International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

Survey Paper

The International Journal of High Performance Computing Applications 1–27


Article reuse guidelines: sagepub.com/journals-permissions
DOI: 10.1177/1094342021990433
journals.sagepub.com/home/hpc

Resilience and fault tolerance in high-performance computing for numerical weather and climate prediction

Tommaso Benacchio[1], Luca Bonaventura[1], Mirco Altenbernd[2], Chris D Cantwell[3], Peter D Düben[4,5], Mike Gillard[6], Luc Giraud[7], Dominik Göddeke[2], Erwan Raffin[8], Keita Teranishi[9] and Nils Wedi[4]

Abstract

Progress in numerical weather and climate prediction accuracy greatly depends on the growth of the available computing power. As the number of cores in top computing facilities pushes into the millions, increased average frequency of hardware and software failures forces users to review their algorithms and systems in order to protect simulations from breakdown. This report surveys hardware, application-level and algorithm-level resilience approaches of particular relevance to time-critical numerical weather and climate prediction systems. A selection of applicable existing strategies is analysed, featuring interpolation-restart and compressed checkpointing for the numerical schemes, in-memory checkpointing, user-level failure mitigation and backup-based methods for the systems. Numerical examples showcase the performance of the techniques in addressing faults, with particular emphasis on iterative solvers for linear systems, a staple of atmospheric fluid flow solvers. The potential impact of these strategies is discussed in relation to current development of numerical weather prediction algorithms and systems towards the exascale. Trade-offs between performance, efficiency and effectiveness of resiliency strategies are analysed and some recommendations outlined for future developments.

Keywords

Fault-tolerant computing, high-performance computing, application-level resilience, numerical weather prediction, iterative solvers

1. Introduction

Since the dawn of computing, numerical weather prediction (NWP) and climate studies have served as a key application to test the performance of cutting-edge hardware architectures. In time-critical operational weather services, supercomputers provide the infrastructure for a round-the-clock enterprise relying on the timely execution of the forecast suite, ranging from handling and assimilating observations for the generation of adequate initial conditions to performing extended-range predictions.

Up to a decade ago, improvement in atmospheric fluid flow simulations followed steady increases in clock speed of supercomputers' processors. This growth has flattened out in the last decade, so that, besides systematic exploitation of alternative architectures such as GPUs, an increase in node count represents the main factor in the current advances in computational performance. Extrapolation of the sheer core count and performance of the machines included in the TOP500 list shows that the one-million core and exaflop (10^{18} double precision floating-point operations per second) limits are bound to be crossed in the next few years (see Meuer et al., 2019).

[1] MOX – Modelling and Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Milan, Italy
[2] Institute for Applied Analysis and Numerical Simulation and Cluster of Excellence 'Data-Driven Simulation Science', University of Stuttgart, Stuttgart, Germany
[3] Department of Aeronautics, Imperial College London, London, UK
[4] European Centre for Medium Range Weather Forecasts, Reading, UK
[5] AOPP, Department of Physics, University of Oxford, Oxford, UK
[6] Loughborough University, Loughborough, UK
[7] HiePACS, Inria Bordeaux, Sud-Ouest, Talence, France
[8] CEPP – Center for Excellence in Performance Programming, Atos, Rennes, France
[9] Sandia National Laboratories, Livermore, CA, USA

Corresponding author:
Tommaso Benacchio, MOX – Modelling and Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milan, Italy.
Email: tommaso.benacchio@polimi.it

Table 1. Subset of the Top500 list (November 2020 standings) with ranking, owning institution, core count (in thousands), and maximum LINPACK performance Rmax for systems exclusively dedicated to numerical weather prediction and climate studies, operational or for research (for more details see Meuer et al., 2019).

Ranking [/500]	Institution	KCores	Rmax [PFLOPs]
30, 34, 196, 201	Météo-France	294.91(2) + 72 + 73.44	8.191 + 7.683 + 2.168 + 2.157
37, 117, 118	United Kingdom Meteorological Office	241.9 + 89.86(2)	7.039 + 2.802(2)
49, 50, 313	Japan Meteorological Agency	135.79(2) + 35.2	5.731(2) + 1.715
60	NCAR (US)	144.9	4.788
74, 75	ECMWF	126.47(2)	3.945(2)
77	Indian Institute of Tropical Meteorology	119.32	3.764
106, 425, 426	NOAA (US)	63.84 + 48.96(2)	3.081 + 1.635(2)
109	Deutsches Klimarechenzentrum	99.07	3.011
140	Deutscher Wetterdienst	14.85	2.596
144	NCMRWF (India)	83.59	2.570
147, 159	China Meteorological Administration	50.82 + 48.13	2.547 + 2.435
164, 165	Korea Meteorological Administration	69.6(2)	2.396(2)
321	Beijing Meteorological Association	28.8	1.683
467	Chinese Academy of Sciences	48	1.428

As of November 2020, the TOP500 list features at least 25 high-performance computing (HPC) systems exclusively devoted to numerical weather and climate prediction, with their combined figures exceeding 2.6 million cores, 89 Petaflops maximum LINPACK performance and memory in the region of petabytes (Table 1). This excludes general-purpose HPC systems – some very high-ranking – on which atmospheric simulations also run routinely. The growing core count and computational performance have enabled a steady increase in model resolution and, ultimately, forecast accuracy (Bauer et al., 2015; Schulthess et al., 2018; Wedi et al., 2020), but have also posed serious challenges to existing modelling systems, numerical algorithms, interplay between different modelling components, computational infrastructure, increasingly hierarchical memory management, data transfer efficiency, multi-node scalability, and energy consumption.

As pointed out in Nielsen (2018), challenges of exascale scientific computing will revolve around *power* (current flops/watt rates projecting to unsustainable consumption in the range of hundreds of megawatts), *concurrency* (reduction of communications in parallel programs), *data locality* (redefining performance from flops to bandwidth/latency), and *memory* (as available memory per core is decreasing). One relatively less considered issue on the path towards exascale simulations in numerical weather prediction concerns the reliability of computing systems. While reliability was a major concern in the early days of computing (see, e.g. the discussion in Lyons and Vanderkulk, 1962), the practical irrelevance of computing hardware faults has now been taken for granted for many years. However, infallibility assumptions for HPC hardware, as well as absence of faults at the bit level, inevitably cease to hold as the number of processors increases, their size shrinks and their energy use varies. Applications spread across a larger number of computational nodes with single points of failure. Bit-level faults may also be exaggerated in the future due to a trend to exploit reduced precision in order to accelerate the time-to-solution (Düben et al., 2014b).

The present paper aims to explore the reliability issue in the context of numerical weather and climate prediction applications by:

- outlining the key algorithmic components of current NWP systems and their associated computational challenges;
- providing an overview of relevant existing and emerging resilience strategies for exascale computing;
- illustrating how such technologies may be applied in the context of NWP systems;
- presenting example applications of these approaches to NWP model components;
- discussing the open challenges and opportunities associated with applying resilience to current and future NWP systems.

Above and in the following, resilience will be used as an umbrella term for techniques that keep applications running despite faults. After a general overview of NWP model components and computational complexity (Section 2) and a taxonomy of resilience-related concepts (Section 3.1), Section 3.2 explores aspects of resilience at the level of HPC hardware, considering the case study of the BullSequana system. Section 3.3 then examines system-level and backup grids-based resilience approaches that do not modify lower-level numerical implementations. Section 3.4 considers algorithmic resilience, discussing the interpolation-restart and compressed checkpointing methods aimed at ensuring continuing execution and solution quality of iterative methods for linear systems upon hard or soft faults. In Section 4 we present numerical results obtained applying some of the described algorithms to implementations derived from numerical weather prediction codes. Section 5 discusses how to compose and apply

the approaches treated in the paper in an NWP context, listing related challenges such as the interplay with emerging domain-specific language-based software frameworks. Section 6 concludes the paper.

We remark that this paper chiefly deals with fault recovery techniques. We refer to the literature (e.g. Bautista-Gomez and Cappello, 2015; Berrocal et al., 2015, 2016; Turnbull and Alldrin, 2003) for fault detection and prediction, as well as for other techniques, such as advanced focused checkpointing strategies (Cores et al., 2013; Islam et al., 2013; Kohl et al., 2019; Losada et al., 2019; Plank et al., 1995; Rodríguez et al., 2010; Sancho et al., 2004; Tao et al., 2018), selective identification of parts of models in need of reliability (Bridges et al., 2012; Hoemmen et al., 2011), self-stabilizing iterative solvers (Sao and Vuduc, 2013), data compression techniques (Di and Cappello, 2016), global view resilience (Chien et al., 2015), and resilience in the framework of domain decomposition preconditioners (Rizzi et al., 2018a, 2018b; Sargsyan et al., 2015).

2. Numerical weather and climate prediction models

Operational weather prediction systems use a complex workflow that involves gathering and selecting observations, assimilating these observations into the forecast model to generate initial conditions, advancing the model to make predictions of future weather, post-processing model output and disseminating the forecast products. Climate predictions require simulations of very long time periods – often hundreds of years – and therefore require high throughput rates and stable predictions systems.

Within the numerical model, the *dynamical core* discretises the governing equations of the atmospheric flow – typically the inviscid compressible Navier-Stokes equations under gravity – using a prescribed computational grid. The typical prognostic variables include wind, temperature, air density or pressure, as well as other thermodynamic and water substance variables – different options for variables and equation sets are discussed, e.g. in White (2002). The numerical solution involves handling the transport by the wind as well as the faster dynamics of internal gravity waves and sound waves, induced by the compressibility and vertical stratification of the atmosphere (Benacchio, 2014; Benacchio et al., 2014; Bonaventura and Budich, 2012; Steppeler et al., 2003). Semi-implicit time discretization strategies are widely employed in atmospheric models in order to achieve efficiency by employing large time steps unconstrained by the speed of fast waves (see, e.g. Mengaldo et al., 2019, and the discussion therein). Semi-implicit approaches, however, imply the formulation and solution of global linear systems that make model discretizations more involved and parallelization harder than with explicit methods. Experience with operational semi-implicit discretizations suggests that preconditioned linear solvers routinely take up sizeable portions – up to 30% – of wall-clock time in dynamical core runs. Conceptually similar linear solvers also feature in the data assimilation process, which takes up as large a share of the total computational time required to produce a weather forecast as the advancement of the dynamical model. Linear solvers resilience is therefore of paramount importance for a daily predictable runtime and timely dissemination of forecast products.

In addition to the inviscid dynamics solved by the dynamical core, many of the processes associated with meteorologically relevant phenomena – convection, radiation, cloud microphysics, boundary layer turbulence, gravity wave drag, and others, customarily referred to as the *physics* in this context – occur at scales that cannot be resolved by the computational grid, so their subgrid effects are parametrized and fed into the dynamical core as source terms. Model complexity is further enhanced by other components, such as the ocean and wave model, the land surface model, the atmospheric chemistry model, and their mutual coupling.

Today's state-of-the-art global operational NWP systems provide forecasts simulated with 25-9 km average grid-spacing in the horizontal direction. Limited area models use grid-spacings down to 1 km. Ensembles with perturbed initial conditions and additional model perturbations with up to 50 members are used to sample the initial condition and model uncertainty. Vertical discretizations use more than 100 unevenly spaced vertical levels out to model tops of around 80 km. This translates into more than 500 million spatial degrees of freedom per variable, requiring more than 2000 forward discrete time steps for a 2-week forecast.

Due to the relatively short window between the arrival of observations, the quality control, assimilation and subsequent dissemination of the forecast, typically only 1 hour is available for a 2-week NWP forecast including uncertainty estimation. Failed tasks or other hardware issues can very quickly lead to delayed forecast dissemination. Efficiency and scalability, with a mixture of both strong scaling and weak scaling requirements, become important when exploiting an increasing amount of distributed computing resources. Recent studies warned that current model efficiency needed to improve by at least two orders of magnitude in order to be able to run timely 1 km global NWP simulations on exascale systems (Neumann et al., 2019; Schulthess et al., 2018). Simulations at 1 km global resolution promise significant improvements in predictions as they allow for the explicit representation of deep convection within simulations.

Even though HPC hardware reliability has been relatively stable in recent years, resilience to hard and soft faults for NWP models should now be investigated more thoroughly, given increasingly tighter production schedules and expanding core counts. Scalability tests with weather models using a significant fraction of some of the world's fastest supercomputers are still feasible without serious problems due to faults (Düben et al., 2020; Fuhrer et al., 2018), but checkpointing intermediate forecast results to

disk is already required. As an example, operational deterministic global forecasts at 9 km resolution at the European Centre for Medium-Range Weather Forecasts (ECMWF) run with 324 compute nodes and 28 I/O server nodes with four MPI tasks per node. This translates into 202 MB of data to be written for a restart file per MPI task – including the models for atmosphere, ocean, waves and ice. Writing a restart dump for this system takes 1 to 3 seconds. During a 10-day forecast, five checkpoints are written, requiring less than 1% of the total forecast runtime. Although these figures suggest that there is still some leeway before checkpointing becomes unaffordable in operational NWP, the procedure already absorbs almost the entire bandwidth of the Lustre file system. At 1 km resolution for the atmospheric model component, the total size of restart files reaches 4 TB, which may be simultaneously written from across thousands of MPI tasks to a single file system. This raises questions regarding the interference among different user jobs on the entire HPC performance. Other strategies for resilience already in place in NWP models include hosting identical twin computing clusters for instant switching of workloads and separating resources for research from those dedicated to operational suites.

3. Resilience methodologies

3.1. Taxonomy

In the reliability literature, *faults* are defined as causes of *errors*, which are parts of the state causing a failure. The latter represents the transition to incorrect service (for taxonomy and wider related literature we refer to the reviews in Avizienis et al., 2004; Cappello, 2009; Cappello et al., 2014; Dongarra et al., 2015; Mittal and Vetter, 2015; Nielsen, 2018; Snir et al., 2014). Parallel computing systems may undergo hardware breakdowns in several of their components – processors, nodes, blades, cabinets. These breakdowns are usually defined as *hard faults*, they are in general reproducible and, if unaddressed, bring programs to halt. *Soft faults* are instead usually caused by fluctuations in radiation that introduce spurious modifications in the program data in the form of *bit flips* and are usually non-reproducible (Cher et al., 2014). Faults are further distinguished into detected and corrected, detected and uncorrectable, and undetected ones. The latter category can take the form of *silent data corruption* (SDC) and lead the program to compute the wrong solution unbeknown to the user (Calhoun et al., 2017; De Oliveira et al., 2017; Elliott et al., 2014, 2015, 2016; Feng et al., 2010; Fiala et al., 2012; Guhur et al., 2017; Li et al., 2018; Michalak et al., 2012). Next to other performance figures in HPC, mean time between failure (MTBF), made of the sum of mean time to interrupt (MTTI) and mean time to repair (MTTR), has arisen as a measure of reliability of a computing system.

Efforts to limit the impact of computing failures have usually revolved around *rollback* strategies, mostly in the form of *checkpoint-restart* (CPR). In this approach, state data are written out to a parallel file system, enabling the application to continue from this consistent state after being restarted following a failure. Checkpointing is intuitive in principle and is commonly agreed to be good programming practice for code bases of a certain size. However, it is computationally expensive as it involves moving data to and from disk, a potential requeuing of the job when run under batch submission systems, and an increase in total system resource and energy usage. Looking ahead, the delays in time-to-solution and overhead caused by CPR may become less and less sustainable for time-bound applications such as NWP, given the inherently increasing failure frequency in future operating schedules. While studies have identified optimal checkpointing intervals (Daly, 2006), CPR clearly stops being worthwhile at the point when the checkpointing procedure takes longer than the average mean time between failure that it is meant to protect from. As an example, Snir et al. (2014) reports CPR times of 2000s for a 64 TB dump on 1000 nodes. Other traditional resilience strategies have used other forms of redundancy, typically replication (Benoit et al., 2017; Dongarra et al., 2015). While these approaches may present some advantages when MTBF is low and CPR is heavy, they also carry sizeable overheads and are generally power-hungry.

To protect the system from bitflips, error-correcting code (ECC) memory features in almost all architectures used in HPC (see, e.g. Mittal and Inukonda, 2018, for a recent review). The associated overhead in terms of performance, storage, and power consumption is not negligible. Depending on the configuration, capacity overhead varies between 10% and 40% (Li et al., 2013) and performance-power overhead around 10–20% (Kim et al., 2015b). In the hardware community, a 12.5% overhead appears to be a commonly accepted value (Sun, 2014). In this context, figures released by hardware vendors tend to include performance penalty only, overlooking the power requirements for ECC/Chipkill (Dell, 1997). Memory (DRAM) protection – increasing refresh rate and adding more parity modules/bits to ECC/Chipkill – also needs energy. The hardware community agreed on the power or performance penalty associated with reliability enhancement. Despite new techniques enabling more efficient reliability enhancement, it is inevitable for computer system architects to allocate some power and performance budget for reliability. For example, in Kim et al. (2015a) a highly reliable Chipkill was implemented with 2 extra chips for a 16-chip memory module. Other authors suggested hybridizing dynamic refresh rates and Chipkill together to tune reliability, power and performance trade-offs.

