Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 175–183
DOI: 10.15439/2025F0376
ISSN 2300-5963 ACSIS, Vol. 43

Slurm plugin for HPC operation with time-dependent cluster-wide power capping

Alexander Kammeyer*†, Florian Burger*, Daniel Lübbert* and Katinka Wolter†
0000-0002-7858-0354, 0000-0003-4745-5515, 0000-0003-3852-5665, 0000-0002-8630-0869
*Physikalisch-Technische Bundesanstalt, Abbestraße 2-12, 10587 Berlin, Germany
Email: {alexander.kammeyer, florian.burger, daniel.luebbert}@ptb.de
†Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
Email: {a.kammeyer,katinka.wolter}@fu-berlin.de

***Abstract*—HPC systems are shared between many users. Managing their resources and scheduling compute jobs is a central task of these clusters. Scheduling also allows to control the workload and energy consumption of an HPC system. A Digital Twin of an HPC cluster can aid in the scheduling process by providing energy measurements about the system and predict scheduling decisions with a simulation. For a real-world use case, an integration of the Digital Twin with the scheduler is necessary. A possible use case are energy limitations as part of a demand response process between the HPC operator and energy supplier.**

Therefore, this paper introduces a plugin for Slurm, an open-source scheduler, that implements a scheduling algorithm for time-dependent cluster-wide power capping. It uses a node energy model to predict the energy consumption of jobs and can start jobs at different frequencies to stay below the configured power limit. The plugin interfaces with the Digital Twin that provides energy measurements for the compute nodes to track the system power consumption in real time and update the power limitations if necessary.

The plugin is tested on a cluster and compared against a scheduling simulation of the algorithm. The analysis compares the power profile of the simulation and the real system and the allocation of the jobs over time. Differences in the execution and the power trace are analysed and discussed.

I. Introduction

THE High-Performance Computing (HPC) community has focussed on achieving maximum performance for a long time. This doctrine has recently come under pressure by rising energy prices and a transformation of the energy market towards renewable energies while HPC systems continue to grow in size and energy demand. A trend that is further accelerated by the recent shift towards artificial intelligence.

An HPC system under full load has a relatively constant energy requirement. However, a shift towards renewable energy production means greater volatility in the energy production. HPC systems need a mechanism to respond to these fluctuations. Solely relying on fossil energy source is also not an option, as regulatory frameworks, such as the Blue Angel for data centres [1], define upper bounds for carbon-dioxide emissions.

Dynamically adapting the energy consumption of an HPC system has further use cases. It can help stabilize the energy grid by reducing the load in the grid or by increasing the load if enough energy is available. Additionally, such a mechanism allows the continued automatic operation of the cluster in case of energy shortages.

Reacting to limited energy availability is sometimes referred to as demand response. While it is common practice for large energy consumers, HPC centres have not yet widely adopted such mechanisms [2]. However, first mechanism have been proposed. One possible demand response mechanism was presented in [3]. A Digital Twin is used to collect data about the system energy consumption and together with information about energy availability, the scheduler can adapt the overall energy consumption to stay below a defined threshold. The algorithm in the original paper was validated with a simulation. This paper implements the algorithm as a plugin for the popular cluster scheduler Slurm [4].

This paper makes the following contributions:

- The algorithm uses a node energy model to predict the energy consumption of the jobs. This paper uses a new test platform for the evaluation of the plugin. Thus, an updated node energy model is necessary that has been created for the new test system.
- The algorithm from the original facilitates time-dependent cluster-wide power capping. This paper implements this algorithm as a plugin for the open-source scheduler Slurm.
- The algorithm was previously only evaluated in a simulation of the Digital Twin. This paper compares the simulation against the performance on a real cluster. The evaluation includes power traces of the real system and the node allocation to the jobs.

The remainder of this paper is structured as follows: Section II introduces Slurm in detail and presents relevant literature, especially regarding cluster operation under a power limitation. Preparatory work such as the updated node energy model and a recap of the scheduling algorithm is presented in Section III. In Section IV, the implementation of the Slurm plugin is discussed. The plugin is then evaluated in the real world and compared to simulation results in Section V.

II. Related Work

An HPC system consists of many individual components and is typically shared among users for running their compute jobs. These systems thus need a resource manager to avoid

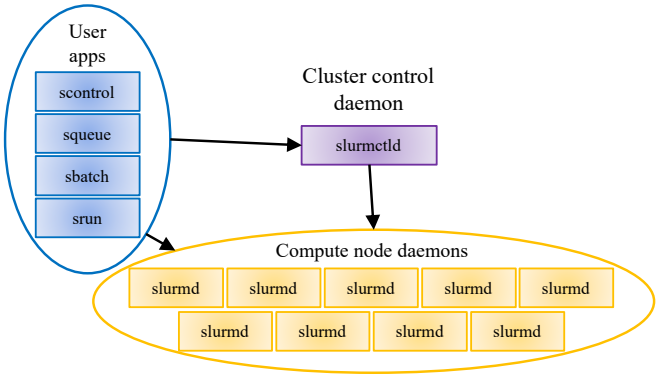


Figure 1: Overview of the Slurm components. The `slurmctld` is the central controller of the cluster while `slurmd` handles each compute node. Client components such as `scontrol`, `squeue`, `sbatch` and `srun` allow the user to interact with the cluster and submit and manage compute jobs.

resource conflicts between users as well as a strategy to assign the available resources to the requests of the users, referred to as scheduling strategy. Different resource managers and schedulers exist such as e.g. Moab, PBS, Grid Engine and Slurm [4].

Being open-source software written in the C programming language, Slurm has been widely adopted by the research community. Slurm works both as a resource manager and as scheduler and also starts and monitors the compute jobs on the cluster. Additionally, it is able to handle accounting of the resource usage and supports resource reservation in advance. The software consists of multiple components: the Slurm control daemon `slurmctld`, the Slurm daemon on the compute nodes `slurmd` and command-line utilities for the users to manage Slurm and submit (`sbatch`) and control (`scontrol`) their jobs. Slurm also handles the MPI configuration for the jobs and offers the launch program `srun`. These components are shown in Figure 1. Slurm has further components that are responsible for accounting and billing but they are not relevant in the context of this paper.

