Improving Supercomputer Usage with Aging Awareness

Robin Boëzennec, Fanny Dufossé, Guillaume Pallez, Alix Tremodeux

Improving Supercomputer Usage with Aging Awareness

Robin Boezennec
INRIA Rennes
Rennes, France
robin.boezennec@inria.fr

Fanny Dufossé
Université Grenoble Alpes, INRIA, CNRS, Grenoble INP
Grenoble, France
fanny.dufosse@gmail.com

Guillaume Pallez
INRIA Rennes
Rennes, France
guillaume.pallez@inria.fr

Alix Tremodeux
ENS Lyon
Lyon, France
alix.tremodeux@irisa.fr

ABSTRACT

Lifetime of electronic devices has a critical impact on their environmental footprint. In addition, the high-demand by AI companies of GPU has reduced tremendously their availability for supercomputing centers. Consequently, improving the duration of CPUs and GPUs is becoming a major issue in High Performance Computing (HPC) domain.

This paper investigates the optimization of a machine usage before a fatal failure and the trade-offs with performance. The lifetime of computing devices is strongly connected with the temperature and thus with the running frequency. We investigate the node frequency reconfiguration to optimize HPC usage. We estimate the benefit of a dedicated scheduling algorithm compared with a constant frequency.

We show that a correct decision can increase considerably the number of FLOP of a machine with a trade-off in terms of performance. Because aging models are currently inaccurate, we consider different models and discuss the robustness of our algorithms to inaccuracy.

KEYWORDS

Ressource management, Hardware aging, Frequency scaling, Sobriety

ACM Reference Format:
Robin Boezennec, Fanny Dufossé, Guillaume Pallez, and Alix Tremodeux. 2025. Improving Supercomputer Usage with Aging Awareness. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3731599.3767561

1 INTRODUCTION

In light of the human-caused climate catastrophe, the High Performance Computing (HPC) community has investigated its environmental impact for decades. Most of this analysis has focused on the energy efficiency and electricity consumption of machines. For instance, TOP500 has provided since 2014 the power consumption of some machines with a list of the HPC machines ranked by FLOPS per watt.

However, electricity carbon footprint is only one part of the impact of HPC. A large part of carbon emission due to HPC comes from the manufacturing of electronic components. This impact is increased by the short lifespan of HPC machines. We can observe that the 50 first machines of TOP500 stay in the list during 3 years in average (with an increase close to 6 years after 2010). In November 2024, only 3 machines of the 50 first machines of Top500 where more than 4 years old. In proportion, this impact may be even more considerable in a future where datacenters are supplied by renewable or low-carbon energy [16].

Similarly the lifespan of components in datacenters is short. Most of Cloud and HPC machines have a replacement cycle of servers of 3 to 5 years only [15], growing to 7 for HPC machines. With the constant shrinking of transistors, the effects of semiconductor aging are taking an increasing importance, mostly related to the lifespan of components [2, 17, 20, 27, 33]. Additionally, GPU procurement gets harder as the demand increases due to current (generative) AI demand and geopolitical situation.

All these reasons strengthen the criticality of rethinking our usage of HPC components: how can we get more computation out of them and at what cost? Note that this is a more complicated question than *simply* increasing their lifespan where the simplest solution would be to turn them off. We present visually this question, with different scenarios on the lifetime of Mira HPC machine in Figure 1a and with the slope in Figure 1b showing the trade-off between the speed at which computation is performed showing how many FLOPs one can get from an HPC machine over its life.

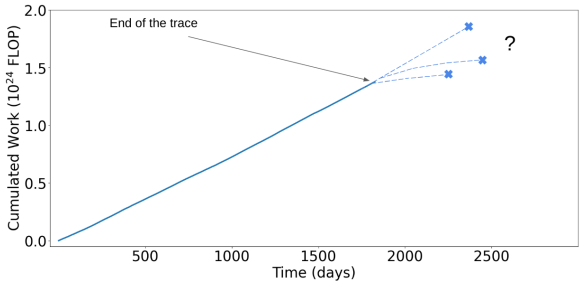
The bibliography on aging mechanisms on computing devices highlights the negative impact of temperature [2, 23, 27]. This paper investigates the impact of the frequency of computing nodes on the lifespan of components. Increasing the frequency of a component accelerates computations but also increases the temperature and thus reduces the lifespan of the node. Our objective is thus to increase the amount of workload completed by the machine over its lifetime.

As detailed in Section 2, we do not provide an exact model for how a machine degrades. This is a very open and active field of research. We make the two following contributions:

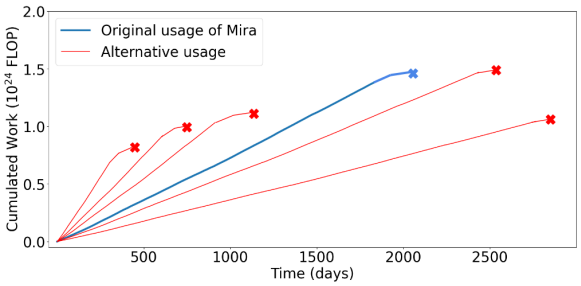
- Given an aging model for computing devices, we demonstrate that it is possible to modulate execution frequencies to increase considerably the global yield of the machine (i.e. the number of FLOP that the machine performs over its lifetime).

SC Workshops '25, November 16–21, 2025, St Louis, MO, USA

ACM ISBN 979-8-4007-1871-7/2025/11
https://doi.org/10.1145/3731599.3767561



(a) Number of FLOP done on Mira during its lifetime, and possible scenarios if we had extended its lifetime. In the optimistic scenarios, nothing changes, while in the realistic scenario, fatal failures are appearing which reduce the computing capacity of the machine over time.



(b) If Mira had been used differently, could we have increased the number of FLOP over its lifetime?

Figure 1: The way we use a machine impacts the number of FLOP that we can get from this machine. This discussion is the challenge of this work.

This can come at a trade-off in terms of average frequency usage.

- We show that our algorithm is robust to inaccuracy in the aging model, making its usage more realistic since predicting precisely the lifetime of computing resources is yet to be solved.

To evaluate our solutions, we instantiate with real data within the limits of what is known. In addition, we study a variety of aging models to test the limits of considering aging. The gains that we obtain on our experiments can be considerable: for some models close to those of the literature, we show that we can increase the yield of a machine over its lifetime by 15%. Compared to a solution that stresses the machine, we can increase by more than 25% the volume of computation done on a machine over its lifetime, for a global smaller than 20%. We also show that if the impact of aging is limited, then selecting the *right* frequency can be enough, without needing complicated algorithms. Other factors impact this benefit, for instance the difference in terms of cooling between the various nodes.

