CALIPER

Shyju PV [shyju.pv@gmail.com]

CONTENTS

- **Introduction to Caliper**
- **Scope of the caliper**
- **Caliper Organization**
- **Caliper work flow**
- **Caliper Execution for Single Target**
- **Caliper Execution flow for Multiple Targets**
- **Caliper report generation**
- **Caliper setup**

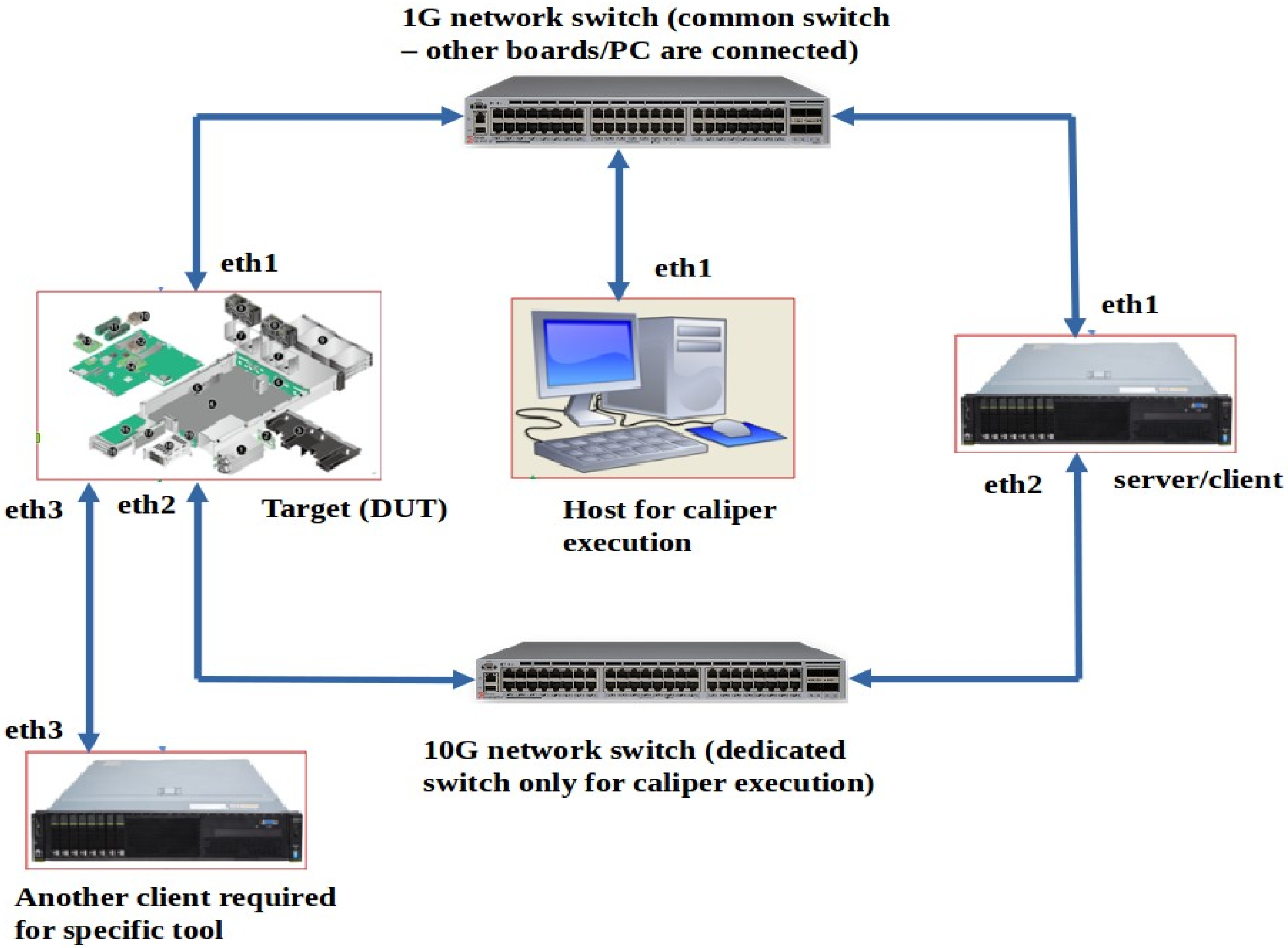
Introduction to Caliper

- **Caliper** is a benchmarking framework for **server platforms**, integrated with industry standard tools and test cases.

- It is a test suite focused on benchmarking and performance evaluation of boards, It not only detects if the hardware and software of the board work well, but also tests the basic functionality.

- Caliper executes on a host system which is connected to the boards through network.

