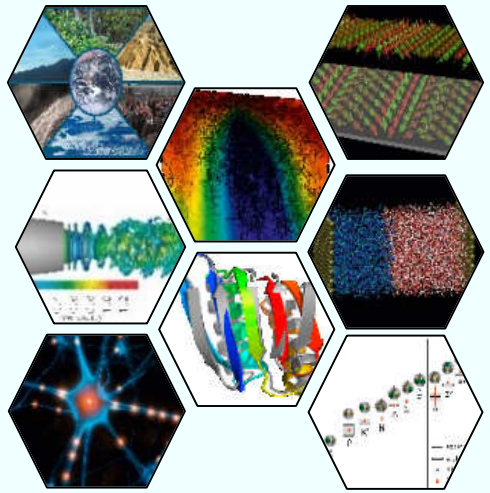
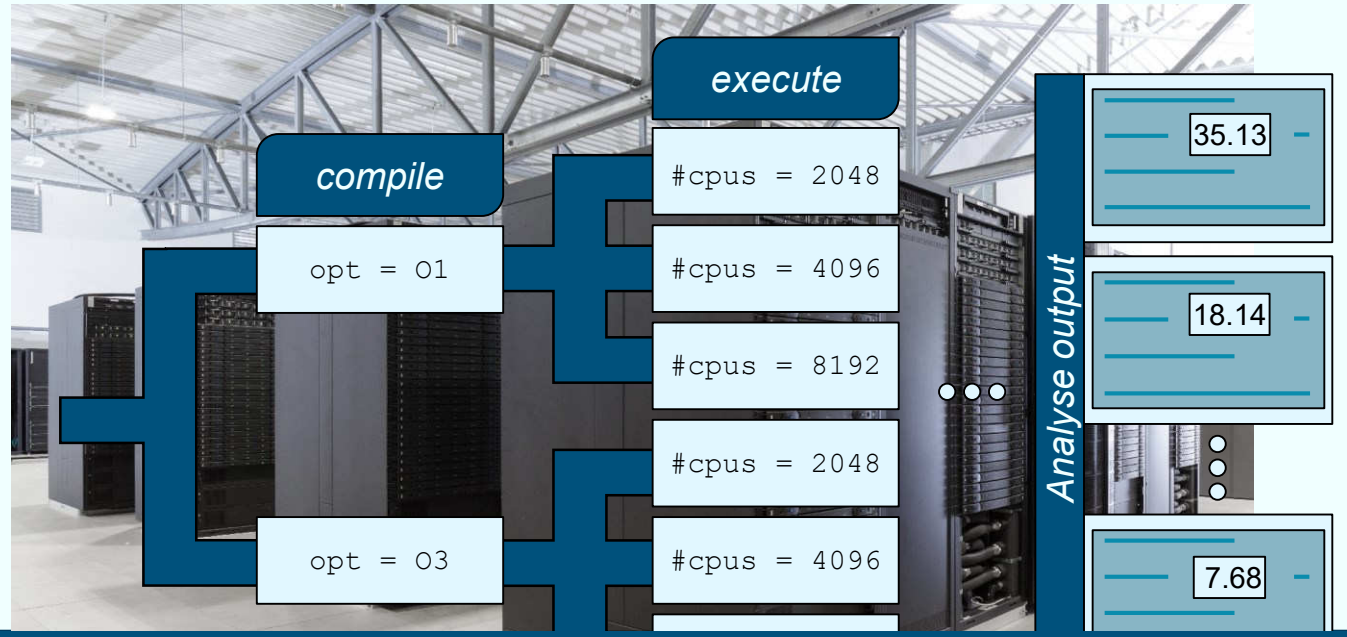




opt	#cpus	time [s]
O1	2048	35.13
O1	4096	18.14
O1	8192	9.32
O3	2048	29.00
O3	4096	15.00
O3	8192	7.68