The rest of the paper is organized as follows: in Section 2 we present an overview of state-of-the-art aging mechanisms and scheduling strategies taking them into account. Then in Section 3 we introduce formally the scheduling problem before providing some theoretical results in Section 4. These theoretical results are used to design several novel algorithms in Section 5, which are then evaluated in Section 6. Finally we provide concluding remarks in Section 7.

2 RELATED WORK: AN OVERVIEW OF AGING MECHANISMS AND RESPONSES

Aging mechanisms of compute resources is an extremely complex phenomenon that has been studied but which is still far from being fully understood. Several work have different interpretations of aging mechanisms, some which may be contradicting. In this section we try to present an overview of several aging mechanisms, as well as solutions that were proposed. In addition, even if it were understood, at this stage there are little solutions offered to measure precisely the state of a resource [32].

There are a lot of mechanisms by which semiconductor deteriorates. The mains mechanism are Negative Bias Temperature Instability (NBTI), Positive Bias Temperature Instability, Hot Carrier Induced Degradation, Stress Migration, ElectroMigration and Time Dependent Dielectric Breakdown [2, 23, 27]. These deteriorations manifest themselves through two effects:

(1) A performance degradation which can be expressed either as a decrease of transistors speed (and thus operations speed), or as an increase of transient failure (i.e. non-fatal failures). We discuss this in Section 2.1.
(2) An increase in the critical/fatal failure probability, which is the core topic of this work. We discuss this in Section 2.2.

Although the question is orthogonal to this study, we present in Section 2.3 some works related to scheduling with consequences of aging-degradation.

2.1 Performance degradation

There are two main types of performance degradation. One of the most studied which is worth mentioning here is the *evolution of transient (i.e. non fatal) failures* [7, 9]. There is a tremendous amount of literature that explains how to deal with non fatal failures, with runtime techniques such as checkpointing or replication [9]. Note that while most of these strategies are based on failure models that consider that inter-arrival time of failures are independent random variable, hence independent of aging, it is possible to adapt them by updating the mean time to (non-fatal) failure [26]. These non fatal failures are orthogonal to this work, since dealing with them can be done jointly with the techniques that we propose here.

The second performance degradation studied in the literature is *hardware slowdown*. Aging causes changes in the *threshold voltage* of transistors which triggers additional delays for the current going through the transistor. This leads to an increased probability of *timing violation*. To avoid such error, the clock time can be increased [31] which reduces the frequency hence the performance.

The impact of aging on performance widely differs depending on the source from the literature. For example, while Amrouch et al. [2] found 1% of performance degradation after 10 years of aging, Masrur et al. [17] obtain 13% of performance degradation over the same period. In front of this uncertainty, we consider the losses of performance due to aging to be negligible and out of scope of this work.

2.2 Critical/Fatal Failures

The other consequence of semiconductor aging is the increase in critical failure probability. Following the literature [12, 22], we use the Mean Time To Failure (MTTF) as the variable that describes the impact of resource usage on the lifespan of the resource. Various works [23, 27, 28, 33] use the notion of mean time to failure with formulas linked to different aging mechanisms. However, the described formulas are not always the same, as the mean time to failure linked to the NBTI effect varies between [27] and [33]. To combine the different aging mechanisms, Srinivasan et al. proposed the RAMP model [29]. It consists in adding the failure rates. However, as [2] underlines, it makes the (unrealistic) hypothesis that the aging mechanisms are independent, which leads to estimation that were contradicted by practical measurements [14, 24].

In practice, Negative Bias Temperature Instability has been described as one of the key aging mechanisms [1, 25, 32], particularly as it impacts the MTTF, hence we focus on this in the remainder of this section.

A note on transient failures. Reliability has always been a key concern in HPC [9]. There is a tremendous amount of work focusing how parts of supercomputers often fail [11], and strategies to deal with checkpoint while they are rejuvenated [7]. Similarly, other work have considered the impact of silent errors [5] on computations. These works are different from what we study here: we are interested by the case where the machine cannot be rejuvenated. We do not study mechanisms to deal with the impact of failures/restart on a given computation, but how much computation can we perform until we cannot reuse the machine.

2.3 Degradation-aware scheduling

Several studies use the performance degradation to schedule jobs onto resources. While they are orthogonal to our work, we find that it is insightful to discuss them here. Note that most of the work analyzing the impact of node degradation come from the real-time community [17, 22, 30], for whom these issues are key because of deadline constraints.

One notable exception is the work by Zimmer et al. [35]. In 2015, Titan supercomputer had to replace a large number of GPU. In reaction to this, OLCF decided to update the job scheduling strategy to run larger jobs on more reliable nodes and smaller jobs on less reliable nodes, which allowed to maintain productivity at reduced capability.

Sun et al. [30] discussed the fact that some nodes may need a period of rest after an intense computation. They make the assumption that node do not deteriorate at long term, but may experience some slowdown right after getting over-stressed. With that respect, they developed a method to assess whether a node was stressed, as well as scheduling strategy to relieve stressed nodes whenever possible. Masrur et al. [17] discuss how aging incurs more soft errors, which slows down the performance of processors. They provide a schedulability analysis framework to take into accounts such aging-induced degradation, with the goal of providing concrete guarantees about full operationality of a machine (i.e. the period of time until which the machine meets all deadlines), which they call the lifetime of a machine. Finally, a series of work such as that of Pang et al. [22] discuss the scheduling of a task graph on resources with different MTTF, scheduling critical tasks on resources whose MTTF is the longest.

A note on frequency scaling. Dynamic Voltage and Frequency Scaling (DVFS) has been used in the past as a way to reduce the energy consumption of a machine [4]. Its impact on reliability has also been studied as it was believed to increase the number of transient failures when the reduction was too important, by merging fault-tolerance technique and DVFS to minimize the energy consumption [6]. The contributions of this work are quite different: we are not interested by the power consumption of the machine but rather by the procurement impact. We only consider how machine frequencies impact premature aging of the machine, and how we can use this to maximize the volume of compute performed by the machine. Our approach is closer to the series of work [3, 8] who have used DVFS to maximize the work done under a maximum temperature constraint.

3 MODEL

In this section, we model the problem of the impact of aging HPC machines. If we consider that compute nodes age and die depending on how they are used, we study the optimization of the work done on the machine during its lifetime. The challenge revolves around modeling aging.

3.1 Architecture

3.1.1 Platform overview. We consider an HPC cluster composed of M compute nodes. These nodes can be either CPU or GPU. For the rest of this section we consider that they have the same type (homogeneity). We are interested in the evolution of the platform when nodes age and ultimately die.

Node aging depends in its largest part on the temperature it runs on [23, 27], which in turn depends on the execution frequency. To regulate the aging function, we consider that we can change the *thermal monitoring function* of a node, to change the moment where it activates itself. The thermal monitoring function is a function that reduces the clock of a GPU when the resource gets over a certain temperature. For simplicity, we consider that we can set a *maximum execution frequency* of a node, i.e. an upper-bound on the frequency at which a node can run.

