Aegis: Partitioning Data Block for Efficient Recovery of Stuck-at-Faults in Phase Change Memory

Jie Fan, Jiwu Shu,
Youhui Zhang, and
Weimin Zhen

Song Jiang

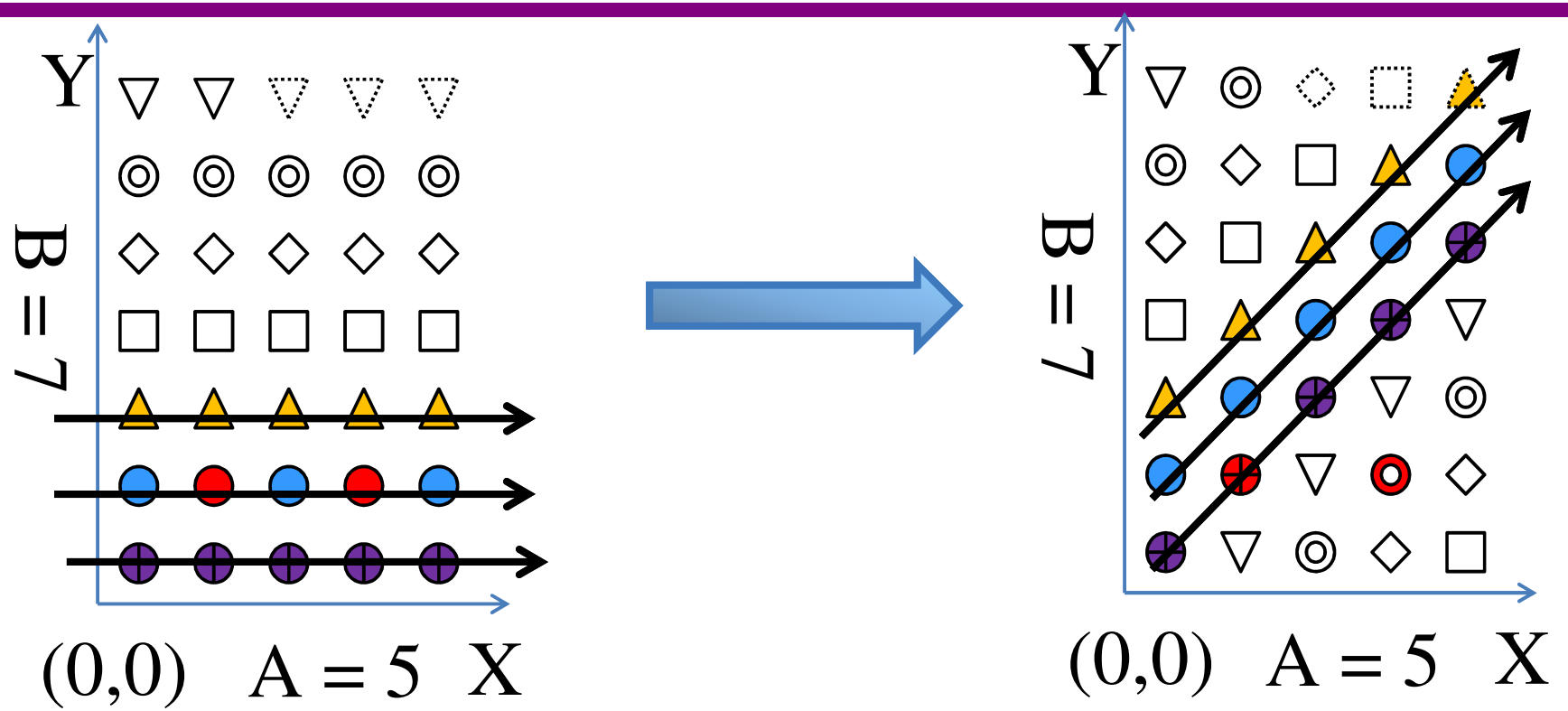
清華大學
Tsinghua University

WAYNE STATE UNIVERSITY

Stuck-at Faults in PCM

- PCM has limited endurance.
- Stuck-at faults occur when memory cell fails to change its value.
  - It is a major type of faults in PCM.
  - This type of faults is permanent and accumulates.
  - Values in such faulty cells can still be read.
- Inversion-based correction
  - Partition data block into a number of groups and exploit the fact that stuck-at values are still readable (e.g., SAFER).
  - Each group can tolerate only one fault.
- **Proposal of an efficient partition scheme separating faults into different groups.**