HPC systems are composed of individual compute nodes which might feature different hardware, e.g. CPU cores, memory and additional components such as storage, GPUs or TPUs. The nodes can be grouped into partitions for bundling nodes with identical hardware or for aiding the scheduling process, e.g. when certain nodes are reserved for short or long running jobs or similar. When users want to run a task on the HPC system, they submit a job. The job defines what the user wants to compute, the required resources to complete the computation and for how long the job will need those resources. Slurm can further divide jobs into job steps, that define tasks that may be executed in parallel or contain pre- and post-processing and do require different resources.

The specific scheduling algorithm and resource allocation can contribute to the energy efficiency and stability of the system and thus has been subject to many research studies. The surveys by Czarnul [5] and Kocot [6] provide an overview of tools for energy and power management and power-aware scheduling.

Simulating the scheduling process is a common approach in HPC research. One example is CQSim [7]. Based on trace data, the authors propose a scheduling approach for demand response, and test it in their simulation [8]. To retrieve such traces, the system needs to be actively monitoring the energy consumption of each job. Not every cluster is designed to do that because the hardware does not provide the necessary measurements, or because the scheduling environment does not track the data, or both. Another important aspect is taking the specifics of the different applications into account, when imposing a power cap on the cluster [9]. The scheduler may automatically change the frequencies and thus runtime of the jobs to dynamically distribute the power. Other demand response methods rely on power traces to create a node model and limit power consumption of the HPC clusters [10]. None of these approaches have been tested in the real world.

One approach in system management and system modelling has attracted increased interest over the last years: Digital Twins [11]. A Digital Twin is a virtual representation of a real-world object, in this context an HPC cluster. A real-world object is represented as a Digital Twin in the virtual world. Data about the object is collected and integrated into the Digital Twin. With their characteristic bi-directional data linkage, changes in the real world are reflected in the digital domain and vice versa. The Digital Twin models the behaviour in the real world and can test new parameters before they are applied to the real system. Simulations are often cheaper and faster and a negative impact on the real system can be avoided. In the HPC context, these simulations are generally scheduling simulations as they describe changes in the system state well.

Digital Twins are under development for HPC systems ranging from small clusters to one of the world's largest systems, Frontier [12]. As part of their smart campus initiative [13], the Physikalisch-Technische Bundesanstalt (PTB) is developing a Digital Twin for their HPC system [14]. It has been demonstrated that it can be used to reduce system emissions. One major use case of the Digital Twin is the operation of HPC systems under a time-dependent cluster-wide power cap. The power can be capped by the Digital Twin, depending on the currently available energy [3]. Precise energy measurements also allow the Digital Twin to track the data centre Power Usage Effectiveness (PUE) [15].

Similar to the node energy model in Section III-B, application power profiles are a common technique used in application modelling and have been done for a wide variety of applications, including WZ factorisation [16] and matrix factorisation [17]. An alternative to Dynamic Voltage and Frequency Scaling (DVFS) is the implementation of a power cap as a hardware driver [18].

III. Preliminary Work

This section introduces the test system used in this paper, and shows how Slurm has been configured for this specific system. Since the test system differs from the system in [3], a new node energy model has been created. Furthermore, a cap on the implemented scheduling algorithm is given.

A. Test System

For the practical test of the Slurm plugin, this paper relies on a cluster consisting of Raspberry Pis. While this setup is not typical for a commercial HPC system, it has a similar structure and allows the development of the plugin without interrupting the production HPC system of PTB. The setup is based upon a design found proposed in [19].

The system is equipped with a total of 10 Raspberry Pi 4B with 8 GB RAM each. They are connected via their $1\,\mathrm{Gbit/s}$ Ethernet ports. Each Pi also has the Power over Ethernet (PoE) HAT (Hardware Attached on Top) installed that allows the nodes to be supplied with power from the Ethernet switch. All run the Raspberry Pi OS Bookworm.

The switch, a Netgear GS316EPP, supplies power via Power over Ethernet (PoE) to the nodes and offers a gigabit interconnect. It has a rudimentary management interface that provides energy information for each of the ports and allows management and power cycling of individual ports. These features thus allow us to track the per-node energy usage and to turn nodes off and on when needed as the Raspberry Pi does not support Wake-On-LAN. The energy data is queried by a collector agent of the Digital Twin and provided via the Digital Twin database.

One node is designated as the head node while the remaining 9 nodes act as compute nodes. The head node provides a global file system via NFS to all compute nodes. For this purpose, a single SDD is connected via an SATA-to-USB adapter and provides the necessary storage space. It also contains scratch space that is shared between the nodes. The compute nodes themselves are disk-less and get their image via TFTP from the head node when they boot. An additional Ethernet-to-USB adapter establishes a connection to the outside world while a DHCP server provides the internal cluster network with addresses for the compute nodes. The head node uses NAT to forward the internet connection to the compute nodes. This is necessary because the Raspberry Pi does not have a real-time clock and the clocks of the compute nodes would be out of sync after a reboot.

As the scheduler and resource manager we have opted to use Slurm [4]. Being open source software, the Digital Twin can be integrated with the scheduler. Slurm is configured with a single partition spanning all 9 compute nodes. Since the cluster is not intended for production use with many users, only basic logging is configured.

The head node is running the `slurmctld` control daemon that manages the scheduling logic and interfaces with the Digital Twin. The compute nodes run the `slurmd` daemons that connect the nodes to the control daemon. The user connects to the head node and can use the Slurm utilities, e.g. `sbatch`, to control the cluster and submit batch jobs.