In general this initial maximum execution frequency is configured to guarantee a certain lifespan for the compute node. We consider that it is not interesting to change the frequency of a node during the execution of an application.

In the rest of this work, we use two main notations to describe aging:

- R_t: The State of Degradation (SoD) of a node, is a measure between 0 and 1, that describes how degraded the machine is. $R_t = 1$ means that the machine is new; $R_t = 0$ means that the machine is not functioning anymore. In critical computing, several tools allow to monitor the degradation status of critical components by providing more or less precise estimation of their Remaining Useful Lifetime [18].
- $\texttt{MTTF}(R_t, f, \zeta)$: The Mean Time To Failure (MTTF) of a node is the life expectancy of the node at a given time if run at constant frequency f. It takes into input its Remaining Useful Lifetime R_t, the frequency f at which we expect the node to

run in the future, and environmental variables ζ such as the external temperature and cooling factor. We only consider critical failures in this work (i.e. failures that render nodes definitely inoperable).

We have the following relations between those two notations:

$$R_t = \frac{\texttt{MTTF}(R_t, 0, \zeta)}{\texttt{MTTF}(1, 0, \zeta)} \tag{1}$$

or said differently, here, the SoD corresponds to the ratio between the node's expected MTTF when idle, and its expected idle lifespan when it was new.

Property 1. *Given a machine with a SoD of R_0 and an execution speed of f, then after t units of time, the new SoD R_t of the machine becomes*

$$R_t = R_0 - \frac{t}{\texttt{MTTF}(1, f, \zeta)}$$

Proof. We have $\texttt{MTTF}(R_t, 0, \zeta) = \texttt{MTTF}(R_0, 0, \zeta) - t$.

With (1), we can replace each $\texttt{MTTF}(R_X, 0, \zeta)$ by R_X and $\texttt{MTTF}(1, 0, \zeta)$, and deduce the final formula. □

Definition 1. *At a given time t_0, a machine of M nodes can be defined using the variables $\mathcal{P}_M = \left(\zeta_i, R^i_{t_0}\right)_{i \le M}$, where for each nodes $i \le M$:*

- *ζ_i: node parameters (including environmental parameters such as cooling factor);*
- *$R^i_{t_0}$: its state of degradation at t_0.*

3.1.2 Comments on aging models (MTTF). The literature is not consistent in terms of aging models [27, 32–34]. This is still an active field of research. There are however some relevant hypothesis on aging models that we discuss.

Work function. Consider a machine of one single node, running at frequency f. Then the expectation of the work done on this machine (i.e. before it fatally fails) is:

$$\mathcal{W}(f) = f \cdot \texttt{MTTF}(R_t, f, \zeta). \tag{2}$$

Studying the amount of work as a function of the frequency corresponds to studying the trade-off between the speed at which we perform work, and how this speed degrades the machine. A natural approximation is to consider this function as increasing and then decreasing: intuitively, when the frequency is too low, the machine stays at constant temperature, hence the work is roughly proportional to the speed. On the contrary, as speed increases, temperature increases much faster than the speed [3]. Hence the decrease of the MTTF becomes too important, and after a certain frequency the work is a decreasing function of the frequency.

Property 2. *$f \mapsto \mathcal{W}(f)$ is increasing then decreasing on $[0, \infty]$. We denote $f_1^\star$ the frequency that maximizes it.*

$f_1^\star$ can be obtained by solving

$$\frac{d\mathcal{W}(f)}{df} = 0$$

As MTTF is a linear function of R_t, $f_1^\star$ is independent of this parameter. For a machine of M nodes, we denote $f_1^\star(i)$ the optimal frequency of node i, that is, its optimal frequency for sequential jobs.

Cooling parameter. We consider homogeneous nodes, however because of their localization in the machine they may not be cooled at the same speed. This is in line with analysis by Ostrouchov et al. [21] showing that on the supercomputer Titan, the GPUs the farthest from the fans were the most likely to fail first. To simulate a difference of temperature due to the relative position of the nodes compared to the position of the fans, we introduce a cooling parameter ν_i and use a few different values of ν_i between the nodes.

In this work, we consider that we have the following natural property on the relation between the cooling parameter ν (a large ν means that the machine cools faster) and the MTTF.

Property 3.

$$\frac{\partial \texttt{MTTF}}{\partial \nu} \ge 0.$$

Said differently, if the resource cools faster, then its life expectation increases.

3.2 Objective function and scheduling problem

At a given initial time t_0, we consider a machine $\mathcal{P}_M = \left(\zeta_i, R^i_{t_0}\right)_{i \le M}$ (see Definition 1).

We consider the problem of scheduling parallel independent jobs. To evaluate different solutions we consider the total volume of work executed over the lifespan of the machine (i.e. until there are no more nodes alive).

Definition 2 (Schedule $\mathcal{S}$). *A schedule $\mathcal{S}$ can be defined as:*

- *The allocation σ of jobs on nodes, along with their start time;*
- *$\mathcal{E} = \{e_0, \cdots, e_{m-1}\}$ the set of time events that include:*
  - *Node changing execution speed*
  - *Node dying*
- *$\mathcal{F} = (f_{i,j})_{i,j}$, such that $f_{i,j}$ is the frequency used by node N_i between e_j and e_{j+1} ($f_{i,j} = 0$ if the node is dead or idle).*

According to Property 1, we have the following lifetime constraint:

$$R^i_0 - R^i_{e_m} \ge \sum_{j=0}^{m-1} \frac{e_{j+1} - e_j}{\texttt{MTTF}(1, f_{i,j}, \zeta)} \text{for all } i \le M$$

Then the total volume of work $\mathcal{W}$ of a schedule $\mathcal{S}$:

$$\mathcal{W}(\mathcal{S}) = \sum_{i \le M} \sum_{j=0}^{m-1} f_{i,j} \cdot \left(e_{j+1} - e_j\right)$$

Analyzing the results. When comparing two strategies, the natural question that one may ask is the following: over the lifetime of the machine, how much has strategy A allowed to provide more computation than strategy B, and at which cost (such as increase in response time).

However, in practice we cannot simply plot this for several reasons: How do we define the lifetime of a machine? Is it the time it takes until no more nodes are alive? Until 50% of the nodes are down? There are several issues in considering until X% of the nodes are down (for $X < 100$). The first one being that an *efficient* strategy to maximize the amount of work done until X% of the nodes are down, would stop using some of the nodes on the verge of

dying, and its relative behavior compared to one that do not use this behavior could be very different for $X' = X + \varepsilon$.

Using $X = 100$ (or any X large enough) is also problematic, because it would imply transformations of the workload to satisfy the decrease in nodes that is too transformative to be really representative, and the output data could be meaningless.

To avoid these bias, instead of focusing on a binary value for a node (being up or down), we use the total cumulative work that the machine would be able to perform if it runs until all its nodes are down.