At any rate, high-performance computing hardware could work much more efficiently if the constraint to always calculate the exact answer could be weakened. A number of studies recently investigated the possibility to reduce numerical precision in weather simulations so as to lower computing costs and energy consumption. Savings can be reinvested to increase model resolution or the

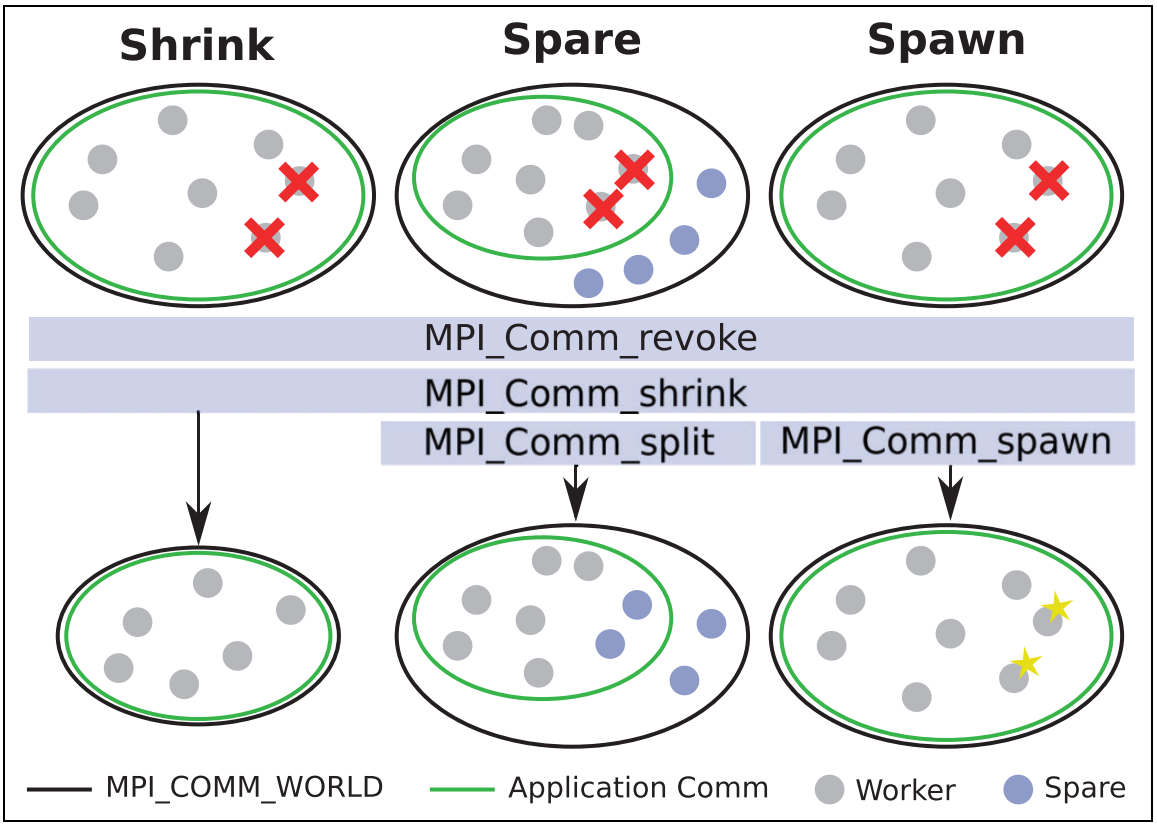

Figure 1. Approaches to communication-channel recovery using user-level failure management extensions proposed for MPI 4.x.

number of ensemble members for ensemble forecasts to improve predictions. Studies have covered atmosphere models (Düben and Palmer, 2014) but also data assimilation (Hatfield et al., 2018) and other model components such as land surface (Dawson et al., 2018). It has also been argued that reduced numerical precision may not necessarily degrade results of simulations for weather predictions. In contrast, in some cases it may even be possible to use variability from rounding errors to improve ensemble simulations (Düben and Dolaptchiev, 2015).

A possible reduction in numerical precision has been investigated using *stochastic processors* (Düben et al., 2014b) and exact arithmetic, for example employing programmable hardware such as Field Programmable Gate Arrays (FPGAs, Russell et al., 2015). Stochastic processors reduce power consumption through voltage over scaling. Here, the voltage applied to the processor is reduced beyond the level at which all operations are calculated correctly. Results suggest that power consumption could indeed be reduced significantly if a low number of faults is acceptable for the applications (Düben et al., 2014a; Lingamneni et al., 2012; Sartori et al., 2011).

3.2. Hardware resilience

The exponential growth in the core counts of high-performance computing systems carries the potential to shrink fault-free computing cycles, dramatically so in some cases. The authors of Gamell et al. (2014) report a MTBF of 9–12 hours. Lower figures were identified for Blue Waters (Di Martino et al., 2015) and the Tianhe-2 machines (Chen et al., 2017). The authors of Gupta et al. (2017) present the normalized MTBF of five systems over 8 years (their Figure 1). As reported in Daly (2006), the MTBF of a system with one million nodes, each of which with a MTBF of 3 years, drops to about 94 seconds.

Published figures about machines used in weather centres are generally hard to come by. The two supercomputers at ECMWF, with a total 7220 Cray XC-40 nodes and a joint peak performance of more than eight petaflops, show about 15 node failures per months, including memory failures, CPU failures and software crashes but excluding preventive maintenance (Christian Weihrauch, personal communication; see also Barker et al., 2016; Bautista-Gomez et al., 2016; Gupta et al., 2017; Schroeder and Gibson, 2009, for more experimental resilience assessments).

Reliability, availability and serviceability (RAS) features of an HPC system include all the components required to keep the system functional for long periods without failures. In HPC, it is generally recognized that the overhead cost associated with permanent RAS features should be kept low compared to the potential impact of computational nodes loss.

The common hardware RAS features are deployed at different levels, from processors to the whole HPC system. Radojkovic et al. (2020) surveyed and set priorities for the

implementation of lower-level RAS features, focusing on CPU, memory, intra-node (socket-to-socket) interconnect and emerging FPGA-based hardware accelerators. A 'must-have' features list in production HPC systems resulted based on three main criteria:

1. Resilience features should ensure that the failure rate of the system stays below an acceptable threshold depending on the deployed technology, system size and target application;
2. Given the high cost of uncorrected errors, frequent hardware errors should be corrected, at low cost, preferably on hardware;
3. HPC systems should monitor the temperature of their components and include mechanisms that prevent overheating, one of the main causes of unreliable device behaviour, while balancing power/energy and performance.

More specifically, essential resilience features according to Radojkovic et al. (2020) include, at the processor and accelerator level, error detection in CPU caches and registers, memory thermal throttling to slow down memory access rate and dynamic voltage and frequency scaling (DVFS). At the memory level, ECC should be included in main and accelerator memory and a mechanism to identify and correct memory errors should be introduced. At intra-node level, packet retry should be deployed in the intra-node interconnect, mostly based on cyclic redundancy check (Peterson and Brown, 1961; Ramabadran and Gaitonde, 1988), and PCIe should be included among the standard RAS features. At inter-node level, stacking network switch allowing redundant network links using Link Aggregation Control Protocol (LACP) is recommended, while at storage level, Redundant Arrays of Inexpensive Disks (RAID, see, e.g. Patterson and Hennessy, 2016) and evolutions such as Declustered RAID, see Qiao et al. (2019), should feature.

Note that some techniques have a larger positive or negative impact on performance than others. For example, ECC and DVFS features lower the performance while LACP can improve performance as it takes advantage to the redundant hardware.

Redundancy is the approach commonly used at the higher infrastructure level, where management of the redundant resources is especially crucial. At the whole system level, a solution is to have two or more HPC facilities physically apart, as done at ECMWF, Météo-France, and the UK Met Office, for example. This method can include also the low-level RAS features.

Case study: The BullSequana XH2000 system

For concreteness, we consider in detail a set of high-level hardware redundancy techniques for power, cooling, management and high-speed interconnect implemented in a real example, the BullSequana XH2000. This system is relevant for the purposes of NWP applications, since it will be installed at three major weather and climate centres in Europe (ECMWF, Météo-France and DKRZ), as well as at supercomputing centres in Europe and beyond, such as CINECA in Italy, GENCI in France, CSC in Finland and C-DAC in India.

In the BullSequana XH2000 there is redundancy at the level of power, cooling, management and high-speed interconnect. For power redundancy, in the XH2000 design the AC/DC conversion within a rack is shared for all resource elements (compute node, switch, hydraulic chassis, etc.). The power section is composed of a power distribution unit (PDU), power supply unit (PSU) shelves, optional ultracapacity module (UCM) and a busbar to distribute power to all the components within the cabinet. The level and type of redundancy is configurable for the PSU shelves, it could be configured at the PSU block or at the PSU shelf level with the following types of configuration: N (no redundancy), $N+1$, $N+2$, and $2N$ redundant. The optional UCM chassis allows the mitigation of micro power outages up to 300 ms at full load when three-phase uninterruptible power supply equipment for system shutdown is not present upstream in the data centre infrastructure. Some HPC centres use an uninterruptible power supply (UPS) so as to maintain the power supply for longer in order to start another power source or properly stop the systems. This method (UPS plus another power source) is so costly with respect to the power at stake that it is rarely deployed. Indeed, the high electrical power demand of current HPC systems is not compatible anymore with the densification and power efficiency imperatives required.

The BullSequana XH2000 Direct Liquid Cooling system is composed of hydraulic chassis (HYC), primary and secondary manifolds, and an expansion tank. The HYCs contain the heat exchanger system that allows it to achieve 95% of heat transfer between the primary and secondary manifolds. Up to three HYC are available depending on the redundancy type desired. To protect the system against cooling lost at rack level and to avoid stopping the production, the rack is immediately put in low power mode to lower its power consumption, and thus its dissipation. In case of persistent failure inducing a temperature rise, the system shuts down automatically.

The network management of an XH2000 rack is based on a redundant network switch stack (one redundant switch for each switch) and redundant network links using LACP, ensuring a minimum redundancy. Moreover, in case of a cell failure, the cell-based architecture prevents from an impact on the rest of the system. On the high-speed interconnect side, in case of node or switch loss, the Infiniband standard allows for recomputing the routing tables in order to find alternative routes and keep the system running by isolating the failed resources.

3.3. Application-level resilience

Application-level resilience techniques take a software-based approach, often with the focus on handling hard

faults that result in the complete loss of one or more MPI processes. The objective is protecting the entire application from permanent failures in the computing resources and enabling it to continue, potentially on a reduced set of resources. As a consequence, application-level resilience may target specific types of software using approaches which cater to a subset of software.

The most basic form of resilience is replication, whereby calculations are performed in duplicate on multiple nodes. This allows immediate continuation of the calculation in the event of a failure, but with a very inefficient use of resources during normal forward-path execution. Process replication has been explored as a simple and effective approach to resilience (Guerraoui and Schiper, 1997; Mohamed, 2016), although it is highly wasteful of resources during fault-free execution.

As noted in Section 2, CPR to stable storage is the classic application-level resilience methodology used to avoid complete re-execution of long-running simulations in the event of faults. Application-based checkpointing is common among production codes, particularly in NWP, allowing targeted data structures to be protected without wasting resources in protecting auxiliary data which can otherwise be reconstructed. In the following, we survey a number of recent developments that have been proposed in order to improve the efficiency of the CPR technique at the system level.

3.3.1. User-level failure mitigation and advanced checkpointing. Rather than forcing a restart of the entire application, localized mitigation of the failure is preferable. One particular challenge in the context of MPI applications has been the difficulty in recovering MPI communication in the event of hard faults. User-level failure mitigation (ULFM) extensions to the MPI standard provide an application-driven mechanism to detect and recover communication channels in the event of a hard fault leading to the loss of one or more processes. These extensions are currently under consideration for inclusion in the MPI 4.x standard (Losada et al., 2020). In prior versions of MPI, communication routines would either trigger an immediate abortion of the program or return control to the application to support a more controlled termination. With the ULFM APIs, application programs are allowed to revoke failed MPI communicators – alerting all surviving processes to the failure occurrence – and subsequently shrink them to exclude all failed processes and restore functionality of the communicator. If supported by the computational resource, additional processes may also be spawned to replace those which have failed. These strategies are illustrated in Figure 1. Although not yet part of the standard, the proposed ULFM extensions have already been applied in a number of studies (see, e.g. Ali et al., 2014; Ashraf et al., 2018; Bland et al., 2013; Cantwell and Nielsen, 2019; Engwer et al., 2018; Fagg and Dongarra, 2000; Gamell et al., 2017a, 2017b; Losada et al., 2020; Teranishi and Heroux, 2014).

Following the restoration of communicators, the application must be returned to a consistent state to continue execution, including restoration of data structures on any replacement processes. This requires mechanisms for both protection and restoration of critical data structures.

In-memory checkpointing avoids writing to the parallel file system. While local memory-based approaches provide excellent performance for application faults, they fail to provide resilience against more serious hardware failures. Remote in-memory checkpointing places checkpoints in the memory of a remote node through a pairwise communication, retaining the performance benefits of avoiding parallel file system access, but at the cost of increased network traffic. Upon failure of one or more processes, remote checkpoints can either be written to disk for use with traditional CPR, or restored directly after ULFM communicator recovery for a *checkpoint-mitigate-rollback* approach.

Data transfer over the network is the main performance bottleneck of remote in-memory checkpointing. Depending on the volume of state data which must be protected per process, the volume of memory available on remote nodes for storing checkpoint data is also a concern. Increasing prevalence of fast NVRAM has addressed this issue to some extent (Kannan et al., 2013; Mittal and Vetter, 2016). Nielsen (2018) considered the application of Reed-Solomon encoding to reduce the storage requirements of checkpoints, storing only checksums in addition to the local data, at the cost of additional computation. An alternative strategy to protect the state of an application is to log communications between processes, thereby allowing recovery of one or more processes to be performed in isolation. Message logging avoids the need for surviving processes to rollback, but the size of message logs may grow significantly, depending on the communication pattern of the application.

Mitigation of hard failures and implementation of application-level resilience techniques generally requires modification of existing codes to varying degrees. A number of libraries and packages already exist to facilitate this effort. Their impact ranges from almost zero intrusion through to a complete rewriting of the source code to incorporate resilience.

ACR (Ni et al., 2013) implements Automatic Checkpoint/Restart using replication of processes in order to handle both soft and hard errors. FA-MPI (Hassani et al., 2014) proposes non-blocking transactional operations as an extension to the MPI standard to provide scalable failure detection, mitigation and recovery. Finally, FT-MPI (Fagg and Dongarra, 2000) was a precursor to ULFM for adding fault tolerance to the MPI 1.2 standard. These approaches, as well as Fenix detailed below, generally require significant intrusion into the application code to mark data structures to be protected and to implement the recovery process.

