ReVivel/O

Josep Torrellas

CS533

Contribution

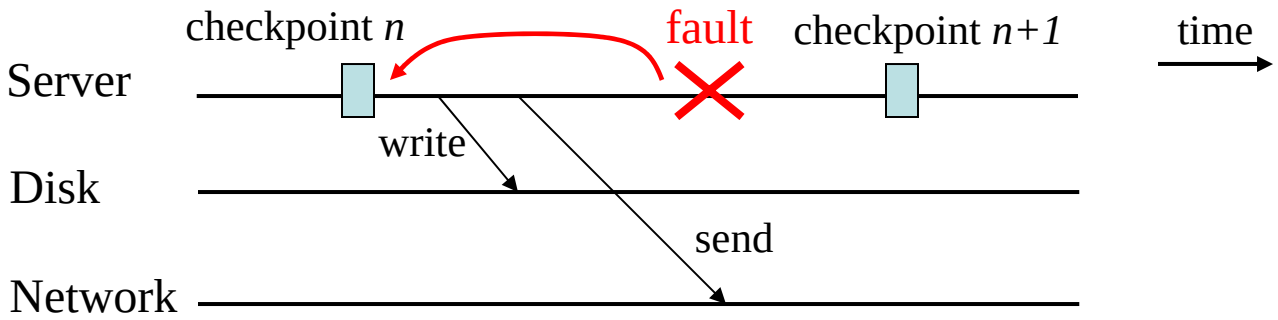
- Deal with I/O in rollback-recovery-based shared memory multiprocessor servers
  - Transparent to OS and applications
  - Recovery from a large class of faults
  - Fast recovery
  - Low overhead during fault-free execution
  - Small impact on hardware cost
- Understand the trade-offs related to checkpoint frequency (sensitivity analysis)

Context and Scope

- Frequently checkpointing shared memory multiprocessor servers that checkpoint processor/memory state such as ReVive and SafetyNet
- Disk and network I/O
- Mainly targeted at throughput-oriented workloads (e.g., backend/database servers)

Problem with I/O

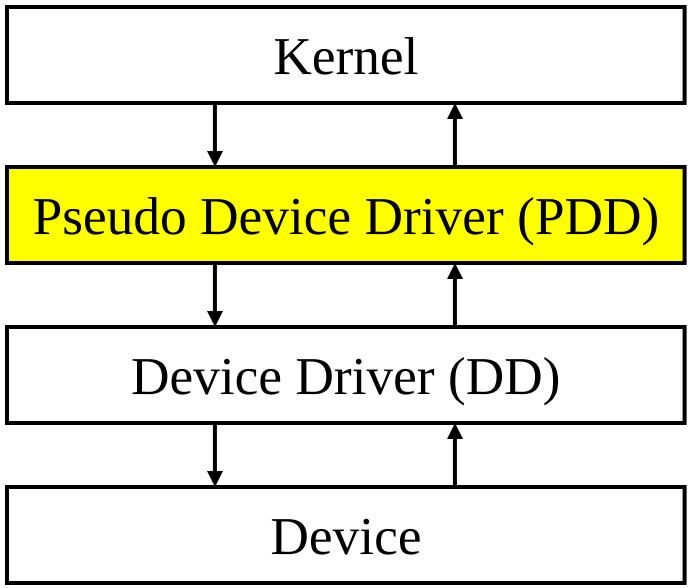
- Output commit problem
  - Output to external world cannot be rolled back
  - An output can be committed only when the system is guaranteed not to roll back to a state prior to the corresponding output request



- Recently proposed transactional memory systems have the same problem
- Simple solution – delay output until after the next checkpoint
  - Faster output commit requires more frequent checkpoint, more frequent checkpoint incurs more overhead

Solution – Pseudo Device Driver

- Pseudo Device Driver (PDD) software layer between the kernel and the device drivers (Masubuchi, *et al.* in FTCS-97)
- No need to modify the kernel or the applications
- PDD delays output operations until next checkpoint to eliminate any inconsistencies after rollback
- Our disk PDD employs buffering or renaming technique to speed up synchronous writes
  - Essential for database workload

