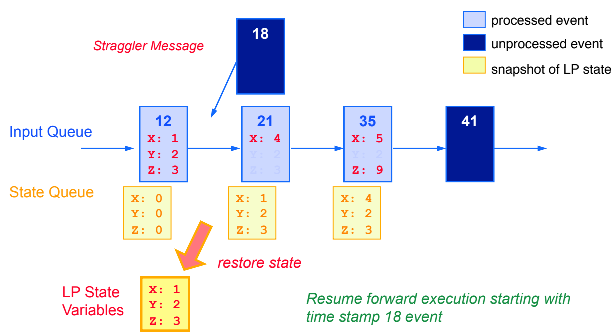
Simulation & Modeling

PDES: Time Warp Mechanism
State Saving and Simultaneous Events

Outline

- **State Saving Techniques**
  - » Copy State Saving
  - » Infrequent State Saving
  - » Incremental State Saving
  - » Reverse Computation
- **Simultaneous Events**

Copy State Save



- **Checkpoint all modifiable state variables of the LP prior to processing each event**
- **Rollback: copy check pointed state to LP state variables**

Copy State Saving

Drawbacks

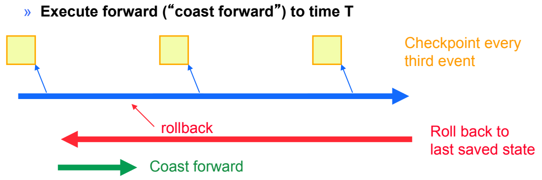
- **Forward execution slowed by checkpointing**
  - » Must state save even if no rollbacks occur
  - » Inefficient if most of the state variables are not modified by each event
- **Consumes large amount of memory**

Copy state saving is only practical for LPs that do not have a large state vector

Largely transparent to the simulation application (only need locations of LP state variables)

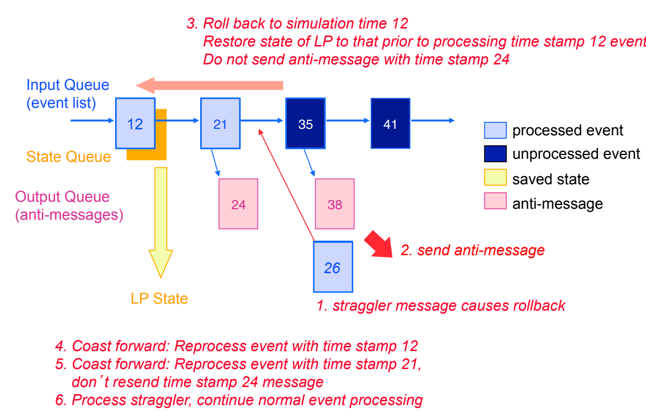
Infrequent State Saving

- **Checkpoint LP periodically, e.g., every Nth event**
- **Rollback to time T: May not have saved state at time T**
  - » Roll back to most recent checkpointed state prior to simulation time T
  - » Execute forward ("coast forward") to time T



- **Coast forward phase**
  - » Only needed to recreate state of LP at simulation time T (no antimsg sends)
  - » Coast forward execution identical to the original execution
  - » Must "turn off" message sends during coast forward, or else
    - – rollback to T could cause new messages with time stamp < T, and roll backs to times earlier than T
    - – Could lead to rollbacks earlier than GVT

Infrequent State Saving Example



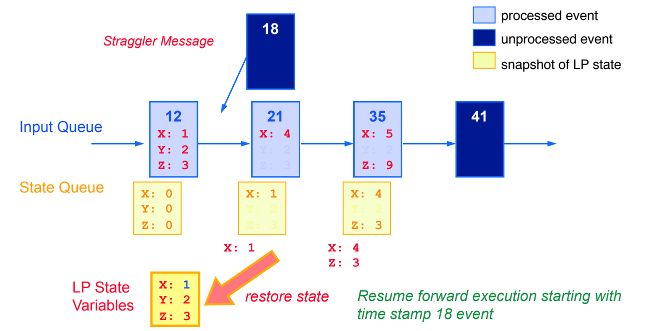
Infrequent State Saving: Pros and Cons

- Reduces time required for state saving
- Reduces memory requirements
- Increases time required to roll back LP
  - » more time to recreate state
- Increases complexity of Time Warp executive
- Largely transparent to the simulation application (only need locations of LP state variables and frequency parameter)

Incremental State Saving

- Only state save variables modified by an event
  - » Generate “change log” with each event indicating previous value of state variable before it was modified
- Rollback
  - » Scan change log in reverse order, restoring old values of state variables

Incremental State Save



- Before modifying a state variable, save current version in state queue
- **Rollback**: Scan state queue from back, restoring old values

Incremental State Saving

- Must log addresses of modified variables in addition to state
- More efficient than copy state save if most state variables are not modified by each event
- Can be used *in addition* to copy state save
- Implementation
  - » Manual insertion of state save primitives
  - » Compiler Support: compiler inserts checkpoint primitives
  - » Executable editing: modify executable to insert checkpoint primitives
  - » Overload assignment operator

Specifying what to Checkpoint

Copy State Saving:

- Transparent to the application program for any frequency (no explicit action need to be taken, once the Time Warp executive now the location of the state save).

