Improving batch job scheduling with AI

AIRA Seminar, 19.01.2023

Hardware Acceleration
Lab

Bartosz Soból

Outline

- What is batch job (scheduling)
- Heuristics-based scheduling
- AI-based schedulers
- Possible improvements

What is a batch job?

- In HPC, batch job is a computing task described in a form of (batch) script
    - Usually a bash script
    - Contains additional directives describing required resources
    - Setups environment (environment variables)
    - Executes commands doing actual work
- Batch jobs are submitted by users (submission/login node)
- Schedulers decide when and where (computing node) they are run
- Most common method of interacting with computing cluster

Heuristics-based scheduling

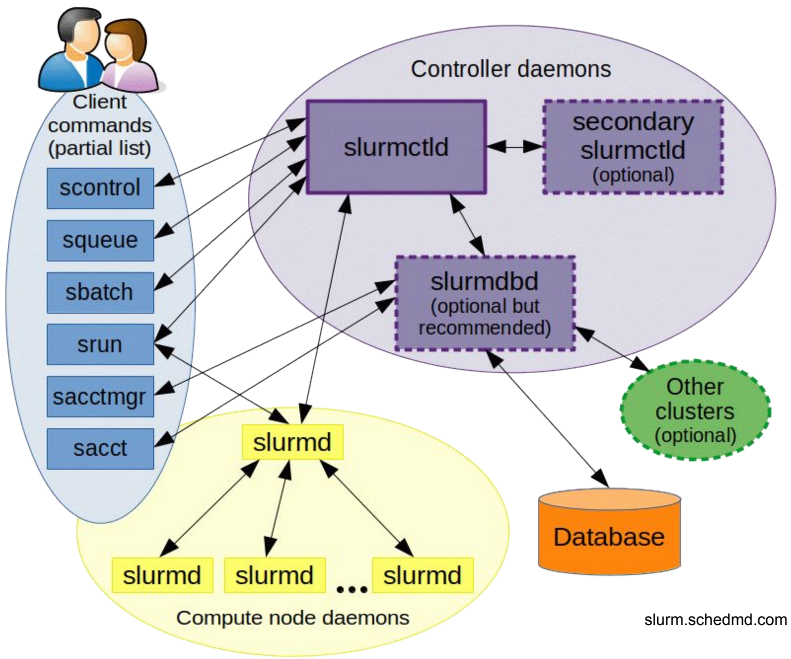
- Typical job lifetime
    a. Submitted by user
    b. Assigned priority
        - Factors: age, size, fair-share, queue, user-controlled priority, etc.
    c. Scheduled with backfilling and executed

HPC Wiki

- Priority factors can be configured and weightened
    a. Lots of possible configurations, more factors increase complexity
- Job prioritization, factor selection and importance is a research topic itself
- Most popular schedulers: Slurm, PBS

Slurm

- Most popular job scheduler and workload manager
  - 60% TOP500 supercomputers in 2019
- Used in all Polish scientific supercomputers (PLGrid network)
- Complex cluster management system with user management, accounting, monitoring and other features



slurm.schedmd.com

Slurm: example

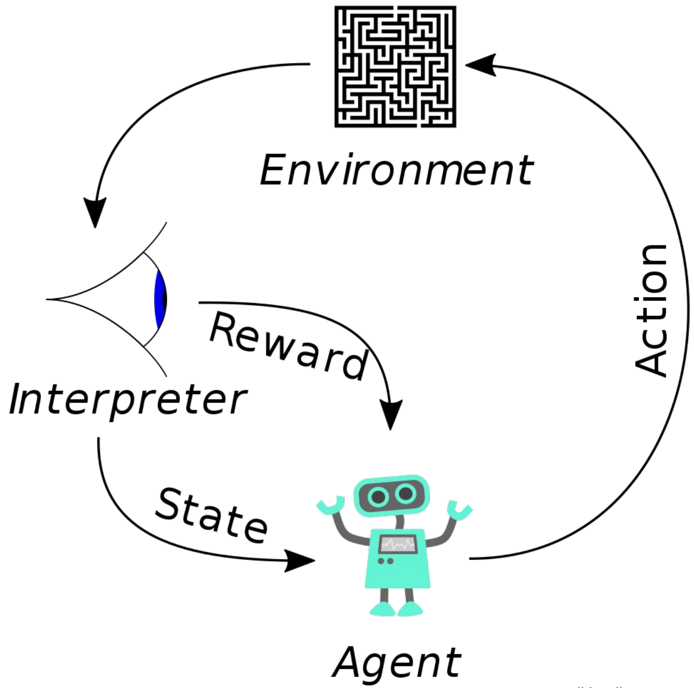
sample_job.sh

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=16G
#SBATCH --gres=gpu:1
#SBATCH --time=02:00:00
#SBATCH --partition=gpu-v100

module add python
module add tensorflow

python actual_task.py
```

```
[ ~]$   sbatch -J test sample_job.sh

Submitted batch job 16846

[ ~]$   squeue

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
16846  gpu-v100 test user PD  0:00    1       (Priority)

