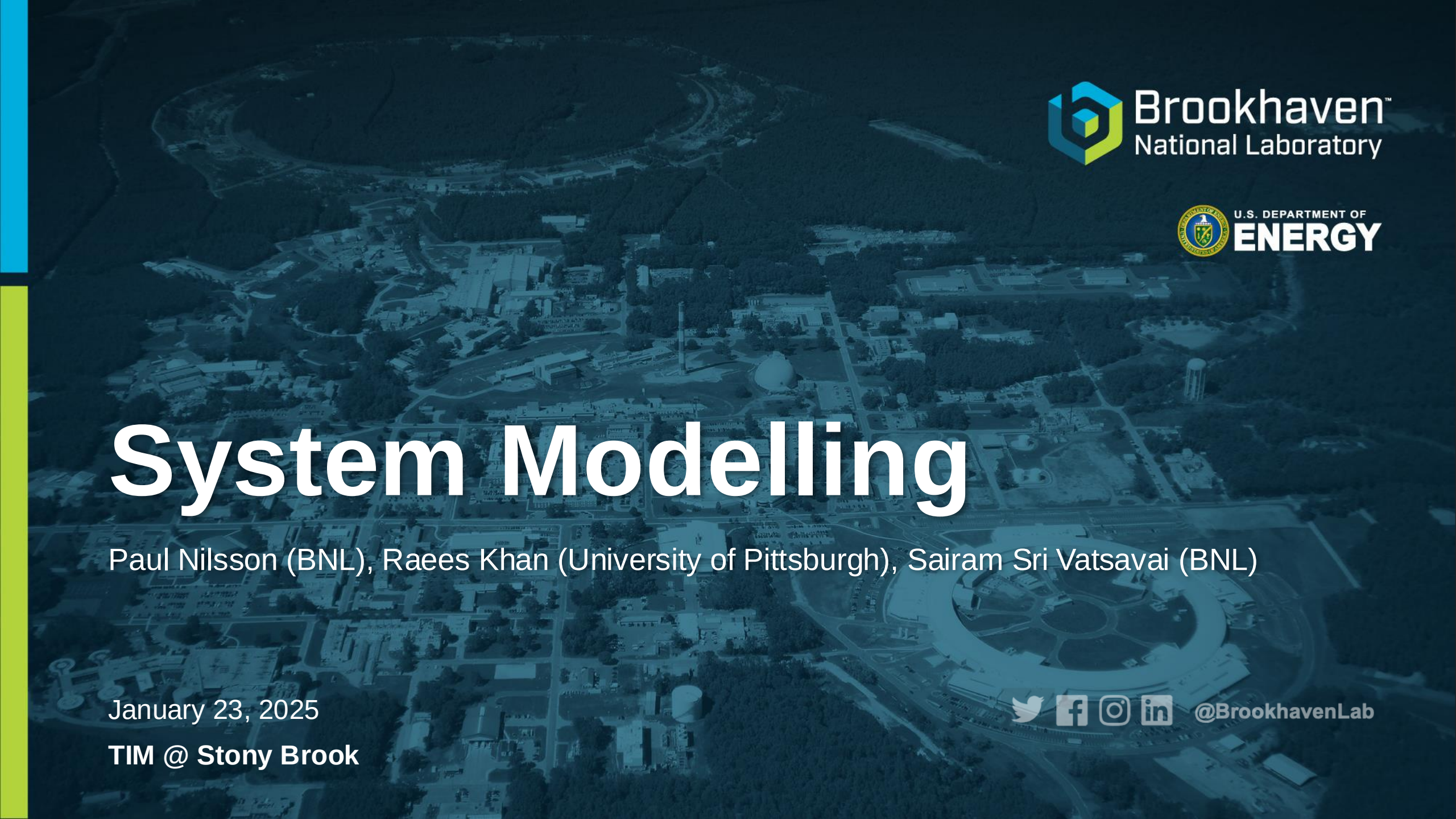


System Modelling

Paul Nilsson (BNL), Raees Khan (University of Pittsburgh), Sairam Sri Vatsavai (BNL)

January 23, 2025

TIM @ Stony Brook

Introduction (and reminders)

- REDWOOD is an ongoing project that started in 2023
- A main objective is the optimization of WFMS to enhance system resilience
- Two targets for optimization
  - Brokerage – where should the job go?
  - Data placement – where should the data go?
- Different approaches will be used, including ML techniques
- This talk will discuss how WFMS could be modelled and simulated
  - Different algorithms for optimization could then be studied
- The presentation is an update with the latest developments