Slurm provides some power saving functionalities [20], most notably node suspension and resumption. The scheduler monitors the nodes. If a node has been idle for a certain amount of time, it is shut down. When the node is needed again, Slurm issues a restart. This is implemented by power-cycling the Ethernet port of the switch, resulting in the Raspberry Pi booting again. The Slurm daemon on the node then contacts the Slurm controller, signalling that the node is ready to receive the next job. For this experiment, Slurm has been configured to turn of a compute node after 10 minutes of inactivity.

B. Node Energy Model

The node energy model is an integral part of the scheduling algorithm. It allows the algorithm to estimate the energy consumption of a job. It is the part that makes the algorithm hardware specific. The remaining logic works independently from any underlying hardware characteristics. However, it needs to be updated if the algorithm should run on a different cluster with new workloads. Hence, this section presents the node energy model for the test system used in this paper. For this experiment, a total of 4 different benchmarks and parallel applications have been selected, namely the HPL and HPCG benchmarks as well as OpenFOAM and Geant4.

The High-Performance Linpack (HPL) [21] is used to create the TOP500 ranking of supercomputers. The benchmark is a linear equation solver and uses highly optimized vector instructions, thus serving as a kind of upper bound for the model. The second benchmark, the High Performance Conjugate Gradients (HPCG) [22], complements the HPL benchmark by adding more memory intensive operations.

The two benchmarks are supplemented by two numeric applications. The first, Open Field Operation And Manipulation (OpenFOAM) [23], is a computational fluid dynamics package and the second, Geometry and Tracking (Geant4) [24], is a Monte Carlo simulation toolkit.

To create the node energy model, each of the 4 jobs was run on the nodes on all supported frequencies. The switch provides an energy measurement that allows to monitor the jobs energy consumption. From these measurements, the Time-to-Solution (TtS) and Energy-to-Solution (EtS) can be derived. Figure 3 shows the results for all supported frequencies from $0.6\,\mathrm{GHz}$ to $1.8\,\mathrm{GHz}$. Pareto-optimal points [25], [26] are coloured in orange. For the model shown in Table I, three points have been selected: the highest frequency of $1.8\,\mathrm{GHz}$ and if available two pareto-optimal points. The scaling factor is used to adjust the job runtime for the increased runtime required if the job runs at a lower frequency setting.

This model shows a downside of the Raspberry Pis as a test platform. They have a relatively high offline energy consumption, especially compared to their idle consumption and the consumption under load [27]. This amounts to $22.1\,\mathrm{W}$ offline consumption for the compute nodes of the cluster. While $2.70\,\mathrm{W}$ is not much compared to other platforms, it is

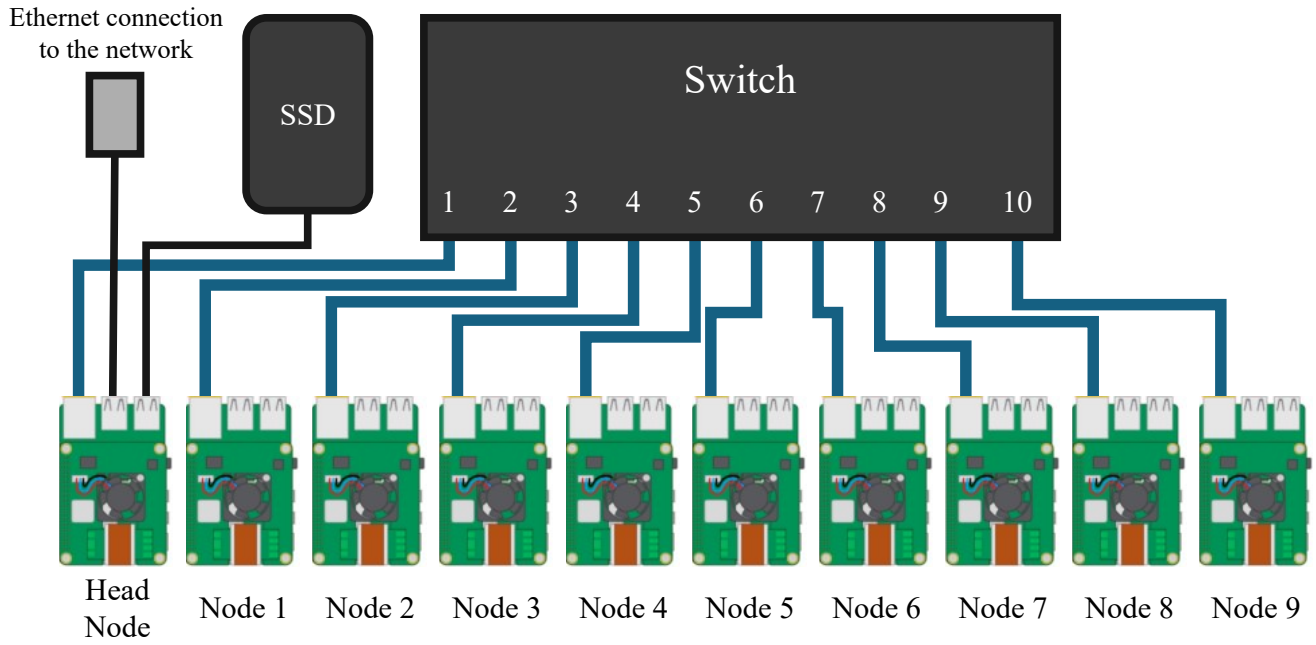


Figure 2: Schematic overview of the test cluster. It consists of one head node and a total of 9 compute nodes. They are connected via Ethernet and receive power via PoE. The head node has an additional Ethernet port to connect the cluster to a network. The SSD is also connected to the head node and provides storage for the compute nodes.

Table I: The node energy model

Job type	Frequency	Power	Scaling factor
HPL	1.8 GHz	6.20 W	1.000
	1.6 GHz	5.74 W	1.049
	1.5 GHz	5.47 W	1.055
HPCG	1.8 GHz	5.05 W	1.000
	1.3 GHz	4.25 W	0.911
	1.2 GHz	4.13 W	0.933
OpenFOAM	1.8 GHz	5.95 W	1.000
	1.4 GHz	5.17 W	1.015
	1.1 GHz	4.89 W	1.064
Geant4	1.8 GHz	6.62 W	1.000
	1.5 GHz	5.55 W	1.208
	1.1 GHz	4.92 W	1.625
Idle	-	3.12 W	-
Offline	-	2.70 W	-

almost half as high as the busy consumption. This behaviour can be changed by forcibly disabling PoE on the switch instead of only issuing a software shutdown.