Analysis per total cumulative work and average frequency
If, for Strategy A, between time t_0 and t_1 each node i went from a state of degradation $R^i_{t_0}$ to a state of degradation $R^i_{t_1}$, and each node performed W_i amount of work, then we say that Strategy A:

- Would perform $\sum_i \frac{W_i}{R^i_{t_0} - R^i_{t_1}}$ units of work over its lifetime in FLOP.
- It has a mean frequency f (in GHz), where f is the average of all jobs frequencies.

Note that $\sum_i \frac{W_i}{R^i_{t_0} - R^i_{t_1}}$ really corresponds to the quantity of work that the platform would perform during its life. Indeed, as the initial state of degradation is 1 and as the node dies when it reaches 0, $R^i_{t_0} - R^i_{t_1}$ corresponds to the fraction of the node potential which has been used. Hence, $\frac{1}{R^i_{t_0} - R^i_{t_1}}$ is how much time we could use the machine in the same way before it breaks. So, finally $\frac{W_i}{R^i_{t_0} - R^i_{t_1}}$ represents the cumulative quantity of work that node i would be able to perform if it keeps being used the same way.

4 THEORETICAL INTUITION

Given a machine $\mathcal{P}_M = \left(\zeta_i, R^i_{t_0}\right)_{i \leq M}$, in this section we study a specific workload: a parallel job of size M, with a total volume of work W, and then as many sequential jobs as possible.

We are looking for the frequency $f^\star_M(\mathcal{P}_M, W)$ to run the parallel job that maximizes the total work done. This is equivalent to maximizing the number of sequential jobs executed: indeed, as soon as the large job is done, the problem reduces to M times the same problem with a one node platform.

Under these assumptions we have the following result:

Theorem 1. *The frequency $f^\star_M(\mathcal{P}_M, W)$ is independent of W and $\left(R^i_{t_0}\right)_i$ and belong to interval $[\min_i f^\star_1(i), \max_i f^\star_1(i)]$.*

This theorem derives directly from the following lemma.

Lemma 1. *Finding the frequency $f^\star_M$ to execute the parallel job on M nodes with a total volume of work W, reduces to finding f that minimizes*

$$F(f) = \frac{1}{f} \sum_i f^\star_1(i) \frac{\mathtt{MTTF}(R^i_0, f^\star_1(i), \zeta)}{\mathtt{MTTF}(R^i_0, f, \zeta)} \tag{3}$$

One should observe that $f^\star_M$ does not depend on W, the volume of work of the parallel job.

Proof. If the parallel job is executed at frequency f, then at most we can execute

$$\mathcal{W}_M(f) = W + \sum_i \left(R^i_0 - \frac{W}{Mf\mathtt{MTTF}(R^i_0, f, \zeta)} \right) f^\star_1(i)\mathtt{MTTF}(R^i_0, f^\star_1(i), \zeta)$$

units of work. This result follows from Prop. 1 which gives the state of degradation of each resource after the execution of the parallel job, and the fact that after this execution, all resources can be seen as individual machines. Then we can observe that $\mathcal{W}_M(f)$ is a linear function of $F(f)$ with negative slope, that means that the frequency that maximizes $\mathcal{W}_M(f)$ is the frequency that minimizes $F(f)$. □

Proof of Theorem 1. Let $F_i(f) = \frac{1}{f \cdot \mathtt{MTTF}(R^i_0, f, \zeta)}$ so that,

$$F(f) = \sum_i f^\star_1(i) \cdot \mathtt{MTTF}(R^i_0, f^\star_1(i), \zeta) \cdot F_i(f).$$

If we consider frequencies in interval $[0, \min_i f^\star_1(i)]$, as all function $F_i(f)$ are decreasing, so the minimum for F is reached at $f = \min_i f^\star_1(i)$. The same way, the minimum of F on interval $[\max_i f^\star_1(i), \infty]$ is reached at $f = \max_i f^\star_1(i)$. The minimum of function F is thus in interval $[\min_i f^\star_1(i), \max_i f^\star_1(i)]$.

□

We conclude that the optimal frequency of a parallel job only depends on the number of nodes requested to execute it and of the set of nodes chosen and not of the size of the job. If nodes are identical, the optimal frequency is the same as the frequency for a sequential job on each of them.

5 SCHEDULING ALGORITHMS

In this Section we discuss scheduling strategies to deal with the node aging problem. The general problem of maximizing the throughput (i.e. utilization) without taking in consideration the aging process is already NP-complete [10].

In our evaluation, our goal is to determine the importance of including aging in our decision. For simplicity we separate the global scheduling problem into two sub-problems:

- Node selection: which node should each job run on (Section 5.1);
- Frequency selection: given a selection of nodes to run a job, which frequency should be used by these nodes (Section 5.2).

5.1 Node selection

Ideally we would like to minimize the impact on job scheduling strategies. Hence we rely on the Easy-backfilling (or EASY-BF) algorithm [19] to schedule jobs. The key idea of EASY-BF is to schedule jobs based on a priority function (often First-Come-First-Served). Whenever possible jobs can be backfilled, as long as they do not delay the job with the largest priority.

In general the implementation focuses on the job priority but not so much on node information –although some version of EASY-BF may use locality to schedule node allocation. In our case, node information is important, hence we consider two node allocation strategies:

- `Vanilla-Easy`: In this case, no sorting operation is performed on the nodes, the algorithm considers them to be all identical. This is the vanilla version of Easy-BF.
- $\nu-$`Easy`: In this strategy, the node are sorted by decreasing ν i.e. the nodes with best cooling parameters first (see Prop. 3).

5.2 Frequency selection

Once an application has been mapped on a set of m nodes, we need an algorithm that decides on the shared frequency at which all these nodes run.

To select this shared frequency, we consider several simple greedy heuristics:

`fStarMin`:	the selected frequency is	$\min_i\{f_1^\star(\tilde{a}_i)\}$
`fStarMax`:	the selected frequency is	$\max_i\{f_1^\star(\tilde{a}_i)\}$
`Arith`:	the selected frequency is	$\frac{1}{m}\sum_i\{f_1^\star(\tilde{a}_i)\}$
`Parallel`:	the selected frequency is	$f_m^\star$.

Note that at this level we only consider application-specific heuristics, that do not depend on the rest of the workload.

6 EXPERIMENTS

The aim of this section is to answer the following questions:

- For a given aging-model, how much can we benefit from taking into consideration node aging in resource management?
- Considering the possible inaccuracy of the aging model, can we still benefit from this model, and how much should one invest into a precise aging model?

We first describe the experimental setup in Section 6.1, before presenting the evaluation results in Section 6.2.

