ParslFest 2021 - Matthew Shaxted

Parsl on Parallel Works for Science and Industry

Parallel Works

PRESENTATION AGENDA

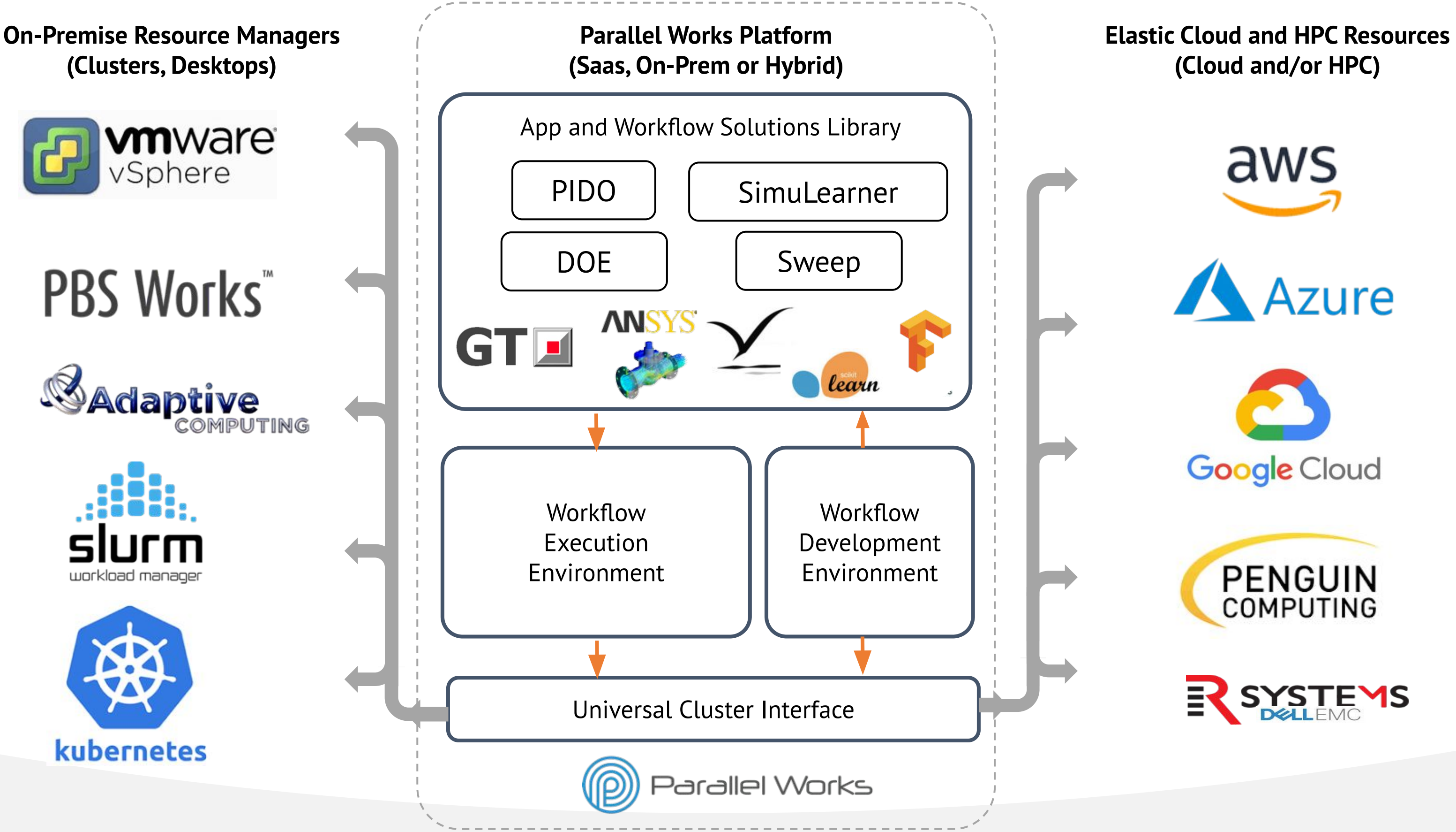
1. What is Parallel Works?
2. Platform Updates & Roadmap
3. Parsl in-the-Wild
    - Tier-1 Auto Supplier
    - Software Vendors
    - University Collaborations
    - NOAA

SECTION ONE

01

WHAT IS PARALLEL WORKS?

MIDDLEWARE HPC PLATFORM



On-Premise Resource Managers
(Clusters, Desktops)

Parallel Works Platform
(Saas, On-Prem or Hybrid)

Elastic Cloud and HPC Resources
(Cloud and/or HPC)

App and Workflow Solutions Library

PIDO

SimuLearner

DOE

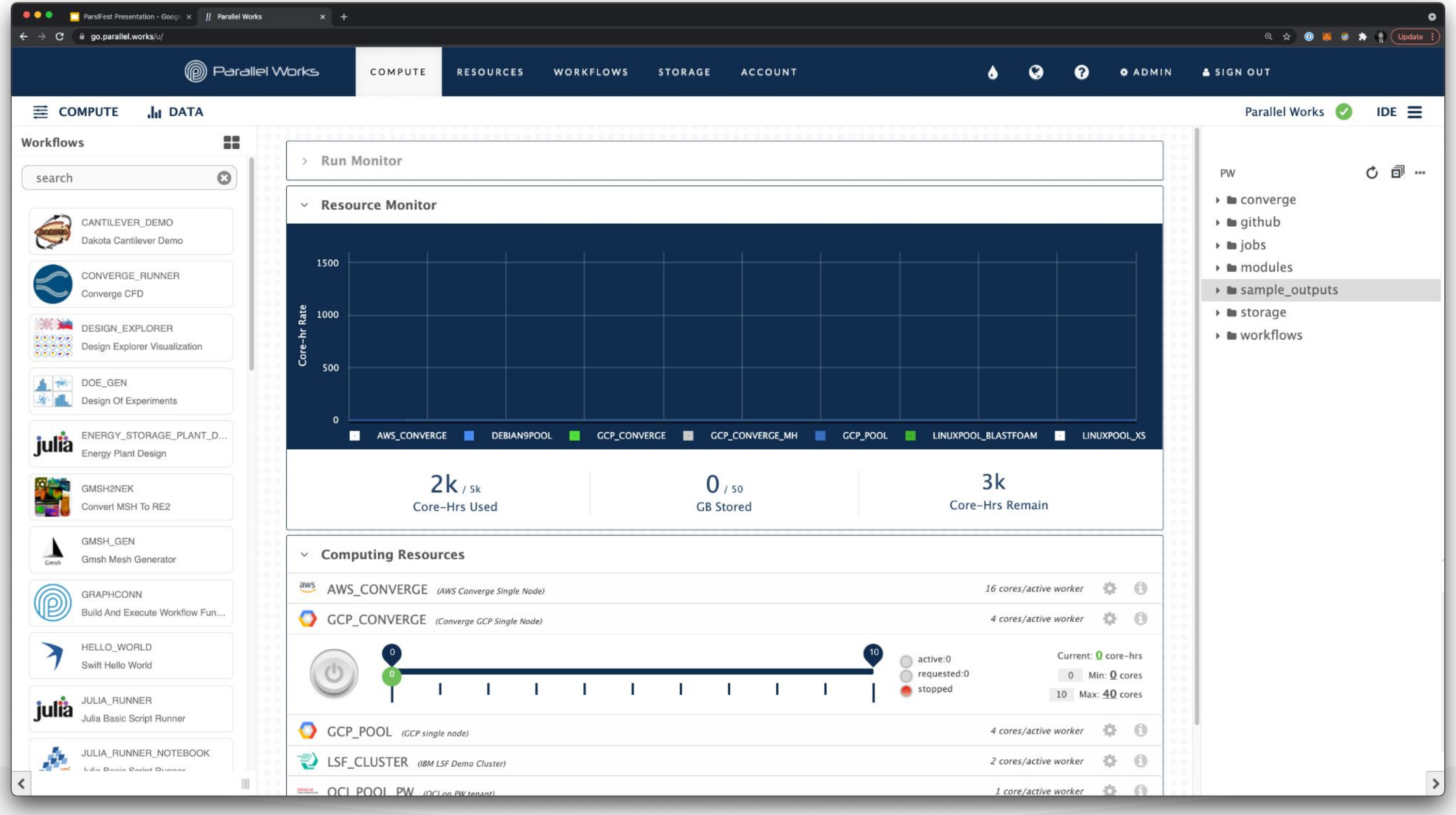
Sweep

Workflow Execution Environment

Workflow Development Environment

Universal Cluster Interface

SINGLE PANE OF GLASS WORKFLOW EXECUTION



CANTILEVER_DEMO
Dakota Cantilever Demo

CONVERGE_RUNNER
Converge CFD

DESIGN_EXPLORER
Design Explorer Visualization

DOE_GEN
Design Of Experiments

ENERGY_STORAGE_PLANT_D...
Energy Plant Design

GMSH2NEK
Convert MSH To RE2

GMSH_GEN
Gmsh Mesh Generator

GRAPHCONN
Build And Execute Workflow Fun...

HELLO_WORLD
Swift Hello World

JULIA_RUNNER
Julia Basic Script Runner

JULIA_RUNNER_NOTEBOOK

2k / 5k
Core-Hrs Used

0 / 50
GB Stored

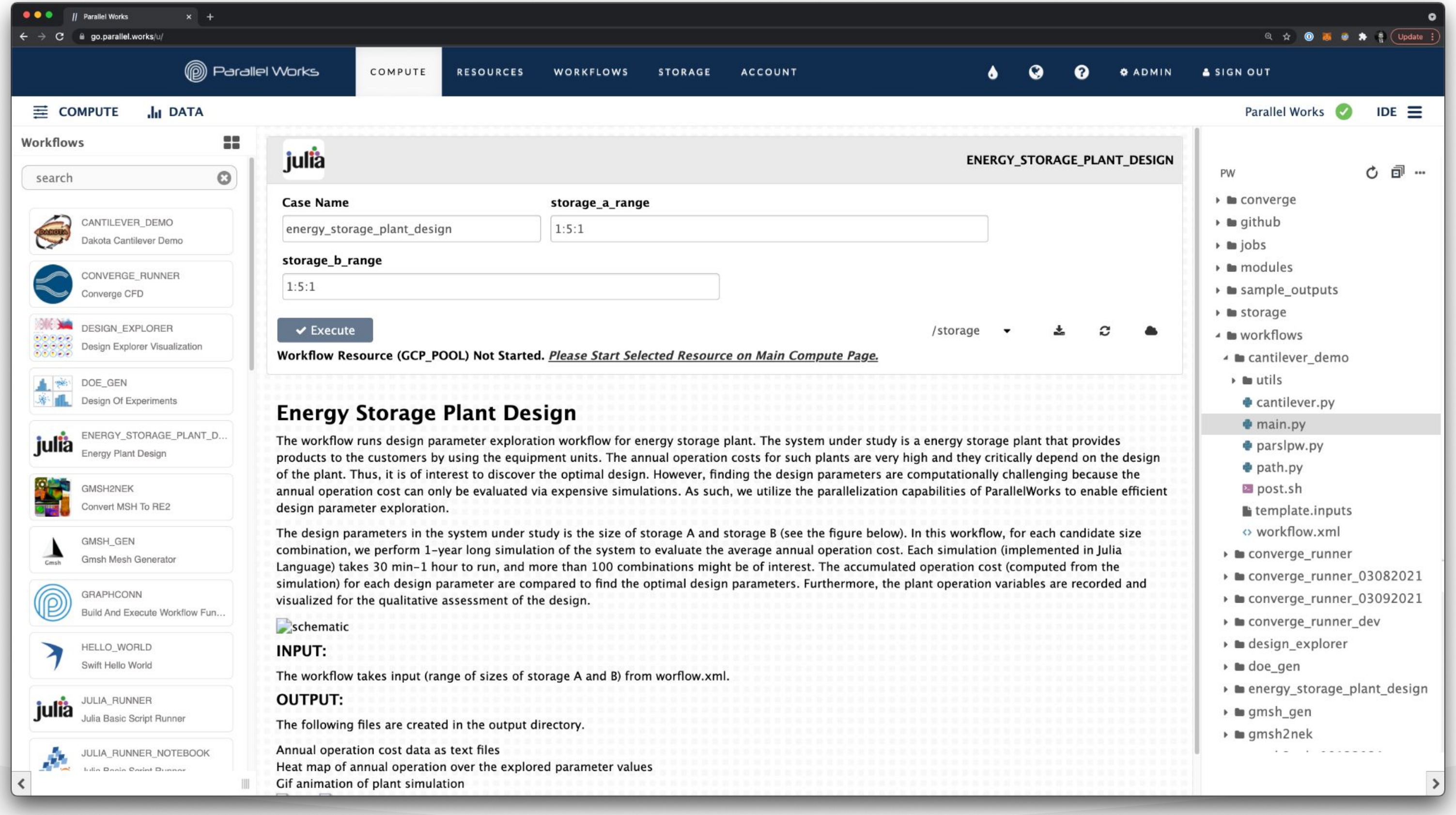
3k
Core-Hrs Remain

AWS_CONVERGE	(AWS Converge Single Node)	16 cores/active worker
GCP_CONVERGE	(Converge GCP Single Node)	4 cores/active worker
GCP_POOL	(GCP single node)	4 cores/active worker
LSF_CLUSTER	(IBM LSF Demo Cluster)	2 cores/active worker
OCI_POOL_PW	(OCI on PW tenant)	1 core/active worker

active:0
requested:0
stopped

Current: 0 core-hrs
0 Min: 0 cores
10 Max: 40 cores

SINGLE PANE OF GLASS WORKFLOW EXECUTION



CANTILEVER_DEMO
Dakota Cantilever Demo

CONVERGE_RUNNER
Converge CFD

DESIGN_EXPLORER
Design Explorer Visualization

DOE_GEN
Design Of Experiments

ENERGY_STORAGE_PLANT_D...
Energy Plant Design

GMSH2NEK
Convert MSH To RE2

GMSH_GEN
Gmsh Mesh Generator

GRAPHCONN
Build And Execute Workflow Fun...