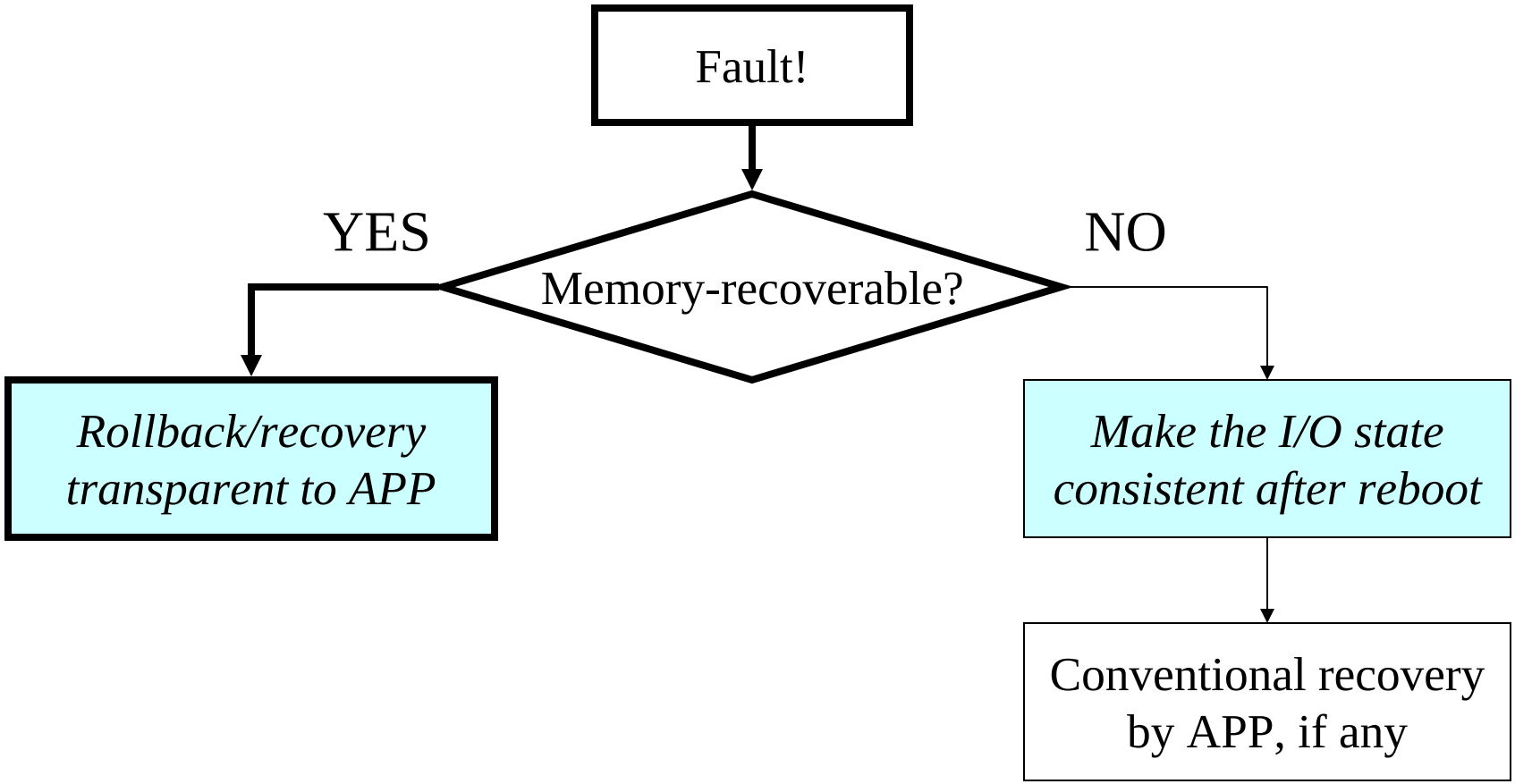


Fault Model

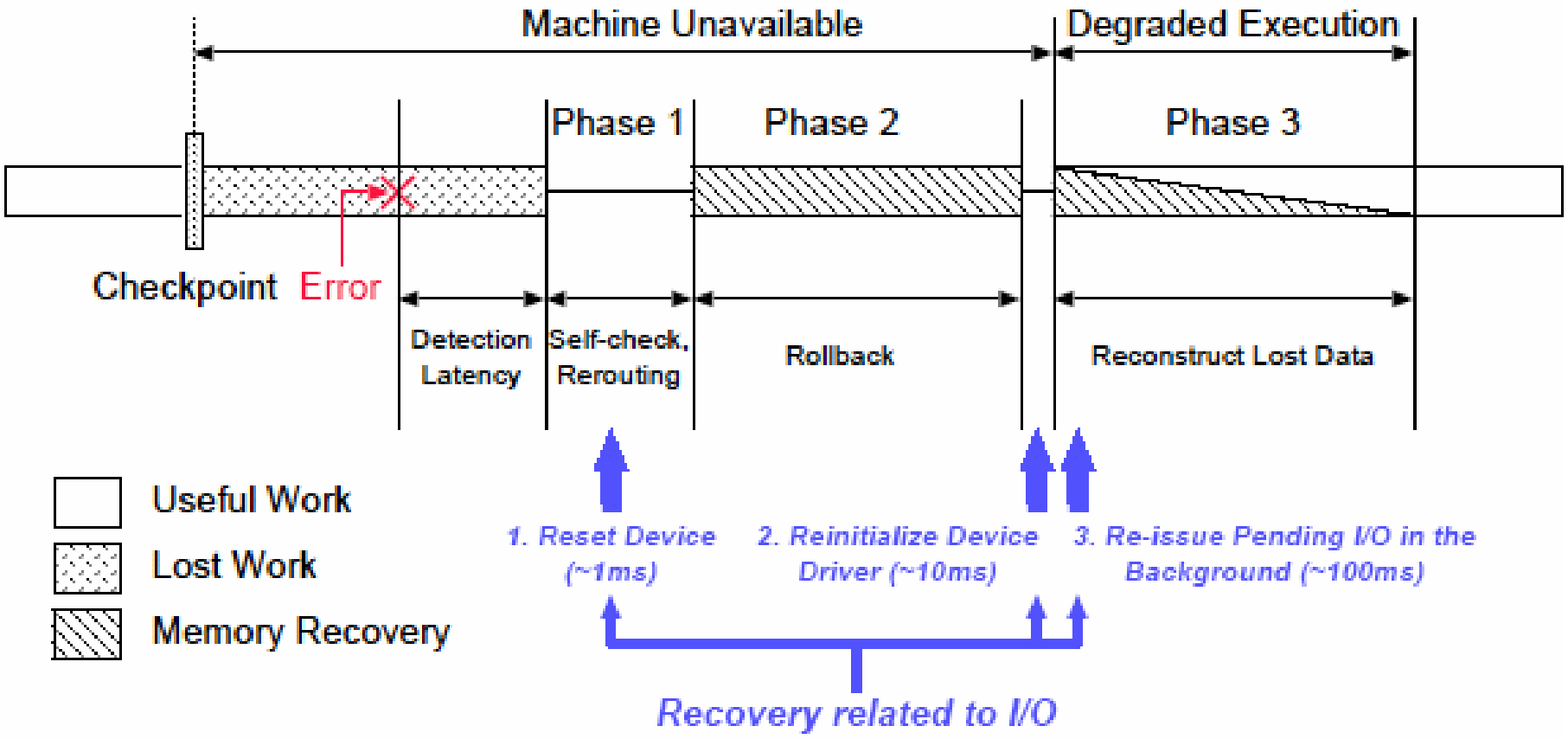
- Memory-recoverable faults
  - The base system (e.g., ReVive) can restore the processor/memory state of the previous checkpoint
  - The PDD mechanism enables recovery from this type of fault transparently to the application

- Non-memory-recoverable faults
  - The system needs to be fixed and rebooted
  - The PDD mechanism still needs to keep the I/O state consistent: otherwise, even the conventional DB recovery does not work

