Funded by the European Union

Towards fault tolerance in HPC for numerical weather and climate prediction

Tommaso Benacchio

19th Workshop on High Performance Computing in Meteorology

ECMWF | 23 September 2021

POLITECNICO MILANO 1863
DIPARTIMENTO DI MATEMATICA

Contents

- Fault-tolerant computing & design
- Taxonomy & existing resilience strategies
- FT-GCR: a fault-tolerant Krylov solver
- Take-home messages and future work

Fault-tolerant computing

First Record of Single-Event Upset on Ground,
Cray-1 Computer at Los Alamos in 1976

"... so we know that during those six months of operation, 152 parity errors were recorded [...] caused by atmospheric neutrons"

Normand et al. 2010

Fault-tolerant design

1. How likely is a component/system to fail?

Fault-tolerant design

1. How likely is a component/system to fail?

Systems	2009	2011	2015	2020 ?
System peak	2 Peta	20 Peta	100-200 Peta	1 Exa
System memory	0.3 PB	1.6 PB	5 PB	10 PB
Node performance	125 GF	200GF	200-400 GF	1-10TF
Node memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	22 GB/s	25 GB/s	50 GB/s
System size (nodes)	18,700	100,000	500,000	O(million)
Total concurrency	225,000	3,200,000	O(50,000,000)	O(billion)
Storage	15 PB	30 PB	150 PB	300 PB
IO	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	4 days	19 h 4 min	3 h 52 min	1 h 56 min
Power	6 MW	~10MW	~10 MW	~20 MW

More and more cores
Smaller circuit sizes
Lower voltages
Thermal/voltage var, radiation

XC-40@ECMWF: 7220 nodes, 8PF
15 node failures/month

Chen et al. '17, Boehm et al. '10, C. Engelmann, B. et al. 2021, Agullo et al. '21

Fault-tolerant design

1. How likely is a component/system to fail?

2. How critical is the component/system?

Fault-tolerant design – NWP & Climate

1. How likely is a component/system to fail?

2. How critical is the component/system?

New York Times

Fault-tolerant design – NWP & Climate

1. How likely is a component/system to fail?

2. How critical is the component/system?

3. How expensive is it to make the component/system fault-tolerant?

Tight operational schedules | High computational intensity | Finer resolutions

Taxonomy

Fault - cause of an error

- Hard – permanent and reproducible (broken hardware)
- Soft – transient and non-reproducible (bit flip in memory)

- Detected and corrected – error-correcting codes, replication, re-execution
- Detected and uncorrectable – application crash
- Undetected – silent data corruption

Error - part of state causing → Failure - transition to incorrect service

Nielsen 2018, B. et al. 2021

Existing strategies - Checkpointing

Store system state at regular intervals | restore | recompute on failure

Costly | heavy I/O load | detection latency / checkpoint frequency trade-off

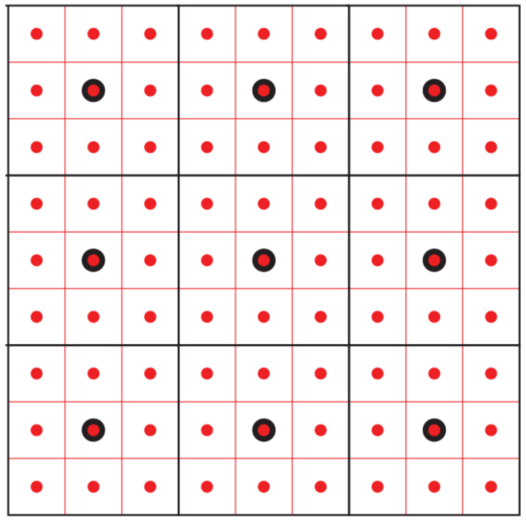
Not viable if $T_{CPR} > MTTI$

For a 10-day forecast @ECMWF: 5 checkpoints written | 1% runtime | FS b/w absorbed