3.3.2. Minimally intrusive approaches. A less intrusive approach (Cantwell and Nielsen, 2019), targeting time-

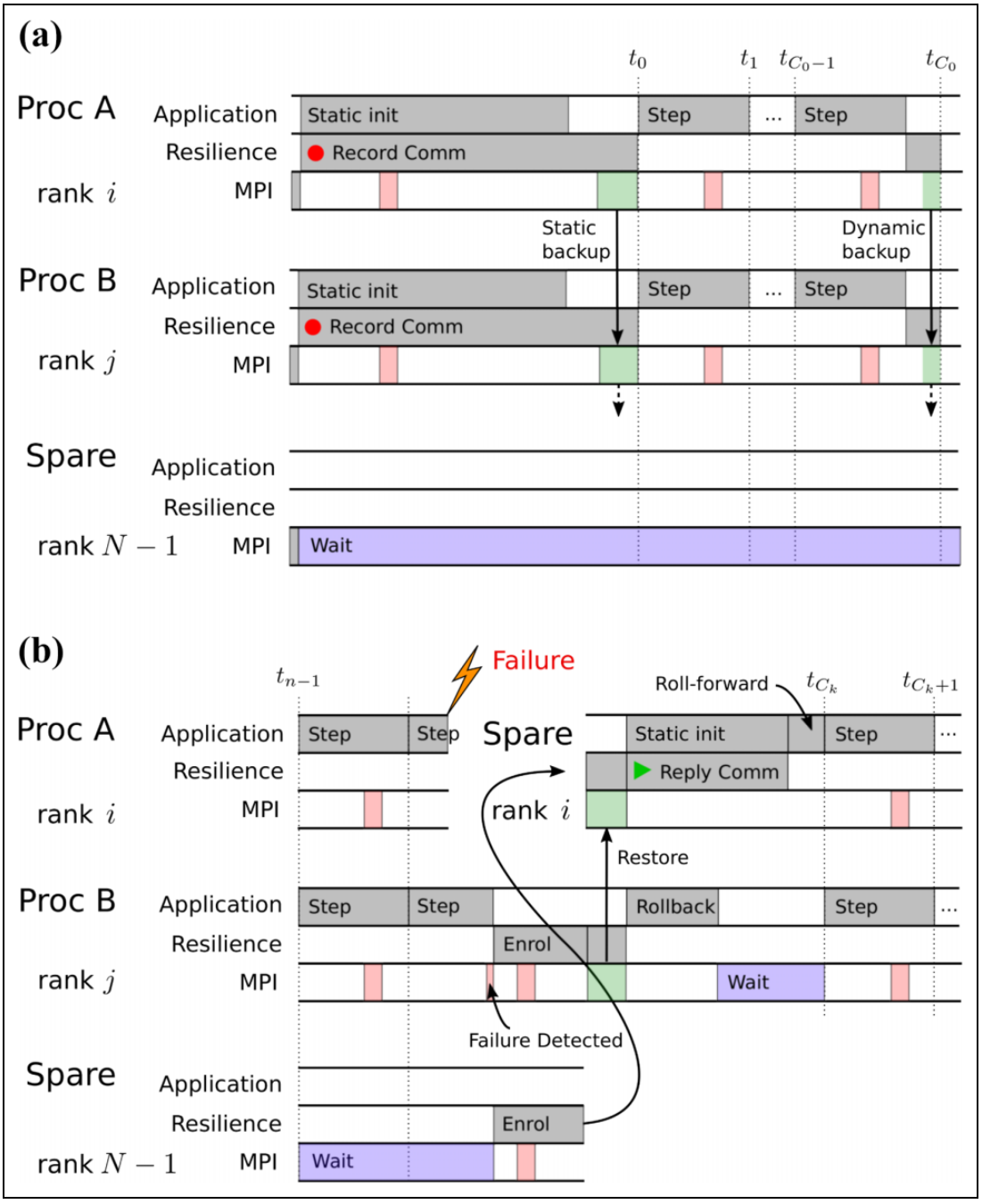

Figure 2. A low intrusive approach to application-level resilience by splitting the algorithm into a static and dynamic phase (Cantwell and Nielsen, 2019). (a) All communication is logged during the static initialization phase by intercepting MPI function calls. Remote in-memory checkpointing is used during the dynamic time-integration phase. (b) Following a failure, spare ranks are used to replace failed ranks. These ranks follow the normal code path but the results of MPI calls are replayed from the message log.

dependent solvers, combines message-logging techniques and remote in-memory checkpointing to reduce this burden in often complex object-oriented applications. It exploits the nature of these codes by transparently logging MPI messages during the relatively short initialization phase of the application, when the data structures remain static, thereby allowing recovering processes to follow the normal unmodified program to reach the time-integration phase. For the time-integration phase, in which the solution data change frequently, remote in-memory checkpointing is used to maintain scalability. This approach is illustrated in Figure 2.

3.3.3. Fenix and MPI_Reinit. Fenix is a framework enabling MPI applications to recover nearly transparently from losses of data or computational resources manifested as observable errors. It is based on the premise that the MPI standard itself provides facilities for trapping and isolating

such errors, allowing the application to retain control of remaining unaffected resources and obviating full program restart. Building on the success of Fenix's prototype implementation (Gamell et al., 2014; Teranishi and Heroux, 2014), a significant extension has been made that provides APIs for online application rollback and data recovery together and serves many different application needs and styles with formal specifications (Gamell et al., 2016). For process recovery, `Fenix_Init()` specifies the point to rollback when failure is detected and the recovery schemes for the lost MPI ranks. For data recovery, Fenix provides several data binding and recovery APIs that leverages the internal information of Fenix's process recovery interface for efficient data recovery. The current implementation of Fenix leverages ULFM, but the specification does not require it. Recently, Fenix has been modified to support asynchronous application recovery leveraging an on-node asynchronous many-task parallel runtime (Paul et al., 2020) to eliminate the need for online rollback of all MPI ranks.

MPI_ReInit (Georgakoudis et al., 2020; Laguna et al., 2016) is a similar approach, but it does not expose any low-level APIs like MPI-ULFM. Instead, it focuses on efficient online rollback recovery, simplifying the low-level fault detection and notification mechanism accommodated by MPI-ULFM. MPI_ReInit does not specify any data recovery schemes, allowing the use of external software. The latest implementation (Georgakoudis et al., 2020) demonstrates better scalability than MPI-ULFM in the absence of failures. On data recovery for MPI programs, Global View Resilience (GVR, Chien et al., 2015) and VeloC (Nicolae et al., 2019) accommodate generic APIs for data persistency. Both Fenix and MPI_ReInit can leverage a combination of these, or others. Fenix chiefly aims at providing fast and in-memory redundant storage for efficient data recovery, whereas GVR and VeloC target secondary storage for data capacity. Nevertheless, VeloC addresses the efficiency issues using multi-level checkpointing that utilizes on-node memory and burst buffer (special staging I/O nodes), achieving scalable and asynchronous checkpointing for the state-of-art HPC platforms.

3.3.4. Kokkos. Kokkos (Bertagna et al., 2019) is a C++ library designed as a performance-portable node-parallel programming model to allow platform-independent parallel implementations across multiple heterogeneous architectures. The main idea of resilient Kokkos is extending the abstraction of parallel computation and data representation to support redundancy. These new features are enabled through template parameters of parallel Kokkos loops and view features to realize redundant program execution and CPR handling both soft and hard error resilience. During program execution, the resilient Kokkos runtime monitors all active resilient view instances and automatically copies the data to the persistent storage as needed. For program recovery, Kokkos exploits the existing I/O and checkpoint library facilitated with annotation capability for Kokkos parallel loops and view instances to locate the point of failure as well as the data objects being lost. Currently, resilient Kokkos provides C++ I/O, HDF5 and VeloC (Nicolae et al., 2019) for the persistent storage options and supports OpenMP and CUDA backend for the execution space. The reader is referred to Miles et al. (2019) for more details on Kokkos resilience capabilities. Of note for atmospheric applications, the E3SM climate model (Golaz et al., 2019) is currently being rewritten in C++/Kokkos and can exploit these features.

3.3.5. Minimal ULFM and fallback. This approach is a much more lightweight wrapper around the ULFM specification, specifically targeted at C++ applications that intend to react to node losses and silent data corruption via C++ `exceptions` (Engwer et al., 2018). The wrapper ensures that in case of a failure an exception can be received on all surviving ranks if it happens inside a guarded block. The term surviving here means, that the rank is capable to continue the computation. We rely on two methods from the ULFM proposal: `MPIX_Comm_revoke`, which revokes the communicator for any communication and `MPIX_Comm_agree` to agree on the error state. Once a rank calls a communication method on a revoked communication an error is raised which is then mapped to an exception. A working communicator can be recovered by calling `MPIX_Comm_shrink` and its siblings, on which the computation can be continued, after the error state has been resolved. This functionality is typically not available in default MPI installations on clusters yet, and thus, such a wrapper can be convenient in a transition phase.

3.3.6. Quality of service. In addition to the system reliability, quality of service (QoS) is increasingly important for NWP to guarantee the prediction code to complete within a reasonable time window. Recently, several papers pointed out the performance variability of large scale HPC systems due to multiple factors. For instance, Bhatele et al. (2013, 2020), Mubarak et al. (2017) and Yang et al. (2016) observe performance interference by multiple concurrent jobs. This is due to the fact that most HPC systems allow multiple jobs to share interconnect network and I/O subsystem for various types of data transfers. In particular, the data traffic generated by checkpointing could impact the performance of message passing and file system accesses of other applications. This performance variability can be mitigated by a sophisticated job placement that considers application specific communication and I/O accessing patterns, but the feasibility of this solution is unclear due to a wide variety of application usages and the sheer complexity of HPC system architecture further aggravated by the manufacture variability of hardware components. Inadomi et al. (2015) observe uneven thermal distributions across the nodes in a single large scale HPC system, causing more than 20% performance degradation. Bhatele et al. (2020) claim that it is possible to apply machine learning to train communication and I/O access patterns of a known workload to optimize job scheduling and placement. Despite the

improvement of the QoS for network, I/O and job schedulers in the near future, the workload and performance variability of future NWP systems should be characterized to adapt the entire system configuration to the weather prediction specific workload.

3.3.7. Backup grids for resilient NWP models. To the best of our knowledge, there has only been a single study to test an approach to secure weather and climate models against hardware faults on a software level (Düben and Dawson, 2017). The approach uses a backup grid to store coarse-resolution copies of prognostic variables. In the presence of a hardware fault, an estimate of the original values of the model fields could be reproduced from the backup grid to enable the simulation to continue with no significant delay. The approach could not reproduce a bit-identical result when compared to a fault-free simulation but it could allow the completion of a simulation in the presence of both soft and hard faults.

The backup system uses the following mechanism (Düben and Dawson, 2017):

- The prognostic variables from the model grid are mapped onto the coarse-resolution backup grid at the end of each time step. The values of the backup grid are stored for one time step to allow a comparison with the model fields at the following time step.
- It is checked whether the fields on the backup grid have changed by an unexpected amount during a time step. The threshold of this check needs to be tuned to the specific model simulation under consideration. If the change of a model variable is suspicious, it is assumed that a hardware fault has perturbed the simulation. Therefore, the specific value on the backup grid is replaced by the corresponding value from the previous time step.
- The corresponding values on the model grid that influenced the erroneous grid value on the backup grid are checked for values outside of a physically meaningful range.
- If the value on the model grid is found to be unreasonable, it is replaced by the value mapped from the backup grid onto the specific position of the model grid (Figure 3).

The approach was tested on numerical simulations with a two-dimensional shallow water model, a standard test bed for numerical schemes in NWP model development. As long as the backup system was used, simulations did not crash and a high level of solution quality could be maintained. The overhead due to the backup system was reasonable with a 13% runtime increase (Düben and Dawson, 2017). Additional storage requirements were small.

However, there are a number of limitations to the approach that would require further research. The backup system, as used in Düben and Dawson (2017), is not able to distinguish between model errors and hardware errors, and it violates local mass conservation and global energy conservation.

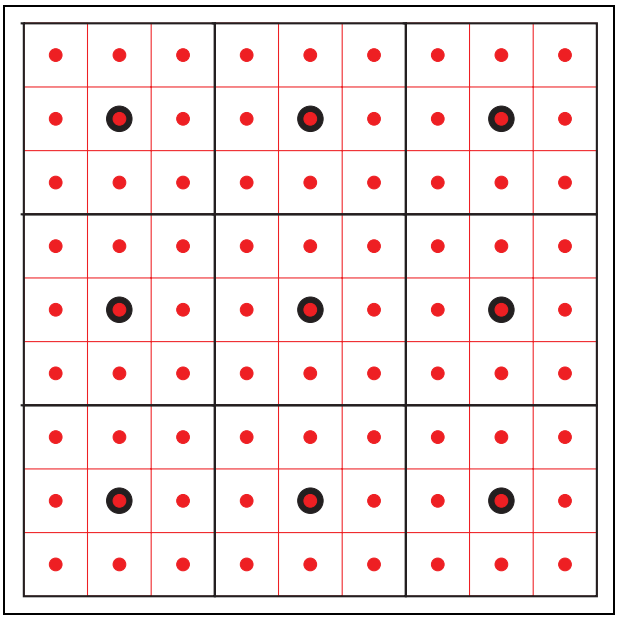

Figure 3. Backup system of Düben and Dawson (2017). The coarse-resolution backup grid (black grid points) holds an approximated representation of the prognostic variables on the fine-resolution grid (red grid points). If a hardware fault is detected, the prognostic fields on the model grid can be reproduced as an approximation from the backup grid or vice versa.

3.4. Algorithmic resilience: Recovery-restart for sparse linear solvers

Numerical models as the ones used in NWP intensively rely on the solution of sparse linear systems that consume a significant part of the simulation time. In this section we focus on possible numerical mitigation techniques to make these kernels resilient. We assume that the core of the application data, that are not updated by the linear solver, are asynchronously checkpointed using appropriated checkpointing techniques by the application before calling the linear solver. An agile combination of the various techniques described in this paper has to be considered to make the full application resilient.

Algorithmic fault tolerance approaches aim at supplying numerical methods with techniques to deal with hard or soft faults. In general, these approaches modify the theoretical formulation and the implementation of the non fault-tolerant version of the methods. Some methods, for example multigrid, structurally lend themselves well to these modifications. The effectiveness of these approaches can be evaluated based on the trade-off between fault tolerance performance on one side and the scope of changes needed for fault-proofing and related computational overhead on the other.

Of particular interest in the NWP context are fault-tolerant versions of iterative algorithms for linear systems

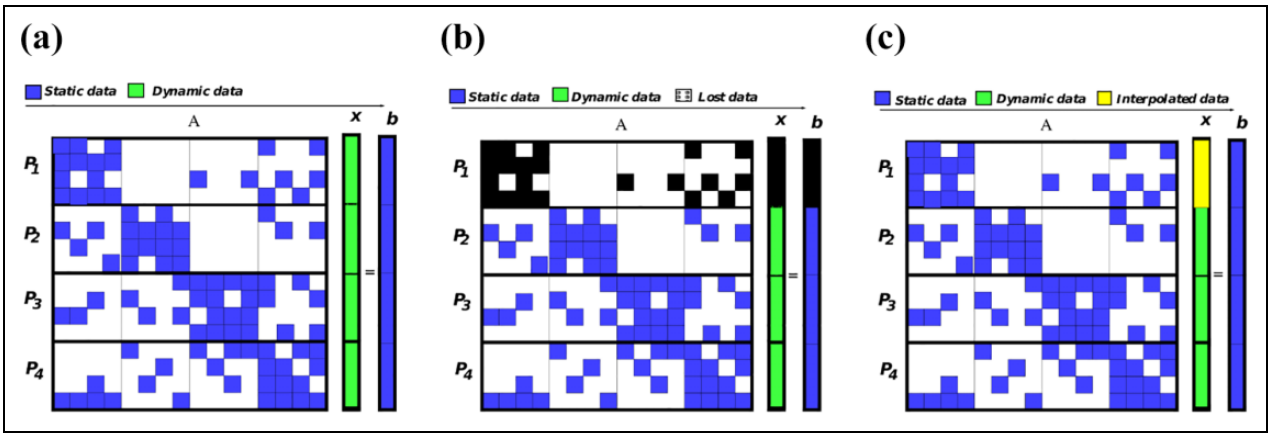

Figure 4. General interpolation scheme: (a) before fault, (b) faulty iteration and (c) interpolation. The matrix is initially distributed with a block-row partition, here on four nodes (a). When a fault occurs on the node P_1, the corresponding data are lost (b). Whereas static data can be immediately restored, dynamic data that have been lost cannot. We investigate numerical strategies for regenerating them (c).

solution. Two examples are given below that can be applied when the fault translates to data that are lost and the location of the lost data is known. We make the assumption that a mechanism is able to notify the solver which data are lost, assuming a separate mechanism for fault detection. In the current software infrastructure, this corresponds well to node failure where all the data stored on the node are lost and MPI is able to detect the failure of the process running on that node. We describe numerical mitigations that can be implemented to ensure the continuation of the solution process with possibly some delay in the convergence compared to the non-faulty case. After data extraction, a well-chosen part of the missing data is regenerated through any of the described strategies, and is then used as meaningful input to restart the iterative scheme.

A non-singular linear system $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$ can be approximately solved by successive application of the Richardson iteration (see, e.g. Quarteroni et al., 2010)

$$x^{(k+1)} = x^{(k)} + \left(b - Ax^{(k)}\right) \tag{1}$$

where $x^{(0)}$ is an appropriate initial guess and k the iteration index.

We consider the formalism proposed in Langou et al. (2007), where data losses are classified into three categories: *computational environment*, *static* data and *dynamic* data. When a node crashes, all available data in its memory are lost. The computational environment consists of all the data needed to perform the computation (program code, environment variables, etc.). Static data are set up during the initialization phase and remain unchanged during the computation. In the case of the solution of a linear system, they correspond to the input data to the problem and include in particular the coefficient matrix A and the right-hand side vector b. Dynamic data are those whose value may change during the computation, that reduces to the iterate $x^{(k)}$ for a single fixed-point iteration scheme. For this section we are only dealing with dynamic data recovery.

Let N be the number of partitions, such that each block-row is mapped to a computing node. For all $p, p \in [1, N]$, I_p denotes the set of row indices mapped to node p. With respect to this notation, node p stores the block-row $A_{I_p,:}$ and x_{I_p} as well as the entries of all the vectors involved in the solver associated with the corresponding row indices of this block-row. If the block A_{I_p,I_q} contains at least one nonzero entry, node p is referred to as neighbour of node q, as communication will occur between those two nodes to perform a parallel matrix-vector product.

Figure 4 shows a block-row distribution on four nodes, with static data (matrix and right-hand side, blue) and dynamic data (the iterate, green) associated with the linear system. If node P_1 fails, the first block-row of A as well as the first entries of x and b are lost (in black in Figure 4). The corresponding matrix block-row as well as the portion of the right-hand side can be restored thanks to the checkpoint performed by the application. By contrast, the dynamic iterate is permanently lost and must be recovered.

A well-suited initial guess for the restart of the iterative solver can be recovered by interpolating available data. Two fully algebraic interpolation policies – recomputing the entries of the faulty iterate based on data living on surviving nodes and a compressed backup-based policy – are considered. These interpolation policies require the solution of either a small linear system or a small least-squares problem. The schemes are designed at application level and require no or only very moderate extra computational units or computing time when no fault occurs.

