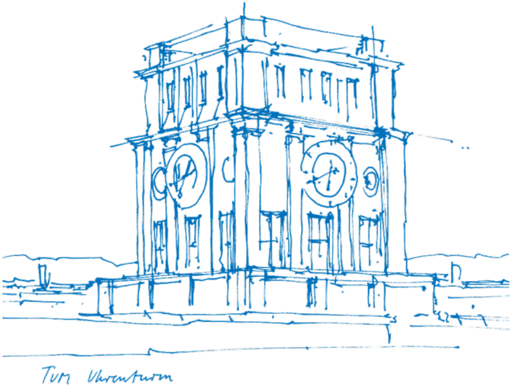
Towards Soft Error Resilience in SWE with TeaMPI

Bachelor's Thesis Talk

Atamert Rahma

Chair of Scientific Computing in Computer Science
Department of Informatics
Technical University of Munich

September 29th, 2021



Outline

Failures in HPC

Hard Errors $\Rightarrow$ stop execution
Soft Errors $\Rightarrow$ no permanent failure

Failures in HPC

Hard Errors $\Rightarrow$ stop execution
Soft Errors $\Rightarrow$ no permanent failure

Cosmic radiation $\rightarrow$ Energetic particles (Neutrons, Alpha particles) hit the silicon device $\rightarrow$ Cause a sufficient charge $\rightarrow$ Inverts the state of a logic device (bitflip) $\rightarrow$ Soft error

Soft errors may lead to

- **DUE** (Detectable and Uncorrectable Error)
- **SDC** (Silent Data Corruption)

Failures in HPC

Hard Errors $\Rightarrow$ stop execution
Soft Errors $\Rightarrow$ no permanent failure

Cosmic radiation $\rightarrow$ Energetic particles (Neutrons, Alpha particles) hit the silicon device $\rightarrow$ Cause a sufficient charge $\rightarrow$ Inverts the state of a logic device (bitflip) $\rightarrow$ Soft error

Soft errors may lead to

- **DUE** (Detectable and Uncorrectable Error)
- **SDC** (Silent Data Corruption)

Increased number of system components (CPU, memory) $\Rightarrow$ **higher error rates!**

Fault Tolerance in HPC

1. **CR** (Checkpoint/Restart)
2. **Replication** (of threads or processes)

Fault Tolerance in HPC

1. **CR** (Checkpoint/Restart)
2. **Replication** (of threads or processes)

CR overhead is increasing $\Rightarrow$ CR is not expected to be the best solution in the future

Checkpoints may include SDCs $\Rightarrow$ **CR alone cannot provide soft error resilience!**

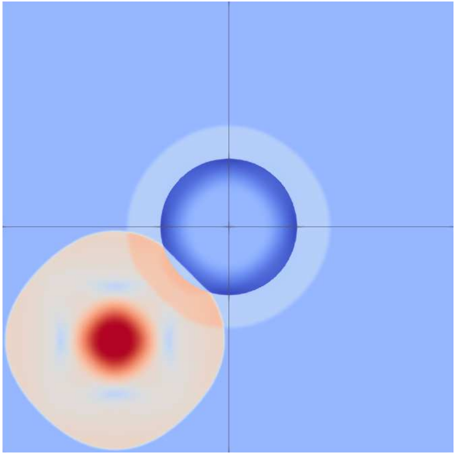
Outline

SWE for Solving Shallow Water Equations

SWE is a teaching code that can simulate

- different wave propagation/tsunami scenarios
- supports different parallel processing models (**MPI**, CUDA)

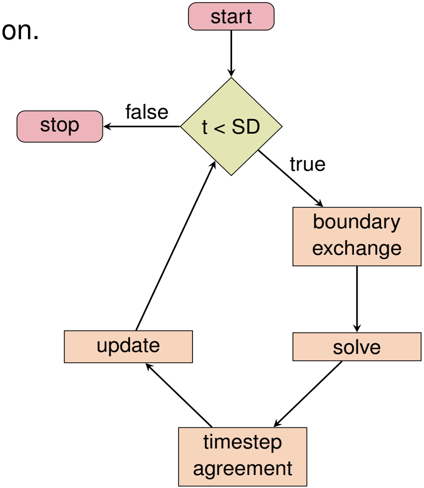
⇒ An example scenario where 4 MPI ranks were used.