AUTOMATED BENCHMARKING WITH JUBE

JUBE BENCHMARKING ENVIRONMENT

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

JÜLICH SUPERCOMPUTING CENTRE
FORSCHUNGSZENTRUM JÜLICH GMBH

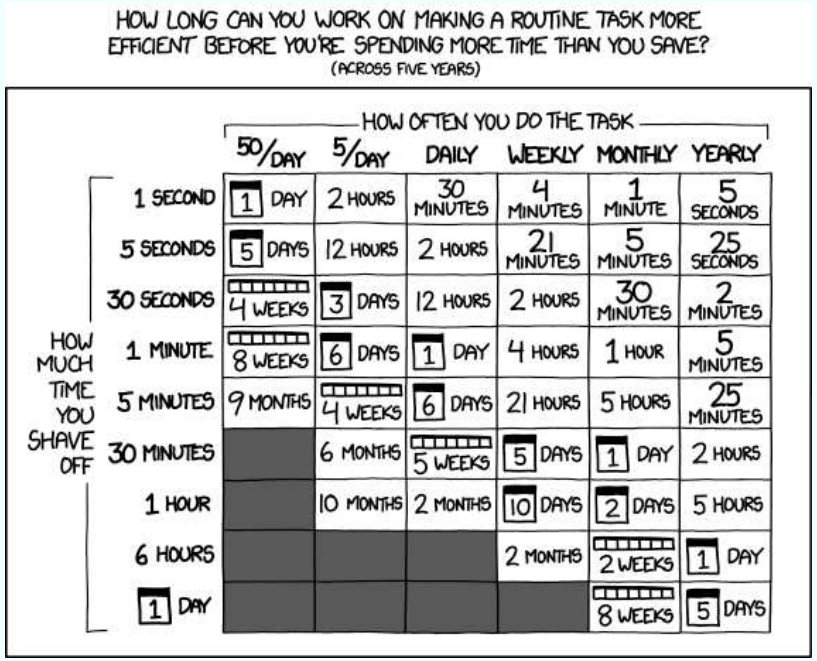
WHAT IS JUBE AND WHY CAN IT HELP?

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

- App. specific  script replacement.

Can be used for:

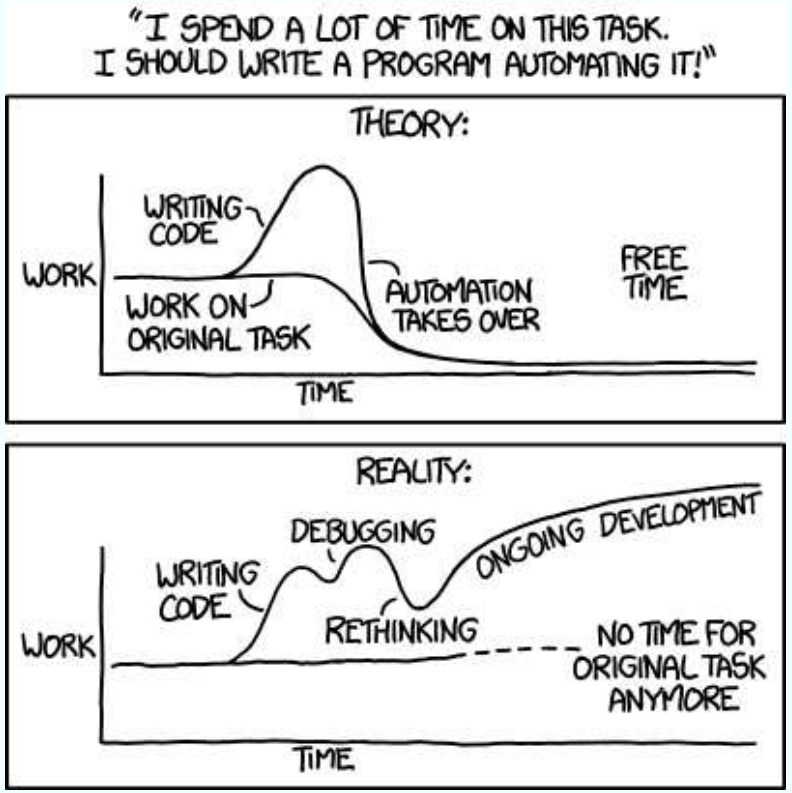
- Benchmarking
- Parameter studies
- Testing
- Profiling
- Production scenarios
- …



