A Generic Reusable Java Framework for Fault-Tolerant Parallelization with the Task Pool Pattern

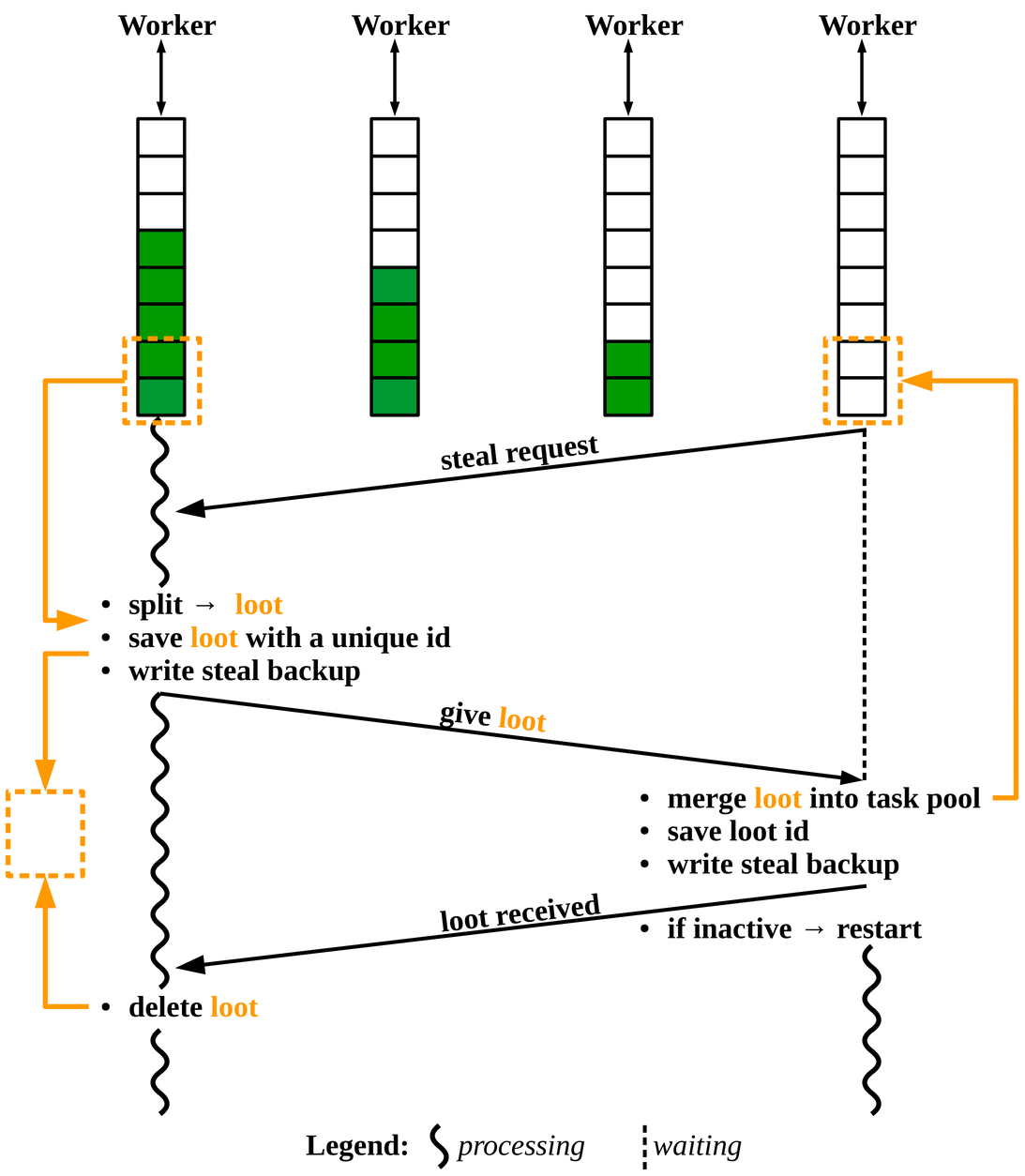
Jonas Posner
Reseach Group Programming Languages / Methodologies
University of Kassel, Germany
jonas.posner@uni-kassel.de