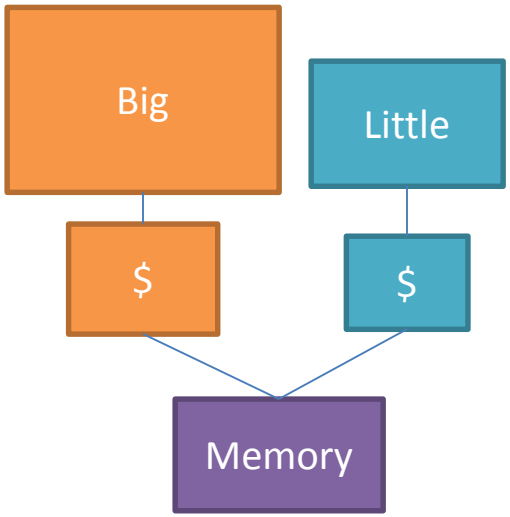
Illustration of Aegis Partition



- PCM bits are laid out on an A × B Cartesian plane.
- Aegis considers all points on a line as a group.
- **Any two bits in the same line will not be in the same line** after Aegis changes slope of the lines.
- Aegis distributes faults more evenly to tolerate more faults with lower overhead.

Fine-grained Heterogeneity

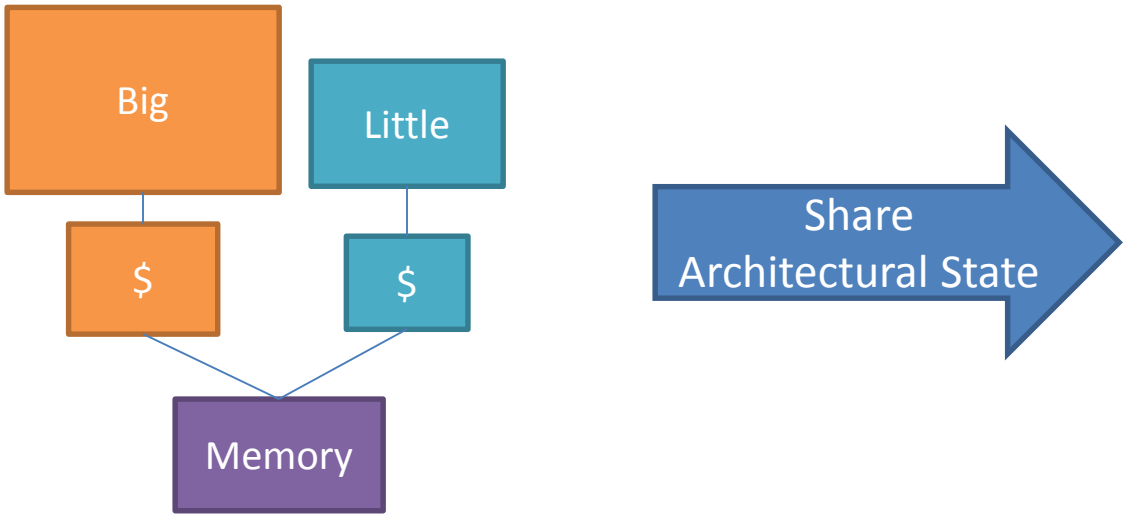
Traditional big.LITTLE Architecture



Transfer Overhead: ~**20K** cycles

Fine-grained Heterogeneity

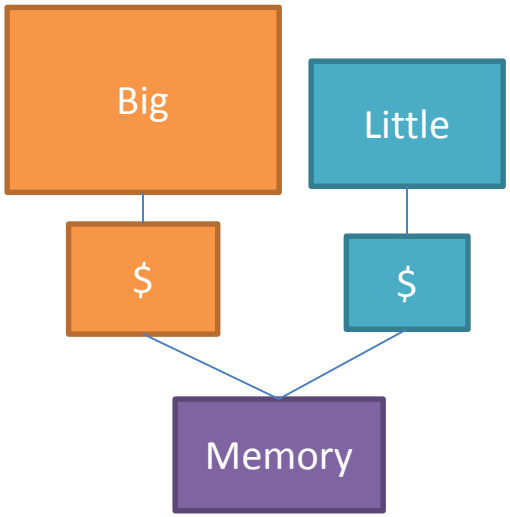
Traditional big.LITTLE Architecture



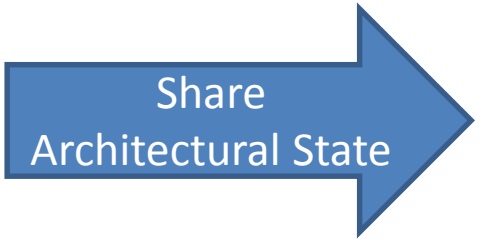
Transfer Overhead: ~**20K** cycles

Fine-grained Heterogeneity

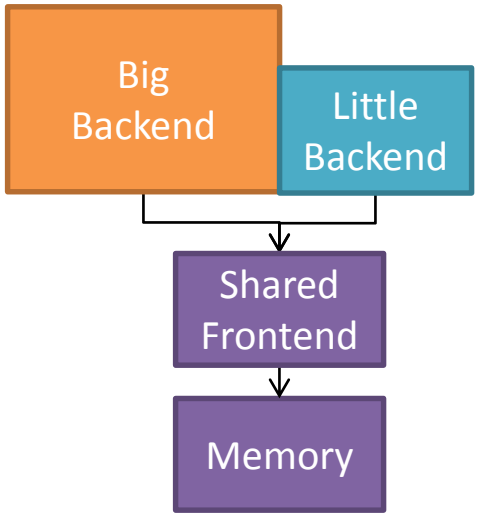
