GEOPM for Power-Aware Scheduling

Overview
GEOPM (Global Extensible Open Power Manager) is a runtime and framework for energy-efficient HPC scheduling. It enables real-time telemetry collection, per-job power capping, and dynamic redistribution of power to balance performance and energy consumption.

What GEOPM Does
GEOPM reads hardware power and performance signals (via RAPL, MSRs, GPU interfaces) and adjusts controls dynamically. It can enforce node or job-level power caps, rebalance power among nodes to reduce stragglers, and integrate seamlessly with Slurm for HPC workloads.

Integration with Slurm
Jobs are launched using `geopmlaunch`, wrapping the standard Slurm `srun`. GEOPM agents control and monitor power usage, while reports and traces are collected for each job.

Example Command
```
geopmlaunch srun -N $NODES -n $TASKS --geopm-ctl=process --geopm-agent=power_balancer --geopm-policy=policy.json --geopm-report=report.txt --geopm-trace=trace.csv -- ./app
```

GEOPM manages CPU affinities and hooks into MPI and OpenMP layers via PMPI/OMPT interfaces.

GEOPM Agents
- **monitor**: Collects telemetry without control (baseline).
- **power_governor**: Applies a uniform per-node power cap.
- **power_balancer**: Dynamically shifts power between nodes to reduce tail latency.
- **frequency_balancer / frequency_map**: Controls CPU frequency scaling based on workload regions.

Policy Configuration
Policies define power caps or frequency targets for agents. Example:

```
geopmagent -a power_governor -p POWER_CAP=250 > policy.json
```

Runtime changes can be applied via the GEOPM endpoint API.

Setup Requirements
1. **MSR Access**: Load `msr` or `msr-safe` kernel modules.
2. **BIOS Configuration**: Enable RAPL and ensure MSR write access.
3. **Software Packages**:
- `geopm-runtime`: for runtime and CLI tools
- `geopm-service`: for privileged hardware access
4. **MPI Integration**: GEOPM integrates via PMPI wrappers.

Example Slurm Script
```
#!/bin/bash
#SBATCH -N 4
#SBATCH -t 00:30:00
#SBATCH -J cg_powercap

module load geopm
geopmagent -a power_balancer -p POWER_CAP=250 > policy.json

geopmlaunch srun -N ${SLURM_NNODES} -n ${SLURM_NTASKS} --geopm-ctl=process --geopm-agent=power_balancer
--geopm-policy=policy.json --geopm-report=geopm_${SLURM_JOB_ID}.txt --geopm-trace=geopm_${SLURM_JOB_ID}.csv -- ./cg.x
```

Monitoring and Telemetry
GEOPM exposes telemetry signals such as CPU/GPU power, frequencies, and temperature. Integration with NVIDIA DCGM and Intel Level Zero allows full-stack observability.

Scheduler Coordination Models
1. **User-driven model**: Users specify agent and power cap at job submission.
2. **Scheduler-budgeted model**: Slurm assigns job-level power budgets enforced by GEOPM.

Tuning and Optimization Steps
1. Run baseline with `monitor` agent.
2. Apply static caps with `power_governor`.
3. Optimize with `power_balancer` to redistribute watts and reduce imbalance.
4. Validate affinity settings and RAPL access.

Example Use Cases

- Power capping on large MPI workloads to reduce cluster-wide power spikes.
- Dynamic power redistribution to improve job throughput under constrained energy budgets.
- Integration with Grafana for visual telemetry dashboards.

Benefits

- Lower energy cost per simulation.
- Improved job predictability and fairness.
- Minimal scheduler modification required.
- Scalable control hierarchy suitable for large HPC systems.

Conclusion

GEOPM provides a flexible framework for implementing power-aware scheduling in HPC environments. Its modular agents, tight Slurm integration, and hardware-level telemetry support make it ideal for balancing performance and energy efficiency in modern supercomputing workloads.