HELLO_WORLD
Swift Hello World

JULIA_RUNNER
Julia Basic Script Runner

JULIA_RUNNER_NOTEBOOK

ENERGY_STORAGE_PLANT_DESIGN

Case Name: energy_storage_plant_design

storage_a_range: 1:5:1

storage_b_range: 1:5:1

Workflow Resource (GCP_POOL) Not Started. *Please Start Selected Resource on Main Compute Page.*

Energy Storage Plant Design

The workflow runs design parameter exploration workflow for energy storage plant. The system under study is a energy storage plant that provides products to the customers by using the equipment units. The annual operation costs for such plants are very high and they critically depend on the design of the plant. Thus, it is of interest to discover the optimal design. However, finding the design parameters are computationally challenging because the annual operation cost can only be evaluated via expensive simulations. As such, we utilize the parallelization capabilities of ParallelWorks to enable efficient design parameter exploration.

The design parameters in the system under study is the size of storage A and storage B (see the figure below). In this workflow, for each candidate size combination, we perform 1-year long simulation of the system to evaluate the average annual operation cost. Each simulation (implemented in Julia Language) takes 30 min-1 hour to run, and more than 100 combinations might be of interest. The accumulated operation cost (computed from the simulation) for each design parameter are compared to find the optimal design parameters. Furthermore, the plant operation variables are recorded and visualized for the qualitative assessment of the design.

schematic

INPUT:

The workflow takes input (range of sizes of storage A and B) from worflow.xml.

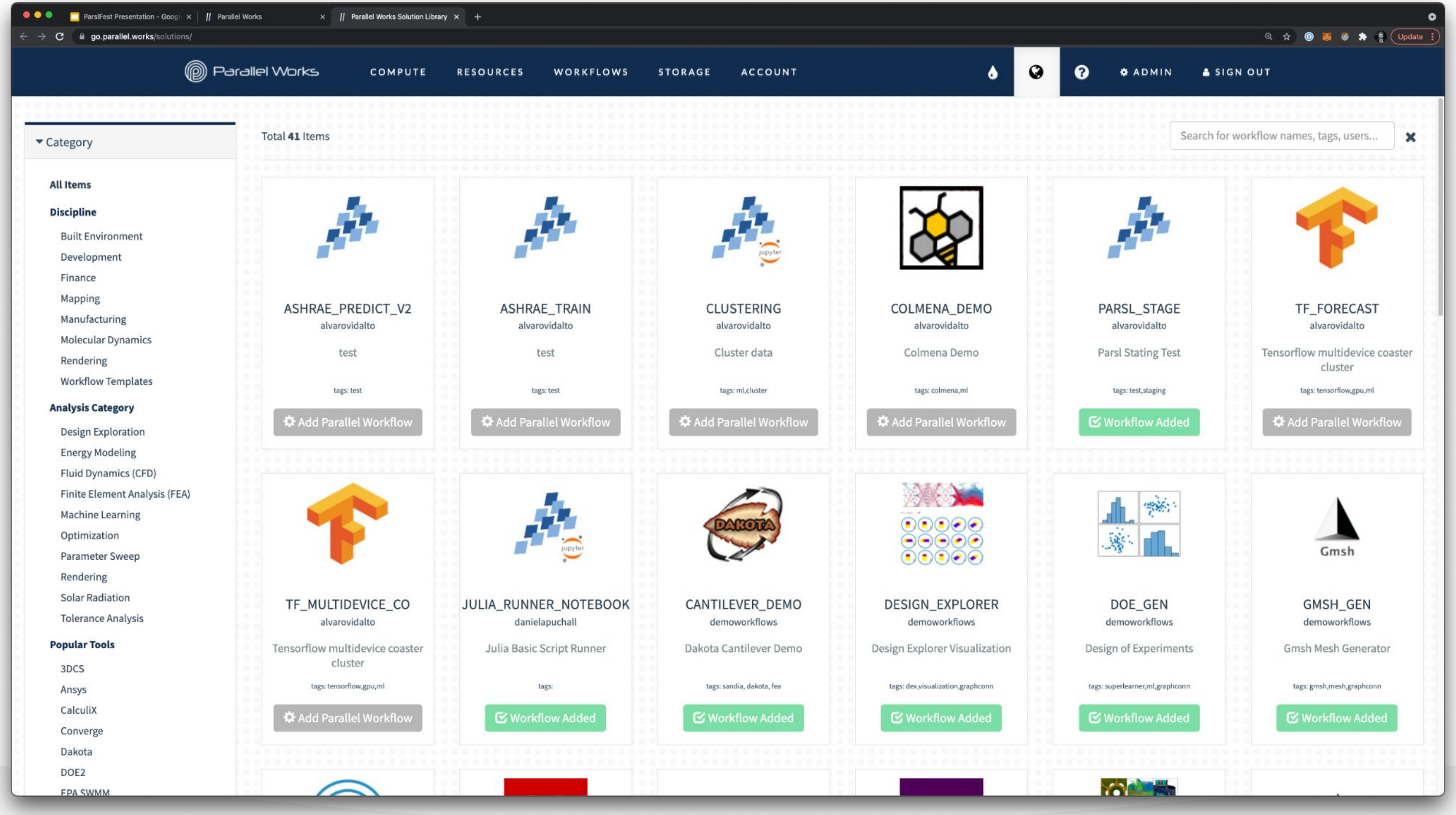
OUTPUT:

The following files are created in the output directory.

Annual operation cost data as text files
Heat map of annual operation over the explored parameter values
Gif animation of plant simulation

- converge
- github
- jobs
- modules
- sample_outputs
- storage
- workflows
  - cantilever_demo
    - utils
    - cantilever.py
    - main.py
    - parslpw.py
    - path.py
    - post.sh
    - template.inputs
    - workflow.xml
  - converge_runner
  - converge_runner_03082021
  - converge_runner_03092021
  - converge_runner_dev
  - design_explorer
  - doe_gen
  - energy_storage_plant_design
  - gmsh_gen
  - gmsh2nek

SINGLE PANE OF GLASS WORKFLOW DISSEMINATION



ParslFest Presentation - Googl | Parallel Works | Parallel Works Solution Library

go.parallel.works/solutions/

Parallel Works | COMPUTE | RESOURCES | WORKFLOWS | STORAGE | ACCOUNT | ADMIN | SIGN OUT

Category

All Items

Discipline
- Built Environment
- Development
- Finance
- Mapping
- Manufacturing
- Molecular Dynamics
- Rendering
- Workflow Templates

Analysis Category
- Design Exploration
- Energy Modeling
- Fluid Dynamics (CFD)
- Finite Element Analysis (FEA)
- Machine Learning
- Optimization
- Parameter Sweep
- Rendering
- Solar Radiation
- Tolerance Analysis

Popular Tools
- 3DCS
- Ansys
- CalculiX
- Converge
- Dakota
- DOE2
- EPA SWMM

Total **41** Items

Search for workflow names, tags, users...

ASHRAE_PREDICT_V2
alvarovidalto
test
tags: test
Add Parallel Workflow

ASHRAE_TRAIN
alvarovidalto
test
tags: test
Add Parallel Workflow

CLUSTERING
alvarovidalto
Cluster data
tags: ml,cluster
Add Parallel Workflow

COLMENA_DEMO
alvarovidalto
Colmena Demo
tags: colmena,ml
Add Parallel Workflow

PARSL_STAGE
alvarovidalto
Parsl Stating Test
tags: test,staging
Workflow Added

TF_FORECAST
alvarovidalto
Tensorflow multidevice coaster cluster
tags: tensorflow,gpu,ml
Add Parallel Workflow

TF_MULTIDEVICE_CO
alvarovidalto
Tensorflow multidevice coaster cluster
tags: tensorflow,gpu,ml
Add Parallel Workflow

JULIA_RUNNER_NOTEBOOK
danielapuchall
Julia Basic Script Runner
tags:
Workflow Added

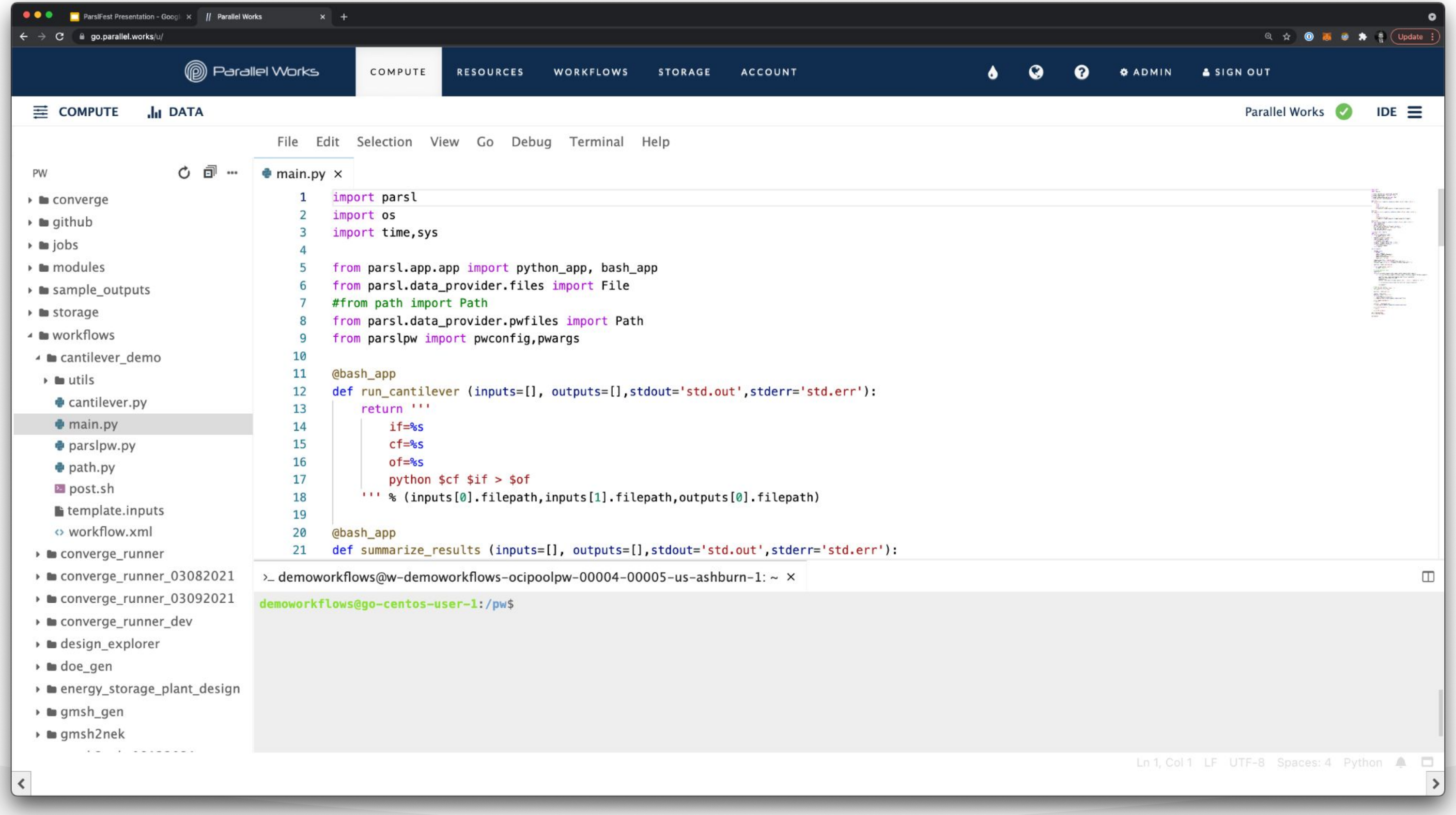
CANTILEVER_DEMO
demoworkflows
Dakota Cantilever Demo
tags: sandia, dakota, fea
Workflow Added

DESIGN_EXPLORER
demoworkflows
Design Explorer Visualization
tags: dex,visualization,graphconn
Workflow Added

DOE_GEN
demoworkflows
Design of Experiments
tags: superlearner,ml,graphconn
Workflow Added

GMSH_GEN
demoworkflows
Gmsh Mesh Generator
tags: gmsh,mesh,graphconn
Workflow Added

SINGLE PANE OF GLASS WORKFLOW DEVELOPMENT

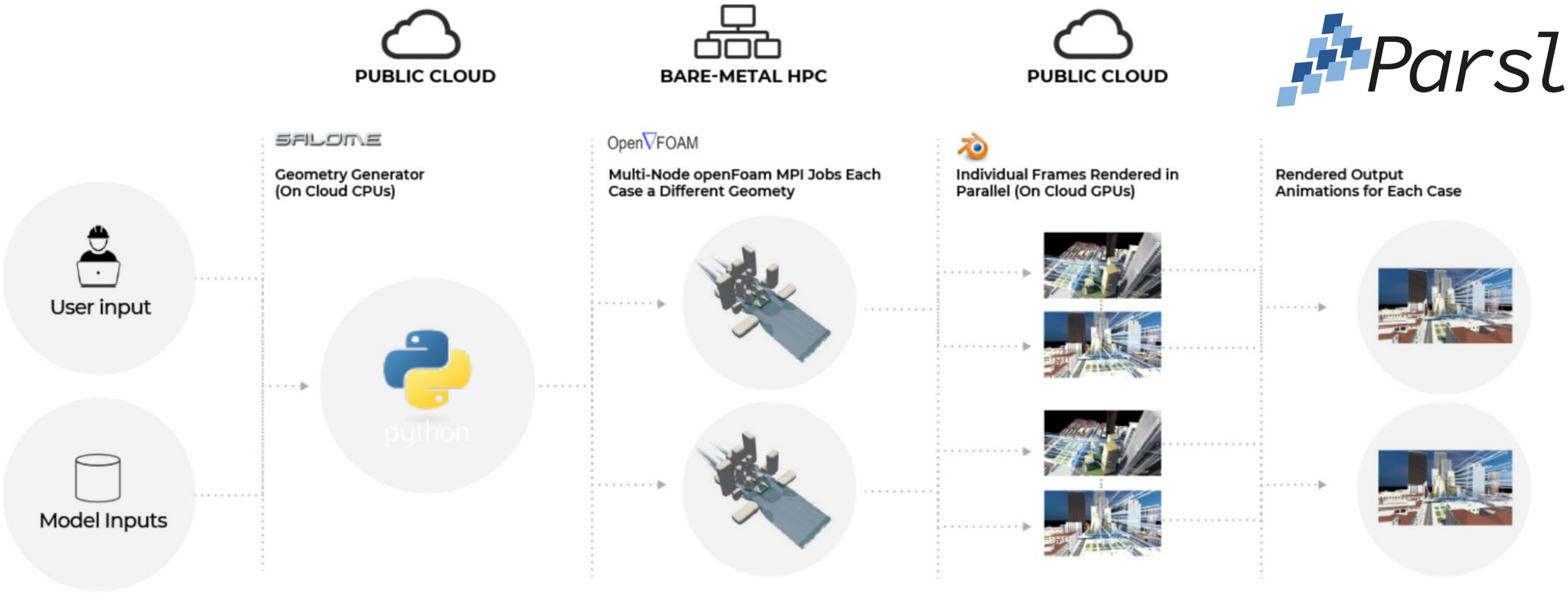


```python
import parsl
import os
import time,sys

from parsl.app.app import python_app, bash_app
from parsl.data_provider.files import File
#from path import Path
from parsl.data_provider.pwfiles import Path
from parslpw import pwconfig,pwargs

@bash_app
def run_cantilever (inputs=[], outputs=[],stdout='std.out',stderr='std.err'):
    return '''
        if=%s
        cf=%s
        of=%s
        python $cf $if > $of
    ''' % (inputs[0].filepath,inputs[1].filepath,outputs[0].filepath)

@bash_app
def summarize_results (inputs=[], outputs=[],stdout='std.out',stderr='std.err'):
```

```
>_ demoworkflows@w-demoworkflows-ocipoolpw-00004-00005-us-ashburn-1: ~
demoworkflows@go-centos-user-1:/pw$
```

LOCATION INDEPENDENT MULTI-SITE WORKFLOW W/ PARSL

Simple automation scripts enable complex workflows to run across multiple Cloud service providers using multiple applications, with automated data movement.