Two Modes of Fault Handling

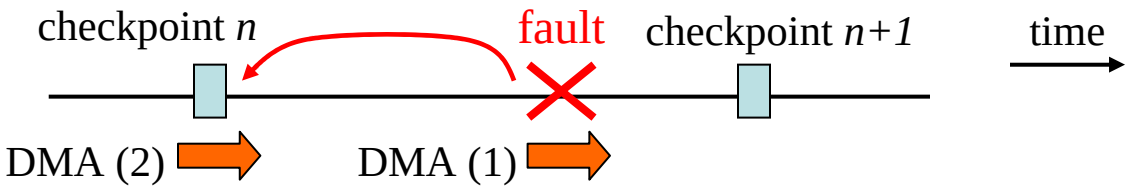


Recovery from Memory-Recoverable Faults



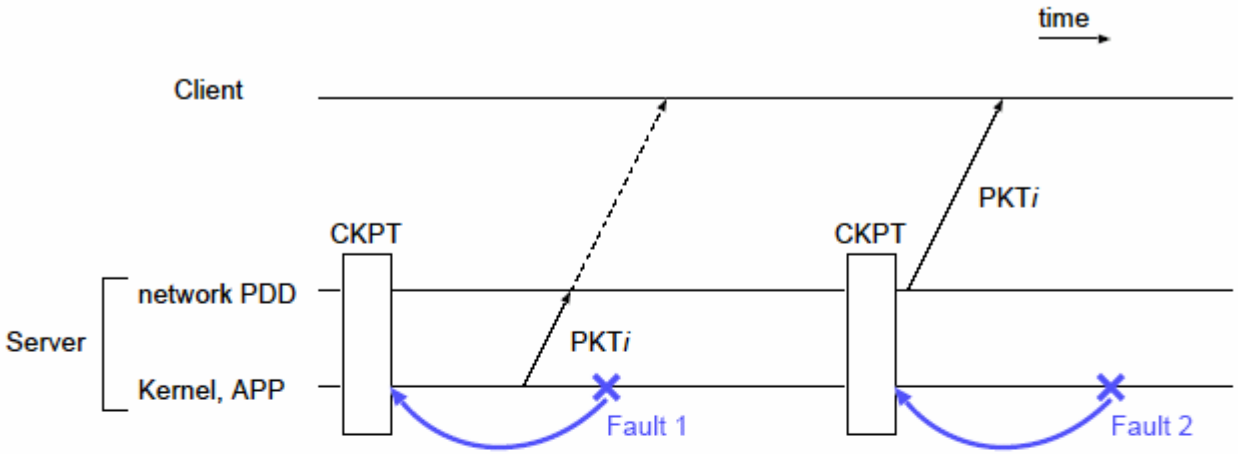
Subtleties about Devices

- Cannot checkpoint/rollback state of devices
- DMA complicates checkpoint/recovery
  - During the rollback, on-going DMA (1) from device to memory may overwrite the memory that is being restored
    → Reset devices before rollback and reinitialize device drivers at recovery time
  - On-going DMA (2) during checkpoint
    → PDD keeps track of pending I/O and reissues them at recovery time (disk I/O is idempotent; for the network I/O…)

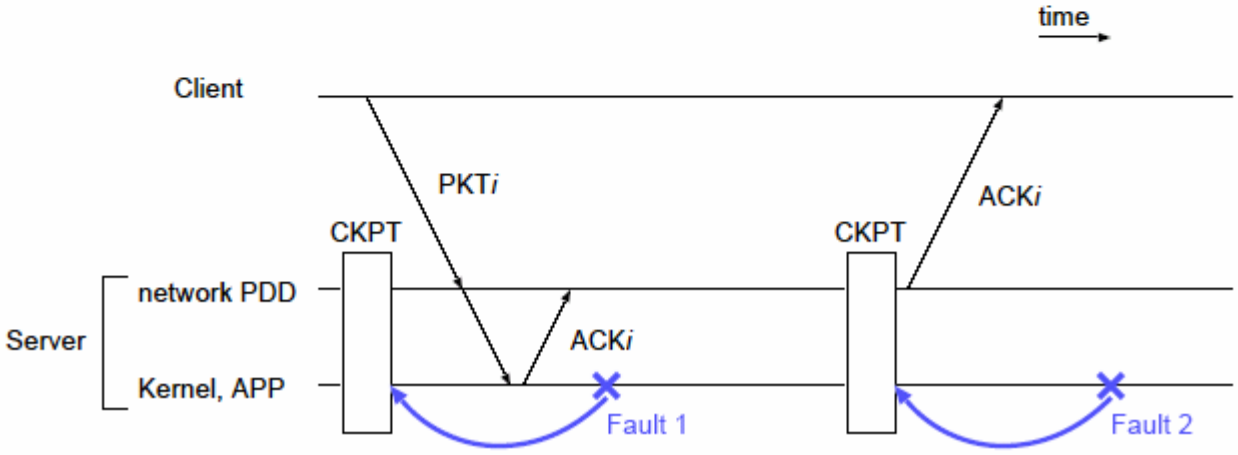


Network PDD and TCP

- Can resend the packets after rollback
  → TCP eliminates duplicate packets



- Can avoid saving inputs for replay
  → In TCP, the sender timeouts and retransmits