Caliper Setup

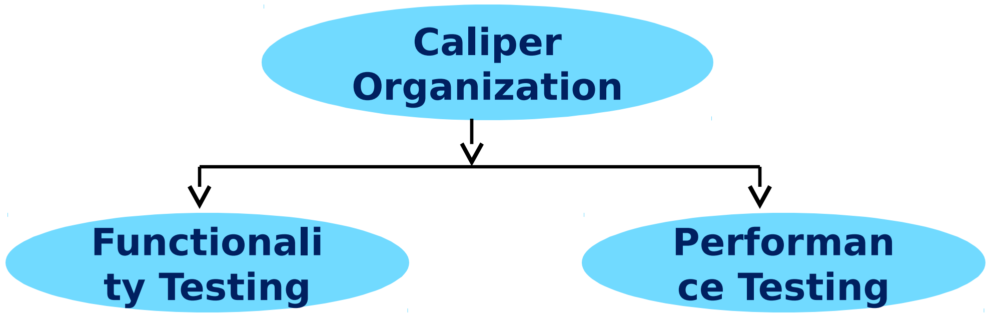


1G network switch (common switch
– other boards/PC are connected)
eth1
eth1
eth1
Target (DUT)
Host for caliper
execution
eth3
eth2
eth2
server/client
eth3
Another client required
for specific tool
10G network switch (dedicated
switch only for caliper execution)

Scope Of Caliper

- Caliper run many use case scenario , test and benchmarking tools and generate a series of test data.
- This can be analysed by test/dev people which may provide inputs for specific hardware tuning or driver optimization.
- Our own test cases/tools can be added to Caliper framework
- It supports aarch_64 & x86_64 architecture

Caliper Organization

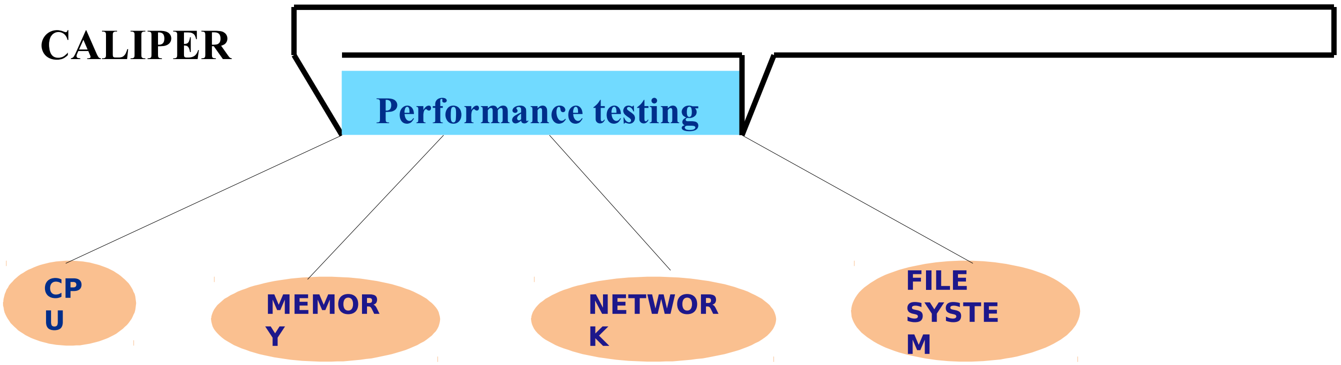


Caliper Organization

Functionali
ty Testing

Performan
ce Testing

- **Functional testing** – Caliper's functional testing mainly uses the LTP (Linux Test Project), which contains a lot of tests focused on systems' functions and features testing.
- **Performance testing** - Caliper mainly includes well known open source tools and benchmarks to evaluate the performance.
- Caliper's performance assessment includes :