PUBLIC CLOUD

BARE-METAL HPC

PUBLIC CLOUD

Parsl

SALOME

Geometry Generator
(On Cloud CPUs)

OpenFOAM

Multi-Node openFoam MPI Jobs Each
Case a Different Geometry

Individual Frames Rendered in
Parallel (On Cloud GPUs)

Rendered Output
Animations for Each Case

User input

Model Inputs

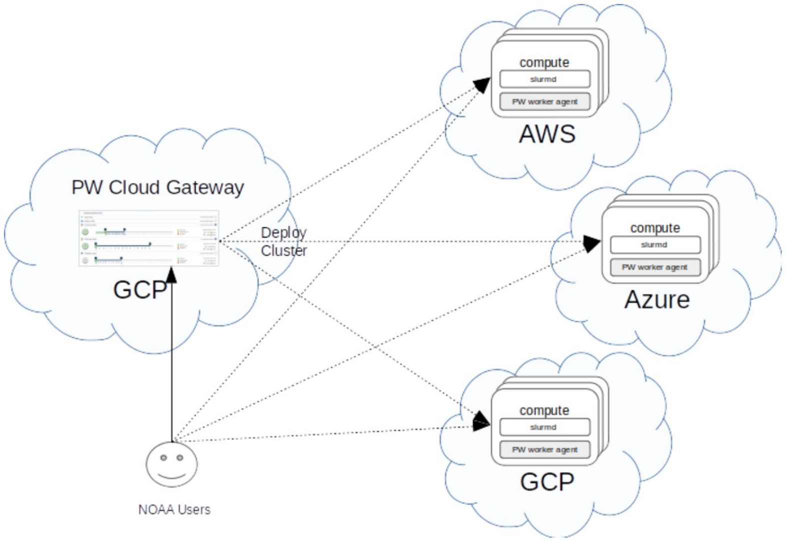
SECTION TWO

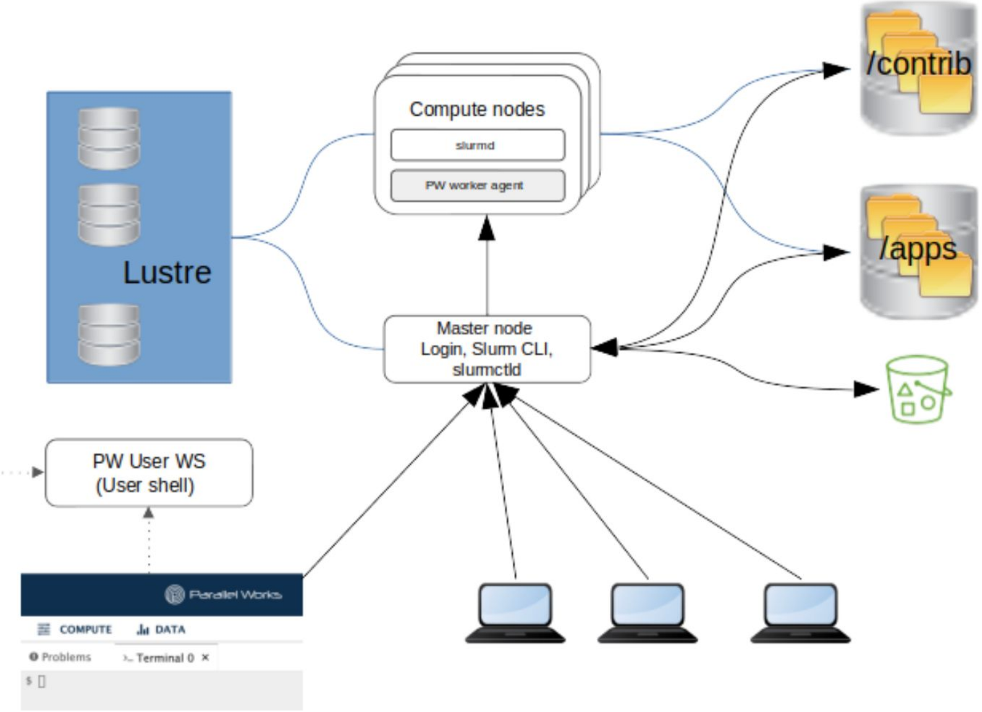
02

PLATFORM UPDATES & ROADMAP

CLUSTERS-IN-THE-CLOUD

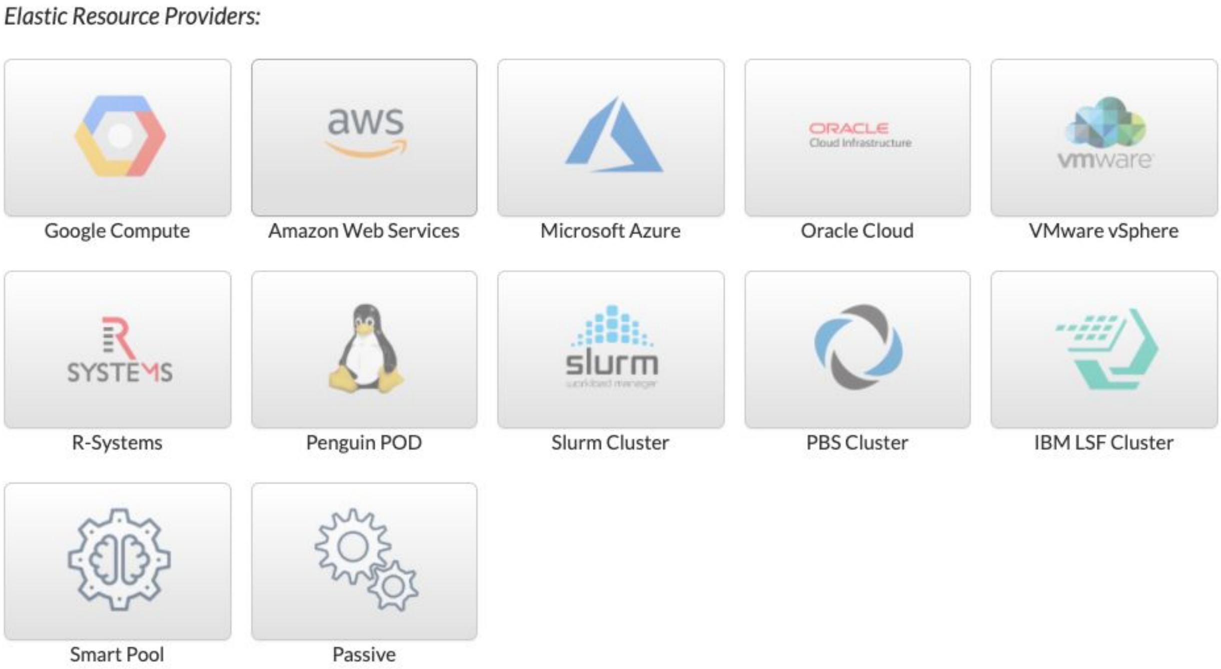
Transient fully provisioned slurm/lustre clusters on AWS, GCP, Azure and OCI.





NEW UNIFORM “ELASTIC” PROVIDERS

Elastic providers running over Swift “Coaster” shell-based service - custom provider integrated into Parsl.



Elastic Resource Providers:

Google Compute

Amazon Web Services

Microsoft Azure

Oracle Cloud

VMware vSphere

R-Systems

Penguin POD

Slurm Cluster

PBS Cluster

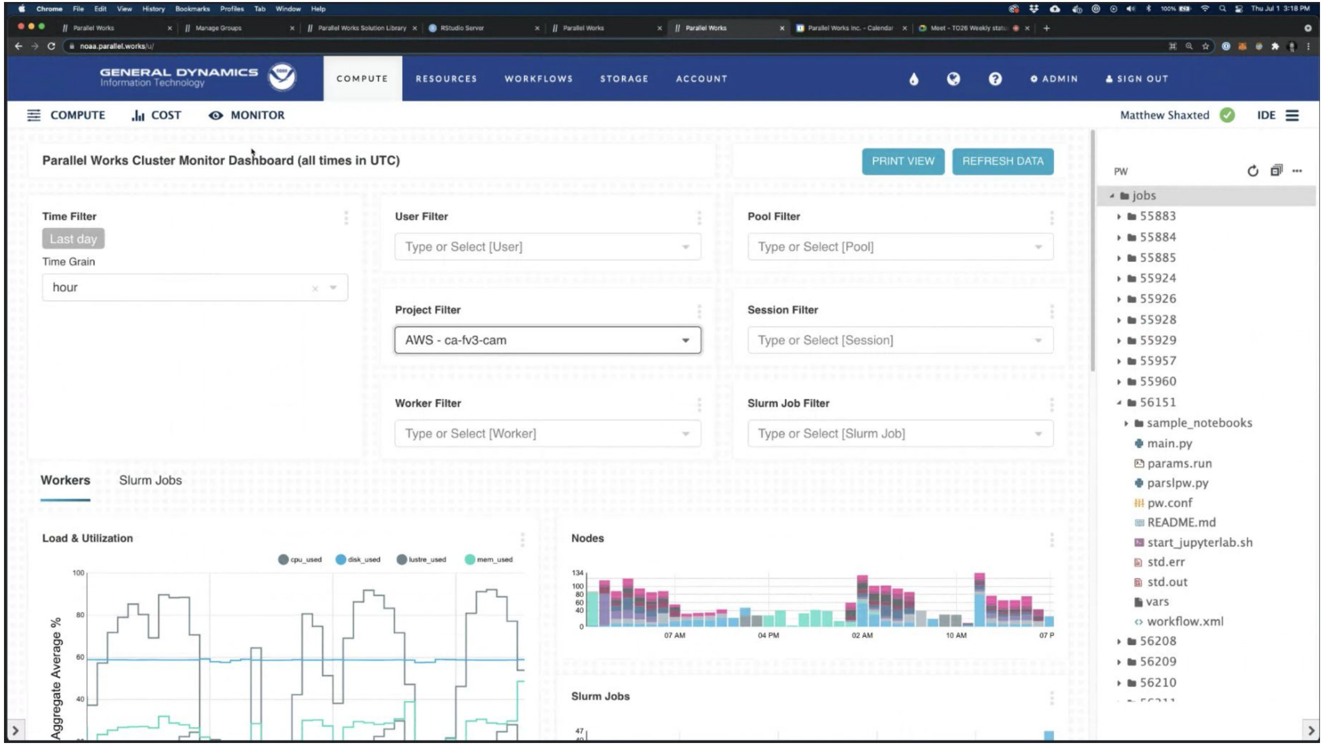
IBM LSF Cluster

Smart Pool

Passive

CLOUD COST & ALLOCATION ENFORCEMENT

Uniform monitoring capability for tracking all costs and node health information across all resource pools and enforcement of project allocations to Parallel Works users.



Parallel Works Cluster Monitor Dashboard (all times in UTC)

Time Filter: Last day

Time Grain: hour

User Filter: Type or Select [User]

Pool Filter: Type or Select [Pool]

Project Filter: AWS - ca-fv3-cam

Session Filter: Type or Select [Session]

Worker Filter: Type or Select [Worker]

Slurm Job Filter: Type or Select [Slurm Job]

TECHNOLOGY ROADMAP

- Policy-Driven Task Placement (e.g. Smart Pools)
- Moving Platform to HA with Kubernetes
- Uniform cluster-in-the-cloud provider w/ terraform
- Exploring use of native parsl for compute providers
- FuncX native integration to the platform
- Supporting Additional Workflow Fabrics (e.g. Dask, Jetstream, Prefect.io, Rocoto)

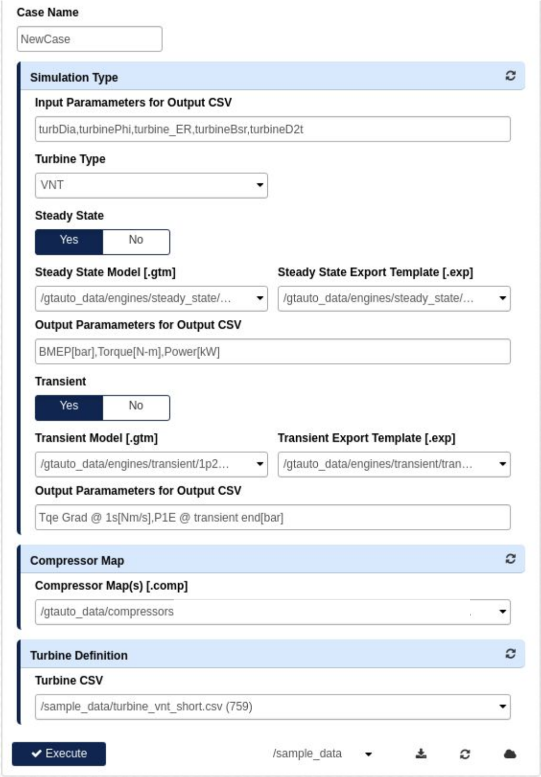
SECTION THREE

03

PARSL IN-THE-WILD

TIER-1 AUTOMOTIVE SUPPLIER

Multi-stage parsl workflow with conditional inputs running a combinatorial sweep of cases on primarily cloud resources connected to on-premise license servers.



