Simulation Monthly Update (Track 4)

Raees Khan, Paul Nilsson,
Sairam Sri Vatsavai

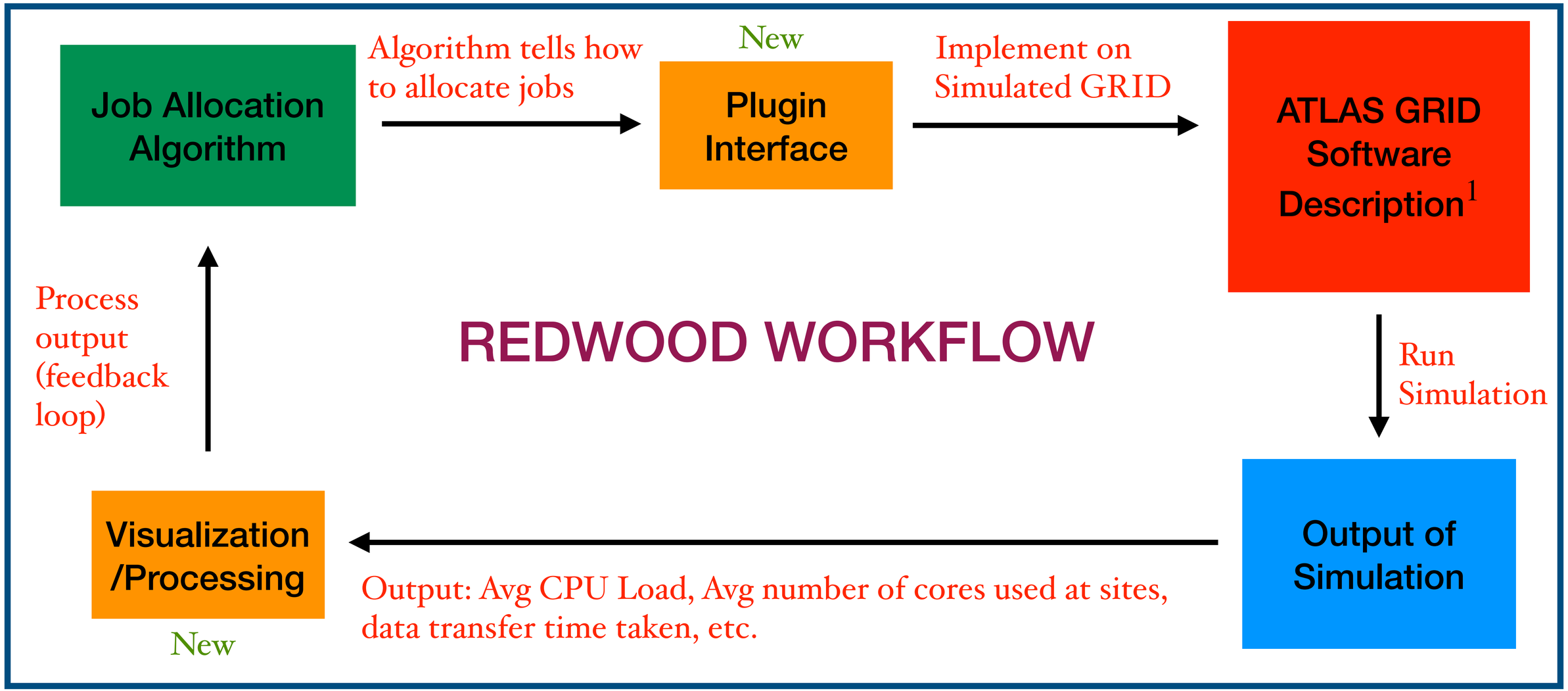
Date: **January 15, 2025**
Meeting: **REDWOOD Monthly Meeting**

Brookhaven National Laboratory

University of Pittsburgh

ATLAS EXPERIMENT

Need for a Distributed Computing Software Description in REDWOOD



1. HEP experiments are using Grid paradigm for distributed computing

Summary of Discussion at Pittsburgh Workshop

- Ignore the complication of breaking tasks into jobs, work at the level of jobs (PanDa experts).
- Take information of Jobs from Historic Data, use this to validate and calibrate the software description.
- Dispatcher (class which assigns jobs to resources) should be abstract so we can plug in any dispatching algorithm whether that's the current PanDa algorithm, Analytic algorithm being developed by CMU or ML algorithm being developed by BNL.
- Need an easy to use visualization tool to show what goes on as the simulation progresses, CPU load, Link load, etc. Moreover it should be interactive that we can turn off clusters or CPUs at various sites and visualize how the job dispatching algorithm deals with this.
- Error Injection into simulator (See Paul's presentation).
- After validation, work with people in Track 4 who can use historic data to generate realistic pseudo jobs using ML.

Simulating Historical Jobs

Historical Jobs:
- Past Panda user jobs collected to build AI surrogate models.

