Environmentally Sustainable AI via Power-Aware Batch Scheduling

Daniel Wilson, Atanas Atanasov, Christopher Cantalupo, Brad Geltz, Lowren Lawson, Asma Al-Rawi, Ali Mohammed, Sid Jana, Alejandro Vilches

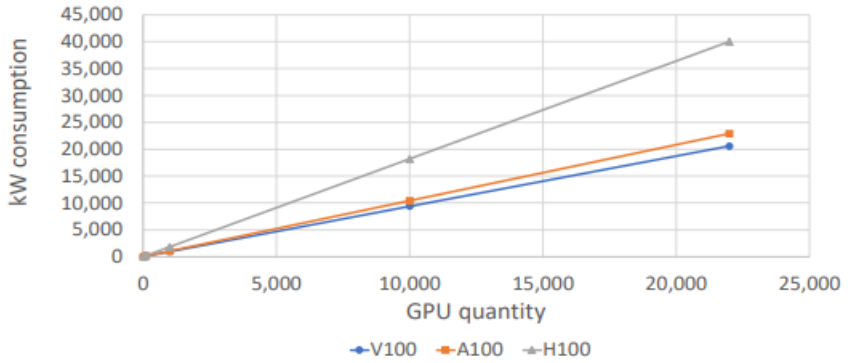
https://geopm.github.io

AI Workloads Demand a Lot of Power

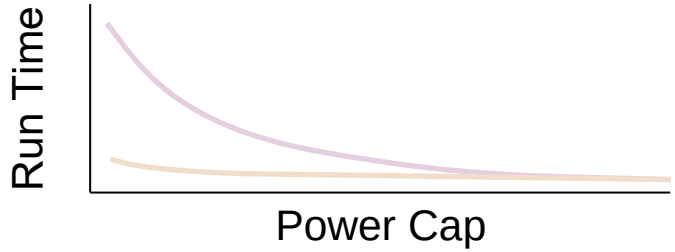
Overview of AI workloads in data centers.

Schneider Electric estimate	2023	2028
Total data center workload	54 GW	90 GW
AI workload	4.3 GW	13.5-20 GW
AI workload (% of total)	8%	15-20%
AI workload (Training vs Inference)	20% Training, 80% Inference	15% Training, 85% Inference
AI workload (Central vs Edge)	95% Central, 5% Edge	50% Central, 50% Edge

Source: Schneider Electric White Paper 110



- Inference uses a larger share of total energy, why target training?
  - Training is a very intense use of energy done once over a limited time window
  - Training is done on dedicated resources similar to HPC configurations
  - Energy efficiency techniques from HPC can be applied
  - Inference is done in a more distributed manner over a range of shared resources
- Some workloads use power more effectively than others
- Estimates for training a single LLM range from 300 - 600,000 tons of CO2



https://dl.acm.org/doi/10.1145/3442188.3445922 & https://arxiv.org/abs/1906.02243

Software Stack for K8 Power Resource

We describe a solution leveraging:

- Kueue's resource management features
- Kubeflow's mpi-operator
- GEOPM's software power management framework

Can we port this solution to less HPC-specific environments?

- Shared computation resources and cloud environments
- Edge resources with cyclic demand
- AI inference computations