@1km resolution: restart files of several TB across 1000s MPI procs to single FS

Existing strategies - Redundancy

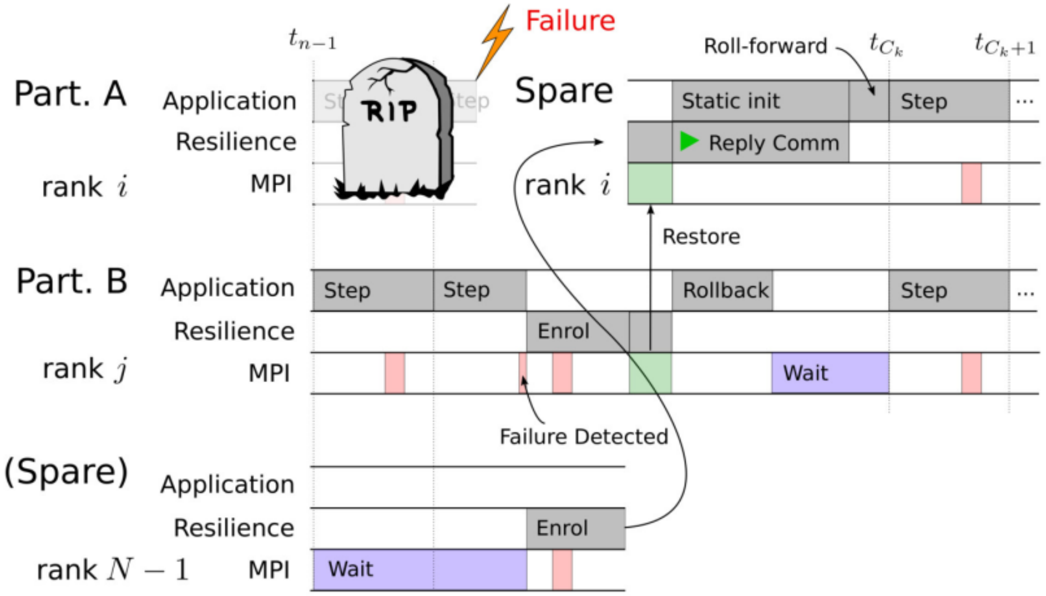
1. Multiple compute clusters/file systems – standard in operational NWP
2. Process replication – clone data and re-run computation
3. Use backup grids where checks are made



Fault detected on coarser backup grid: replace with previous known good value then map to full grid

Düben & Dawson '17

In-memory checkpointing



In-memory storage of distributed solver iterate

On failure

1. Revokes MPI comm
2. Recovers remote in-memory CP
3. Replaces failed process with spare

Computation continues, other procs unaffected

Attractive system resilience option matched with algorithmic fault-tolerance

Cantwell & Nielsen '18

Fault-tolerant linear solvers – FT -GCR

Resilience add-on for linear solver

ESCAPE elliptic solver dwarf

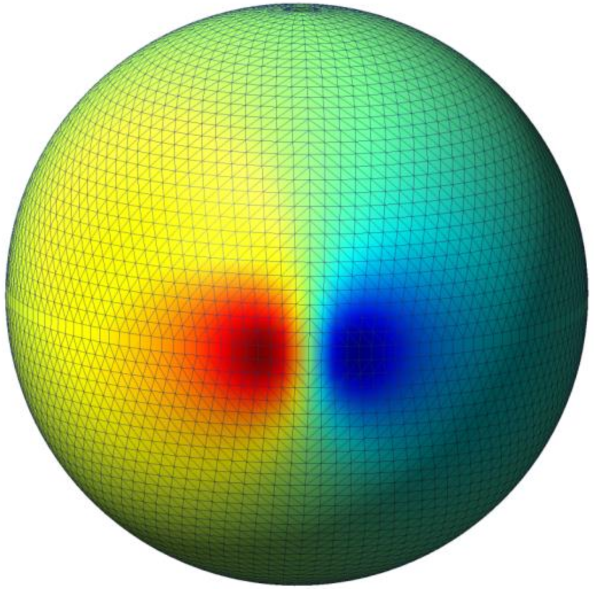
Mirroring IFS-FVM dynamical core solver

Monitor residual norm, if nondecreasing reset iteration in Krylov subspace

Algorithm 1 FT-GCR(k):

For any initial guess, ϕ^0, set $r^0 = \mathcal{L}(\phi^0) - \mathcal{R}$, $p^0 = \mathcal{P}^{-1}(r^0)$; then iterate:
for $n = 1, 2, \ldots$ until convergence **do**
  for $\nu = 0, \ldots, k-1$ **do**
    $\beta = -\dfrac{\langle r^\nu \mathcal{L}(p^\nu)\rangle}{\langle \mathcal{L}(p^\nu)\, \mathcal{L}(p^\nu)\rangle}$
    $\phi^{\nu+1} = \phi^\nu + \beta p^\nu$
    $r^{\nu+1} = r^\nu + \beta \mathcal{L}(p^\nu)$
    if $\|r^{\nu+1}\|_2 \leq \epsilon$ **then**
      exit
    end if
    if $\|r^{\nu+1}\|_2 \geq \|r^\nu\|_2$ **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
    $e = \mathcal{P}^{-1}(r^{\nu+1})$
    Compute $\mathcal{L}(e)$
    $\alpha_l = -\dfrac{\langle \mathcal{L}(e)\mathcal{L}(p^l)\rangle}{\langle \mathcal{L}(p^l)\, \mathcal{L}(p^l)\rangle} \qquad \forall\, l = 0, \ldots, \nu$
    $p^{\nu+1} = e + \sum_{l=0}^{\nu} \alpha_l p^l$
    $\mathcal{L}(p^{\nu+1}) = \mathcal{L}(e) + \sum_{l=0}^{\nu} \alpha_l \mathcal{L}(p^l)$
  end for
  reset $[\phi, r, p, \mathcal{L}(p)]^k$ to $[\phi, r, p, \mathcal{L}(p)]^0$
end for