CALIPER

Performance testing

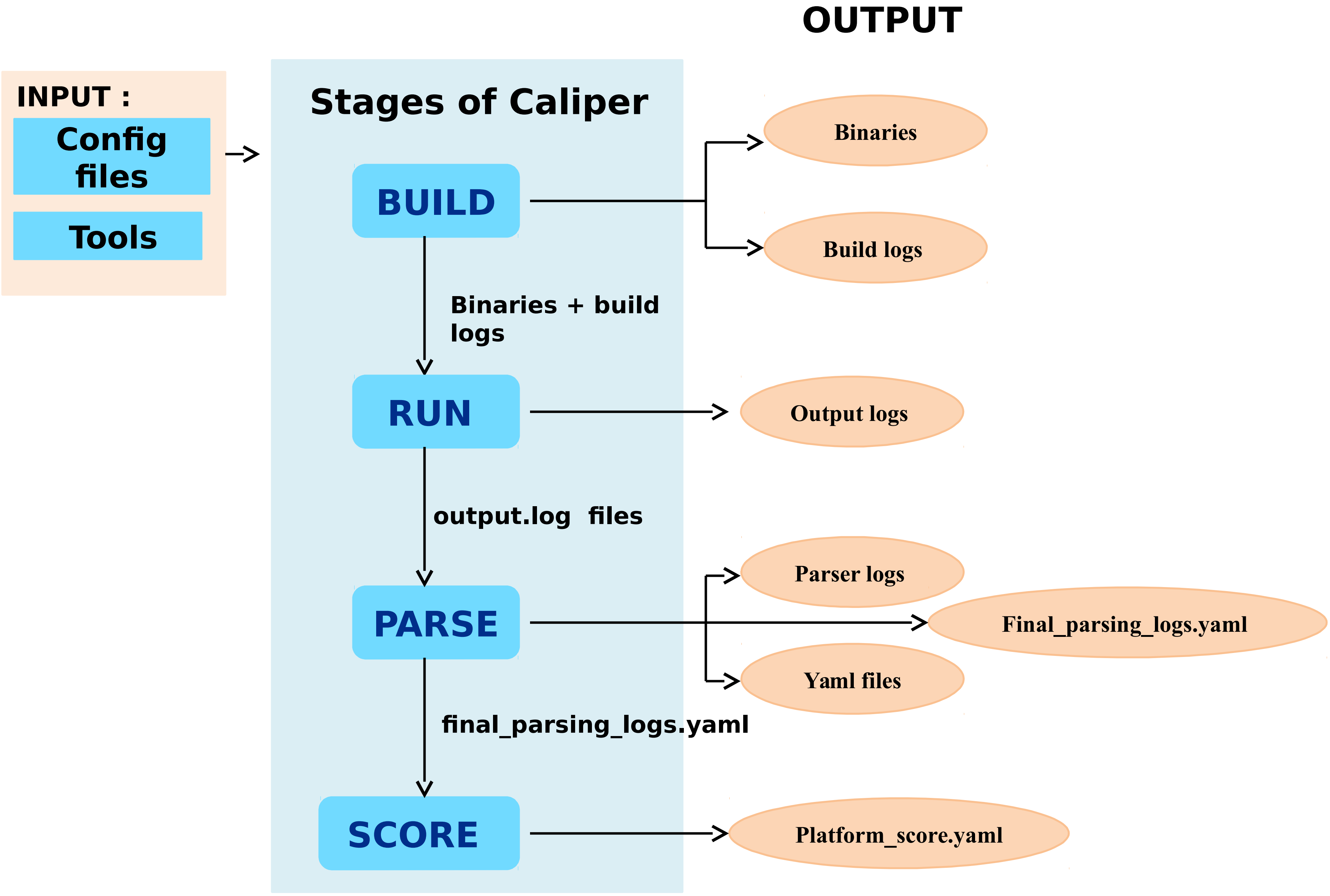
CP
U

MEMOR
Y

NETWOR
K

FILE
SYSTE
M

Caliper work Flow



OUTPUT

INPUT :

Config files

Tools

Stages of Caliper

BUILD

Binaries + build logs

RUN

output.log files

PARSE

final_parsing_logs.yaml

SCORE

Binaries

Build logs

Output logs

Parser logs

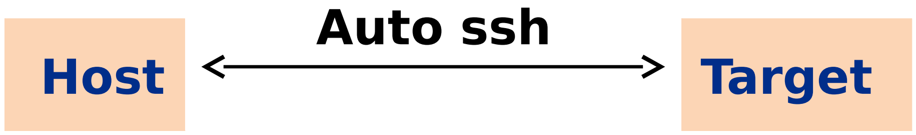
Final_parsing_logs.yaml

Yaml files

Platform_score.yaml

Caliper Execution for Single Target

1.

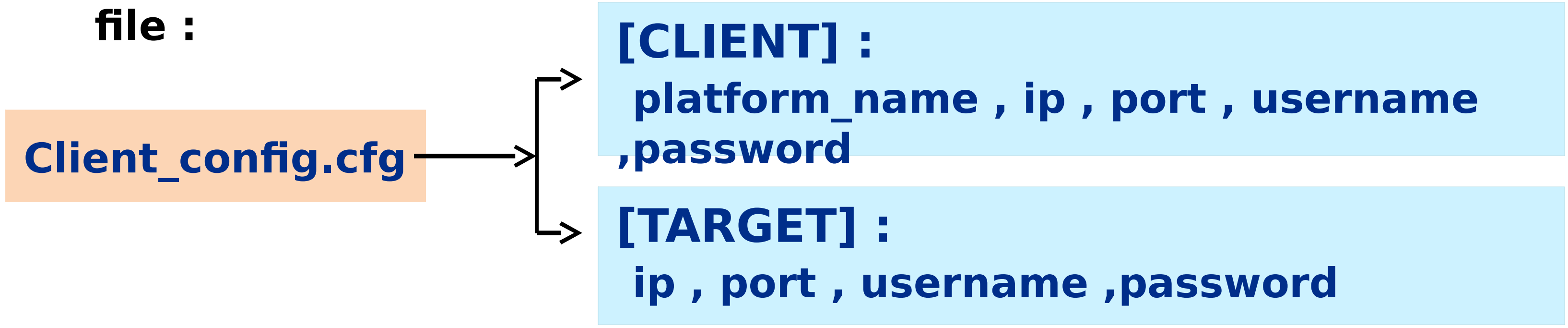


Host

Auto ssh

Target

2. To Configure the environment fill the fields in the input file :



Client_config.cfg

[CLIENT] :

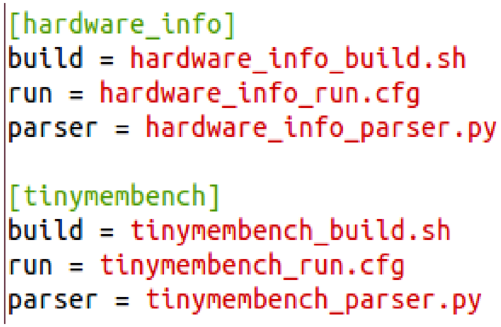
platform_name , ip , port , username ,password

