A Comparison of the Use of Virtual Versus Physical Snapshots for Supporting Update-Intensive Workloads

Darius Šidlauskas
Aalborg University
darius@cs.aau.dk

Christian S. Jensen
Aarhus University
csj@cs.au.dk

Simonas Šaltenis
Aalborg University
simas@cs.aau.dk

DaMoN 2012
Scottsdale, Arizona, USA
May 21, 2012

Motivation (1)

- Parallelism in chip multi-processors continues to increase
- Writing parallel code is difficult
  - Complications of concurrency control
  - Phantom problems
  - Deadlock situations
  - Serialization bottlenecks
  - Especially, for workloads that intermix frequent updates with queries

Motivation (2)

- A snapshot-based approach
  - Write-only snapshot (S_1) for write operations
  - Read-only snapshot (S_2) for read operations
  - Regularly, S_2 is refreshed based on S_1
- Advantages:
  - Isolates otherwise conflicting operations
  - Enables atomic (full) data scans
  - Simplifies concurrency control (and its verification)
  - Current technologies permit very frequent snapshotting

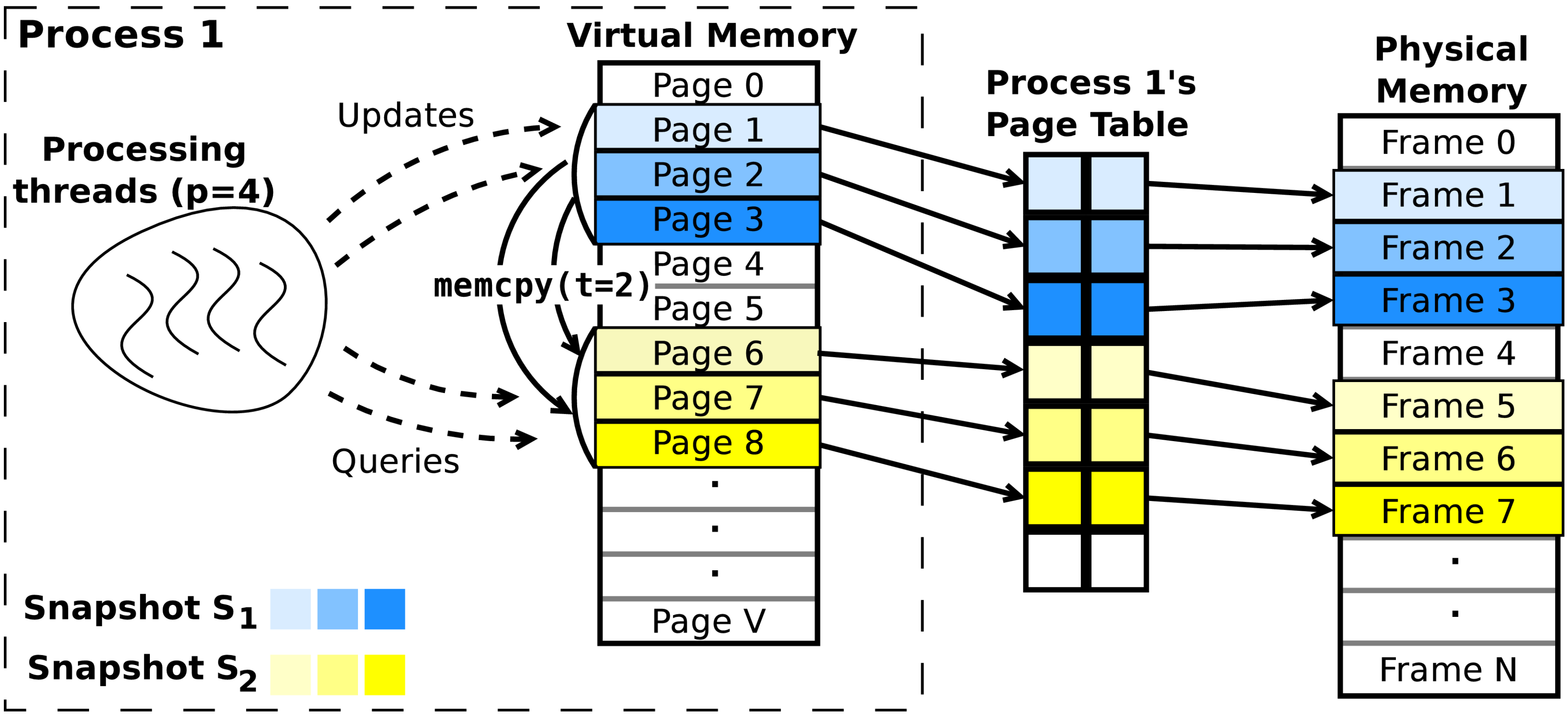
How to create a snapshot?

