Time Warp Simulation on Multi-core Processors and Clusters

Douglas A. Weber
Masters Thesis Defense
Advisor: Professor Philip A. Wilsey

University of Cincinnati

March 8, 2016

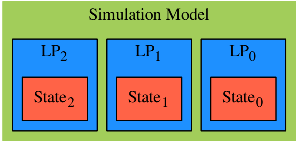
Discrete Event Simulation

- Three main components
  - State variables
  - Simulation clock
  - Pending event set

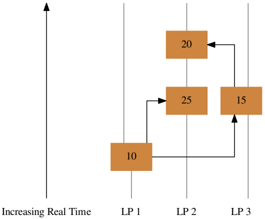
- Unprocessed events stored in pending event set
- Events processed in time stamp order
- Simulation clock and state variables updated only when event occurs

Parallel Discrete Event Simulation

- Model system as a set of Logical Processes (LPs)
- Events exchanged between LPs

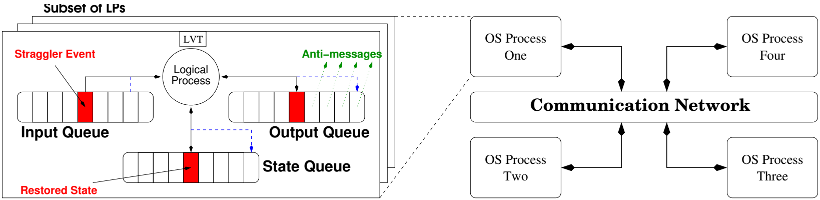


Possible Causality Violations



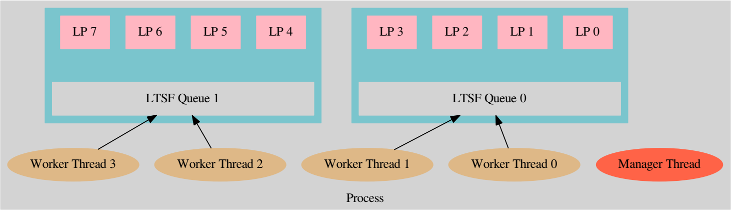
- Events can be received *and* processed out of order
- Two solutions
  - Conservative and Optimistic

Optimistic Mechanism



- Rollback Mechanism
  - State Restoration, Anti-Messages
- Local Virtual Time (LVT) & Global Virtual Time (GVT)
- Fossil Collection

WARPED2 Process



- Worker Threads and Manager Thread
- LTSF Queues/Worker Thread - Contention vs Rollbacks
- LP Partitioning

UNIVERSITY OF Cincinnati

Worker Threads

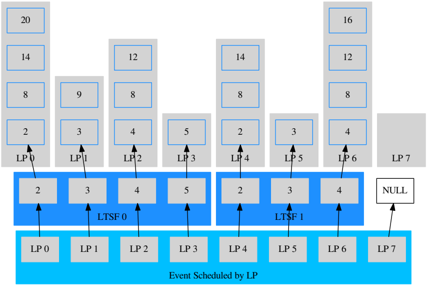
while *termination not detected* **do**
    $e \leftarrow$ `getNextEvent()`
    $lp \leftarrow$ receiver of e

    if $e <$ *last processed event for lp* **then**
        rollback lp

    if e *is an anti-message* **then**
        cancel event with e
        schedule new event for lp
        continue

    process event e
    save state of lp
    send new events

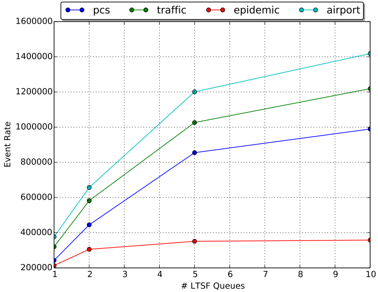
    move e to processed queue
    replace scheduled event for lp

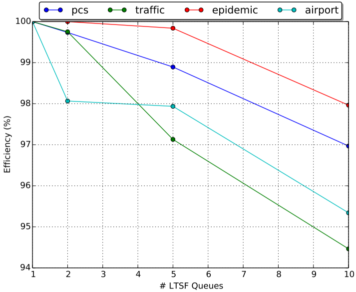


- Unprocessed and Processed Queues
- Scheduling to LTSF
- Rollbacks

Sharing LTSF Queues

- 10 worker threads



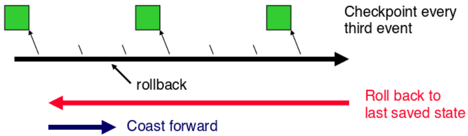


$$EventRate = \frac{CommitedEvents}{Runtime} \quad Efficiency = \frac{CommittedEvents}{ProcessedEvents} * 100\%$$