[ ~]$   sacct

JobID JobName Partition Account AllocCPUS State ExitCode
16846    test  gpu-v100 testAcc         8 RUNNING    0:0
```

Reinforcement learning

- Agent: scheduler
- Environment: cluster, computing nodes with resources, queues, users, jobs
- Action: decision on job execution
- Possible reward factors for scheduling
  - Job waiting time
  - Utilization of reserved resources
  - Idle resources
  - Job execution time
  - ...



wikipedia.org

AI-supported scheduling

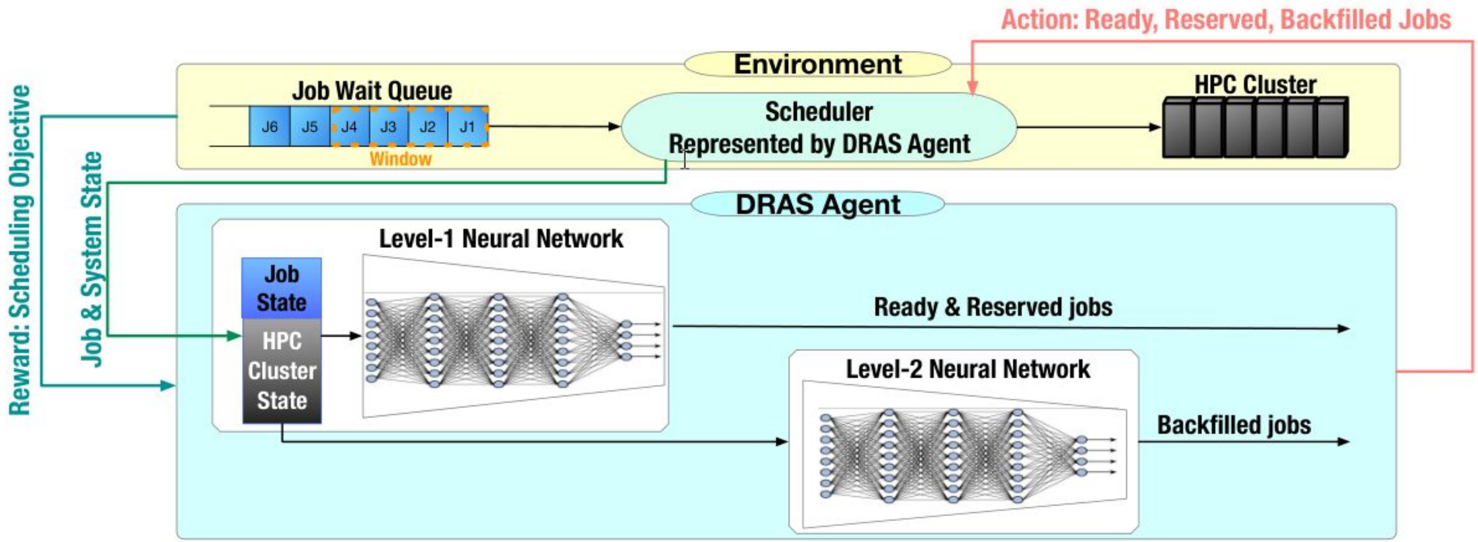
- Use reinforcement learning techniques
- Two approaches:
  - Standalone scheduler: Make AI-based decisions
    - Usage of more scheduling factors doesn't multiply configurable parameters
    - Black-box scheduling
  - Additional layer on top of existing scheduler:
    Alter decisions of a classic scheduler underneath
    - Well-known haurustics part is still there
    - Can still benefit from additional factors and reinforcement learning
    - Better explainability

Example: DRAS (Deep Reinforcement Agent for Scheduling)

- Uses the first approach - standalone scheduler
- Takes cluster state and job queue as input
- Selects jobs to start execution
- Two neural networks
  - first select job for immediate execution, second directs backfilling
  - 22 to 162 million trainable parameters - depending on cluster size
  - convolutional and fully-connected layers
- Two RL approaches tested: policy gradient and Q-learning
- Two job trace datasets: 121K and 2.5M jobs for training and evaluation

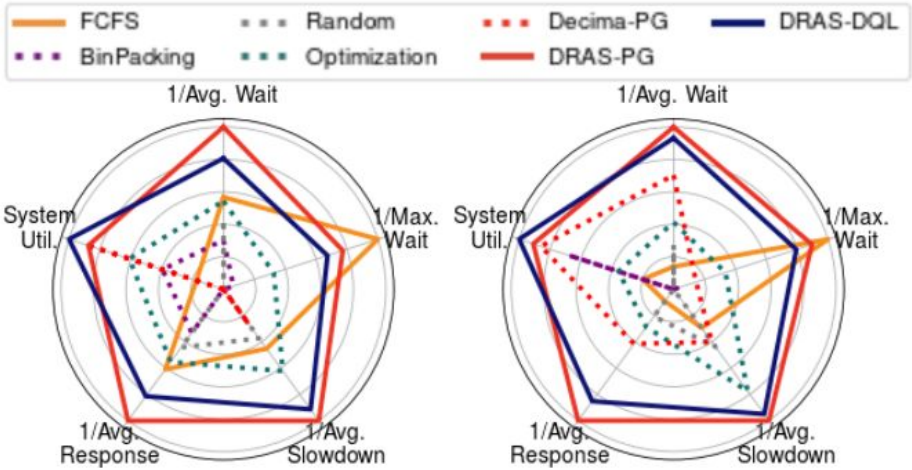
Example: DRAS (Deep Reinforcement Agent for Scheduling)

Architecture



Example: DRAS (Deep Reinforcement Agent for Scheduling)

- Evaluation and results:
  - Simulated environment
  - Tested against various simple scheduling policies
  - Not evaluated against complex systems as Slurm