- Physical snapshots
  - `memcpy`-based
  - Full snapshots (eager copying)
  - Built-in hardware support: prefetching
  - TwinGrid (*SSTD '11*)
- Virtual Snapshots
  - `fork`-based
  - Incremental snapshots (copy-on-write)
  - Built-in hardware support: MMU, TLB
  - HyPer (*ICDE '11*)

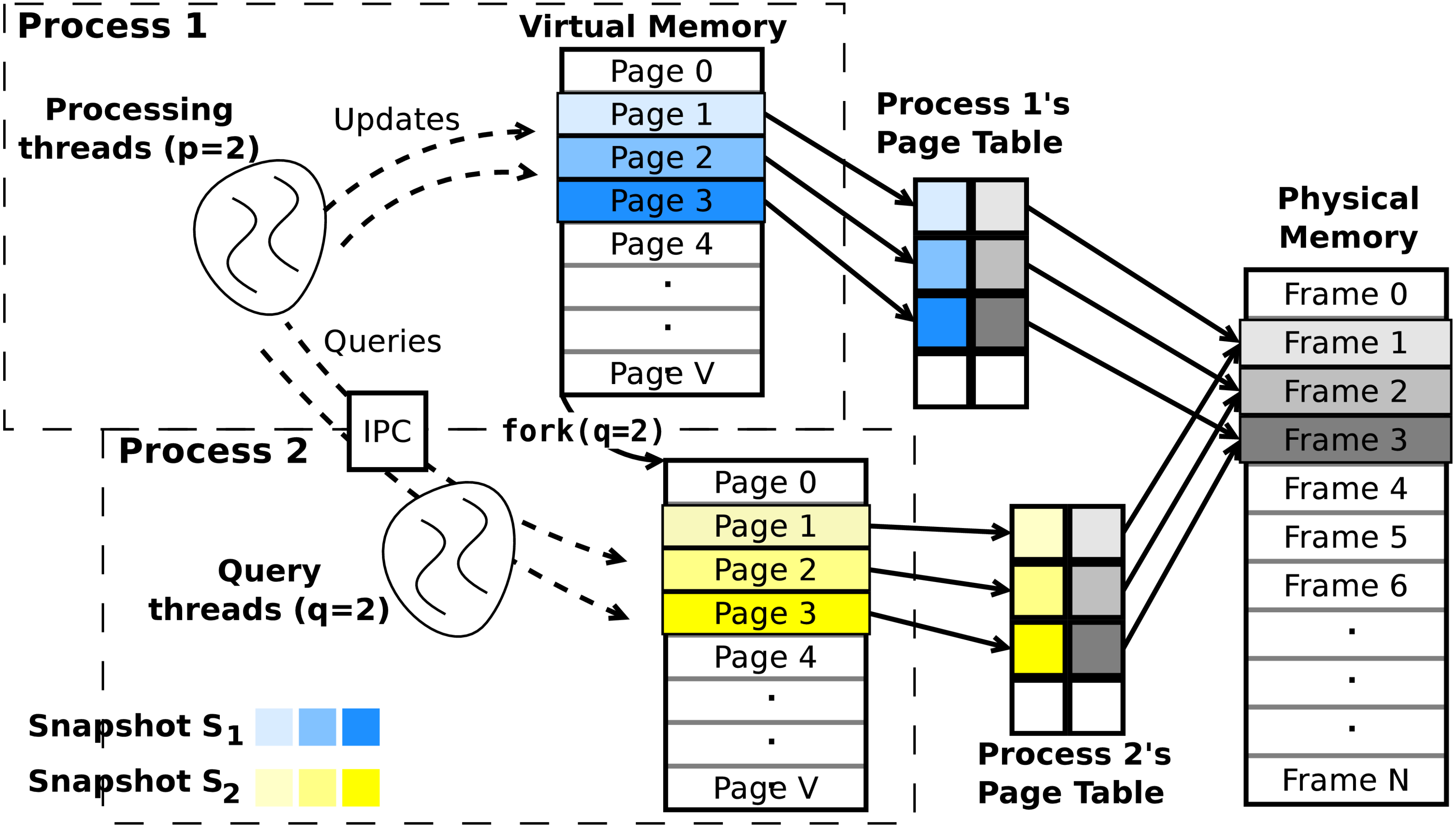
Setting

- Two snapshots
  - Each is an array of N items of fixed size (64 bits)
  - S_1 is for updates (atomic 64-bit writes)
  - S_2 is for queries (range scans)
- Snapshotting frequency (F_s)
  - Expressed as a number of updates
  - E.g., snapshotting occurs every 1000 updates
  - Controls the freshness of S_2

