Deploying Machine Learning Workflows into HPC environment

HPC-DES

(BEE Team)

Ragini Gupta

08/12/2020

LA-UR-xxxxx

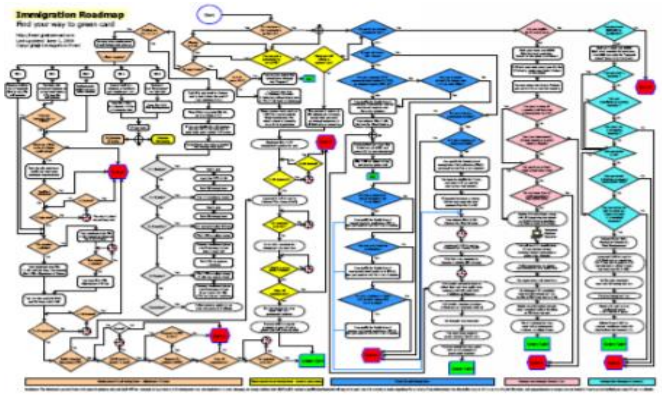
Outline

- Workflows Overview
- Common Workflow Language (CWL)
  – Example of a CWL
- BEE Overview
- Machine Learning Components
- Machine Learning Scientific Workflow using CWL → A new test case for BEE
- Discussion: Benefits and Caveats of current ML workflow
- Conclusion

What is a Workflow

- Workflows: Collection of computer applications, scripts or codes for **computational data analysis and simulations**
  - How applications are configured
  - How data elements flow between them

- Described as chains of interconnected tasks/tools
  - Tools such as Command line tools are programs for processing data



Common Workflow Language (CWL)

- An open-source standard for describing tools and workflows
- Described as :
  - Series of steps which could be a single command line tools
  - Subsequent steps are triggered by an intermediate outcome from preceding steps

- Types of interpreters/workflow management system used for CWL:

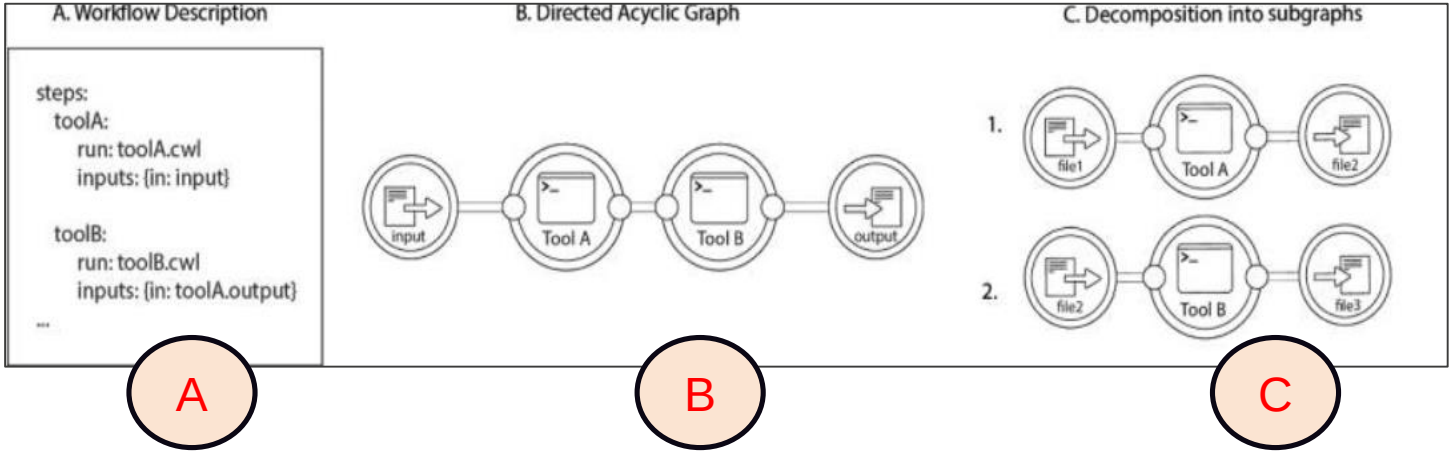
cwl-runner, cwltool, Toil, RABIX workflow engine, LANL-BEE

