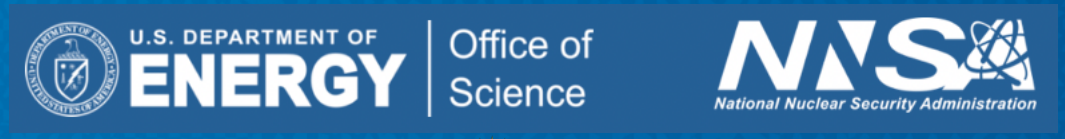
KENT STATE
UNIVERSITY

BEE-Chameleon Connecting Cloud to HPC

Qiang Guan

Kent State University



KENT STATE UNIVERSITY

Chameleon User Meeting 2019

BEE and BEE Ecosystem

KENT STATE UNIVERSITY

Challenge of deploying applications on HPC systems

Applications → Configure → Compile/ Install →

git

GNU Make

CMake
Cross-platform Make

User needs to know how to use all these tools

BASH
THE BOURNE-AGAIN SHELL

MPI

SSH

Package Manager

Example: ParaView ~50 configurable parameters ~1h compilation time

KENT STATE UNIVERSITY

Challenge of deploying applications on different systems

Applications

Configure

Compile/ Install

User needs to configure applications for each platform

Configure

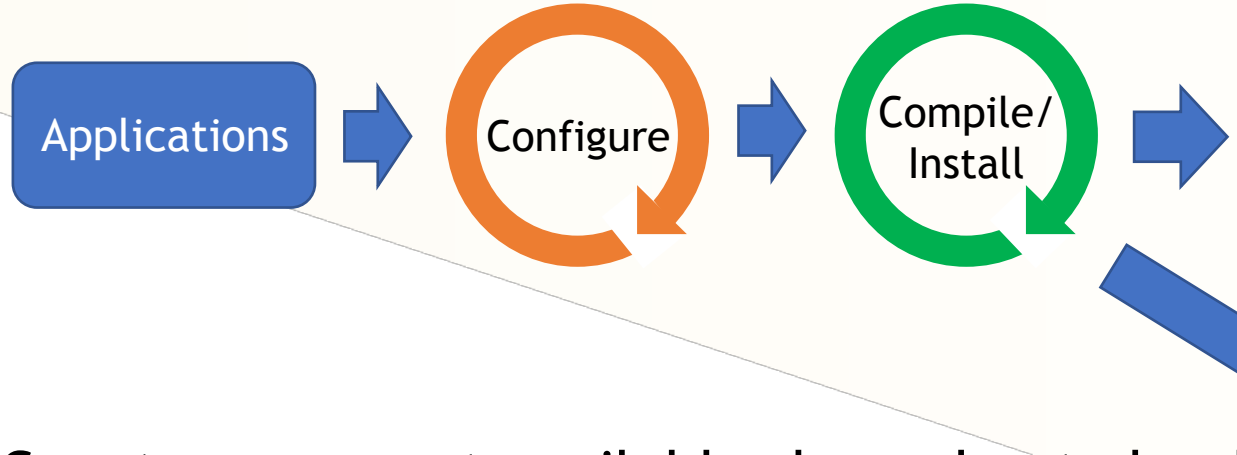
Compile/ Install

Configure

Compile/ Install

KENT STATE UNIVERSITY

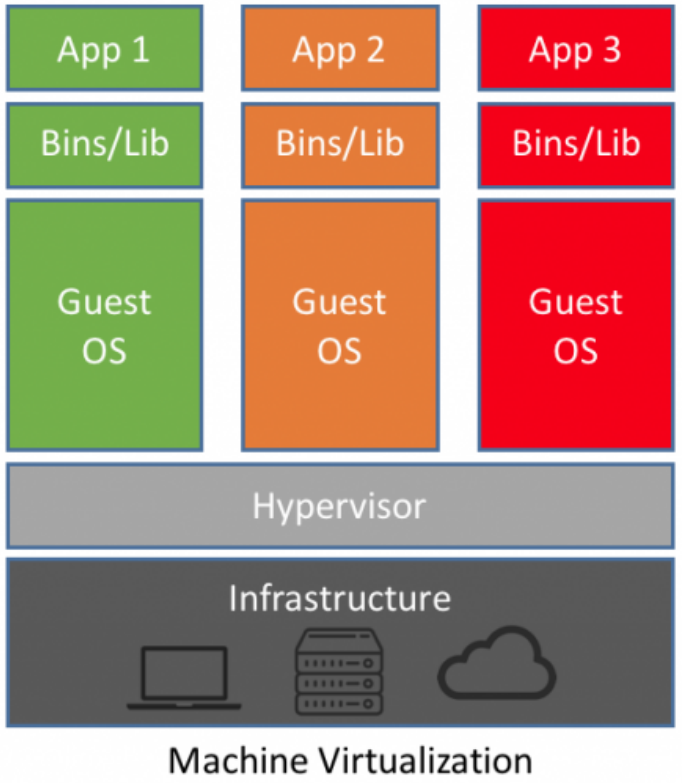
Challenge of using systems with limited resources



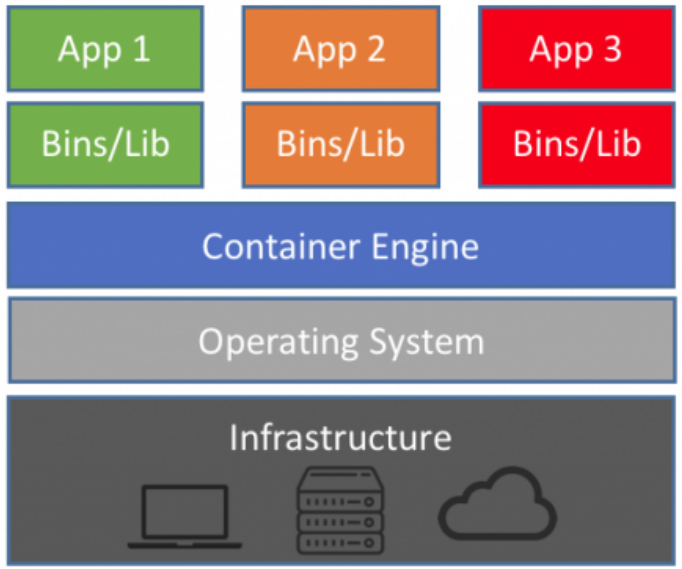
- When HPC systems are not available, how about cloud?
- There are many commercial cloud platforms that can provide computing resources.
- However, we still need to configure applications for cloud.

KENT STATE UNIVERSITY

How to improve portability? Isolation



Machine Virtualization



Containers

Traditional way, large image

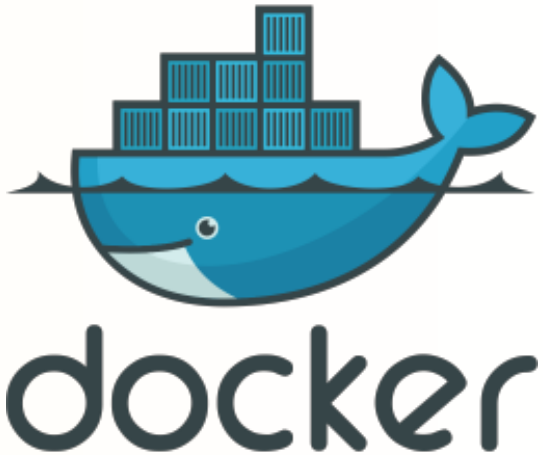
Modern way, light-weight image

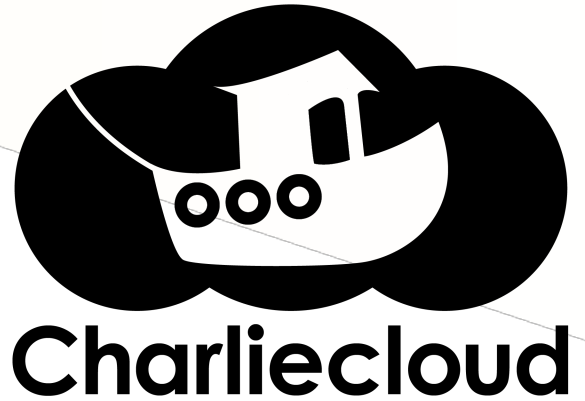
KENT STATE UNIVERSITY

Using Linux containers on HPC systems is difficult

Which Linux container to use?

- Docker
  - Most widely used ✓
  - Requires privileged access to systems → usually hard to obtain on HPC systems ✗
- Charliecloud *(developed by LANL)**
  - Based on Linux user namespace → No privileged operations ✓
  - Supports Docker images after conversion ✓