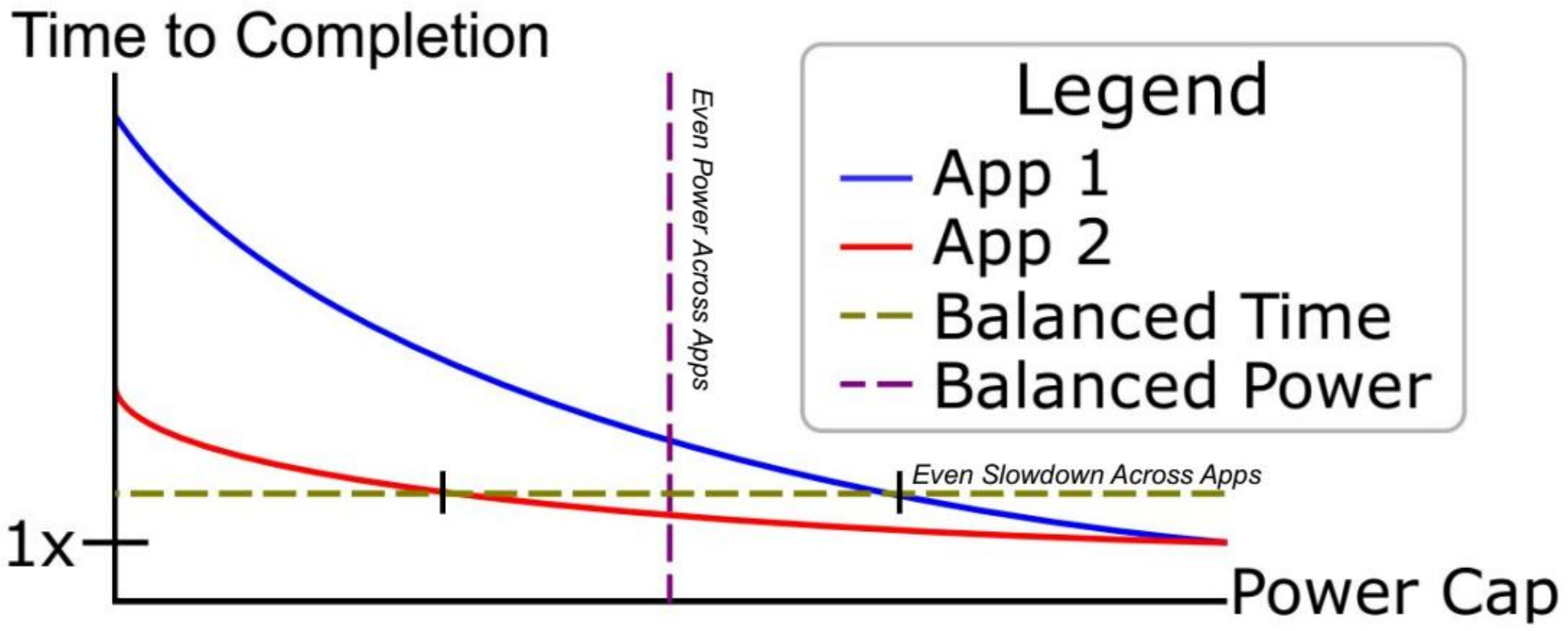
Batch Scheduling AI Workloads in k8s

Specialized software stacks address AI training requirements

- Distribute AI training computation across cluster:
  - Kueue
  - Kubeflow
  - Volcano
- Internode communication with support for HPC fabric:
  - mpi-operator
  - Horovod
- Abstract compute engines for highly optimized solutions:
  - Tensorflow
  - Pytorch

Power Cap Analysis of AI Workloads (Ctd.)

Use Power Models to Limit Slowdown While Sharing Power



Limit Power With Job Performance Targets

Configure Power Limits

- Apply a Kueue configuration with maximum queue power

Model

- Run partial workloads under multiple power caps
- Generate a power model

Run

- Select a desired performance target
- Submit Kueue jobs with model-recommended power targets
- Old power settings are restored automatically after runs

Power as a 1st class Resource with OSS Features

Three new features of Open-Source Software

1. Resource flavor extensions for Kueue (Kubernetes > 1.23)
   - System power as a first-class resource that can be admin managed and client requested
   - Patch enables integration of power resource with mpi-operator