Traditional big.LITTLE Architecture



Transfer Overhead: ~**20K** cycles

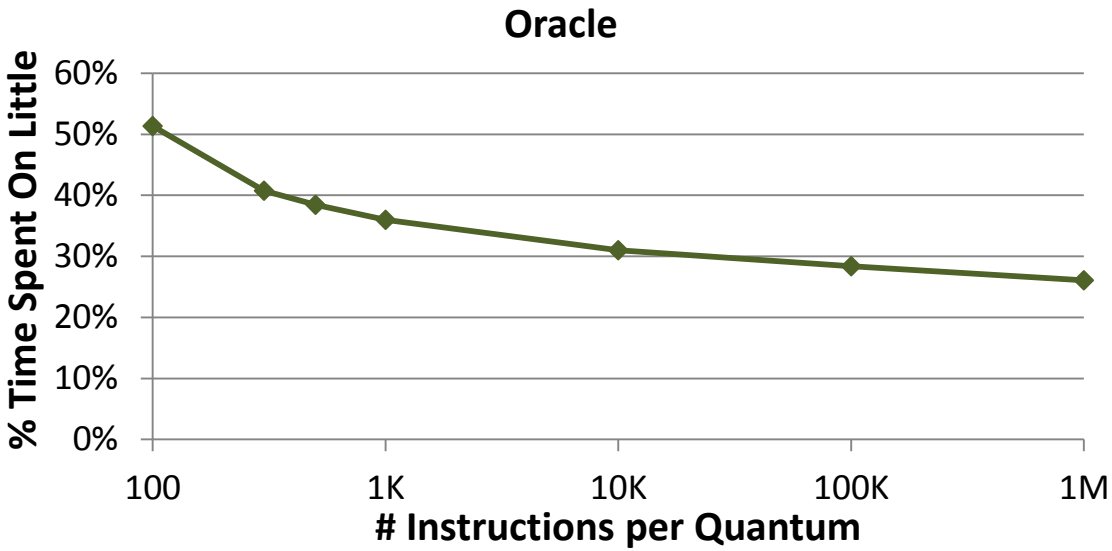


Composite Cores Architecture

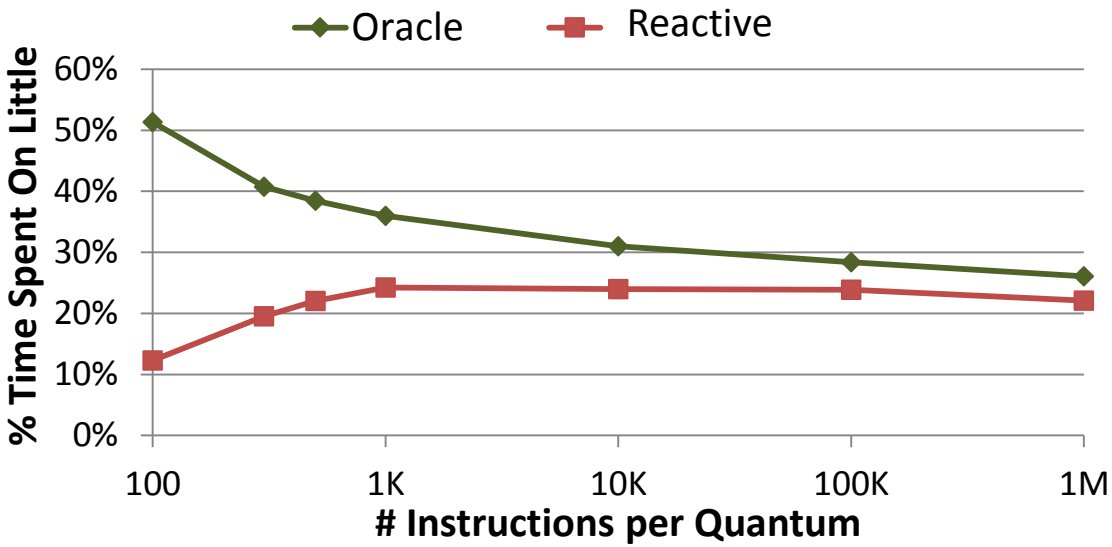


Transfer Overhead: ~**35** cycles

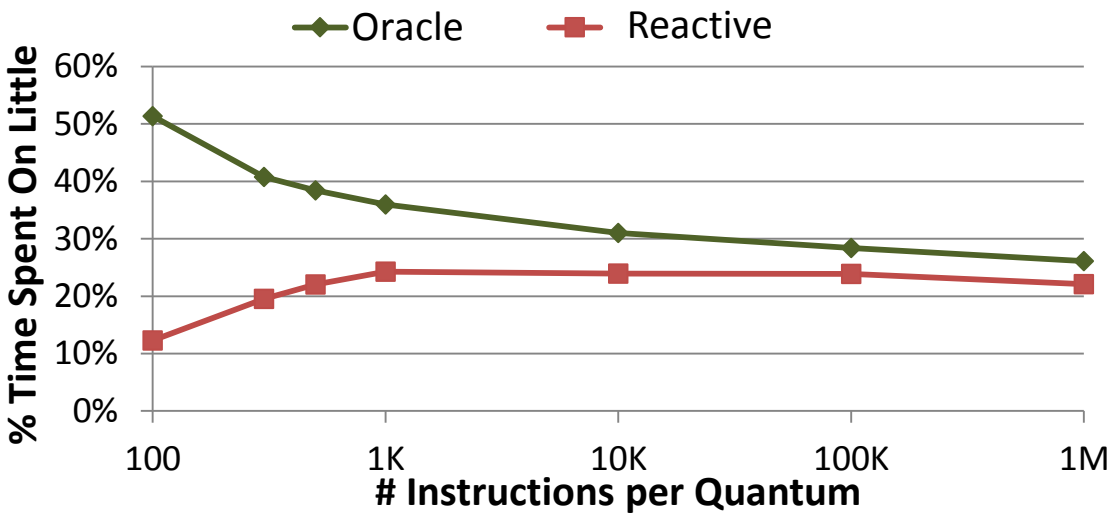
Traditional Reactive Controllers



Traditional Reactive Controllers

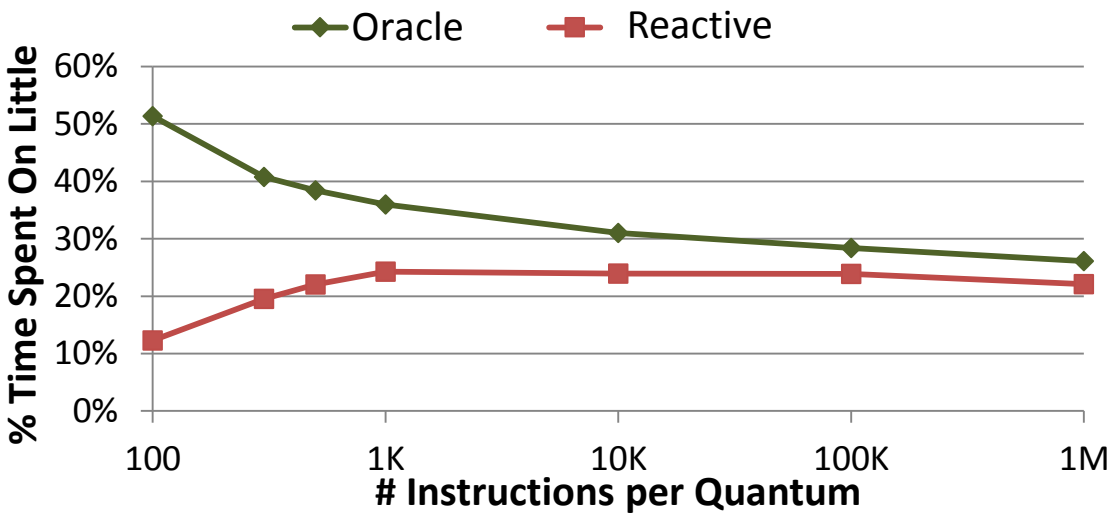