Case Name

NewCase

Simulation Type

Input Paramameters for Output CSV

turbDia,turbinePhi,turbine_ER,turbineBsr,turbineD2t

Turbine Type

VNT

Steady State

[✓ Yes] No

Steady State Model [.gtm]

/gtauto_data/engines/steady_state/...

Steady State Export Template [.exp]

/gtauto_data/engines/steady_state/...

Output Paramameters for Output CSV

BMEP[bar],Torque[N-m],Power[kW]

Transient

[✓ Yes] No

Transient Model [.gtm]

/gtauto_data/engines/transient/1p2...

Transient Export Template [.exp]

/gtauto_data/engines/transient/tran...

Output Paramameters for Output CSV

Tqe Grad @ 1s[Nm/s],P1E @ transient end[bar]

Compressor Map

Compressor Map(s) [.comp]

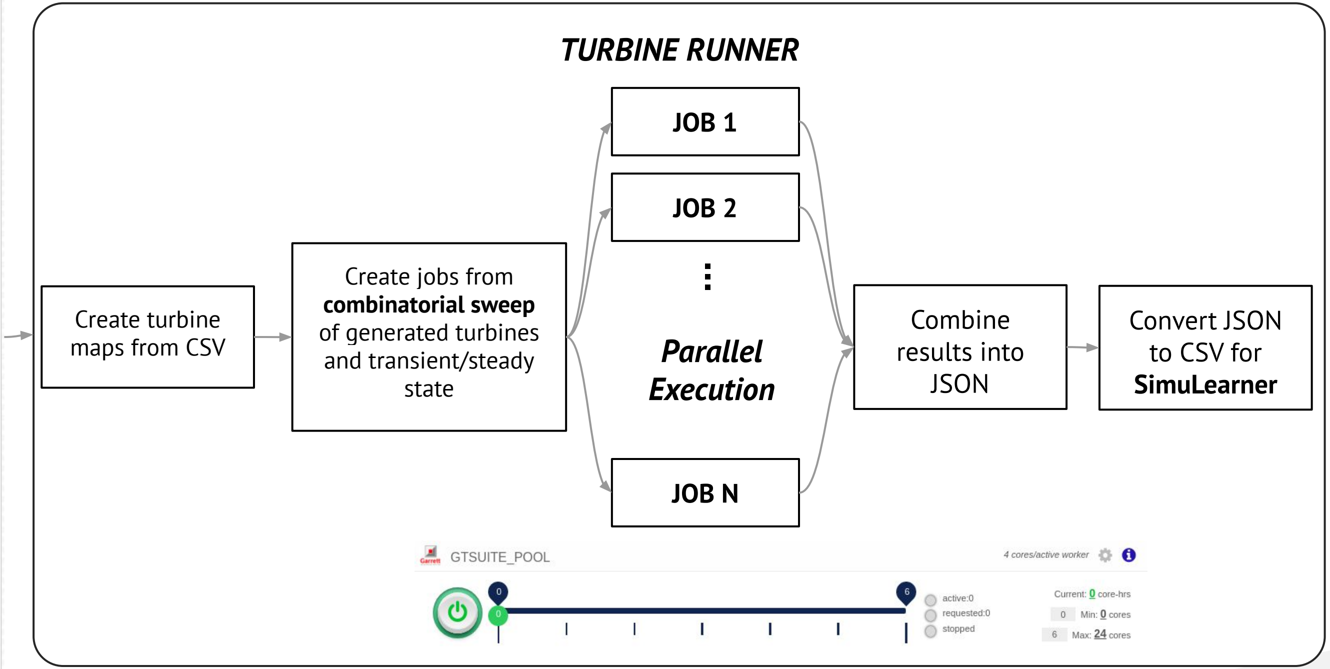
/gtauto_data/compressors

Turbine Definition

Turbine CSV

/sample_data/turbine_vnt_short.csv (759)

✓ Execute   /sample_data



TURBINE RUNNER

Create turbine maps from CSV

Create jobs from **combinatorial sweep** of generated turbines and transient/steady state

JOB 1

JOB 2

⋮

Parallel Execution

JOB N

Combine results into JSON

Convert JSON to CSV for **SimuLearner**

GTSUITE_POOL

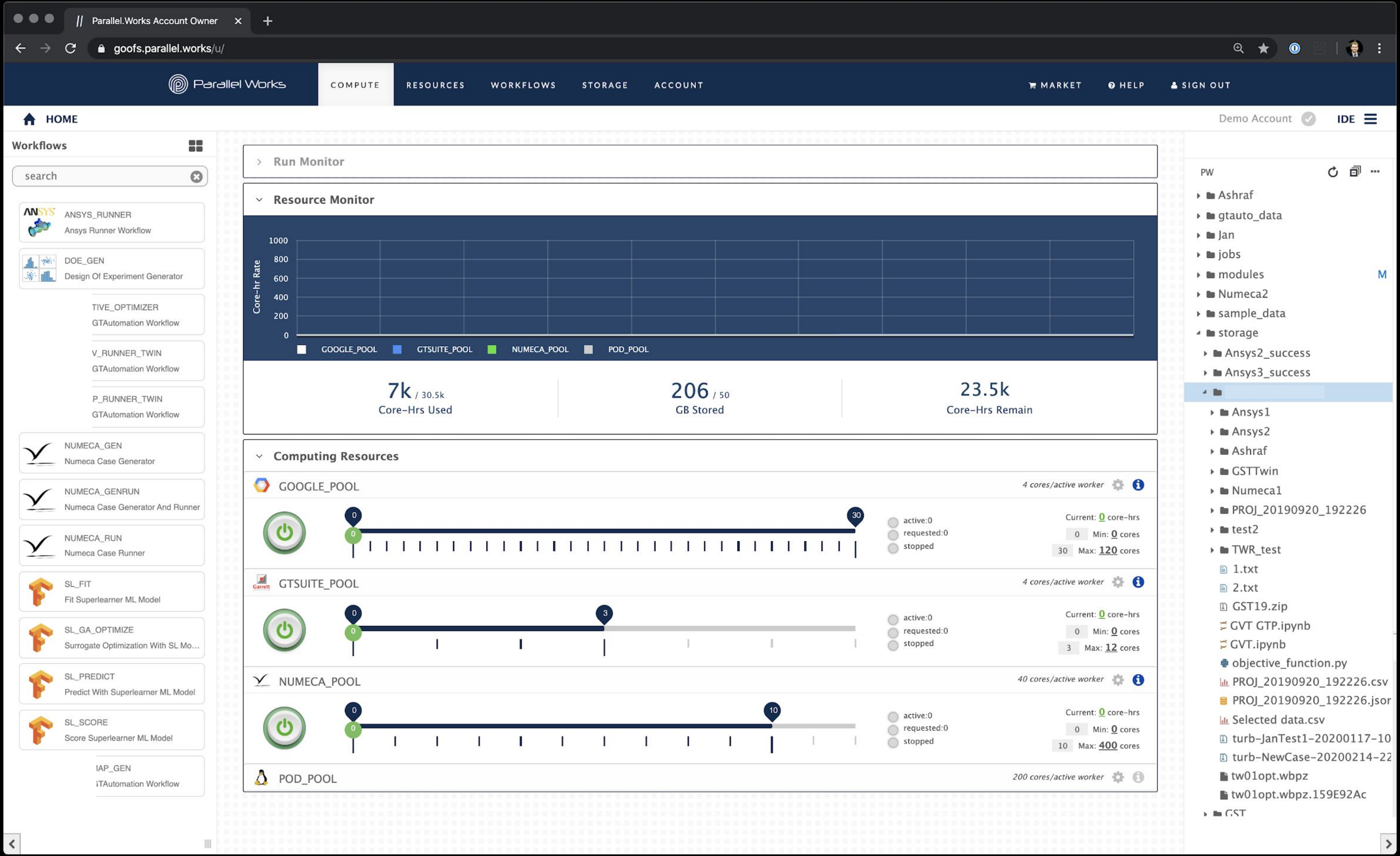
4 cores/active worker

○ active:0
○ requested:0
○ stopped

Current: 0 core-hrs

0 Min: 0 cores

6 Max: 24 cores



Workflows

ANSYS_RUNNER	Ansys Runner Workflow
DOE_GEN	Design Of Experiment Generator
TIVE_OPTIMIZER	GTAutomation Workflow
V_RUNNER_TWIN	GTAutomation Workflow
P_RUNNER_TWIN	GTAutomation Workflow
NUMECA_GEN	Numeca Case Generator
NUMECA_GENRUN	Numeca Case Generator And Runner
NUMECA_RUN	Numeca Case Runner
SL_FIT	Fit Superlearner ML Model
SL_GA_OPTIMIZE	Surrogate Optimization With SL Mo...
SL_PREDICT	Predict With Superlearner ML Model
SL_SCORE	Score Superlearner ML Model
IAP_GEN	;TAutomation Workflow

Run Monitor

Resource Monitor

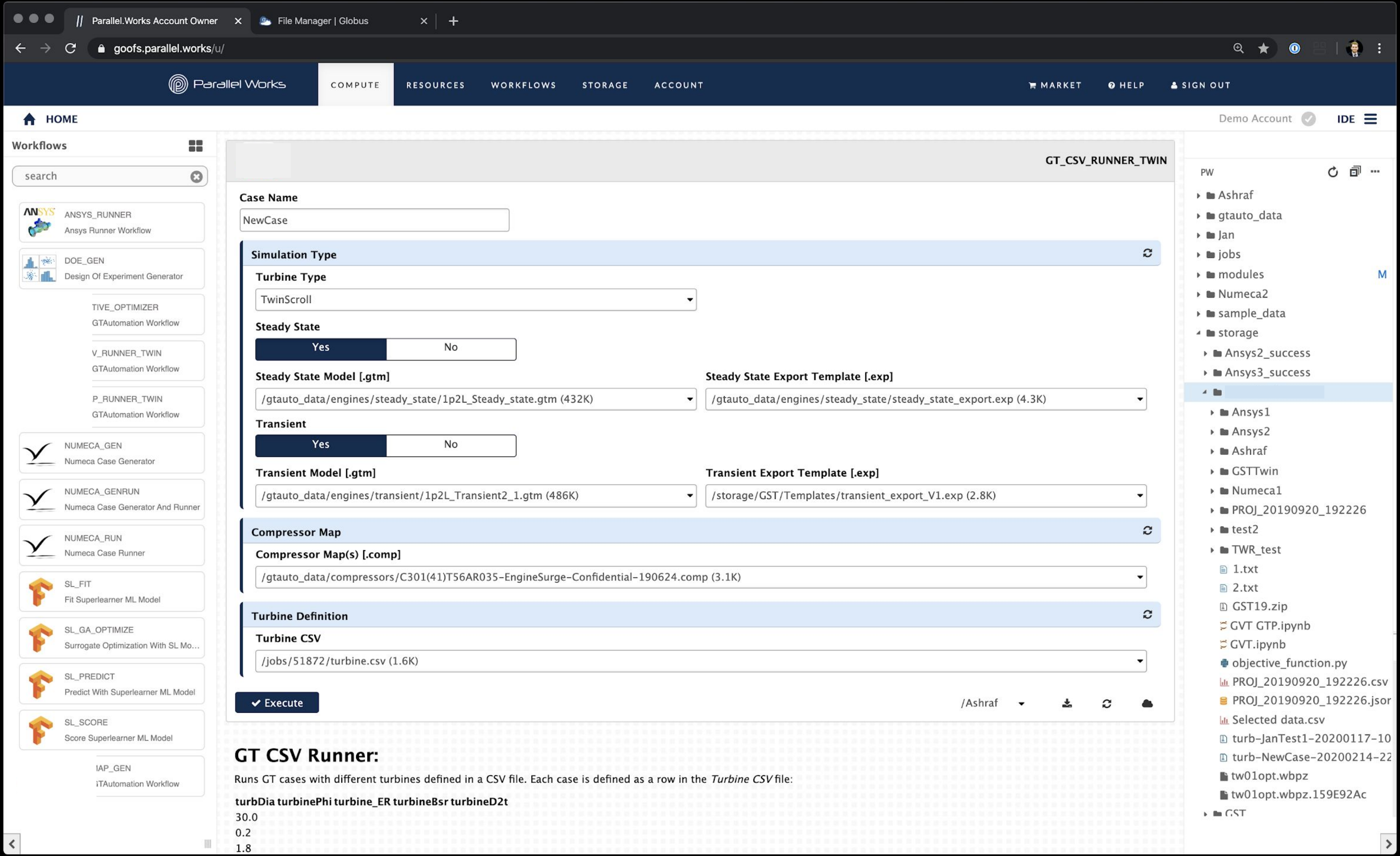
7k / 30.5k	206 / 50	23.5k
Core-Hrs Used	GB Stored	Core-Hrs Remain

Computing Resources

Pool	Workers	Slider	Status	Current	Min	Max
GOOGLE_POOL	4 cores/active worker	0 – 30	active:0 requested:0 stopped	0 core-hrs	0 cores (0)	120 cores (30)
GTSUITE_POOL	4 cores/active worker	0 – 3	active:0 requested:0 stopped	0 core-hrs	0 cores (0)	12 cores (3)
NUMECA_POOL	40 cores/active worker	0 – 10	active:0 requested:0 stopped	0 core-hrs	0 cores (0)	400 cores (10)
POD_POOL	200 cores/active worker					

- Ashraf
- gtauto_data
- Jan
- jobs
- modules
- Numeca2
- sample_data
- storage
  - Ansys2_success
  - Ansys3_success
  -
    - Ansys1
    - Ansys2
    - Ashraf
    - GSTTwin
    - Numeca1
    - PROJ_20190920_192226
    - test2
    - TWR_test
    - 1.txt
    - 2.txt
    - GST19.zip
    - GVT GTP.ipynb
    - GVT.ipynb
    - objective_function.py
    - PROJ_20190920_192226.csv
    - PROJ_20190920_192226.jsor
    - Selected data.csv
    - turb-JanTest1-20200117-10
    - turb-NewCase-20200214-22
    - tw01opt.wbpz
    - tw01opt.wbpz.159E92Ac
  - GST

Account containing multiple private workflows, S3 data stores, and various computing resource pools.