Optimizing workflow parsing and execution as follows:

A. Interpretation of machine readable workflow description

B. Generation of workflow DAG (Nodes → cmd tools & Edges → flow of data elements, dependencies/rel.)

C. Decomposition into individual jobs (subgraphs) that can be sent to backend for scheduling or execution

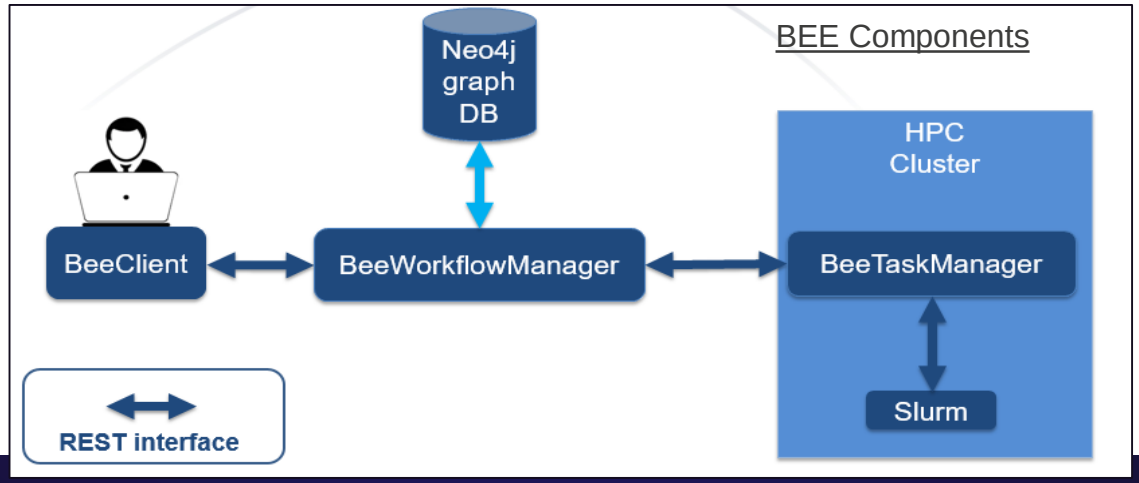


BEE: A Scientific Application Workflow Engine

- Facilitate automatic job execution from workflows, job scheduling based on computing resources, scaling across clusters/nodes/or even continents
- Designed for: **HPC simulations, automations, reproducible/complex**
- CWL serves as an adaptable workflow framework for BEE:
  - Automatic Setup of HPC Requirements
  - Supports the use of HPC Containers, Charliecloud, etc.

A typical cycle of a workflow execution on BEE →

✓ **CWL workflows submitted as sequence of steps (jobs) → BEE Client → Workflow Manager (parses workflow) → Graph Database (generates DAGs, metadata, send ready workflow tasks) → Task Manager (schedules job execution across compute nodes)**



Motivation

- Modern big data analysis involves complex machine learning models and algorithms
- Basic ML methodology runs on a single application, single compiler, fixed number of cores
- Serialized execution of ML models is computationally expensive, scales poorly, time-consuming

- **Problem Statement:** Serialized execution of ML models is computationally expensive, scales poorly, time-consuming for big data and/or data mining applications

- **Proposed solution:** **Using CWL as a bridge for deploying ML models in HPC**

