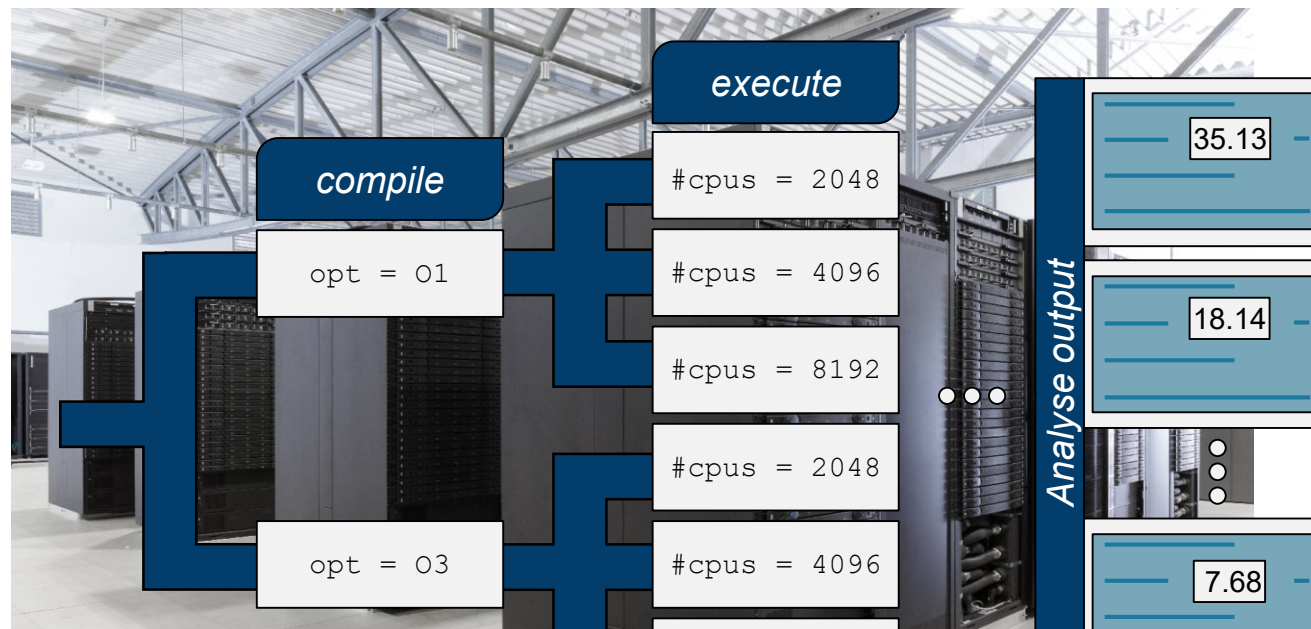
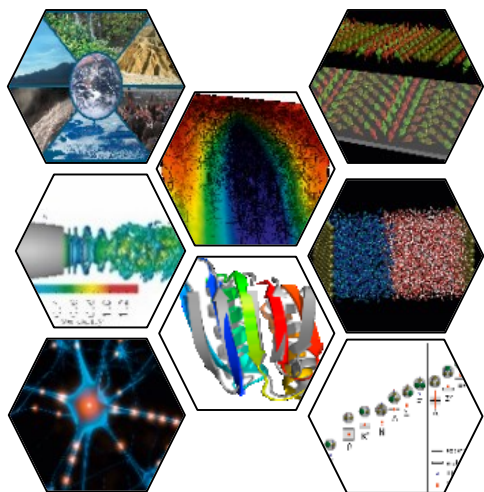




<i>opt</i>	<i>#cpus</i>	<i>time [s]</i>
01	2048	35.13
01	4096	18.14
01	8192	9.32
03	2048	29.00
03	4096	15.00
03	8192	7.68

AUTOMATED BENCHMARKING WITH



19 JUNE 2020, HPCKP'20 | SEBASTIAN LÜHRS, S.LUEHRS@FZ-JUELICH.DE

JÜLICH SUPERCOMPUTING CENTRE
FORSCHUNGSZENTRUM JÜLICH GMBH

WHAT IS JUBE AND WHY CAN IT HELP?

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

- App. specific    script replacement.

