND JUBE BENCHMARKING ENVIRONMENT

Potential use cases

- Benchmarking
- Parameter studies
  - Example: Learning to learn and JUPeX

*What is your use case?*
*What are your requirements?*

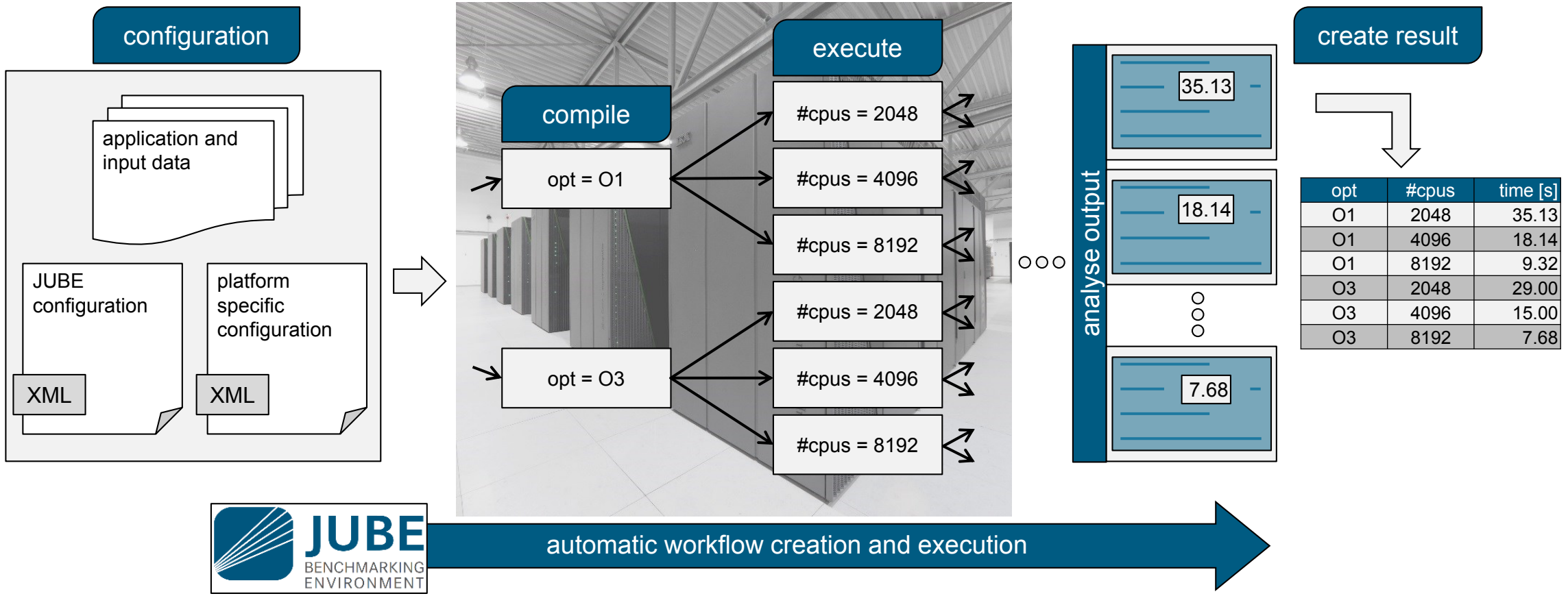



# BENCHMARKING AND PARAMETER STUDIES WITH JUBE

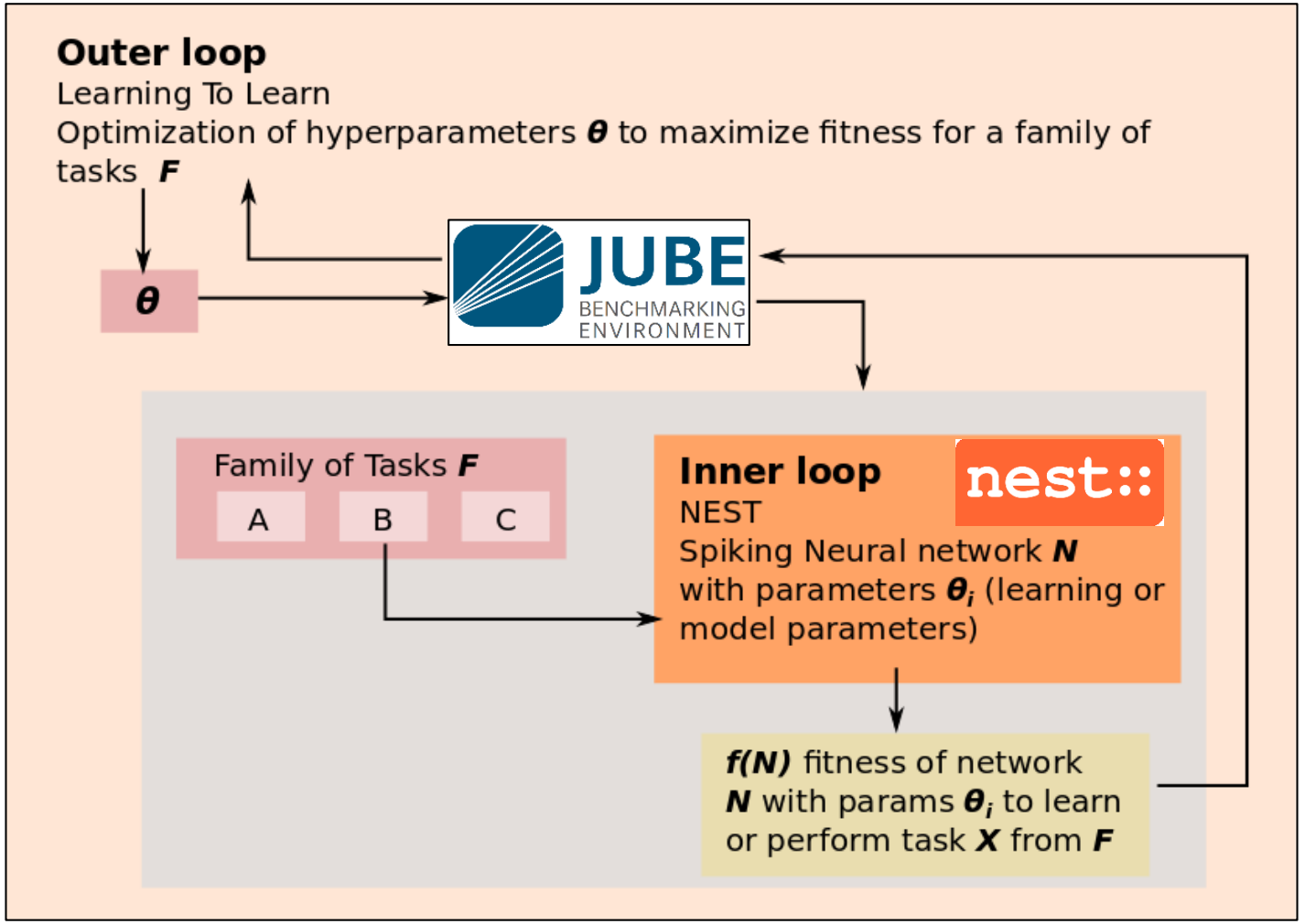
26 JUNE 2018 I SEBASTIAN LÜHRS

nest:: Conference 2018

# WHAT IS JUBE?


configuration
application and input data
JUBE configuration
XML
platform specific configuration
XML
compile
opt = O1
opt = O3
execute
#cpus = 2048
#cpus = 4096
#cpus = 8192
#cpus = 2048
#cpus = 4096
#cpus = 8192
analyse output
35.13
18.14
7.68
create result
opt | #cpus | time [s]
O1 | 2048 | 35.13
O1 | 4096 | 18.14
O1 | 8192 | 9.32
O3 | 2048 | 29.00
O3 | 4096 | 15.00
O3 | 8192 | 7.68
JUBE BENCHMARKING ENVIRONMENT
automatic workflow creation and execution

# JUBE IN CONTEXT OF PARAMETER STUDIES




Learning to learn with JUPeX



JÜLICH Forschungszentrum

# JUBE AVAILABILITY

Download, Tutorials and Documentation:

- **www.fz-juelich.de/jsc/jube**

Prerequisites:

- OS: Linux
- Python 2.6, Python 2.7, Python 3.2 (or a more recent version)

Contact:

- jube.jsc@fz-juelich.de