SWE for Solving Shallow Water Equations

Computation loop of a simple SWE application.

- **t** = time
- **SD** = simulation duration



TeaMPI Library

Wrapper library for MPI that utilizes **process replication**.

TeaMPI Library

Wrapper library for MPI that utilizes **process replication**.

- **Heartbeats**: non-blocking messages between the replicated ranks
- **Task Sharing**: to reduce redundant computation time

Outline

1. Failures in HPC
2. SWE with TeaMPI
3. Integrating Soft Error Resilience Techniques into SWE
4. Comparison of Different Soft Error Resilience Techniques
5. Conclusion

Soft Error Detection with Hashes

Hashes

Idea: **Process replication** + **Hash value comparison** of the redundantly computed results

Soft Error Detection with Hashes

Hashes

Idea: **Process replication** + **Hash value comparison** of the redundantly computed results

- Data arrays of our application: **bathymetry**, **water height** and **momentum arrays**

Soft Error Detection with Hashes

Hashes

Idea: **Process replication** + **Hash value comparison** of the redundantly computed results

- Data arrays of our application: **bathymetry**, **water height** and **momentum arrays**
- **Single heartbeats** carry the hash values to replicas

Soft Error Detection with Hashes

Hashes

Idea: **Process replication** + **Hash value comparison** of the redundantly computed results

- Data arrays of our application: **bathymetry**, **water height** and **momentum arrays**
- **Single heartbeats** carry the hash values to replicas
- Comparison is handled **transparently** in TeaMPI

Soft Error Detection with Hashes

Hashes

Idea: **Process replication** + **Hash value comparison** of the redundantly computed results

- Data arrays of our application: **bathymetry**, **water height** and **momentum arrays**
- **Single heartbeats** carry the hash values to replicas
- Comparison is handled **transparently** in TeaMPI
- A mismatching hash value is assumed to be a sign of an SDC

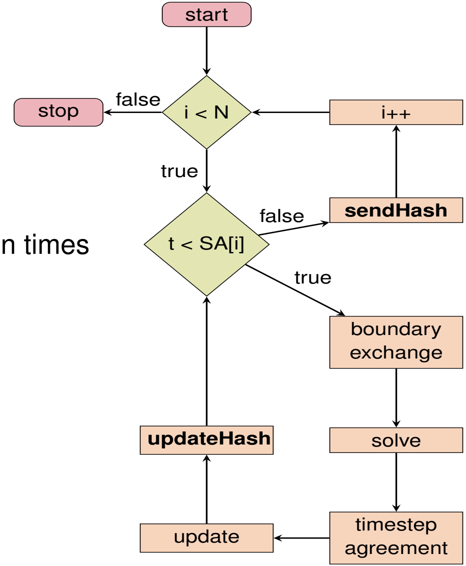
$\Rightarrow$ can provide SDC detection!

Hashes – Implementation

Computation loop for *Hashes*

- **t** = time, **N** = total hash sends
- **SA** = SendAt = equally distanced simulation times to send the hashes
- **i** = counter

This can only provide soft error detection!



Soft Error Resilience Using Verification and Task Sharing

Sharing

Idea: **Process replication** + **Admissibility checks** + **Task sharing**

Soft Error Resilience Using Verification and Task Sharing

Sharing

Idea: **Process replication** + **Admissibility checks** + **Task sharing**

- Admissibility validation of the data and **update** arrays in our solver scheme

Soft Error Resilience Using Verification and Task Sharing

Sharing

Idea: **Process replication** + **Admissibility checks** + **Task sharing**

- Admissibility validation of the data and **update** arrays in our solver scheme
- Recovery using one of the "healthy" replicas in case of violation

Soft Error Resilience Using Verification and Task Sharing

Sharing

Idea: **Process replication** + **Admissibility checks** + **Task sharing**

- Admissibility validation of the data and **update** arrays in our solver scheme
- Recovery using one of the "healthy" replicas in case of violation
- **Primary** and **secondary** blocks

Soft Error Resilience Using Verification and Task Sharing

Sharing

Idea: **Process replication** + **Admissibility checks** + **Task sharing**

- Admissibility validation of the data and **update** arrays in our solver scheme
- Recovery using one of the "healthy" replicas in case of violation
- **Primary** and **secondary** blocks

$\Rightarrow$ An undetected SDC may **propagate** to the replicated ranks and corrupt them as well due to task sharing!