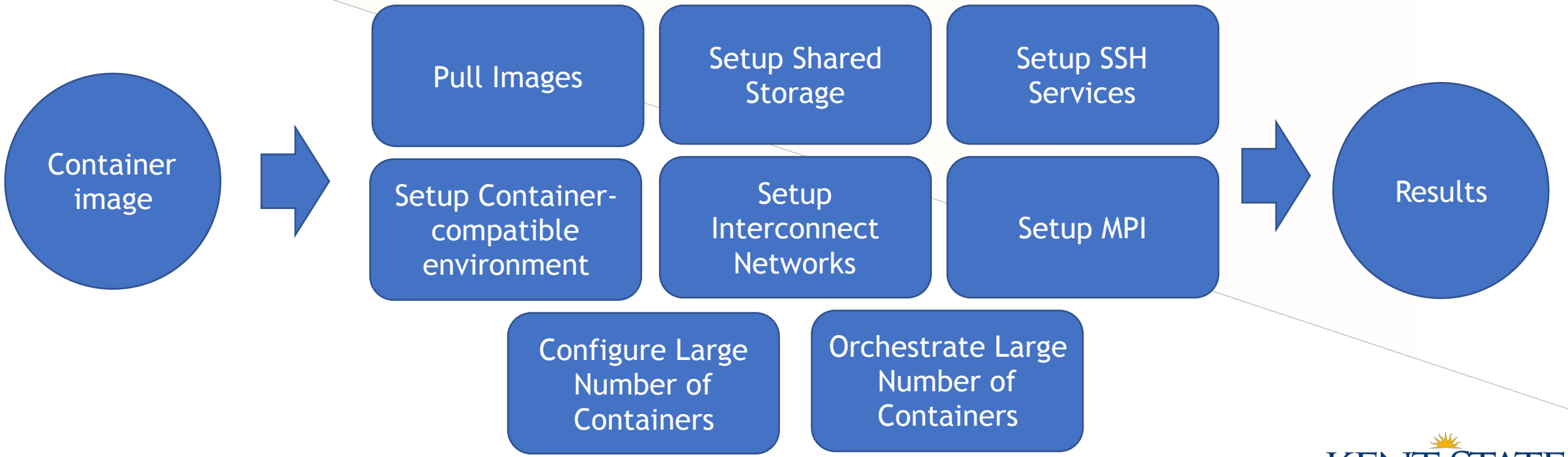




*source: https://github.com/hpc/charliecloud

Deploying containers on target systems is non-trivial

Need to create HPC-like environment for applications inside container



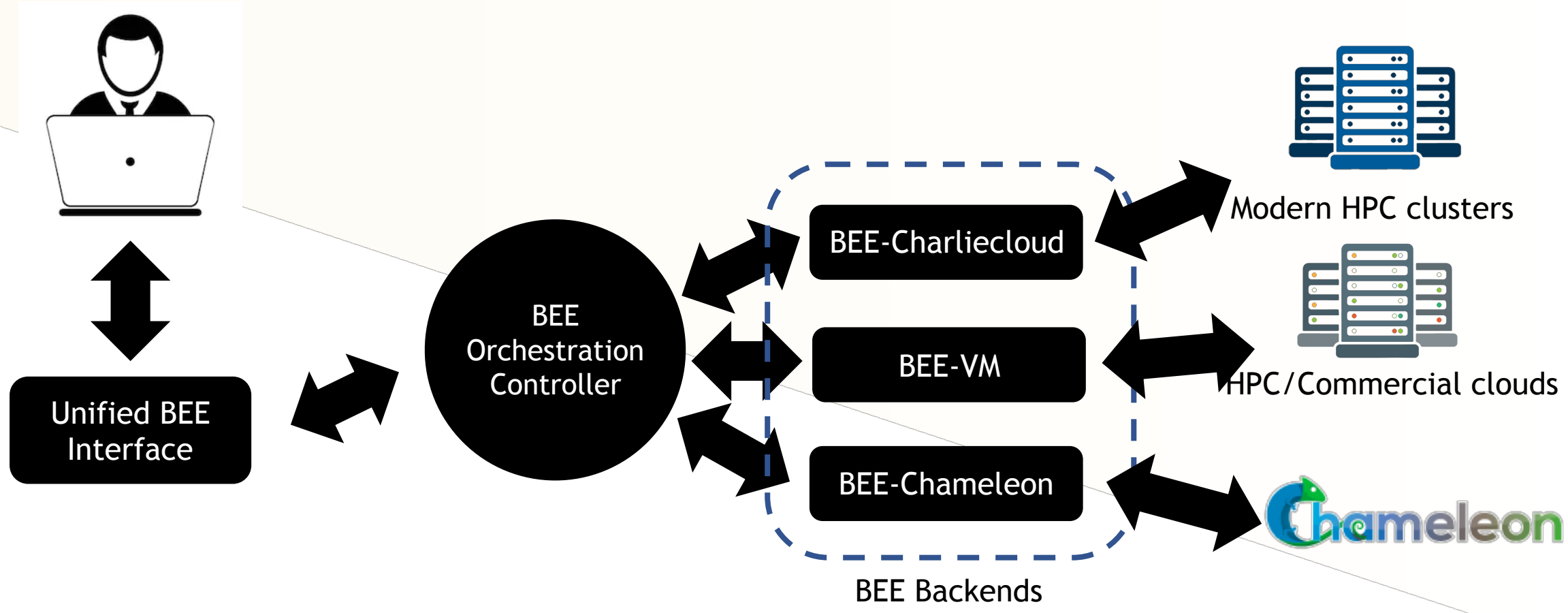
KENT STATE UNIVERSITY

Build and Execution Environment (BEE)

- **BEE: a containerized build and execution framework**
  - **Docker image support**
    - → **Avoid complicated application configurations**
  - **Multi-platform support:** HPC, Cloud
    - → **Enables high resource usability**
  - **Unified User Interface:** one interface for all platform
    - → **Avoid application configuration for each platforms**
  - **Enables high portability:** similar containerized environment across platforms
    - → **Make sure applications can yield same result regardless the platform chosen.**
  - **Automation:** all platform-related setup, configuration, and launching are automatic.
    - → **Avoid requiring detailed technical knowledge.**
  - **No privileged access required**
    - → **Any user can use**

KENT STATE UNIVERSITY

BEE framework



KENT STATE UNIVERSITY

Unified User Interface of BEE

BeeFile + Run Scrips + Docker Image

One set of input enables application running everywhere

```
"task_conf": {
 "task_name": <task name>,
 "exec_target": bee_cc|bee_vm|bee_aws|
     bee_os,
 "general_run": [
  { "script": <script 1> },
  { "script": <script 2> }, ...
 ],
 "mpi_run": [
  { "script": <script 1> },
  { "script": <script 2> }, ...
 ]
},
"docker_conf": {
 "docker_img_tag": <docker image>,
 "docker_username": <username>,
 "docker_shared_dir": <dir>
},
"exec_env_conf": {
 "bee_cc": {...} or
 "bee_vm": {...} or
 "bee_aws": {...} or
 "bee_os": {...}
}
```