Can be used for:

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

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

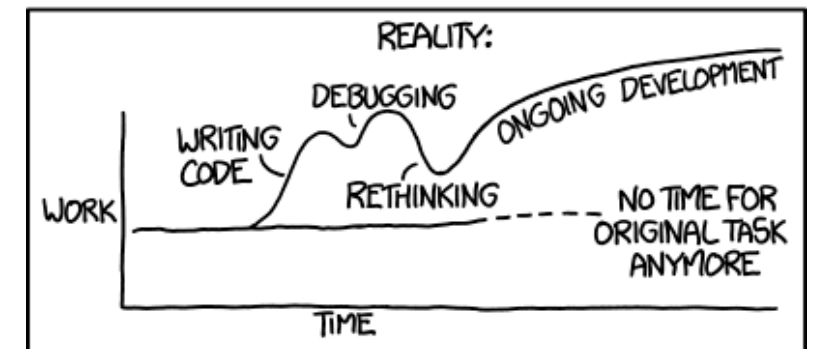
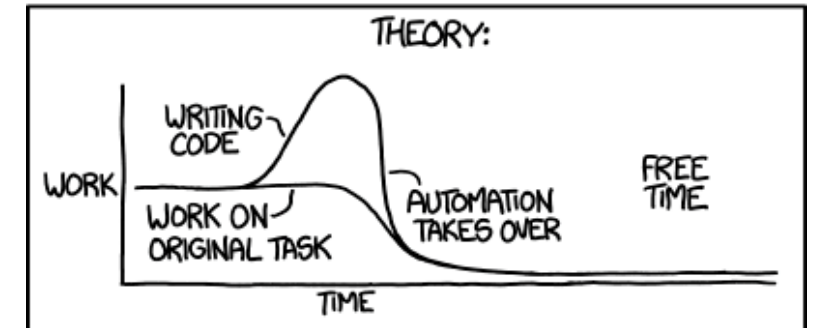
	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS

<https://xkcd.com/1205/>
19 June 2020

WHAT JUBE IS NOT?

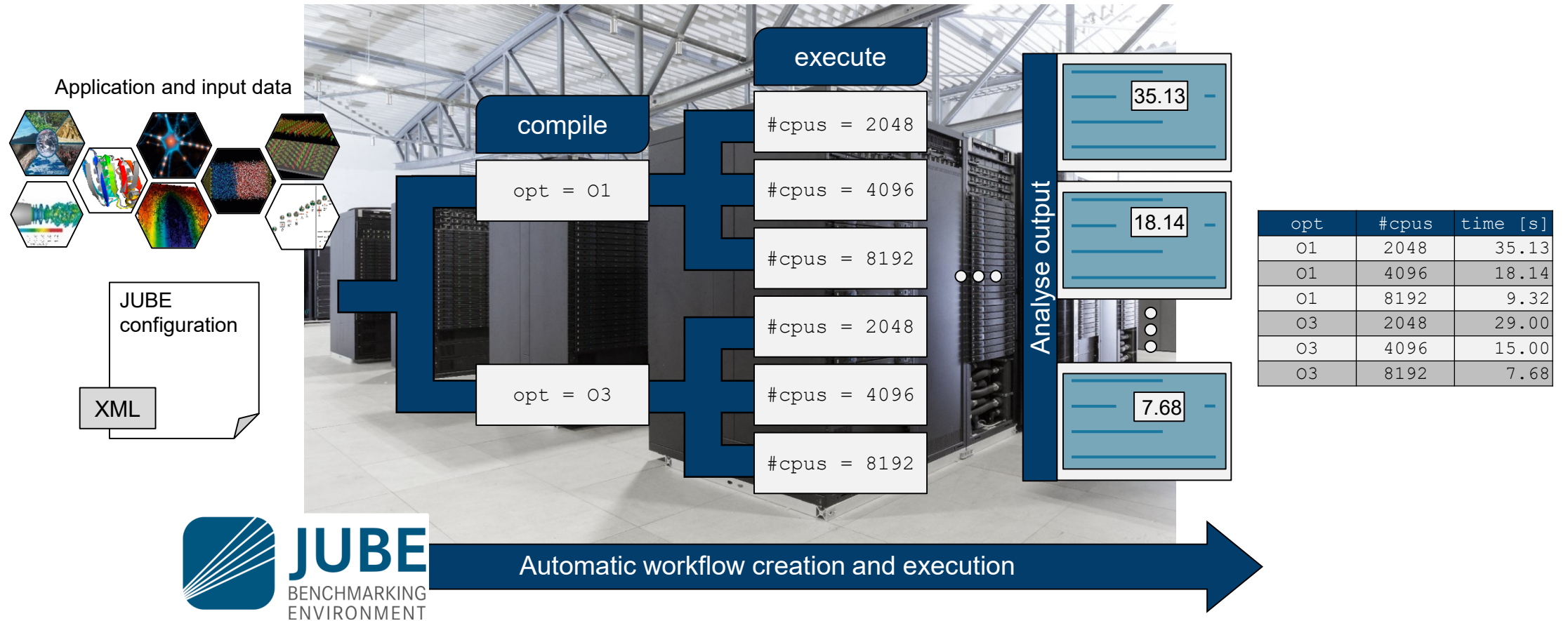
- A profiling tool: *Metrics are extracted not generated*
- A building tool: *JUBE will follow the same process as done manually → use EasyBuild or Spack*
- Free lunch

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



<https://xkcd.com/1319/>

WHAT IS JUBE?