[TARGET] :

ip , port , username ,password

3. Select the tools to be executed in below 3 files:

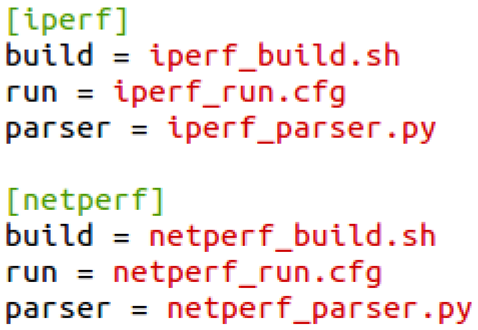
common_cases_def.cfg



```
[hardware_info]
build = hardware_info_build.sh
run = hardware_info_run.cfg
parser = hardware_info_parser.py

[tinymembench]
build = tinymembench_build.sh
run = tinymembench_run.cfg
parser = tinymembench_parser.py
```

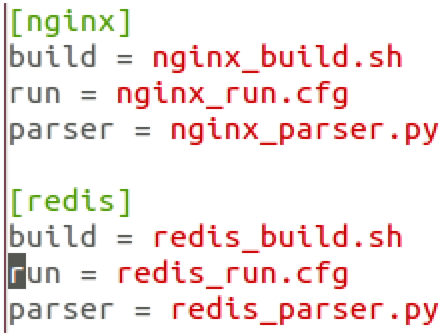
server_cases_def.cfg



```
[iperf]
build = iperf_build.sh
run = iperf_run.cfg
parser = iperf_parser.py

[netperf]
build = netperf_build.sh
run = netperf_run.cfg
parser = netperf_parser.py
```

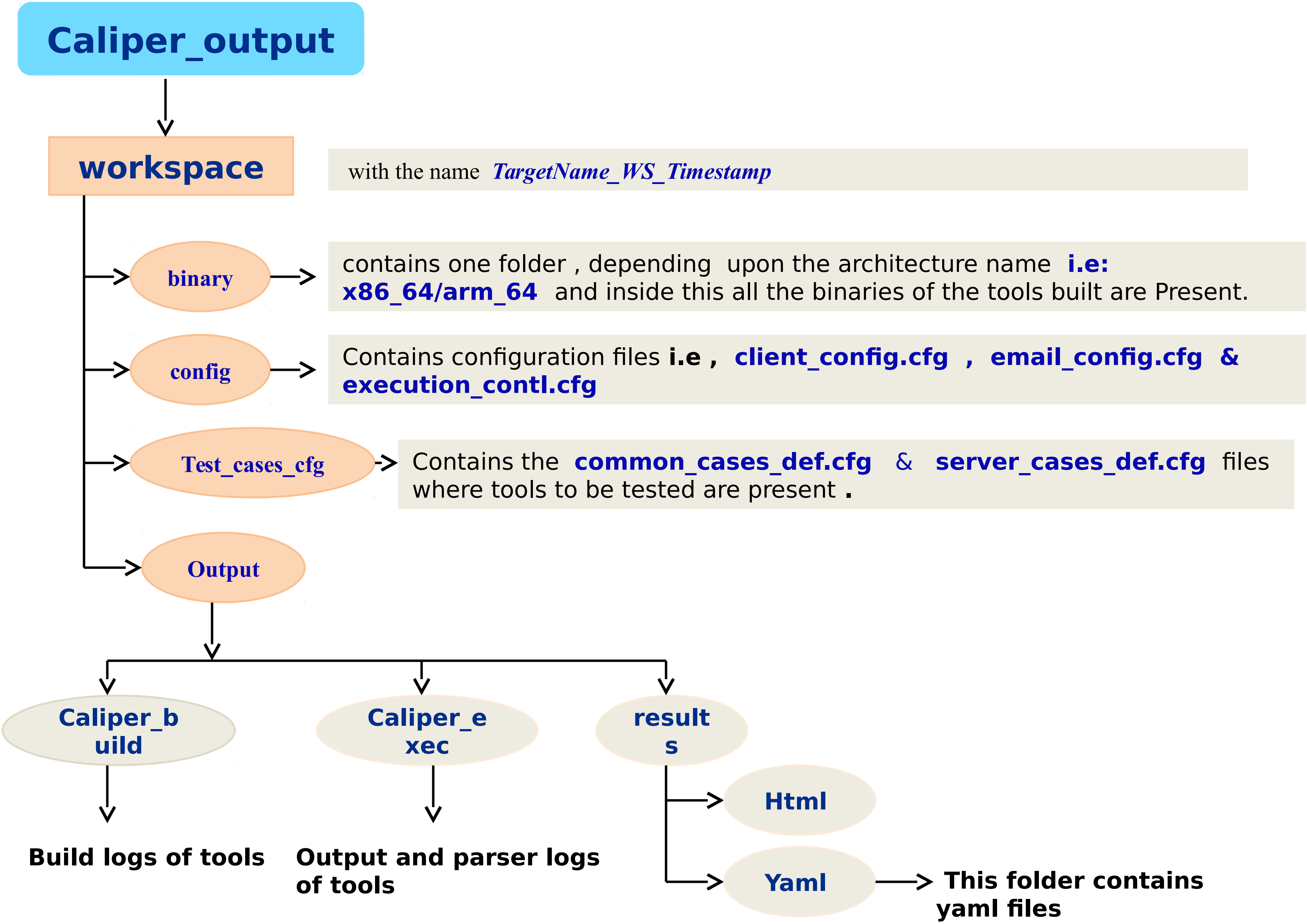
application_cases_def.cfg



```
[nginx]
build = nginx_build.sh
run = nginx_run.cfg
parser = nginx_parser.py

[redis]
build = redis_build.sh
run = redis_run.cfg
parser = redis_parser.py
```