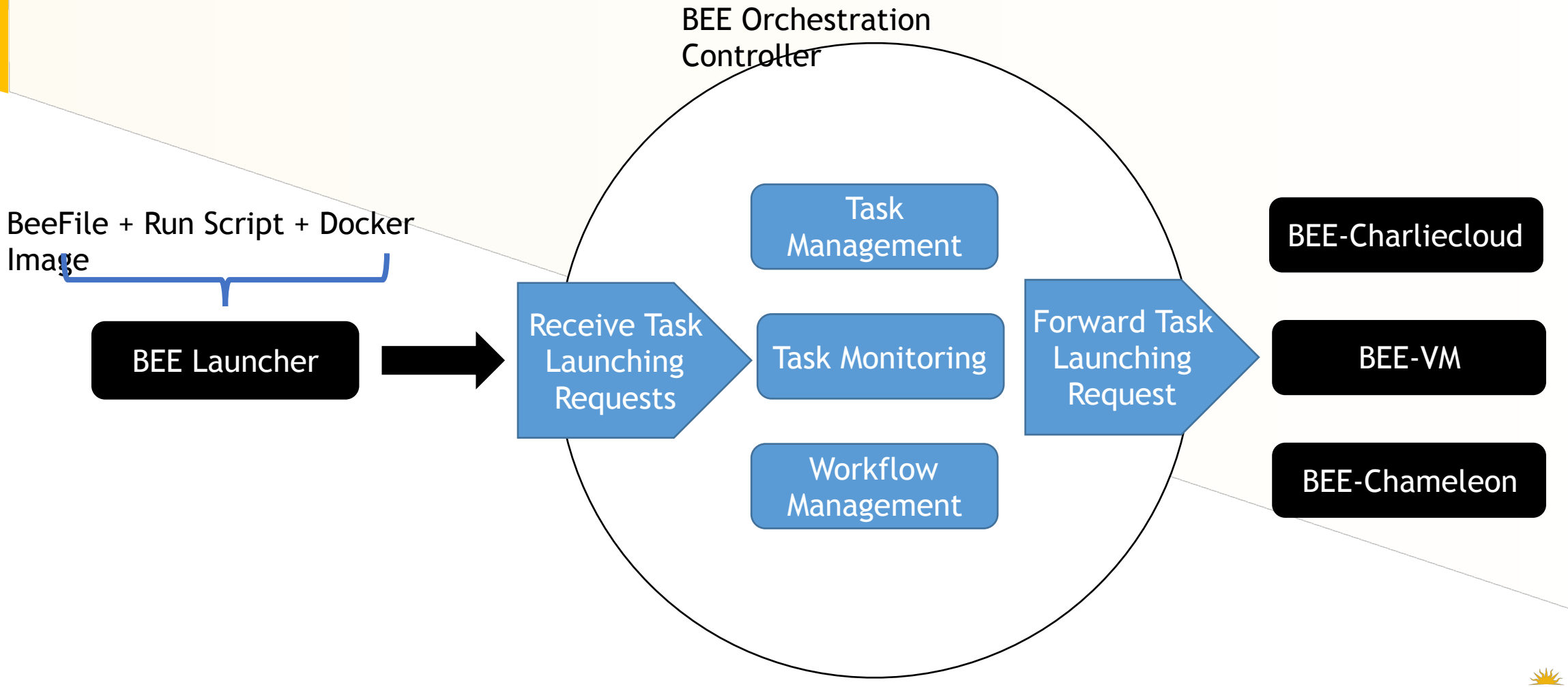
Template of BeeFile

Run Script for starting applications

Dockerized applications

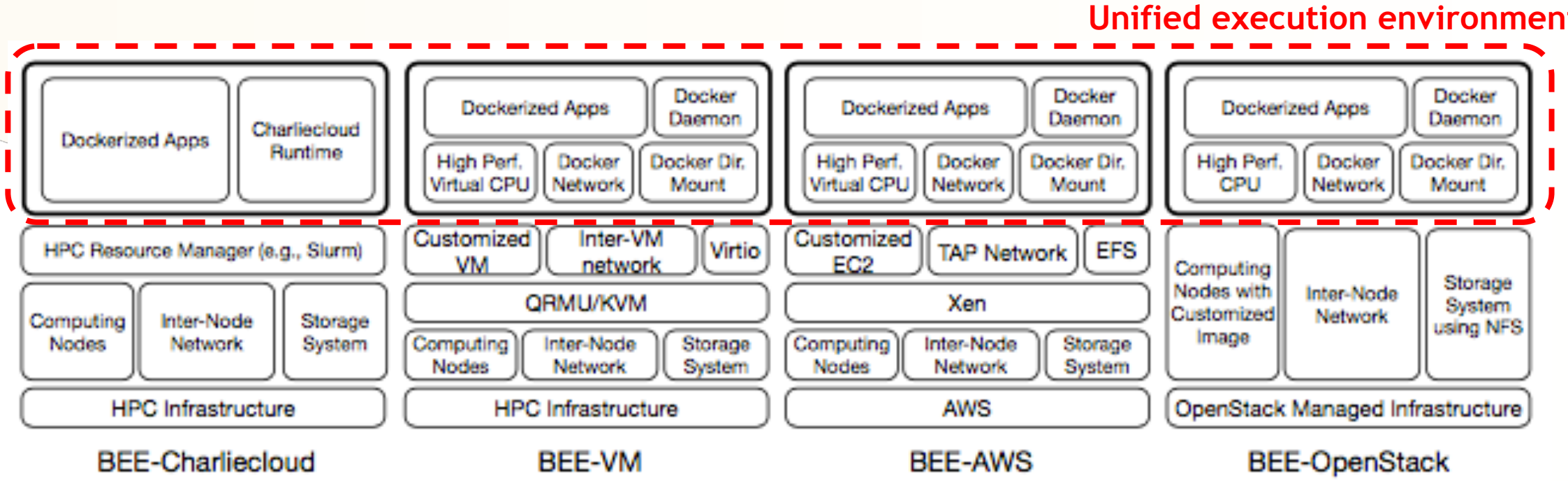
KENT STATE UNIVERSITY

BEE Launcher and Orchestration Controller



KENT STATE UNIVERSITY

BEE Backends



- Support modern HPC infrastructures with Linux namespace support;
- Low performance overhead;
- Slurm Integration.

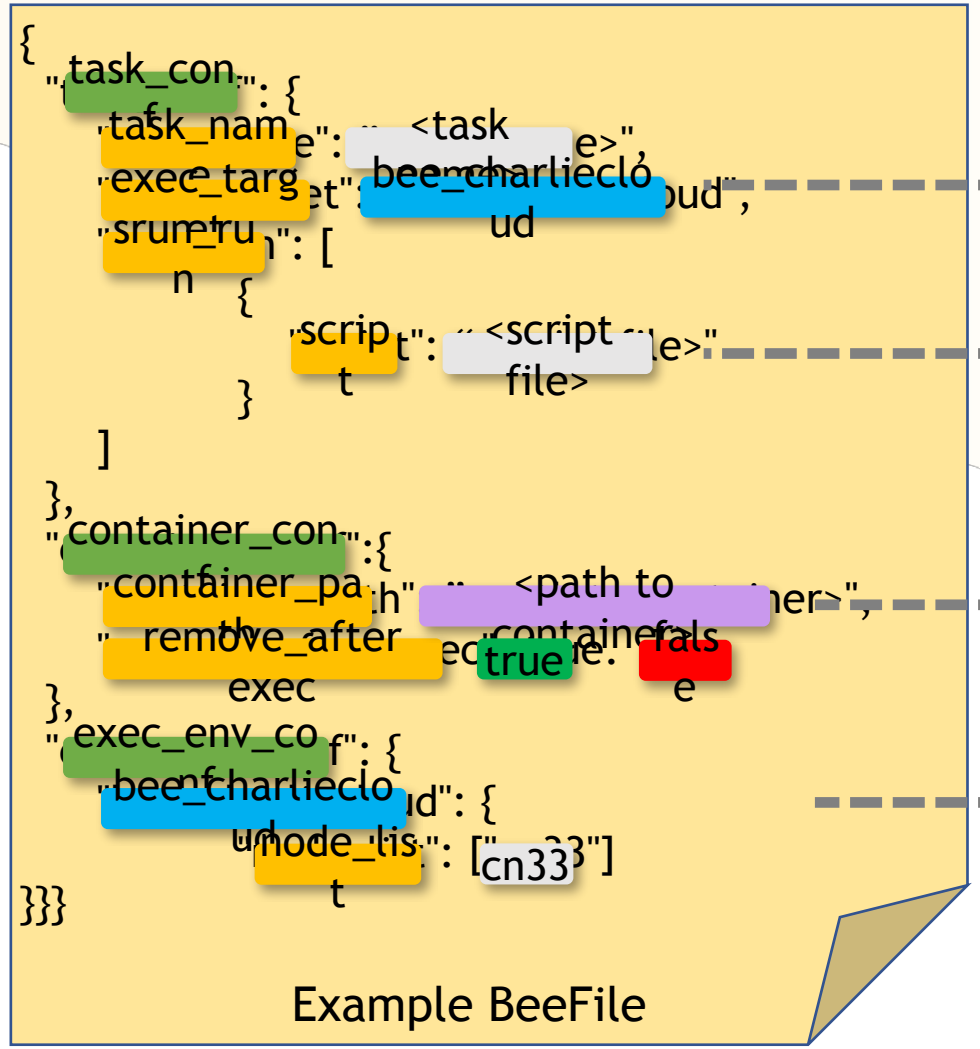
- Support most HPC infrastructures via QEMU/KVM;
- High generality.

- Support AWS EC2
- No local HPC infrastructures required

- Support OpenStack based infrastructures;
- No local HPC infrastructures required

KENT STATE UNIVERSITY

BeeFile: Task description file for BEE



Example BeeFile

Target platform to run application.

Path to run script.

Path to container.

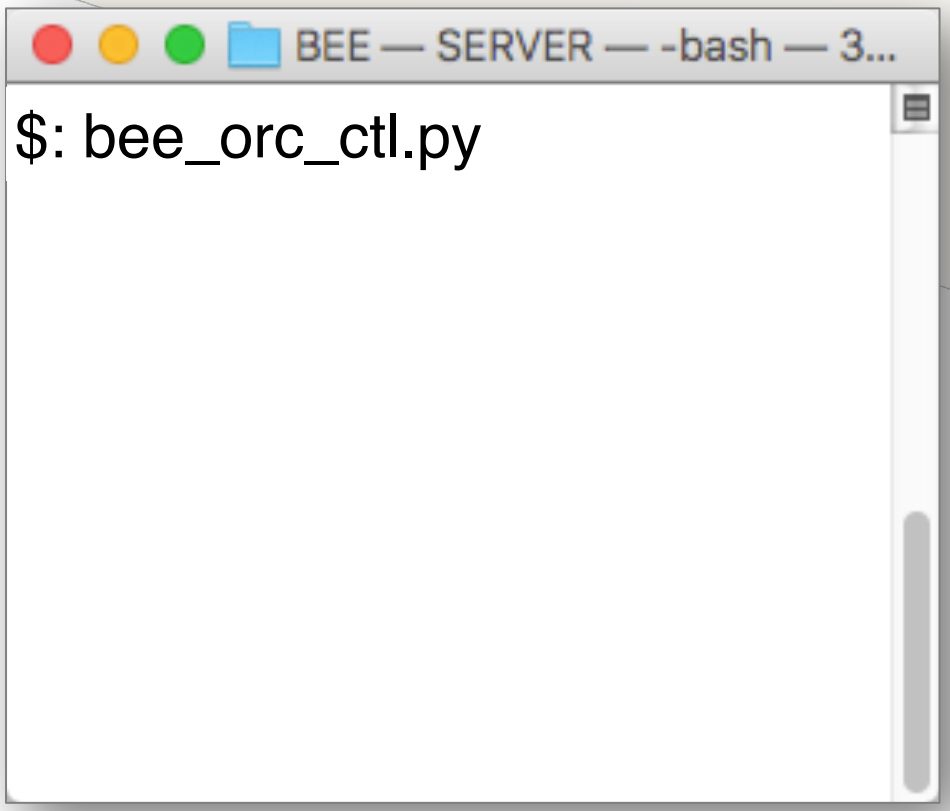
Platform specific configurations.

KENT STATE UNIVERSITY

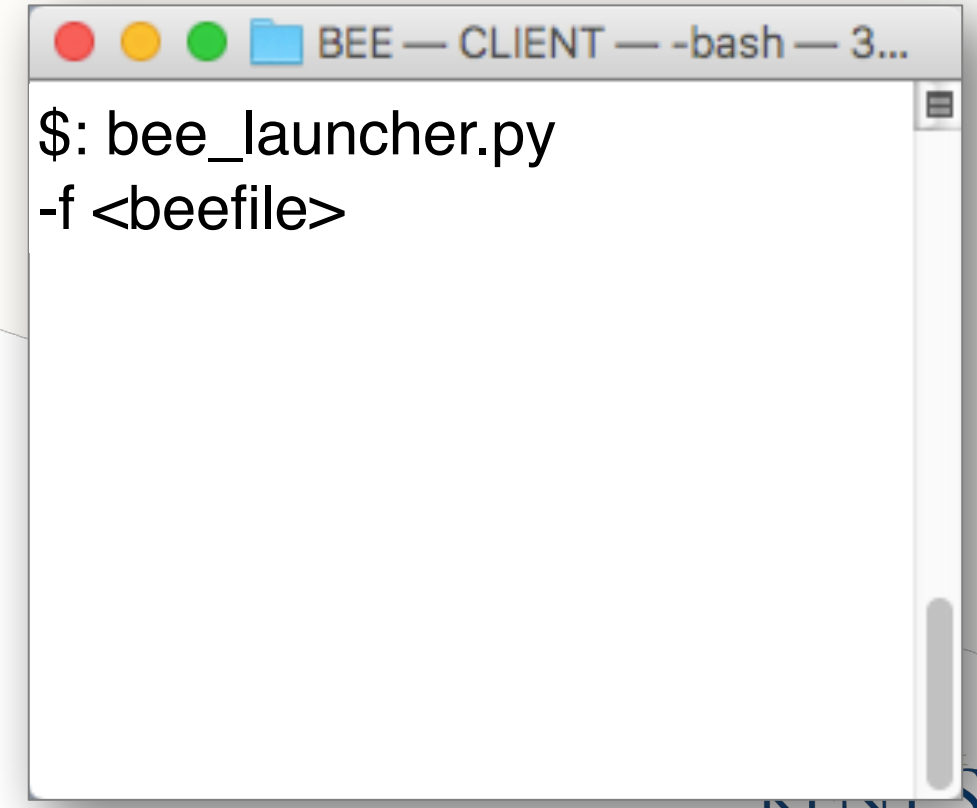
How to launch a task using BEE?