Because the primary interest here lies in the numerical behaviour of the schemes, we make strong assumptions on the parallel environment. In particular, when a fault occurs we assume that the crashed node is replaced and the associated computational environment and static data are restored (Langou et al., 2007) thanks to some checkpointing mechanism.

3.4.1. Interpolation-restart. Interpolation-Restart (IR) techniques are designed to cope with node crashes (hard faults) in a parallel distributed environment (Agullo et al., 2015, 2017, 2016a, 2016b). The methods can be designed at the algebraic level for the solution both of linear systems and of eigenvalue problems.

The idea consists in interpolating the lost entries of the iterate using interpolation strategies adapted to the linear systems to be solved. The interpolated entries and the current values available on the other nodes define the recovered iterate which is then used as an initial guess to restart the Richardson iteration.

A first interpolation strategy, introduced in Langou et al. (2007), consists in interpolating lost data by using data from surviving nodes. Let $x^{(k)}$ be the approximate solution when a fault occurs. After the fault, the entries of $x^{(k)}$ are known on all nodes except the failed one. This Linear Interpolation (LI) strategy computes a new approximate solution by solving a local linear system associated with the failed node. If node p fails, $x^{(LI)}$ is computed via

$$\begin{cases} x_{I_p}^{(LI)} = A_{I_p,I_p}^{-1}(b_{I_p} - \sum_{q \neq p} A_{I_p,I_q} x_{I_q}^{(k)}), \\ x_{I_q}^{(LI)} = x_{I_q}^{(k)} \qquad \text{for } q \neq p. \end{cases} \tag{2}$$

This basic idea can be adapted to more sophisticated linear solvers based on Krylov subspace methods (Agullo et al., 2016a; Langou et al., 2007) or to eigensolvers (Agullo et al., 2016b) and more robust interpolation can be designed to tackle the possible singularity of the diagonal block A_{I_p,I_p}^{-1}. We refer to Section 4.1 for a more detailed description of these ideas in the context of the GMRES method.

Finally, we note that when the linear system or eigenproblem arises from the discretization of a partial differential equation (PDE), each node of the parallel platform usually handles the unknowns associated with a subdomain. In the context of a linear system solution, the interpolation policy can be viewed as the solution of the PDE on the failed subdomain with Dirichlet boundary conditions defined by the current values on the interfaces with the neighbouring domains. In that context, more sophisticated boundary conditions can be considered as presented for example in Huber et al. (2016).

3.4.2. Checkpoint-restart using lossy compression. Checkpointing techniques offer another opportunity to recover locally lost data. Although checkpointing and classic CPR techniques in general introduce a significant overhead on computation and bandwidth, utilizing lossy compression for local backups of the dynamic data can enable checkpoint based recovery approaches.

Contrary to lossless compression methods, e.g. *zip*, *png*, *gif* and others, lossy compression like *mp3* or *jpeg* neglects some information to increase the compression factor. Therefore, the decompressed data differ from the input data. However, since iterative solvers and preconditioners are by definition not exact, the accuracy loss due to compression can in fact be tolerated up to a certain degree. The loss can be qualified as benign if the compression error is smaller than the error within the solver, for example the residual error accepted by some stopping criterion, and than the model numerical truncation error. In this case, the decompressed data can actually lead to results with similar quality as data from a lossless compression method with the advantage of a smaller size.

The efficient implementation of such compressed checkpointing crucially relies on ULFM or similiar system-level techniques, see Section 3.3, and on improvements to classical CPR. Compressed checkpointing can be used in the node loss scenario, but also in case of soft faults such as SDC. In the following, we describe some ideas for the former, more specifically for in-memory compressed checkpoints, and refer to Altenbernd and Göddeke (2018) for the SDC case. Mathematical and conceptual details can be found in Göddeke et al. (2015), and more details on implementation issues in Engwer et al. (2018). The key point is that, by combining compression techniques for backup creation with local-failure local-recovery (LFLR, Teranishi and Heroux, 2014) approaches for iterative solvers, e.g. multigrid preconditioners, the overhead in fault-free scenarios can be reduced significantly compared to classical CPR techniques, while still providing a speed-up for recovery in the presence of hard faults or data loss.

Recovery can be further improved by storing additional information in the backup like the search direction for the conjugate gradient method. This can be especially useful if many data losses occur. In the best case, the number of iterations until convergence of the fault-prone solver before and after the restart adds up to the same amount as in the fault-free scenario. The generation of the initial guess based on the backup data can be done in multiple ways which we describe later.

We evaluate three techniques to create compressed backups: zero backup, multigrid, and SZ compression (Di and Cappello, 2016; Liang et al., 2018; Tao et al., 2017). In the zero backup technique, lost data are reinitialized with zeros. Multigrid can be interpreted as a lossy compression technique, with a number of mathematical peculiarities that need consideration (Göddeke et al., 2015). Multigrid algorithms use a hierarchy of grids to solve linear systems in an asymptotically optimal way. This hierarchy can be used to restrict, i.e. lossily interpolate, the iterate from fine to coarse grids. Such lower-resolution representation of the iterate can then be stored as a compressed backup. Later, the multigrid prolongation (coarse-to-fine grid interpolation) operator is used to decompress the data. Obviously, the quality of the backup is strongly dependent on the grid where it is restricted to. This compression technique can easily be applied if a multigrid solver or preconditioner is used anyway, because the operators are readily available. Furthermore, this principle can be used in combination with other hierarchic methods.

SZ compression, on the other hand, does not use operators from a specific solver or preconditioner but can be applied to all kinds of field data, although it prefers, by construction, structured data ideally associated with a structured grid. Another important feature is that the compression accuracy can be prescribed and adapted to the situation. Unfortunately, a higher compression accuracy usually leads to a lower compression factor and higher compression time, which is crucial in terms of resilience overhead.

4. Application of resilience methodologies to NWP

This section contains illustrative applications of the fault-tolerant algorithmic techniques described in Section 3.4. In 4.1–4.2, we illustrate the robustness of two numerical mitigation approaches that can be implemented in linear solvers if they receive notification of possible data loss by some layers of the system stack. The notification corresponds to node failure that can be detected by the MPI runtime and notified to the still alive processes. In 4.3, a more holistic technique is investigated that addresses both the detection and the mitigation of faults in the context of soft errors (e.g., bit flip in some data). Note that the GCR solver considered in 4.3 is mathematically equivalent to GMRES, the Krylov solver used in 4.1 in the context of node failure. We should point out that these two techniques are complementary, since they address faults of different natures and could be combined either in the context of GCR or GMRES to increase the resiliency of both numerical linear solvers.

4.1. Resilient GMRES with interpolation-restart

The GMRES method is one of the most popular solvers for the solution of non-symmetric linear systems. It belongs to the class of Krylov solvers that minimize the 2-norm of the residual associated with the iterates built in the sequence of Krylov subspaces (MINRES, Paige and Saunders, 1975, is another example of such a solver). In contrast to many other Krylov methods, GMRES does not update the iterate at each iteration but only either when it has converged or when it restarts every m steps in the so-called restarted GMRES (GMRES(m), Saad and Schultz, 1986).

Most of the parallel GMRES implementations rely on a block-row partitioning of the matrix (Balay et al., 2019) that induces a similar distribution of the Krylov orthonormal basis, while the small upper Hessenberg matrix resulting from the Arnoldi procedure is replicated. The memory footprint and extra local computational cost of this replication are negligible, and it avoids extra expensive global communications.

When a node crashes, the approximate solution is not available. The Hessenberg matrix is replicated on each node and the least-squares problem is also solved redundantly. Consequently, each individual node ℓ still in operation can compute its entries I_ℓ of the iterate when a fault occurs. Following the general idea of the IR policy, the lost entries of the current iterate can be approximated by solving a local linear system defined by equation (2) or a more robust recovery technique, referred to as LSI (Least-Squares Interpolation) that solves the local least squares. If node p fails, $x^{(LSI)}$ is computed as follows:

$$\begin{cases} x_{I_p}^{(LSI)} = \underset{x_{I_p}}{\operatorname{argmin}} \parallel (b - \sum_{q \neq p} A_{:,I_q} x_q^{(k)}) - A_{:,I_p} x_{I_p} \parallel, \\ x_{I_q}^{(LSI)} = x_{I_q}^{(k)} \qquad \text{for } q \neq p. \end{cases} \tag{3}$$

In addition to removing the assumption on the non-singularity of the A_{I_p,I_p} block made to define the LI policy, the LSI strategy ensures the monotonic decrease of the residual norm, a key property of GMRES.

To study the numerical features of the proposed IR strategies, we display the convergence history as a function of the iterations (Figure 5), that also coincide with the number of preconditioned matrix-vector products.

The recovery policy can induce delay in the convergence for two reasons. The first source of possible delay is the quality of the interpolation, the second is related to the restart that is performed after a fault. To distinguish between the effect of the two sources of delay, we implement a GMRES where we do not inject faults but impose a restart at each iteration the faulty run experienced a fault (it corresponds to a variable restarted policy for GMRES). We refer to this strategy as Enforced Restart (ER, yellow line in Figure 5). Furthermore, we also depict in red a straightforward strategy where the lost entries of the iterate are replaced by the corresponding ones of the first initial guess. This simple approach is denoted as Reset and corresponds to the zero backup in Section 3.4.2.

We solve the linear system $Ax = b$ with GMRES(100) and taking as A the matrix Averous from the Florida matrix collection (Davis and Hu, 2011), varying the volume of lost entries between 3% and 0.001% at each fault (a single entry is lost in the latter case). For these experiments, in order to enable cross comparison, the number of faults is kept the same and the faults occur at the same iteration for all the runs. The straightforward restarting Reset policy does not lead to convergence (Figure 5). Each peak in the convergence history corresponds to a fault showing that the solver is not able to converge. These characteristics are similar in other cases like multigrid methods for symmetric positive definite systems (Göddeke et al., 2015). In contrast, the IR strategies ensure convergence and have very similar convergence behaviour and robustness capabilities. Furthermore, the convergence behaviour of IR strategies appears scarcely affected by the amount of data loss.

In Figure 6, we investigate the robustness of the IR strategies when the number of faults is varied while the amount of recovered entries remains fixed at 0.2% after each fault. For those experiments, we consider restarted GMRES(100) to solve a linear system associated with the matrix Kim1 from the Florida matrix collection (Davis and

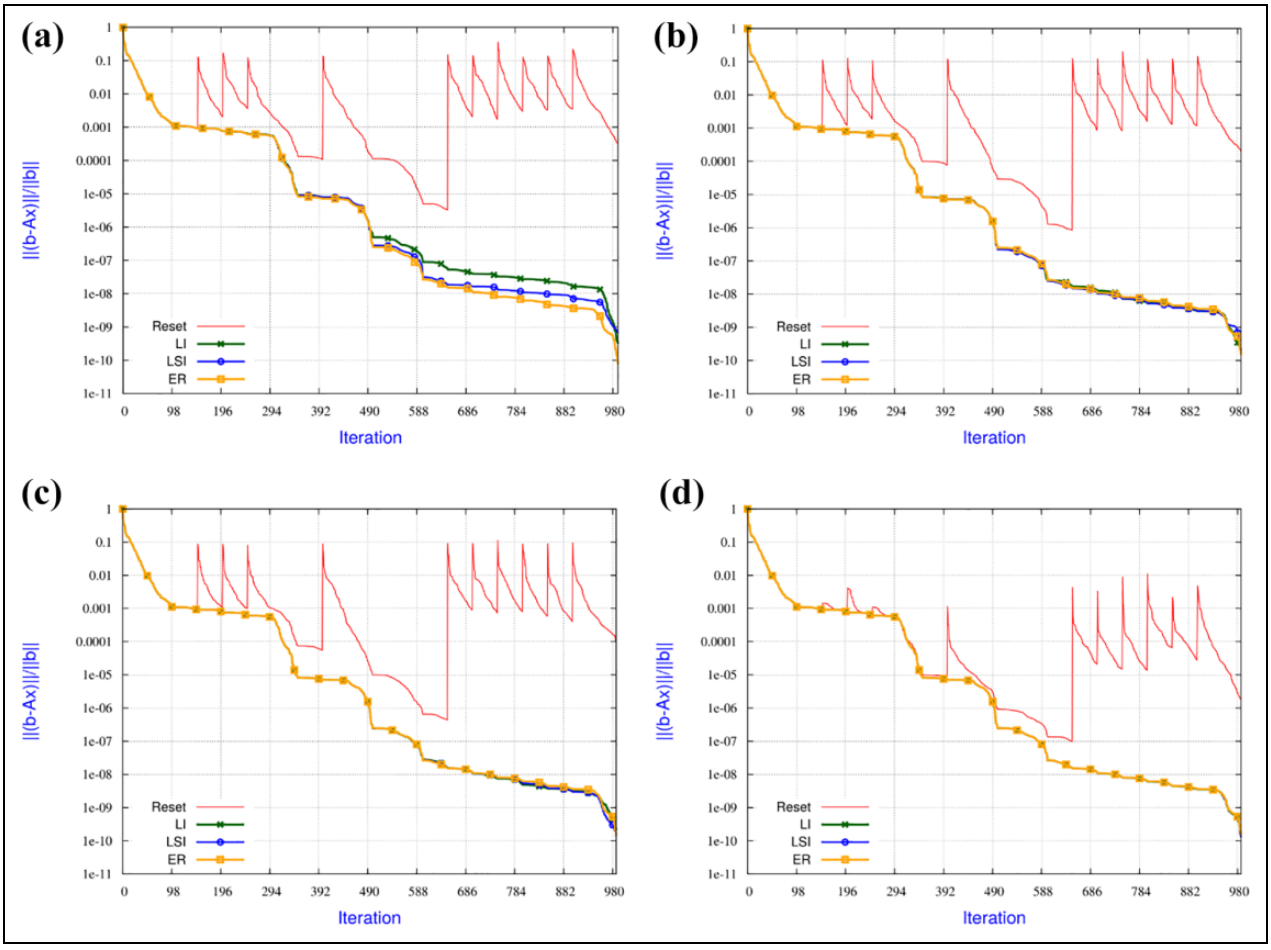

Figure 5. Numerical behaviour of the IR strategies applied to the GMRES(100) solution of the linear system with matrix Averous/epb3 and 10 faults when the amount of data loss is varied: (a) 3% data lost, (b) 0.8% data lost, (c) 0.2% data lost and (d) 0.001% data lost.

Hu, 2011). An expected general trend that can be observed on this example: the larger the number of faults, the slower the convergence. When only a few faults occur, the convergence penalty is not significant compared to the non-faulty case. For a large number of faults, the solver still converges, albeit more slowly. For instance, for an expected accuracy of 10^{-7}, the iteration count with 40 faults is twice the one without faults.

4.2. Compressed checkpointing

In order to showcase the backup and recovery options with the compressed checkpointing method described in Section 3.4.2, we consider the test problem

$$\begin{cases} -\nabla \cdot \left[\begin{pmatrix} 1 & 0 \\ 0 & 0.01 \end{pmatrix} \nabla u \right] = b & \text{in } \Omega = (0,1)^2 \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{4}$$

with b such that $u(x,y) = \sin(\pi x^2)\sin(\pi y^2)$ is the exact solution. This problem is solved within the DUNE PDE framework by a conjugate gradient method with an algebraic multigrid solver as preconditioner provided by the Iterative Solver Template Library (DUNE-ISTL, Blatt and Bastian, 2007). The problem is solved in parallel on 52 ranks using an overlapping Schwarz approach with minimal overlap (using the template `ISTLBackend_CG_AMG_SSOR` as solver from DUNE-PDELab, Bastian et al., 2010). The iterative solving procedure is stopped after a relative residual reduction of 10^{-8}. In the fault-free case the solver needs 115 iterations to fulfil this criterion for this particular demonstrator problem.

We suggest and evaluate three recovery techniques. The first, baseline approach mimics CPR and simply replaces the *global* iterate with its decompressed representation, independently of the compression strategy. The second approach follows the LFLR strategy and, similarly to interpolation-restart, re-initializes only the *local* data lost on faulty ranks by using backup data stored on neighbouring nodes. Unlike the baseline approach, this is purely local and only needs minimal communication to receive a remotely stored backup. In particular, the recovery procedure itself does not involve the participation of other ranks.