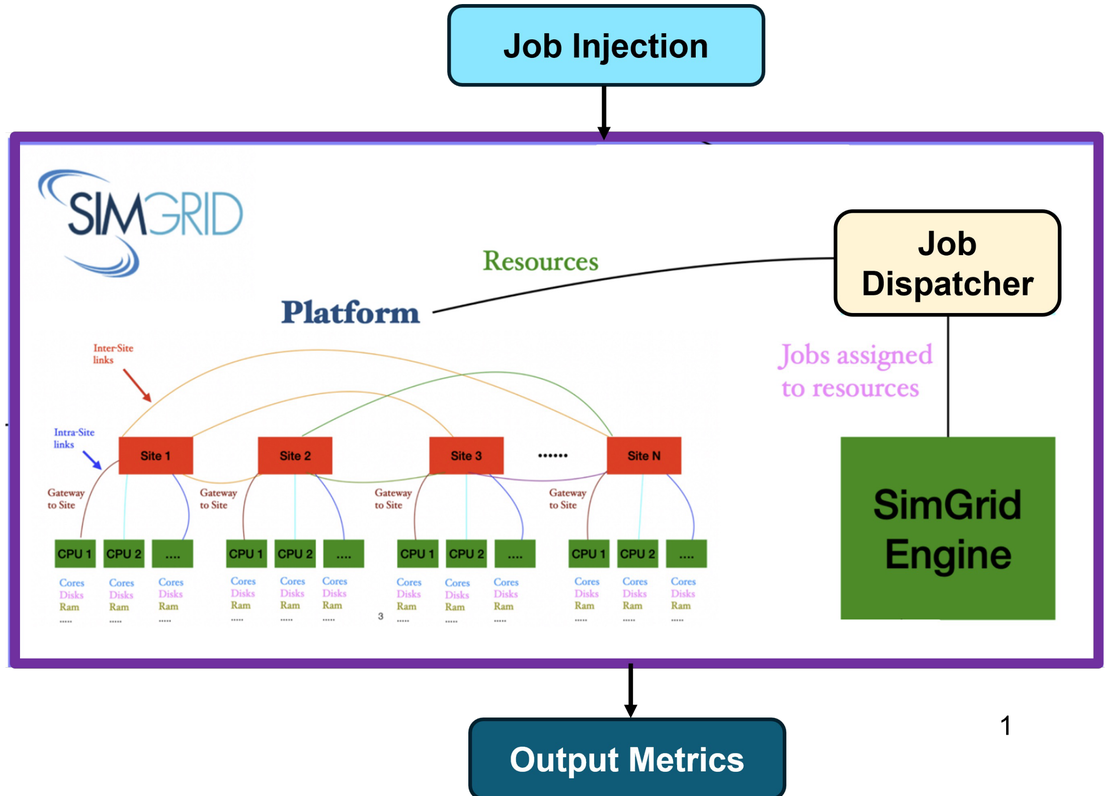
Job Injection (Latest Development):
- An updated parser that reads historical jobs from a data source (currently from a CSV file).
- The CSV file is created using V1 data.
- A new job manager to create jobs and send them to the job dispatcher for resource allocation.

Job Dispatcher (Latest Development):
- Allocates resources based on computing site information from historical jobs.

SimGrid Engine:
- Simulates the execution of jobs on the platform.

Output Metrics:
- Generates job-specific execution metrics, such as execution time, I/O read size, and write size.

Test With 1M Jobs (Latest Development):
- Ran on Lenovo Thinkpad with 13th Gen Intel(R) Core(TM) i9-13900H 2.60 GHz.
- Finished in 1hr 30mins.



Brookhaven National Laboratory

Plugin Architecture

- We have many different kinds of job allocation algorithms we will use in this project.
  - Simple algorithms for initial testing, Historic Data for validation, more complex algorithms being developed by Track 1 & Track 4.

- Need a flexible design which let's any developer design a job allocation algorithm and plug it in the simulation, without having to change the core code.

- Solution: Plugins in the form of shared libraries (.dylib or .so) that can be plugged in the simulator at run time.

Plugin Architecture cont'd.

- To aid in the writing of plugins, the core codebase will come with an abstract class, which will serve as a blueprint on how to write a job allocation algorithm and will be installed in some standard location.

- The plugin can then be developed completely separately, inheriting from the provided abstract class and built into a shared library.

- Once the plugin is ready it can be plugged into the simulation, by specifying the path to the plugin (shared library) in the simulator configuration file.

Visualization Tool (beta)

WebApp using REACT.

Simulation dumps information about sites and jobs, live.

CPU load & job info to be displayed live.

Current work on integration with backend.

Outlook

- Ongoing work on collecting missing information about PanDa sites.
- Validation & Calibration of simulator results with actual historic data.
- We plan to communicate with algorithms team working on job scheduling as soon as the plugin architecture is finalized, even something preliminary we can start to test.
- Work on visualization tool (will propose as a summer student project at CERN).
- In discussion about writing a paper on the simulator framework, calibration with ATLAS data, flexible plugin architecture, visualization application.
- Suggestions & comments? Checkout the Github repo here