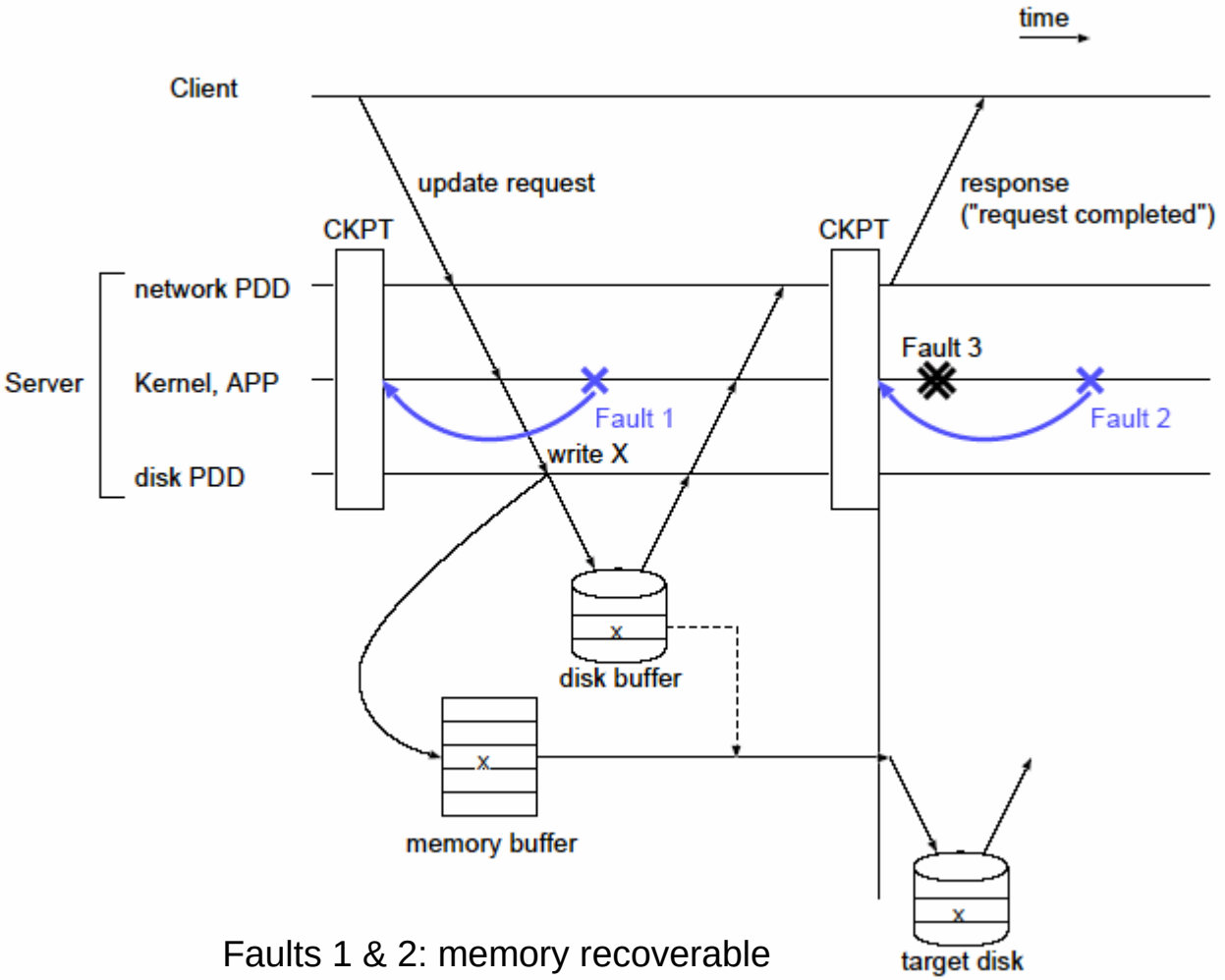


Disk PDD Schemes

× Stall	Stall the write request until next checkpoint Too slow for synchronous writes	1 write
× Logging	Writes data in place Copy old data elsewhere on the disk Too costly	1 read + 2 writes
∨ Buffering	Writes data to disk buffer Commit to the target disk after ckpt Simple	2 writes
∨ Renaming	Writes data to “renamed” location Needs to maintain logical→physical mapping	1 write

Disk PDD – Buffering

- Data written to the memory and the disk buffer
- Disk buffer can be located in the target disk (but not desirable for performance)
- Memory buffer speeds up the commit to the target disk