6.1 Experimental setup

For the experiment, we designed an event based simulator to experiments our placement and frequency heuristics. This simulator is coded in Python and available online[1]. In coherence with the literature, we consider that the time for processors to reach steady temperature with task changes in the platform is typically much shorter than the execution time of tasks [13], and hence we consider it instantaneous. This also includes the fact that when nodes are idle they are aged using the formula of the MTTF with a null frequency. At this time we do not consider node locality in our simulator.

For the analysis, we base the architecture and workload on Mira[2] using data made available by Argonne National Laboratory. The Mira supercomputer was launched in 2012 at the 3rd place of TOP500[3] HPC centers. It ran 49,152 nodes and was maintained until 2019. The available trace covers years 2014 to 2018 and contains a total of 330k jobs. Each node is composed of 16 cores running at 1.6 GHz.

6.1.1 The workload. To accelerate the simulation process we divide the number of nodes of each job by 128 and round up the result. This change only affects Mira's traces by a negligible factor as the overwhelming majority of Mira's jobs have a number of node which is a multiple of 128.

[1]https://gitlab.inria.fr/rboezenn/script-improving-supercomputer-usage-with-aging-awareness
[2]https://reports.alcf.anl.gov/data/
[3]https://top500.org/

For each of our experiment, we used 50 batches of 1000 jobs from Mira traces (which are made of the first 50 000 jobs of the trace). These samples provide a wide variability of workloads: on Mira they span from 2 days of consecutive submissions to 100 days, with a mean duration of 6 days. Each of our results is an average over these 50 samples. The first 200 jobs of each sample were put in the waiting queue directly at the start of the simulation to ensure that there is enough waiting jobs to fill the whole machine from the start of the simulation. As we fixed the optimal frequencies around 2 GHz (superior to Mira's 1.6 GHz), we multiplied all the runtimes to simulate a trace meant to be used at 2.5 GHz. This ensures the absence of holes in the schedule.

Note that at this time, in coherence with the discussion in Section 3.2 we analyze periods of time that are short enough so that there are no actual node failures. This allows not to arbitrarily transform the trace as the number of nodes of the platform decreases.

6.1.2 The platform. In the absence of a consensus in an aging model, we study various generic node-level aging models. Specifically, we consider several type of convexity properties on the aging models, based on the relation between the optimal frequency for a single node and the cooling parameter:

- The *Exponential* scenario: a scenario where $\frac{\partial f_1^\star}{\partial \nu} > 0$, i.e. as the cooling parameter increases, the frequency that maximizes the work done on this resource increases

$$\texttt{MTTF}(R_t, f, \nu) = R_t e^{b-\frac{f^\alpha}{\nu}}; b > 0, \alpha > 0 \tag{4}$$

- The *Polynomial* scenario: a scenario where $\frac{\partial f_1^\star}{\partial \nu} < 0$.

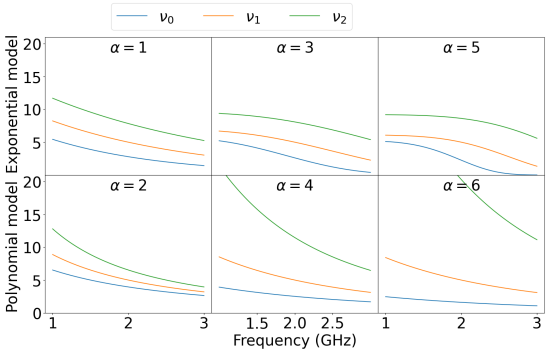
$$\texttt{MTTF}(R_t, f, \nu) = R_t \frac{b}{(\frac{1}{\nu}+f)^\alpha}; b > 0, \alpha > 1 \tag{5}$$

Note that the exponential scenario with $\alpha = 3$, is a good approximation of the node aging model proposed by Zafar et al. [34] .

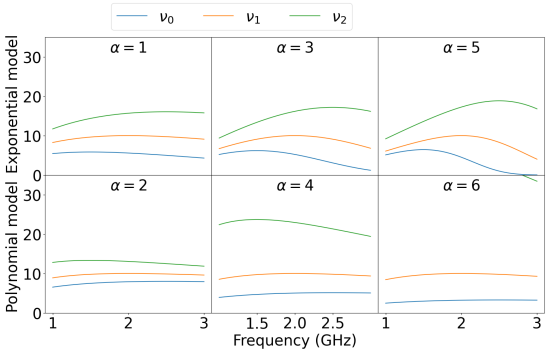
Model Instantiation. For each instantiation, we chose to have three groups of nodes with three different cooling parameters $\nu_0 < \nu_1 < \nu_2$. We chose those so that the values of $f_1^\star$ of the nodes correspond to $(2-\Delta f, 2, 2+\Delta f)$ GHz with $\Delta f = 0.5$ GHz, so that we can compare all models together. Note that however, depending on the positivity of $\frac{\partial f_1^\star}{\partial \nu}$, for the exponential model, 1.5 GHz is the optimal frequency for nodes with the lowest cooling factor (ν_0), while for the polynomial model it is the optimal frequency for the nodes with the highest cooling factor (ν_2). b is set to ensure that the MTTF of a node of the second group is 5 year if this node runs one node jobs at its optimal frequency.

Finally, for a variety of models, we use $\alpha \in \{1, 3, 5\}$ (resp. $\{2, 4, 6\}$) for the exponential (resp. polynomial) model. We plot in Figure 2 the shape of the MTTF and work functions for these various instantiations.

Platform size adjustment. For faster simulation, and in coherence with the transformation described in the previous section 6.1.1, we use a platform with $49,152/128 = 384$ nodes while running Mira's traces. Finally, as we used a couple (platform,trace) with less node that the original, we multiplied the results to by the factor that was



(a) MTTF(f)



(b) $\mathcal{W}(f)$

Figure 2: Shape of different functions depending on the model and parametrization.

used to reduce the number of nodes (= 128) to have a number of FLOP consistent with what we could have in reality.

Extrapolating the number of flops. To extrapolate the quantity of FLOP from the number of clock turns, we consider that Mira's 10 petaFLOPS are attained when all the cores are running at 1.6 GHz (based on the public data of ANL[4]) and apply a proportionality rule. This methodology gives a number of flops per node and per clock turn of 127.

6.1.3 Updating Easy-BF. As jobs have execution times which depend on their frequencies, we have to modify the Easy-BF algorithm as a consequence. First, when a job is placed, its walltime is divided by the frequency at which it runs (compared to if was running à 1 GHz). Then, during backfilling, three (overlapping) sets of nodes are explored to place a job:

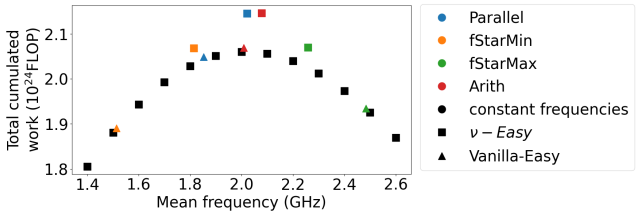
- the set of all free nodes
- the set of free nodes on which no job has been reserved for later
- the set of free nodes on which a job has been reserved for later

Then, for each of these sets (if they contain enough nodes) the scheduler tries to schedule the job. It takes the first n nodes of each set (n being the number of processors that the task requires) and computes the corresponding frequency (with the heuristic frequency). If the job fits (i.e. if it can finish before the reserved job starts or if it only runs on nodes without the booked job), it is then scheduled.

