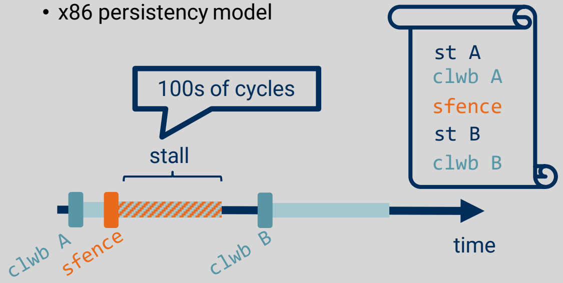
COSPlay: Using Coroutines for High-Throughput Synchronous Persistence

Marina Vemmou, Alexandros Daglis

School of Computer Science, Georgia Institute of Technology

Challenges of Persistent Memory

- ➢PM applications: need to be crash consistent
  - **Challenge: Volatile order ≠ persistent order**
- ➢Must explicitly control order of PM updates
  - x86 persistency model

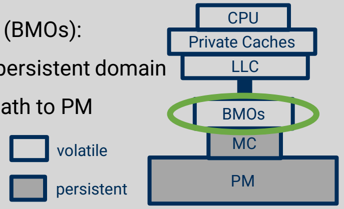


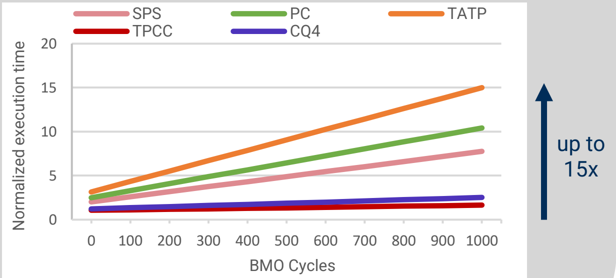
```
st A
clwb A
sfence
st B
clwb B
```

Persist Latency and BMOs

Backend Memory Operations (BMOs):

- performed before reaching persistent domain
- can add 100s of cycles on path to PM

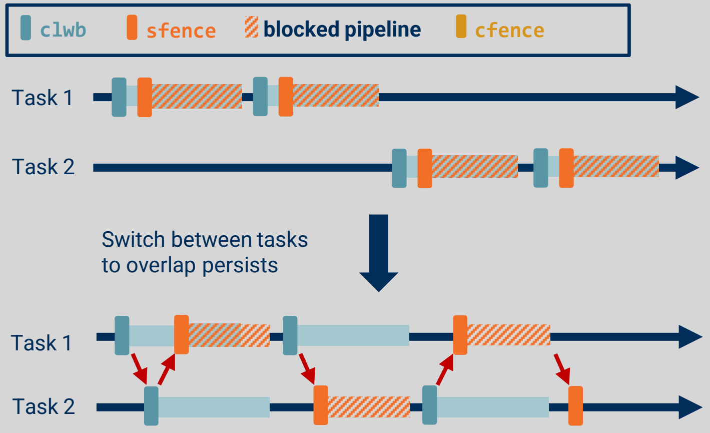




x86 persistence + high persist latency = slowdown up to 15x

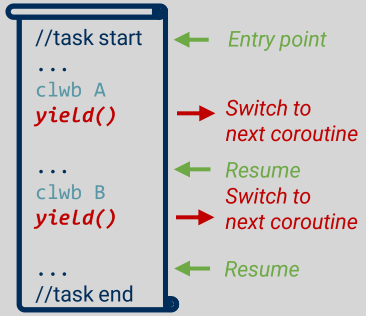
Hiding Persist Latency Through Parallelism

➢Key insight: many PM programs have inherent task parallelism



Switch between tasks to overlap persists

➢Coroutines: rapid task switching



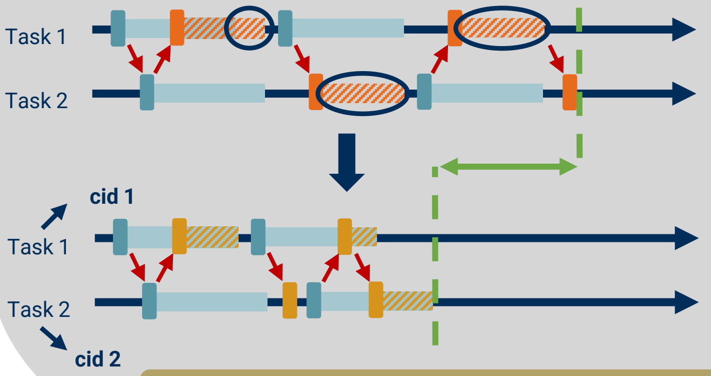
```
//task start
...
clwb A
yield()

...
clwb B
yield()

...
//task end
```

Coroutines for ns-scale task switching

Context-Aware Hardware for Selective Persist Ordering



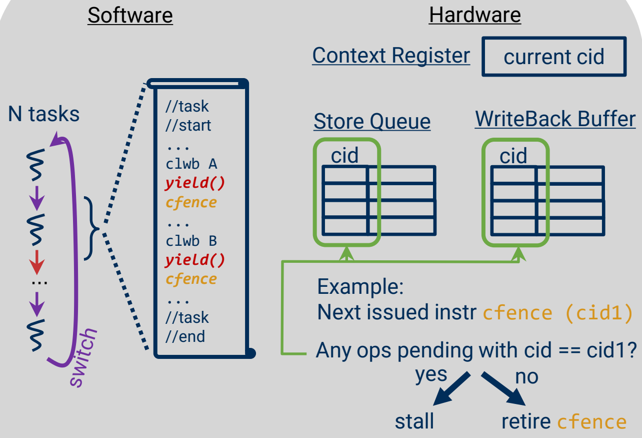
sfence waits for pending clwbs of all tasks

Insight: only need to maintain intra-task order

- ➢Each coroutine given a context-id (cid)
- ➢Each clwb associated with current cid
- ➢Introducing cfence: waits only for clwbs of a *specific* cid

COSPlay: relax inter-task persist order, preserve intra-task persist order

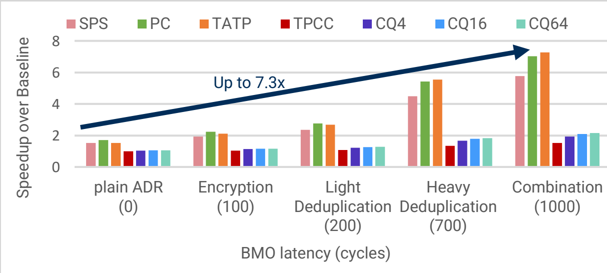
Putting It All Together



```
//task
//start
...
clwb A
yield()
cfence
...
clwb B
yield()
cfence
...
//task
//end
```

Example:
Next issued instr cfence (cid1)
Any ops pending with cid == cid1?

COSPlay Performance



- ➢Speedup even without BMOs
- ➢Speedup increases with BMO latency

COSPlay improves throughput of PM programs by up to 7.3x

This work appeared at the 54th International Symposium on Microarchitecture (MICRO'21)