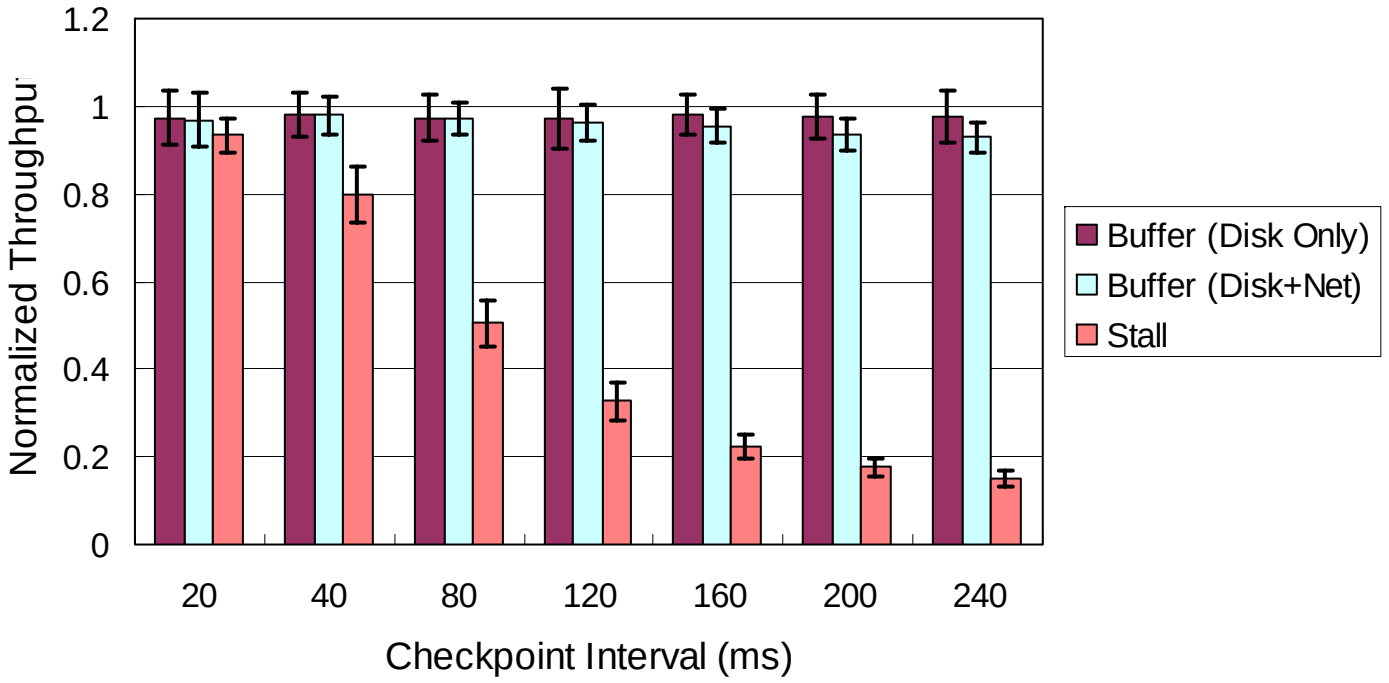
Faults 1 & 2: memory recoverable
Fault 3: non memory recoverable

Experiments

- PDDs are implemented on Linux 2.4 as loadable kernel modules

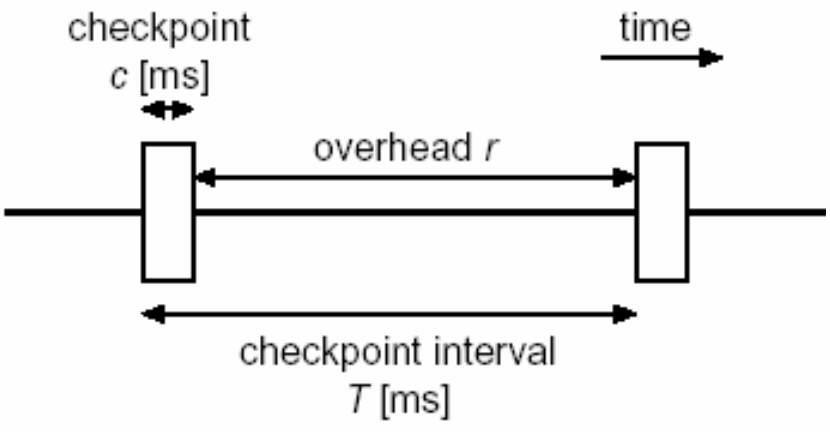
- Two dual 1.5GHz Athlon machines
  - Server (with PDD) for TPC-C and Webstone
  - Clients running normal Linux