Simulators

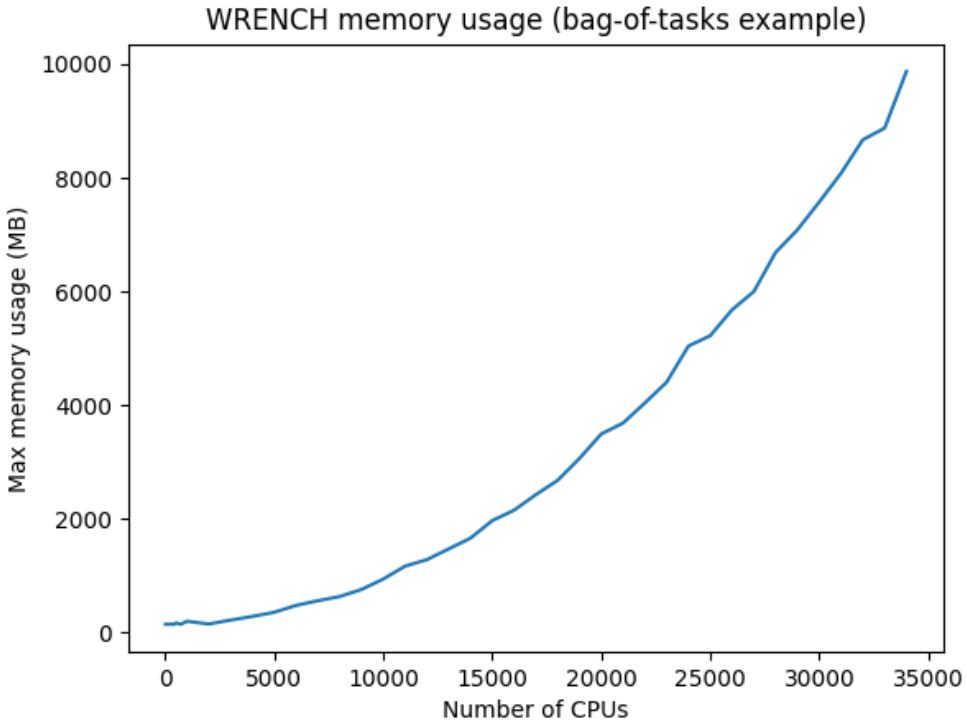
Which tools are or interest?

A well-known framework for developing simulators of distributed applications targeting distributed platforms, which can in turn be used to prototype, evaluate and compare relevant platform configurations, system designs, and algorithmic approaches. In active development for over 20 years

An open-source framework designed to make it easy for users to develop accurate and scalable simulators of distributed computing applications, systems, and platforms. WRENCH is built op top of SimGrid.