https://xkcd.com/1205/

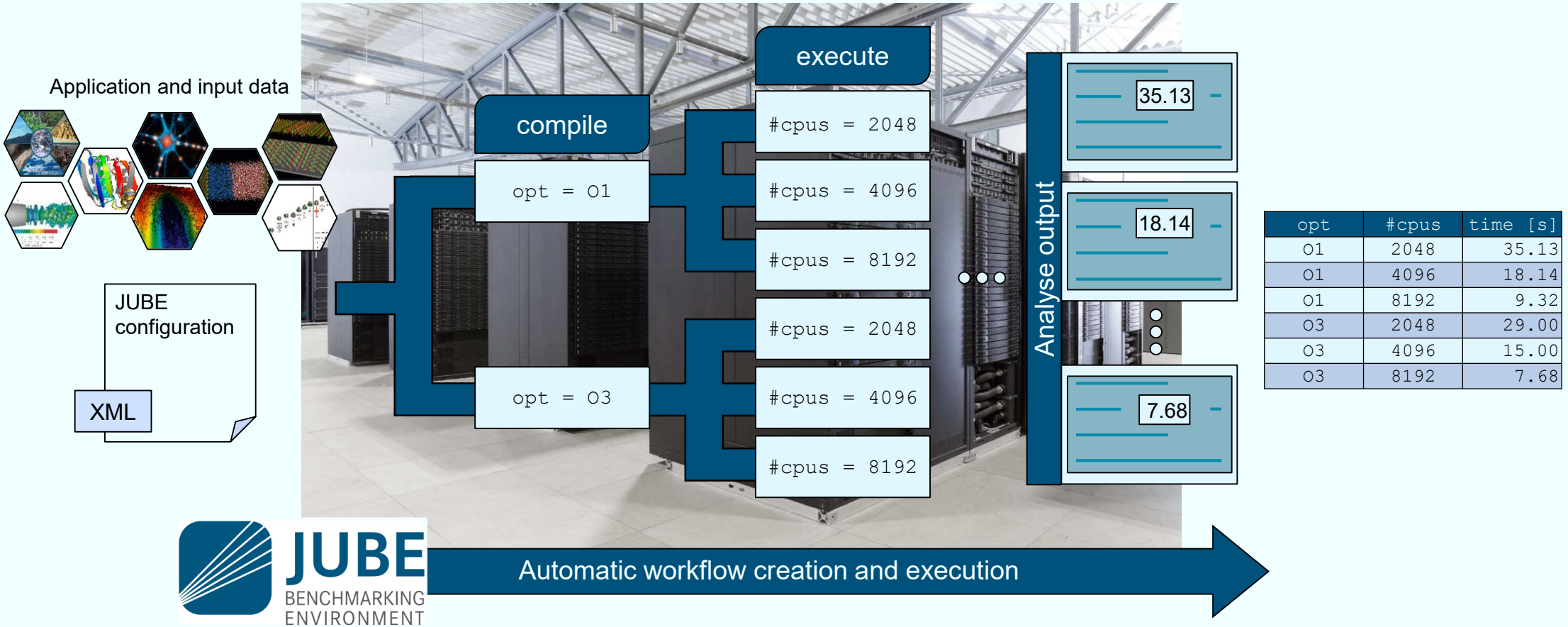
WHAT JUBE IS **NOT**?

- 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

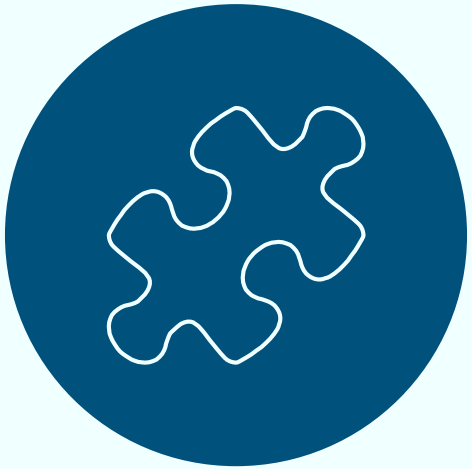


https://xkcd.com/1319/

WHAT IS JUBE?