C. Algorithm Recap

The goal is to run the HPC cluster under a variable power limitation while maximizing the system usage. The algorithm's pseudocode is shown in Algorithm 1. This section briefly describes the idea behind the algorithm while an in-depth description was given in [3]. The algorithm uses two main concepts to improve the throughput: DVFS, and turning off nodes. The node energy model provides the optimal frequency settings, while turning off unused compute nodes further decreases the system power consumption.

The scheduling algorithm receives a list of eligible jobs, and sorts them by time of arrival. The first "if" statement checks whether there are enough online and offline nodes available to start the job, and starts offline nodes if necessary.

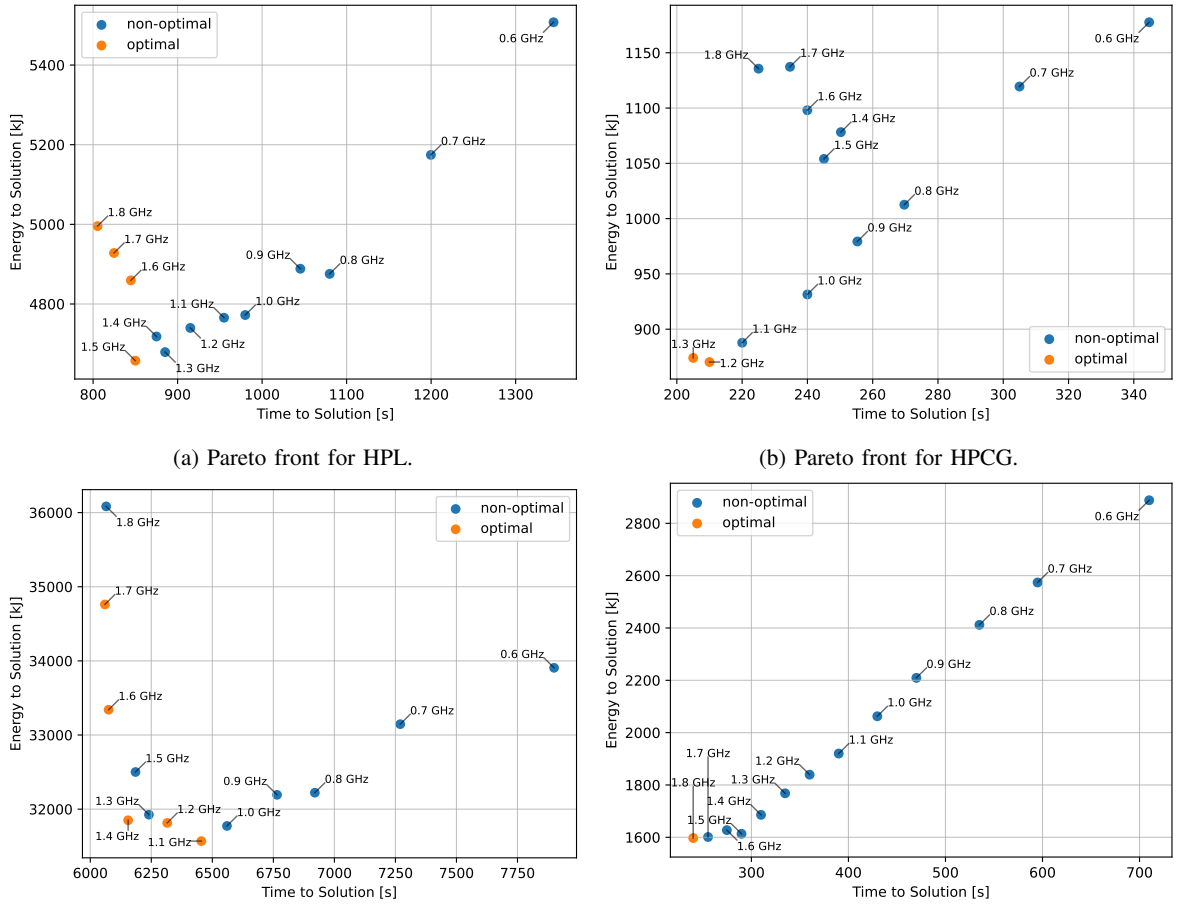
The second "if" statement checks whether enough online nodes are available to start the job. If that is the case, it uses the node energy model to estimate the job's power consumption. If starting the job would exceed the power limit, the job is deferred; otherwise, the job is started.

When a job is deferred, the routine moves on to the next job in the queue. This allows the algorithm to backfill jobs that might fit within the available resources.

IV. Slurm Plugin

The Slurm control daemon `slurmctld` is responsible for job scheduling. Two scheduling algorithms are included in the package as plugins. The first plugin is called `builtin` and implements a simple FCFS algorithm. The second plugin is called `backfill` and, as the name implies, implements FCFS with backfilling. This paper presents an additional, new scheduling plugin which implements the algorithm from Section III-C that can be used instead of the integrated plugins.

By default, Slurm schedules jobs automatically without consulting the scheduling plugin, as long as no resource conflicts arise. E.g. a job submitted to an empty cluster that fits in terms of compute nodes gets scheduled without the scheduling plugin. This circumvents the energy limit check of the scheduling plugin, and in our case could lead to the power consumption exceeding defined limits. To avoid such a



(a) Pareto front for HPL.

(b) Pareto front for HPCG.

(c) Pareto front for OpenFOAM.

(d) Pareto front for Geant4.