Physical Snapshotting



Virtual Snapshotting

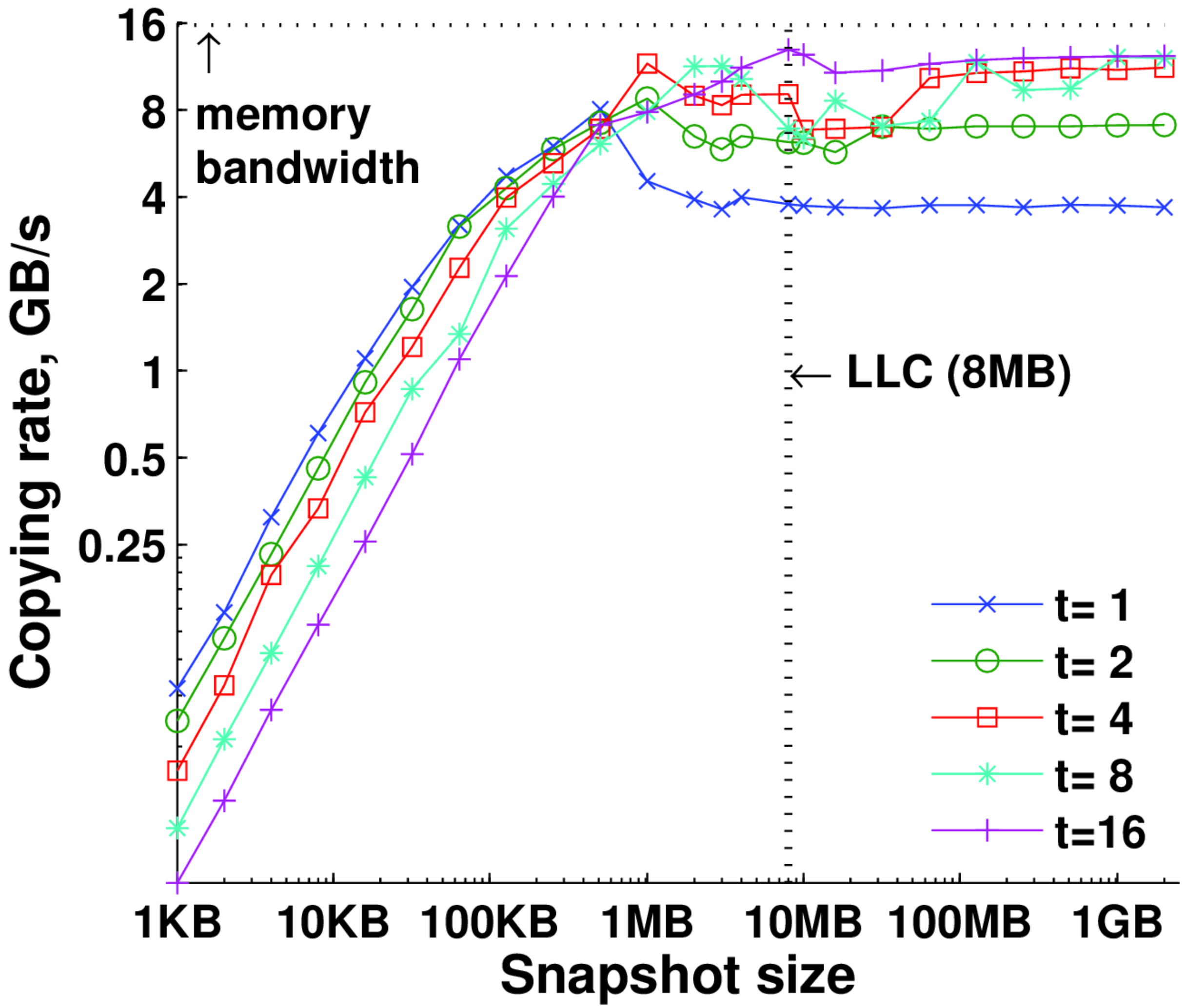


Performance Study

- Considers
  - Time per snapshot creation
  - Snapshot updating under different update distributions
  - Cost to query a snapshot
  - Memory consumption
- Includes four diverse multi-core platforms
  - Lean-camp and fat-camp CMP designs
  - Single- and multi-socket configurations
  - Two different Unix flavors