HOW TO START A JUBE INTEGRATION? *TODAY'S EXAMPLE: IOR*

*IOR is a parallel IO benchmark that can be used to test the performance of parallel storage systems using various interfaces and access patterns:
<https://github.com/hpc/ior>*

```
<?xml version="1.0" encoding="UTF-8"?>
<jube>
  <benchmark name="ior" outputpath="./runs">
    <step name="execute">
      <do>ior</do>
    </step>
  </benchmark>
</jube>
```

Yes it is XML ☹️

Assuming `ior` is sitting in `$PATH`

Benchmark output directory

```
> jube run ior.xml
```

```
#####
# benchmark: ior
# id: 0
#
#
#####

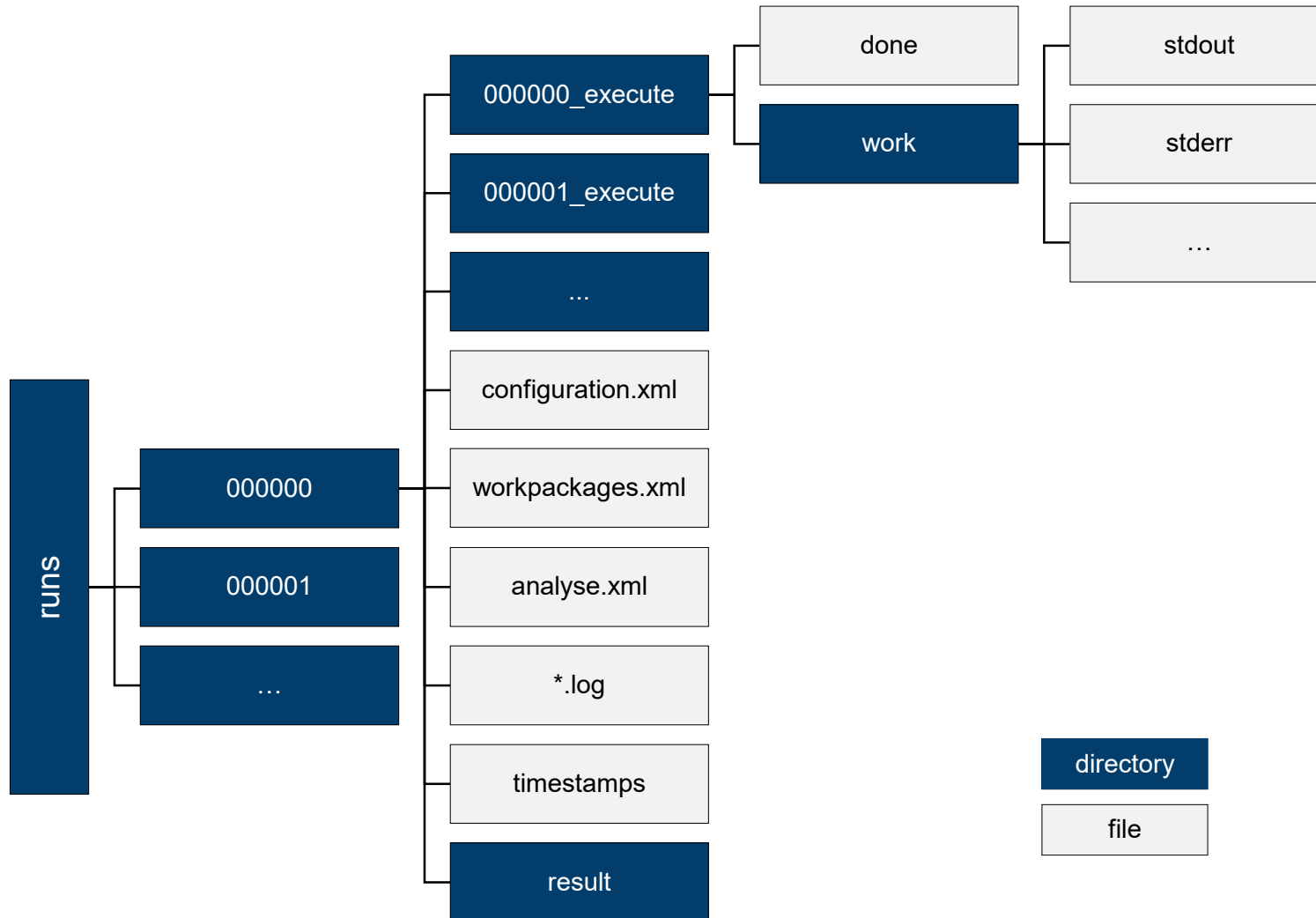
Running workpackages (#=done, 0=wait, E=error):
##### ( 1/ 1)

stepname | all | open | wait | error | done
-----+-----+-----+-----+-----+-----
execute | 1 | 0 | 0 | 0 | 1

>>>> Benchmark information and further useful commands:
>>>> id: 0
>>>> handle: runs
>>>> dir: runs/000000
>>>> analyse: jube analyse runs --id 0
>>>> result: jube result runs --id 0
>>>> info: jube info runs --id 0
>>>> log: jube log runs --id 0
#####
```