Figure 3: Pareto fronts for the four different applications in the node energy model. Pareto-optimal frequencies in terms of EtS and TtS are coloured orange. The remaining frequencies are coloured blue.

situation, all jobs must be submitted in the `hold` state with a priority of zero. This prevents Slurm from automatically scheduling jobs [28].

The Slurm control daemon periodically consults the scheduling plugin. The plugin first consults the database and updates the power limits and queries the current power consumption of the system from the database. It contains a set of functions that allow it to connect to the InfluxDB of the Digital Twin. InfluxDB does not offer a native C client implementation. However, since Slurm includes the cURL library, the plugin uses cURL to connect to the REST API of the database in order to query both the power consumption of the nodes and the power limitations. This provides a level of abstraction of the database calls from the rest of the code. Furthermore, helper functions convert the reply from csv to C-structs for further use.

During this periodic invocation, the plugin uses the core scheduling routine `_backfill_power()` which is shown in Algorithm 2. It does not use the usual scheduling queue but the entire job list. First, it checks if any jobs are pending but have not started yet in the first while loop (lines 13-21). Those are jobs that have been scheduled by the plugin but have not yet been started by Slurm. This avoids releasing to many jobs that later conflict with one another. If a job is waiting to start, the function is exited and waits for the periodic scheduling before making the next attempt.

The second while loop (starting in line 25) handles the actual scheduling. The plugin first consults `job_test_resv()` (lines 30.32), a Slurm function that tests whether enough nodes are available. Afterwards, `select_g_job_test()` (lines 31-49) tries to select the optimal nodes and tests whether the job can start immediately. Last but not least, `check_power_dvfs()` (lines 57-58) uses the node energy model to estimate the energy requirement of the job and whether the current power limitations allow the job to start. It is also responsible to set the frequency and adjust the runtime of the job according to the numbers in Table I. If enough energy is available, the priority of the job is increased which releases the job from the `hold` state. The `slurmctld` recognizes that and starts the job. It is also responsible to

Algorithm 1 Pseudocode of the core scheduling routine

```
backfill_power(eligible_jobs) {
  for (Job j : eligible_jobs) {

    // test if nodes need to be booted
    // check if enough nodes are offline
    if (j.nodes > online_nodes(j) &&
        j.nodes <= online_and_offline_nodes(j)) {
      // how many nodes need to be booted?
      toboot = j.nodes - online_nodes(j);

      // check if the nodes would
      // exceed the power limit
      if(!check_power(j, get_offline(j, toboot))){
        continue;
      }
      boot_nodes(j, toboot);
    }

    // test if enough nodes are online
    // and available
    if (j.nodes <= online_nodes(j)) {

      // check if the job and nodes would
      // exceed the power limit
      if (!check_power_dvfs(j)) {
        continue;
      }

      assign_nodes(j);
      j.wait_time = tick - j.submit_time;
      running.add(j);
      eligible_jobs.remove(j);

      // trigger an event in the simulation
      // on job completion
      e = new JobEvent(tick + j.run_time_scaled,
                j, JobState.COMPLETED);
      eventQueue.add(e);
    }
  }
}
```

select the nodes and start them if necessary. Slurm also sets the configured frequency.

So far, the implementation has been tested on a homogeneous cluster with a single partition. It does not yet consider multi-partition scheduling or other features such as multi-step jobs. Another aspect the algorithm cannot control yet is energy usage during the boot process of the nodes, which can lead to power spikes. One possible way to control this is by introducing a delay in the boot script that Slurm uses to power-cycle the ports on the switch. An early loaded power cap driver, as proposed in [18] might also help.

V. Evaluation

This paper presents a Slurm plugin based on a scheduling algorithm, evaluated previously only through simulation. To compare the simulated results with the real-world, both the simulation and the plugin are configured with the node energy model from Section III-B and an identical number of compute nodes.

The workload for the evaluation is generated with the improved Feitelson job model [29], [30]. The model generates a job trace in the Standard Workload Format (SWF) [31]. While the format encodes which application was run, the

Algorithm 2 Actual code: Core scheduling routine of the Slurm plugin

```
static void _backfill_power(void) {
  int j, rc = SLURM_SUCCESS, job_cnt = 0;
  job_record_t* job_ptr;
  bitstr_t *alloc_bitmap = NULL;
  bitstr_t *avail_bitmap = NULL;
  bool resv_overlap = false;
  resv_exc_t resv_exc = { 0 };
  time_t now = time(NULL);
  list_itr_t *iter = NULL;

  // check if a job has been scheduled
  // and has not started yet, defer further
  // scheduling in this case
  iter = list_iterator_create(job_list);
  while ((job_ptr = list_next(iter))) {
    if(IS_JOB_PENDING(job_ptr) == true
      && job_ptr->priority > 0) {
      list_iterator_destroy(iter);
      return;
    }
  }
  list_iterator_destroy(iter);

  // go through the list of held jobs
  iter = list_iterator_create(job_list);
  while ((job_ptr = list_next(iter))) {
    if(IS_JOB_PENDING(job_ptr) == true
      && job_ptr->priority == 0) {
      // Determine which nodes a job can use
      // based upon reservations
      j = job_test_resv(job_ptr, &now, true,
        &avail_bitmap, &resv_exc,
        &resv_overlap, false);
      if (j != SLURM_SUCCESS) {
        FREE_NULL_BITMAP(avail_bitmap);
        reservation_delete_resv_exc_parts(
          &resv_exc);
        continue;
      }
      // Select the "best" nodes for given job
      // from those available
      rc = select_g_job_test(job_ptr,
        avail_bitmap,
        job_ptr->details->min_nodes,
        job_ptr->details->max_nodes,
        job_ptr->details->max_nodes,
        SELECT_MODE_RUN_NOW,
        NULL, NULL,
        &resv_exc,
        NULL);
      FREE_NULL_BITMAP(avail_bitmap);
      reservation_delete_resv_exc_parts(&resv_exc);
      if (rc != SLURM_SUCCESS) {
        continue;
      }
      // We are certain that a job can start
      // check the power limit and adjust DVFS
      if(check_power_dvfs(job_ptr,
          get_req_nodes(job_ptr)) < 0) {
        continue;
      } else {
        // release the job
        // slurmctld will start the job
        job_ptr->priority = 1;
        job_ptr->state_reason = WAIT_NO_REASON;
        break;
      }
    }
  }
  list_iterator_destroy(iter);
}
```

Feitelson model does not generate this information. Hence, an equal distribution is used to generate the workload information with equal probability for each of the 4 different applications. Slurm only supports runtime limits with minute resolution for the job while the Feitelson model produces job lengths with seconds resolution. All job runtimes are therefore rounded up to the next full minute.