GT_CSV_RUNNER_TWIN

Case Name: NewCase

Simulation Type

Turbine Type: TwinScroll

Steady State: [✓ Yes] No

Steady State Model [.gtm]: /gtauto_data/engines/steady_state/1p2L_Steady_state.gtm (432K)

Steady State Export Template [.exp]: /gtauto_data/engines/steady_state/steady_state_export.exp (4.3K)

Transient: [✓ Yes] No

Transient Model [.gtm]: /gtauto_data/engines/transient/1p2L_Transient2_1.gtm (486K)

Transient Export Template [.exp]: /storage/GST/Templates/transient_export_V1.exp (2.8K)

Compressor Map

Compressor Map(s) [.comp]: /gtauto_data/compressors/C301(41)T56AR035-EngineSurge-Confidential-190624.comp (3.1K)

Turbine Definition

Turbine CSV: /jobs/51872/turbine.csv (1.6K)

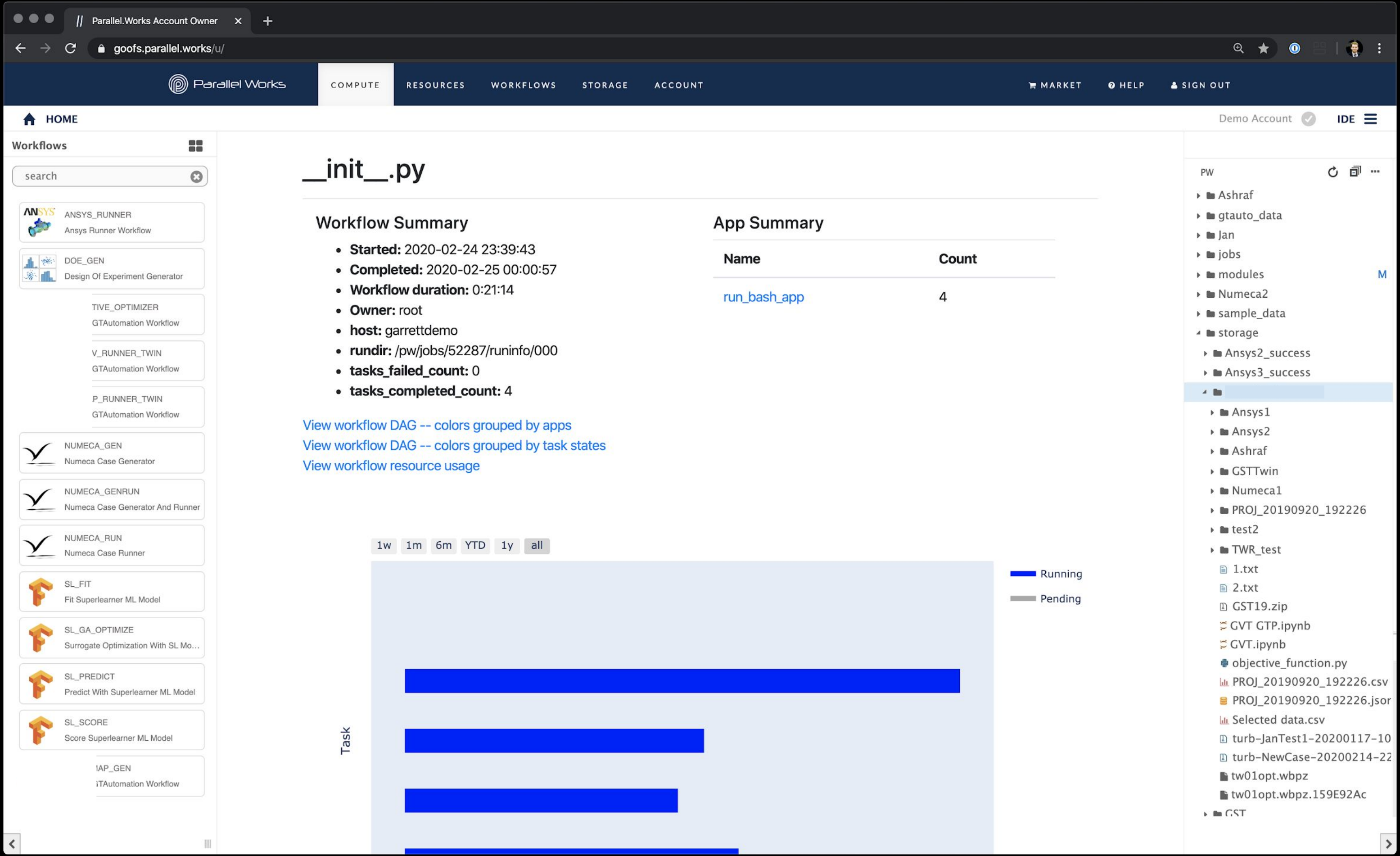
GT CSV Runner:

Runs GT cases with different turbines defined in a CSV file. Each case is defined as a row in the *Turbine CSV* file:

turbDia turbinePhi turbine_ER turbineBsr turbineD2t

30.0
0.2
1.8

GT Automation tool combines turbine generation with GT Suite compressor / turbine map executions.



__init__.py

Workflow Summary

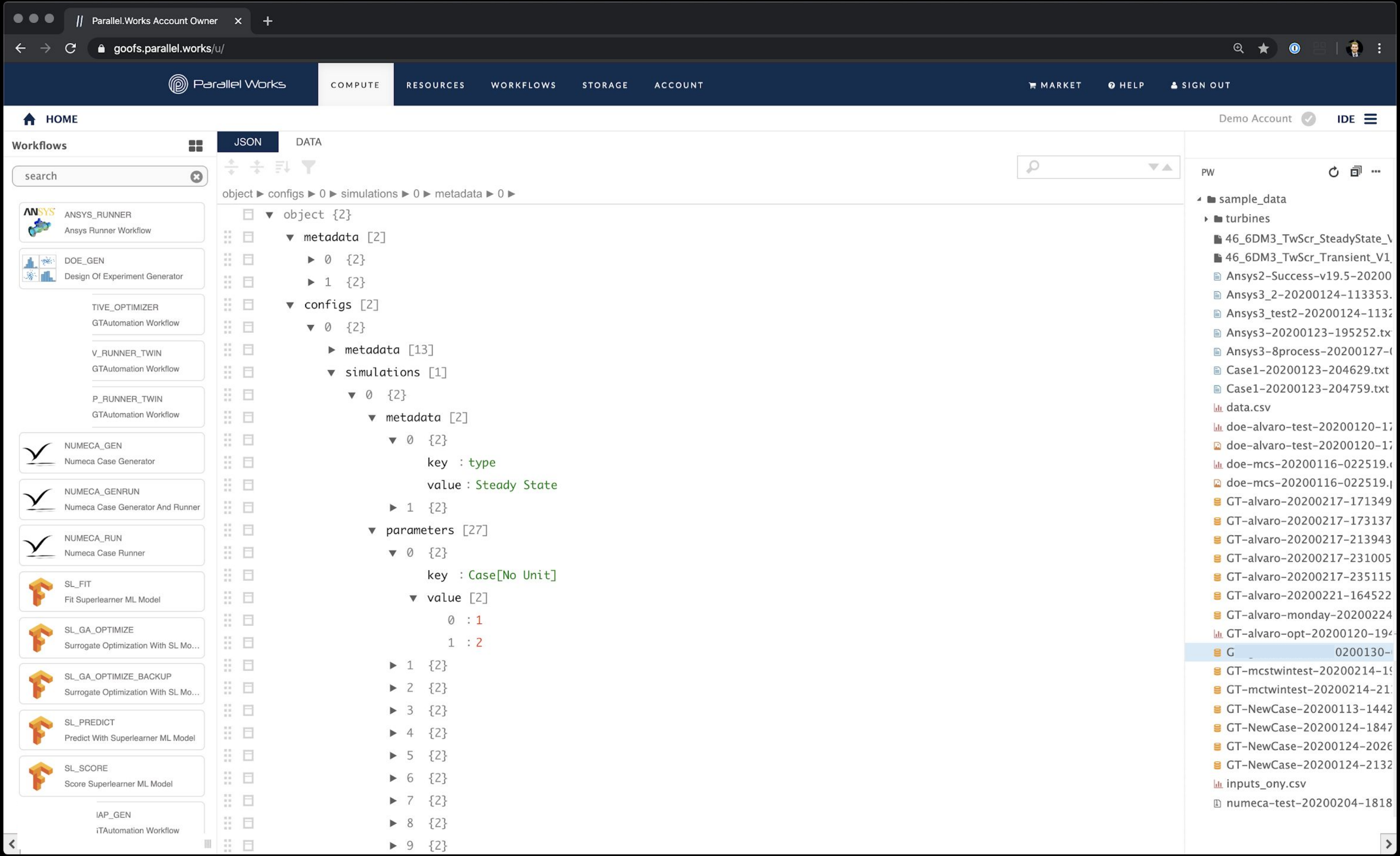
- **Started:** 2020-02-24 23:39:43
- **Completed:** 2020-02-25 00:00:57
- **Workflow duration:** 0:21:14
- **Owner:** root
- **host:** garrettdemo
- **rundir:** /pw/jobs/52287/runinfo/000
- **tasks_failed_count:** 0
- **tasks_completed_count:** 4

View workflow DAG -- colors grouped by apps
View workflow DAG -- colors grouped by task states
View workflow resource usage

App Summary

Name	Count
run_bash_app	4

When a workflow is launched, resources activate elastically and monitor appears to view progress.



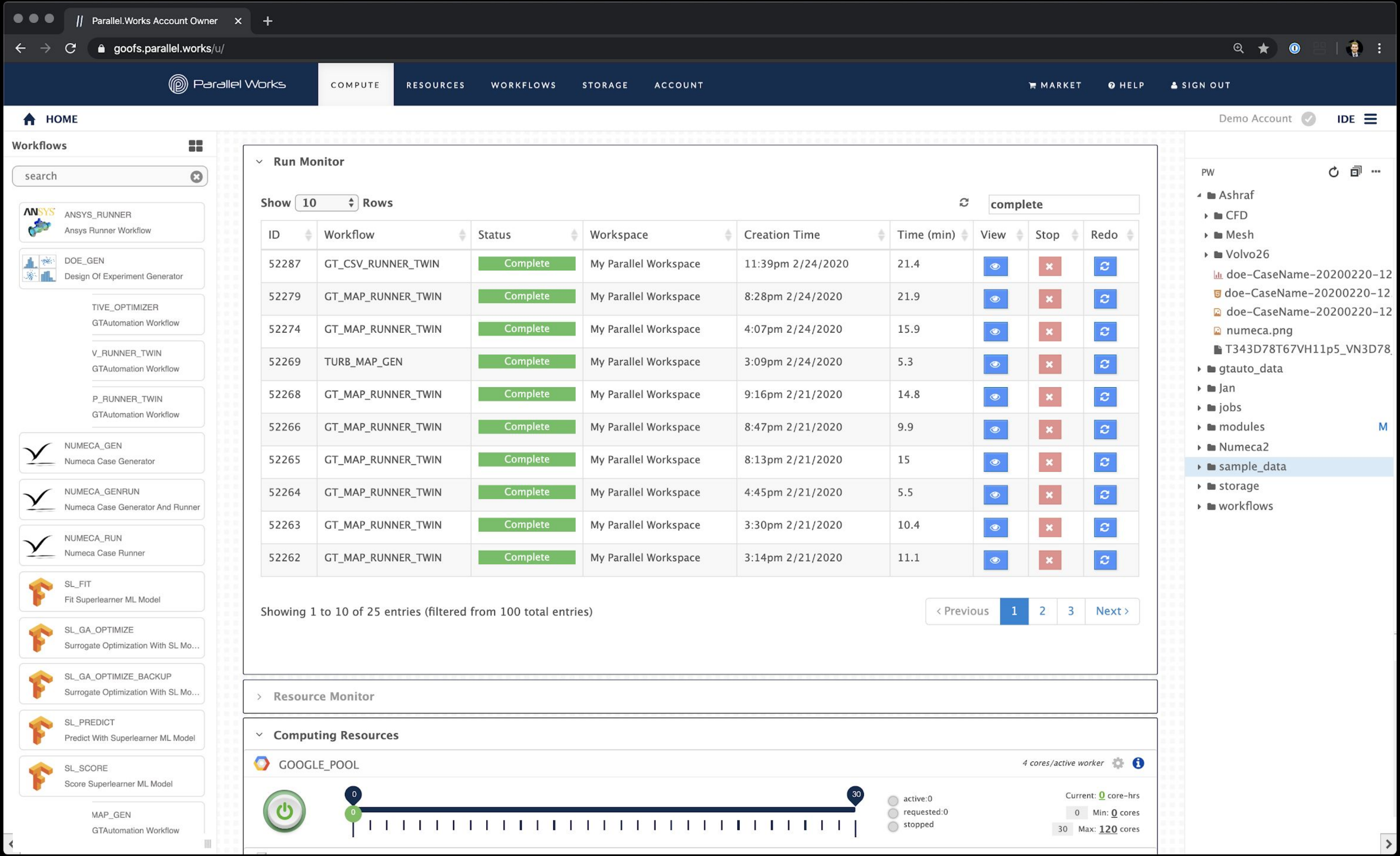
- ANSYS_RUNNER — Ansys Runner Workflow
- DOE_GEN — Design Of Experiment Generator
- TIVE_OPTIMIZER — GTAutomation Workflow
- V_RUNNER_TWIN — GTAutomation Workflow
- P_RUNNER_TWIN — GTAutomation Workflow
- NUMECA_GEN — Numeca Case Generator
- NUMECA_GENRUN — Numeca Case Generator And Runner
- NUMECA_RUN — Numeca Case Runner
- SL_FIT — Fit Superlearner ML Model
- SL_GA_OPTIMIZE — Surrogate Optimization With SL Mo...
- SL_GA_OPTIMIZE_BACKUP — Surrogate Optimization With SL Mo...
- SL_PREDICT — Predict With Superlearner ML Model
- SL_SCORE — Score Superlearner ML Model
- IAP_GEN — iTAutomation Workflow

object ▶ configs ▶ 0 ▶ simulations ▶ 0 ▶ metadata ▶ 0 ▶

```
▼ object {2}
  ▼ metadata [2]
    ▶ 0 {2}
    ▶ 1 {2}
  ▼ configs [2]
    ▼ 0 {2}
      ▶ metadata [13]
      ▼ simulations [1]
        ▼ 0 {2}
          ▼ metadata [2]
            ▼ 0 {2}
                key : type
                value : Steady State
            ▶ 1 {2}
          ▼ parameters [27]
            ▼ 0 {2}
                key : Case[No Unit]
              ▼ value [2]
                  0 : 1
                  1 : 2
            ▶ 1 {2}
            ▶ 2 {2}
            ▶ 3 {2}
            ▶ 4 {2}
            ▶ 5 {2}
            ▶ 6 {2}
            ▶ 7 {2}
            ▶ 8 {2}
            ▶ 9 {2}
```

- sample_data
  - turbines
  - 46_6DM3_TwScr_SteadyState_\
  - 46_6DM3_TwScr_Transient_V1_
  - Ansys2-Success-v19.5-20200
  - Ansys3_2-20200124-113353.
  - Ansys3_test2-20200124-1132
  - Ansys3-20200123-195252.tx
  - Ansys3-8process-20200127-(
  - Case1-20200123-204629.txt
  - Case1-20200123-204759.txt
  - data.csv
  - doe-alvaro-test-20200120-17
  - doe-alvaro-test-20200120-17
  - doe-mcs-20200116-022519.(
  - doe-mcs-20200116-022519.(
  - GT-alvaro-20200217-171349
  - GT-alvaro-20200217-173137
  - GT-alvaro-20200217-213943
  - GT-alvaro-20200217-231005
  - GT-alvaro-20200217-235115
  - GT-alvaro-20200221-164522
  - GT-alvaro-monday-20200224
  - GT-alvaro-opt-20200120-19
  - G _ 0200130-
  - GT-mcstwintest-20200214-1
  - GT-mctwintest-20200214-21
  - GT-NewCase-20200113-1442
  - GT-NewCase-20200124-1847
  - GT-NewCase-20200124-2026
  - GT-NewCase-20200124-2132
  - inputs_ony.csv
  - numeca-test-20200204-1818

Results post-processed directly into JSON and pushed into private S3 data lake bucket.