Reducing contention more important than reducing rollbacks

Periodic State Saving - Overview

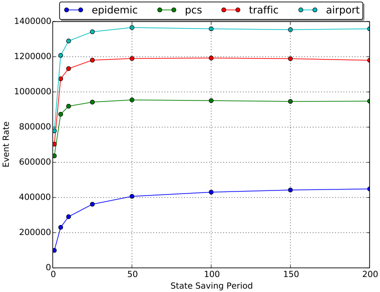
Save state only once every *N* events

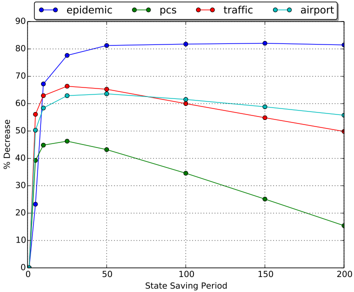


- Not all states available to roll back to
- Must ”Coast Forward” to reproduce state
- Increases time to rollback
- Decrease time to save states and reduce memory footprint

Periodic State Saving - SMP Machine

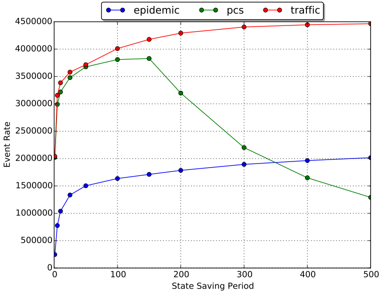
Intel® Xeon® X5675

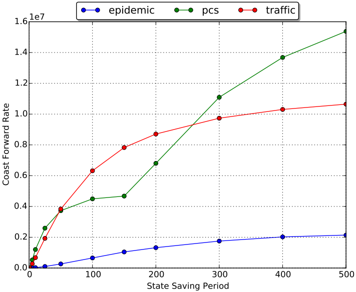




Periodic State Saving - Cluster

8 Nodes - Intel® Xeon® E5410



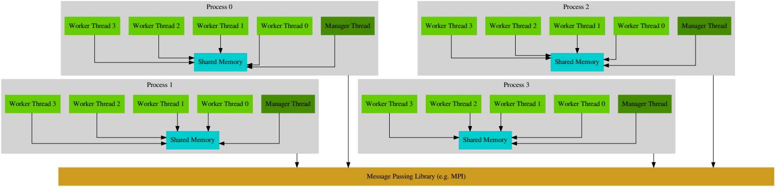


$$\mathit{CoastForwardRate} = \frac{\mathit{CoastForwardEvents}}{\mathit{Rollbacks}} * \mathit{RollbackRate}$$

Communication Model

Shared Memory vs Message Passing

- Sending Events, GVT, Termination



Thread Message Passing Models

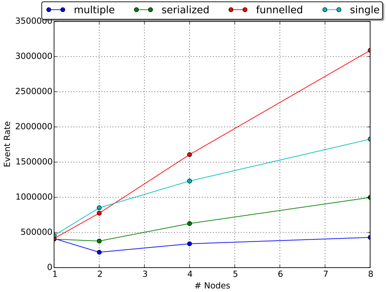
- Single threaded
- Funnelled
- Serialized
- Multiple

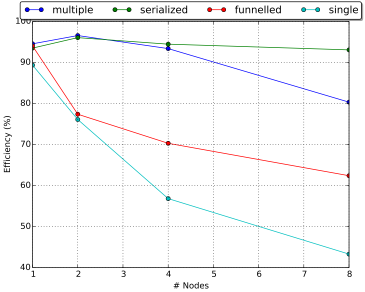
Communication Model - Traffic Model

UNIVERSITY OF Cincinnati

8 Nodes - Intel® Xeon® E5410

- Single - 1 process per processor
- Others - 7 worker threads and manager thread





- Funneled/Single - Minimal Synchronization, High Latency
- Serialized/Multiple - Lots of Synchronization , Low Latency
- Performance will vary with model and partitioning.

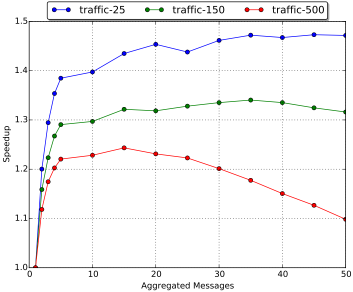
Message Aggregation

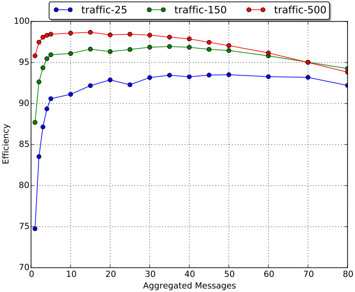
- Wait
  for *N* messages with same destination *process* and send together

Num Msgs 4 bytes	Length 0 (L[0]) 4 bytes	Length 1 (L[1]) 4 bytes	Length 2 (L[2]) 4 bytes	...	Message 0 L[0]	Message 1 L[1]	Message 2 L[2]	...