Individual run ID

DIRECTORY STRUCTURE



```

<?xml version="1.0" encoding="UTF-8"?>
<jube>
  <benchmark name="ior" outpath="./runs">

    <parameterset name="ior_parameter">
      <parameter name="blocksize">100m</parameter>
      <parameter name="transfersize">
        1m,10m,100m
      </parameter>
    </parameterset>

    <step name="execute">
      <use>ior_parameter</use>
      <do>ior -b $blocksize -t $transfersize</do>
    </step>
  </benchmark>
</jube>

```

Three
configurations to
test

Not an environment
variable

```
> jube run ior.xml
```

```

#####
# benchmark: ior
# id: 1
#
#
#####

Running workpackages (#=done, 0=wait, E=error):
##### ( 3/ 3)

  stepname | all | open | wait | error | done
  -----+-----+-----+-----+-----+-----
  execute |   3 |    0 |    0 |    0 |    3

```

Three individual
runs

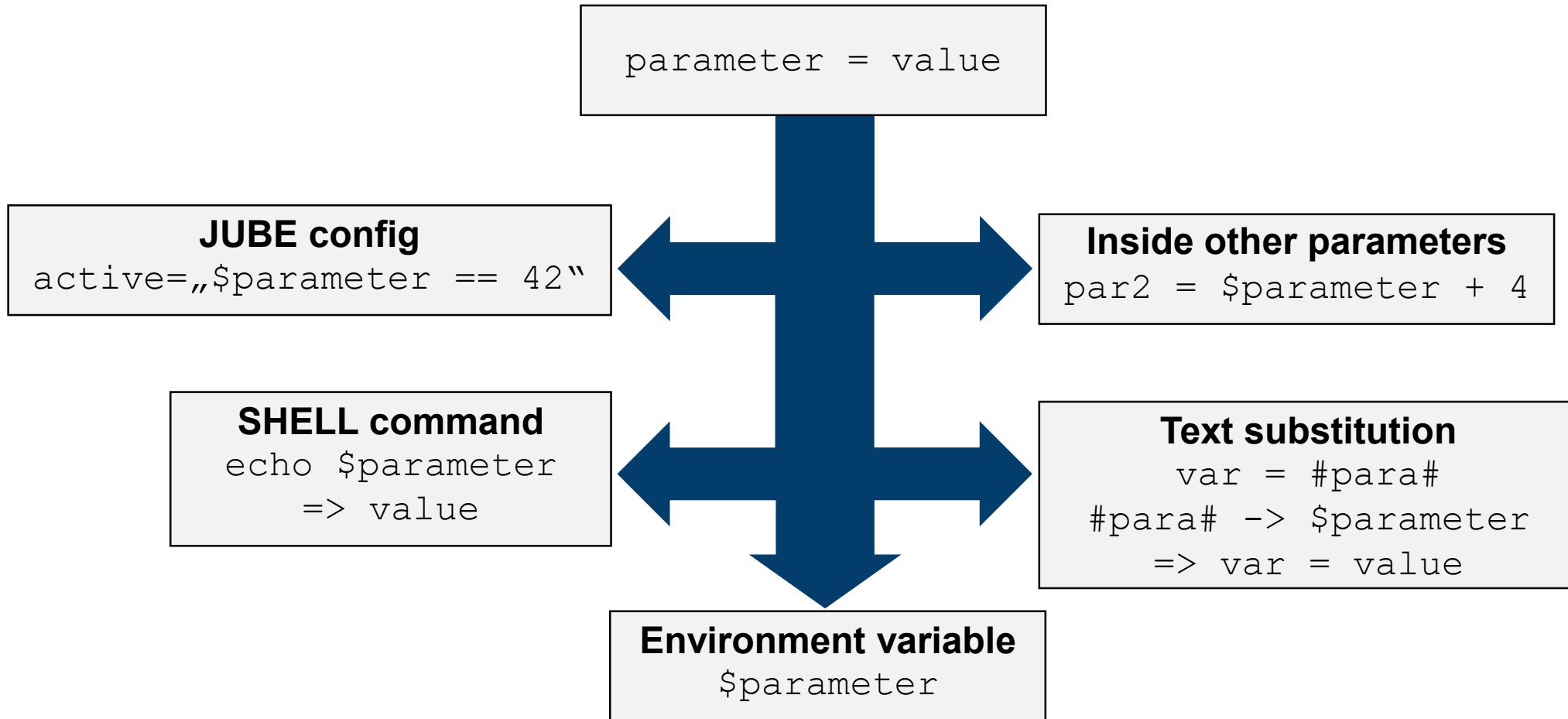
```

>>>> Benchmark information and further useful commands:
>>>>      id: 1
>>>>    handle: runs
>>>>      dir: runs/000001
>>>>  analyse: jube analyse runs --id 1
>>>>   result: jube result runs --id 1
>>>>     info: jube info runs --id 1
>>>>      log: jube log runs --id 1
#####