Workflows

search

- ANSYS_RUNNER — Ansys Runner Workflow
- DOE_GEN — Design Of Experiment Generator
- TIVE_OPTIMIZER — GTAutomation Workflow
- V_RUNNER_TWIN — GTAutomation Workflow
- P_RUNNER_TWIN — GTAutomation Workflow
- NUMECA_GEN — Numeca Case Generator
- NUMECA_GENRUN — Numeca Case Generator And Runner
- NUMECA_RUN — Numeca Case Runner
- SL_FIT — Fit Superlearner ML Model
- SL_GA_OPTIMIZE — Surrogate Optimization With SL Mo...
- SL_GA_OPTIMIZE_BACKUP — Surrogate Optimization With SL Mo...
- SL_PREDICT — Predict With Superlearner ML Model
- SL_SCORE — Score Superlearner ML Model
- MAP_GEN — GTAutomation Workflow

Run Monitor

Show 10 Rows

complete

ID	Workflow	Status	Workspace	Creation Time	Time (min)	View	Stop	Redo
52287	GT_CSV_RUNNER_TWIN	Complete	My Parallel Workspace	11:39pm 2/24/2020	21.4	👁	✕	↻
52279	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	8:28pm 2/24/2020	21.9	👁	✕	↻
52274	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	4:07pm 2/24/2020	15.9	👁	✕	↻
52269	TURB_MAP_GEN	Complete	My Parallel Workspace	3:09pm 2/24/2020	5.3	👁	✕	↻
52268	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	9:16pm 2/21/2020	14.8	👁	✕	↻
52266	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	8:47pm 2/21/2020	9.9	👁	✕	↻
52265	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	8:13pm 2/21/2020	15	👁	✕	↻
52264	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	4:45pm 2/21/2020	5.5	👁	✕	↻
52263	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	3:30pm 2/21/2020	10.4	👁	✕	↻
52262	GT_MAP_RUNNER_TWIN	Complete	My Parallel Workspace	3:14pm 2/21/2020	11.1	👁	✕	↻

Showing 1 to 10 of 25 entries (filtered from 100 total entries)

‹ Previous 1 2 3 Next ›

Resource Monitor

Computing Resources

GOOGLE_POOL

4 cores/active worker

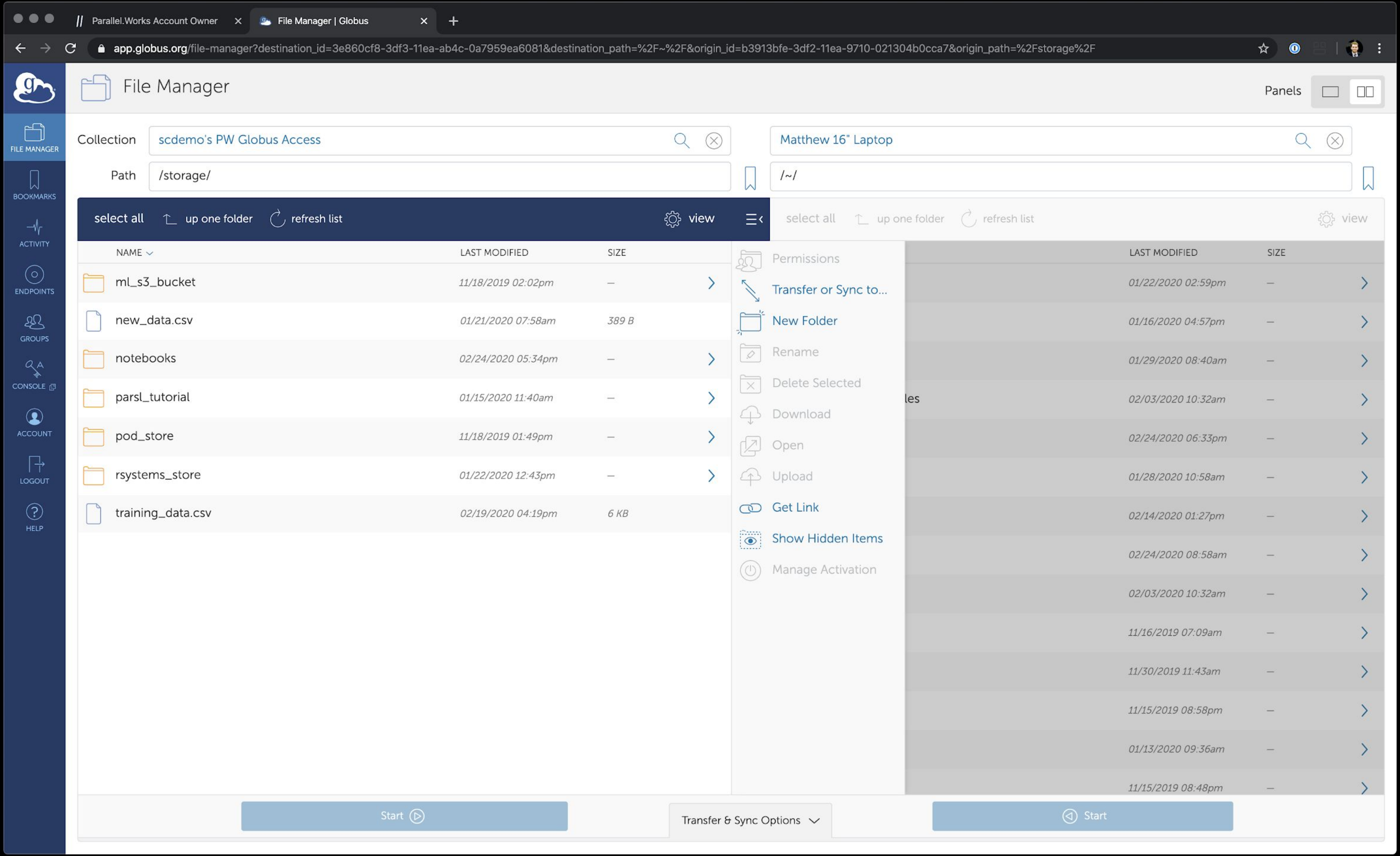
active:0
requested:0
stopped

Current: 0 core-hrs
0 Min: 0 cores
30 Max: 120 cores

PW

- Ashraf
  - CFD
  - Mesh
  - Volvo26
  - doe-CaseName-20200220-12
  - doe-CaseName-20200220-12
  - doe-CaseName-20200220-12
  - numeca.png
  - T343D78T67VH11p5_VN3D78
- gtauto_data
- Jan
- jobs
- modules M
- Numeca2
- sample_data
- storage
- workflows

Access or rerun any previous workflow executions.



app.globus.org/file-manager?destination_id=3e860cf8-3df3-11ea-ab4c-0a7959ea6081&destination_path=%2F~%2F&origin_id=b3913bfe-3df2-11ea-9710-021304b0cca7&origin_path=%2Fstorage%2F

File Manager

Collection: scdemo's PW Globus Access

Path: /storage/

NAME	LAST MODIFIED	SIZE
ml_s3_bucket	11/18/2019 02:02pm	–
new_data.csv	01/21/2020 07:58am	389 B
notebooks	02/24/2020 05:34pm	–
parsl_tutorial	01/15/2020 11:40am	–
pod_store	11/18/2019 01:49pm	–
rsystems_store	01/22/2020 12:43pm	–
training_data.csv	02/19/2020 04:19pm	6 KB

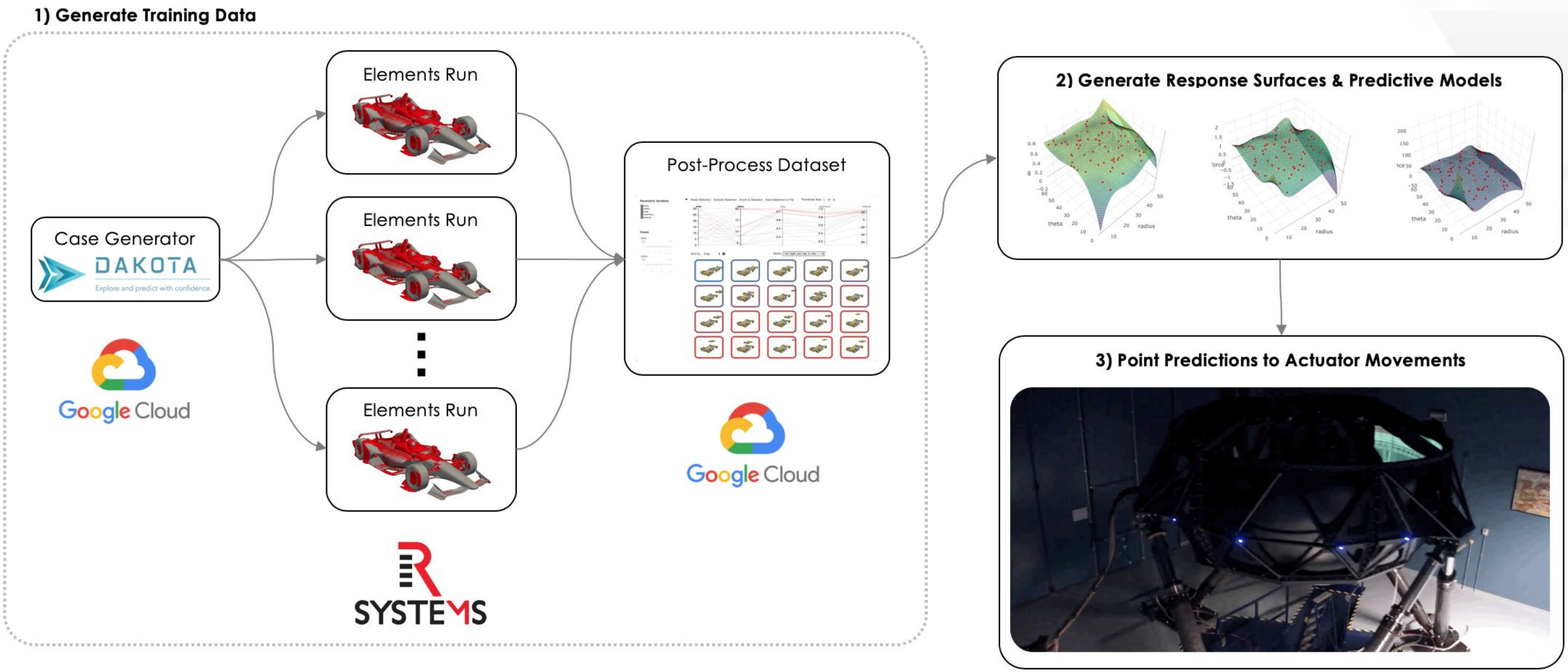
Matthew 16" Laptop

/~/

	LAST MODIFIED	SIZE
	01/22/2020 02:59pm	–
	01/16/2020 04:57pm	–
	01/29/2020 08:40am	–
les	02/03/2020 10:32am	–
	02/24/2020 06:33pm	–
	01/28/2020 10:58am	–
	02/14/2020 01:27pm	–
	02/24/2020 08:58am	–
	02/03/2020 10:32am	–
	11/16/2019 07:09am	–
	11/30/2019 11:43am	–
	11/15/2019 08:58pm	–
	01/13/2020 09:36am	–
	11/15/2019 08:48pm	–

Large datasets and directories synced in Parallel Works user environment with Globus.