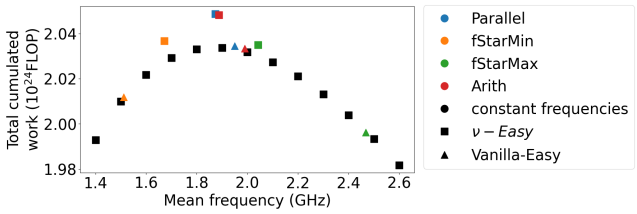
[4]https://www.alcf.anl.gov/alcf-resources/mira

6.2 Experimental evaluation

We first discuss in Section 6.2.1 a scenario where the aging model is known with accuracy by the system, and study the various algorithms. Then in Section 6.2.2 we discuss the impact of inaccuracy in the aging model on the algorithms performance.



(a) Exponential model with $\alpha = 3$



(b) Polynomial model with $\alpha = 4$

Figure 3: Evaluation of the heuristics in terms of compute speed and in terms of aging efficiency, we represent both node reordering strategies (`Vanilla-Easy`, ν – `Easy`), and frequency reconfiguration strategies.

6.2.1 With accurate knowledge on the aging model. As discussed in Section 3.2, we study on the one hand the **average frequency**, i.e. how fast one can obtain results, as well as **the number of FLOP that one can obtain from the machine overt its lifetime**. We compare the strategies presented in Section 5.2 to various static frequencies between 1.4 GHz and 2.6 GHz.

We present some results visually in Figure 3. (The results for the other aging models look similar). Note that the two objectives can be opposing, hence the performance form a Pareto front[5], where ideal performance is higher (i.e. more FLOPs per unit of degradation), and right-most (i.e. faster execution). As an example, with the exponential model, $\alpha = 3$ (Fig. 3a), and while only considering constant frequencies, the best constant frequency for the degradation criteria would be 2 GHz for ν – `Easy` and `Vanilla-Easy`.

We first discuss the constant speed heuristics, then the variable speed heuristics with `Vanilla-Easy` ordering strategy, and then with ν – `Easy` ordering strategy.

[5]A Pareto front is a representation of the trade-off between conflicting objectives https://en.wikipedia.org/wiki/Pareto_front

`Parallel`		$\alpha = 1$	$\alpha = 3$	$\alpha = 5$
2.6 GHz	Speed	−21.27%	−22.25%	−22.99%
	Work	3.79%	14.74%	26.60%
2 GHz	Speed	2.35%	1.08%	0.12%
	Work	1.36%	4.16%	7.31%

`Arith`		$\alpha = 1$	$\alpha = 3$	$\alpha = 5$
2.6 GHz	Speed	−20.24%	−20.04%	−19.57%
	Work	3.81%	14.77%	26.30%
2 GHz	Speed	3.69%	3.94%	4.55%
	Work	1.38%	4.19%	7.05%

Table 1: Performance of `Parallel` and `Arith` compared to two constant heuristics (2 GHz and 2.6 GHz) for the exponential aging model.

`Parallel`		$\alpha = 2$	$\alpha = 4$	$\alpha = 6$
2.6 GHz	Speed	−27.94%	−31.79%	−34.40%
	Work	3.38%	7.77%	11.04%
1.9 GHz	Speed	−1.402%	−6.67%	−10.22%
	Work	0.74%	1.10%	1.33%

`Arith`		$\alpha = 2$	$\alpha = 4$	$\alpha = 6$
2.6 GHz	Speed	−27.39%	−31.14%	−33.68%
	Work	3.35%	7.67%	10.86%
1.9 GHz	Speed	−0.65%	−5.77%	−9.25%
	Work	0.71%	1.01%	1.16%

Table 2: Performance of `Parallel` and `Arith` compared to two constant heuristics (1.9 GHz and 2.6 GHz) for the polynomial aging model.

Impact of ordering strategy. The constant frequencies heuristics behave identically independently of the placement algorithm (ν – `Easy`, `Vanilla-Easy`). Indeed, aging depends only on the aging model and on the various frequencies used by the node. Since the nodes use a single identical frequency, their aging is independent of placement.

Interestingly, variable frequencies heuristics with `Vanilla-Easy` ordering strategy are on the same Pareto front that the constant frequencies. This is because most of the trace use a large number of nodes. As per the law of large numbers (random ordering of the nodes) most tasks are running on a set of nodes with always the same proportion of each node type. In addition, our four variable frequency heuristics give the same frequency as long as the proportion of each node type is the same. This implies that when paired with the `Vanilla-Easy` ordering strategy, variable frequency heuristics mostly always run at the same frequency. Hence they are on the constant frequency Pareto front.

Take-away 1: If node reordering is not available, it is not necessary to do advanced heuristics.

General Discussion. The first observation from Figure 3 is that for constant speed frequency, frequencies below 2 GHz are sub-optimal since they are both slower and perform less work. The second observations is that the Pareto front formed by the variables speed heuristics is consistently better than the one formed by the constant speed heuristics: it produces more work at the same frequency or compute faster for the same quantity of work.

In Tables 1 and 2, we provide the difference between the behavior at the constant speed that maximizes the work 2 GHz (1.9 GHz for polynomial models), and 2.6 GHz (constant speed that maximizes the speed), which are the extremities of the Pareto front provided by the constant speed heuristics. We compare those to the two best heuristics: `Parallel` and `Arith`.

For most models (polynomial and exponential with $\alpha = 1$) using the fastest frequency allows to compute much faster (22% for exponential models, up to 30% for polynomial model) a gain in terms of work between 4 and 25%. The difference between the behavior for the exponential and the polynomial model is interesting with lesser gain for the polynomial model. These lesser gains can be explained by looking at the flatness of the $\mathcal{W}(f)$ function (see Fig.2b). Note that only a little variability is enough for considerable gains (Exponential with $\alpha \geq 3$).

`Parallel`		$\Delta f = 0.25$	$\Delta f = 0.5$	$\Delta f = 0.75$
2.6 GHz	Speed	−22.82%	−22.24%	−21.93%
	Work	14.92%	14.74%	11.09%
2 GHz	Speed	0.34%	1.08%	1.49%
	Work	1.06%	4.16%	8.00%

`Arith`		$\Delta f = 0.25$	$\Delta f = 0.5$	$\Delta f = 0.75$
2.6 GHz	Speed	−22.29%	−20.04%	−15.70%
	Work	17.93%	14.78%	12.06%
2 GHz	Speed	1.03%	3.94%	9.59%
	Work	1.07%	4.19%	8.94%

Table 3: Performance of `Parallel` and `Arith` compared to two constant heuristics (2 GHz and 2.6 GHz) for the exponential aging model with $\alpha = 3$. The Δ between the optimal frequencies changes between the columns.