4. Run the command **Caliper -brps**



Caliper_output

workspace

with the name *TargetName_WS_Timestamp*

binary

contains one folder , depending upon the architecture name **i.e: x86_64/arm_64** and inside this all the binaries of the tools built are Present.

config

Contains configuration files **i.e ,** **client_config.cfg** **,** **email_config.cfg** **&** **execution_contl.cfg**

Test_cases_cfg

Contains the **common_cases_def.cfg** & **server_cases_def.cfg** files where tools to be tested are present **.**

Output

Caliper_build

Caliper_exec

results

Html

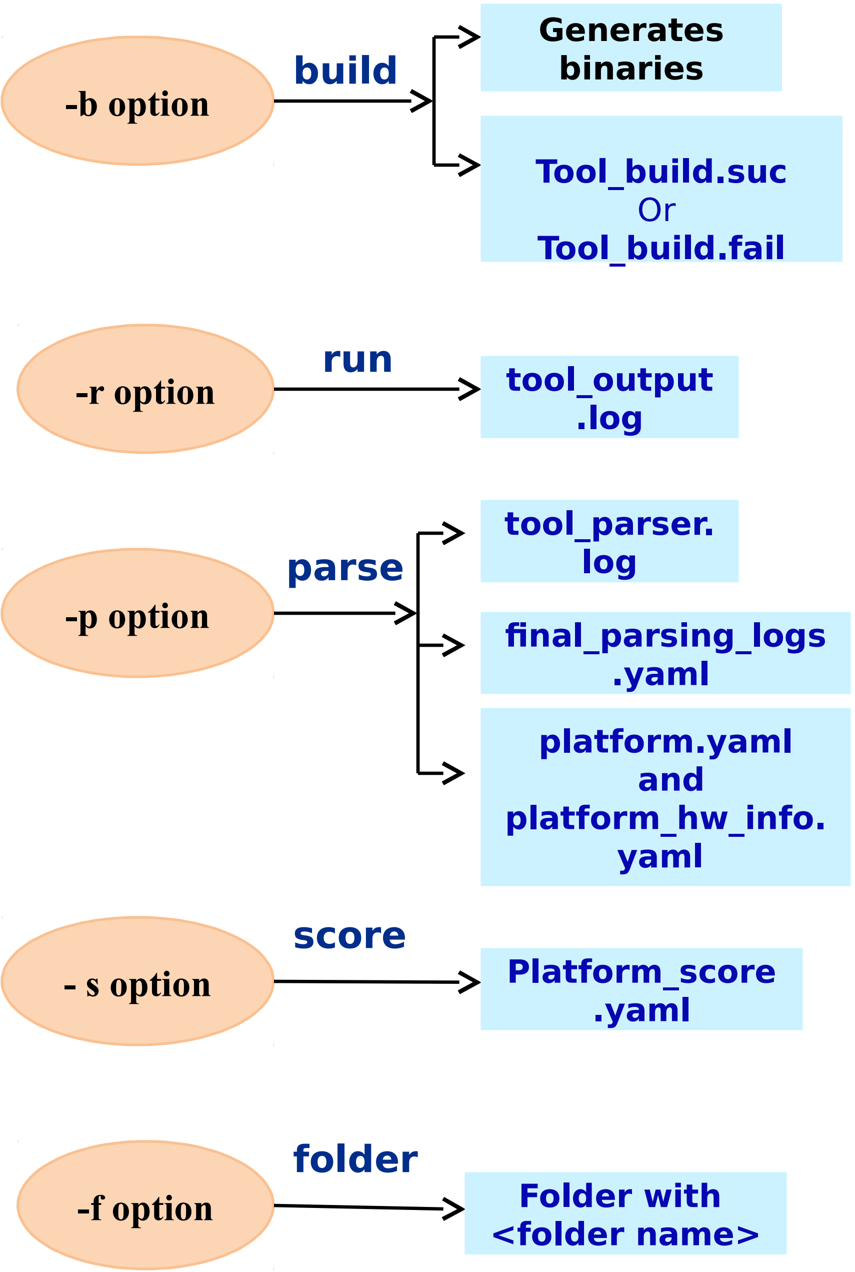
Yaml

Build logs of tools

Output and parser logs of tools

This folder contains yaml files

Output Structure



-b option → build → Generates binaries — of the tools built in **binary folder.**