AEROKIT DESIGN EXPLORATION



1) Generate Training Data

Case Generator

Elements Run

Elements Run

Elements Run

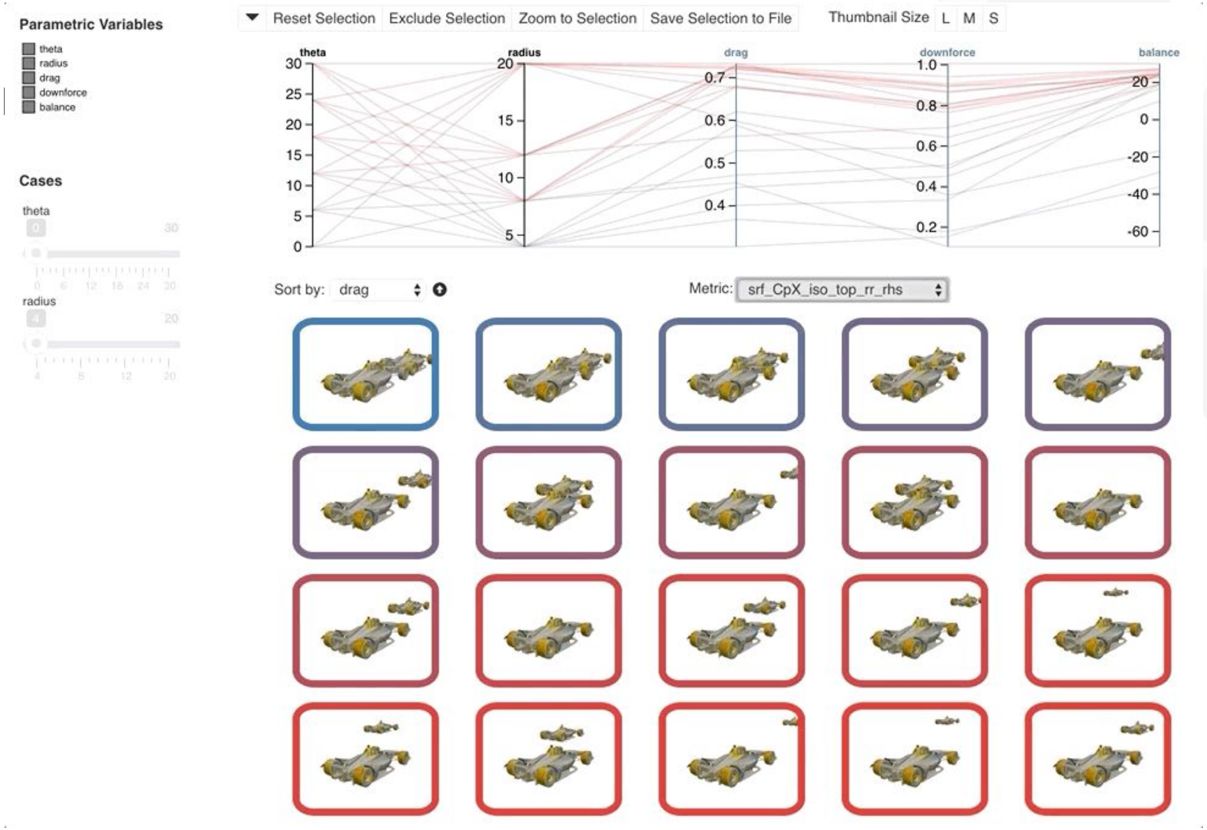
Post-Process Dataset

2) Generate Response Surfaces & Predictive Models

3) Point Predictions to Actuator Movements

Achieving the Vision

1. Generate a parametric model
2. Monte carlo training dataset
3. Generate response surfaces
4. Create predictive model



Sort by: drag

Metric: srf_CpX_iso_top_rr_rhs

- Outputs of parameter sweep put into parametric design explorer (right).
- Allows design engineers to parametrically explore run results
- For example, as the trailing car moves closer into the slipstream, drag decreases.

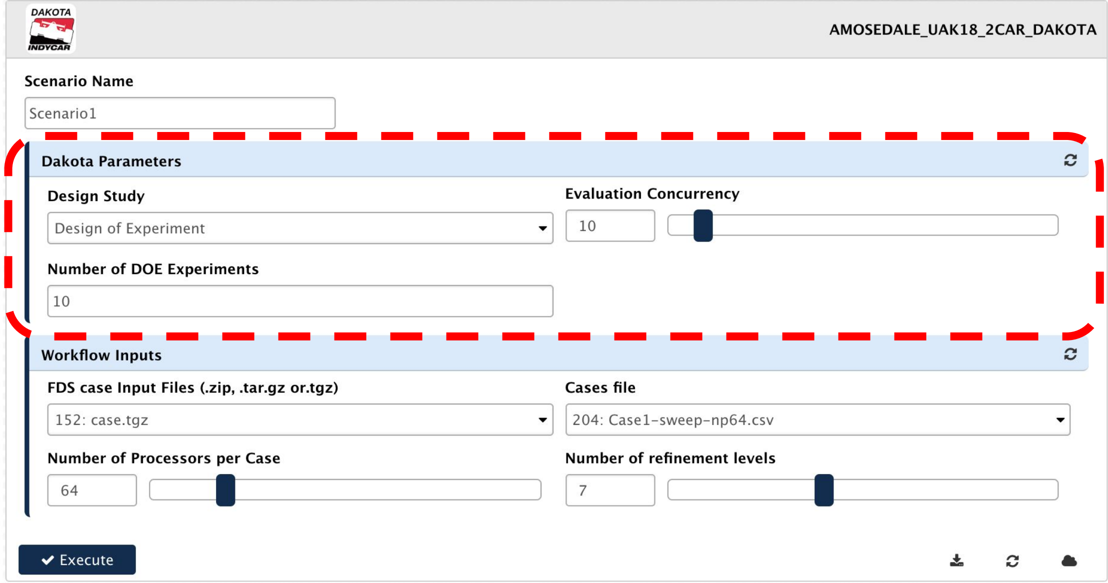
Achieving the Vision

Parallel Works

DAKOTA
Explore and predict with confidence.

1. Generate a parametric model
2. Monte carlo training dataset
3. Generate response surfaces
4. Create predictive model

- Modular approach - parameter sweep workflows plug into Sandia's Dakota design exploration methods.
- Complex Dakota routines easy to run via Parallel Works form execution.



AMOSEDALE_UAK18_2CAR_DAKOTA

Scenario Name: Scenario1

Dakota Parameters

Design Study: Design of Experiment

Evaluation Concurrency: 10

Number of DOE Experiments: 10

Workflow Inputs

FDS case Input Files (.zip, .tar.gz or.tgz): 152: case.tgz

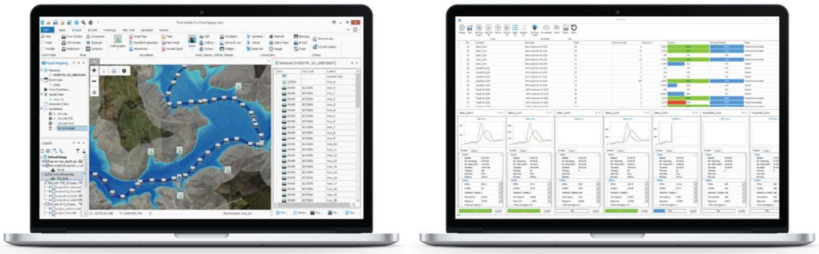
Cases file: 204: Case1-sweep-np64.csv

Number of Processors per Case: 64

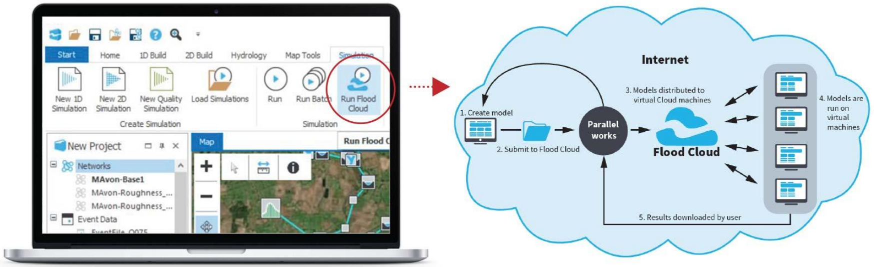
Number of refinement levels: 7

SOFTWARE VENDORS “POWERED BY PARALLEL WORKS”

Software vendors develop Parsl workflows on Parallel Works, and then integrate our REST API into their softwares to get a quick on-ramp to a cloud solution.



JACOBS Flood Cloud
Innovation by Jacobs • Powered by Parallel Works



Internet

1. Create model

2. Submit to Flood Cloud

Parallel works

3. Models distributed to virtual Cloud machines

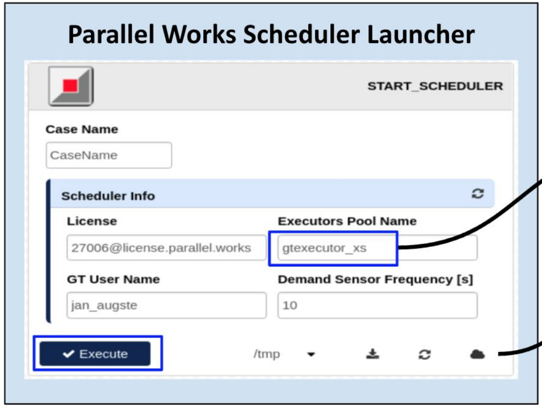
Flood Cloud

4. Models are run on virtual machines

5. Results downloaded by user

Flood Cloud, Powered By The Parallel Works API

GT Gamma Technologies



Parallel Works Scheduler Launcher

START_SCHEDULER

Case Name: CaseName

Scheduler Info

License: 27006@license.parallel.works

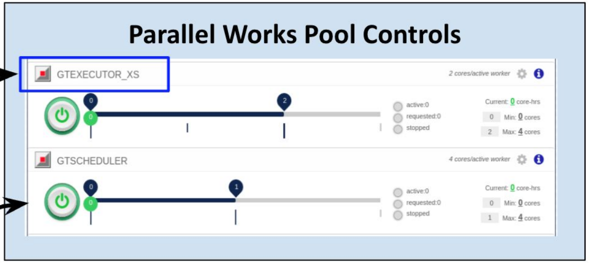
Executors Pool Name: gtexecutor_xs

GT User Name: jan_augste

Demand Sensor Frequency [s]: 10

✔ Execute

/tmp



Parallel Works Pool Controls

GTEXECUTOR_XS — 2 cores/active worker

0 — 2

active:0
requested:0
stopped

Current: 0 core-hrs
0 Min: 0 cores
2 Max: 4 cores

GTSCHEDULER — 4 cores/active worker

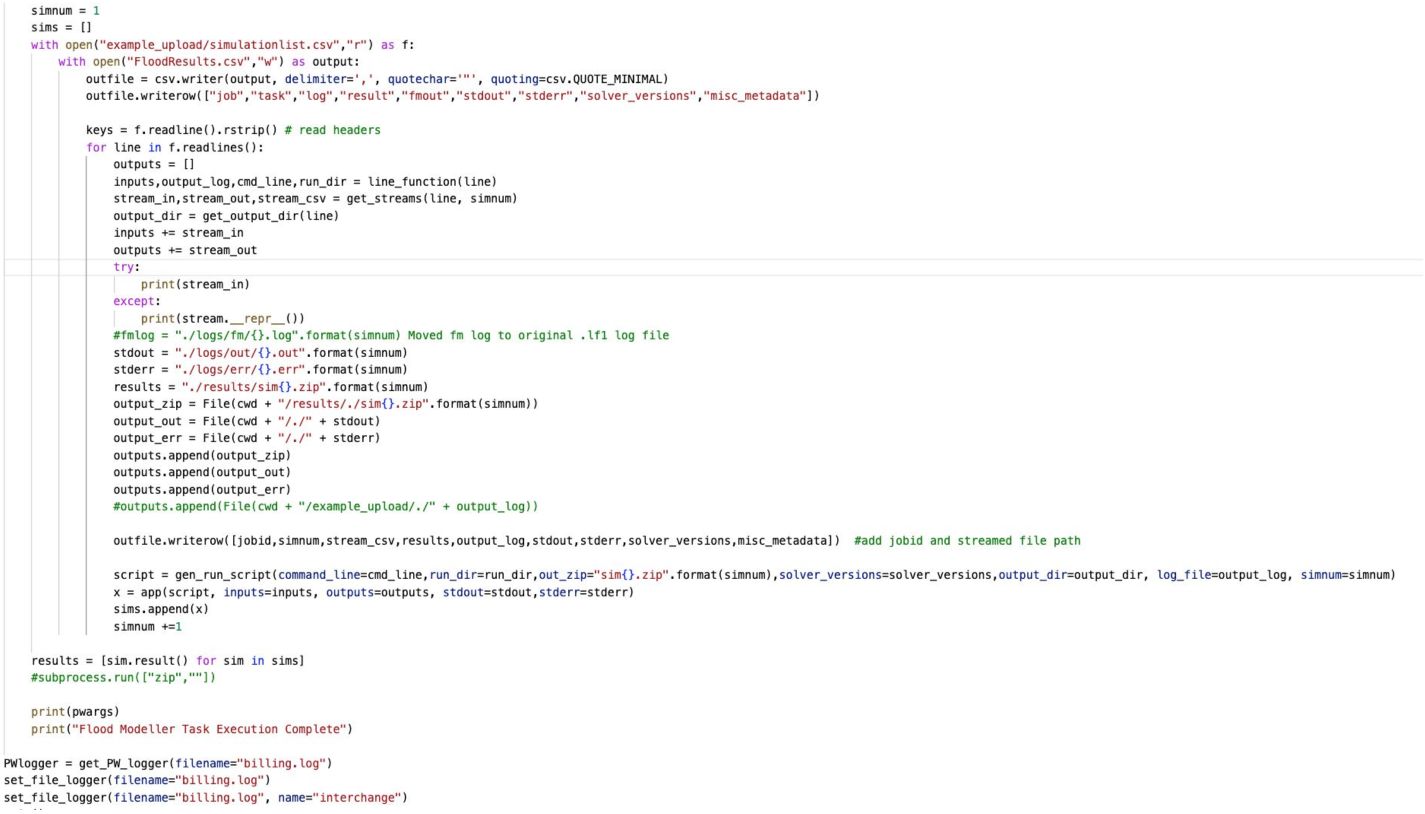
0 — 1

active:0
requested:0
stopped

Current: 0 core-hrs
0 Min: 0 cores
1 Max: 4 cores

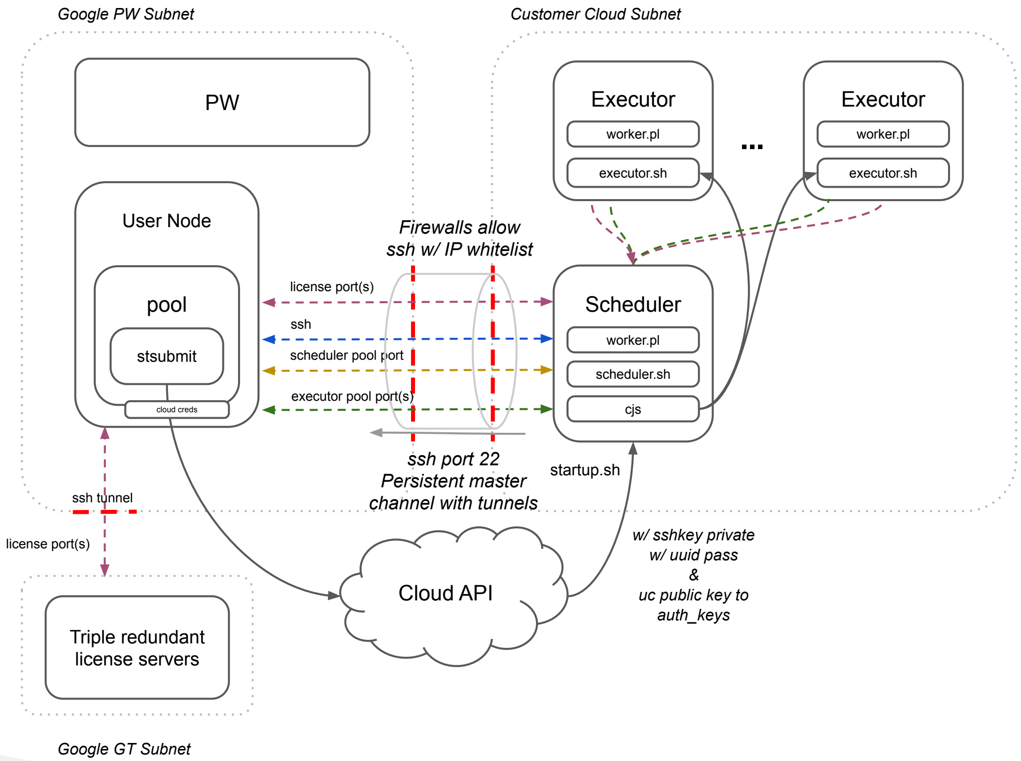
Use the cloud icon to change / select the scheduler’s pool