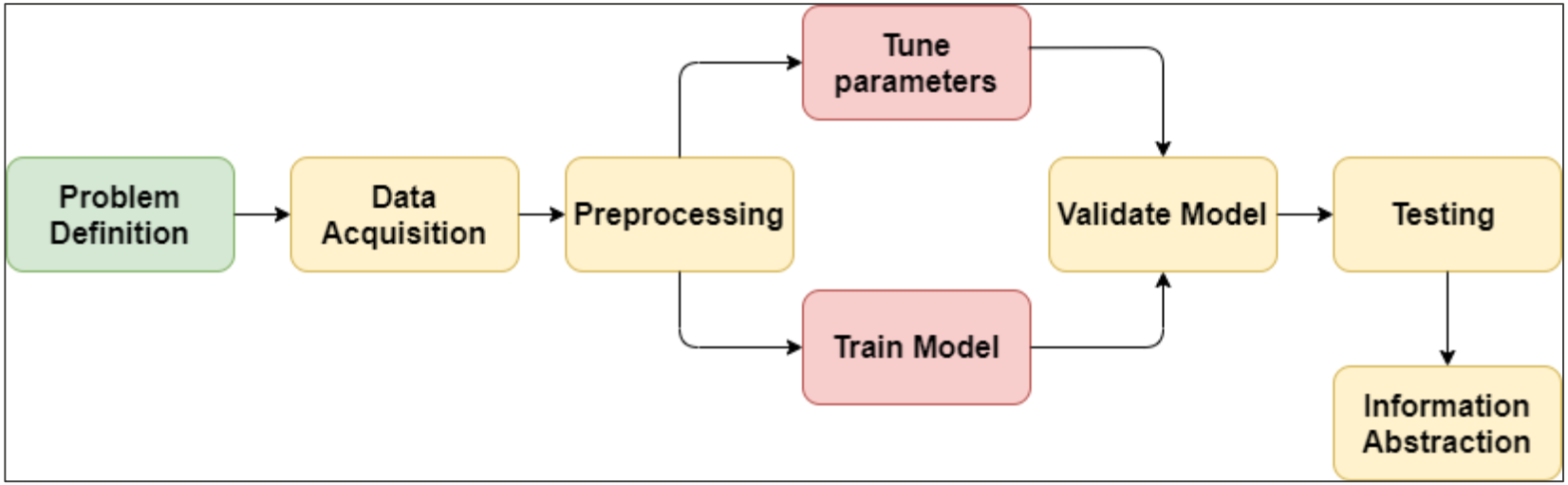
- *To have a reproducible and reusable method to parallelize machine learning models across HPC nodes*
- *Decouple execution of machine learning models running on same dataset*

Scaling BEE with Machine Learning Workflows in CWL

- Using CWL based ML workflow as a relatively new use-case and a proof-of-concept with BEE

- *ML stages can be mapped into ML tasks can be managed, pipelined and orchestrated using adaptable CWL workflow that builds ML model, that work on any type of data or any size*

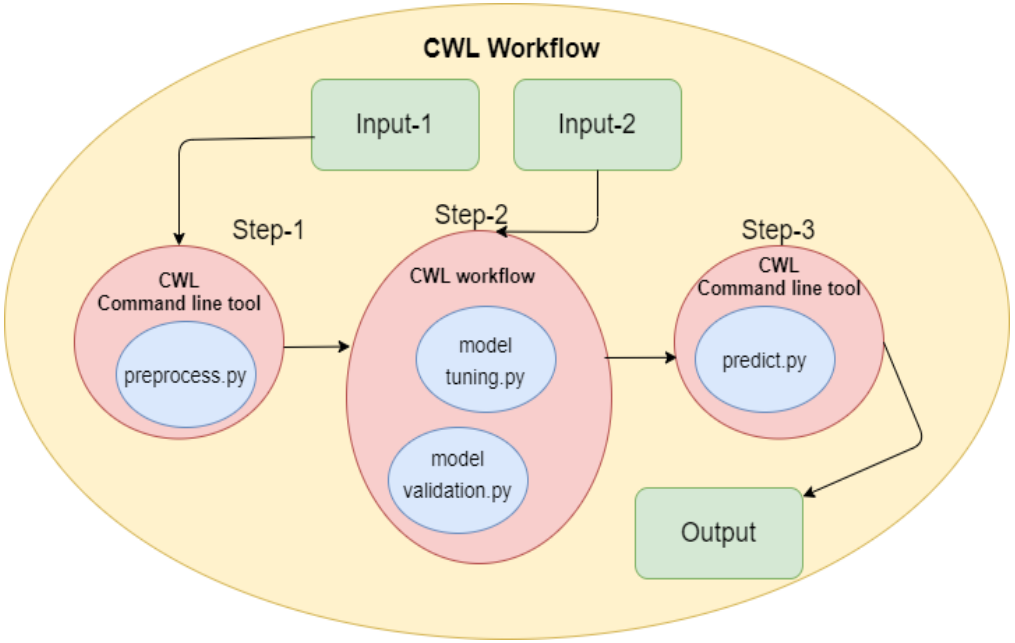
ML Stages Mapped into CWL Workflow Steps