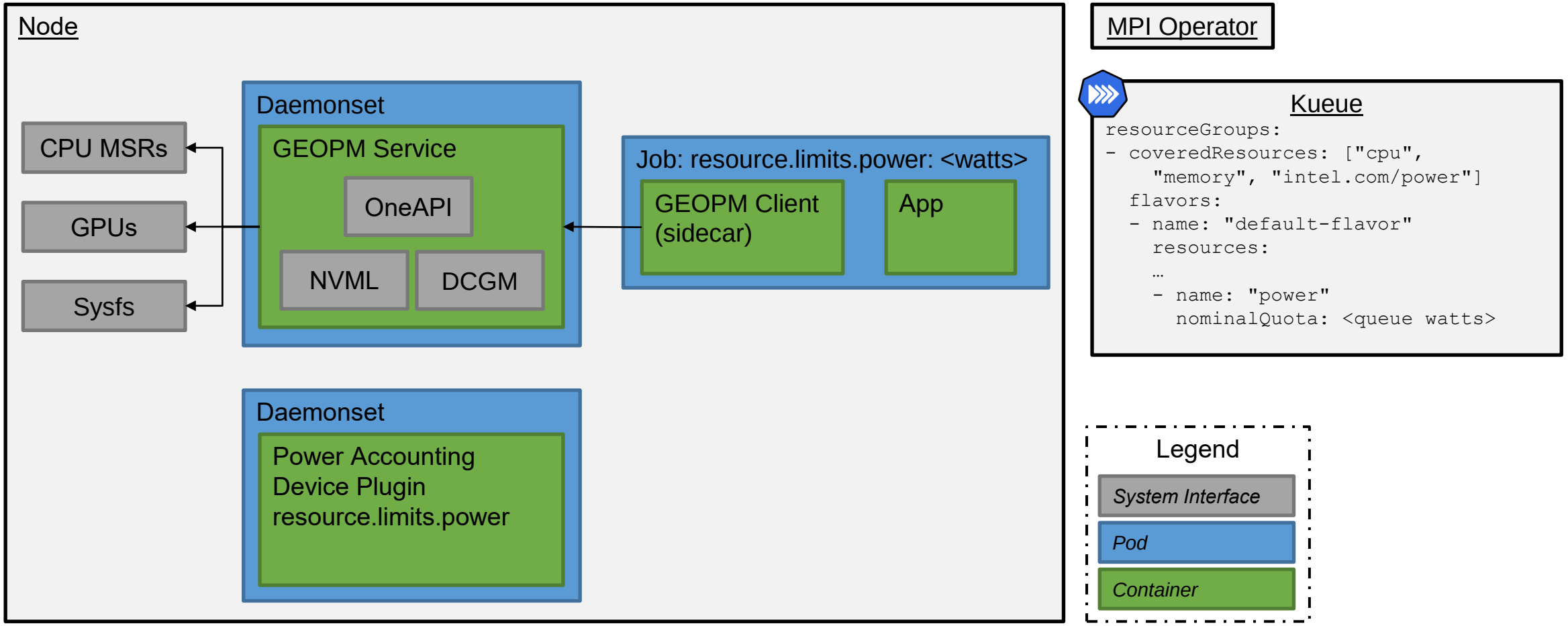
2. Sidecar Containers (alpha feature of Kubernetes > 1.28)
   - Loose coupling between application and control system software
   - Facilitates epilog and prolog extensions for batch jobs

3. Container support for GEOPM Service (gRPC alpha extension to GEOPM 3.0)
   - Control low level hardware knobs and sample energy metrics in unprivileged sidecar

Combining existing packages creates value

- AI Training can use energy optimization techniques developed for HPC
- Save energy and make efficient use of resources
- Software deployment is simplified
- Components have been tested independently

A K8S Architecture to Manage Job Power



Job Power Management Workflow

Kueue

TensorFlow Jobs

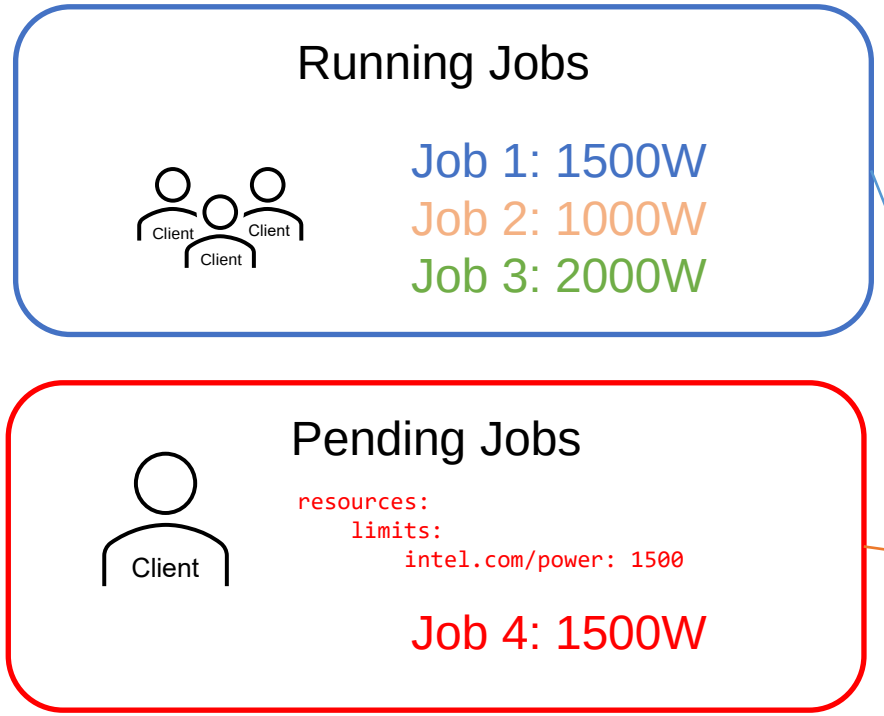
- Client requests queue with **cluster power limit**
- Client requests **Power Cap** via the **power resource**