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



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

Yes it is XML ☹

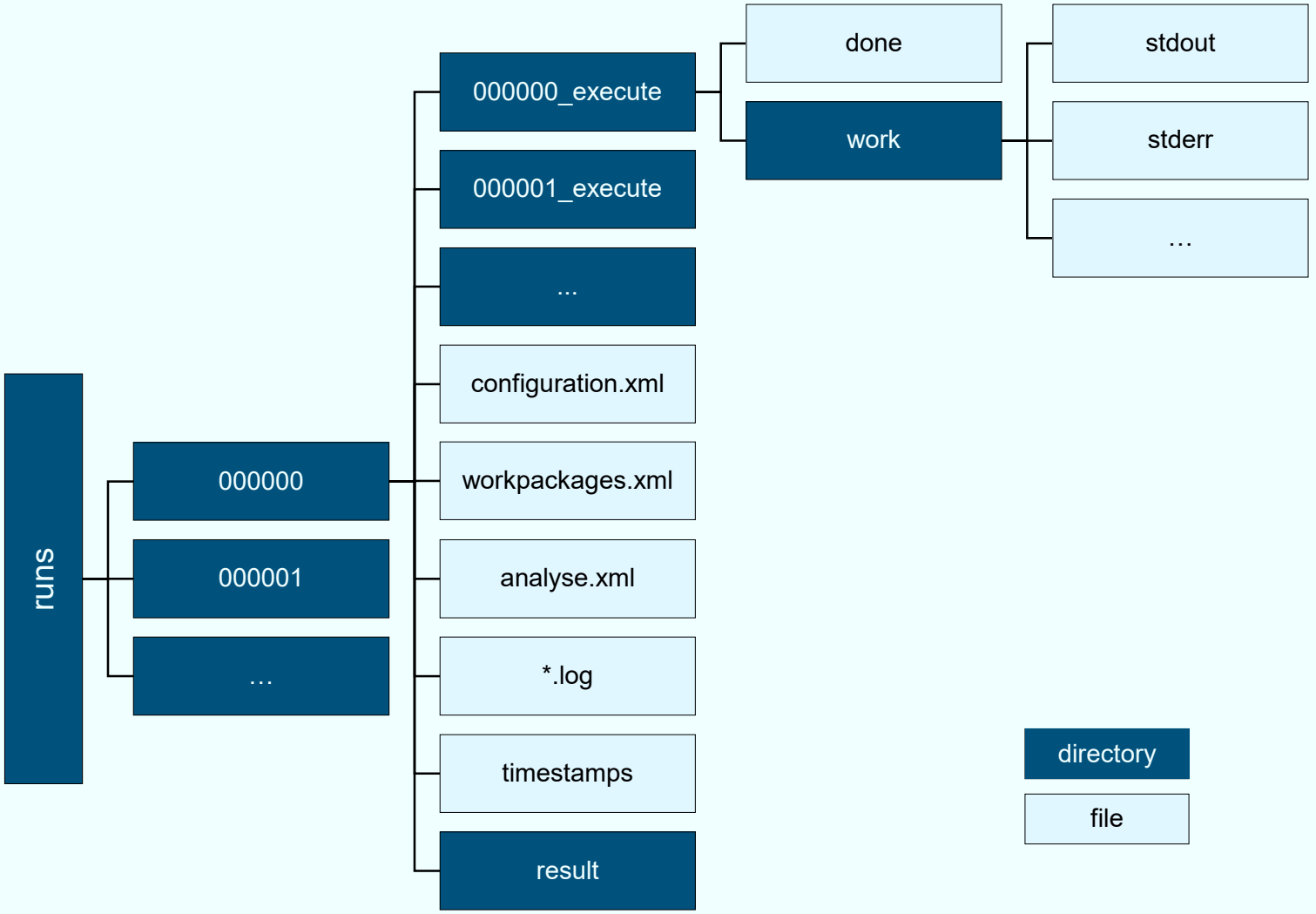
Assuming `ior` is sitting in `$PATH`

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
<?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