As a baseline, the job trace was simulated by the scheduling simulation component integrated in the Digital Twin. The power trace is shown in Figure 4a and the allocation of the jobs to the nodes in Figure 5a. In comparison, the same job trace was run with identical power limitations on the cluster. A script processed the job trace file and submitted the jobs accordingly. The power trace is shown in Figure 4b and the node allocation in Figure 5b.

In the simulation, the algorithm uses the gaps between the power limit while staying below the set limit. Larger jobs did not fit below the limit and were delayed beyond the limitation. This effect can also be seen in the node allocation in Figure 5a. The results are comparable to the first iteration of tests with the other node energy model [3].

In the experiment on the test system, looking at the power trace, the cluster mostly stays below the limit. Sometimes the limit is slightly exceeded in the order of $\leq 3\%$ which can be attributed to booting nodes and job initialization. In contrast to the simulated system where energy needed to start the nodes is not accounted for, the real system constantly tracks consumed energy also while booting additional nodes. When the operating system and Slurm are fully loaded, controlling the energy use is rather simple. However, especially during the early boot phase, the system draws a lot of power while not offering a mechanism to control it.

Another difference is that larger jobs start earlier in the real system test. This increases the system utilization and reduces the time required to complete the job trace. This again hints towards slight differences in the energy calculation between the plugin and the simulation. While the simulation considers the idle energy usage of the nodes, the plugin considers the energy usage of all running jobs and assumes that unused nodes are shut down by Slurm. The plugin only checks whether the running jobs allow an additional job to start. Furthermore, the node energy model upon which in both cases scheduling decisions rely on, represents the in average power consumption over time of jobs - not their maximum values. Therefore it is clear, that the real system will exhibit these maxima in power consumption by slightly surpassing the power limit. This could of course be cured in future by a more refined and time-resolved node energy model.

The time the scheduling and start of the job actually takes makes up a third difference between the simulation and the real world. In the simulated run, jobs are tightly packed as the simulation starts jobs instantly. In contrast, the Slurm plugin first releases them from the hold state and subsequently waits for the job to actually start before making the next scheduling decision. This causes a slight delay, thus creating small gaps between the jobs visible in Figure 5b.

To summarize, this experiment has demonstrated that the power-capping algorithm is feasible on a real system via a Slurm plugin, with a few initial limitations. The next step in the development will be to align the real-world and the simulation more closely with one another. The Slurm logic is complex and hard to replicate in a pure simulation. The simulation also does not simulate everything. This includes accurate power models for the boot process, and time required to start and stop jobs on the nodes.

The test system only has a single node type and has one partition configured. The plugin remains to be tested in a heterogeneous system configuration: Slurm allows to configure different hardware in a single partition, which will make the check more complex. Our plugin does not yet consider shared node allocation, but this can be complemented in a next step, as it is supported by Slurm.

In the current implementation, the node energy model is hard-coded in the plugin, which reduces portability of the code. For a broader use, the parameters will be moved to a configuration file. The accuracy of the algorithm depends on the accuracy of the node energy model. So far, no changes in the power consumption of the jobs have been considered. Additionally, the model is static and does not adapt to new workloads. Another possible improvement for the plugin are malleable jobs, which can be extended in their runtime when they are switched to a lower frequency during their runtime.

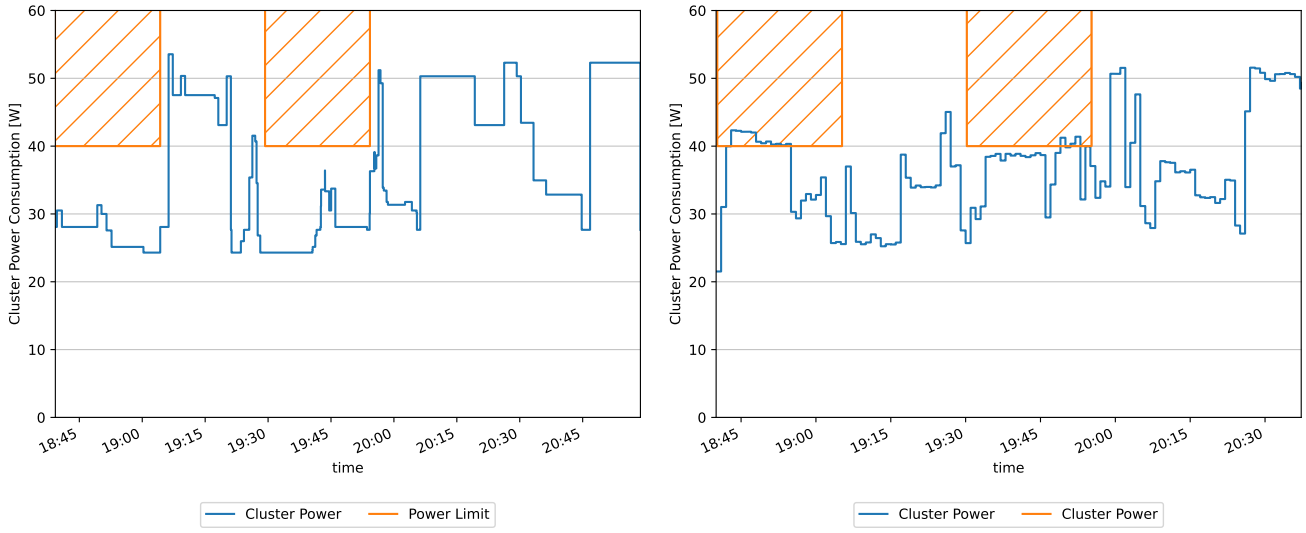
VI. Conclusion and Future Work

This paper presented a new scheduling plugin for the widely used scheduling and resource management software Slurm. The plugin allows an HPC cluster to operate under a system-wide power cap. The plugin works by holding jobs until enough power is available. Jobs can be configured by the plugin with a specific frequency according to the node energy model to reduce energy consumption. If the budget allows, the plugin releases the job from the hold state and sets the frequency. The Slurm infrastructure then handles the job start and node configuration.