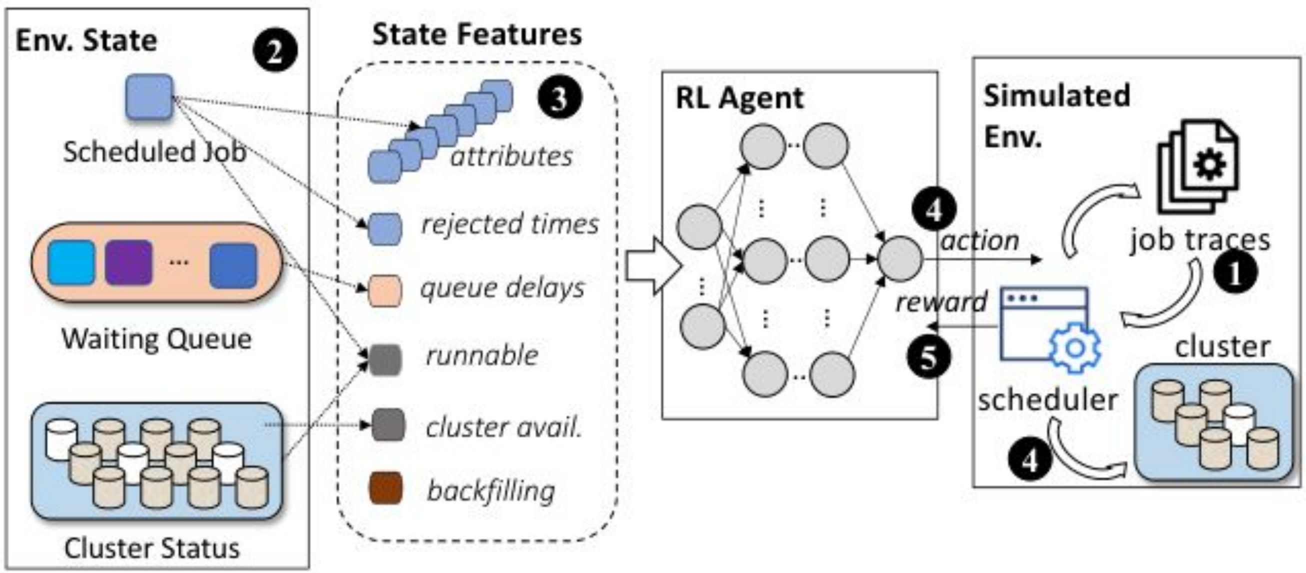
(a) Theta (b) Cori

Example: SchedInspector

- Lives on top of existing classic scheduler
- Analyses submitted job, other jobs in queue, live cluster status and other factors
- Can reject decisions of underneath scheduler
  - e.g. delay longer job in order to run more shorter jobs
- Very simple model design - 2 MLPs with only ~2k parameters total
  - Actor-critic model
- Training:
  - Dataset: real accounting data (but from very old clusters)
  - Batches of 256 jobs

Example: SchedInspector

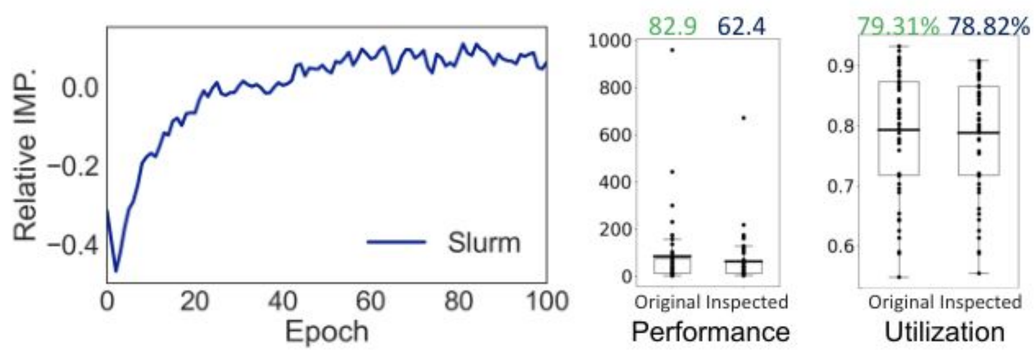
Architecture



Example: SchedInspector

Evaluation and results:

- Simulated simplified cluster environment
    - Assumption that runtime of the same does not change
- Tested against various simple scheduling policies
- And standard Slurm priority multifactor backfilling
- Performance measure average of $(max((w_j + e_j)/max(e_j, 10), 1)$ over 50 jobs -> lower is better



Example: SchedInspector

- As being independent from base scheduler, can be deployed gradually (compared to DRAS)
  - For specific types of jobs, only on certain queues, etc.
  - Simpler for administrators
  - Better explainability
  - Good as a first step for adoption AI in this application
- Current evaluation methods and training datasets are not ideal
  - Architectures of HPC clusters changed a lot
- Authors plan to integrate SchedInspector with Slurm

  in real-life environment

Possible improvements

Training and evaluation methods

- Datasets should include job traces from modern clusters
  - Different cluster architectures and node types
  - Large multi-socket nodes
  - GPUs
- More realistic environment
  - Using real cluster for training and evaluation might be impossible
  - Simulated environments can be improved
    - Introduce random variance of execution time
    - Add I/O and network bottleneck simulation