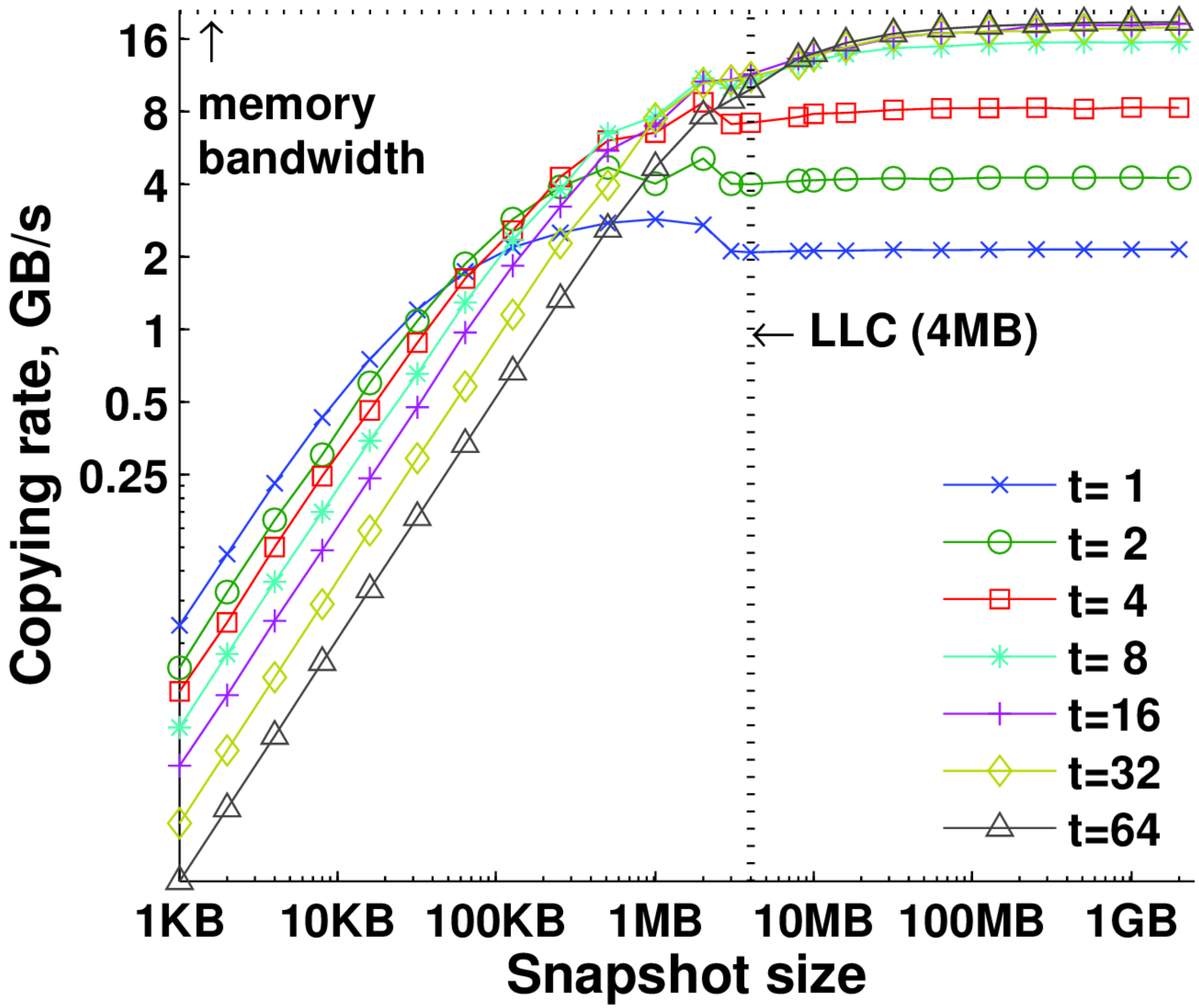
`memcpy` Performance

- Dual 4-core Intel Xeon X5550 