Admissibility Criteria

1. Physical Admissibility Criteria
   - ☐ Constant Bathymetry
   - ☐ Non-negative Water Height
2. Numerical Admissibility Criteria
   - ☐ No Float Errors (NaNs)

Admissibility Criteria

1. Physical Admissibility Criteria
   - ☐ Constant Bathymetry
   - ☐ Non-negative Water Height
2. Numerical Admissibility Criteria
   - ☐ No Float Errors (NaNs)
   - ☐ Relaxed **DMP** (Discrete Maximum Principle)

$$\underbrace{\min_{y \in V_{i,j}} u(y, t^n)}_{\text{minimum neighbor}} - \delta \leq \overbrace{u^*(x, t^{n+1})}^{\text{candidate solution}} \leq \underbrace{\max_{y \in V_{i,j}} u(y, t^n)}_{\text{maximum neighbour}} + \delta$$

Admissibility Criteria

1. Physical Admissibility Criteria
   - ☐ Constant Bathymetry
   - ☐ Non-negative Water Height
2. Numerical Admissibility Criteria
   - ☐ No Float Errors (NaNs)
   - ☐ Relaxed **DMP** (Discrete Maximum Principle)

$$\underbrace{\min_{y \in V_{i,j}} u(y, t^n)}_{\text{minimum neighbor}} - \delta \leq \overbrace{u^*(x, t^{n+1})}^{\text{candidate solution}} \leq \underbrace{\max_{y \in V_{i,j}} u(y, t^n)}_{\text{maximum neighbour}} + \delta$$

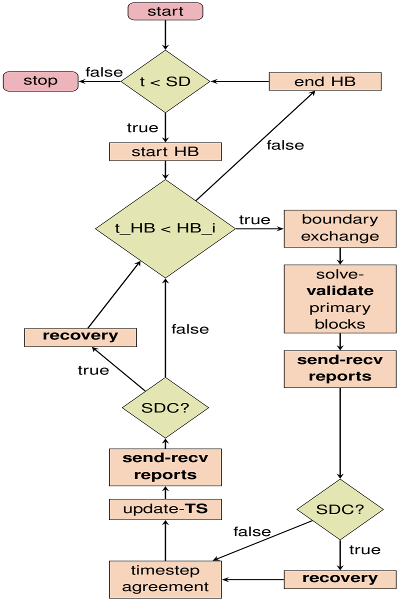
$\Rightarrow$ Relaxed DMP is not a strong condition and can give false positives!

Sharing – Implementation

Computation loop for *Sharing*

- **t** = time, **SD** = simulation duration
- **t_HB** = time since last heartbeat
- **HB_i** = heartbeat interval
- **TS** = task sharing

Additional soft error resilience with recovery!



Soft Error Resilience Using Redundant Computation

Redundant

Idea: **Process replication** + **Admissibility checks**

- Only **primary** blocks
- Teams run independently (**no task sharing**)

Soft Error Resilience Using Redundant Computation

Redundant

Idea: **Process replication** + **Admissibility checks**

- Only **primary** blocks
- Teams run independently (**no task sharing**)

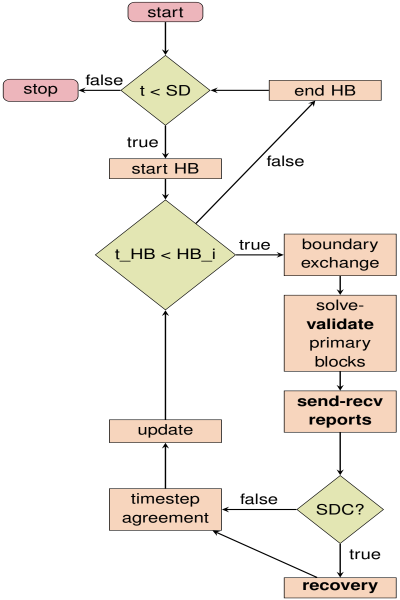
$\Rightarrow$ A possible SDC in one team **cannot propagate** to other teams!