```

PARAMETER SUBSTITUTION

A configured parameter can be used in different places:



```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<jube>
```

```
  <benchmark name="ior" outpath="./runs">
```

```
    <parameterset name="ior_parameter">
```

```
      <parameter name="blocksize">100m</parameter>
```

```
      <parameter name="transfersize">
```

```
        1m,10m,100m
```

```
      </parameter>
```

```
    </parameterset>
```

```
  <fileset name="source">
```

```
    <copy>ior</copy>
```

```
  </fileset>
```

```
  <step name="compile">
```

```
    <use>source</use>
```

```
    <do work_dir="ior">
```

```
      ./bootstrap & & ./configure
```

```
    </do>
```

```
    <do work_dir="ior">make</do>
```

```
  </step>
```

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

```
    <use>ior_parameter</use>
```

```
    <do>compile/ior/src/ior -b $blocksize -t $transfersize</do>
```

```
  </step>
```

```
</benchmark>
```

```
</jube>
```

Folder/files
needed for
compilation

Yes, still
XML ☹

Dependent
step reference

Link to access
depend step

```
> jube run ior.xml
```

```
#####
```

```
# benchmark: ior
```

```
# id: 2
```

```
#
```

```
#
```

```
#####
```

```
Running workpackages (#=done, 0=wait, E=error):
```

```
##### ( 4/ 4)
```

stepname	all	open	wait	error	done
compile	1	0	0	0	1
execute	3	0	0	0	3

Two steps

```
>>>> Benchmark information and further useful commands:
```

```
>>>>      id: 2
```

```
>>>>    handle: runs
```

```
>>>>      dir: runs/000002
```

```
>>>> analyse: jube analyse runs --id 2
```

```
>>>>   result: jube result runs --id 2
```

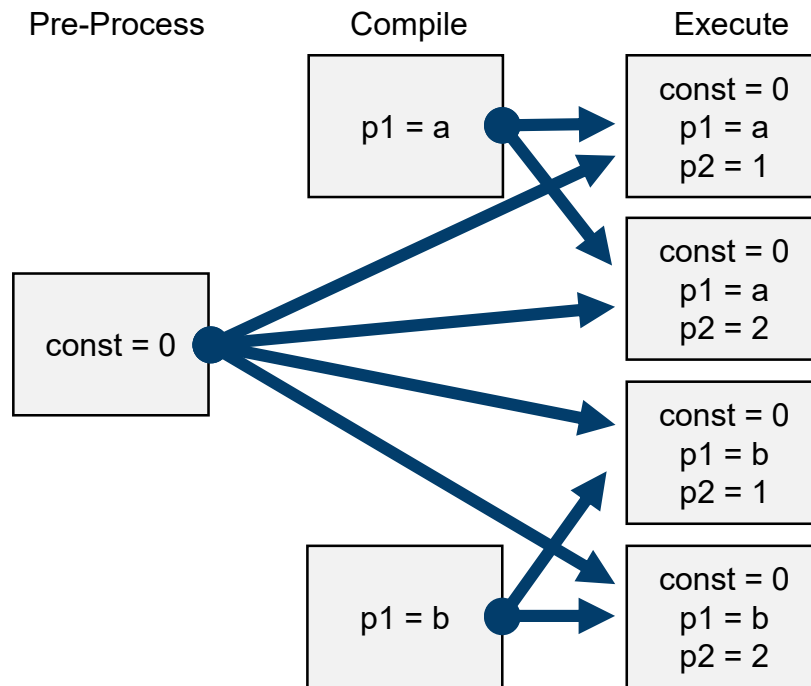
```
>>>>    info: jube info runs --id 2
```

```
>>>>     log: jube log runs --id 2
```

```
#####
```