Machine learning Workflow Example

- **Objective:** Prediction on Employee's salary based on interview score, test score, experience
- Workflow is described in four Steps:
  - **Four steps:** Each step corresponds to a machine learning stage and/or machine learning model with it's respective CWL tool description
    - Each step (tool) references to a Python script written to preprocess input dataset, execute ML models, and make prediction
    - Execution order between intertwined steps of a workflow is determined by the connections between steps

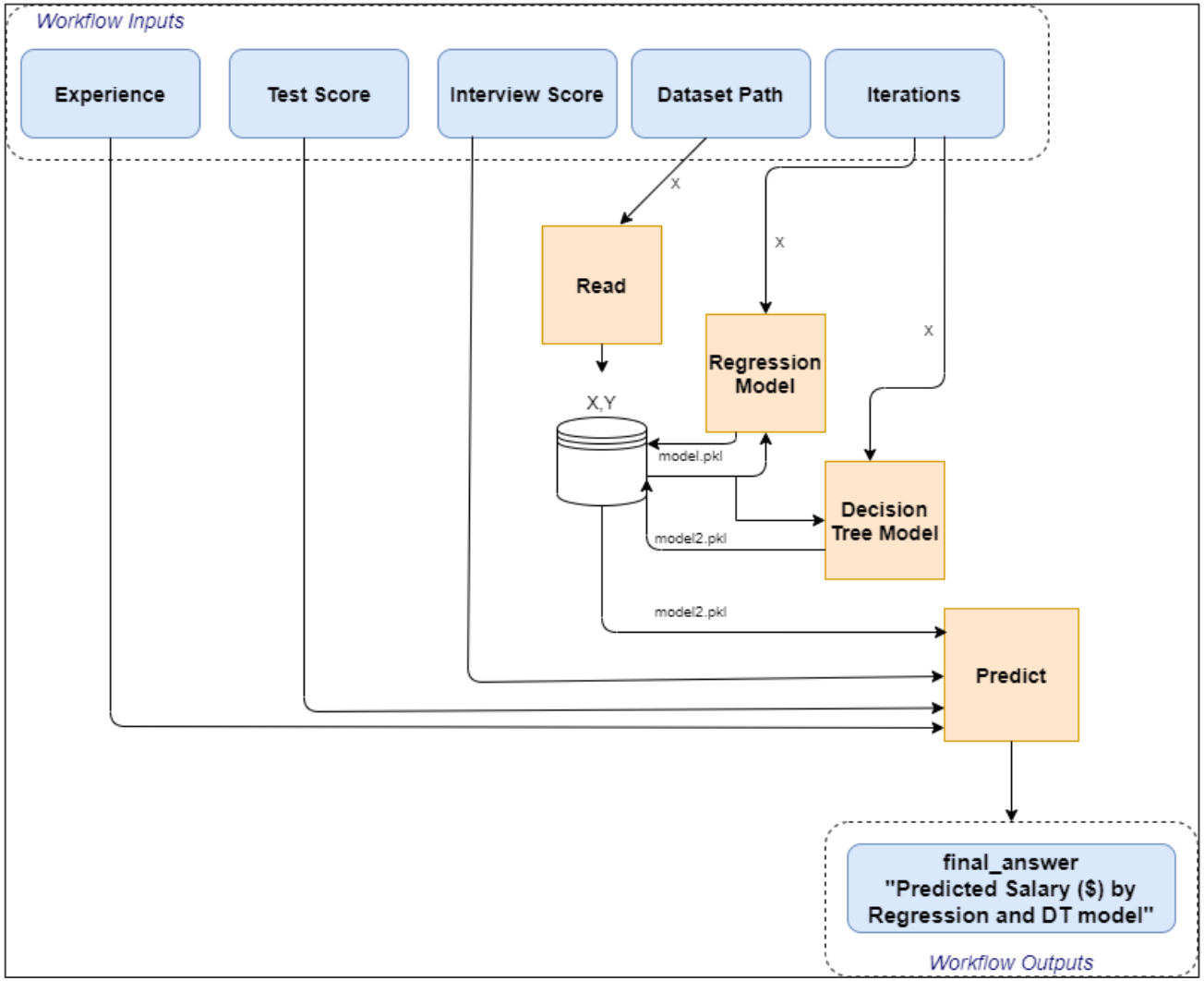
- **Testing Environment:** Fedora 32 VM
- **Interpreter used:** CWL-Runner
- **ML Models used:**
  - Linear Regression and Decision Tree Classifier
  - Python package: Scikit learn



