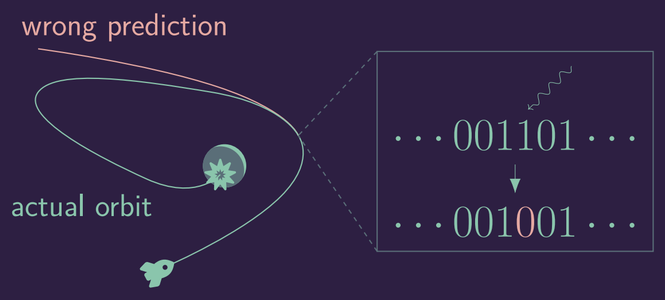
Job Level Communication-Avoiding Detection and Correction of Silent Data Corruption in HPC Applications

Laslo Hunhold and Stefan Wesner (adviser)
Parallel and Distributed Systems Group, University of Cologne

UNIVERSITY OF COLOGNE

Motivation

- *Silent Data Corruption (SDC)*: Computing error undetected during execution.
  - Mostly caused by silent hardware defects and ionizing particles.
  - Rate increases with system size (every few minutes in the exascale[1]).
- Even small computing errors can be devastating for many applications.



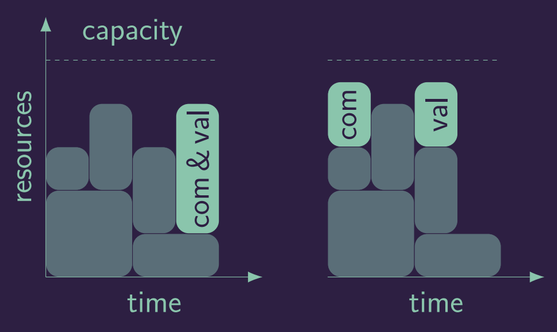
- All program steps are at risk; method robustness is only a partial solution.

State of the Art

- *General SDC Detection*: Run computation twice and compare results.
- MPI-based approaches have up to 823x synchronisation overhead[3][4].
- *Selective instruction duplication*[2] just provides partial CPU coverage.
- *General problem*: Computation and validation are coupled within the job.

Goals

- Decouple computation and validation into separate jobs for each job.
- Higher job scheduling efficiency.



Methodology

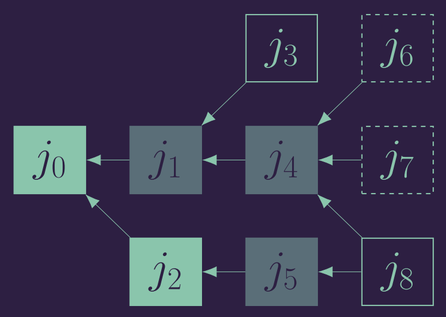
- *Assumption*: Jobs are reschedulable and idempotent, DAG job model.
- Schedule each job j twice and reschedule until two local output data hashes match (majority vote).
- Jobs depending on j do not wait for its validation, rescheduled if invalid.

Example

Job legend

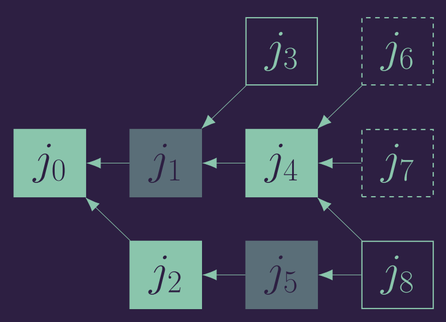
- j pending
- j running
- j finished, waiting for validation
- j validated
- j invalidated

1. **Initial state**



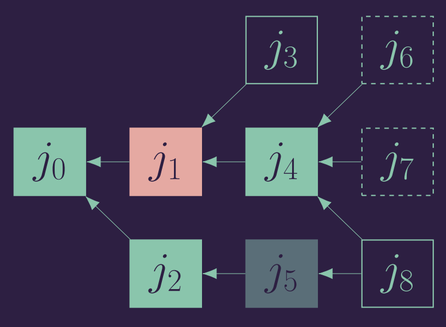
The running jobs j_3 and j_8 are already using the output data of the yet unvalidated jobs j_1, j_4 and j_5.

2. **Validation of job j_4**



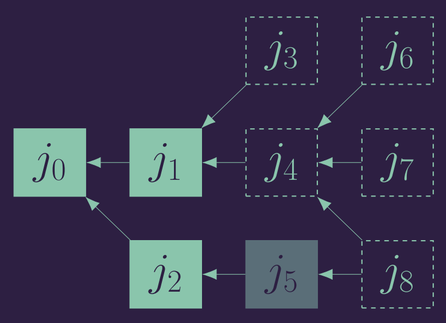
Validation happens independently from the implicit DAG-hierarchy.

3. **Error detection in job j_1**



SDC affected the results of j_1 and thus all its subsequent jobs.

4. **Rescheduling of j_1-subtree**



Subsequent jobs of j_1 are rescheduled to depend on its validated run.

Proposed Implementation

- SLURM workload manager plugin.
- Minimal necessary batch script modifications.
- Metadata is stored in the job's `SystemComment`.
- Node-overlap between job runs is avoided with `ExcNodeList`.
- Invalid completed jobs are invalidated via `DerivedExitCode`.

Conclusion

- *Novelty*: SDC detection and correction at the general DAG job level.
- *More efficient scheduling*:
  - Validation is independent from job DAG hierarchy.
  - No validation-stalling of subsequent tasks.
- *Adaptive optimization*: Change validation priority depending on SDC error rate and rescheduling cost.

References

[1] David Fiala et al. 'Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing'. In: *SC '12* (Salt Lake City, UT, USA). IEEE, Feb. 2013, pp. 1–12. DOI: 10.1109/SC.2012.49.

[2] Yafan Huang et al. 'Mitigating Silent Data Corruptions in HPC Applications across Multiple Program Inputs'. In: *SC '22* (Dallas, TX, USA). IEEE, Feb. 2023, pp. 1–14. DOI: 10.1109/SC41404.2022.00022.

[3] P. Samfass et al. 'Doubt and Redundancy Kill Soft Errors—Towards Detection and Correction of Silent Data Corruption in Task-based Numerical Software'. In: *2021 IEEE/ACM 11th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2021, pp. 1–10. DOI: 10.1109/FTXS54580.2021.00005.

[4] Guozhen Zhang et al. 'Efficient detection of silent data corruption in HPC applications with synchronization-free message verification'. In: *The Journal of Supercomputing* 78 (June 2021), pp. 1381–1408. DOI: 10.1007/s11227-021-03892-4.