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,compile">
  <use>execset</use>
</step>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<jube>
  <benchmark name="ior" outpath="./runs">
    <parameterset name="ior_parameter">
      <parameter name="blocksize">100m</parameter>
      <parameter name="transfersize">
        1m,10m,100m
      </parameter>
    </parameterset>
    <fileset name="source">
      <copy>ior</copy>
    </fileset>
    <step name="compile">
      <use>source</use>
      <do work_dir="ior">
        ./bootstrap & & ./configure
      </do>
      <do work_dir="ior">make</do>
    </step>
    <step name="execute" depend="compile">
      <use>ior_parameter</use>
      <do>compile/ior/src/ior -b $blocksize -t $transfersize</do>
    </step>
  </benchmark>
</jube>
```

```
<patternset name="ior_pattern">
  <pattern name="write_bandwidth">
    write\s+$jube_pat_fp
  </pattern>
</patternset>
```

Regex magic

```
<analyser name="ior_analyser">
  <use>ior_pattern</use>
  <analyse step="execute">
    <file>stdout</file>
  </analyse>
</analyser>
```

File to be scanned

```
<result>
  <use>ior_analyser</use>
  <table name="result_table" style="pretty">
    <column>transfersize</column>
    <column>write_bandwidth</column>
  </table>
</result>
```

Patterns and parameter can be mixed in the result

```
</benchmark>
</jube>
```

```
> jube run ior.xml -r
```

Run implicit result generation

```
#####
# benchmark: ior
# id: 3
#
#
#####
```

Running workpackages (#=done, 0=wait, E=error):

```
##### ( 4/ 4)
```

stepname	all	open	wait	error	done
compile	1	0	0	0	1
execute	3	0	0	0	3

>>>> Benchmark information and further useful commands:

```
>>>> id: 3
>>>> handle: runs
>>>> dir: runs/000003
>>>> analyse: jube analyse runs --id 3
>>>> result: jube result runs --id 3
>>>> info: jube info runs --id 3
>>>> log: jube log runs --id 3
```

```
#####
```

>>> Start analyse

>>> Analyse finished

result_table:

transfersize	write_bandwidth
1m	1828.65
10m	2058.20
100m	2489.41

Finally some output

PATTERN EXTRACTION

- A set of regular expressions can be configured to extract relevant output data