2.67GHz with 16 thread contexts

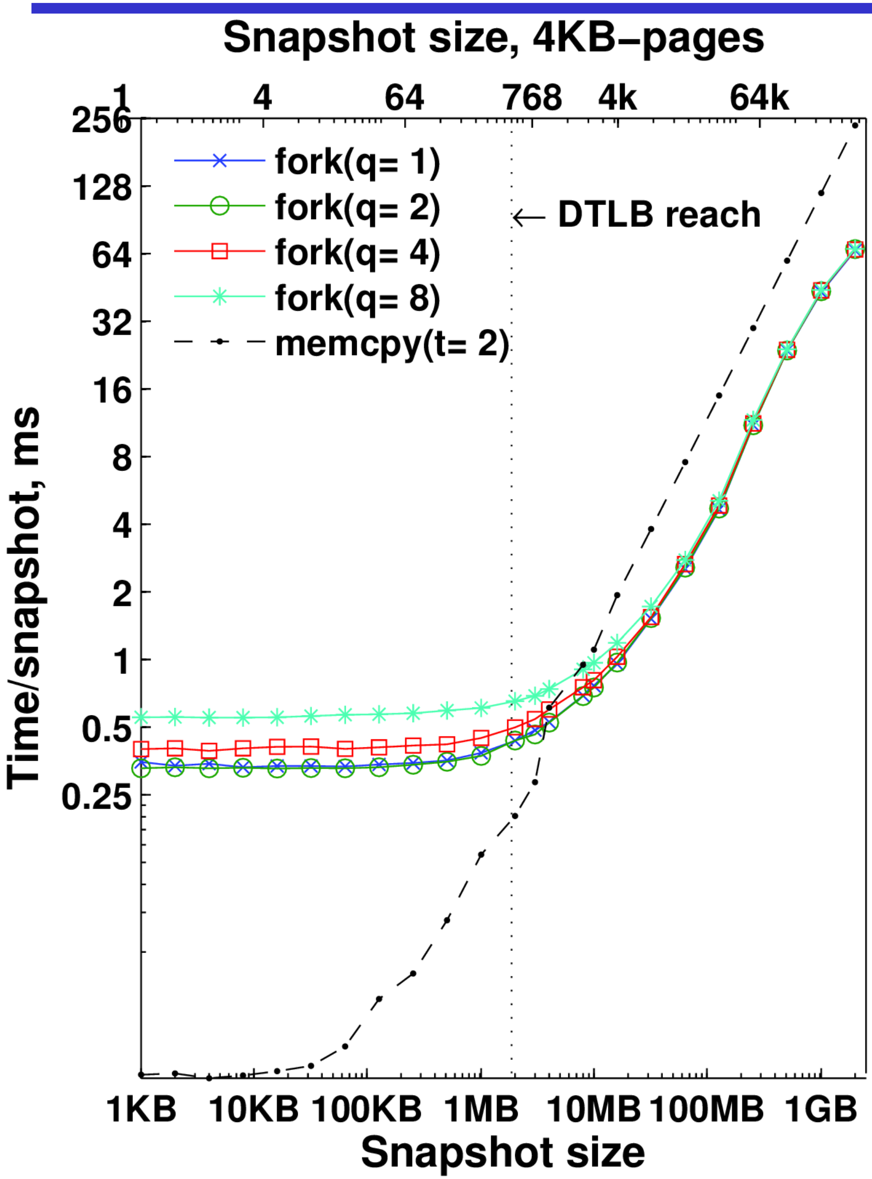


`memcpy` Performance