Two steps, the rest are all automatic.

Step 1: Start Orchestration Controller



Step 2: Start task using BEE Launcher

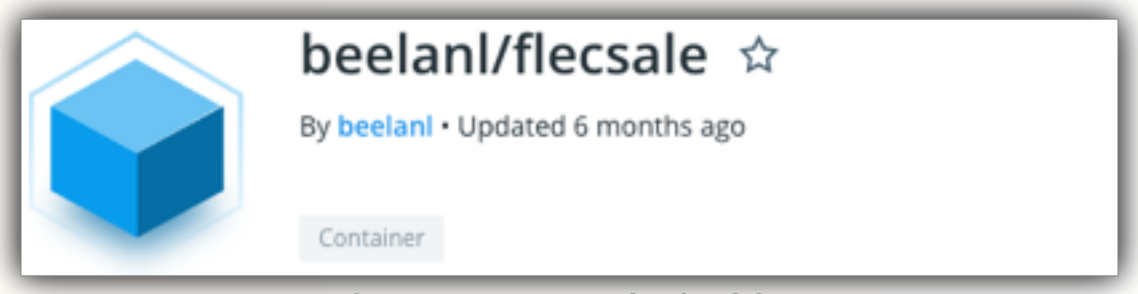


KENT STATE UNIVERSITY

Example: input for launching FleCSALE

```
{
  "task_conf": {
      "task_name": "flecsale_mpi",
      "exec_target": "bee_charliecloud",
      "mpi_run": [
          {
            "script": "flecsale_mpi_run.sh",
            "node_list": ["cn30", "cn31"],
            "flags": {
                "--map-by": "ppr:5:node"
            }} ] },
  "container_conf":{
    "container_path": .flecsale.tar.gz",
    "container_tar2dir": "/var/tmp",
    "remove_after_exec": true
  },
  "exec_env_conf": {
      "bee_charliecloud": {
        "node_list": ["cn30", "cn31"]
}}}
```

Beefile for FleCSALE



Containerized FleCSALE

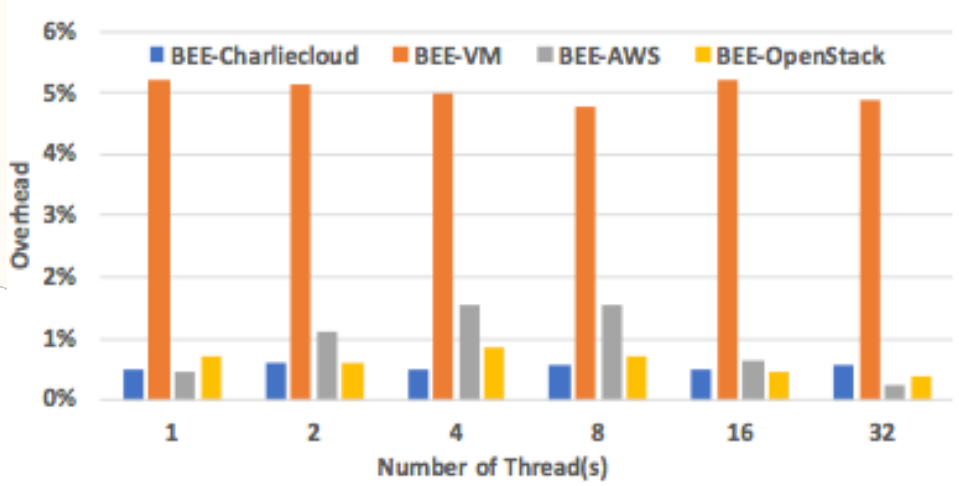
```
#! /bin/bash
mkdir -p output
ch-run -w -v --join --no-home -b output --cd /mnt/0 /
var/tmp/flecsale -- /home/flecsi/flecsale/build/
apps/hydro/2d/hydro_2d -m  /home/flecsi/flecsale/
specializations/data/meshes/2d/square/
square_32x32.g
```

Run script for FleCSALE

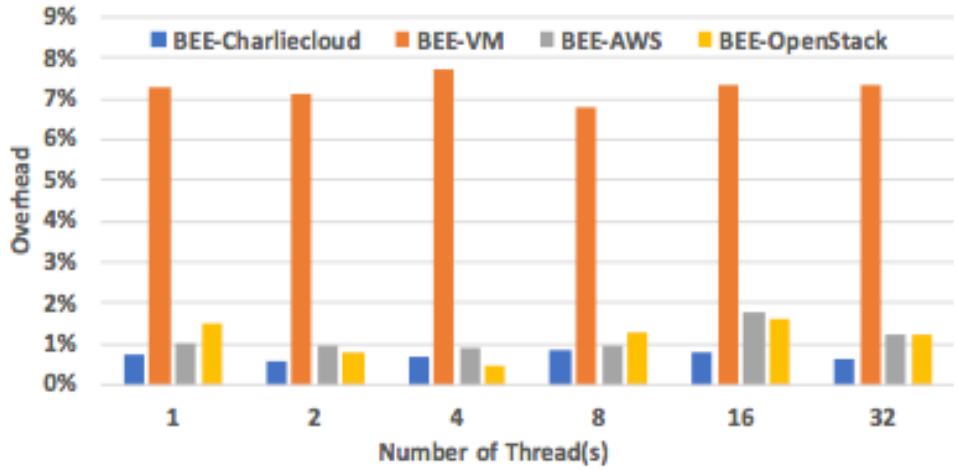
KENT STATE UNIVERSITY

Overhead

Computation

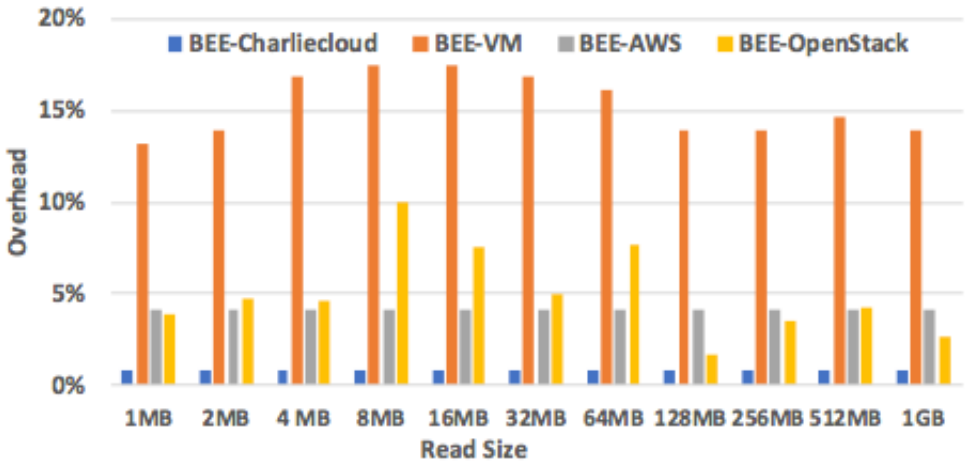


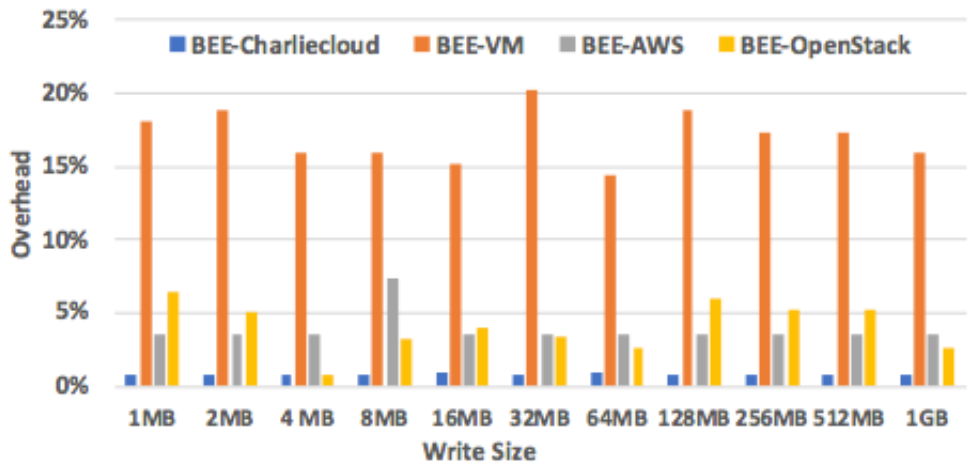
(a) Compute intensive workload (BT)



(b) Memory intensive workload (IS)