```
read_pattern = read\s+(\d*?\.\d*)\s+
```

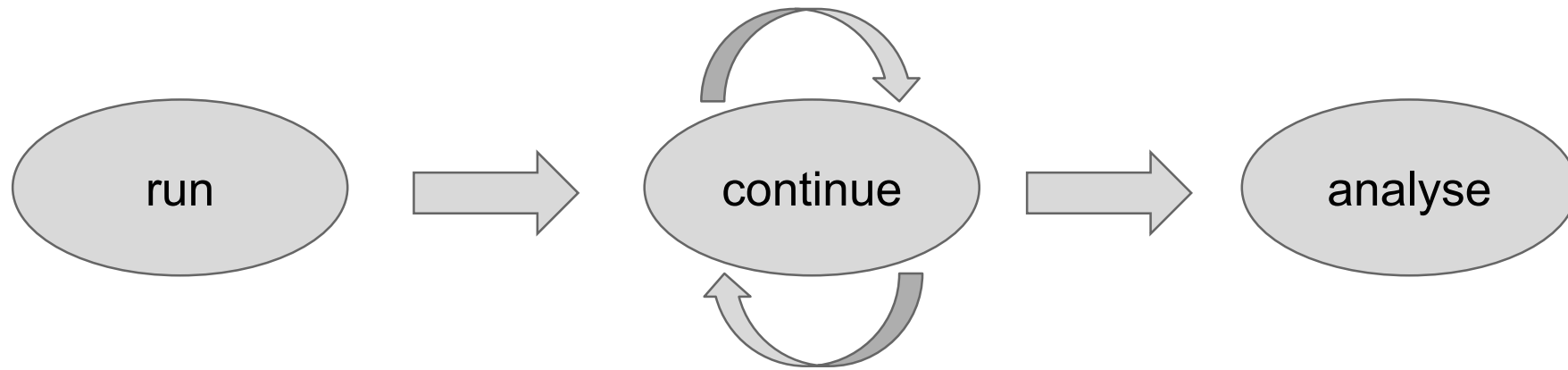
access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)
-----	-----	-----	-----	-----	-----	-----
Commencing read performance test: Wed Nov 28 16:52:48 2018						
read	89.52	1191936	2.27	0.004406	1248.26	0.001339
remove	-	-	-	-	-	-



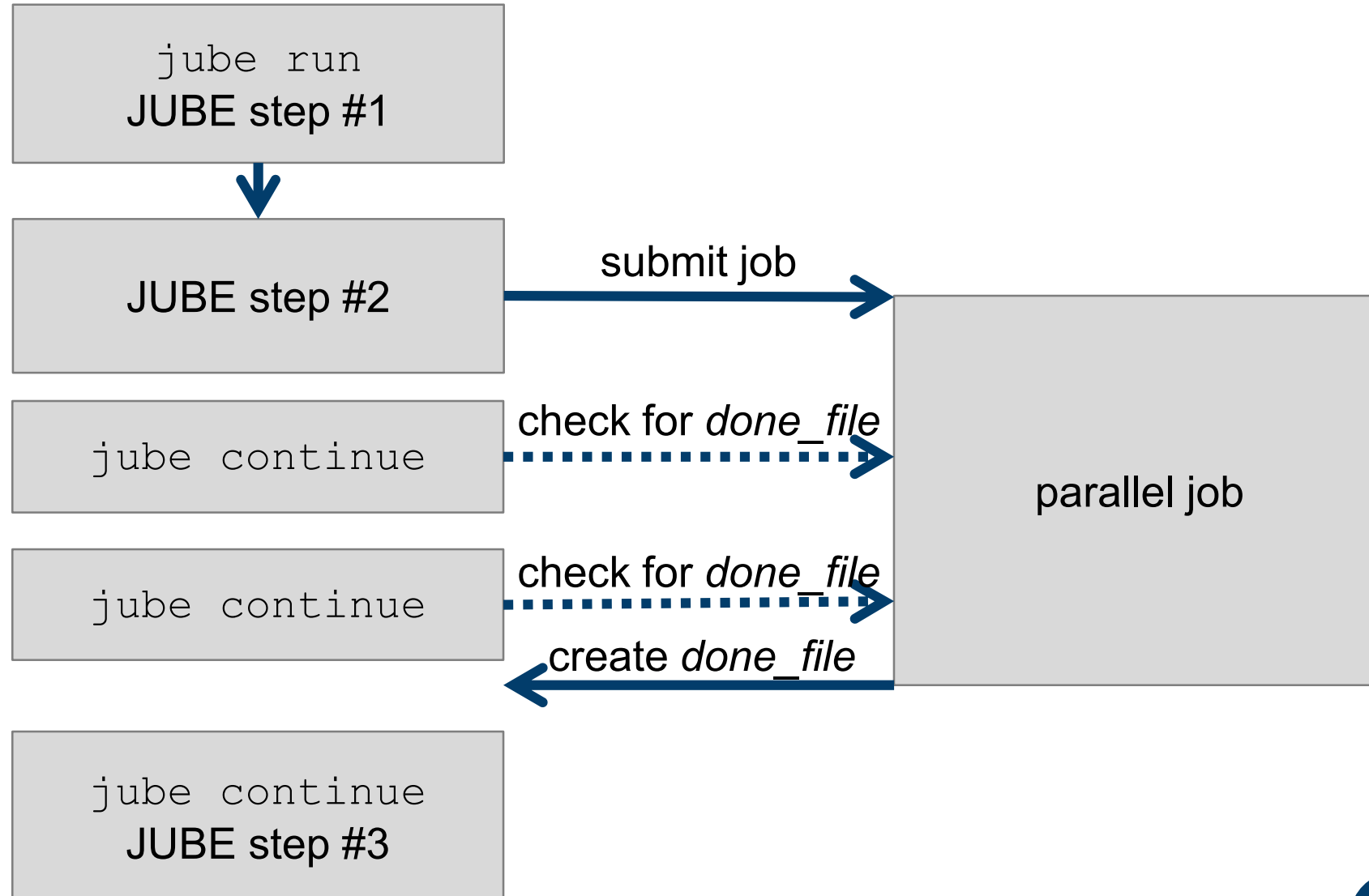
Parameter	read_pattern
value	89.52

JOB SUBMISSION

- The JUBE kernel has no direct knowledge about HPC scheduling systems
- A job template and the substitution system can be used to generalize the job submission process
- JUBE supports asynchronous command execution
- A marker file is used to mark the end of the job; it must be generated by the job script after the parallel part was executed
- jube continue triggers JUBE to check all available marker files



JOB SUBMISSION



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>]
```