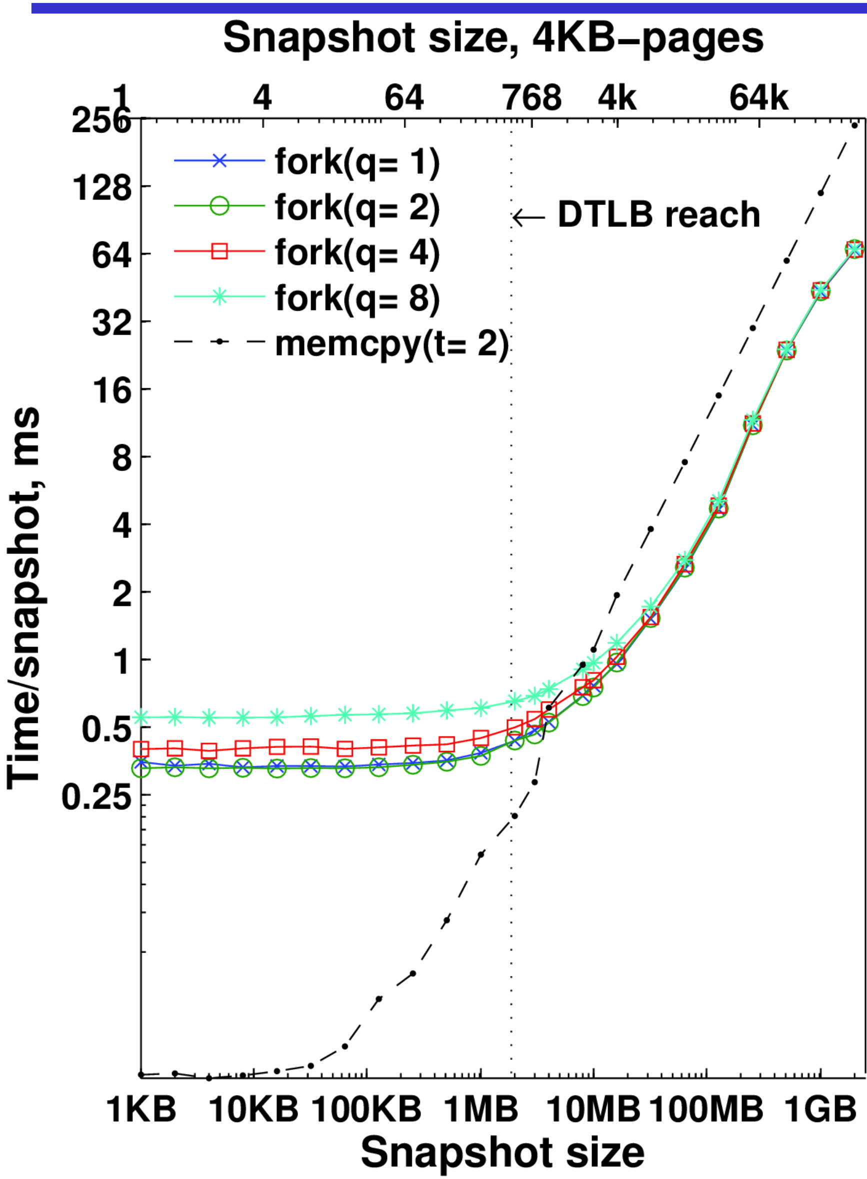
- 8-core UltraSPARC-T2 1.2GHz (Niagara 2) with 64 threads