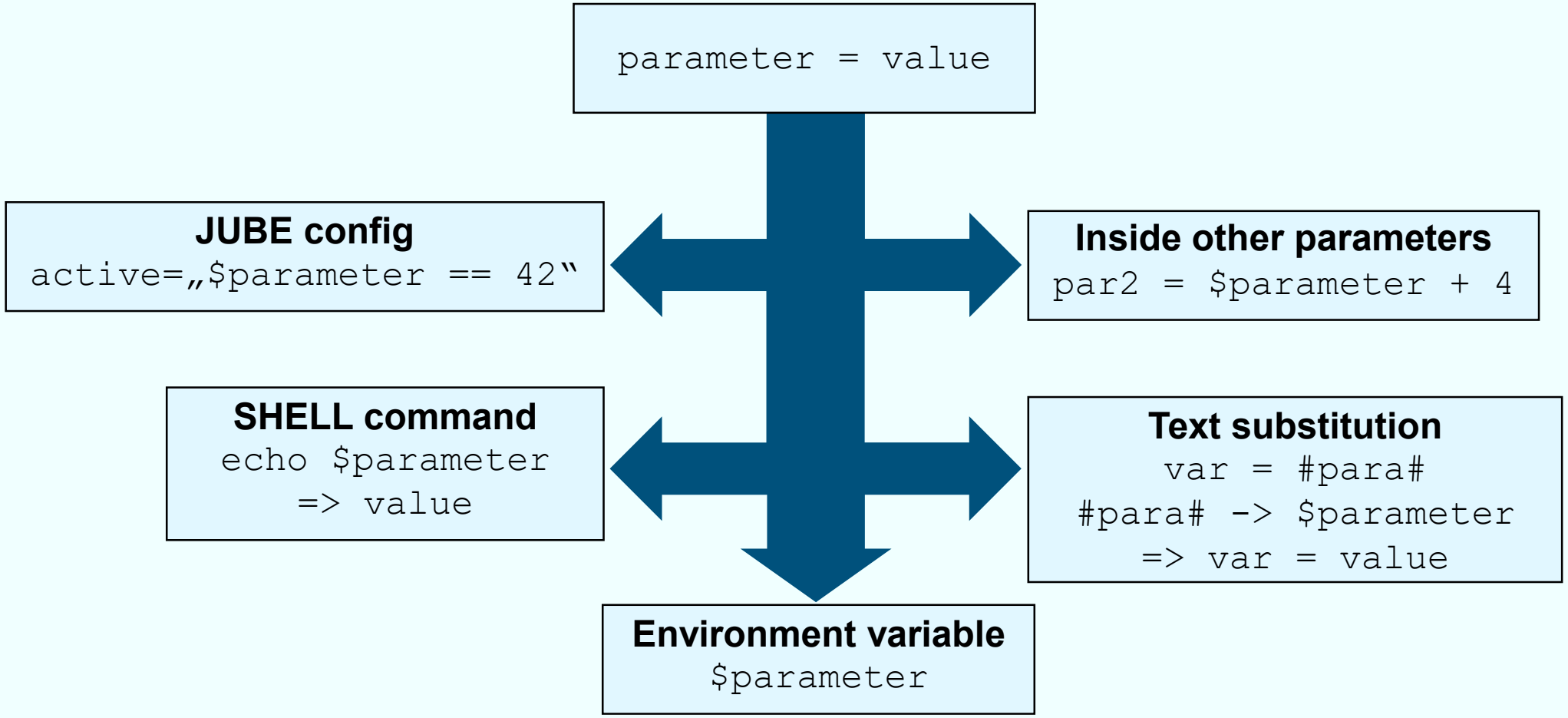
Not an environment variable

```
> 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

>>>> 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
################################################################
```

Three individual runs

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 &amp;&amp; ./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