Redundant – Implementation

Computation loop for *Redundant*

- **t** = time, **SD** = simulation duration
- **t_HB** = time since last heartbeat
- **HB_i** = heartbeat interval

Additional soft error resilience with redundant computation!



Outline

Bitflip Injections

We inject only a single bitflip per run. Bitflips are injected

- at a fixed location in the source code,
- into the data and update arrays at a random number and bit.

Soft Error Outcome Rates
Random Bitflip Injections

10000 injections into each listed array below (random injection with 30000 injections).

Sharing ⇒

$outcome\backslash array$	*h*	*hu*	*hv*	updates	random
Correctable	18.55%	13.83%	14.47%	**0%**	15.22%
DUE	0.25%	0.13%	0.04%	7.32%	4.36%
SDC	81.21%	86.03%	85.49%	92.68%	80.41%
Negligible	2.95%	3.26%	3.31%	28.17%	19.84%

Redundant ⇒

$outcome\backslash array$	*h*	*hu*	*hv*	updates	random
Correctable	18.6%	13.63%	15%	**6.09%**	19.06%
DUE	0.03%	0.1%	0.04%	0.96%	0.43%
SDC	81.37%	86.26%	84.96%	92.95%	80.52%
Negligible	2.91%	3.32%	2.95%	28.09%	19.56%

Soft Error Outcome Rates

Relaxation Factor (δ)

Correctable outcome rates of 2000 injections into the leftmost 10 bits of randomly selected floats.

	Sharing				*Redundant*			
$\delta \backslash array$	*h*	*hu*	*hv*	updates	*h*	*hu*	*hv*	updates
80	**59.44%**	**43.37%**	**50.44%**	**0%**	**60.34%**	**45.31%**	**50.7%**	**15.76%**
100	59.95%	43.47%	47.07%	0%	58.97%	43.91%	47.74%	15.34%
10000	39.69%	29.91%	29.24%	0%	39.96%	28.1%	30.62%	11.21%
1000000	**34.9%**	**22.07%**	**25.32%**	**0%**	**37.68%**	**24%**	**25.46%**	**10.16%**

Soft Error Outcome Rates

Type of the Simulation Scenario (s)

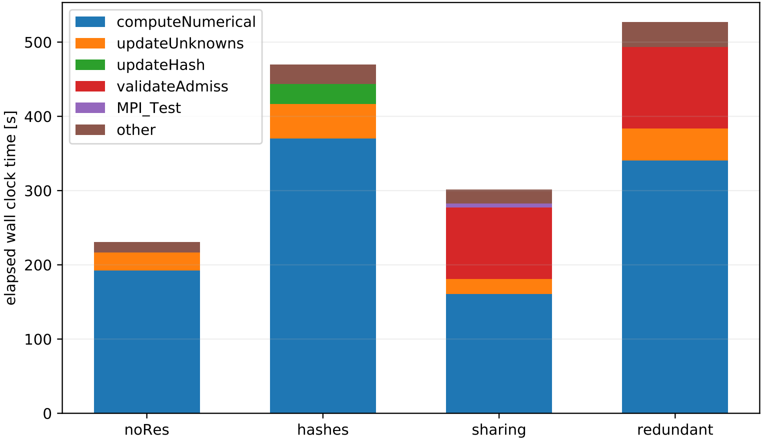
Correctable outcome rates of 2000 injections into the leftmost 10 bits of randomly selected floats.