TPC-C Throughput



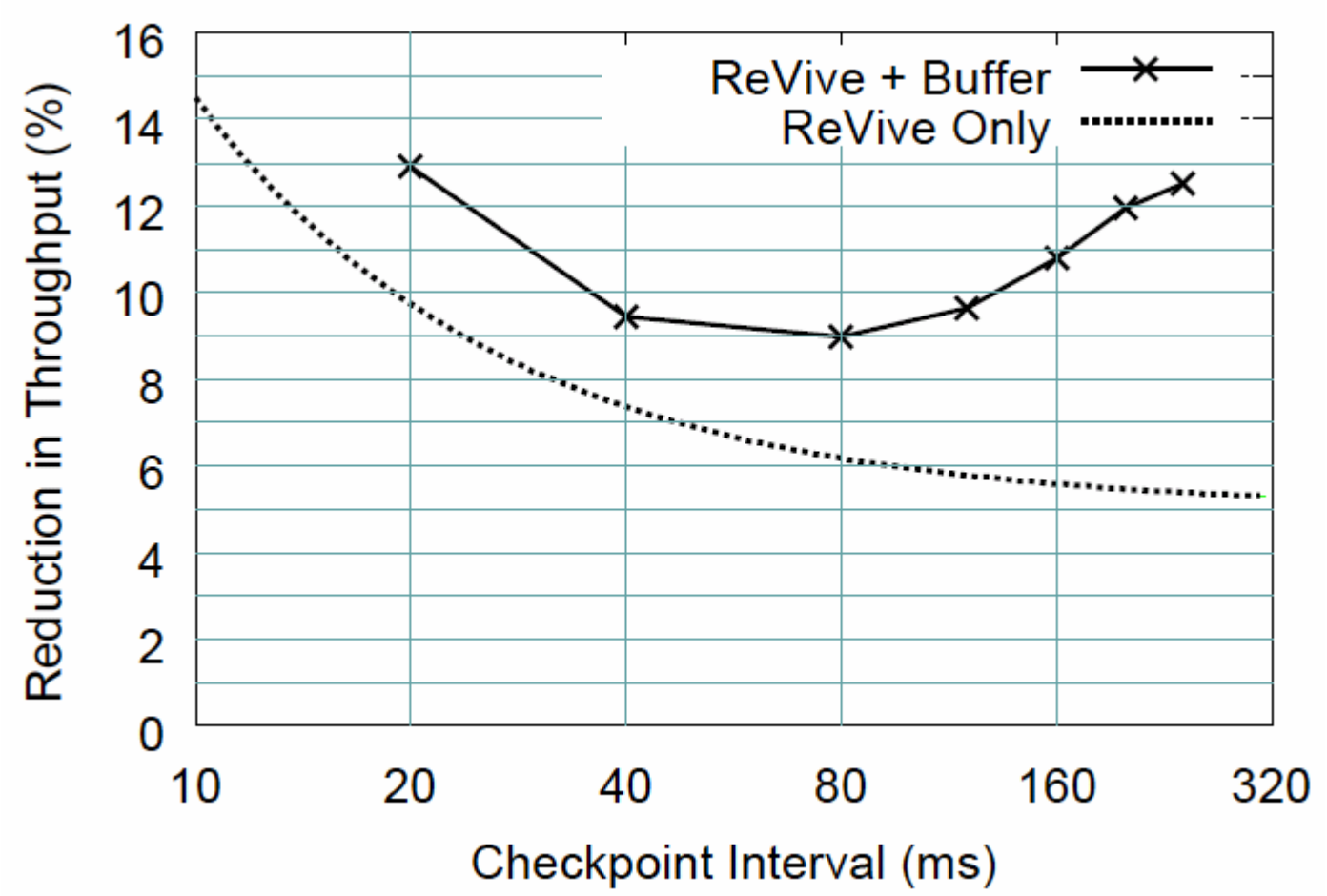
- TPC-C on Oracle9*i* with 32 warehouses and 30 clients
- Stalling scheme (simply delaying writes) incurs too much overhead
- Buffering incurs overhead of only 5% for $T \leq 160$ ms