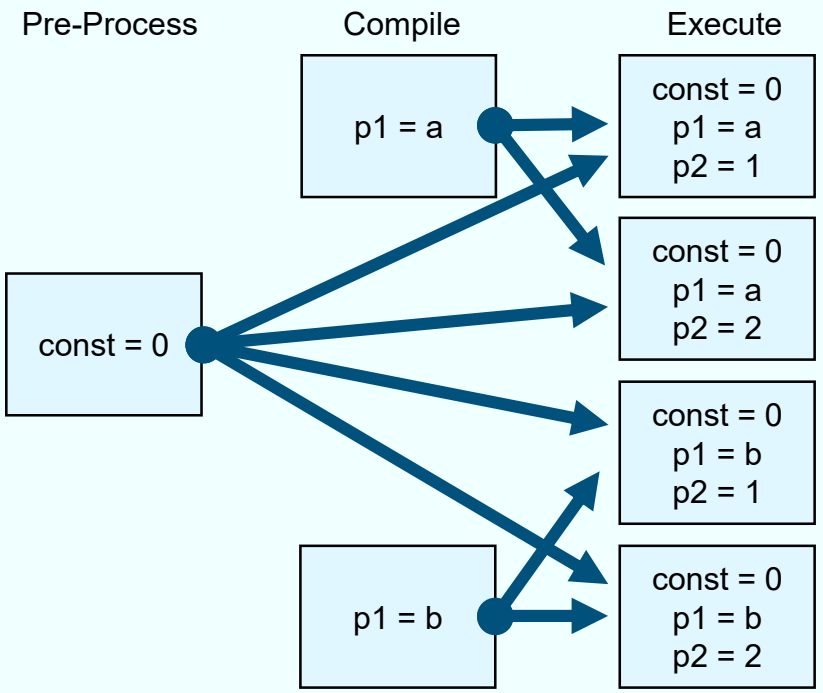
Link to access depend step

```
> 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

>>>> 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
################################################################################
```

Two steps

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>
```

```
<?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 &amp;&amp; ./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>

    <patternset name="ior_pattern">
      <pattern name="write_bandwidth">
        write\s+$jube_pat_fp
      </pattern>
    </patternset>
    <analyser name="ior_analyser">
      <use>ior_pattern</use>
      <analyse step="execute">
        <file>stdout</file>
      </analyse>
    </analyser>
    <result>
      <use>ior_analyser</use>
      <table name="result_table" style="pretty">
        <column>transfersize</column>
        <column>write_bandwidth</column>
      </table>
    </result>

  </benchmark>
</jube>
```

Regex magic

File to be scanned

Patterns and parameter can be mixed in the result

```
> 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
```

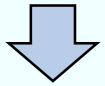
Finally some output

PATTERN EXTRACTION

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

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

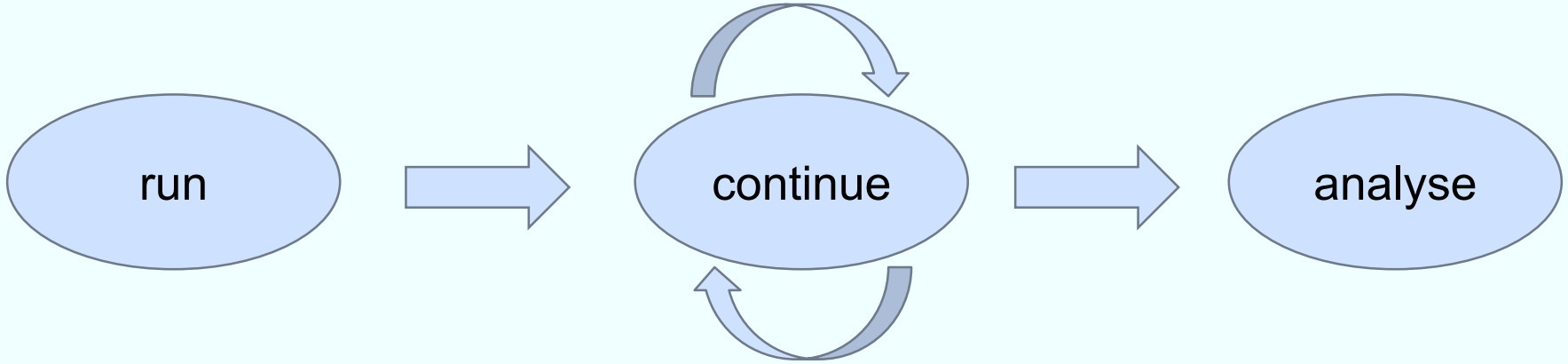
```
access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s)
------    ---------  ---------- ---------  --------   --------   --------
Commencing read performance test: Wed Nov 28 16:52:48 2018
read      89.52      1191936    2.27       0.004406   1248.26    0.001339
remove    -    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