The scheduling algorithm uses a node energy model to estimate the energy consumption of a compute job. For this paper, a new model has been created for the specific hardware of the test system. It has been demonstrated that this approach can be adapted to different systems, making the scheduling algorithm portable.

The plugin was demonstrated on the test system against a simulation with an identical job trace. Differences between the simulation and plugin have been discussed, especially job start times and energy calculation. While the plugin was able to stay below the power limit, booting compute nodes might briefly surpass the configured limit. Possible fixes have been discussed. The plugin was able to schedule jobs efficiently, and shows higher utilization than the simulation.

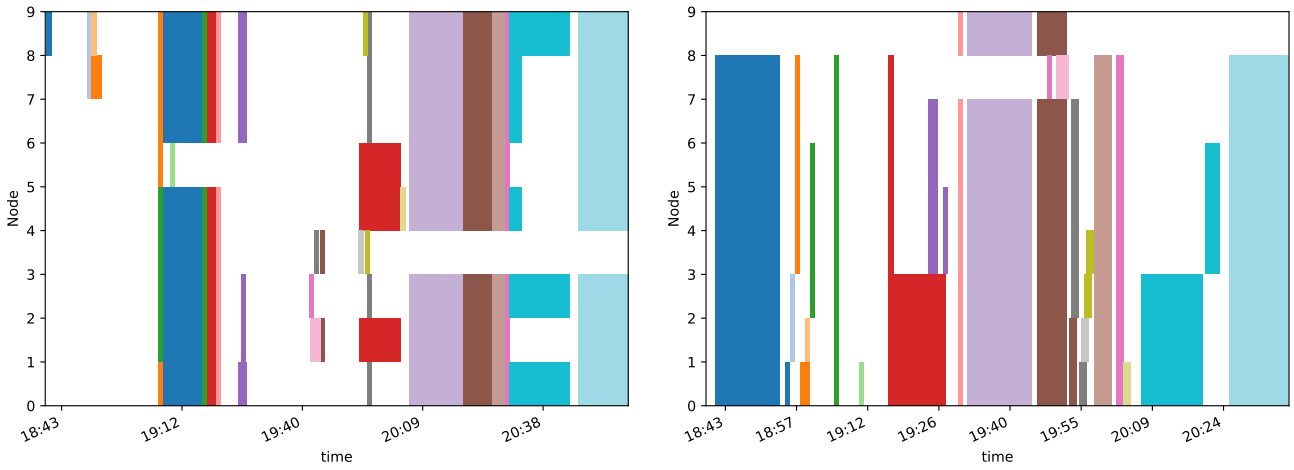
The Slurm plugin presented in this paper was tested with a homogeneous system configuration with identical nodes in a first step. Modern HPC systems often use heterogeneous system configurations with different node types and different



(a) Power trace of the simulation by the Digital Twin.

(b) Power trace of the test cluster.

Figure 4: Power traces of the experiment. A job trace was simulated and run on the test cluster to compare the performance of the algorithm in the simulation and the real world. The power trace is shown in blue and the limit is shown as the orange dashed surface.



(a) Node allocation of the simulation.

(b) Node allocation of the real system.

Figure 5: Node allocation of the experiments. The width of the rectangles corresponds to the runtime and the height shows which nodes where allocated to the job.

generations of CPUs. The test system could be extended with newer Raspberry Pi 5s or NVIDIA Jetson boards to create such a heterogeneous system.

Currently, the node energy model is hard-coded in the plugin. This would need to be shifted to a configuration file. The plugin has not been tested with heterogeneous or multi-partition cluster configurations. Since these are common nowadays, further development is necessary to support these configurations.

ACKNOWLEDGEMENT

We would like to thank our colleagues from department Q.13, first and foremost S. Thümmler, for their help with building the case for the Raspberry Pi cluster.

REFERENCES

[1] RAL UMWELT, *Rechenzentren DE-UZ 228*, 2nd ed., Fränkische Straße 7, 53229 Bonn, Feb. 2025. [Online]. Available: https://produktinfo.blauer-engel.de/uploads/criteriafile/de/DE-UZ-228-280225-de-Kriterien-V3.pdf

[2] T. Patki, N. Bates, G. Ghatikar, A. Clausen, S. Klingert, G. Abdulla, and M. Sheikhalishahi, "Supercomputing centers and electricity service providers: A geographically distributed perspective on demand management in europe and the united states," in *High Performance Computing*, J. M. Kunkel, P. Balaji, and J. Dongarra, Eds. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-41321-1_13. ISBN 978-3-319-41321-1 p. 243–260.

[3] A. Kammeyer, F. Burger, D. Lübbert, and K. Wolter, "HPC operation with time-dependent cluster-wide power capping," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 39, 2024. doi: 10.15439/2024F1066 p. 385–393.

[4] M. A. Jette and T. Wickberg, "Architecture of the slurm workload manager," in *Job Scheduling Strategies for Parallel Processing*, D. Klusáček, J. Corbalán, and G. P. Rodrigo, Eds. Cham: Springer Nature Switzerland, 2023. ISBN 978-3-031-43943-8 p. 3–23.