- Pros
  - Single buffer allocated for multiple messages
  - Latency shared by multiple messages (TCP/IP/Ethernet/...)
  - Events destined for same LP may be sent together
- Cons
  - May increase event latency too high

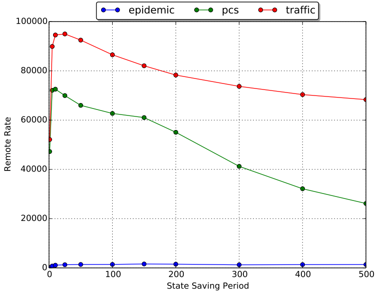
Message Aggregation - Traffic Model

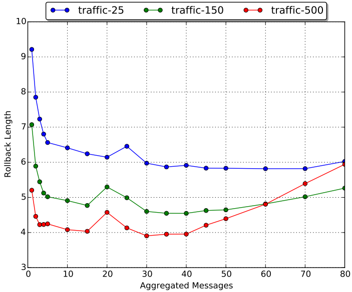




- Best with small number of aggregated events
- Different behavior with different state saving periods

Message Aggregation - More Observations





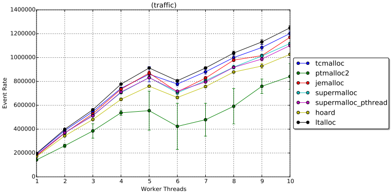
- Faster rate of remote events better
- Can lower rollback length?

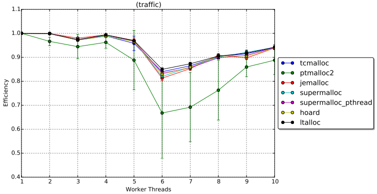
$$AverageRollbackLength = RolledBackEvents/Rollbacks$$

UNIVERSITY OF Cincinnati

Intel® Xeon® X5675

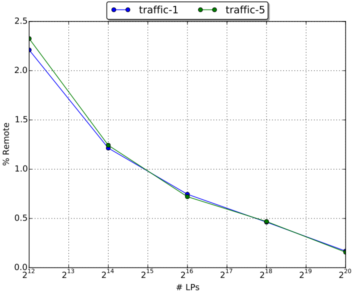
Hyperthreading - 6 Cores, 12 Threads

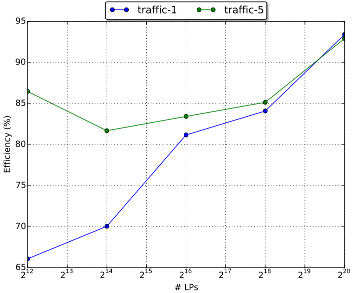




- Imbalance at 6 worker threads
- ptmalloc2 default in GLIBC - broken?

Scaling Simulation Models - Advantages

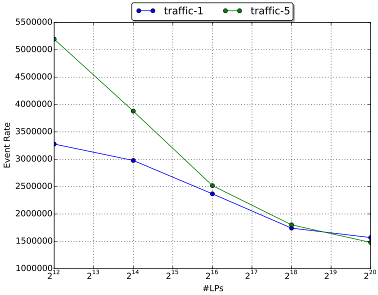


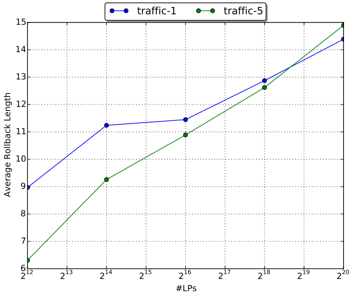


- Coarser Granularity
- Higher Efficiency

Scaling Simulation Models - Disadvantages

UNIVERSITY OF Cincinnati





- More LPs to fossil collect
- Longer Rollbacks

Concusionsi & Future Research

UNIVERSITY OF Cincinnati

- Conclusions
  - Synchronization should always be avoided
    - Interprocess communication - Higher latency over synchronization
    - 1 LTSF queue per worker thread
  - Avoid communication ... If possible
    - If it cannot be avoided, aggregating messages may help
    - Hard to calculate GVT efficiently
    - Memory consumption - May need flow control
  - Alternative multithreaded memory allocators better than GLIBC default
- Future Research
  - Optimistic Fossil Collection
  - Scaling up