Steps in a ML based CWL

Step-1	Step-2	Step-3	Step-4
Load data and data cleaning	Train the model for Multi-Linear Regression	Train the model for Decision Tree Classifier	Make prediction on a new data instance *(test and information abstraction)*
Inputs: Dataset file path	Inputs: X,Y	Inputs: X,Y	Inputs: X,Y
Preprocess, and export Independent (interview, test, experience scores) and Dependent variables (salary) as X and Y	Finds model parameters (slopes/y intercept)	Finds root and leaf nodes, splitting feature for tree	Apply inputs to the model_1 and model_2
Output: X,Y into the file disk	Output: model_1.pkl	Output: model_2.pkl	Output: JSON object: Expected Salary of Candidate

ML Workflow Graphical overview



Index:

CWL Tools

CWL Workflow I/O elements

ML Workflow DAG and Commands

CWL-Runner Sequence Commands

- cwl-runner -- validate myworkflow.cwl “/home/bee/dataset.csv” -- interviewscore 5 -- testscore 3 -- experience 3 -- iterations 50

- cwl-runner --validate myworkflow.cwl myinput.yaml # OK

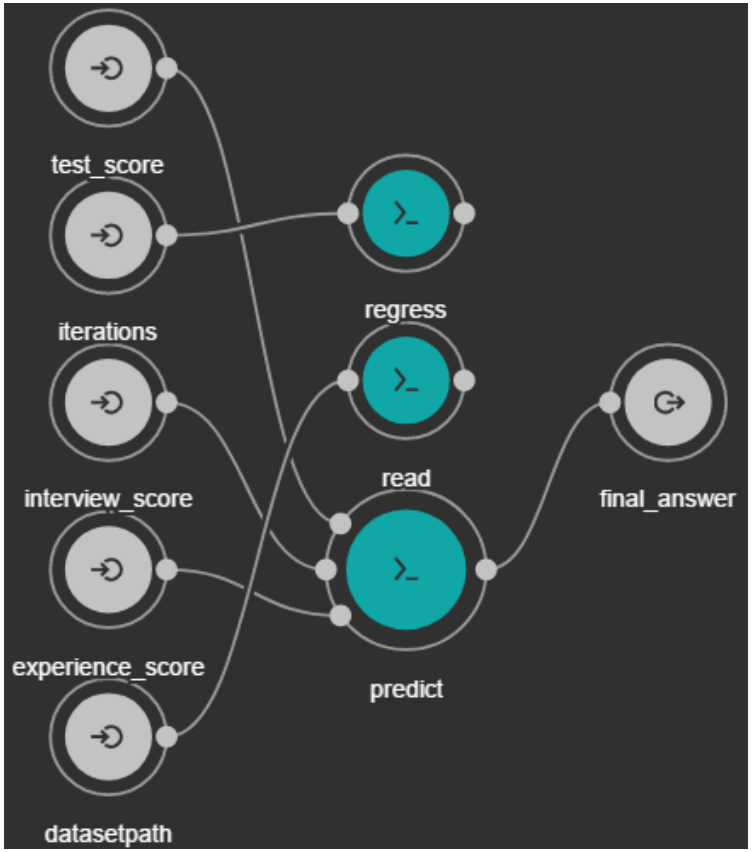
(Note: YAML file specifies input parameters such as file path, number of iterations, in the structure of map)

- cwl-runner --validate main.cwl # OK

- cwl-runner main.cwl main-input.yml # Fails

- cwl-runner --debug main.cwl main-input.yml # Debug ?