I/O





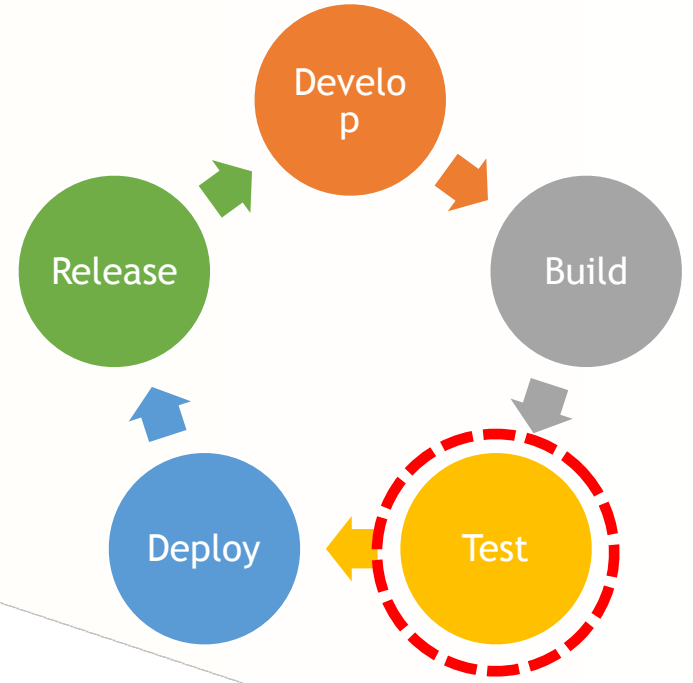
KENT STATE UNIVERSITY

BEE Swarm: Scalability Performance Test CI

KENT STATE UNIVERSITY

Scalability test is missing in Continuous Integration

- **Continuous Integration (CI) services enables:**
  - Full automation test;
  - Catching code issues early;
  - Testing in a clone of production environment;
- **However, scalability testing is missing in current CI services.**



Only correctness check is not sufficient for HPC applications.

KENT STATE UNIVERSITY

Why do scalability testing in HPC?

- Reconfiguration and deployment are complicated and time consuming.
- Need performance tests across platforms, infrastructures and hardware.
- Complexity of the computation kernels needs to track the changes reflecting on the changes of performance.

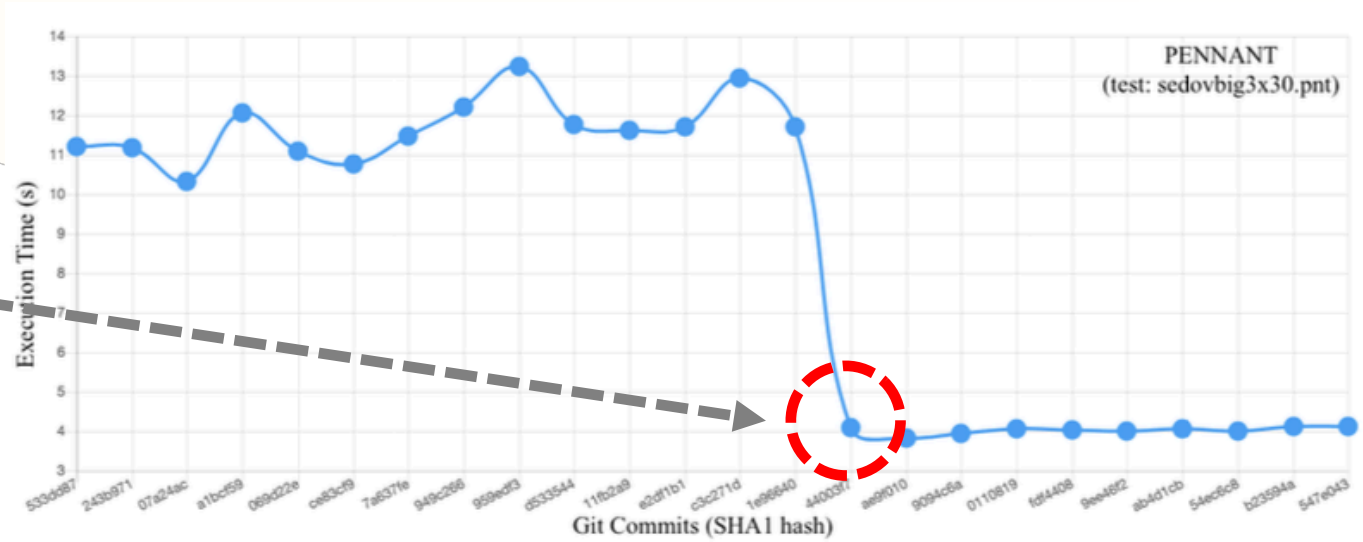
KENT STATE UNIVERSITY

Example: Performance/scalability can change

- Performance/scalability can change as developers make progress
- Keeping track of performance/scalability is necessary for developers

Commit HASH	Commit message
725e549dc	legion: fixing a potential hang with old-style bounds checking
3edff3290	regent: small bug fix to openmp code generation for regent
d0b157755	tools: small bug fix for legion prof ascii deserializer
1162649ea	legion: small bug fix for dependence analysis of close operations involving different children in different modes for the same field
2818b5fe9	legion: small bug fix for remote advances of version numbers
824d6c77d	legion: fixing a bug where we were not properly paging in version states for remote virtual mappings

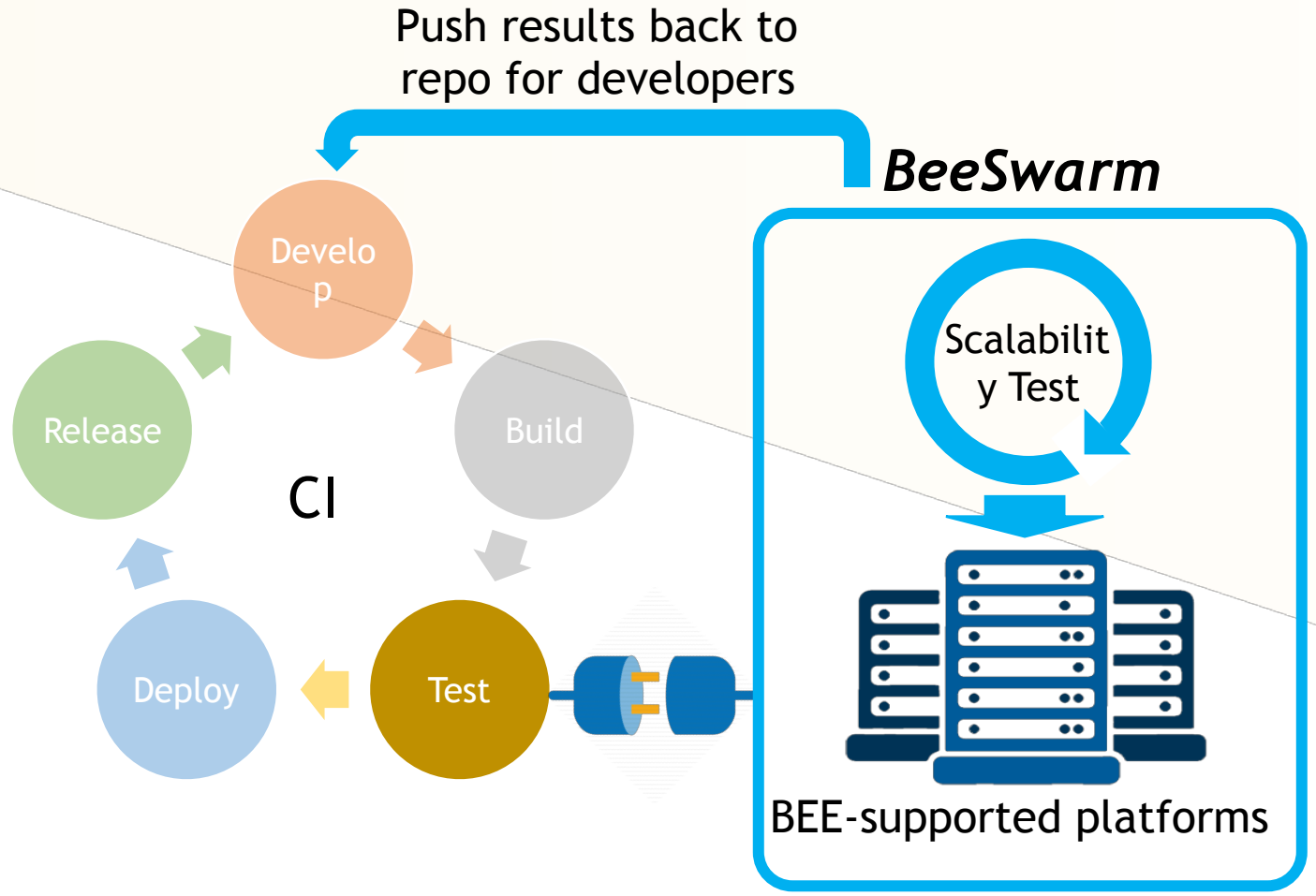
Several commits in the legion commit tree that may cause the performance improvement after commit 4400



Performance of PENNANT on Legion

KENT STATE UNIVERSITY

BeeSwarm: Scalability test plugin for CI



Supported CI services:

Travis CI

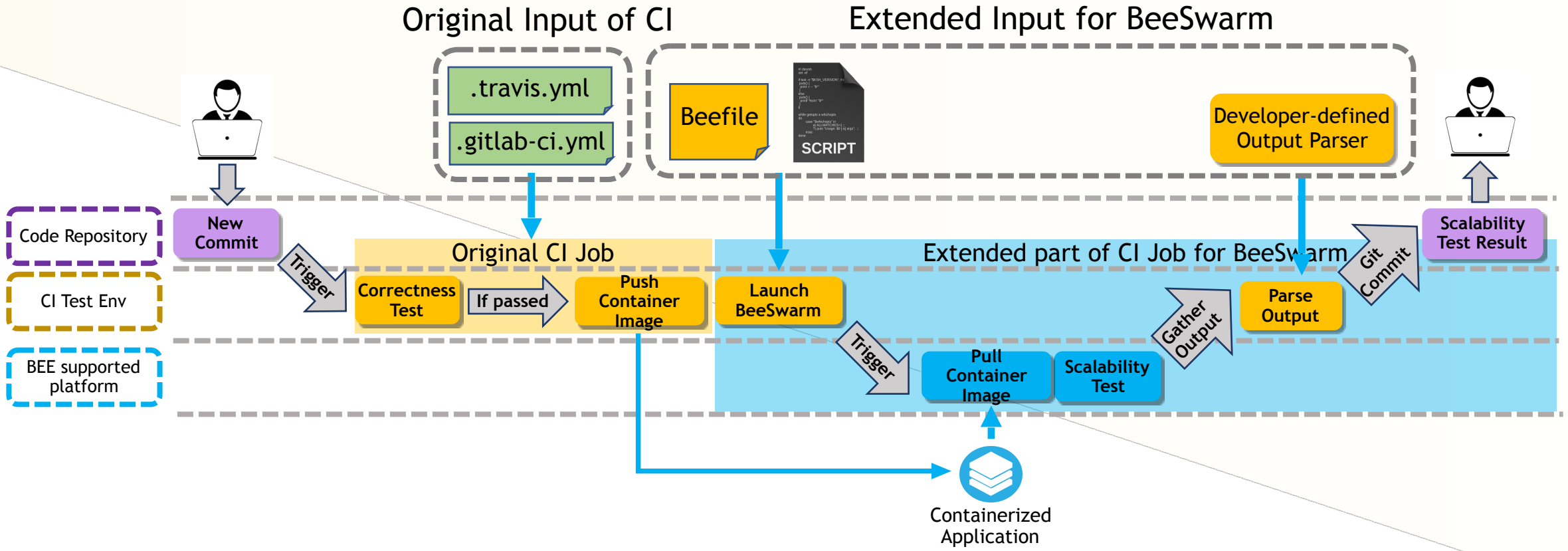
GitLab CI

Supported scalability test platform*:

Chameleon

KENT STATE UNIVERSITY

Workflow of BeeSwarm



KENT STATE UNIVERSITY

Input of BeeSwarm

Beefile

Scalability test configurations.

Developer-defined Output Parser

Developer defined executable used to parse output of each scalability test run

Run Script for applications

Run script(s) for scalability test to run against.

.travis.yml for Travis CI

.gitlab-ci.yml for GitLab CI

CI script to configure and launch BeeSwarm in CI environment.

KENT STATE UNIVERSITY

How to compose beefile for scalability test?

```
{
  "task_conf": {
      "task_name": "flecsale",
      "exec_target": "bee_os",
      "general_run": [
                      {
                        "script": "flecsale_run.sh",
                      }
      ],
      "scalability_run": {
          "mode" : "linear" or "exp2" or "exp10" or
"independent",
          "node_range": [1,2],
          "proc_range": [1,16],
          "script": "flecsale_run.sh"
      }
  },
  "docker_conf":{
      "docker_img_tag": "beelanl/flecsale",
      "docker_username": "flecsi",
      "docker_shared_dir": "/mnt/docker_share"
  },
  "exec_env_conf": {
      "bee_os": {
          "num_of_nodes": "2"
}}}
```

mode: linear | node_range: [1,2] | proc_range: [1,4]

Nodes	Proc/ node
1	1
1	2
1	3
1	4
2	1
2	2
2	3
2	4

mode: exp2 | node_range: [1,2] | proc_range: [1,8]

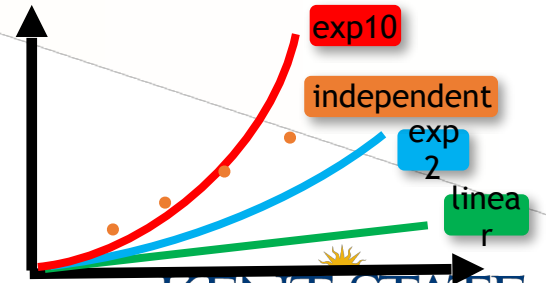
Nodes	Proc/ node
1	1
1	2
1	4
1	8
2	1
2	2
2	4
2	8

mode: exp10 | node_range: [1,100] | proc_range: [1,10]

Nodes	Proc/ node
1	1
1	10
10	1
10	10
100	1
100	10

mode: independent | node_range: [2, 3, 4, 5] | proc_range: [6, 7, 8, 9]

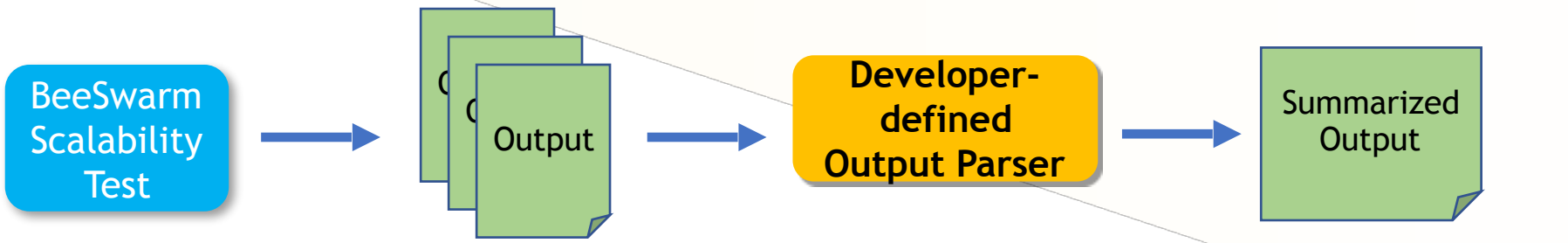
Nodes	Proc/ node
2	6
3	7
4	8
5	9



Scalability test progress

Developer-defined Output Parser

- **The definition of *performance* usually varies from application to application.**
- **→ We left the job of parsing out performance results to developers.**



One output file per test configuration.

Summarized output file to be pushed back to code repository.

KENT STATE UNIVERSITY

How to compose CI script for BeeSwarm?

.travis.yml or .gitlab-ci.yml

```
# Setup BeeSwarm
- git clone https://@github.com/lanl/BEE.git && cd ./BEE
- ./install_for_ci.sh

# Start scalability test
- cd ${CI_PROJECT_DIR}/bee_scalability_test
- bee_ci_launcher.py -l flecsale -r $OS_RESERVATION_ID

# Parse output
- ./output_parser.py

# Push results back to repo
- git add
bee_scalability_test_result_build_$CI_COMMIT_SHA.csv
- git commit --message "[skip ci]"
- git push remote_repo $CI_COMMIT_REF_NAME
```

Install BEE on CI environment.

Launch BeeSwarm scalability test.

Parse and gather results.

Push scalability test results back to repo.

KENT STATE UNIVERSITY

Example: Configure FleCSALE with BeeSwarm

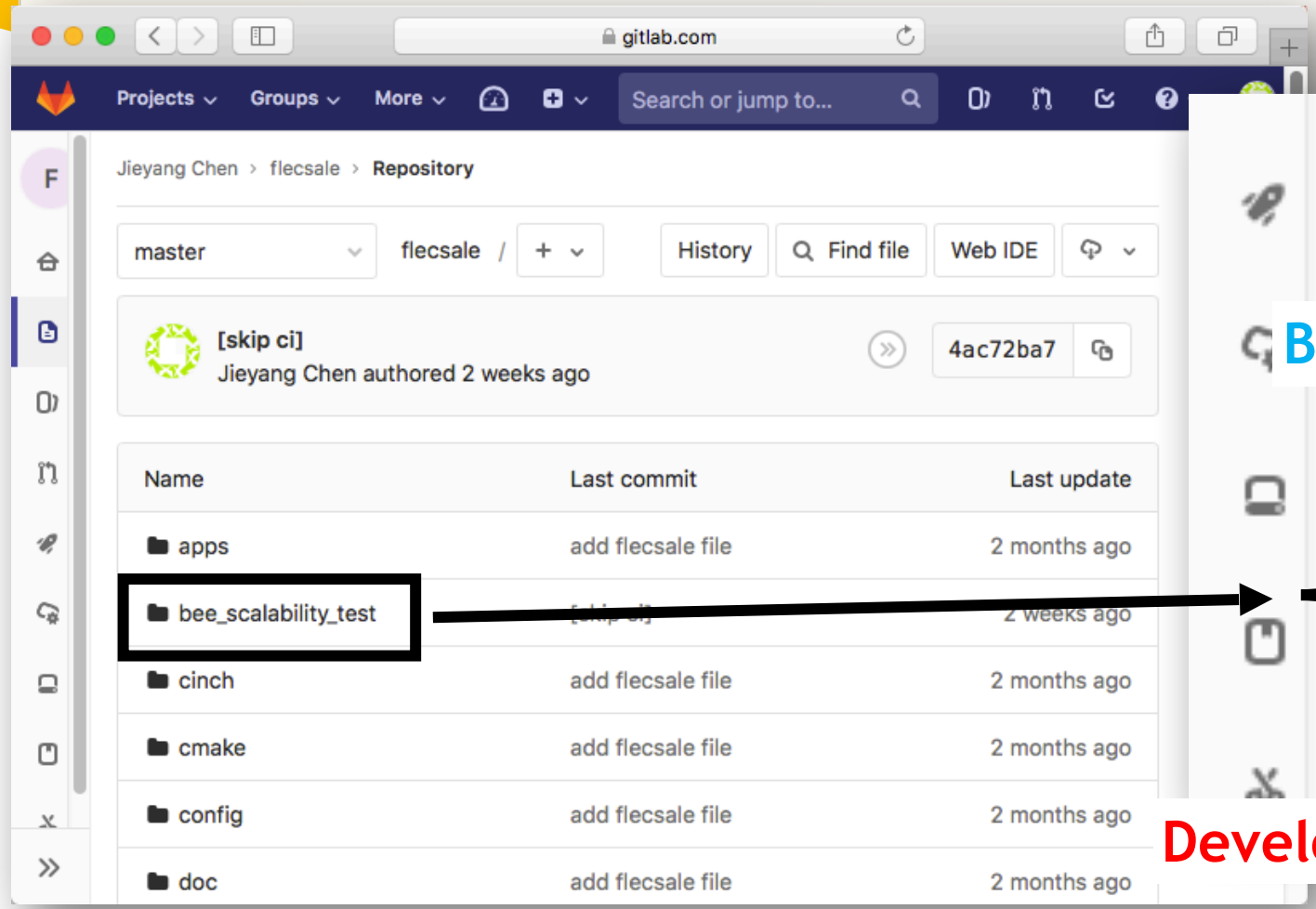
- **Step 1: Create a directory in the repository dedicated for scalability test**
  - E.g.: <repo>/beeswarm_scalability_test
- **Step 2: Create a beefile**
- **Step 3: Add an executable result parser**
- **Step 4: Setup environment variable in Travis CI or GitLab CI**
- **Step 5: Set timeout for scalability test command**
- **Step 6: Configure the `.travis.yml` file for Travis CI or `.gitlab-ci.yml` for GitLab CI**