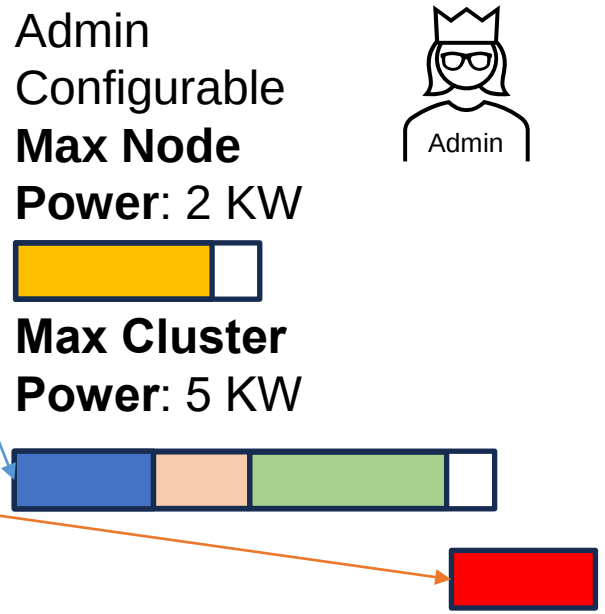
Cluster Queue Resources

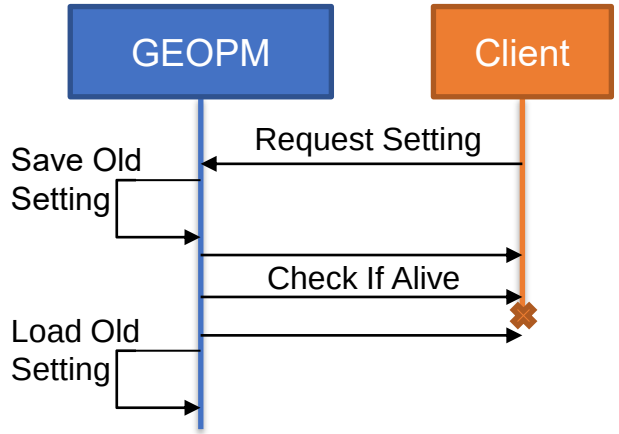
- CPU
- Memory
- GPU queue limit
- Power Limit for the full queue

GEOPM DaemonSet

- Enforce power cap during job execution
- Restore previous settings after job completes







Queue Power Limits in Kueue

- Use a nominal quota for a cluster-level power limit
- Kueue supports *CPU*, *memory* and *device* resources
- Use a **device plugin** to access/request node power
  - Requires a countable resource per node
- Limits are defined in Watts