ML Workflow DAG



Code Snippet and Output

```
[bee@localhost cwl2]$ cwl-runner machinelearning_pipeline.cwl --experience 5 --interview 4 --test 3 --iterations 1 --datasetpath /home/bee/cwl
2/hiring1.txt
INFO /usr/local/bin/cwl-runner 3.0.20200720165847
INFO Resolved 'machinelearning_pipeline.cwl' to 'file:///home/bee/cwl2/machinelearning_pipeline.cwl'
INFO [workflow ] start
INFO [workflow ] starting step decisiontree
INFO [step decisiontree] start
INFO [job decisiontree] /tmp/rza3wm8y$ python \
    /home/bee/cwl2/finalregress2.py \
    1 > /tmp/rza3wm8y/output3.txt
INFO [job decisiontree] Max memory used: 42MiB
INFO [job decisiontree] completed success
INFO [job regress] Max memory used: 63MiB
INFO [job regress] completed success
INFO [step regress] completed success
INFO [workflow ] completed success
{
    "final_answer": {
        "location": "file:///home/bee/cwl2/expectedValue.txt",
        "basename": "expectedValue.txt",
        "class": "File",
        "checksum": "sha1$2a0cb4114edffb82c107f157854caa635843fa46",
        "size": 198,
        "path": "/home/bee/cwl2/expectedValue.txt"
    }
}
```

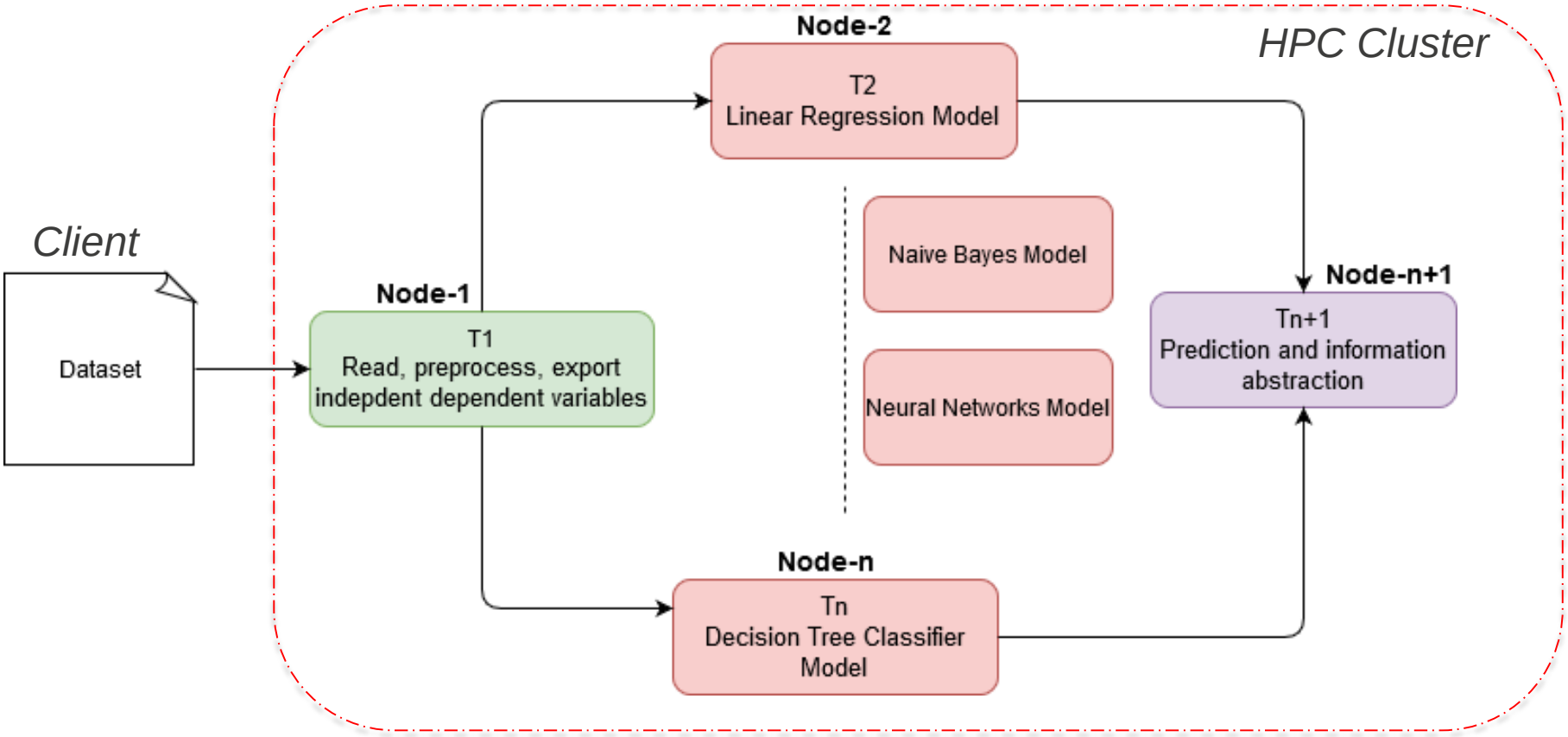
```
  GNU nano 4.9.3                        expectedValue.txt
Expected Salary from Regression is $ [45618.17859389]
Expected Salary from DT is $ [70000]
{"Predicted Salary (S) by Regression and DT Model": ["[45618.1785938945]", "[70000]"]}
```