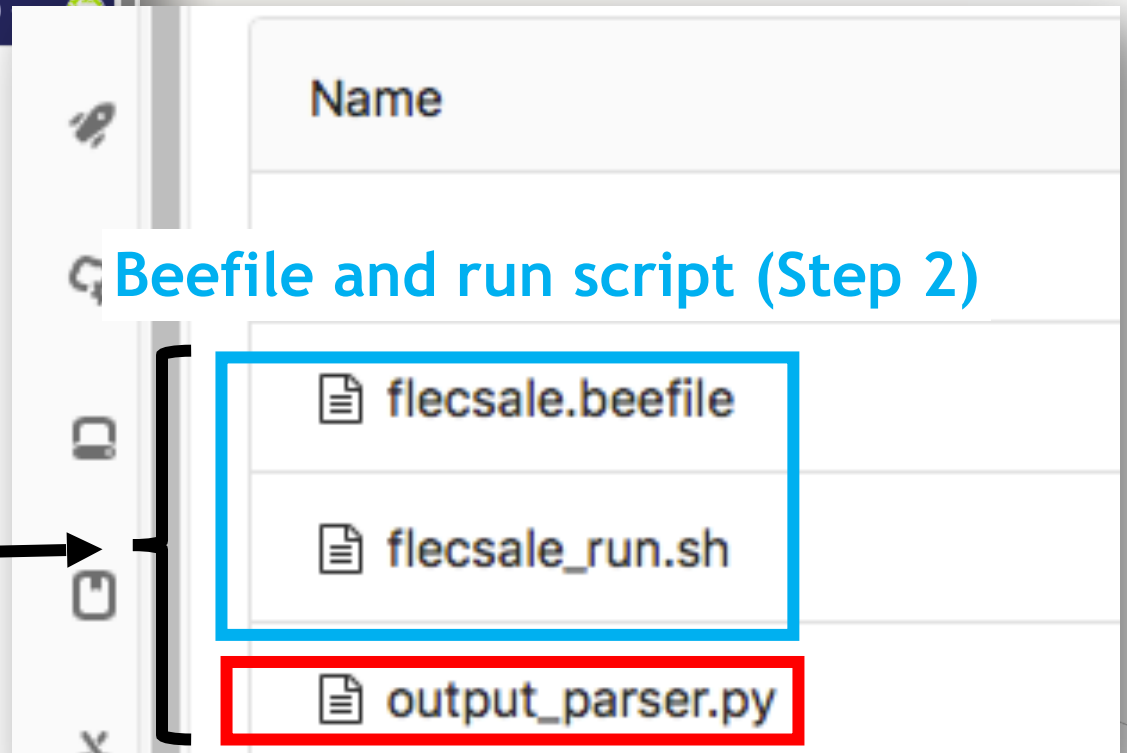
KENT STATE UNIVERSITY

Example: Configure FleCSALE with BeeSwarm

Step 1: Create a directory in the repository dedicated for scalability test.

- E.g.: <repo>/beeswarm_scalability_test





Beefile and run script (Step 2)

Developer-defined Output Parser (Step 3)

Example: Configure FleCSALE with BeeSwarm

Step 2: Create a beefile

```
{
  "task_conf": {
      "task_name": "flecsale",
      "exec_target": "bee_os",
  },
      "scalability_run": {
          "mode": "exp2",
          "node_range": [1,2],
          "proc_range": [1,16],
          "script": "flecsale_run.sh"
      }
  },
  "docker_conf": {
      "docker_img_tag": "beelanl/flecsale",
      "docker_username": "flecs",
      "docker_shared_dir": "/mnt/docker_share"
  },
  "exec_env_conf": {
      "bee_os": {
          "num_of_nodes": "2"
      }
  }
}
```

mode: exp2
node_range: [1,2]
proc_range: [1,8]

Nodes	Proc/ node
1	1
1	2
1	4
1	8
2	1
2	2
2	4
2	8

```
#!/bin/bash
/home/flecsi/flecsale/build/
apps/hydro/2d/hydro_2d -m  /
home/flecsi/flecsale/
specializations/data/meshes/
2d/square/square_32x32.g
```

Run script (flecsale_run.sh)

Containerized Application

Example: Configure FleCSALE with BeeSwarm

Step 3: Add an executable result parser

```
=============================================
| Step: 23 | Time: 1.846518e-01 | Step Size: 7.354746e-03 |
=============================================
| Step: 24 | Time: 1.919882e-01 | Step Size: 7.336449e-03 |
=============================================
| Step: 25 | Time: 1.993145e-01 | Step Size: 7.326294e-03 |
=============================================
```

One output log file per test configuration.

```
#!/usr/bin/python
import getpass
import glob
output_file = open('bee_scalability_test_parsed.output','w')
output_file.write("Number of nodes, Process per node, Execution time(s)\n")
path = "bee_scalability_test_*_.output"
for filename in sorted(glob.glob(path)):
    parsed_filename = filename.split("_")
    num_of_node = parsed_filename[len(parsed_filename)-3]
    proc_per_node = parsed_filename[len(parsed_filename)-2]
    with open(filename) as f:
        lines = f.readlines()
        for line in lines:
            if "Elapsed wall time" in line:
                words = line.split(" ")
                word = words[len(words) - 1]
                output_file.write("%13d,%15d,%15f\n" %
(int(num_of_node), int(proc_per_node), float(word
3])))
output_file.close()
```

output_parser.py

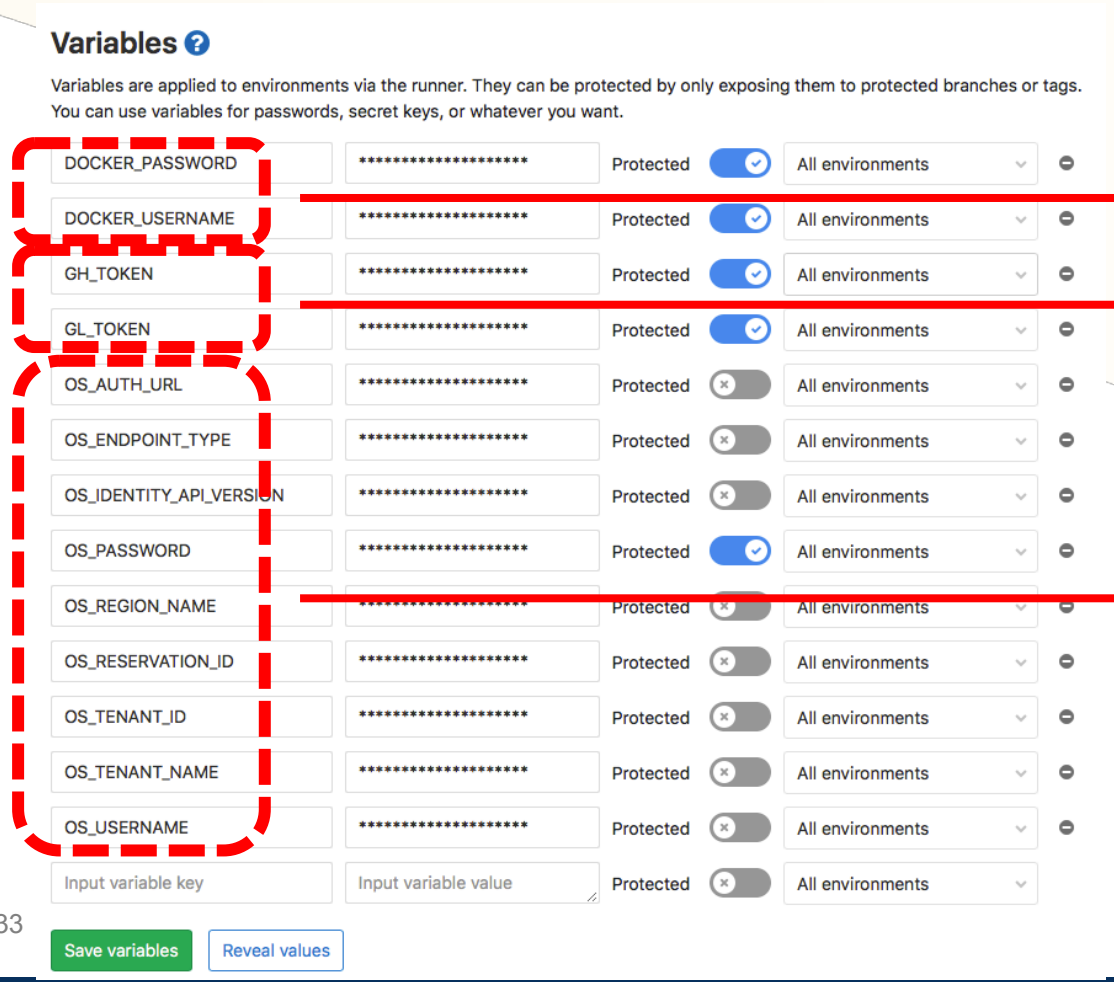
```
Number of nodes, Process per node, Execution time(s)
            1,              1,       0.120100
            1,              2,       0.065000
            1,              4,       0.036400
            1,              8,       0.021600
            1,             16,       0.014300
            2,              1,       0.092900
            2,              2,       0.085700
            2,              4,       0.074500
            2,              8,       0.072800
            2,             16,       0.068600
```