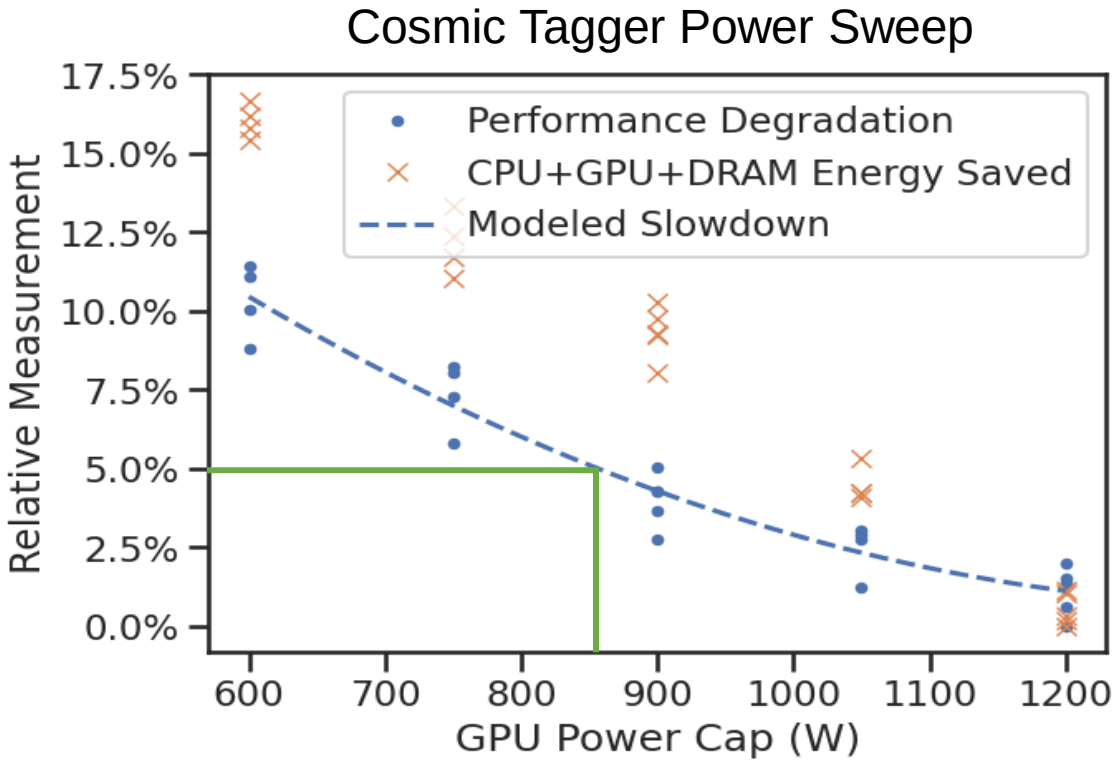

```yaml
apiVersion: kueue.x-k8s.io/v1beta1
kind: ResourceFlavor
metadata:
  name: "default-flavor"
---

apiVersion: kueue.x-k8s.io/v1beta1
kind: ClusterQueue
metadata:
  name: "cluster-queue"
spec:
  namespaceSelector: {} # match all.
  resourceGroups:
  - coveredResources: ["cpu", "memory",
                       "intel.com/power"]
    flavors:
    - name: "default-flavor"
      resources:
      - name: "cpu"
        nominalQuota: 9
      - name: "memory"
        nominalQuota: 36Gi
      - name: "intel.com/power"
        nominalQuota: 10000
---

apiVersion: kueue.x-k8s.io/v1beta1
kind: LocalQueue
metadata:
  namespace: "default"
  name: "user-queue"
spec:
  clusterQueue: "cluster-queue"
```

Job Power Limits in Kueue

- Requires a job supporting MPI, TF, PT, prolog & epilog, user power limit
- Use MPIJob type from Kueue
- Sidecar init containers (v1.28) for prolog/epilogue
  - Sidecar requests X Watt units from the power device-plugin
  - Sidecar invokes GEOPM to set a power cap
- Auto-restore pre-job caps when sidecars finish


```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: pi1
  labels:
    kueue.x-k8s.io/queue-name: user-queue
<…>
Worker:
   replicas: 2
      template:
        spec:
          initContainers:
          - image: dannosliwcd/geopm-service:0.0.7
            name: mpi-prologue
            command: ['sh', '-c',
              'geopmwrite board 0 $USER_POWER_CAP; \
               sleep infinity']
            resources:
              limits:
                cpu: 8
                memory: 1Gi
                intel.com/power: 500
            restartPolicy: Always
```

Extracting Models From Power Sweeps



- Execute application (or a proxy) under multiple caps
  - Generate a parallel processing template*
  - Apply an instance for each power level
- Generate a performance model from sweeps
  - Extract logs from *geopm-client* container
  - Optionally get a figure of merit from app container
  - Model slowdown† from power cap as

$$S(P) = A\left(P_0 - \frac{P}{P_{max}}\right)^2 + B\left(P_0 - \frac{P}{P_{max}}\right) + C$$