Integration with ReVive



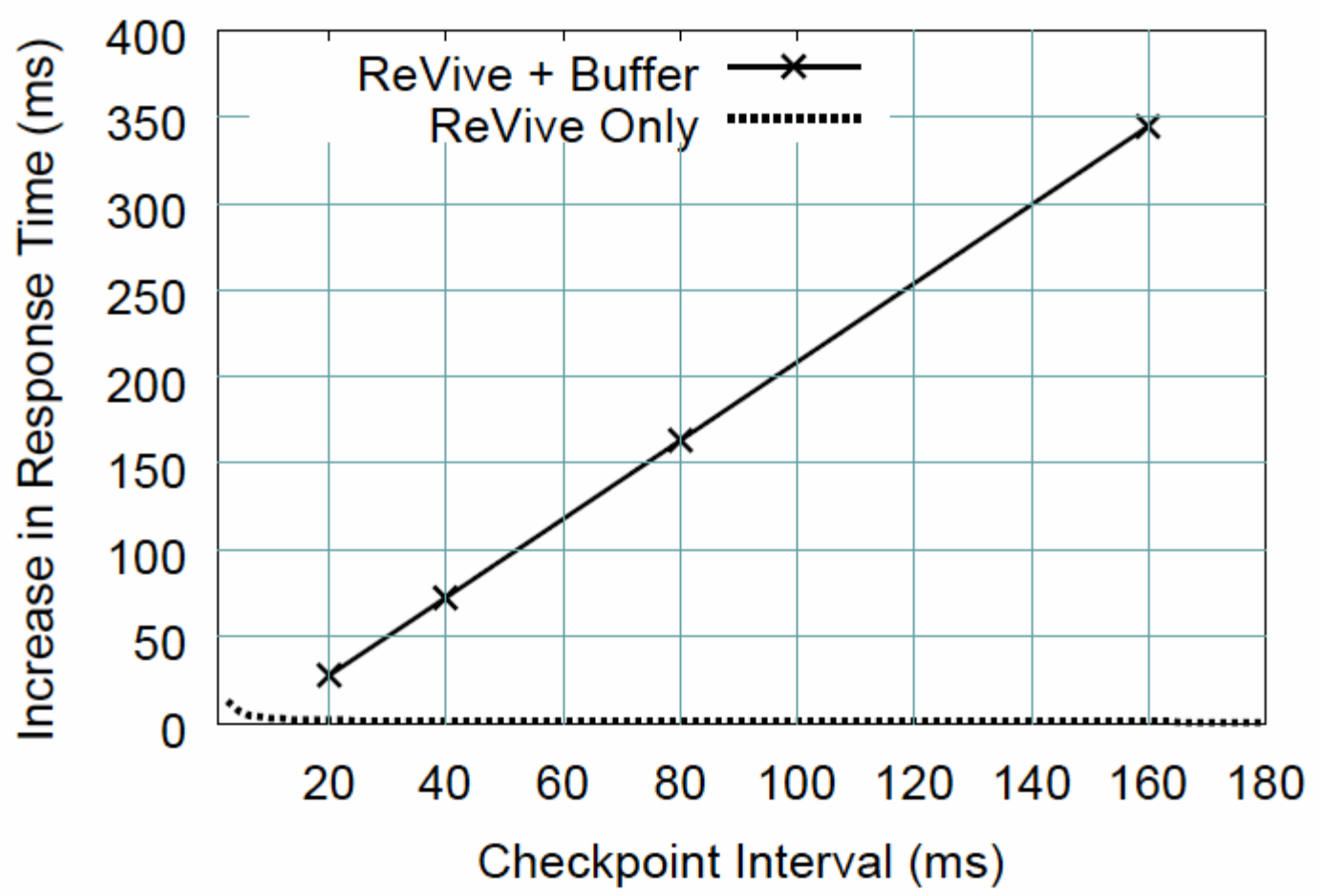
- Simple model of ReVive overhead
  - *c* and *r* are constant (regardless of checkpoint frequency)
  - For instance, *c* = 1 ms, *r* = 5%, and *T* = 100 ms give the overhead of 6%

Integration with ReVive



- Reduction in throughput for TPC-C with 32 warehouses and 30 clients (*c* = 1 ms, *r* = 5%)

Integration with ReVive



- Increase in response time for WebStone with 30 clients (*c* = 1 ms, *r* = 5%)

Space Overhead

- Network PDD
  - Need to increase OS's TCP sliding window size
  - e.g., Gigabit Ethernet, T = 100 ms $\rightarrow$ 12 MB
- Disk PDD (Buffering)
  - PDD needs private memory area for buffering data
  - e.g., 40 MB/s disk, T = 100 ms $\rightarrow$ 4 MB

Summary

- Support of disk/network I/O for memory-checkpointing systems
  - No modifications to the OS/applications
  - Latency of I/O-related recovery is negligible (< 200 ms)
  - Low space overhead
- Throughput-oriented workload
  - Checkpoint interval between 20 ms and 160 ms incurs less than 5% overhead for the transaction throughput
- Latency-sensitive workload
  - Tolerable checkpoint interval should be determined from computer-human interaction point of view in addition to the communication pattern
  - For Webstone, it is fair to say checkpoint interval up to 50 ms is acceptable
- Applicable to transactional memory system