Traditional Reactive Controllers



Don’t React – Predict!

Traditional Reactive Controllers

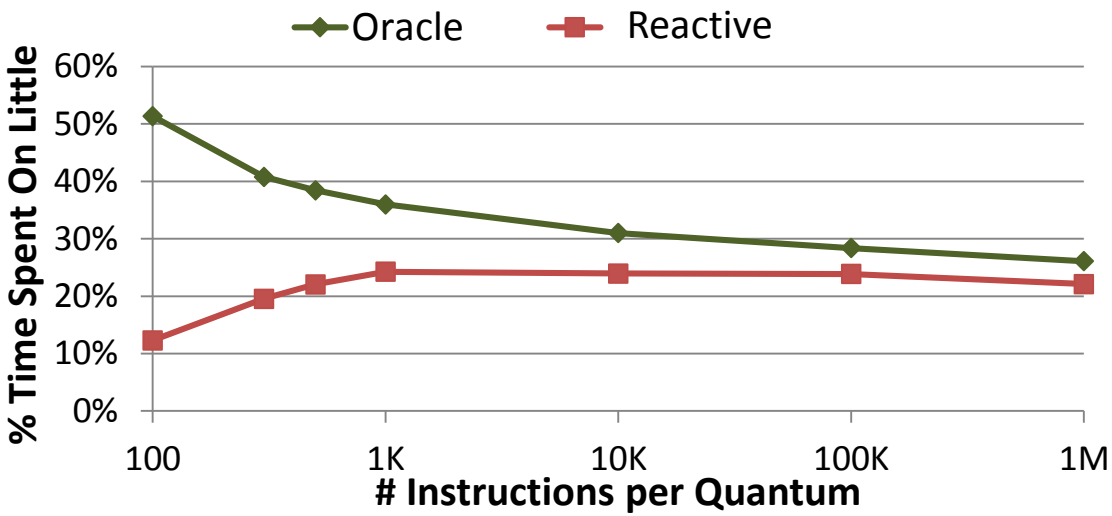


Don't React – Predict!

Code repeats (loops, functions)

Behavior repeats in the same program context

Traditional Reactive Controllers



Don't React – Predict!

Code repeats (loops, functions)

Behavior repeats in the same program context