WRENCH

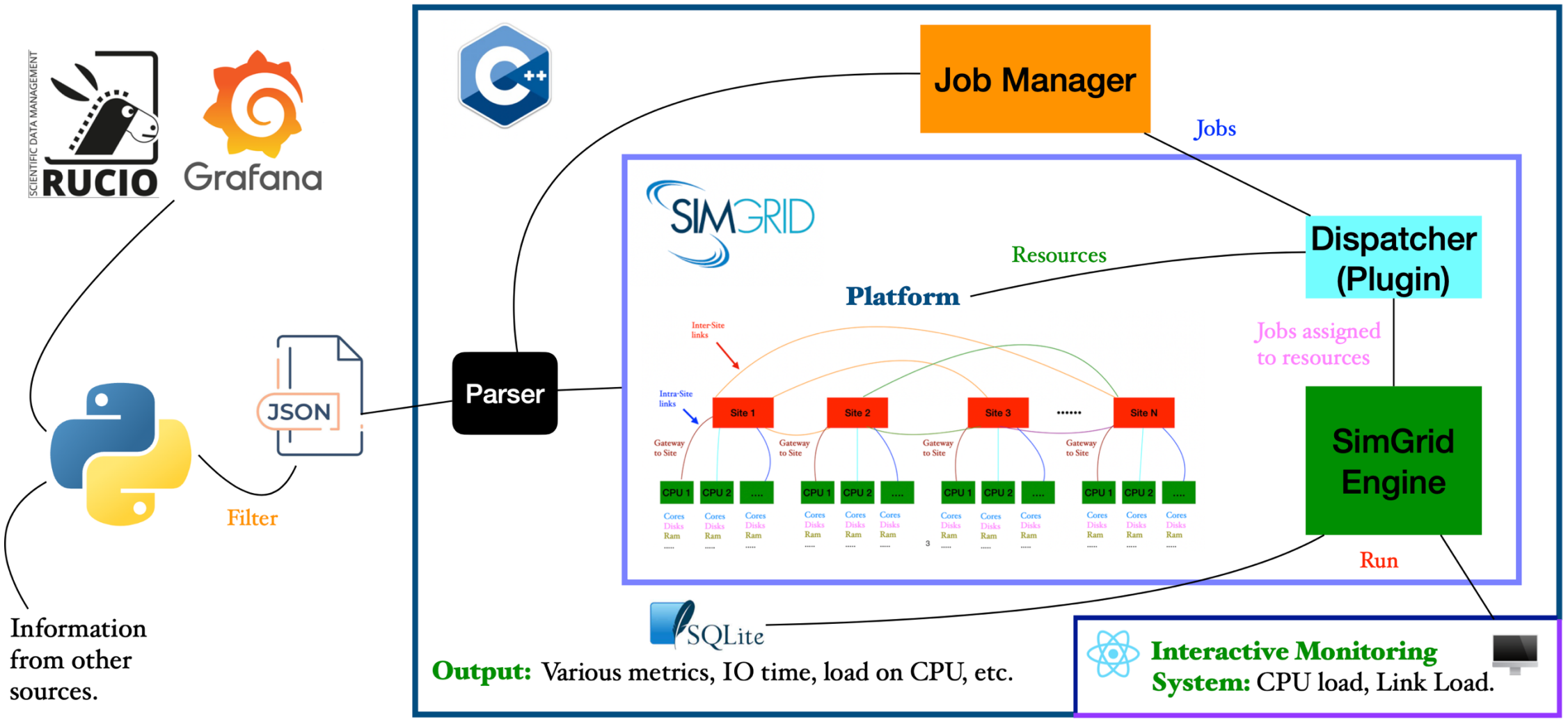
- WRENCH would be a good choice for building a simulator as it was developed to reduce the complexity of SimGrid, and make things simpler for the developer
- However, testing showed that it is too memory hungry for a larger amount of CPUs
  - Modified standard example
    - Dynamically creating simple geometry for single computing resource
  - Dramatic increase of used memory as number of hosts increase
  - Site of BNL size needs around 8 GB to run simulation
    - I.e. 150+ grid sites won't be possible
  - Most of the memory is used by geometry, another 10-20% increase in memory usage when simulation starts (10,000 jobs)



SimGrid

- Simulator for ATLAS using latest version of SimGrid is in development
  - Code repository https://github.com/REDWOOD24/ATLAS-GRID-SIMULATION
- Topology generated from JSON files using metrics data from Rucio and from other sources
  - Connections between sites and bandwidths used to construct grid of sites
  - Number of CPUs/cores extracted from job records using data reported by the pilot
  - Scaled corepower used as GFLOPS
- Trivial job generator used so far
  - Jobs have so far been generated from a gaussian distribution
  - Also, jobs can be generated using historical data
- A study of memory usage as function of number of CPUs looks good
  - Memory usage did not pass 3 GB while running 5k tasks with full ATLAS grid geometry
- Error handling to be implemented
  - I.e. currently 100% of jobs succeeds

