**Deterministic Multiprocessor Replay**

**Motivation:** Deterministic replay helps debug multithreaded programs, provide fault tolerance, aid computer forensics, etc.

**Current approach:** Run each thread of the program on a separate processor
- Log all shared-memory dependencies for offline replay
- Log subset of dependencies: search for valid execution

**Challenge:** Logging and replaying shared-memory dependencies adds significant runtime overhead
- Log subset: cannot guarantee execution replay

**Insight**

**Epoch-parallel execution**
- Timeslice multiple threads on a single processor
- Run multiple intervals (epochs) concurrently on separate processors to achieve scalability

**Benefit:** Only one thread in an epoch executes on the processor at any given time
- No need to log shared-memory dependencies as threads never simultaneously access the same memory
- Instead, log thread schedule (low overhead)

**DoublePlay**

- Thread-parallel execution provides a speculative log of non-deterministic events (syscalls, signals, synchronization operations) and epoch checkpoints.

- Epoch-parallel execution runs epochs in parallel from checkpoints, and uses online replay to verify that the logged results can reproduce the execution. A verified log and each epoch’s thread schedule guarantee deterministic replay.

- Offline replay proceeds with no additional overhead.