CASPAR: BREAKING SERIALIZATION IN LOCK-FREE MULTICORE SYNCHRONIZATION

Tanmay Gangwani
University of Illinois, Urbana Champaign

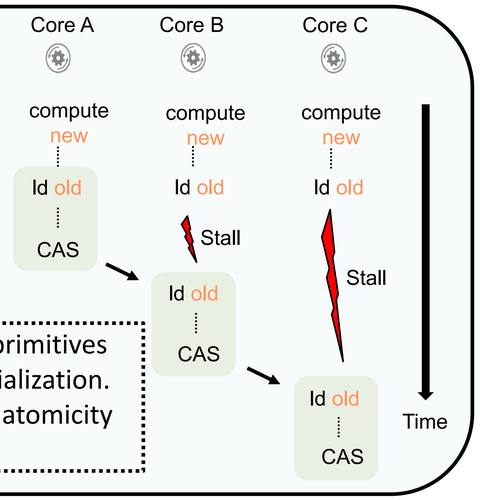
Adam Morrison
Technion – Israel Institute of Technology

Josep Torrellas
University of Illinois, Urbana Champaign

1. MOTIVATION: SERIALIZATION

Adding node to shared stack

```
void push () {
  Node *new_top = malloc();
  while(true) {
Atomic{
    old_top = stack;
    new_top->next = old_top;
    if(CAS(&stack, old_top, new_top))
      return; }}
```



Multi-threaded programs using lock-free primitives (e.g. CAS) suffer from CAS failures and serialization. The former is solved by using load-to-CAS atomicity [1]. CASPAR tackles serialization.

2. INSIGHT

```
void push () {
  Node *new_top = malloc();
  while(true) {
Atomic{
    old_top = stack;
    new_top->next = old_top;
    if(CAS(&stack, old_top, new_top))
      return; }}
```

new_top generated independent of old_top

new_top can be passed to the next core in the synchronization queue, which can use it as a speculative response to old_top.

3. CASPAR MODULES

Module I : Hardware Queueing

- Enforces load-to-CAS atomicity; marks loads as *triggering*
- Creates a queue of synchronizing cores at directory

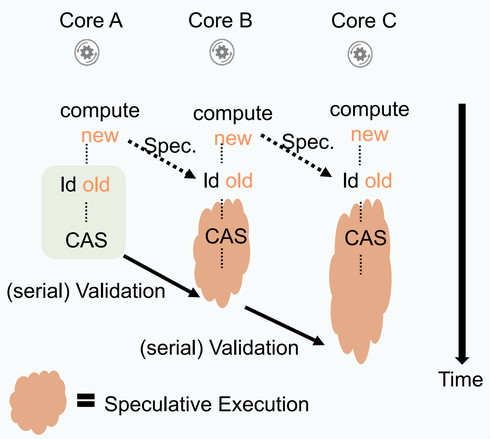
Module II : Eager-Forwarding

- Directory facilitates forwarding of values between cores
- Cores load forwarded values and go speculative
- Sequential passing of full cache line for validation

Module III : Group Commit

- Directory orchestrates parallel validation of cores using two-phase commit protocol

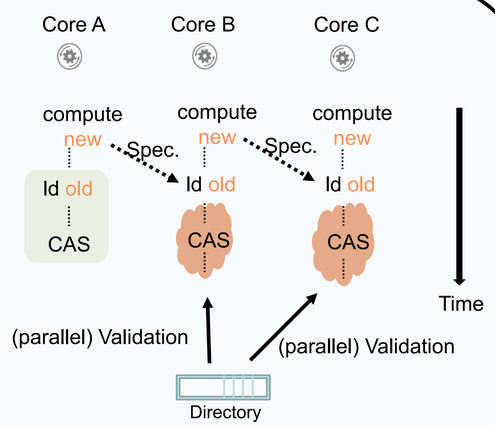
4. EAGER-FORWARDING



A typical sequence of events at a core is -

1. Take a checkpoint and start speculative execution once the *triggering* load (old_top) reaches the ROB head
2. Compute new data and forward it to directory, eagerly
3. If directory provides a value from predecessor, use it speculatively for old
4. Once the full line arrives, validate and terminate speculation

5. GROUP COMMIT

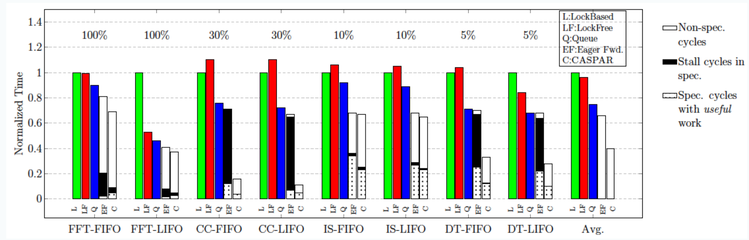


This design uses the directory to achieve simultaneous validation of multiple cores in speculation. This reduces stalls and speculation aborts.

Two Phase Commit Protocol –

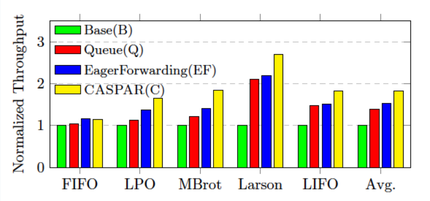
1. Directory sends 'prepare to commit' to cores
2. Cores reply with 'ack' or 'nack'
3. Directory sends 'commit' or 'resume' to cores

6. EVALUATION



Sniper simulator
64 cores, OOO
3 Galois applications
1 BOTS application
5 kernels

CASPAR improves throughput of kernels by 32% on average, and reduces execution time of the *sections considered* in lock-free versions of applications by 47% on average, compared to existing proposals with hardware queues.



7. LIMITATIONS/ FUTURE WORK

- Is effective when new is independent of old
- Dynamic re-ordering of the queued-up cores by the directory can help uncover further parallelism
- Can be used to break conflict serialization in TM designs

8. ALSO IN PAPER

- Information on proposed hardware changes
- Detailed working of Eager Forwarding and 2PC protocol
- Memory consistency considerations
- Supporting other primitives (e.g. LL/SC)

[1] Goodman et al. QOSB/QOLB