As a worst-case fallback when the backup data are not sufficient, we established a third *improved* recovery approach, which is still mostly local. An auxiliary problem is constructed, either by domain decomposition overlap or

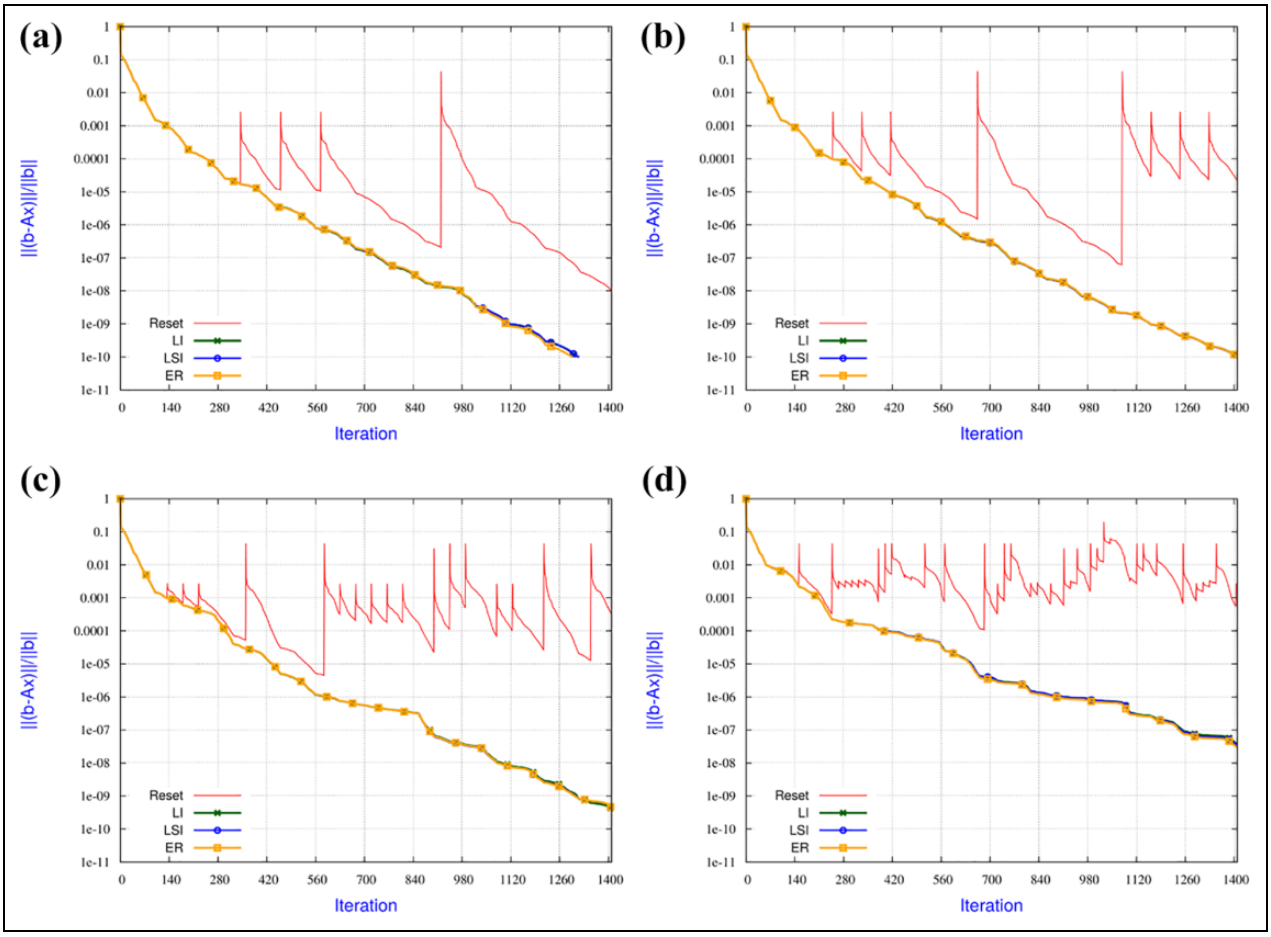

Figure 6. Numerical behaviour of the IR strategies applied to the GMRES(100) solution of the linear system with matrix Kim/kim1, varying number of faults, and fixed 0.2% data loss at each fault: (a) 4 faults, (b) 8 faults, (c) 17 faults and (d) 40 faults.

Table 2. Average number of iterations to convergence in the solution of the anisotropic diffusion problem (4) with faults, using different approaches for backup and recovery (see text for details).

	Zero	MG	SZ_1e-7	SZ_1e-3	aSZ_1e-2	aSZ_1e-1	aSZ_1e0	aSZ_1e1
global	200.00	200.00	154.00	193.00	116.00	115.00	119.00	120.00
local	199.33	194.67	134.33	173.00	114.33	115.67	115.67	114.67
improved	113.00	113.00	113.00	113.00	113.00	113.00	113.00	113.00
aux_iter	190.00	118.33	57.33	106.67	23.00	23.00	24.00	28.33

the operator structure, with boundary data from the neighbouring ranks, and solved iteratively with an initial guess based on the checkpoint data. This improves the quality of the recovered data and is closely related to equation (2) in Section 3.4.1, where a pure algebraic approach is considered. This approach can nearly always restore the convergence from the fault-free scenario independently of the used backup technique, only the number of additional local iterations varies (see also Altenbernd and Göddeke, 2018; Huber et al., 2016). It can be mandatory in highly-frequent fault scenarios to maintain convergence.

In addition to the recovery approaches, for backup we consider the aforementioned techniques of Zero Backup, Multigrid compression (MG, Göddeke et al., 2015), and SZ compression, as well as a fourth adaptive SZ compression (aSZ) technique. MG is configured to a fixed backup depth of two which reduces the data volume to $1/16$ in our 2D demonstrator. For standard SZ compression, we prescribe a compression accuracy of 1e-7 and 1e-3 as benchmarks for high and low accuracy. For adaptive SZ compression, the compression accuracy is coupled to the normalized local vector norm of the current residual multiplied by an additional tolerance $\texttt{tol}_{\texttt{aSZ}} \in \{1e-2, 1e-1, 1e0, 1e1\}$:

$$\frac{\| Ax^{(i)} - b\|_2}{\| b\|_2} \texttt{tol}_{\texttt{aSZ}}. \tag{5}$$

Table 2 shows the average number of iterations to convergence in the iterative solution of problem 4 when a

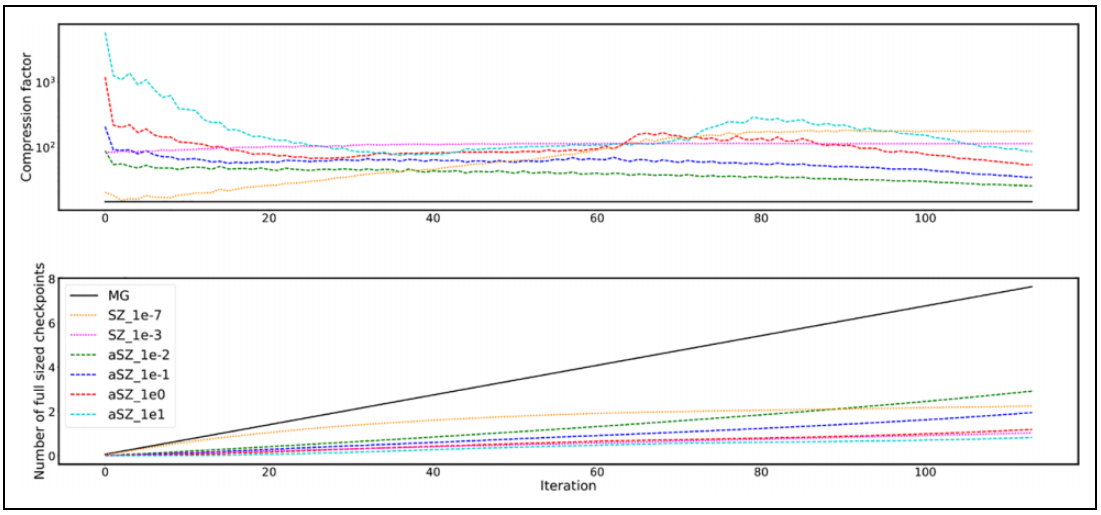

Figure 7. History of the average compression factor (top) and cumulative backup size compared to full-sized checkpoint (bottom) over the iterative process using different backup techniques in the iterative solution of the anisotropic diffusion problem (4) with faults.

fault happens in iteration $i \in \{10, 40, 75, 110\}$ on rank $r \in \{0, 21, 37\}$, using different combinations of recovery approaches (first three rows) and backup techniques. Furthermore, the last row (aux_iter) shows the number of additional iterations for the local auxiliary problem when the improved recovery is applied, using the different backups as initial guesses.

Local recovery is always superior to global recovery while the overhead is similar or even less. The improved recovery always yields the best possible results but the amount of additional iterations of the auxiliary solver varies. Zero backup, MG and the less accurate SZ compression technique SZ_1e-3 result in a significant number of additional iterations, either global or auxiliary ones. For local recovery, high accurate fixed SZ compression (SZ_1e-7) is good if a fault happens early but is not competitive if a fault happens late in the iterative procedure (e.g. in iteration 110, not shown). Adaptive SZ compression seems to be the best and aSZ_1e0 seems to mark the sweet spot as increasing the accuracy does not yield a big advantage for global and local recovery and a decrease in accuracy (aSZ_1e1) yields around 20% more additional iterations when using improved recovery.

In terms of compression factors (Figure 7, top) and cumulative data size (Figure 7, bottom), most of the SZ compressed backups yield a compression factor of around 100 for the given data size. Therefore, the cumulative size over 115 iterations is less than the size of two classic checkpoints of 1 iteration. Furthermore, the adaptive approach takes advantage of the low accuracy of the solver in the early stages and obtains higher compression factors. At the end of the iterative solve, the adaptive compression is not as strong as the non-adaptive one, but it enables a more efficient recovery as shown before. It should be noted that SZ compression gives even better compression factors when used for more data points. The cumulative checkpoint size could be further reduced by reducing its frequency, i.e. a checkpoint every n-th iteration. This could also reduce the communication overhead because the backups on different ranks could be created in different iterations and thus the communication would be better distributed.

4.3. Soft fault detection and correction in the Generalized Conjugate Residual method

A new nonhydrostatic dynamical core, the Finite Volume Module (FVM, Kühnlein et al., 2019), is being developed at ECMWF for next-generation NWP. The module uses a preconditioned Generalized Conjugate Residual (GCR) elliptic solver (Smolarkiewicz and Margolin, 2000; Smolarkiewicz and Szmelter, 2011) to solve the boundary value problem that arises from the application of the Non-oscillatory Forward-in-Time integrator based on the MPDATA advection scheme (Smolarkiewicz and Szmelter, 2005) and results in an equation which can symbolically be written as

$$\mathcal{L}(\phi) = \mathcal{R} \tag{6}$$

where $\mathcal{L}$ is the elliptic operator and $\mathcal{R}$ represents the right-hand side of the problem at hand. The elliptic solver (Algorithm 1) belongs to a class of Krylov subspace methods, minimizes the L^2-norm of the residual, r, of the model (6), and solves the k^{th}-order damped oscillator equation with preconditioner operator $\mathcal{P}$:

$$\frac{\partial^k \mathcal{P}(\phi)}{\partial \tau^k} + \frac{1}{T_{k-1}(\tau)} \frac{\partial^{k-1} \mathcal{P}(\phi)}{\partial \tau^{k-1}} + \ldots + \frac{1}{T_1(\tau)} \frac{\partial^1 \mathcal{P}(\phi)}{\partial \tau^1} = \mathcal{L}(\phi) - \mathcal{R} \tag{7}$$

See Smolarkiewicz and Szmelter (2011) for a description of the physically motivated stopping criteria. The damped oscillator equation (7) is discretized in pseudo-time τ and optimal parameters $T_1, ..T_{k-1}$ are determined to assure the minimization of $< r, r >$ for the field ϕ (see

Algorithm 1. FT-GCR(k).

For any initial guess, ϕ^0, set $r^0 = \mathcal{L}\left(\phi^0\right) - \mathcal{R}$, $p^0 = \mathcal{P}^{-1}\left(r^0\right)$; then iterate:
for $n = 1, 2, \ldots$ until convergence **do**
  for $\nu = 0, \ldots, k-1$ **do**
    $\beta = \dfrac{\langle r^\nu \mathcal{L}\left(p^\nu\right)\rangle}{\langle \mathcal{L}\left(p^\nu\right)\mathcal{L}\left(p^\nu\right)\rangle}$
    $\phi^{\nu+1} = \phi^\nu + \beta p^\nu$
    $r^{\nu+1} = r^\nu + \beta \mathcal{L}\left(p^\nu\right)$
    if $\|r^{\nu+1}\| \leq \epsilon$ **then**
      exit
    end if
    if $r^{\nu+1} \geq r^\nu$ **then**
      n=n-1
      reset $[\phi, r, p, \mathcal{L}(p)]^0$ to $[\phi, r, p, \mathcal{L}(p)]^*$
    else if $\nu = 0$ **then**
      set $[\phi, r, p, \mathcal{L}(p)]^*$ to $[\phi, r, p, \mathcal{L}(p)]^0$
    end if
    $e = \mathcal{P}^{-1}\left(r^{\nu+1}\right)$
    $\mathcal{L}(e) = \sum_{l=1}^{3} \frac{1}{\zeta_l^*} \nabla \cdot \zeta_l^* C \nabla e$
    for $l = 0, ..\nu$ **do**
      $\alpha_l = \dfrac{\langle \mathcal{L}(e)\mathcal{L}\left(p^l\right)\rangle}{\langle \mathcal{L}\left(p^l\right)\mathcal{L}\left(p^l\right)\rangle}$
      $p^{\nu+1} = e + \sum_{l}^{\nu} \alpha_l p^l$
      $\mathcal{L}\left(p^{\nu+1}\right) = \mathcal{L}(e) + \sum_{l}^{\nu} \alpha_l \mathcal{L}\left(p^l\right)$
    end for
  end for
  reset $[\phi, r, p, \mathcal{L}(p)]^k$ to $[\phi, r, p, \mathcal{L}(p)]^0$
end for

Smolarkiewicz and Margolin, 2000; Smolarkiewicz and Szmelter, 2011, for a detailed discussion).

The solver employs bespoke left-preconditioning (for details see, e.g. Kühnlein et al., 2019) to address the large condition number induced by the domain anisotropy for global atmospheric problems. Since the elliptic solver takes a significant ($\sim$40%) proportion of the computational resources of the dynamical core, it is quite likely to encounter soft faults if they were to occur during computation.

The iterative nature of GCR provides ample opportunity to recompute faulty data at minimal cost to the solver, given an ability to detect such faults. A basic detection method can be derived, given that for each iteration of GCR (n in Algorithm 1), the k^{th} order dampening takes place on a Krylov subspace, K_n, such that $K_n \subseteq K_{n+1}, \forall n \in 1, 2, \ldots$. As such, each iteration of GCR minimizes the L^2-norm of the residual, r, on that Krylov subspace. Since the norm is non-increasing on each subspace, it is also non-increasing between subspaces. If during computation the discrete L^2-norm value increases, that most likely indicates a problem with the solver. Testing the resilience of Algorithm 1 by injecting faults (bit flips) into various stages of GCR often cause the solver to stall for an iteration, after which the routine usually continues – sometimes more slowly but often without other noticeable signs that a problem has been encountered. Even when a larger proportion of data is corrupted by a fault, GCR is usually able to converge with about 10–20% computational overhead. However, this can cause some erratic behaviour of the solver – including large jumps in the maximum absolute value of the residual in the domain, with the potential to induce undesired modes into the solution, even if the solver itself still converges.

Assuming a fault has been detected, the solver can be easily reverted to a state previously deemed good (i.e., no fault detected), at the cost of at most a full iteration of Algorithm 1 (if a fault occurs during the $\nu = k - 1$ pass). Algorithm 1 describes the fault-tolerant GCR (FT-GCR) method, including snap-shotting and fault detection. The instructions needed for fault tolerance are marked in red.

In order to test the effectiveness of FT-GCR (algorithm 1), the GCR elliptic solver in Mengaldo (2016) was adapted to include fault injection, detection, and resetting. The method solves potential flow for an isolated hill on an Earth-like domain. An O32 Octahedral reduced Gaussian mesh (corresponding to $\sim$280 km horizontal resolution) with 51 vertical layers was used for the simulations.

Faults are injected after the preconditioning stage with an implausibly high 2% probability, with a maximum of 10 individual fault occurrences allowed. The fault causes a given number of entries of $e = P^{-1}(r^{\nu+1})$, the preconditioner output, to suffer a bit flip. Figure 8 illustrates graphically the response of the protected and unprotected simulations, for different levels of data corruption per fault encountered. Histograms a), b), c), and d) correspond to 1 (0.0004%), 100 (0.04%), 500 (0.2%), and 2700 (1%) entries corrupted (% of total data), respectively. Table 3 documents averaged statistics for each of these simulation sets. Each set contains between 100 and 200 simulation results, where faultless simulations are disregarded.

It is clear that GCR is highly resilient to bit flip faults. Indeed, even when a larger amount of data is corrupted, the unprotected algorithm usually manages to converge. Additionally, it can be observed that minor data corruption events have little effect on convergence (Figure 8(a)), which is also the likely reason why such small corruption events are difficult to detect. When a sufficient proportion of data is corrupted, FT-GCR detects between 60–80% of faults (Table 3), and convergence delay is reduced by five to eight iterations compared to the unprotected runs. The overhead of this fault-tolerant method is small, with one backup step required per full GCR iteration – including a small amount of extra memory to store the backups. The cost of detection is also minimal, since the expensive global sum to compute the L^2-norm of the residual has already been carried out.