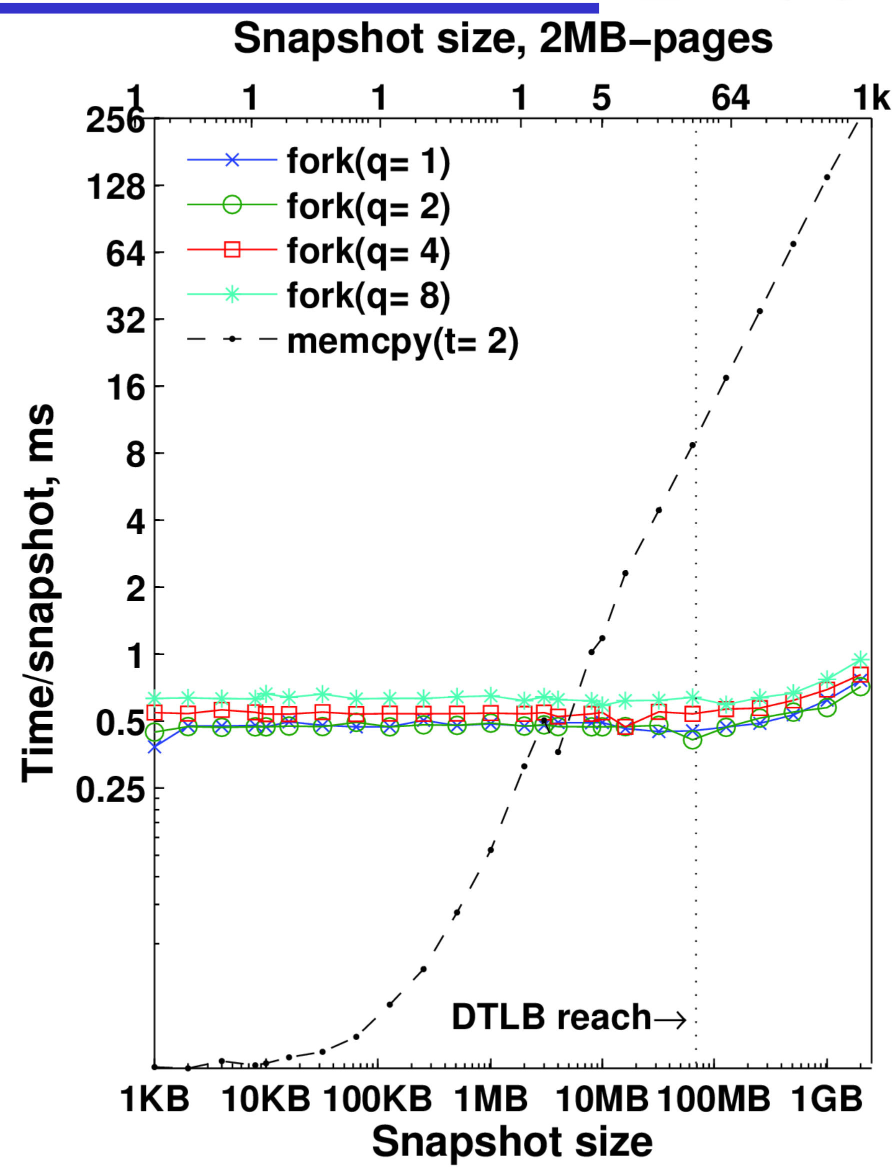


fork VS memcpy



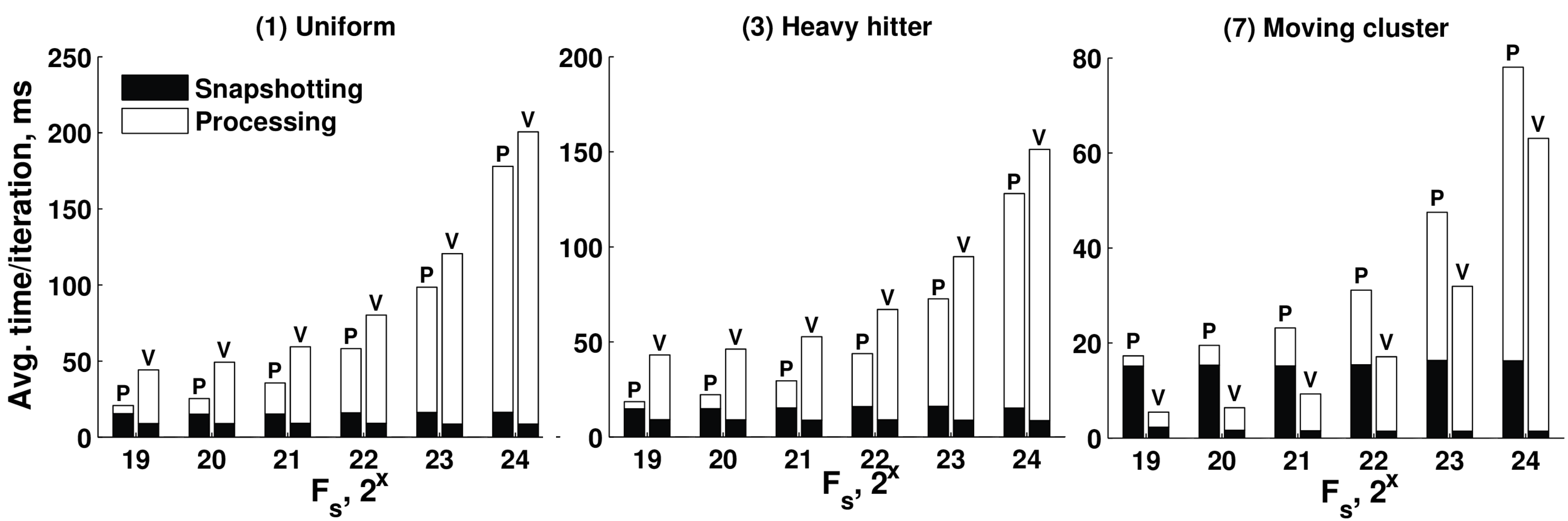
fork VS memcpy





Update Performance

- Snapshot size: 2^{24} items or 128MB
- Y-axis: msec per iteration (snapshotting + update processing)
- X-axis: snapshotting frequency



Conclusions

- Virtual Snapshotting
  - Implicitly supports linked data structures (pointers and non-continuous snapshots)
  - Benefits greatly from huge pages
  - Smaller memory footprint
- Physical Snapshotting
  - Easier to implement query support
  - Creation of large snapshots is very competitive
  - Efficient on all considered platform
  - Surprisingly efficient under many update distributions