Potential flow test on the sphere



$$\mathbf{v} = \mathbf{v_a} - \nabla\phi$$
$$\nabla \cdot (\rho\mathbf{v}) = 0$$

$\Rightarrow$

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

Fault simulation: injecting bit flips

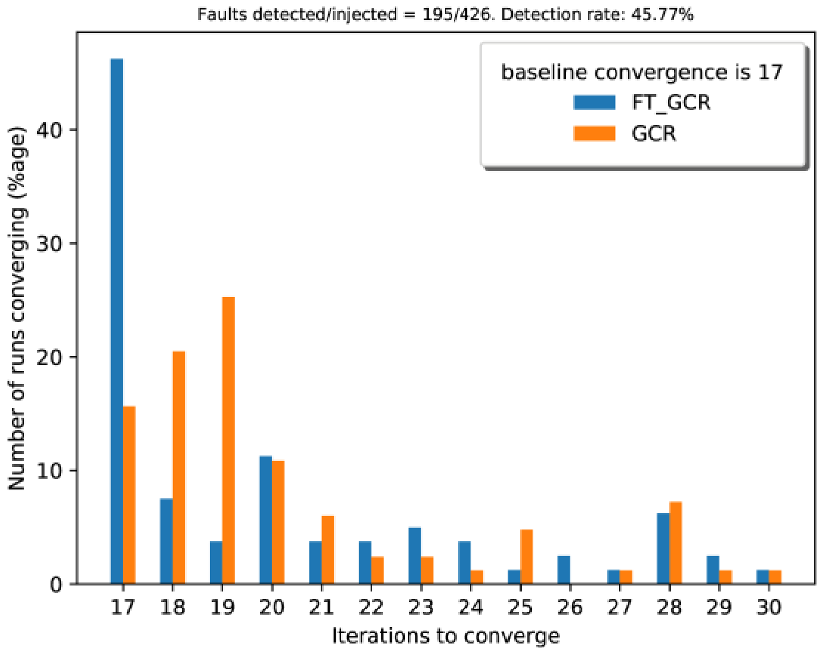
Comparing FT-GCR with unprotected GCR

O40 (~220 km) to O640 (~15 km) grids

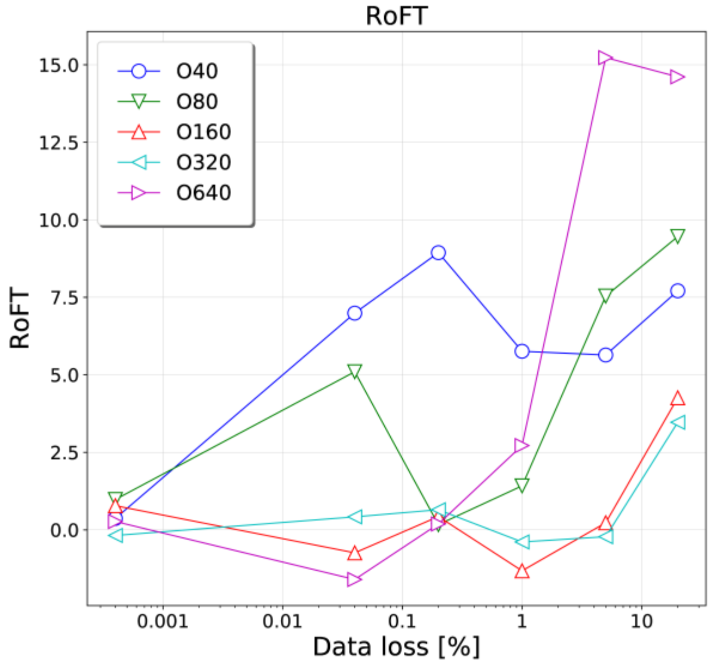
On up to 3240 MPI processes

Thomas et al. 2003

Results



O80 grid , 1% data loss



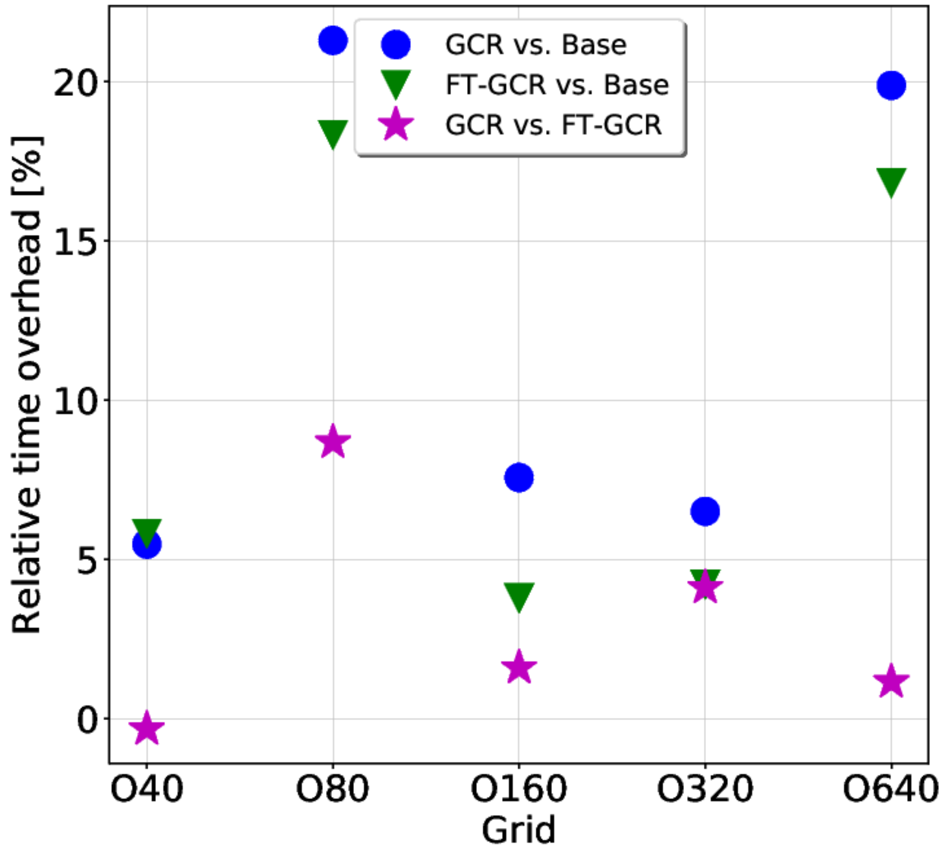
$$\mathrm{RoFT} = \frac{\mathrm{GCR} - \mathrm{FT\text{-}GCR}}{\mathrm{baseline}}$$

1. Protected runs converge in fewer iterations

2. Return on Fault Tolerance grows with > Larger data corruption
   > Higher detection rates

Gillard & B. '21

Results - Timings & costs



Faults have impact

GCR vs. Baseline > FT-GCR vs. Baseline

FT-GCR up to 10% faster than unprotected GCR

Memory overhead < ~20%

Gillard & B. '21

Limitations

Benchmarking of proposed solutions needs extension

Data size per process O(10-100) smaller than operational configs

Tests with set fault frequencies, limited testing on real systems

Take-home messages and next steps

Resilience techniques in NWP/climate models not future-proof

Lightweight novel approaches to be explored

Trials possible on exascale demonstrators such as ESCAPE-2 dwarfs

Future: performance trade-off studies on operational data volumes

Work with vendors to testing fault frequency on future HPC

Questions?

- Benacchio, T., Bonaventura, L., Altenbernd, M., Cantwell, C.D., Düben, P. D.,Gillard, M., Giraud, L., Göddeke, D., Raffin, E., Teranishi, K., Wedi, N. (2021), *Resilience and fault-tolerance in high-performance computing for numerical weather and climate prediction*, The International Journal of High Performance Computing Applications, Vol. 35, pp. 285-311

- Agullo, E., Altenbernd, M., Anzt, H., Bautista-Gomez, L., Benacchio, T., et al. (2021), *Resiliency in Numerical Algorithm Design for Extreme Scale Simulations*, accepted, https://arxiv.org/abs/2010.13342

- Gillard, M., Benacchio, T. (2021), *FT-GCR: a fault-tolerant generalized conjugate residual elliptic solver*, under revision, https://arxiv.org/abs/2103.07210

The ESCAPE-2 project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 800897.
This material reflects only the author's view and the Commission is not responsible for any use that may be made of the infomation it contains.