myWorkflow.cwl

```
#!/usr/bin/env cwl-runner
cwlVersion: v1.0
class: Workflow
inputs:
  experience: int
  interview: int
  test: int
  iterations: int
  datasetpath: string
outputs:
  final_answer:
    outputSource: predict/answer
    type: float
steps:
  read:
    run: /home/bee/cwl2/read.cwl
    in:
      x: datasetpath
    out:
    - answer
  regress:
    run: /home/bee/cwl2/regress.cwl
    in:
      x: iterations
    out:
    - answer
  decisiontree:
    run: /home/bee/cwl2/decisiontree.cwl
    in:
      x: iterations
    out:
    - answer
  predict:
    run: /home/bee/cwl2/predict.cwl
    in:
      x: experience
      y: interview
      z: test
    out:
    - answer
```

Parallelizing ML Tasks across cluster nodes using BEE engine

- Since CWL-Runner executes each step sequentially, BEE engine can facilitate simultaneous execution of ML models across HPC nodes



Discussion

- Representing Machine learning models for data analyses using a CWL workflow is relatively a new use-case in an HPC environment

- Benefits of new workflow:
  - **Three key properties:** *Portable, flexible, reproducible (1-workflow → ML cases)*
  - Automate machine learning stages into an HPC cluster
  - Parallel execution of machine learning algorithms across modern hardware architectures in a cluster
  - Dynamic scheduling of ML models to reduce computational overhead

The proposed CWL ML workflow is loosely-coupled between steps

- Helps avoid redundant execution of steps on the same dataset
  - *If the workflow contains a job that has previously been executed and the outputs are still available, the engine can reuse them even if the job was part of a different workflow run*
- Scalable to 'n' number of ML models

- Caveats:
  - *Assumes that the ML models have been learned, and is frozen, and now needs to be applied to a large batch of data for distributed processing across cluster*
  - *For ML code optimization and scalable auto-tuning, a more complex CWL workflow would be required*

Future Work and Conclusion

- CWL workflows are designed for extensive data analyses applications
  - Allows data elements to be transformed during runtime
  - Nested workflows can be written as steps in CWL for reusing existing code

- Using a CWL workflow for ML algorithms:
  - We can make use of available computational capacity for evaluating performance of different ML models operating on large scale data

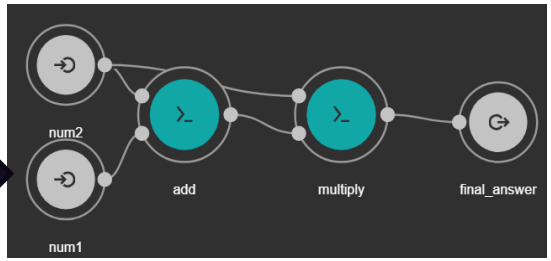
- **To further explore:**
  - BEE engine can be integrated with the proposed ML workflow to parallelize execution of different models on the dataset
    - ❑Parsing and executing machine learning workflows with BEE
  - Containerize current workflow to test for software dependencies and portability