Summarized performance results

KENT STATE UNIVERSITY

Example: Configure FleCSALE with BeeSwarm

- **Step 4: Setup environment variable in Travis CI or GitLab CI**



Access DockerHub

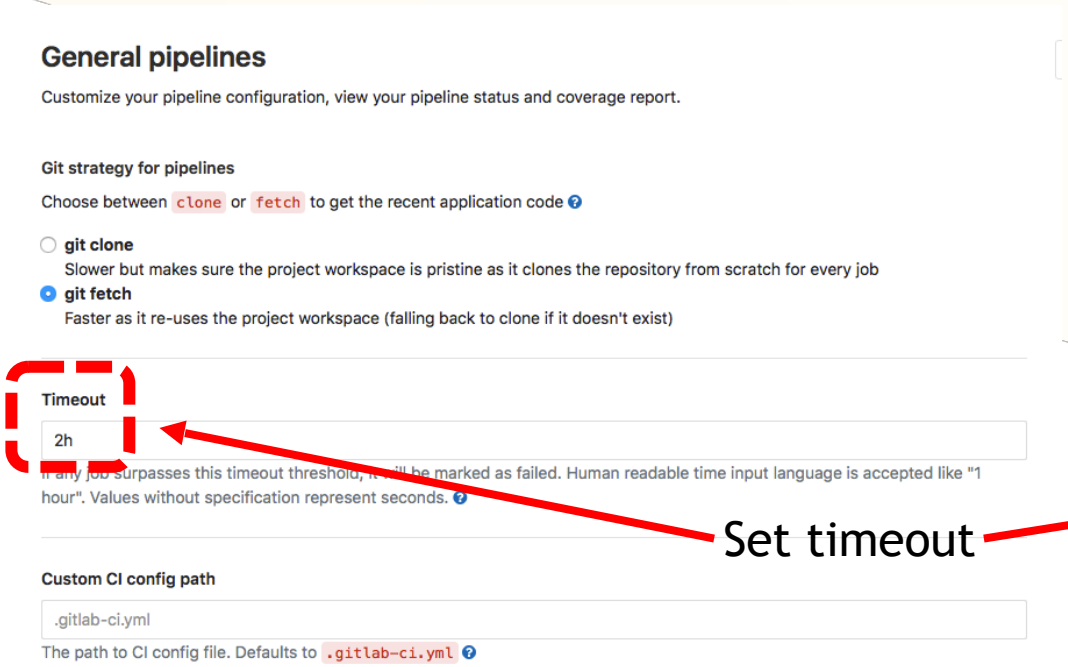
Access GitLab or GitHub

Access OpenStack Platform

KENT STATE UNIVERSITY

Example: Configure FleCSALE with BeeSwarm

- **Step 5: Set timeout for scalability test command**



Set timeout

```
48  # Setup BeeSwarm
49   - cd ${HOME}
50   - git clone https://${GH_TOKEN}@github.com/lanl/BEE_Private.git && cd ./BEE_Private
51   - git checkout beeswarm-enhancement
52   - ./install_for_ci.sh
53   - export PATH=$(pwd)/bee-launcher:$PATH
54   - export PATH=${HOME}/build/${TRAVIS_REPO_SLUG}/bee_scalability_test:$PATH
55
56   # Start scalability test
57   - cd ${HOME}/build/${TRAVIS_REPO_SLUG}/bee_scalability_test
58   - travis_wait 120 bee_ci_launcher.py -l flecsale -r $OS_RESERVATION_ID
59
60   # Parse results
61   - output_parser.py
```

GitLab CI settings page

Travis CI script file

KENT STATE UNIVERSITY

Example: Configure FleCSALE with BeeSwarm

- **Step 6: Configure the `.travis.yml` file for Travis CI or `.gitlab-ci.yml` for GitLab CI**

.travis.yml or .gitlab-ci.yml

```
# Setup BeeSwarm
- git clone https://@github.com/lanl/BEE.git && cd ./BEE
- ./install_for_ci.sh

# Start scalability test
- cd ${CI_PROJECT_DIR}/bee_scalability_test
- bee_ci_launcher.py -l flecsale -r $OS_RESERVATION_ID

# Parse output
- ./output_parser.py

# Push results back to repo
- git add
bee_scalability_test_result_build_$CI_COMMIT_SHA.csv
- git commit --message "[skip ci]"
- git push remote_repo $CI_COMMIT_REF_NAME
```

Install BEE on CI environment.

Launch BeeSwarm scalability test.

Parse and gather results.

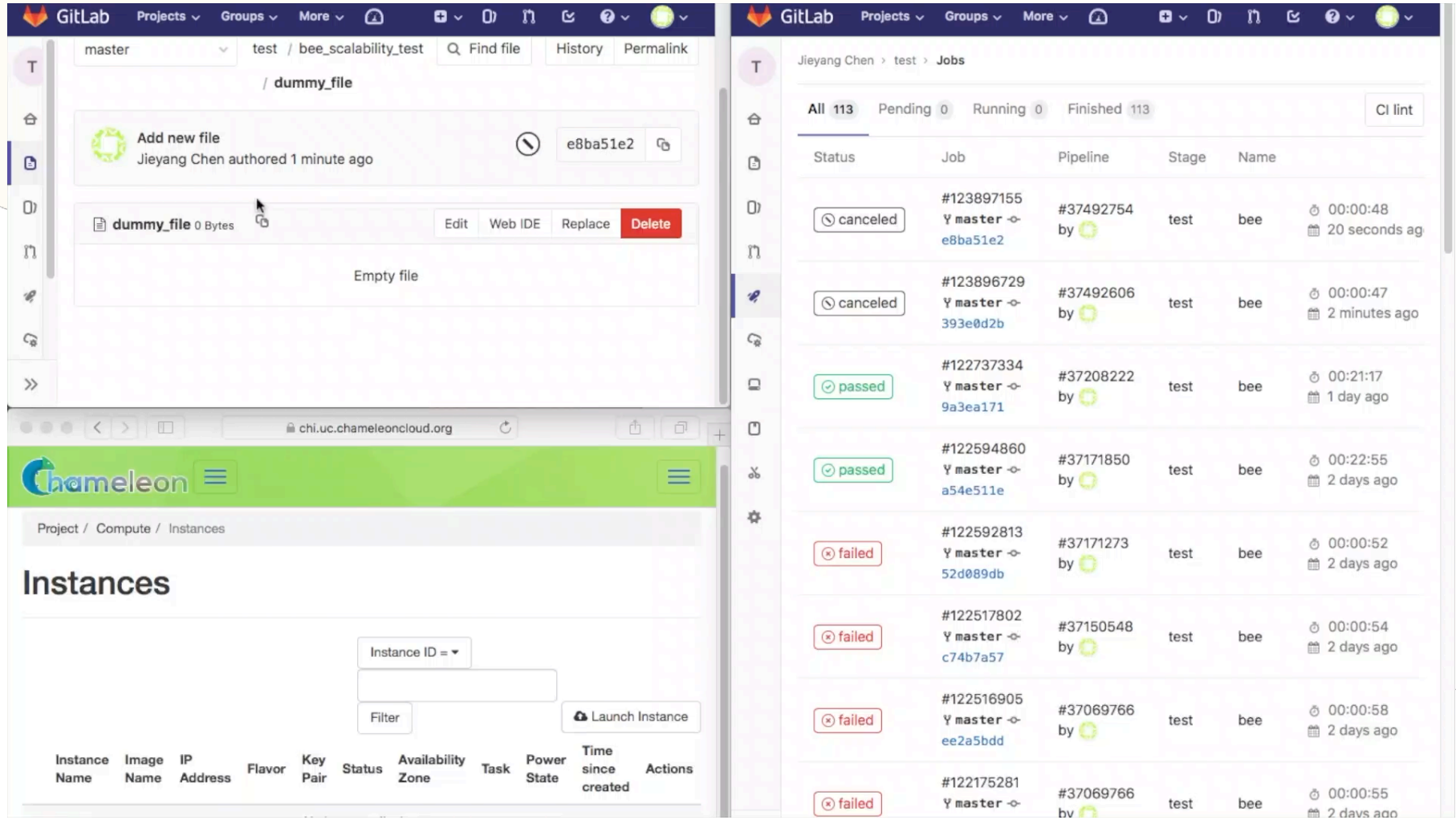
Push scalability test results back to repo.

KENT STATE UNIVERSITY

Video demo

GitLab code repository

Chameleon cloud



GitLab CI

KENT STATE UNIVERSITY

BEE Team and Collaborators

BEE Team Members:

- Tim Randles (*PI*) - Los Alamos National Laboratory - trandles@lanl.gov
- Paul Bryant - New Mexico Consortium - pbryant@newmexicoconsortium.org
- Jieyang Chen - University of California Riverside - jchen098@ucr.edu
- Patricia Grubel - Los Alamos National Laboratory - pagrubel@lanl.gov
- Qiang Guan - Kent State University - qguan@kent.edu
- Li-Ta (Ollie) Lo - Los Alamos National Laboratory - ollie@lanl.gov
- Allen McPherson - Los Alamos National Laboratory - mcpherson@lanl.gov
- Jim Ahrens – Los Alamos National Laboratory

Collaborators:

- CharlieCloud – LANL
- Flecscale – LANL
- VPIC -- LANL

Publications:

- ICDCS 2018
- IEEE Bigdata 2018
- ISSTA 2019 (submitted)

KENT STATE UNIVERSITY

Thank You.

www.cs.kent.edu/~qguan