Parallaft: Runtime-based CPU Fault Tolerance via Heterogeneous Parallelism

Boyue Zhang [1] Sam Ainsworth [2] Lev Mukhanov [3] Timothy M. Jones [1]

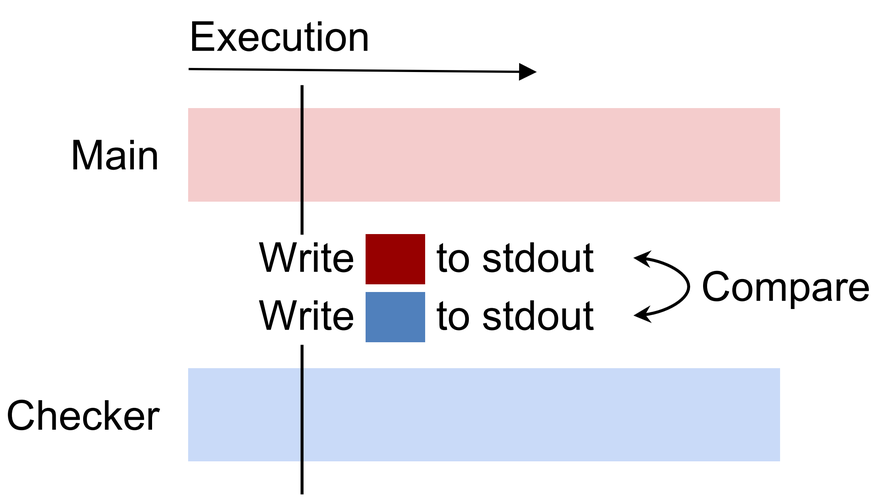
[1]University of Cambridge [2]University of Edinburgh [3]Queen Mary University London

Summary

- CPUs are increasingly susceptible to **errors** leading to program **crashes** or **data corruption**.
- Existing software-based detection techniques suffer from **high power and performance overhead**, while current hardware schemes remain **too complex for production deployment**.
- We propose **Parallaft**, a **runtime-based** error-detection solution that exploits **parallelism** in the error-checking computation, by slicing it into **independent segments**, combined with the use of **heterogeneous** processors to **minimise power and performance overhead**.
- Our evaluation on an Apple M2 processor shows Parallaft **halves the energy overhead** compared with the previous state-of-the-art runtime-based solution, RAFT [1] while maintaining **comparable performance overhead**.

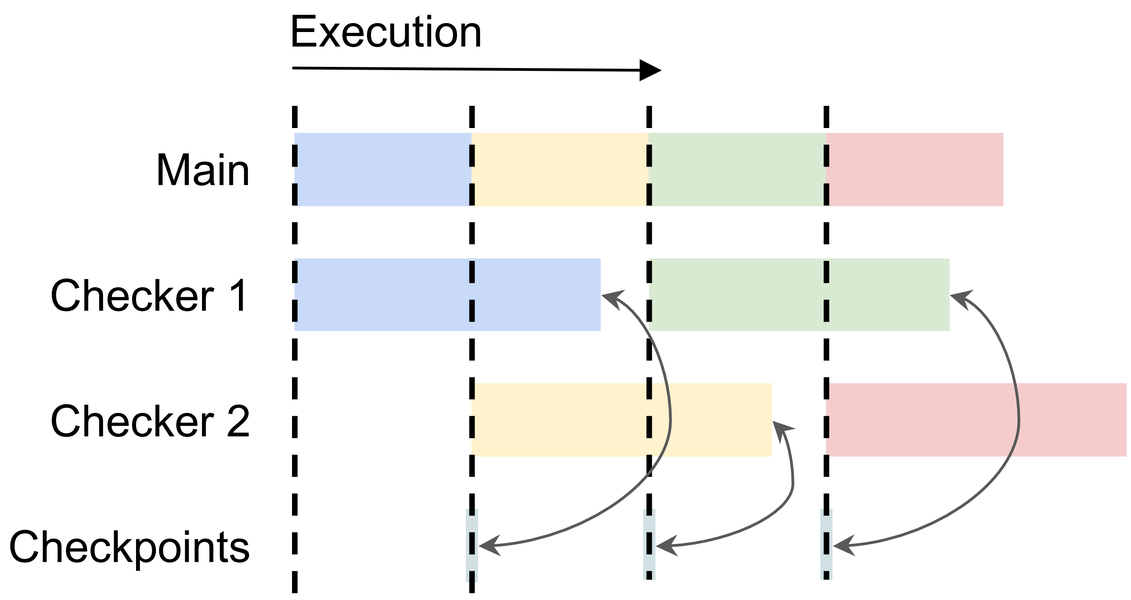
Previous state-of-the-art, RAFT [1]

- **RAFT** is a runtime-based solution that **runs the program twice in parallel on two different CPU cores**. The results from the two executions, such as system calls that write to the standard output, are **compared at runtime**.