A Python based handy tool for creating CWL workflows

- CWL→ specific schema, fields, and style to adhere
- **Alternative:** *ScriptCWL*, a python package for creating workflows in CWL,
- If we input a number of CWL Command line tools, it generates a workflow

```python
from scriptcwl import WorkflowGenerator

with WorkflowGenerator() as wf:
  wf.load(steps_dir = '/Users/raginigupta/PythonCodes/cwl')

  num1 = wf.add_input(num1='int')
  num2 = wf.add_input(num2='int')

  answer1 = wf.add(x = num1, y = num2)
  answer2 = wf.multiply(x = answer1, y = num2)

  wf.add_outputs(final_answer = answer2)

  wf.save('add_multiply_example_workflow.cwl')
```

```yaml
#!/usr/bin/env cwl-runner
cwlVersion: v1.0
class: Workflow
inputs:
  num1: int
  num2: int
outputs:
  final_answer:
    outputSource: multiply/answer
    type: int
steps:
  add:
    run: add.cwl
    in:
      x: num1
      y: num2
    out:
    - answer
  multiply:
    run: multiply.cwl
    in:
      x: add/answer
      y: num2
    out:
    - answer
```



Existing Workflow Systems

Computational Data Analysis Workflow Systems

Permalink: https://s.apache.org/existing-workflow-systems

An incomplete list

Please add new entries at the bottom.

See also: https://github.com/pditommaso/awesome-pipeline

1. Arvados http://arvados.org
2. Taverna http://www.taverna.org.uk/
3. Galaxy http://galaxyproject.org/
4. SHIWA https://www.shiwa-workflow.eu/
5. Oozie https://oozie.apache.org/
6. DNANexus https://wiki.dnanexus.com/API-Specification-v1.0.0/IO-and-Run-Specifications# https://wiki.dnanexus.com/API-Specification-v1.0.0/Workflows-and-Analyses#
7. BioDT http://www.biodatomics.com/
8. Agave http://agaveapi.co/live-docs/
9. DiscoveryEnvironment http://www.iplantcollaborative.org/ci/discovery-environment
10. Wings http://www.wings-workflows.org/
11. Knime https://www.knime.org/
12. make, rake, drake, ant, scons & many others. Software developmer manage workflows related to compiling and packaging applications. For the most part these are file based and usually run on a single node, usually supporting parallel steps (make -j) and in some cases able to dispatch build steps to other machines (https://code.google.com/p/distcc/) https://github.com/Factual/drake
13. Snakemake https://bitbucket.org/snakemake/snakemake
14. BPipe http://bpipe.org
15. Ruffus https://code.google.com/p/ruffus/
16. NextFlow http://nextflow.io
17. Luigi http://github.com/spotify/luigi
18. SciLuigi. Helper library built on top of Luigi to ease development of Scientific workflows in Luigi: http://github.com/pharmbio/sciluigi
19. Luigi Analysis Workflow (LAW) https://github.com/riga/law
20. GATK Queue https://www.broadinstitute.org/gatk/guide/topic?name=queue
21. Yabi https://ccg.murdoch.edu.au/yabi
22. seqware Workflows are written in Java and executed using the Oozie Workflow Engine on Hadoop or SGE clusters. Uses Zip64 files to group the workflow definition file, workflow itself, sample settings, and data dependencies in a single file that can be exchanged between SeqWare users or archived. https://seqware.github.io/ https://seqware.github.io/docs/6-pipeline/
23. Ketrew https://github.com/hammerlab/ketrew

References

1. www.ncbi.nlm.nih.gov/pmc/articles/PMC5166558
2. Dataset source: https://www.kaggle.com/pankeshpatel/hiring
3. https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems
4. CWL guide: http://www.commonwl.org/user_guide/21-1st-workflow/index.html

Acknowledgements

- HPC Supercomputing Institute
  - Julie Wiens
- BEE Team
- Mentors: Tim Randles, Pat Grubel

Thank you

Questions?