Take-away 2:

- When $\mathcal{W}$ has little variability, then selecting an *incorrect* frequency has little impact on the global work done by the machine even if the machine ages *slightly* faster. On the contrary, the benefits can be important even with a minimal variability.
- While the `Parallel` heuristic seems to work better, the `Arith` heuristic which is much simpler to implement is competitive with `Parallel` and seems like a reasonable solution.

Finally, our last analysis is to understand the impact of cooling parameters on the performance. Intuitively, the less variability there is on nodes MTTF function due to difference in cooling, the less gain we can obtain. We confirm this in Table 3, where we evaluate

the gain when the difference in node categories increases compared to the constant heuristic that performs the most work (2 GHz). This difference is measured by the delta between the optimal speed for each category of nodes. As described in section 6.1.2, the Δf is the difference between optimal frequencies of the 3 groups of nodes. The optimal frequencies of the 3 groups of nodes are $(2 - \Delta f, 2, 2 + \Delta f)$ (in GHz).

Take-away 3: An increasing difference in node cooling reinforces the need for aging-aware strategies, particularly when comparing to the constant speed strategy that maximizes work.

6.2.2 Impact of the aging model inaccuracy. In practice, the MTTF is only a mean across all the hardware stock, as factors such as the manufacturing steps create differences between the components.

In this section we consider the scenario where the scheduling algorithms use as information the average behavior (a_i and b from the previous section), but where the real behavior differs from this behavior by a factor. We only consider the exponential model with $\alpha = 3$, which corresponds the most to the node aging model proposed by Zafar et al. [34]. We consider that the model type and the value of α is perfectly known by the scheduler. Only a_i and b have values which are known with a margin of error.

To study the impact of such variations, we chose to multiply each $\tilde{a_i}$ and $\tilde{b_i}$ by $1 + \varepsilon$, with ε a normal variable of mean 0 and standard deviation σ. An example of the action of this transformation is presented in Figure 4 for $\sigma = 0.05$. The scheduling algorithm, however, only has access to the values of a_i and b before they get multiplied by ε. The impact of this noise is plotted in Figure 5.

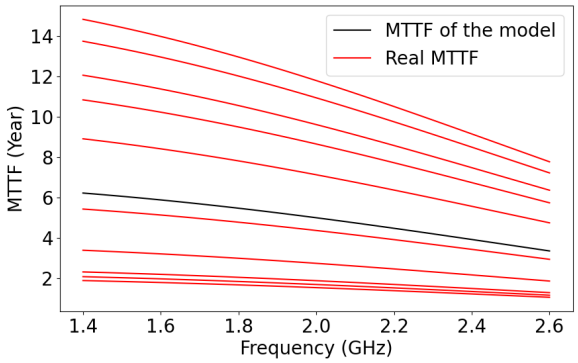
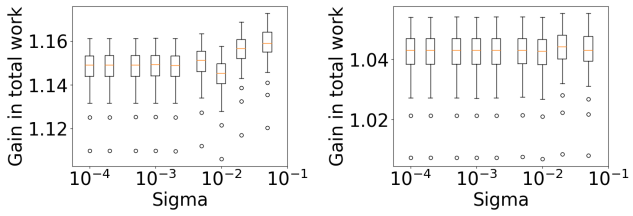


Figure 4: Here, the black curve is what the algorithm sees as node behavior, while the red curves are ten generated node behavior with $\sigma = 0.05$.

The mean frequencies of `Parallel` are independent of the noise since the information that the algorithm has do not change. However, there is a difference in terms of work. This is due to two factors. The first one is that the scheduler has no access to the real MTTF values. The second is that, due to noise, the platform is fully heterogeneous. The worst noise considered here is $\sigma = 0.05$. Even for this level of noise, the impact on the total quantity of work is negligible.



(a) Compared to 2.6 GHz **(b) Compared to 2 GHz**

Figure 5: Quantity of work performed by `Parallel` compared to 2 and 2.6 GHz, as a function to the level of noise.

Take-away 4: `Parallel` is quite robust to model inaccuracy. Even with large variability between the real behavior of the nodes, and the information that the algorithm use, the scheduling strategies that we derive have a positive impact on the total work that can be done on an HPC machine.

7 CONCLUSION

Up to now, when considering its environmental impact, HPC resource management has mostly focused on improving energy efficiency or power usage. Indeed, renewing hardware allowed to improve the energy consumption while improving the performance of HPC machines. Research has focused on how to improve the power consumption of a machine, or how to rebalance workload to use renewable sources of energy. This strategy is not sustainable in a future where access to compute resources becomes scarce. Particularly given the fact that it becomes less and less obvious that we need a scale much larger than exascale.

In this work we have studied the importance of taking aging-related parameters into consideration when managing resources. We have proposed strategies where we can adjust the frequencies used by nodes, with the constraints that all nodes shared by a job use the same frequency during the execution of the node. These adjustments are based on aging properties of these nodes. We have demonstrated on some examples that these strategies allow for non-negligible gain in terms of work done. In addition, we have also been able to show that this strategy is useful, even with inaccurate aging models. Finally, we have studied the limits of these considerations, particularly when the function work is flat. Importantly, we have reminded that at this stage, more research is needed for accurate aging models, and we believe that our results should motivate this direction.

We believe that this work is an important first step to demonstrate the feasibility and importance of considering aging in order to maximum the yield that one can obtain from HPC machines. It comes at a trade-off in terms of response time. Future work should consider aging of other components.

ACKNOWLEDGMENTS

We want to thank the reviewers for their thorough evaluation. Parts of this work have been supported by the Inria Exploratory Project REPAS and by the Exa-DoST project (PEPR NumPEx), reference ANR-22-EXNU-0004.

Some of the data from this work was generated from resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

REFERENCES

[1] Muhammad Ashraful Alam and Souvik Mahapatra. A comprehensive model of PMOS NBTI degradation. *Microelectronics Reliability*, 45(1):71–81, 2005.

[2] Hussam Amrouch, Victor M. van Santen, Thomas Ebi, Volker Wenzel, and Jörg Henkel. Towards interdependencies of aging mechanisms. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 478–485. IEEE, 2014.

[3] Leon Atkins, Guillaume Aupy, Daniel Cole, and Kirk Pruhs. Speed scaling to manage temperature. In *Theory and Practice of Algorithms in (Computer) Systems: First International ICST Conference, TAPAS 2011, Rome, Italy, April 18-20, 2011. Proceedings*, pages 9–20. Springer, 2011.