5. Discussion

5.1. Composable resilience in NWP models

No single resilience technique provides a perfect solution to ensure reliable execution of NWP models. Recently, Hukerikar and Engelmann (2017) discussed how effective resilience can be achieved through a composition of multiple resilience techniques in the area of HPC. They claimed the importance of classifications to identify

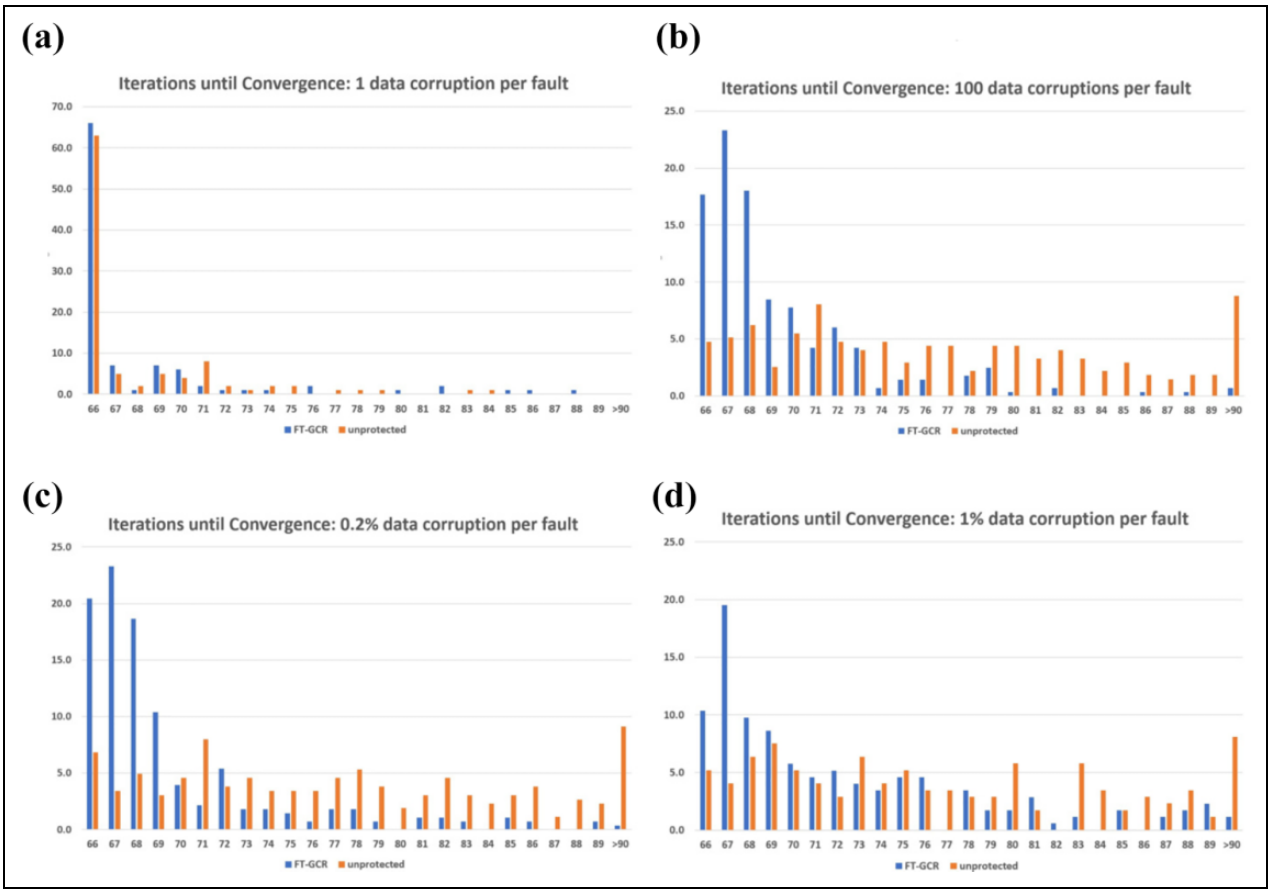

Figure 8. Numerical behaviour of GCR and FT-GCR when faults are introduced. Each histogram displays the percentage of runs that converged at a given iteration. Fault-free convergence is reached after 66 iterations. (a) 0.0004% data lost, (b) 0.04% data lost, (c) 0.2% data lost and (d) 1% data lost.

Table 3. The average number of faults per run, average number of faults detected, average iterations for FT-GCR, and the average number of iterations for unprotected GCR for each plot in Figure 8.

Histogram	% data corrupted	Faults per run	Faults detected	Convergence FT-GCR	Convergence GCR
a	0.0004%	4.17	0.1	68.0	68.0
b	0.04%	4.38	3.72	69.4	77.1
c	0.2%	4.33	3.66	69.6	77.8
d	1.0%	4.61	3.28	72.2	77.3

resilience design patterns across the layers of HPC systems and applications, facilitating seamless systematic compositions of these techniques towards resilience-oriented application-system co-design. Such a co-design balances resilience and performance, both in the forward execution and recovery execution path, ensuring that NWP models can recover in the most efficient way from a variety of failure types.

Resilience in NWP models will therefore result from a combination of hardware, programming environment, algorithmic and application resilience measures. Some of the approaches presented in this paper can be combined straightforwardly. For example, hardware resilience techniques and algorithmic resilience methodologies, or application resilience and soft fault detection/correction, can be applied in a relatively independent way.

As an example of composition of resilience methods, compressed and in-memory checkpointing as seen in Section 3.4.2 could be incorporated into the FT-GCR solver. Assuming that checkpointing will have to play a more dominant role to maintain robustness of solvers, our experiments show that compression can increase efficiency and reduce overhead substantially, without compromising the numerical accuracy. Using these techniques in an adaptive way yields low overhead at the start of the iterative procedure, when a restart or information loss has little effect. Later, when a simple restart will cause the loss of a large amount of invested computational effort, the compression accuracy increases, as well as the overhead, but in return the recovery is as good as with no compression. The computational overhead introduced by the SZ compression is in the low single-digit percentage range, while its achieved compression factors can decrease the memory overhead significantly. Small scale numerical tests (cf. Figure 7) already provide promising compression factors, although SZ compression requires much larger data sets to exploit its full potential.

Introducing these techniques into the FT-GCR solver could further reduce its communication and storage overhead while maintaining its recovery properties. In addition, backup grid approaches as seen in Section 3.3 can easily be implemented in other numerical frameworks, for example in the modal DG framework (Tumolo and Bonaventura, 2015) by storage of the lowest order modal coefficients of the solution for each element.

5.2. Open challenges and opportunities

The techniques surveyed in this paper to secure NWP applications against faults are not yet implemented within operational environments, but may start to play a more significant role in the next few years. As exascale architectures start to be deployed, the feasibility of disk or memory CPR will ultimately be contingent on the evolution of memory latency and bandwidth compared with the growing size of restart files in NWP applications at km-scale resolution. In addition, to assume that all calculations will be performed with no faults will become increasingly costly. Computational cost could be significantly reduced if constraints for fault-free calculations are weakened (see for example discussion of ECC memory checking and stochastic processors in Section 3.1).

The ongoing transition to exascale NWP models constitutes an excellent framework to set out objectives and requirements for resilience and fault tolerance, as well as to test different options to include resilience features in future atmospheric models that may translate into cost savings. More economical resilience strategies should be investigated that could flank or supplant CPR, which should ideally become the technique of last resort. On the road to those solutions, decisions and compromises need to be made on a number of aspects, for instance whether to focus exclusively on fault recovery or also fault detection, and on hard or soft faults.

In the United States, efforts targeting improvement of hardware resilience are under way in the framework of the Exascale computing project. In particular, application of the above-mentioned VeloC has enabled to significantly reduce CPR time (Di Martino et al., 2015). With a final project target of a 1-week MTBF at exascale, on some machines a three-fold increase in MTBF has been obtained.

In Europe, at ECMWF energy efficiency and model sustainability efforts are focusing on reshaping model implementation into more manageable independent units called dwarfs, developed in the framework of the ESCAPE (Müller et al., 2019) and ESCAPE-2[1] projects. The dwarfs are eventually expected to cover around 60% of the forecast model code, so improved checkpointing procedures can be tested on existing dwarfs with a view to establishing a resilience framework applying to future ones. For instance, experiments with checkpointing in memory (Cantwell and Nielsen, 2019) rather than to disk could be carried out. Avoiding the parallel file system may give a sizeable boost in performance when a large number of nodes write simultaneously. With this and other possible solutions, performance gains from fault recovery using spares and spawning, as well as savings in requeueing time and start-up cost of the NWP code, should be weighed against development overhead for the tools themselves. From the systems point of view, standards are already starting to provide resilience tools, an example being the potential inclusion of ULFM as an extension to version 4.x of MPI.

In addition, performance and portability across multiple architectures with a single source code requires NWP models to shift to domain-specific-language (DSL)-based approaches, such as Kokkos (Bertagna et al., 2019), Psyclone (Adams et al., 2019), and Stella/GridTools (Fuhrer et al., 2014; Thaler et al., 2019). Fault tolerance efforts requiring code modifications will more naturally interface with DSLs, see Section 3.3.4 in this paper for an example.

On the hardware side, securing stronger support from HPC vendors in resilience efforts will require clear use cases from the NWP community. ESCAPE and ESCAPE-2 dwarfs can provide an ideal environment for such experiments. When a dwarf-tested application resilience technique evolves into a de facto standard in the weather and climate community, it can become a differentiator for vendors targeting this HPC market.

While vendors have generally been reluctant to publish hardware resilience statistics such as MTBF for HPC architectures, the request for this information could be included in future procurements. Hardware could include system monitoring functionalities, so that users can isolate nodes that look suspicious and distinguish between healthy and unhealthy ones. In fact, unlike faults on data, soft faults impacting instruction sets of hardware as well as integer arithmetic are not yet addressed by any of the methods in this survey, which have not yet been tested for a specific hardware configuration. Atos, an ESCAPE and ESCAPE-2 project member, takes part in MPI Forum, the MPI standardization body (MPIForum, 2020). Currently there is a strong interest in the high-performance Fault Tolerance Interface (FTI) for hybrid systems (Bautista-Gomez et al., 2011), and Atos is already delivering FTI as part of its software stack called Super Computer Suite.

From the algorithms viewpoint, properties of the methods shown in this paper should be explored in more realistic NWP settings. In particular, interpolation-restart techniques could be deployed to fault-proof linear solvers in atmospheric dynamical cores under development such as the ECMWF's FVM (Kühnlein et al., 2019) or the Discontinuous Galerkin (DG) dynamical core under development in ESCAPE-2 along the lines proposed in Tumolo and Bonaventura (2015). Because of the nature of solvers used in NWP, data recovery approaches that are *local* in nature will be of particular interest given the increasing communication/computation time ratio, and contrasting the global connectivity of global spectral solvers still used effectively in NWP today.

As shown by the resilience experiments with the FT-GCR solver in this paper, small numbers of faults in the elliptic solver are hard to detect and generally have a limited impact on convergence. If we can better detect soft faults in the elliptic solver, repair techniques like the one proposed here would be able to adequately protect this part of the model. FVM uses a GCR solver, so that implementing the FT-GCR version introduced here would give an immediate comparison between it and, for example, IR-GMRES in terms of resilience and overhead. Results could be compared to those obtained with disk checkpointing or replication to evaluate trade-offs and inform decisions.

Compressed checkpointing aside, the methods analysed in this paper chiefly, though not exclusively, concern recovery from hard faults. The frequency of soft errors in modern supercomputers presently is not precisely known to model developers. Bit-reproducibility tests could relatively easily be set up for large model configurations in order to start filling this knowledge gap. Hardware vendors could offer valuable advice in these efforts, which would have to be intertwined with verification and validation (V&V) strategies. Indeed, bit-reproducibility tests are a common verification step in the testing of NWP models, but the protocol for these tests and the interpretation of their results may need to be revisited if more frequent soft faults have to be taken into account. Indeed one strategy has been to routinely test with alternative compilers and libraries to sample potential reproducibility issues. In addition to soft faults, the use of massively threaded systems (accelerators), FPGA, and special-purpose – such as AI – chips creates a huge challenge for bitwise reproducibility. Overall bit-reproducibility is realistic when executing only a few test runs, while model checking can be used in workflows with repeated code execution. For the latter, it is essential to explore more sophisticated V&V techniques to assess the correctness of NWP code. Today, a variety of V&V techniques have been practised in other areas of computational science and engineering (Roy, 2005; Roy and Oberkampf, 2011), and the HPC community has recognized the importance of V&V techniques in the areas of compiler optimization (Bentley et al., 2019), runtime (MPI and thread scheduler, Chapp et al., 2015; Evans, 2018) and numerical kernels (Demmel and Nguyen, 2015).

Finally, it can be argued that the operational use of ensemble simulations for weather and climate introduces another level of resilience, since individual ensemble members are run in parallel. If a single simulation is crashing due to a hardware fault, at least most of the members may finish in time and still allow the delivery of a forecast. However, it will be difficult to perform reliable forecasts if the number and quality of ensemble simulations varies. In general, the error of ensemble predictions will be proportional to the sum of a constant that is dependent on the quality of the forecast model plus a term which is proportional to the inverse of the number of ensemble members (see Leutbecher and Ben Bouallègue, 2020).

6. Conclusion

Performance of HPC facilities is expected to grow by an order of magnitude and cross the exascale threshold in the next decade. Time-critical numerical weather prediction, climate monitoring and climate projection are at the forefront of exploiting the available HPC hardware and are preparing for the challenges of using it effectively. Projected figures for failure frequencies question the assumptions on fault-free operating cycles that scientists have been taking for granted. As research teams and operational centres overhaul their codes to rise up to the exascale challenge, enhanced scalable performance will be of little use with algorithms operating in reduced precision, but plagued by unchecked soft faults, and within workflows based on unaffordable checkpointing schedules running on machines with unknown reliability.

This report has reviewed potential pathways to algorithmic fault tolerance and computational resilience in numerical weather and climate prediction. Existing numerical techniques have been surveyed, ranging from interpolation-restart and compressed checkpointing approaches for iterative solvers to in-memory checkpointing, ULFM-based process and data recovery for MPI, and advanced backup techniques. The methods were selected because of their proximity to NWP practice – iterative solvers for linear systems being a major component of semi-implicit methods employed in most operational dynamical cores worldwide, and MPI being the de facto standard for distributed-memory parallelism.

In order to strengthen the case for hardware and software resilience, NWP scientists should reach out to other scientific communities confronted with related issues. In particular, machine learning efforts currently exert strong impact on hardware developments. Therefore, an investigation on the need for resilience in machine learning could offer valuable insights and help to drive hardware development. Wide-ranging multinational exascale projects currently under way provide ideal scientific arenas in which to discuss and develop mitigating strategies against the threats posed by increasing fault rates. The pressing need for reliable and time-critical production of weather and climate forecasts makes a compelling case for accelerating the pace of such endeavours.

Authors' note

The present paper stems from the ESCAPE-2 workshop on fault-tolerant algorithms and resilient approaches for exascale computing held in January 2019 at the Mathematics Department of Politecnico di Milano. An earlier version of the present paper was submitted as an internal ESCAPE-2 project deliverable.

Acknowledgements

We thank the authors of Agullo et al. (2016a, 2016b), namely E Agullo, L Giraud, A Guermouche, J Roman, P Salas, and M Zounon, for the permission to report the

description and numerical testing of the interpolation-restart strategy in Sections 3.4 and 4, and Dr Christian Kühnlein for the data on solver share of wall-clock time in dynamical core runs. PDD gratefully acknowledges funding from the Royal Society for his University Research Fellowship.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the ESCAPE-2 project, European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 800897); the ESiWACE2 Centre of Excellence, European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 823988); and the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC-2075 (Grant Agreement No. 390740016).

ORCID iDs

Tommaso Benacchio https://orcid.org/0000-0002-0732-7167
Erwan Raffin https://orcid.org/0000-0003-0359-5372

Note

1. http://www.hpc-escape2.eu/.

References

Adams S, Ford R, Hambley M, et al. (2019) LFRic: meeting the challenges of scalability and performance portability in weather and climate models. *Journal of Parallel and Distributed Computing* 132: 383–396.

Agullo E, Cools S, Giraud L, et al. (2017) Hard Faults and Soft-Errors: Possible Numerical Remedies in Linear Algebra Solvers. In: I Dutra, R Camacho, J Barbosa, et al. (eds) *High Performance Computing for Computational Science – VECPAR 2016. VECPAR 2016. Lecture Notes in Computer Science*, vol 10150. Springer, Cham. DOI: 10.1007/978-3-319-61982-8_3.

Agullo E, Giraud L, Guermouche A, et al. (2016a) Numerical recovery strategies for parallel resilient Krylov linear solvers. *Numerical Linear Algebra with Applications* 23(5): 888–905.

Agullo E, Giraud L, Salas P, et al. (2016b) Interpolation-restart strategies for resilient eigensolvers. *SIAM Journal on Scientific Computing* 38(5): C560–C583.