Incremental State Saving:

- Manual insertion of state save primitives
- Compiler Support: compiler/pre-processor inserts checkpoint primitives (cost)
- Executable editing: modify executable to insert checkpoint primitives (not portable)
- Overload assignment operator

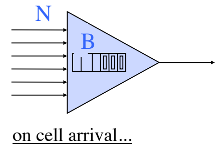
Approaches to Checkpointing

Technique	Advantage	Disadvantage
Manual	Easy to implement (executive)	Tedious an error prone
Compiler/pre-processor	Portable	Cost to develop and maintain
Executable editing	Language independent, source code not needed	Not easily ported to new architectures
Operator Overloading	Easy to implement	Restricted to languages allowing overloading assignment

Reverse Computation

- Rather than state save, recompute prior state
  - » For each event computation, need inverse computation
  - » Instrument forward execution to enable reverse execution
- Advantages
  - » Reduce overhead in forward computation path
  - » Reduce memory requirements
- Disadvantages
  - » Tedious to do by hand, requires automation

RC - Example: ATM Multiplexer



Original

```
if( qlen < B )
   qlen++
   delays[qlen]++
else
   lost++
```

State Size
B+2 words

Forward

```
if( qlen < B )
   b = 1
   qlen++
   delays[qlen]++
else
   b = 0
   lost++
```

Reverse

```
if( b == 1 )
   --delays[qlen]
   --qlen
else
   --lost
```

State Size
1 bit

Outline

- State Saving Techniques
  - » Copy State Saving
  - » Infrequent State Saving
  - » Incremental State Saving
  - » Reverse Computation
- Simultaneous Events

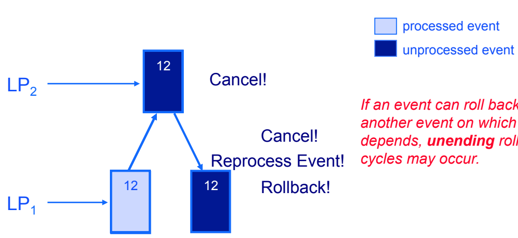
Issues

- Zero lookahead:
  - » An LP has zero lookahead if it can schedule an event with time stamp equal to the current simulation time of the LP
- Simultaneous events:
  - » Events containing the same time stamp; in what order should they be processed?
- Repeatability:
  - » An execution mechanism (e.g., Time Warp) is repeatable if repeated executions produce exactly the same results
  - » Often a requirement
  - » Simplifies debugging

Zero Lookahead & Simultaneous Events

Time Warp: Do simultaneous event cause rollback?

- A possible rule:
  - » If an LP processes an event at simulation time T and then receives a new event with time stamp T, roll back the event that has already been processed.



*If an event can roll back another event on which it depends, **unending** rollback cycles may occur.*

Approach 1

- Prevent Un-Ending Rollback Cycles: *Straggler does not roll back already processed events with the same time stamp.*
  - » What are problem(s) with this approach?

Approach 2

- **Prevent Un-Ending Rollback Cycles:** ***Disallow stragglers rolling back its scheduling dependent events (or indirect scheduling depended events).***

Wide Virtual Time (WVT)

Approach

- **Application uses time value field to indicate "time when the event occurs"**
- **Tie breaking field used to order simultaneous events (events with same time value)**

Time stamp

time value	tie breaking fields

- **Tie breaking field can be viewed as low precision bits of time stamp**
- **Question: How or what should the bits represent?**

An Approach Using WVT

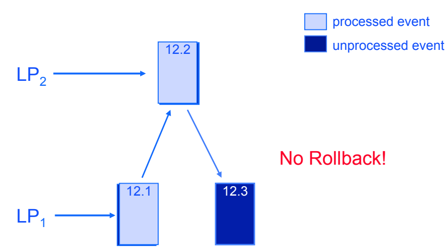
Time stamp:

time value	age

- **Avoid rollback cycles**
  - » **Age field to order scheduling dependent lookahead events**
  - » **Non-zero lookahead events: Age = 1**
  - » **Zero lookahead events: Age = Current Age + 1**

WVT Example

Avoid rollback cycles despite zero lookahead events (using age)



- **Question: Can there be two or more events containing the same time stamp and age scheduled by the same LP? Why or why not?**

An Approach Using WVT

Time stamp:

time value	priority	age	LP ID	Seq #

- **Application specific ordering of events**
  - » **Application specific priority field**
  - » **Constraint on zero lookahead events**
- **Avoid rollback cycles**
  - » **Age field to order dependent lookahead events**
  - » **Non-zero lookahead events: Age = 1**
  - » **Zero lookahead events: Age = Current Age + 1**
- **Repeatable execution**
  - » **ID field identifying LP that scheduled the event**
  - » **Sequence number indicating # of events scheduled**

Summary

- Copy State Saving
  - » Efficient if LP state small
  - » Can be made transparent to application
- Infrequent state saving
  - » Must turn off message sending during coast forward
  - » Reduced memory requirements
  - » less time for state saving
  - » Increased rollback cost
- Incremental State Saving
  - » Preferred approach if large state vectors
  - » Means to simplify usage required

Summary (cont)

- Reverse computation
  - » Efficient, requires automation
- Zero lookahead and simultaneous events
  - » Can lead to unending rollbacks
  - » Wide Virtual Time provides one solution