-b option → build → **Tool_build.suc** Or **Tool_build.fail** — Files **caliper_build folder**

-r option → run → **tool_output .log** — file in **caliper_exec folder**

-p option → parse → **tool_parser. log** — file in **caliper_exec folder**

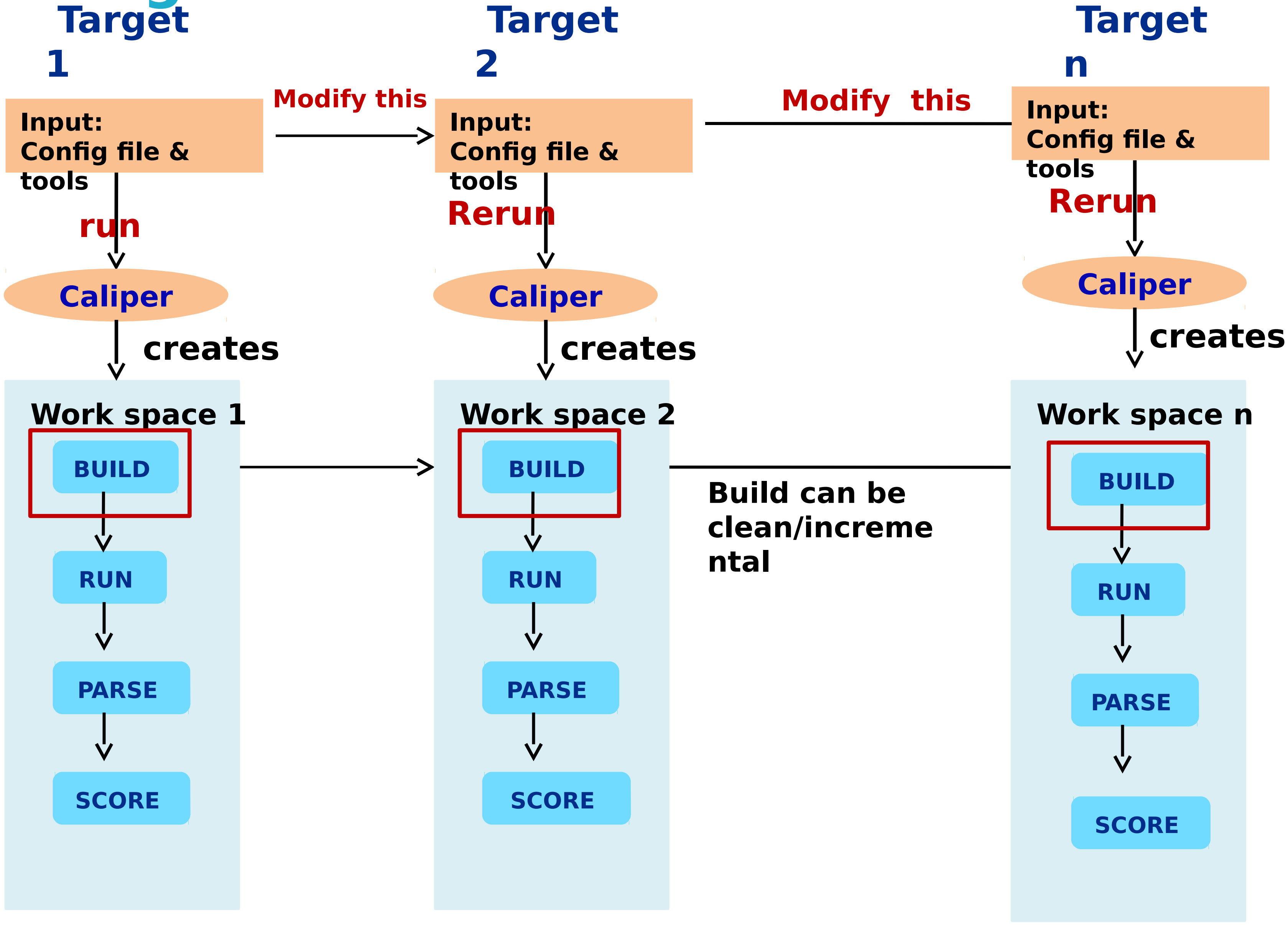
-p option → parse → **final_parsing_logs .yaml** — file in **output folder**

-p option → parse → **platform.yaml and platform_hw_info. yaml** — files in **yaml folder**

- s option → score → **Platform_score .yaml** — file in **yaml folder**

-f option → folder → **Folder with <folder name>** — Creates **workspace** with the **<folder name >** in **~/caliper_output/** directory