- Users can budget power from their time constraints (5% slowdown example annotated)

* https://kubernetes.io/docs/tasks/job/parallel-processing-expansion/. See cosmic-tagger-power-sweep.yaml in our examples

† Run `./model_sweep.py <paths to geopm-client log files> --power-at-slowdown 0.05` to get a recommendation for 5% slowdown

Learnings and Future Work

- New Sidecar Container Feature Simplifies Job Wrappers
- Kueue can limit continuous resources (like power) with adaptations
  - Need to represent power as a discrete collection of devices (e.g., 1 W == 1 device)
  - Allocate and request the resource in units of the adapter devices
  - Would be nice to enable native continuous resources in the future
- Opportunities to Build on Job Power Capping:
  - Evaluate power oversubscription opportunities while capping less-power-sensitive jobs
  - Integrate with container-scoped power metrics (e.g., Kepler)
  - Investigate elastic resource allocation. Guarantee minimal power, more when system permits
    - Example: Allow more power during periods of low carbon intensity or cheap power
    - Example: Measure application performance at run time & boost power while in efficient phases

K8S Sidecars Simplify Job Wrappers

- Kubernetes introduced sidecar init containers as an alpha feature in v1.28
- Init container is a pre-requisite for app container, and ends when the app ends
- Useful for any prologue/epilogue work (like setting a power cap during a job)
- Would like to see sidecar support in future MPI Operator and Kueue releases

Without Sidecars:
App needs to communicate with wrapper

```
shareProcessNamespace: true
containers:
- name: cosmic-tagger
  command: ["mpirun", …]
  <…>
- name: geopm-client
  command: [
    '/usr/bin/sh', '-c',
    'geopmwrite …; tail -f --pid "$(pgrep mpirun)" /dev/null;']
```

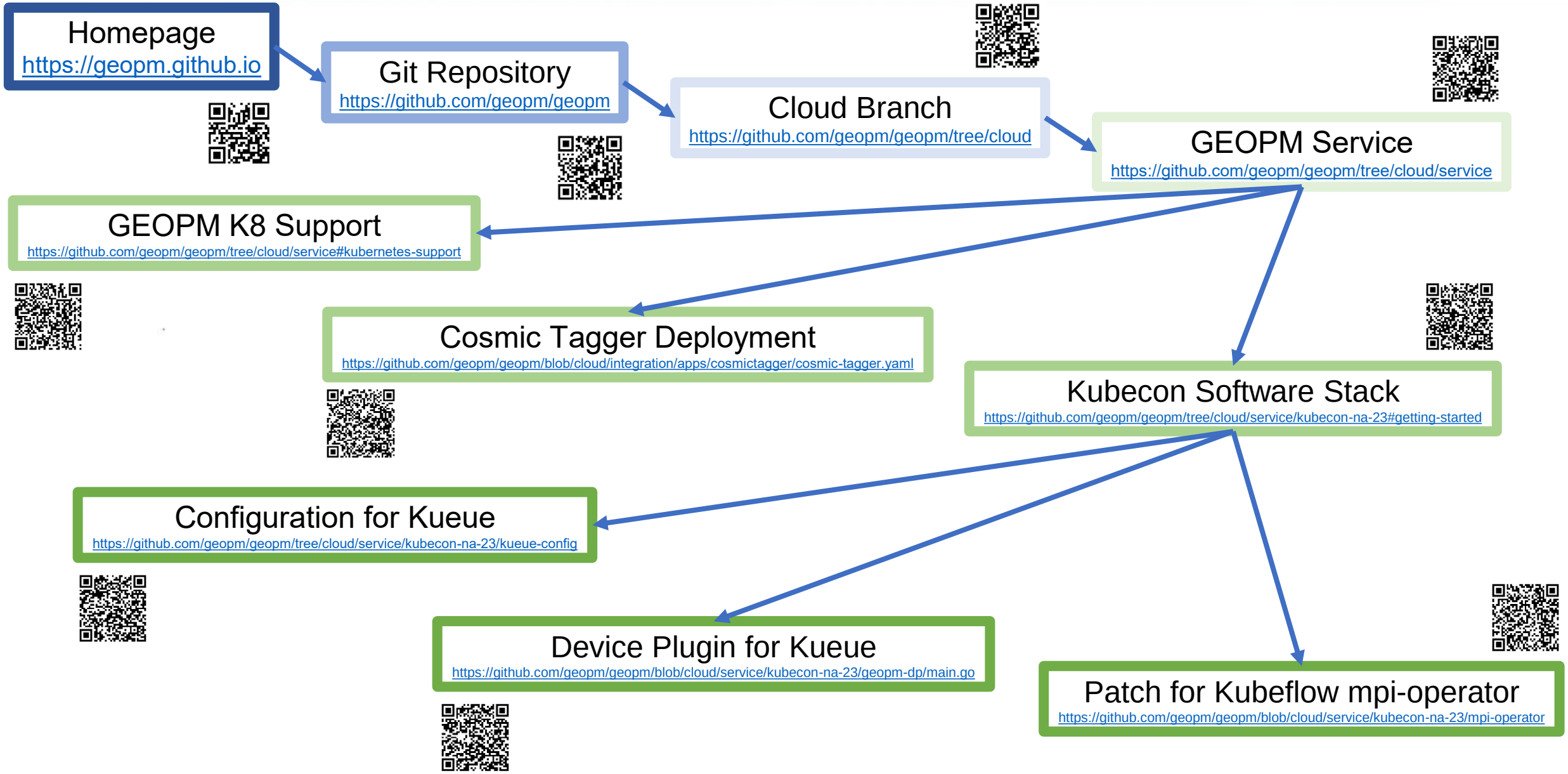
Keep geopm-client alive until process named mpirun terminates