FLOOD CLOUD WORKFLOW W/ STREAMING & BILLING MANAGER



```
    simnum = 1
    sims = []
    with open("example_upload/simulationlist.csv","r") as f:
        with open("FloodResults.csv","w") as output:
            outfile = csv.writer(output, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
            outfile.writerow(["job","task","log","result","fmout","stdout","stderr","solver_versions","misc_metadata"])

            keys = f.readline().rstrip() # read headers
            for line in f.readlines():
                outputs = []
                inputs,output_log,cmd_line,run_dir = line_function(line)
                stream_in,stream_out,stream_csv = get_streams(line, simnum)
                output_dir = get_output_dir(line)
                inputs += stream_in
                outputs += stream_out
                try:
                    print(stream_in)
                except:
                    print(stream.__repr__())
                #fmlog = "./logs/fm/{}.log".format(simnum) Moved fm log to original .lf1 log file
                stdout = "./logs/out/{}.out".format(simnum)
                stderr = "./logs/err/{}.err".format(simnum)
                results = "./results/sim{}.zip".format(simnum)
                output_zip = File(cwd + "/results/./sim{}.zip".format(simnum))
                output_out = File(cwd + "/./" + stdout)
                output_err = File(cwd + "/./" + stderr)
                outputs.append(output_zip)
                outputs.append(output_out)
                outputs.append(output_err)
                #outputs.append(File(cwd + "/example_upload/./" + output_log))

                outfile.writerow([jobid,simnum,stream_csv,results,output_log,stdout,stderr,solver_versions,misc_metadata])  #add jobid and streamed file path

                script = gen_run_script(command_line=cmd_line,run_dir=run_dir,out_zip="sim{}.zip".format(simnum),solver_versions=solver_versions,output_dir=output_dir, log_file=output_log, simnum=simnum)
                x = app(script, inputs=inputs, outputs=outputs, stdout=stdout,stderr=stderr)
                sims.append(x)
                simnum +=1

    results = [sim.result() for sim in sims]
    #subprocess.run(["zip",""])

    print(pwargs)
    print("Flood Modeller Task Execution Complete")

PWlogger = get_PW_logger(filename="billing.log")
set_file_logger(filename="billing.log")
set_file_logger(filename="billing.log", name="interchange")
```

GT SUITE CLOUD EXECUTION ENGINE



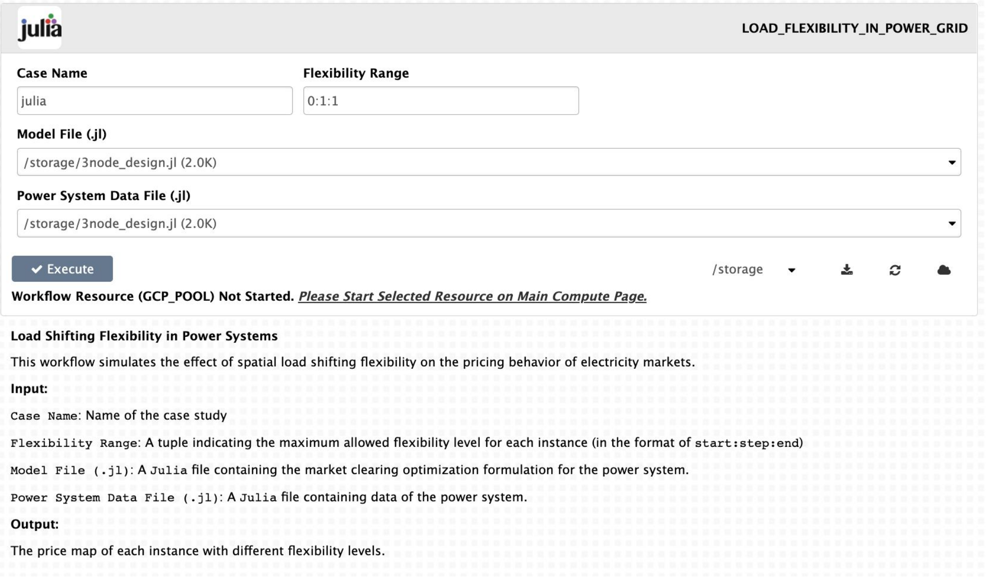
GT Customers submit jobs to the GT scheduler.

The scheduler senses the load and starts executor workers with cog-job-submit (cjs).

The executor script is called on the spawned executor nodes, connecting them to the Scheduler and allowing them to checkout the proper GT licenses (by accessing the license server ports on the Scheduler node).

Executors shutdown automatically when not in use. When scheduler is shutdown, all tunnels are removed.

UNIVERSITY COLLABORATIONS



LOAD_FLEXIBILITY_IN_POWER_GRID

Case Name: julia

Flexibility Range: 0:1:1

Model File (.jl): /storage/3node_design.jl (2.0K)

Power System Data File (.jl): /storage/3node_design.jl (2.0K)

✔ Execute   /storage

Workflow Resource (GCP_POOL) Not Started. ***Please Start Selected Resource on Main Compute Page.***

Load Shifting Flexibility in Power Systems

This workflow simulates the effect of spatial load shifting flexibility on the pricing behavior of electricity markets.

Input:

`Case Name`: Name of the case study

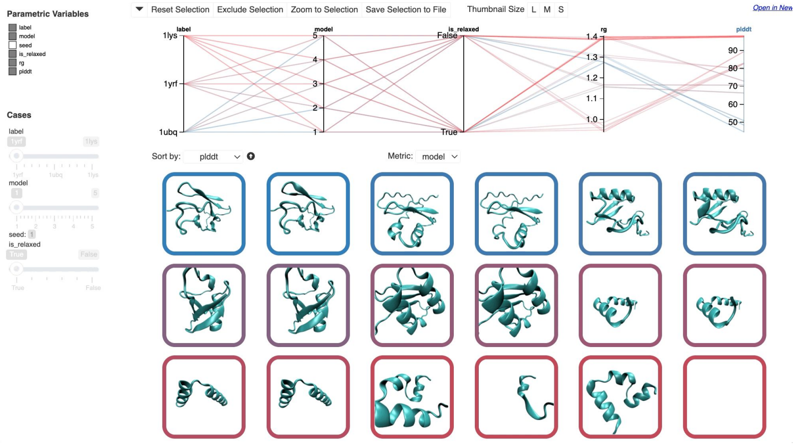
`Flexibility Range`: A tuple indicating the maximum allowed flexibility level for each instance (in the format of `start:step:end`)

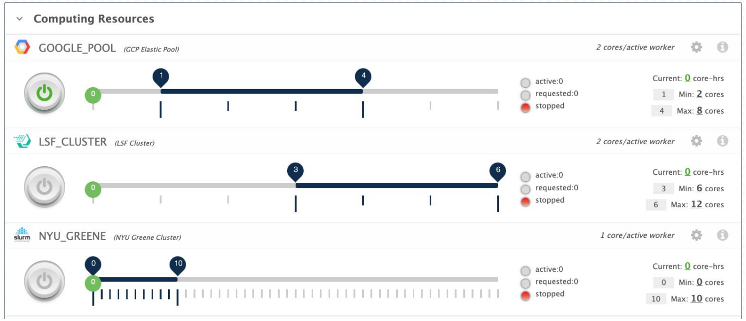
`Model File (.jl)`: A `Julia` file containing the market clearing optimization formulation for the power system.

`Power System Data File (.jl)`: A `Julia` file containing data of the power system.

Output:

The price map of each instance with different flexibility levels.





Computing Resources

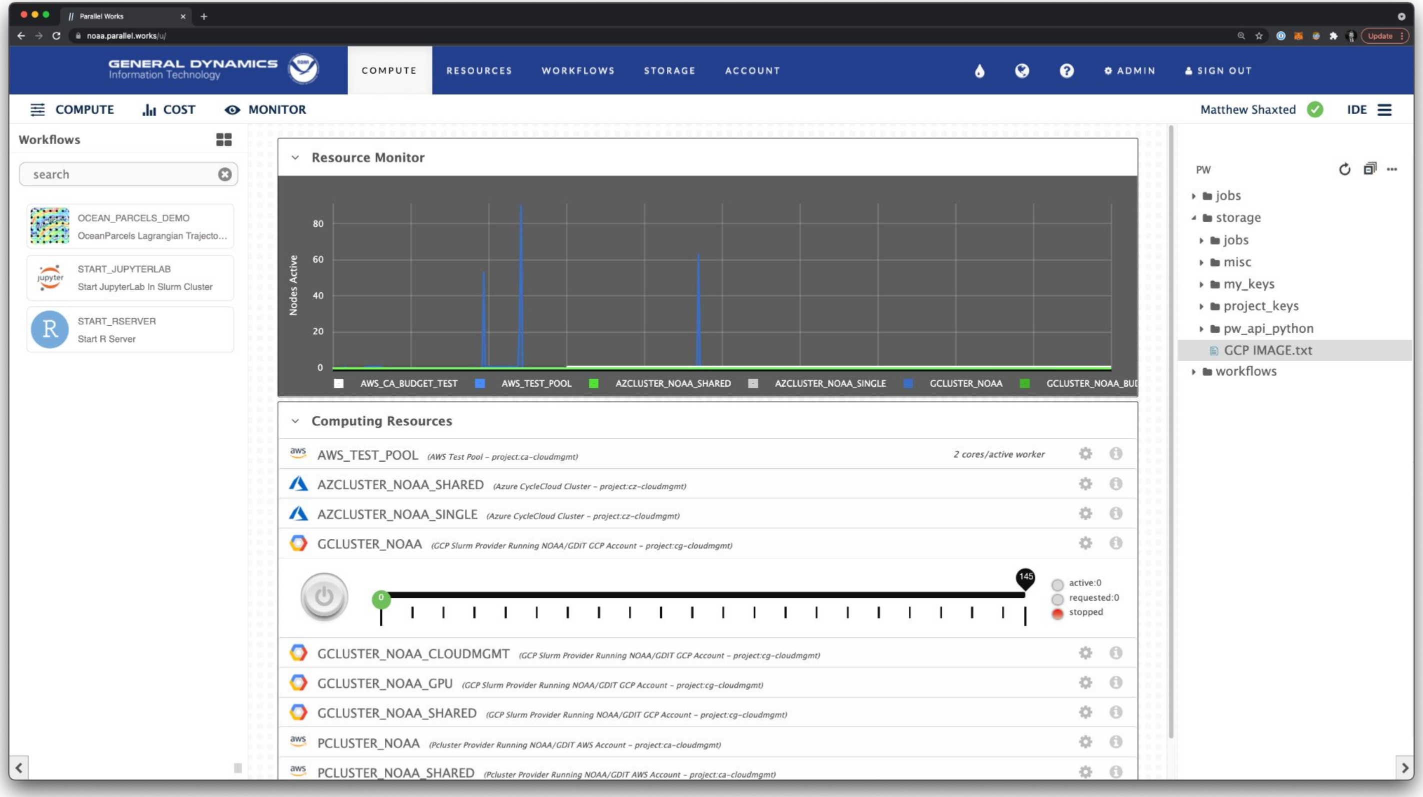
GOOGLE_POOL *(GCP Elastic Pool)* — *2 cores/active worker*
active:0
requested:0
stopped
Current: 0 core–hrs
1 Min: 2 cores
4 Max: 8 cores

LSF_CLUSTER *(LSF Cluster)* — *2 cores/active worker*
active:0
requested:0
stopped
Current: 0 core–hrs
3 Min: 6 cores
6 Max: 12 cores

NYU_GREENE *(NYU Greene Cluster)* — *1 core/active worker*
active:0
requested:0
stopped
Current: 0 core–hrs
0 Min: 0 cores
10 Max: 10 cores

Images courtesy of Hocky Group at NYU and Zavala Group at U Wisc.

NOAA RDHPCS PLATFORM DEPLOYMENT



OCEAN_PARCELS_DEMO
OceanParcels Lagrangian Trajecto...

START_JUPYTERLAB
Start JupyterLab In Slurm Cluster

START_RSERVER
Start R Server

Resource Monitor

Computing Resources

AWS_TEST_POOL *(AWS Test Pool – project:ca-cloudmgmt)* *2 cores/active worker*

AZCLUSTER_NOAA_SHARED *(Azure CycleCloud Cluster – project:cz-cloudmgmt)*

AZCLUSTER_NOAA_SINGLE *(Azure CycleCloud Cluster – project:cz-cloudmgmt)*

GCLUSTER_NOAA *(GCP Slurm Provider Running NOAA/GDIT GCP Account – project:cg-cloudmgmt)*

○ active:0
○ requested:0
● stopped

GCLUSTER_NOAA_CLOUDMGMT *(GCP Slurm Provider Running NOAA/GDIT GCP Account – project:cg-cloudmgmt)*

GCLUSTER_NOAA_GPU *(GCP Slurm Provider Running NOAA/GDIT GCP Account – project:cg-cloudmgmt)*

GCLUSTER_NOAA_SHARED *(GCP Slurm Provider Running NOAA/GDIT GCP Account – project:cg-cloudmgmt)*

PCLUSTER_NOAA *(Pcluster Provider Running NOAA/GDIT AWS Account – project:ca-cloudmgmt)*

PCLUSTER_NOAA_SHARED *(Pcluster Provider Running NOAA/GDIT AWS Account – project:ca-cloudmgmt)*

Parallel Works

INTERESTED IN TRYING PARALLEL WORKS?

Can get demo accounts created on current system, and/or put you on a preview list for upcoming release.

SHAXTED@PARALLELWORKS.COM