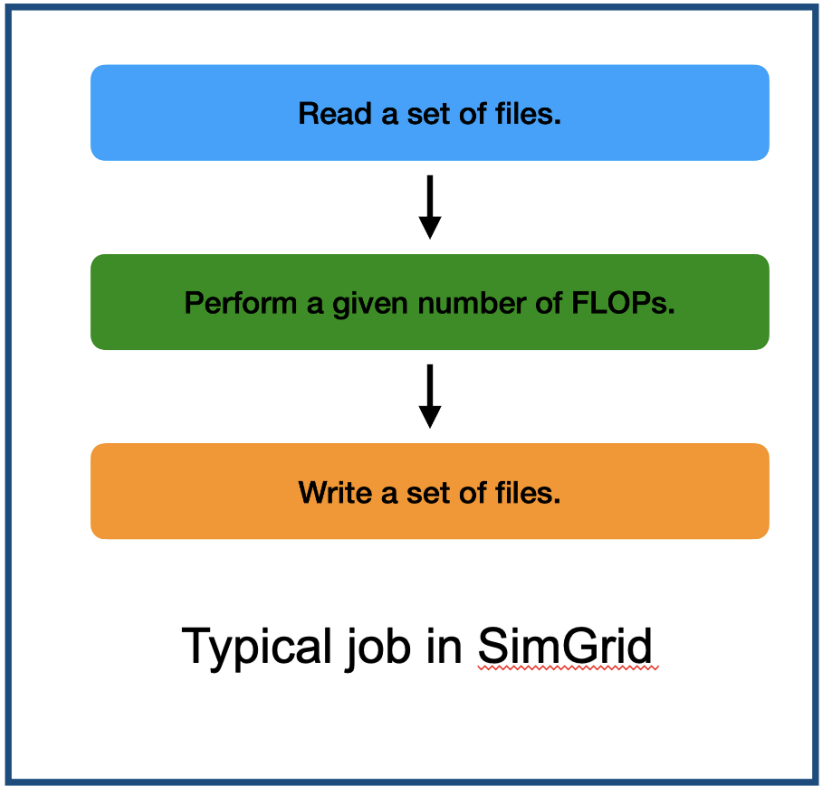
Workflow



R. Khan

Dispatcher

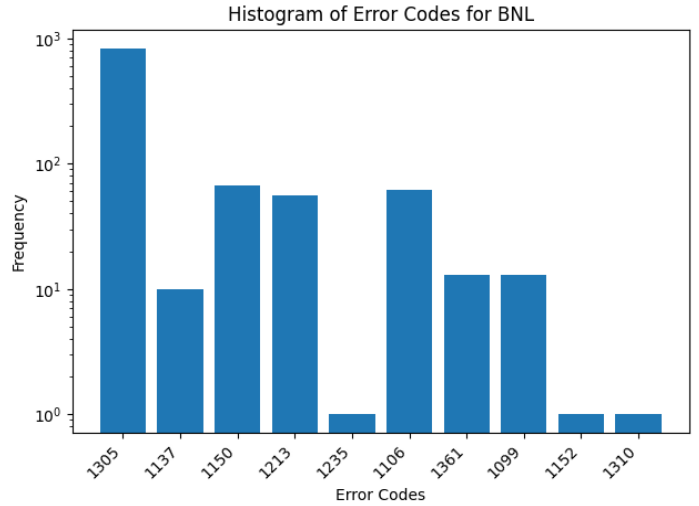
- Dispatcher takes two input: prioritized job queue & platform resource details
- Jobs are then assigned to CPUs at various sites
- Initial “simple” algorithm is used to match jobs to resources
  - Sites ranked according to quality of CPUs, e.g. storage, cores, speed, and computational capacity, etc
  - For each job, an optimal CPU is found by looking at various sites (starting off with the best site and search depth limited to 20 for efficiency). CPU quality reduces as load on it increases
  - Round-robin strategy used once 50% of site CPUs are allocated
- Dispatched jobs sent to SimGrid for execution



Plug-in Architecture

- There are several different kinds of job allocation algorithms we will use in this project
  - Simple algorithm for initial testing, historic data for validation, more complex algorithms to be developed by REDWOOD 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: Plug-ins in the form of shared libraries that can be plugged in the simulator at run time

Simulation of Errors



- 5-10% of all grid jobs fail for different reasons
  - Need to mirror this for the SimGrid simulation to be realistic
- Simulate errors in SimGrid using real data from PanDA
  - Relevant categories depending on how deep the simulation will go; errors on the simulated “worker node” easiest to start with
- Data source: (pilot errors from Tania)
  - [last 24h] https://bigpanda.cern.ch/errors/?json&hours=24&limit=10000000&fields=errsBySite
  - [historical period] https://bigpanda.cern.ch/errors/?json&limit=10000000&fields=errsBySite&date_from=2024-12-01T00:00:00&date_to=2024-12-02T00:00:00
- When a given job runs, simulation should decide if job should finish or fail
  - Need to know total number of jobs for same time period (add to JSON)
- If failure, draw random error from distribution for the queue the job is running on
  - Depending on which error it is, set job info accordingly (stage-in error e.g., will not spend any CPU, or very little)
  - Also affects simulated data transfers (no output files will be produced when there is a stage-in error, only the job log will be staged out)

Visualization

- Now exporting job and CPU/worker node status to sqlite file that can be read by monitor
  - Provides live info from running simulation
- Would also like to be able to switch on/off worker nodes or even whole sites on the fly from the monitor

Status and Plans

- We have tested both WRENCH and SimGrid, and decided on building the simulation around SimGrid
  - Developed a preliminary software description of the ATLAS grid and tested submitting jobs to it and collecting the output statistics
- Need to make things more accurate (CPU, DISK, etc)
  - Currently, the number of CPUs are randomly distributed (to realistic order) [we have the info], disk sizes are preliminary
  - Investigate if disk info is available for all queues, chase it down if missing
- Want to do basic error injection in simulator
- Develop monitoring further
- Move from simple to realistic brokering (longer term goal)
  - Need to discuss/think how exactly to implement this (the algorithm used by PanDA is rather complex)
  - Implement in steps of increasing complexity