[4] Guillaume Aupy, Anne Benoit, Fanny Dufossé, and Yves Robert. Reclaiming the energy of a schedule: models and algorithms. *Concurrency and Computation: Practice and Experience*, 25(11):1505–1523, 2013.

[5] Guillaume Aupy, Anne Benoit, Thomas Hérault, Yves Robert, Frédéric Vivien, and Dounia Zaidouni. On the combination of silent error detection and checkpointing. In *2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing*, pages 11–20. IEEE, 2013.

[6] Guillaume Aupy, Anne Benoit, and Yves Robert. Energy-aware scheduling under reliability and makespan constraints. In *2012 19th International Conference on High Performance Computing*, pages 1–10. IEEE, 2012.

[7] Guillaume Aupy, Yves Robert, Frédéric Vivien, and Dounia Zaidouni. Checkpointing algorithms and fault prediction. *Journal of Parallel and Distributed Computing*, 74(2):2048–2064, 2014.

[8] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM (JACM)*, 54(1):1–39, 2007.

[9] Jack Dongarra, Thomas Herault, and Yves Robert. *Fault tolerance techniques for high-performance computing*. Springer, 2015.

[10] Michael R Garey and David S Johnson. "strong"np-completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM)*, 25(3):499–508, 1978.

[11] Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. Failures in large scale systems: long-term measurement, analysis, and implications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.

[12] Lin Huang, Feng Yuan, and Qiang Xu. On task allocation and scheduling for lifetime extension of platform-based MPSoC designs. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2088–2099, 2011.

[13] Lin Huang, Feng Yuan, and Qiang Xu. On task allocation and scheduling for lifetime extension of platform-based MPSoC designs. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2088–2099, 2011.

[14] Vincent Huard, CR Parthasarathy, Alain Bravaix, Chloe Guerin, and Emmanuel Pion. CMOS device design-in reliability approach in advanced nodes. In *2009 IEEE International Reliability Physics Symposium*, pages 624–633. IEEE, 2009.

[15] Heather Klemick, Elizabeth Kopits, and Ann Wolverton. How do data centers make energy efficiency investment decisions? qualitative evidence from focus groups and interviews. *Energy efficiency*, 12:1359–1377, 2019.

[16] Liuzixuan Lin and Andrew A. Chien. Adapting datacenter capacity for greener datacenters and grid. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*, pages 200–213, 2023.

[17] Alejandro Masrur, Philipp Kindt, Martin Becker, Samarjit Chakraborty, Veit Kleeberger, Martin Barke, and Ulf Schlichtmann. Schedulability analysis for processors with aging-aware autonomic frequency scaling. In *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 11–20, 2012.

[18] Md Moniruzzaman, Ahmed H. Okilly, Seungdeog Choi, Jeihoon Baek, Tahmid Ibne Mannan, and Zeenat Islam. A comprehensive study of machine learning algorithms for GPU based real-time monitoring and lifetime prediction of igbts. In *2024 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pages 2678–2684. IEEE, 2024.

[19] Ahuva W. Mu'alem and Dror G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE transactions on parallel and distributed systems*, 12(6):529–543, 2001.

[20] Fabian Oboril and Mehdi B Tahoori. Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12. IEEE, 2012.

[21] George Ostrouchov, Don Maxwell, Rizwan A. Ashraf, Christian Engelmann, Mallikarjun Shankar, and James H. Rogers. GPU lifetimes on Titan supercomputer: Survival analysis and reliability. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2020.

[22] Desong Pang, Dawen Xu, Ying Wang, and Huaguo Liang. MTTF-aware reliability task scheduling for PIM-based heterogeneous computing system. In *2018 IEEE International Test Conference in Asia (ITC-Asia)*, pages 25–30, 2018.

[23] Prem Kumar Ramesh, Viswanathan Subramanian, and Arun K Somani. System level analysis for achieving thermal balance and lifetime reliability in reliably overclocked systems. *International Journal on Advances in Systems and Measurements*, 2(4):258–268, 2010.

[24] Yoann Mamy Randriamihaja, Vincent Huard, Xavier Federspiel, Alban Zaka, Pierpaolo Palestri, Denis Rideau, David Roy, and Alain Bravaix. Microscopic scale characterization and modeling of transistor degradation under HC stress. *Microelectronics Reliability*, 52(11):2513–2520, 2012.

[25] Vijay Reddy, Anand T Krishnan, Andrew Marshall, John Rodriguez, Sreedhar Natarajan, Tim Rost, and Srikanth Krishnan. Impact of negative bias temperature instability on digital circuit reliability. *Microelectronics Reliability*, 45(1):31–38, 2005.

[26] Purushottam Sigdel, Xu Yuan, and Nian-Feng Tzeng. Realizing best checkpointing control in computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(2):315–329, 2020.

[27] Jayanth Srinivasan, S aritaV. Adve, Pradip Bose, and Jude A. Rivers. Lifetime reliability: toward an architectural solution. *IEEE Micro*, 25(3):70–80, 2005.

[28] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pages 276–287, 2004.

[29] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, Jude A. Rivers, and Chao-Kun Hu. Ramp: A model for reliability aware microprocessor design. *IBM research report*, 2003.

[30] Jin Sun, Roman Lysecky, Karthik Shankar, Avinash Kodi, Ahmed Louri, and Janet Roveda. Workload assignment considering NBTI degradation in multicore systems. *J. Emerg. Technol. Comput. Syst.*, 10(1), January 2014.

[31] Victor M. van Santen, Hussam Amrouch, Narendra Parihar, Souvik Mahapatra, and Jörg Henkel. Aging-aware voltage scaling. In *2016 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 576–581, 2016.

[32] Jyothi B. Velamala, Ketul B. Sutaria, Takashi Sato, and Yu Cao. Aging statistics based on trapping/detrapping: Silicon evidence, modeling and long-term prediction. In *2012 IEEE International Reliability Physics Symposium (IRPS)*, pages 2F.2.1–2F.2.5, 2012.

[33] Kai-Chiang Wu, Ming-Chao Lee, Diana Marculescu, and Shih-Chieh Chang. Mitigating lifetime underestimation: A system-level approach considering temperature variations and correlations between failure mechanisms. In *2012 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1269–1274, 2012.

[34] Sufi Zafar, Byoung Hun Lee, James Stathis, Alessandro Callegari, and Tak Ning. A model for negative bias temperature instability (NBTI) in oxide and high $\varkappa$ pfets. In *Digest of Technical Papers. 2004 Symposium on VLSI Technology, 2004.*, pages 208–209, 2004.

[35] Christopher Zimmer, Don Maxwell, Stephen McNally, Scott Atchley, and Sudharshan S. Vazhkudai. GPU age-aware scheduling to improve the reliability of leadership jobs on Titan. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 83–93, 2018.