- A = *radialBathymetryDamBreak* $\rightarrow$ lower water height (~15.1 meters)
- B = *splashingPool* $\rightarrow$ higher water height (~245 meters)
- C = *seaAtRest* $\rightarrow$ constant water height (10 meters)

	Sharing				*Redundant*			
$s\backslash array$	*h*	*hu*	*hv*	updates	*h*	*hu*	*hv*	updates
A	59.95%	43.47%	47.07%	0%	58.97%	43.91%	47.74%	15.34%
B	89.77%	89.39%	90.28%	0%	99.95%	90.1%	90.5%	15.99%
C	58.25%	0%	0%	0%	61.65%	0%	0%	0%
C (r=0)	100%	100%	100%	0%	100%	100%	100%	99.8%

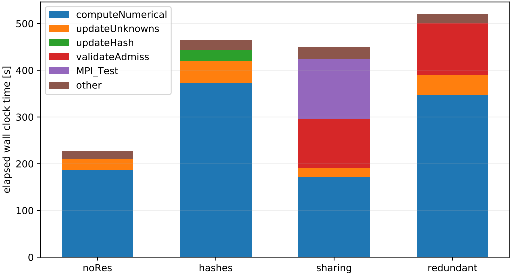
Performance Comparison – Single Node

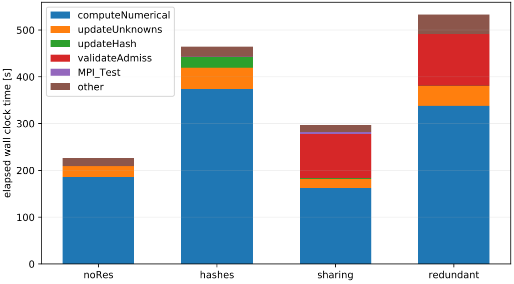
Profiling on a single node. (**noRes** = no resilience)



Performance Comparison – Multiple Nodes

Teams are mapped onto the same node vs. Replicas are mapped onto the same node

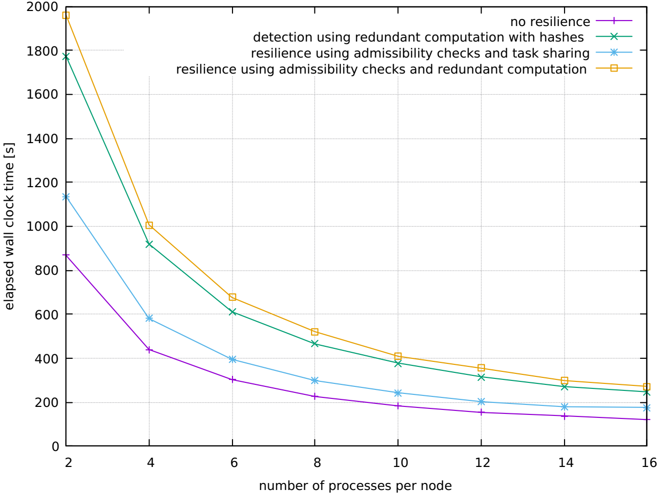




$\Rightarrow$ increased communication between teams due to task sharing!

Performance Comparison – Multiple Nodes

Strong Scaling – replicas on the same node



Outline

Summary & Future Work

1. *Hashes*: Soft error **detection** using hash comparison
2. *Sharing*: Soft error recovery using **validation** and **task sharing**
3. *Redundant*: Soft error recovery using **validation** and **redundant computation**

Summary & Future Work

1. *Hashes*: Soft error **detection** using hash comparison
2. *Sharing*: Soft error recovery using **validation** and **task sharing**
3. *Redundant*: Soft error recovery using **validation** and **redundant computation**

$\Rightarrow$ Resilience depends on different factors (δ, simulation duration and type).
$\Rightarrow$ *Redundant* can **additionally recover from some SDC cases** that *Sharing* cannot.
$\Rightarrow$ *Sharing* has the **the best performance**.

Summary & Future Work

1. *Hashes*: Soft error **detection** using hash comparison
2. *Sharing*: Soft error recovery using **validation** and **task sharing**
3. *Redundant*: Soft error recovery using **validation** and **redundant computation**

$\Rightarrow$ Resilience depends on different factors (δ, simulation duration and type).
$\Rightarrow$ *Redundant* can **additionally recover from some SDC cases** that *Sharing* cannot.
$\Rightarrow$ *Sharing* has the **the best performance**.

Future work:

- hard error resilience evaluation together with our resilience techniques
- more random bitflip injections (different areas in the code)
- additional admissibility criteria and improved recovery to cover more SDC cases