With Sidecars:
Wrapper automatically ends with app

```
containers:
- name: cosmic-tagger
  command: ["mpirun", …]
  <…>
initContainers:
- name: geopm-client
  restartPolicy: Always
  command: ['/usr/bin/sh', '-c', 'geopmwrite …; sleep infinity;']
```

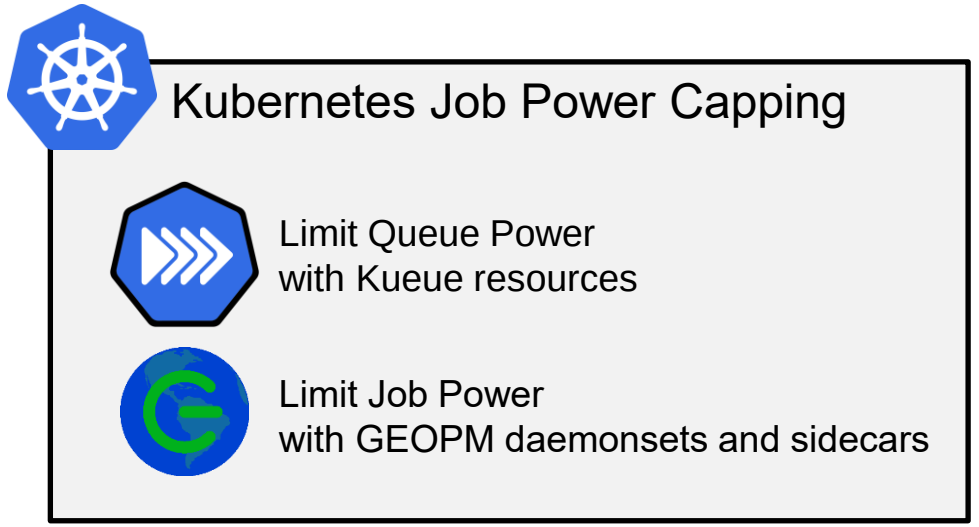
Keep geopm-client alive until main container finishes

GEOPM Site Map to Related Software



Closing Remarks

- AI workloads demand a lot of power
- Power-cap sensitivity varies by workload
- Kueue and Sidecar containers make it easy to use GEOPM for job-level power caps
- Future work should continue making power management broadly available across containerized jobs
- Experimental cloud branch of GEOPM provides a gRPC protocol supporting containers



More Information
Email: danielcw@bu.edu, atanas.atanasov@intel.com, & christopher.m.cantalupo@intel.com
Web: http://geopm.github.io
Software in this talk: https://github.com/geopm/geopm/tree/cloud/service/kubecon-na-23

Scan to leave feedback on this session