UNIKASSEL VERSITÄT

Overview

- We developed a fault-tolerant cooperative work stealing technique and implemented it in a reusable framework.
- Users can focus on coding sequential tasks for their problem.
- The framework is written in Java to facilitate wide usage.
- The APGAS library is utilized for parallel programming.
- Backup data are stored in Hazelcast's distributed and fault-tolerant `IMap`.
- Backups of each local task pool are written at fixed times, in the event of work stealing, and during recovery.
- Loot is separately stored during steal operations.
- The number of backup copies can be configured to control how many simultaneous failures are tolerated.
- If a worker fails, each worker automatically takes appropriate action, e.g., restores tasks.
- The algorithm is correct in the sense that the computed result is the same as in non-failure case, or the program aborts with an error message.

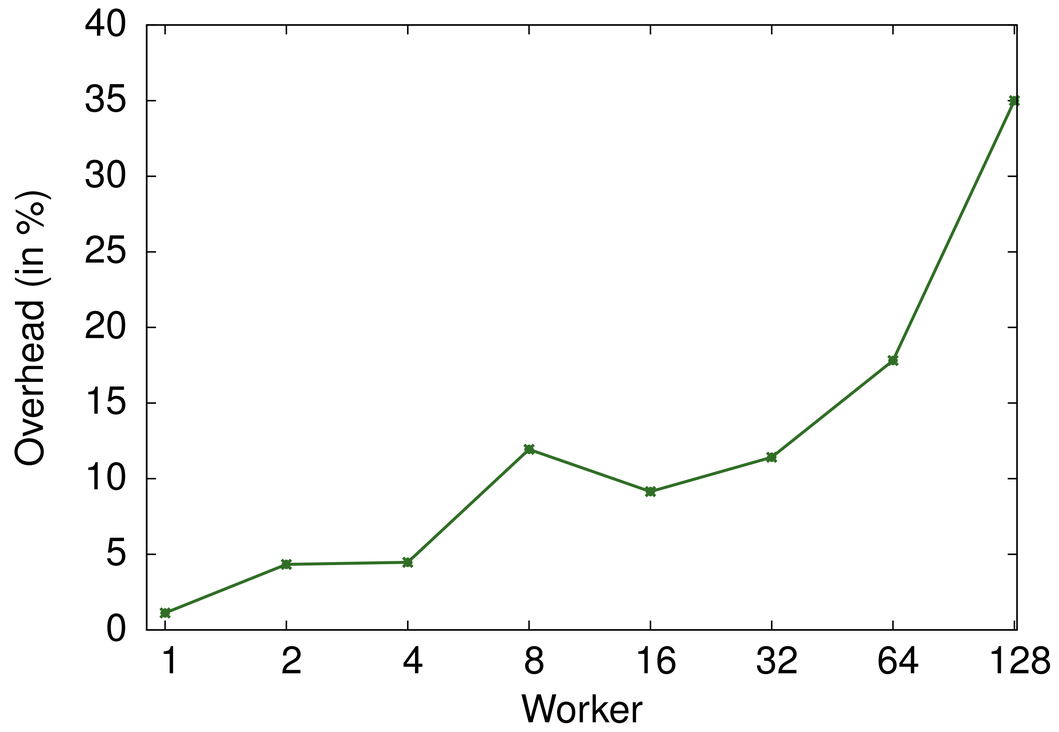
Steal Protocol



Worker Main Loop

```
while (tasks available) {
  while (task pool is not empty) {
    synchronized (worker object) {
      process up to n tasks;
      for each recorded steal request {
        write steal backup;
        send tasks to recorded thief;
      }
      if (k == currentK++)
       write regular backup;
    }
  }
  synchronized (worker object) {
    try to steal from up to w+z victims;
  }
}
write final backup;
```

Fault Tolerance Overhead with BC



References

[1] Jonas Posner and Claudia Fohry. Cooperation vs. coordination for lifeline-based global load balancing in APGAS. In *Proc. ACM SIGPLAN X10 Workshop*, 2016.

[2] Jonas Posner and Claudia Fohry. Fault tolerance for cooperative lifeline-based global load balancing in Java with APGAS and Hazelcast. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2017.

Acknowledgments

This work is supported by the Deutsche Forschungsgemeinschaft, under grant FO 1035/5-1. Experiments were carried out on the Lichtenberg High Performance Computer of TU Darmstadt, under project ID 382.