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



```xml
<?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 &amp;&amp; ./configure
      </do>
      <do work_dir="ior">make</do>
    </step>

    <parameterset name="systemParameter" init_with="platform.xml">
      <parameter name="nodes" type="int">1,2,4</parameter>
      <parameter name="taskspernode" type="int">8</parameter>
      <parameter name="executable">compile/ior/src/ior</parameter>
      <parameter name="args_exec">-b $blocksize -t $transfersize</parameter>
    </parameterset>

    <step name="execute" depend="compile">
      <use>ior_parameter</use>
      <use>systemParameter</use>
      <use from="platform.xml">executeset,executesub,jobfiles</use>
      <do done_file="$done_file">$submit $submit_script</do>
    </step>

    <patternset name="ior_pattern">
      <pattern name="write_bandwidth">
        write\s+$jube_pat_fp
      </pattern>
    </patternset>
    <analyser name="ior_analyser">
      <use>ior_pattern</use>
      <analyse step="execute">
        <file>job.out</file>
      </analyse>
    </analyser>
    <result>
      <use>ior_analyser</use>
      <table name="result_table" style="pretty">
        <column>nodes</column>
        <column>transfersize</column>
        <column format=".1f">write_bandwidth</column>
      </table>
    </result>
  </benchmark>
</jube>
```

Use default platform configuration

Overwrite defaults where necessary

Use predefined job templates and parameter substitutions

Wait until done_file is created

Scan job output instead of stdout

```
> jube run ior.xml

################################################################################
# benchmark: ior
# id: 4
#
#
################################################################################

Running workpackages (#=done, 0=wait, E=error):
######0000000000000000000000000000000000000000000000000000 (  1/ 10)

  stepname | all | open | wait | error | done
  ---------+-----+------+------+-------+-----
   compile |   1 |    0 |    0 |     0 |    1
   execute |   9 |    0 |    9 |     0 |    0

>>>> Benchmark information and further useful commands:
>>>>       id: 4
>>>>   handle: runs
>>>>      dir: runs/000004
>>>> continue: jube continue runs --id 4
>>>>  analyse: jube analyse runs --id 4
>>>>   result: jube result runs --id 4
>>>>     info: jube info runs --id 4
>>>>      log: jube log runs --id 4
################################################################################>
```