Agullo E, Giraud L and Zounon M (2015) On the resilience of parallel sparse hybrid solvers. In: *2015 IEEE 22nd international conference on high performance computing (HiPC)*, Bangalore, India, 16–19 December 2015, pp. 75–84. New York, NY: IEEE.

Ali MM, Southern J, Strazdins P, et al. (2014) Application level fault recovery: using fault-tolerant Open MPI in a PDE solver. In: *2014 IEEE international parallel & distributed processing symposium workshops (IPDPSW),* Phoenix, AZ, United States, 19–23 May 2014, pp. 1169–1178. New York, NY: IEEE.

Altenbernd M and Göddeke D (2018) Soft fault detection and correction for multigrid. *The International Journal of High Performance Computing Applications* 32(6): 897–912.

Ashraf RA, Hukerikar S and Engelmann C (2018) Shrink or substitute: handling process failures in HPC systems using in-situ recovery. In: *2018 26th Euromicro international conference on parallel, distributed and network-based processing (PDP)*, Cambridge, UK, 21–23 March 2018, pp. 178–185. New York, NY: IEEE.

Avizienis A, Laprie JC, Randell B, et al. (2004) Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1(1): 11–33.

Balay S, Abhyankar S, Adams MF, et al. (2019) PETSc Web page. Available at: https://www.mcs.anl.gov/petsc (accessed 21 January 2021).

Barker A, Bernholdt D, Bland A, et al. (2016) *High Performance Computing Facility Operational Assessment 2015: Oak Ridge Leadership Computing Facility*. Oak Ridge, TN: *ORNL*.

Bastian P, Heimann F and Marnach S (2010) Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (DUNE). *Kybernetika* 46: 294–315.

Bauer P, Thorpe A and Brunet G (2015) The quiet revolution of numerical weather prediction. *Nature* 525(7567): 47–55.

Bautista-Gomez L and Cappello F (2015) Detecting silent data corruption for extreme-scale MPI applications. In: *Proceedings of the 22nd European MPI users' group meeting*, Bordeaux, France, September 2015, p. 12. ACM.

Bautista-Gomez L, Tsuboi S, Komatitsch D, et al. (2011) FTI: high performance fault tolerance interface for hybrid systems. In: *SC '11: Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, Seattle WA, USA, November 2011, 13–18 November 2016, pp. 1–12. New York, NY: IEEE.

Bautista-Gomez L, Zyulkyarov F, Unsal O, et al. (2016) Unprotected computing: a large-scale study of DRAM raw error rate on a supercomputer. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*, Salt Lake City, UT, USA, p. 55. New York, NY: IEEE.

Benacchio T (2014) *A blended soundproof-to-compressible model for small- to mesoscale atmospheric dynamics*. PhD Thesis, Freie Universität Berlin, Germany. Available at: https://refubium.fu-berlin.de/handle/fub188/8152 (accessed 21 January 2021)

Benacchio T, O'Neill W and Klein R (2014) A blended soundproof-to-compressible model for atmospheric dynamics. *Monthly Weather Review* 142(12): 4416–4438.

Benoit A, Cavelan A, Cappello F, et al. (2017) Identifying the right replication level to detect and correct silent errors at scale. In: *Proceedings of the 2017 workshop on fault-tolerance for HPC at extreme scale*, Washington, DC, USA, June 2017, pp. 31–38. New York, NY: ACM.

Bentley M, Briggs I, Gopalakrishnan G, et al. (2019) Multi-level analysis of compiler-induced variability and performance tradeoffs. In: *Proceedings of the 28th international symposium on high-performance parallel and distributed computing*, Phoenix, AZ, USA, June 2019, pp. 61–72. New York, NY: ACM.

Berrocal E, Bautista-Gomez L, Di S, et al. (2015) Lightweight silent data corruption detection based on runtime data analysis for HPC applications. In: *Proceedings of the 24th international symposium on high-performance parallel and distributed computing*, Portland, Oregon, USA, June 2015, pp. 275–278. New York, NY: ACM.

Berrocal E, Bautista-Gomez L, Di S, et al. (2016) Exploring partial replication to improve lightweight silent data corruption detection for HPC applications. In: *European conference on parallel processing*, (eds. D Pierre-Fran{\c{c}}ois and T Denis), pp. 419–430. Berlin: Springer.

Bertagna L, Deakin M, Guba O, et al. (2019) HOMMEXX 1.0: a performance-portable atmospheric dynamical core for the Energy Exascale Earth System Model. *Geoscientific Model Development* 12(4): 1423–1441.

Bhatele A, Mohror K, Langer SH, et al. (2013) There goes the neighborhood: performance degradation due to nearby jobs. In: *SC '13: Proceedings of the international conference on high performance computing, networking, storage and analysis*, Denver, CO, United States, 17–22 November 2013, pp. 1–12.

Bhatele A, Thiagarajan JJ, Groves T, et al. (2020) The case of performance variability on dragonfly-based systems. In: *2020 IEEE international parallel and distributed processing symposium (IPDPS)*, New Orleans, LA, United States, 18–22 May 2020, pp. 896–905.

Bland W, Bouteiller A, Herault T, et al. (2013) An evaluation of user-level failure mitigation support in MPI. *Computing* 95(12): 1171–1184.

Blatt M and Bastian P (2007) The iterative solver template library. In: B Kågström, E Elmroth, J Dongarra, and J Waśniewski (eds) *Applied Parallel Computing. State of the Art in Scientific Computing*. Berlin: Springer Berlin Heidelberg, pp. 666–675.

Bonaventura RLR and Budich R (2012) *Earth System Modelling 2: Algorithms, Code Infrastructure and Optimisation*. New York, NY: Springer Verlag.

Bridges PG, Ferreira KB, Heroux MA, et al. (2012) Fault-tolerant linear solvers via selective reliability. *arXiv preprint arXiv: 1206.1390*.

Calhoun J, Snir M, Olson LN, et al. (2017) Towards a more complete understanding of SDC propagation. In: *Proceedings of the 26th international symposium on high-performance parallel and distributed computing*, Washington, DC, USA, June 2017, pp. 131–142. New York, NY: ACM.

Cantwell CD and Nielsen AS (2019) A minimally intrusive low-memory approach to resilience for existing transient solvers. *Journal of Scientific Computing* 78(1): 565–581.

Cappello F (2009) Fault tolerance in petascale/exascale systems: current knowledge, challenges and research opportunities. *The International Journal of High Performance Computing Applications* 23(3): 212–226.

Cappello F, Geist A, Gropp W, et al. (2014) Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations* 1(1): 5–28.

Chapp D, Johnston T and Taufer M (2015) On the need for reproducible numerical accuracy through intelligent runtime selection of reduction algorithms at the extreme scale. In: *2015 IEEE international conference on cluster computing*, Chicago, IL, USA, 8–11 September 2015, pp. 166–175. New York, NY: IEEE.

Chen C, Du Y, Zuo K, et al. (2017) Toward fault-tolerant hybrid programming over large-scale heterogeneous clusters via checkpointing/restart optimization. *The Journal of Supercomputing* 75: 4226–4247.

Cher CY, Gupta M, Bose P, et al. (2014) Understanding soft error resiliency of Blue Gene/Q compute chip through hardware proton irradiation and software fault injection. In: *SC'14: Proceedings of the international conference for high performance computing, networking, storage and analysis*, New Orleans, LA, USA, 16–21 November 2014, pp. 587–596. New York, NY: IEEE.

Chien A, Balaji P, Beckman P, et al. (2015) Versioned distributed arrays for resilience in scientific applications: global view resilience. *Procedia Computer Science* 51: 29–38.

Cores I, Rodríguez G, González P, et al. (2013) Improving scalability of application-level checkpoint-recovery by reducing checkpoint sizes. *New Generation Computing* 31(3): 163–185.

Daly JT (2006) A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems* 22(3): 303–312.

Davis TA and Hu Y (2011) The University of Florida sparse matrix collection. *J-TOMS* 38(1): 25.

Dawson A, Düben PD, MacLeod DA, et al. (2018) Reliable low precision simulations in land surface models. *Climate Dynamics* 51(7): 2657–2666.

De Oliveira D, Pilla L, Hanzich M, et al. (2017) Radiation-induced error criticality in modern HPC parallel accelerators. In: *2017 IEEE international symposium on high performance computer architecture (HPCA)*, Austin, TX, USA, 4–8 February 2017, pp. 577–588. New York, NY: IEEE.

Dell TJ (1997) A white paper on the benefits of chipkill-correct ECC for PC server main memory. *IBM Microelectronics Division* 11: 1–23.

Demmel J and Nguyen HD (2015) Parallel reproducible summation. *IEEE Transactions on Computers* 64(7): 2060–2070.

Di S and Cappello F (2016) Fast error-bounded lossy HPC data compression with SZ. In: *2016 IEEE international parallel and distributed processing symposium*, Chicago, IL, USA, 23–27 May 2016, pp. 730–739 New York, NY: IEEE.

Di Martino C, Kramer W, Kalbarczyk Z, et al. (2015) Measuring and understanding extreme-scale application resilience: a field study of 5,000,000 HPC application runs. In: *2015 45th annual IEEE/IFIP international conference on dependable systems and networks* (ed. DJ Rio), 22–25 June 2015, pp. 25–36. New York, NY: IEEE.

Dongarra J, Herault T and Robert Y (2015) Fault tolerance techniques for high-performance computing. In: *Fault-tolerance*

techniques for high-performance computing, pp. 3–85. Berlin: Springer.

Düben PD and Dawson A (2017) An approach to secure weather and climate models against hardware faults. *Journal of Advances in Modeling Earth Systems* 9(1): 501–513.

Düben PD and Dolaptchiev SI (2015) Rounding errors may be beneficial for simulations of atmospheric flow: results from the forced 1D Burgers equation. *Theoretical and Computational Fluid Dynamics* 29: 311–328.

Düben PD, Joven J, Lingamneni A, et al. (2014a) On the use of inexact, pruned hardware in atmospheric modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372(2018): 20130276.

Düben PD, McNamara H and Palmer T (2014b) The use of imprecise processing to improve accuracy in weather & climate prediction. *Journal of Computational Physics* 271: 2–18.

Düben PD and Palmer TN (2014) Benchmark tests for numerical weather forecasts on inexact hardware. *Monthly Weather Review* 142(10): 3809–3829

Düben PD, Wedi NP, Zeman C, et al. (2020) Global simulations of the atmosphere at 1.45 km grid-spacing with the Integrated Forecasting System. *Journal of the Meteorological Society of Japan* 98: 551–572.

Elliott J, Hoemmen M and Mueller F (2014) Evaluating the impact of SDC on the GMRES iterative solver. In: *2014 IEEE 28th international parallel and distributed processing symposium*, Phoenix, AZ, USA, 19–23 May 2014, pp. 1193–1202. New York, NY: IEEE.

Elliott J, Hoemmen M and Mueller F (2015) A numerical soft fault model for iterative linear solvers. In: *Proceedings of the 24th international symposium on high-performance parallel and distributed computing*, Portland, OR, USA, June 2015, pp. 271–274. New York, NY: ACM.

Elliott J, Hoemmen M and Mueller F (2016) Exploiting data representation for fault tolerance. *Journal of Computational Science* 14: 51–60.

Engwer C, Altenbernd M, Dreier NA, et al. (2018) A high-level C++ approach to manage local errors, asynchrony and faults in an MPI application. In: *2018 26th Euromicro international conference on parallel, distributed and network-based processing (PDP)*, 21–23 March 2018, Cambridge, UK, pp. 714–721. New York, NY: IEEE.

Evans N (2018) Verifying Qthreads: is model checking viable for user level tasking runtimes? In: *2nd IEEE/ACM international workshop on software correctness for HPC applications, CORRECTNESS@SC 2018*, Dallas, TX, United States, 12 November 2018, pp. 25–32. IEEE.

Fagg GE and Dongarra JJ (2000) FT-MPI: fault tolerant MPI, supporting dynamic applications in a dynamic world. In: *European parallel virtual machine/message passing interface users' group meeting* (eds. J Dongarra, P Kacsuk and N Podhorszki), pp. 346–353. Berlin: Springer.

Feng S, Gupta S, Ansari A, et al. (2010) Shoestring: probabilistic soft error reliability on the cheap. SIGARCH Comput. *Archit. News* 38(1): 385–396. DOI: 10.1145/1735970.1736063.

Fiala D, Mueller F, Engelmann C, et al. (2012) Detection and correction of silent data corruption for large-scale high-performance computing. In: *Proceedings of the international conference on high performance computing, networking, storage and analysis*, Salt Lake City, UT, USA, 10–16 November 2012, p. 78. New York, NY: IEEE Computer Society Press.

Fuhrer O, Chadha T, Hoefler T, et al. (2018) Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0. *Geoscientific Model Development* 11(4): 1665–1681.

Fuhrer O, Osuna C, Lapillonne X, et al. (2014) Towards a performance portable, architecture agnostic implementation strategy for weather and climate models. *Supercomputing Frontiers and Innovations* 1(1): 45–62.

Gamell M, Katz D, Kolla H, et al. (2014) Exploring automatic, online failure recovery for scientific applications at extreme scales. In: *SC'14: Proceedings of the international conference for high performance computing, networking, storage and analysis*, New Orleans, LA, USA, 16–21 November 2014, pp. 895–906. New York, NY: IEEE.

Gamell M, Teranishi K, Kolla H, et al. (2017a) Scalable failure masking for stencil computations using ghost region expansion and cell to rank remapping. *SIAM Journal on Scientific Computing* 39(5): S347–S378.

Gamell M, Teranishi K, Mayo J, et al. (2017b) Modeling and simulating multiple failure masking enabled by local recovery for stencil-based applications at extreme scales. *IEEE Transactions on Parallel and Distributed Systems* 28(10): 2881–2895.

Gamell M, Teranishi K, Van Der Wijngaart R, et al. (2016) *Fenix, A Fault Tolerant Programming Framework for MPI Applications 1.0.1*. Technical Report SAND No. 2016-10522, Sandia National Laboratories, Livermore, CA, United States.

Georgakoudis G, Guo L and Laguna I (2020) Reinit^{++}: evaluating the performance of global-restart recovery methods for MPI fault tolerance. In: *High performance computing – 35th international conference, ISC High Performance 2020, Frankfurt/Main, Germany, 22–25 June 2020, Proceedings, Lecture Notes in Computer Science*, volume 12151, pp. 536–554. Berlin: Springer.

Göddeke D, Altenbernd M and Ribbrock D (2015) Fault-tolerant finite-element multigrid algorithms with hierarchically compressed asynchronous checkpointing. *Parallel Computing* 49: 117–135.

Golaz JC, Caldwell PM, Van Roekel LP, et al. (2019) The DOE E3SM coupled model version 1: overview and evaluation at standard resolution. *Journal of Advances in Modeling Earth Systems* 11(7): 2089–2129.

Guerraoui R and Schiper A (1997) Software-based replication for fault tolerance. *Computer* 30(4): 68–74.

Guhur PL, Constantinescu E, Ghosh D, et al. (2017) Detection of silent data corruption in adaptive numerical integration solvers. In: *2017 IEEE international conference on cluster computing (CLUSTER)*, 5–8 September 2017, Honolulu, HI, pp. 592–602. New York, NY: IEEE.

Gupta S, Patel T, Engelmann C, et al. (2017) 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, Denver, Colorado, November 2017, p. 44. ACM.

Hassani A, Skjellum A and Brightwell R (2014) Design and evaluation of FA-MPI, a transactional resilience scheme for non-blocking MPI. In: *2014 44th annual IEEE/IFIP international conference on dependable systems and networks*, Atlanta, GA, 23–26 June 2014, pp. 750–755. New York, NY: IEEE.

Hatfield S, Düben P, Chantry M, et al. (2018) Choosing the optimal numerical precision for data assimilation in the presence of model error. *Journal of Advances in Modeling Earth Systems* 10(9): 2177–2191.

Hoemmen MF, Heroux MA, Ferreira KB, et al. (2011) *Fault-Tolerant Iterative Methods via Selective Reliability*. Technical Report, Sandia National Lab (SNL-NM), Albuquerque, NM, United States.

Huber M, Gmeiner B, Rüde U, et al. (2016) Resilience for massively parallel multigrid solvers. *SIAM Journal on Scientific Computing* 38(5): S217–S239.

Hukerikar S and Engelmann C (2017) Resilience design patterns: a structured approach to resilience at extreme scale. *Supercomputing Frontiers and Innovations* 4(3). Available at: https://superfri.org/superfri/article/view/138 (accessed 21 January 2021).

Inadomi Y, Patki T, Inoue K, et al. (2015) Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*, Austin, TX, 15–20 November 2015, pp. 1–12. ACM.

Islam TZ, Mohror K, Bagchi S, et al. (2013) McrEngine: a scalable checkpointing system using data-aware aggregation and compression. *Scientific Programming* 21(3–4): 149–163.

Kannan S, Gavrilovska A, Schwan K, et al. (2013) Optimizing checkpoints using NVM as virtual memory. In: *2013 IEEE 27th international symposium on parallel and distributed processing*, Boston, MA, 20–24 May 2013, pp. 29–40. Washington, DC: IEEE Computer Society.