[5] P. Czarnul, J. Proficz, and A. Krzywaniak, "Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments," *Scientific Programming*, vol. 2019, p. 8348791, 2019. doi: 10.1155/2019/8348791. [Online]. Available: https://doi.org/10.1155/2019/8348791

[6] B. Kocot, P. Czarnul, and J. Proficz, "Energy-aware scheduling for high-performance computing systems: A survey," *Energies*, vol. 16, no. 2, 2023. doi: 10.3390/en16020890. [Online]. Available: https://www.mdpi.com/1996-1073/16/2/890

[7] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for hpc systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: Association for Computing Machinery, 2013. doi: 10.1145/2503210.2503264. ISBN 9781450323789. [Online]. Available: https://doi.org/10.1145/2503210.2503264

[8] S. Wallace, X. Yang, V. Vishwanath, W. E. Allcock, S. Coghlan, M. E. Papka, and Z. Lan, "A data driven scheduling approach for power management on hpc systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. IEEE Press, 2016. doi: 10.5555/3014904.3014979. ISBN 9781467388153

[9] D. Bodas, J. Song, M. Rajappa, and A. Hoffman, "Simple power-aware scheduler to limit power consumption by hpc system within a budget," in *2014 Energy Efficient Supercomputing Workshop*, 2014. doi: 10.1109/E2SC.2014.8 p. 21–30.

[10] K. Ahmed, J. Liu, and K. Yoshii, "Enabling demand response for hpc systems through power capping and node scaling," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2018. doi: 10.1109/HPCC/SmartCity/DSS.2018.00133 p. 789–796.

[11] ISO Central Secretary, "Digital twin – concepts and terminology," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 30173:2023, Nov. 2023. [Online]. Available: https://www.iso.org/standard/81442.html

[12] W. Brewer, M. Maiterth, V. Kumar, R. Wojda, S. Bouknight, J. Hines, W. Shin, S. Greenwood, D. Grant, W. Williams, and F. Wang, "A digital twin framework for liquid-cooled supercomputers as demonstrated at exascale," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024. doi: 10.1109/SC41406.2024.00029 p. 1–18.

[13] B. Jung, A. Kammeyer, V. Peltason, M. Ulbig, M. Wehming, and D. Hutzschenreuter, "Systems metrology in future cities – the example smart metrology campus (smc)," *Measurement: Sensors*, p. 101800, 2024. doi: 10.1016/j.measen.2024.101800. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917424007761

[14] A. Kammeyer, F. Burger, D. Lübbert, and K. Wolter, "Developing a digital twin to measure and optimise hpc efficiency," *Measurement: Sensors*, p. 101481, 2024. doi: 10.1016/j.measen.2024.101481. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917424004574

[15] ——, "Determining data centre pue with a digital twin," in *Sensor and Measurement Science International*, ser. SMSI 2025. AMA Service GmbH, May 2025. doi: 10.5162/SMSI2025/A7.1 p. 71–72.

[16] B. Bylina, J. Bylina, and M. Piekarz, "Impact of processor frequency scaling on performance and energy consumption for wz factorization on multicore architecture," in *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 35, 2023. doi: 10.15439/2023F6213 p. 377–383.

[17] B. Bylina and M. Piekarz, "The scalability in terms of the time and the energy for several matrix factorizations on a multicore machine," in *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 35, 2023. doi: 10.15439/2023F3506 p. 895–900.

[18] A. Krzywaniak, J. Proficz, and P. Czarnul, "Analyzing energy/performance trade-offs with power capping for parallel applications on modern multi and many core processors," in *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2018. doi: 10.15439/2018F177 p. 339–346.

[19] Raspberry Pi Ltd., "How to build a Raspberry Pi cluster," May 2025. [Online]. Available: https://www.raspberrypi.com/tutorials/cluster-raspberry-pi-tutorial/

[20] SchedMD LLC, "Slurm power saving guide," May 2025. [Online]. Available: https://slurm.schedmd.com/power_save.html

[21] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," Dec. 2018, version 2.3. [Online]. Available: https://www.netlib.org/benchmark/hpl/

[22] J. Dongarra, M. A. Heroux, and P. Luszczek, "High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems," *The International Journal of High Performance Computing Applications*, vol. 30, no. 1, p. 3–10, 2016. doi: 10.1177/1094342015593158. [Online]. Available: https://doi.org/10.1177/1094342015593158

[23] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computer in Physics*, vol. 12, no. 6, p. 620–631, 11 1998. doi: 10.1063/1.168744. [Online]. Available: https://doi.org/10.1063/1.168744

[24] S. Agostinelli, J. Allison, K. Amako *et al.*, "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, p. 250–303, 2003. doi: 10.1016/S0168-9002(03)01368-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203013688

[25] N. Sudermann-Merx, *Fortgeschrittene Modellierungstechniken*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2023, p. 161–193. ISBN 978-3-662-67381-2. [Online]. Available: https://doi.org/10.1007/978-3-662-67381-2_7

[26] D. Kolossa and G. Grübel, "Evolutionary computation and nonlinear programming in multi-model-robust control design," in *Real-World Applications of Evolutionary Computing*, S. Cagnoni, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. ISBN 978-3-540-45561-5 p. 147–157.

[27] "Rpi 4 consumes 2.5w when shut down," May 2025. [Online]. Available: https://raspberrypi.stackexchange.com/questions/104944/rpi-4-consumes-2-5w-when-shut-down

[28] R. P. Becker, "Entwurf und implementierung eines plugins für slurm zum planungsbasierten scheduling," Bachelor's thesis, Freie Universität Berlin, Berlin, 2021.

[29] D. G. Feitelson, "Packing schemes for gang scheduling," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. ISBN 978-3-540-70710-3 p. 89–110.

[30] D. G. Feitelson and M. A. Jettee, "Improved utilization and responsiveness with gang scheduling," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. ISBN 978-3-540-69599-8 p. 238–261.

[31] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, "Benchmarks and standards for the evaluation of parallel job schedulers," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. doi: 10.1007/3-540-47954-6_4. ISBN 978-3-540-47954-3 p. 67–90.