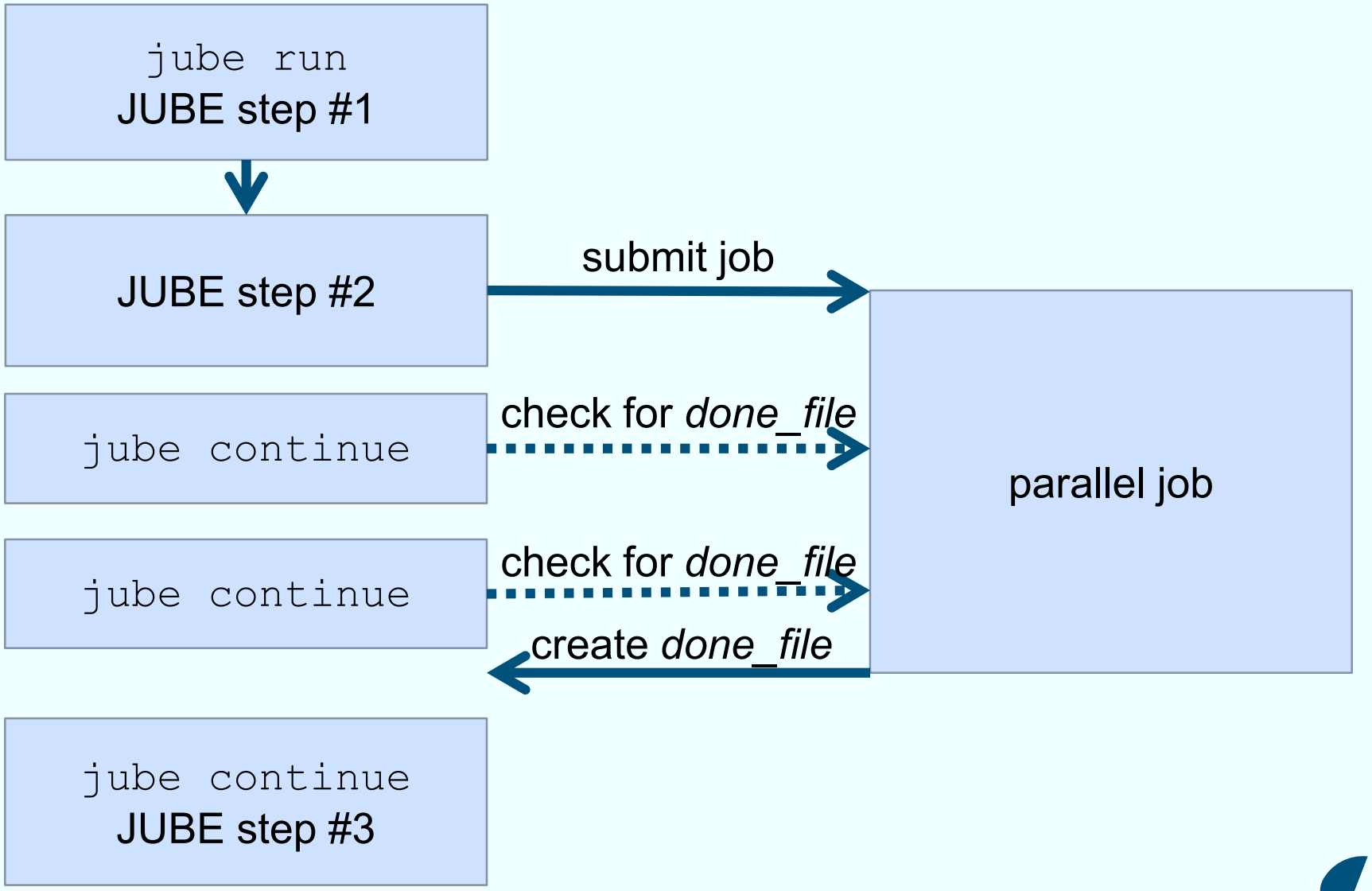
Jobs were queued

Nine jobs in total

```
> jube continue runs -r
…
result_table:
nodes | transfersize | write_bandwidth
------+--------------+----------------
    1 |           1m |          5780.0
    1 |          10m |          3193.3
    1 |         100m |          3064.4
    2 |           1m |          5866.0
    2 |          10m |          6462.0
    2 |         100m |          6227.0
    4 |           1m |          6946.0
    4 |          10m |         10535.0
    4 |         100m |          8609.0
```

Check job state and create result

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>]
```

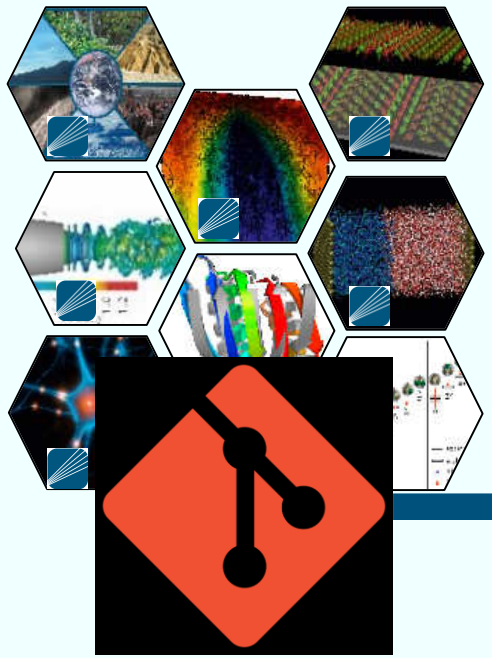
- Debug mode

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

- Verbose mode

```
jube -v[vv] run …
```

AUTOMATED BENCHMARKING PLANNED @JSC



Benchmark application suite, prepared for JUBE

rage of
chmark
result
iles

GitLab

CI schedu

RUNNER

System specific JUBE execution

System access

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: taskspernode, 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