Kim J, Sullivan M and Erez M (2015a) Bamboo ECC: strong, safe, and flexible codes for reliable computer memory. In: *2015 IEEE 21st international symposium on high performance computer architecture (HPCA)*, Burlingame, CA, United States, 7–11 February 2015, pp. 101–112.

Kim J, Sullivan M, Gong SL, et al. (2015b) Frugal ECC: efficient and versatile memory error protection through fine-grained compression. In: *SC '15: Proceedings of the international conference for high performance computing, networking, storage and analysis, SC '15*, Austin TX, 15–20 November 2015, pp. 1–12. ACM.

Kohl N, Hötzer J, Schornbaum F, et al. (2019) A scalable and extensible checkpointing scheme for massively parallel simulations. *The International Journal of High Performance Computing Applications* 33(4): 571–589. DOI: 10.1177/1094342018767736.

Kühnlein C, Deconinck W, Klein R, et al. (2019) FVM 1.0: a nonhydrostatic finite-volume dynamical core formulation for IFS. *Geoscientific Model Development Discussions* 12: 651–676.

Laguna I, Richards DF, Gamblin T, et al. (2016) Evaluating and extending user-level fault tolerance in MPI applications. *The International Journal of High Performance Computing Applications* 30(3): 305–319.

Langou J, Chen Z, Bosilca G, et al. (2007) Recovery patterns for iterative methods in a parallel unstable environment. *SIAM Journal on Scientific Computing* 30: 102–116.

Leutbecher M and Ben Bouallègue Z (2020) On the probabilistic skill of dual-resolution ensemble forecasts. *Quarterly Journal of the Royal Meteorological Society* 146(727): 707–723.

Li D, Chen Z, Wu P, et al. (2013) Rethinking algorithm-based fault tolerance with a cooperative software-hardware approach. In: *SC '13: Proceedings of the international conference on high performance computing, networking, storage and analysis*, New Orleans, LA, United States, November 2014, pp. 1–12.

Li G, Pattabiraman K, Hari S, et al. (2018) Modeling soft-error propagation in programs. In: *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, Luxembourg City, 25–28 June 2018, pp. 27–38. New York, NY: IEEE.

Liang X, Di S, Tao D, et al. (2018) Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In: *2018 IEEE international conference on big data (Big Data)*, Seattle, WA, United States, 10–13 December 2018, Seattle, WA, USA, 10–13 December 2018, pp. 438–447.

Lingamneni A, Muntimadugu KK, Enz C, et al. (2012) Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling. In: *Proceedings of the 9th conference on computing frontiers*, Cagliari, Italy, May 2012, pp. 3–12. ACM.

Losada N, Bosilca G, Bouteiller A, et al. (2019) Local rollback for resilient MPI applications with application-level checkpointing and message logging. *Future Generation Computer Systems* 91: 450–464.

Losada N, González P, Martin MJ, et al. (2020) Fault tolerance of MPI applications in exascale systems: the ULFM solution. *Future Generation Computing Systems* 106: 467–481.

Lyons R and Vanderkulk W (1962) The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development* 6(2): 200–209.

Mengaldo G (2016) *Batch 1: Definition of Several Weather & Climate Dwarfs*. Technical Report, ECMWF. Available at: https://arxiv.org/abs/1908.06089 (accessed 21 January 2021).

Mengaldo G, Wyszogrodzki A, Diamantakis M, et al. (2019) Current and emerging time-integration strategies in global numerical weather and climate prediction. *Archives of Computational Methods in Engineering* 26(3): 663–684.

Meuer H, Strohmaier E, Dongarra JJ, et al. (2019) Top500 supercomputer sites. Available at: http://www.top500.org/ (accessed 21 January 2021).

Michalak SE, DuBois AJ, Storlie CB, et al. (2012) Assessment of the impact of cosmic-ray-induced neutrons on hardware in the Roadrunner supercomputer. *IEEE Transactions on Device and Materials Reliability* 12(2): 445–454.

Miles JS, Teranishi K, Morales NM, et al. (2019) *Software Resilience Using Kokkos Ecosystem*. Technical Report SAND2019-3616, Sandia National Laboratories.

Mittal S and Inukonda MS (2018) A survey of techniques for improving error-resilience of DRAM. *Journal of Systems Architecture* 91: 11–40.

Mittal S and Vetter JS (2015) A survey of techniques for modeling and improving reliability of computing systems. *IEEE Transactions on Parallel and Distributed Systems* 27(4): 1226–1238.

Mittal S and Vetter JS (2016) A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Transactions on Parallel and Distributed Systems* 27(5): 1537–1550.

Mohamed MF (2016) Service replication taxonomy in distributed environments. *Service Oriented Computing and Applications* 10(3): 317–336.

MPIForum (2020) Available at: https://www.mpi-forum.org (accessed 22 November 2019).

Mubarak M, Carns P, Jenkins J, et al. (2017) Quantifying I/O and communication traffic interference on dragonfly networks equipped with burst buffers. In: *2017 IEEE international conference on cluster computing (CLUSTER)*, 5–8 September 2017, Honolulu, HI, USA, pp. 204–215. New York, NY: IEEE.

Müller A, Deconinck W, Kühnlein C, et al. (2019) The ESCAPE project: energy-efficient scalable algorithms for weather prediction at exascale. *Geoscientific Model Development* 12(10): 4425–4441.

Neumann P, Düben P, Adamidis P, et al. (2019) Assessing the scales in numerical weather and climate predictions: will exascale be the rescue? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 377(2142): 20180148.

Ni X, Meneses E, Jain N, et al. (2013) ACR: automatic checkpoint/restart for soft and hard error protection. In: *Proceedings of the international conference on high performance computing, networking, storage and analysis*, November 2013, Denver, CO, USA, p. 7. New York, NY: ACM.

Nicolae B, Moody A, Gonsiorowski E, et al. (2019) VeloC: towards high performance adaptive asynchronous checkpointing at large scale. In: *IPDPS'19: The 2019 IEEE international parallel and distributed processing symposium*, Rio de Janeiro, Brazil, pp. 911–920. Available at: https://veloc.readthedocs.io/en/latest/ (accessed 21 January 2021).

Nielsen AS (2018) *Scaling and resilience in numerical algorithms for exascale computing*. PhD Thesis, École Polytechnique Fédérale de Lausanne. Available at: https://infoscience.epfl.ch/record/258087/files/EPFL_TH8926.pdf (accessed 21 January 2021).

Paige CC and Saunders MA (1975) Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 12: 617–629.

Patterson D and Hennessy J (2016) Parallelism and the memory hierarchy: redundant arrays of inexpensive disks. In: DA Patterson, and JL Hennessy (eds) *Computer Organization and Design ARM Edition*. Cambridge, MA: Morgan Kaufmann, Chapter 5.11.

Paul S, Hayashi A, Whitlock M, et al. (2020) Integrating internode communication with a resilient asynchronous many-task runtime system. In: *2020 Workshop on Exascale MPI (ExaMPI)*, Atlanta, GA, USA, 2020, pp. 41–51. DOI: 10.1109/ExaMPI52011.2020.00010.

Peterson WW and Brown DT (1961) Cyclic codes for error detection. *Proceedings of the IRE* 49(1): 228–235.

Plank JS, Beck M and Kingsley G (1995) Compiler-assisted memory exclusion for fast checkpointing. *IEEE Technical Committee on Operating Systems and Application Environments* 7(4): 10–14.

Qiao Z, Liang S, Fu S, et al. (2019) Characterizing and modeling reliability of declustered raid for HPC storage systems. In: *2019 49th annual IEEE/IFIP international conference on dependable systems and networks – industry track*, Portland, OR, USA, 24–27 June 2019, pp. 17–20. IEEE.

Quarteroni A, Sacco R and Saleri F (2010) *Numerical Mathematics,* Volume 37. Berlin: Springer Science & Business Media.

Radojkovic P, Marazakis M, Carpenter P, et al. (2020) *Towards Resilient EU HPC Systems: A Blueprint*. Research Report, European HPC Resilience Initiative. Available at: https://hal.inria.fr/hal-02922257 (accessed 21 January 2021).

Ramabadran TV and Gaitonde SS (1988) A tutorial on CRC computations. *IEEE Micro* 8(4): 62–75.

Rizzi F, Morris K, Sargsyan K, et al. (2018a) Exploring the interplay of resilience and energy consumption for a task-based partial differential equations preconditioner. *Parallel Computing* 73: 16–27.

Rizzi F, Morris K, Sargsyan K, et al. (2018b) Partial differential equations preconditioner resilient to soft and hard faults. *The International Journal of High Performance Computing Applications* 32(5): 658–673.

Rodríguez G, Martín MJ, González P, et al. (2010) CPPC: a compiler-assisted tool for portable checkpointing of message-passing applications. *Concurrency and Computation: Practice and Experience* 22(6): 749–766.

Roy CJ (2005) Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics* 205(1): 131–156.

Roy CJ and Oberkampf WL (2011) A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing. *Computer Methods in Applied Mechanics and Engineering* 200(25): 2131–2144.

Russell FP, Düben PD, Niu X, et al. (2015) Architectures and precision analysis for modelling atmospheric variables with chaotic behaviour. In: *2015 IEEE 23rd annual international symposium on field-programmable custom computing machines*, 2–6 May 2015, Vancouver, BC, Canada, pp. 171–178. New York, NY: IEEE.

Saad Y and Schultz MH (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 7(3): 856–869.

Sancho J, Petrini F, Johnson G, et al. (2004) On the feasibility of incremental checkpointing for scientific computing. In: *2004 proceedings of 18th international parallel and distributed processing symposium*, 26–30 April 2004, Santa Fe, NM, USA, p. 58. New York, NY: IEEE.

Sao P and Vuduc R (2013) Self-stabilizing iterative solvers. In: *Proceedings of the workshop on latest advances in scalable algorithms for large-scale systems*, November 2013, Denver, CO, USA, p. 4. New York, NY: ACM.

Sargsyan K, Rizzi F, Mycek P, et al. (2015) Fault resilient domain decomposition preconditioner for PDEs. *SIAM Journal on Scientific Computing* 37(5): A2317–A2345.

Sartori J, Sloan J and Kumar R (2011) Stochastic computing: embracing errors in architecture and design of processors and applications. In: *Proceedings of the 14th international conference on compilers, architectures and synthesis for embedded systems*, 9–14 October 2011, Taipei, pp. 135–144. New York, NY: ACM.

Schroeder B and Gibson G (2009) A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing* 7(4): 337–350.

Schulthess TC, Bauer P, Wedi N, et al. (2018) Reflecting on the goal and baseline for exascale computing: a roadmap based on weather and climate simulations. *Computing in Science & Engineering* 21(1): 30–41.

Smolarkiewicz PK and Margolin L (2000) Variational methods for elliptic problems in fluid models. In: *Proceedings of the Workshop on Developments in Numerical Methods for Very High Resolution Global Models*, ECMWF, Reading, UK, 5–7 June 2000, pp. 137–159.

Smolarkiewicz PK and Szmelter J (2005) MPDATA: an edge-based unstructured-grid formulation. *Journal of Computational Physics* 206: 624–649.

Smolarkiewicz PK and Szmelter J (2011) A nonhydrostatic unstructured-mesh soundproof model for simulation of internal gravity waves. *Acta Geophysica* 59: 1109–1134.

Snir M, Wisniewski RW, Abraham JA, et al. (2014) Addressing failures in exascale computing. *The International Journal of High Performance Computing Applications* 28(2): 129–173.

Steppeler J, Hess R, Doms G, et al. (2003) Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics* 82: 287–301.

Sun G (2014) *Exploring Memory Hierarchy Design with Emerging Memory Technologies*. Berlin: Springer.

Tao D, Di S, Chen Z, et al. (2017) Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In: *2017 IEEE international parallel and distributed processing symposium (IPDPS)*, Orlando, FL, USA, 29 May–2 June 2017, pp. 1129–1139. New York, NY: IEEE.

Tao D, Di S, Liang X, et al. (2018) Improving performance of iterative methods by lossy checkponting. In: *Proceedings of the 27th international symposium on high-performance parallel and distributed computing*, Tempe, AZ, USA, June 2018, pp. 52–65. New York, NY: ACM.

Teranishi K and Heroux MA (2014) Toward local failure local recovery resilience model using MPI-ULFM. In: *Proceedings of the 21st European MPI users' group meeting*, Kyoto, Japan, September 2014, p. 51. New York, NY: ACM.

Thaler F, Moosbrugger S, Osuna C, et al. (2019) Porting the COSMO weather model to manycore CPUs. In: *Proceedings of the platform for advanced scientific computing conference*, Zurich, Switzerland, June 2019, p. 13. New York, NY: ACM.

Tumolo G and Bonaventura L (2015) A semi-implicit, semi-Lagrangian discontinuous Galerkin framework for adaptive numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society* 141(692): 2582–2601.

Turnbull D and Alldrin N (2003) *Failure Prediction in Hardware Systems*. Technical Report, University of California, San Diego.

Wedi NP, Polichtchouk I, Düben P, et al. (2020) A baseline for global weather and climate simulations at 1 km resolution. *Journal of Advances in Modeling Earth Systems* 12(11). DOI: 10.1029/2020MS002192.

White A (2002) A view of the equations of meteorological dynamics and various approximations. *Large-Scale Atmosphere–Ocean Dynamics* 1: 1–100.

Yang X, Jenkins J, Mubarak M, et al. (2016) Watch out for the bully! Job interference study on dragonfly network. In: *SC '16: Proceedings of the international conference for high performance computing, networking, storage and analysis*, Salt Lake City, UT, 13–18 November 2016, pp. 750–760.

Author biographies

Tommaso Benacchio is a Research Fellow with MOX – Modelling and Scientific Computing Laboratory, Department of Mathematics, Politecnico di Milano, Italy. He received a PhD from Freie Universität Berlin in 2014 and an MSc from Politecnico di Milano in 2010. His research interest include numerical methods for partial differential equations, numerical weather prediction and high-performance computing applications.

Luca Bonaventura is Associate Professor at Politecnico di Milano. He received a Master degree in Mathematics from Università Sapienza of Rome, Italy in 1989 and a PhD in Mathematics from Università di Trento, Italy, in 1994. His main research interests focus on adaptive time and space discretization methods for PDEs, with applications to geophysical problems and especially numerical weather prediction.

Mirco Altenbernd is a research assistant at the chair for Computational Mathematics for Complex Simulation in Science and Engineering at the University of Stuttgart, Germany. He pursues his PhD in Simulation Technology at the Simtech Cluster of Excellence in cooperation with the Faculty of Mathematics. His current research interests are fault-tolerant numerical methods for arising exascale computer systems with a focus on algorithm-based fault tolerance and lossy compression techniques. He obtained his master degree at the TU Dortmund in 2015.

Chris D Cantwell is a senior lecturer in the Department of Aeronautics at Imperial College London, UK. He received his PhD in Scientific Computing from the University of Warwick

in 2009. His research interests include high-order spectral/hp element methods, software resilience and machine learning, with applications in fluid dynamics and biomedicine.

Peter D Düben holds a University Research Fellowship of the Royal Society and is the AI and Machine Learning Coordinator of ECMWF. He has written his PhD thesis at the Max Planck Institute for Meteorology and has worked as PostDoc at the University of Oxford before moving to ECMWF.

Mike Gillard is a Senior Research Associate at the School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University. He received a MSci degree from the University of Nottingham in 2005, and from Durham University an MSc in 2007 and a PhD in 2010. His research interests include Numerical methods for nonlinear PDEs and software development for scientific computing.

Luc Giraud is a senior researcher at Inria. He received an engineer degree in 1988 and a PhD in 1991 from Institut National Polytechnique Toulouse in computer science and applied mathematics. His research interests include high-performance computing and numerical linear algebra.

Dominik Göddeke studied computer science and mathematics at TU Dortmund, Germany, from 1999 to 2004. He obtained his PhD in applied mathematics from TU Dortmund in 2010. He was appointed junior professor at TU Dortmund in 2011 and is, since 2015, professor for Computational Mathematics for Complex Simulations at the University of Stuttgart, Germany. Since 2016, he is the director of the Institute of Applied Analysis and Numerical Simulation, and a Fellow of the Stuttgart Center of Simulation Science, both at the University of Stuttgart.

Erwan Raffin is an HPC Distinguished Expert at Atos' Center for Excellence in Performance Programming. He received a PhD degree in computer science in 2011 from the University of Rennes 1, France in collaboration between Technicolor and the Cairn INRIA project-team. His research interests include scientific applications optimization especially for Weather and Climate Modelling.

Keita Teranishi is a principal member of technical staff at Sandia National Laboratories, USA. He received the BS and MS degrees from the University of Tennessee, Knoxville, in 1998 and 2000, respectively, and the PhD degree from Penn State University, in 2004. His research interests are fault-tolerant programming models, sparse matrix and tensor computation and software ecosystem for high-performance computing systems.

Nils Wedi leads ECMWF's Earth System Modelling section that addresses all aspects of scientific and computational performance relating to ECMWF's forecast model. He has worked at ECMWF for the past 25 years. He received his PhD from the Ludwig-Maximilians-Universität München in 2005.