Caliper Execution flow for Multiple Targets



Target 1

Input: Config file & tools

run

Caliper

creates

Work space 1

BUILD

RUN

PARSE

SCORE

Modify this

Target 2

Input: Config file & tools

Rerun

Caliper

creates

Work space 2

BUILD

RUN

PARSE

SCORE

Modify this

Build can be clean/incremental

Target n

Input: Config file & tools

Rerun

Caliper

creates

Work space n

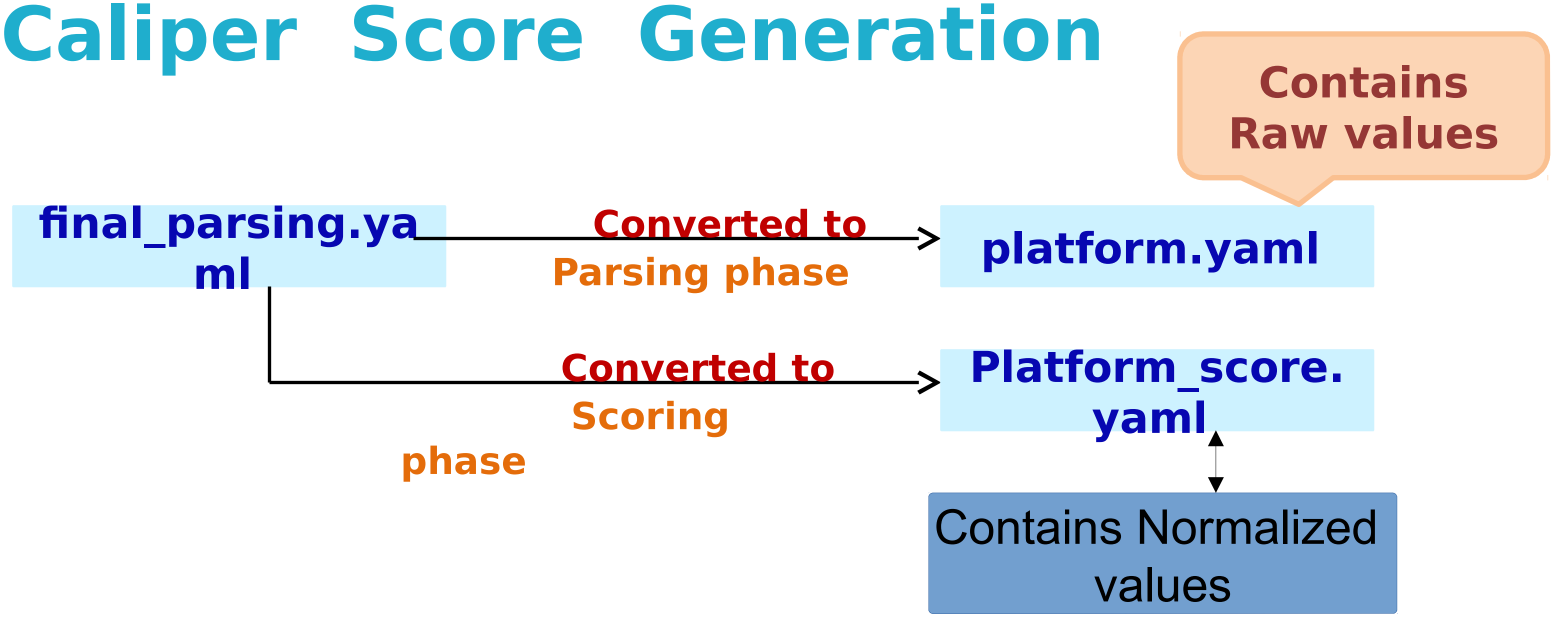
BUILD

RUN

PARSE

SCORE

Caliper Score Generation



Contains Raw values

final_parsing.yaml

Converted to

Parsing phase

platform.yaml

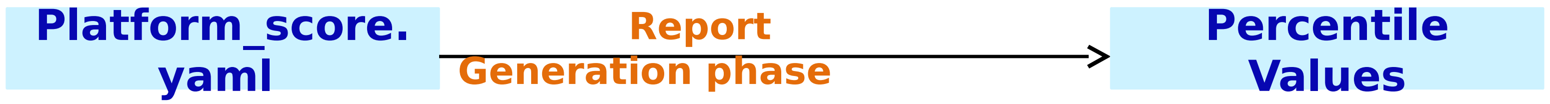
Converted to

Scoring phase

Platform_score.yaml

Contains Normalized values

During Report Generation Phase Normalized Values will be converted in to PERCENTILES