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>]
```

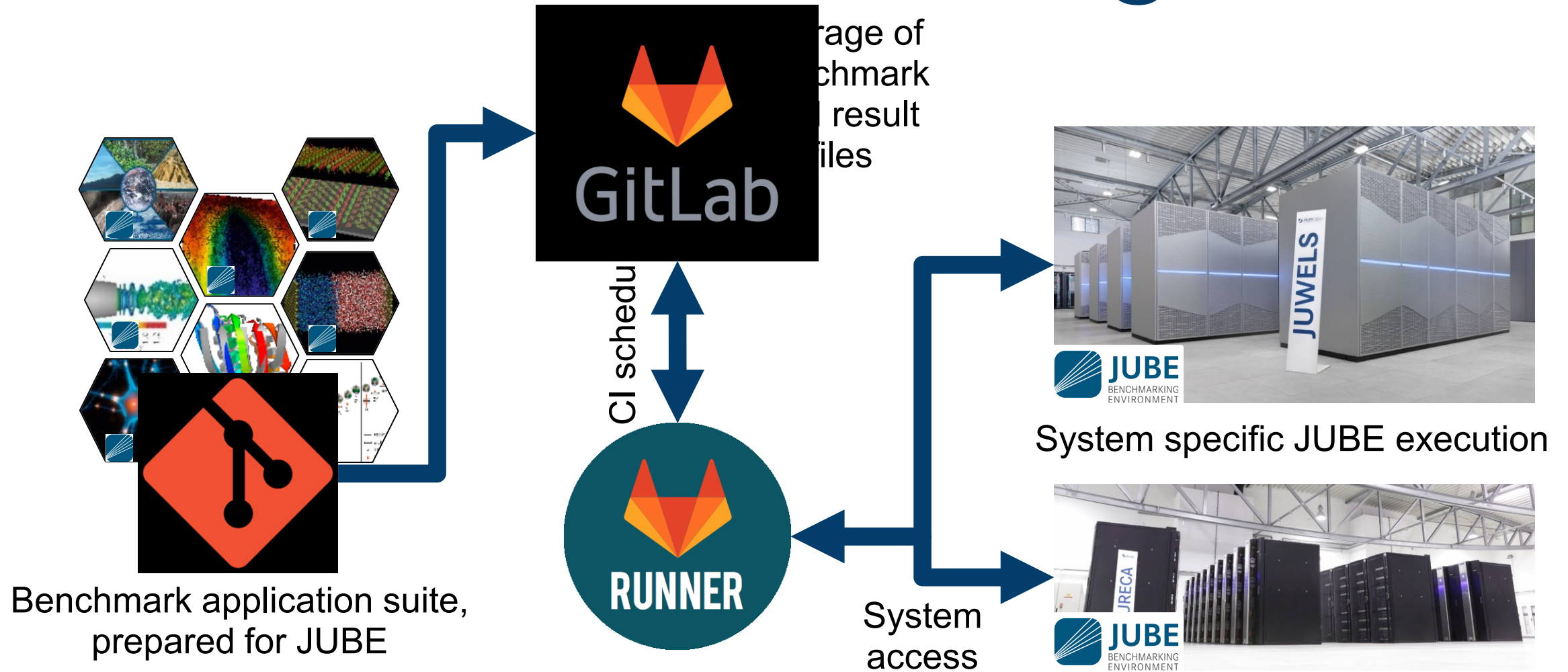
- Debug mode

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

- Verbose mode

```
jube -v[vv] run ...
```

AUTOMATED BENCHMARKING PLANNED @JSC



JUBE AVAILABILITY

- Download, Tutorials and Documentation:
www.fz-juelich.de/jsc/jube
- Prerequisites:
 - OS: Linux
 - Python 3 (Python 2 is still also fine)
- Contact:
jube.jsc@fz-juelich.de



ONE MORE THING

```
name: ior
outpath: ./runs
parameterset:
  - name: ior_parameter
    parameter: [{name: blocksize, _: 100m}, {name: transfersize, _: "1m,10m,100m"}]
  - name: systemParameter
    init_with: platform.xml
    parameter: [{name: nodes, type: int, _: "1,2,4"}, {name: taskspnode, type: int, _: 8},
                {name: executable, _: compile/ior/src/ior}, {name: "args_exec", _: -b $blocksize -t $transfersize}]
fileset:
  name: source
  copy: ior
step:
  - name: compile
    do: {work_dir: ior, _: [./bootstrap && ./configure,make]}
  - name: execute
    use: [ior_parameter,systemParameter,{from: platform.xml, _: [executeset,executesub,jobfiles]}]
    do: {done_file: $done_file, _: $submit $submit_script}
patternset:
  name: ior_pattern
  pattern: {name: write_bandwidth, _: write\s+$jube_pat_fp}
analyser:
  name: ior_analyser
  use: ior_pattern
  analyse:
    step: execute
    file: job.out
result:
  use: ior_analyser
  table:
    name: result_table
    style: pretty
    column: [nodes,transfersize,{format: .1f, _: write_bandwidth}]
```

Not a friend of XML?
Use **YAML!**
Supported in upcoming
JUBE release 2.4.0