- However, RAFT requires **2× CPU dynamic power**.

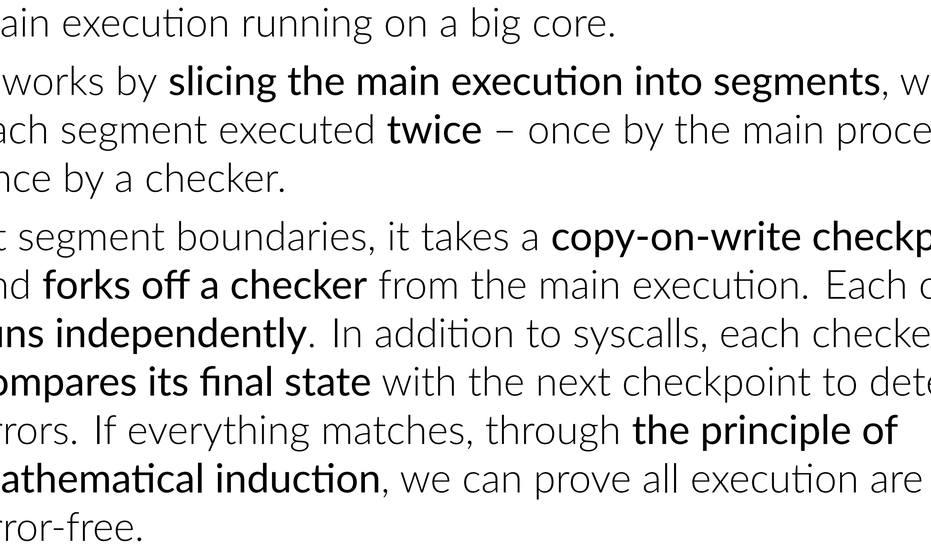
Parallaft's solution

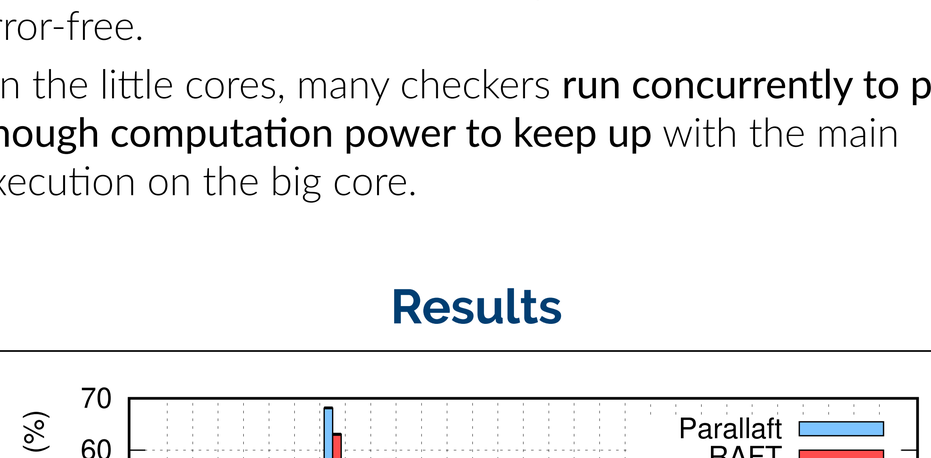


Parallaft's operation

- Parallaft improves energy efficiency by **running error checkers on multiple little cores in a heterogeneous processor**, with the main execution running on a big core.
- It works by **slicing the main execution into segments**, with each segment executed **twice** – once by the main process and once by a checker.
- At segment boundaries, it takes a **copy-on-write checkpoint** and **forks off a checker** from the main execution. Each checker **runs independently**. In addition to syscalls, each checker also **compares its final state** with the next checkpoint to detect errors. If everything matches, through **the principle of mathematical induction**, we can prove all execution are error-free.
- On the little cores, many checkers **run concurrently to provide enough computation power to keep up** with the main execution on the big core.

Results





Parallaft only incurs **15.9% performance overhead and 44.3% energy overhead**, compared with 16.2% performance overhead and 87.8% energy for RAFT under like-for-like threat models on our Apple M2 machine.

References

[1] Yun Zhang, Soumyadeep Ghosh, Jialu Huang, Jae W. Lee, Scott A. Mahlke, and David I. August. Runtime asynchronous fault tolerance via speculation. In *Proceedings of the Tenth International Symposium on Code Generation and Optimization (CGO)*, 2012. doi: 10.1145/2259016.2259035.

bz275@cam.ac.uk