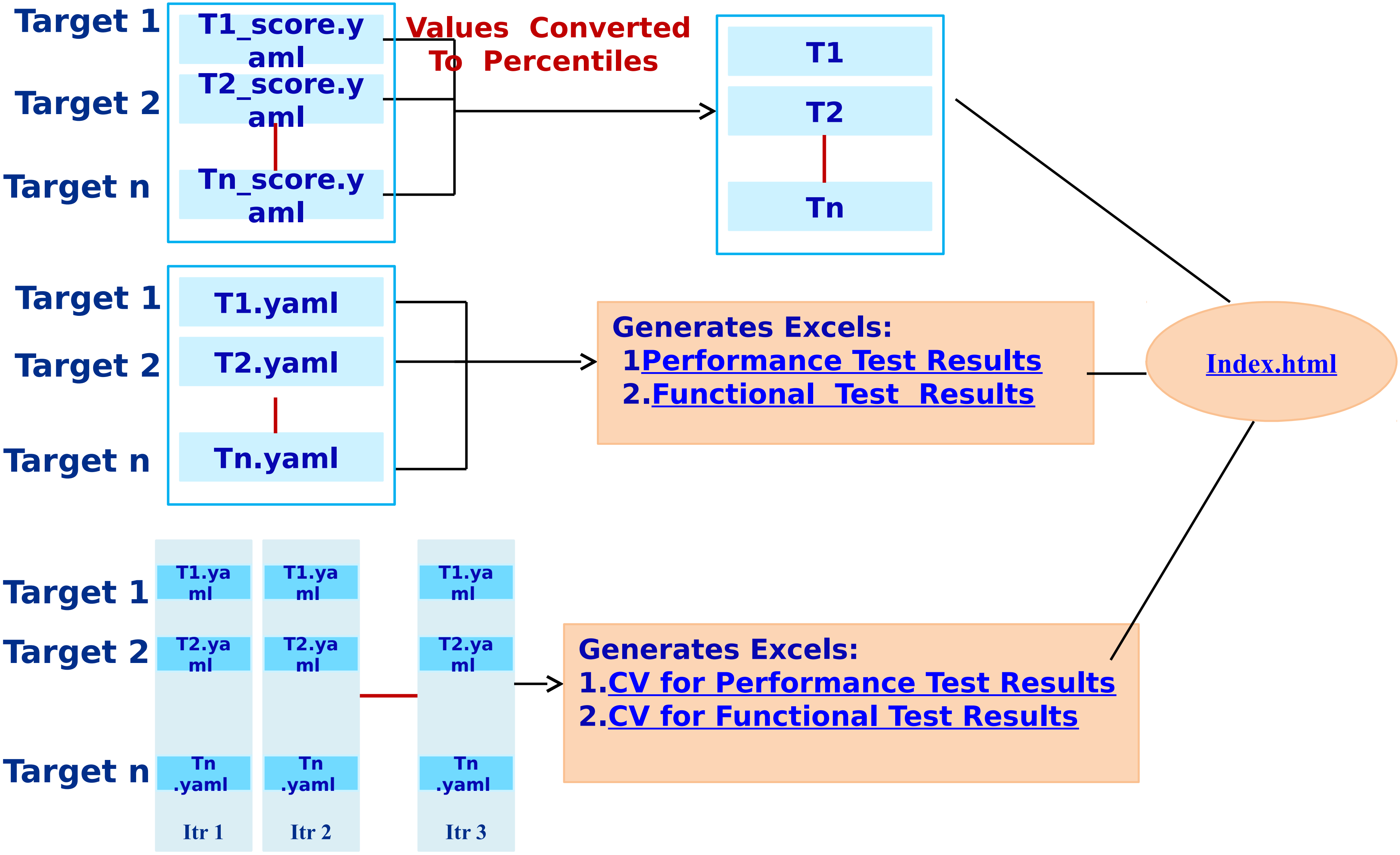
Platform_score.yaml

Report Generation phase

Percentile Values

Caliper Report Generation

index.html will be generated in Report Generation Phase



Target 1
Target 2
Target n

T1_score.yaml
T2_score.yaml
Tn_score.yaml

Values Converted To Percentiles

T1
T2
Tn

Target 1
Target 2
Target n

T1.yaml
T2.yaml
Tn.yaml

Generates Excels:
1Performance Test Results
2.Functional Test Results

Index.html

Target 1
Target 2
Target n

T1.yaml
T2.yaml
Tn .yaml
Itr 1

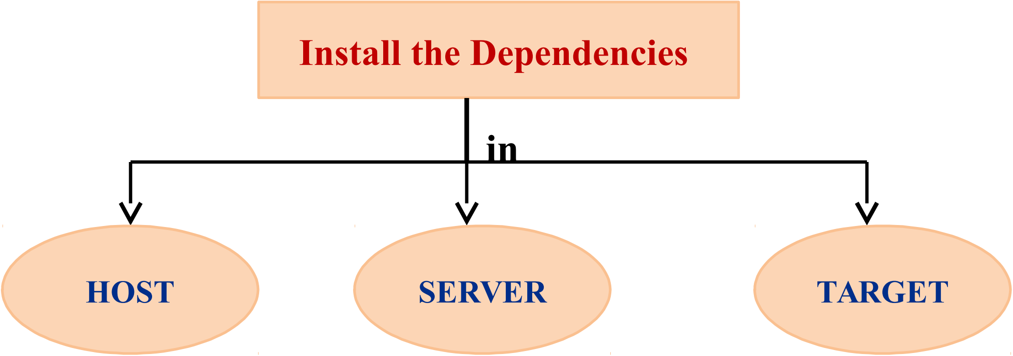
T1.yaml
T2.yaml
Tn .yaml
Itr 2

T1.yaml
T2.yaml
Tn .yaml
Itr 3

Generates Excels:
1.CV for Performance Test Results
2.CV for Functional Test Results

Caliper Setup

1. **Software Requirements:** certain dependencies for tools in caliper which should be satisfied before executing those tools.



Using http://open-estuary.org/caliper/benchmarking

2. **Download Caliper :**

```
git clone https://github.com/open-estuary/caliper.git
```

3. **Install Caliper :**

```
sudo python setup.py install
```

THANK YOU