Possible improvements

I/O requirements for job

- HPC systems usually use distributed filesystems (lustre) based on HDDs
  - Access to SSD drives is still limited even on newest systems
  - I/O-heavy jobs can execute many times longer if filesystem is busy
  - Often, delaying job (even for hours) can lead to lower queue+execution time
- I/O characteristics may be made a priority/scheduling factor
- How can scheduler know if jobs is I/O-heavy?
  - Additional `#SBATCH`-like directive limiting I/O
    - Complicated for users
  - Scheduler can learn from common jobs and filesystem characteristics

Possible improvements

Access to script content

- What if scheduler can *read* more parts of job script?
  - Software modules used
  - Commands to be executed
- Difficult to accomplish
  - Large model
  - Difficult to train for general usage
  - Privacy issues?

sample_job.sh

```

module add python
module add tensorflow

python actual_task.py
```

Summary

- Heuristics-based batch job scheduling methods and schedulers are used for many years, well established and understood
- AI-based solutions are emerging
    - Can take into account more factors
    - Open possibilities to provide better fine-tuned scheduling
    - Interesting topic and ongoing research from various groups
    - Real-life evaluation is necessary

Bibliography

1. *Deep Reinforcement Agent for Scheduling in HPC*
   Yuping Fan, Zhiling Lan, Taylor Childers, Paul Rich, William Allcock, Michael E. Papka
   https://arxiv.org/abs/2102.06243
2. *SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning*
   Di Zhang, Dong Dai, Bing Xie
   https://dl.acm.org/doi/abs/10.1145/3502181.3531470
3. *RLScheduler: An Automated HPC Batch Job Scheduler Using Reinforcement Learning*
   Di Zhang, Dong Dai, Youbiao He, Forrest Sheng Bao, Bing Xie
   https://arxiv.org/abs/1910.08925