CSCI 4210/6210
Parallel & Distributed Simulation

PDES: Time Warp Mechanism

Computing Global Virtual Time

Outline

- **GVT Computations: Introduction**
  - » Synchronous vs. Asynchronous
  - » GVT vs. LBTS
- **Computing Global Virtual Time**
  - » Transient Message Problem
  - » Simultaneous Reporting Problem
- **Samadi Algorithm**
  - » Message Acknowledgements
  - » Marked Acknowledgment Messages

Global Virtual Time

- **Problems:**
  - » **Need to Fossil Collect:**
    - – The Time Warp algorithm consumes more and more memory throughout the execution via the creation of new events.
    - – Need to reclaim memory used for
      - processed events,
      - anti-messages, and the
      - state history that is no longer needed.
  - » **Need a mechanism for operations that cannot be rolled back, e.g. I/O cannot be un-done.**
- **Observations:**
  - » TWLPs only roll back as a result of receiving a message.
  - » Positive messages can only be **created** by an **unprocessed** or partially processed message.

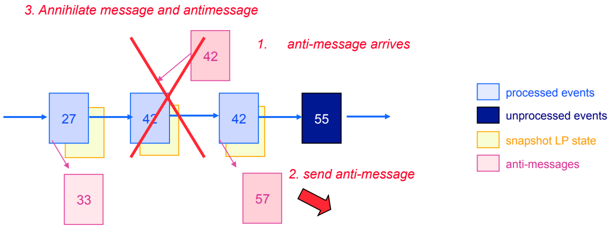
Global Virtual Time

*GVT(t): minimum time stamp among **all** unprocessed or partially processed messages at wallclock time t.*

- **Computing GVT trivial if an instantaneous snapshot of the computation could be obtained: compute minimum time stamp among:**
  - » Unprocessed events & anti-messages within each LP
  - » Transient messages (messages sent before time t that are received after time t)

- **Memory associated with events with a TS equal to GVT cannot be reclaimed because GVT could be equal to the TS of an anti-message that has not been processed.**
  - » Such an anti-message could require one to roll back events with time stamp exactly equal to GVT.

GVT = unprocessed anti-message



Events with time stamps equal to GVT is needed:

- » GVT is 42, and
- » There are **two processed** events with TS 42.
- » In the first the TWLP processed is canceled by an anti-message with time stamp equal to GVT.

Global Virtual Time

*GVT(t): minimum time stamp among **all** unprocessed or partially processed messages at wallclock time t.*

- **Computing GVT trivial if an instantaneous snapshot of the computation could be obtained: compute minimum time stamp among:**
  - » Unprocessed events & anti-messages within each LP
  - » Transient messages (messages sent before time t that are received after time t)

- **Synchronous vs. Asynchronous GVT computation**
  - » **Synchronous GVT algorithms: LPs stop processing events once a GVT computation has been detected**
  - » **Asynchronous GVT algorithms: LPs can continue processing events and schedule new events while the GVT computation proceeds "in background"**

GVT vs. LBTS

GVT algorithms can be used to compute LBTS and vice versa (assuming a fully connected topology and zero lookahead).

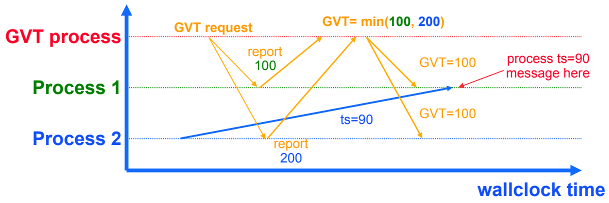
- **Both determine the minimum time stamp of messages (or anti-message) that may later arrive**
  - » **Historically, developed separately**
  - » **Often developed using different assumptions (lookahead, topology, etc.)**
- **Time Warp**
  - » **Latency to compute GVT typically less critical than the latency to compute LBTS (need to compute LBTS often).**
  - » **Asynchronous execution of GVT computation preferred to allow optimistic event processing to continue**

Asynchronous GVT

- **An incorrect GVT algorithm:**
  - » **Controller process: broadcast “compute GVT request”**
  - » **upon receiving the GVT request, each process computes its local minimum and reports it back to the controller**
  - » **Controller computes global minimum, broadcast to others**
- **Difficulties:**
  - » ***transient message problem*: messages sent, but not yet received must be considered in computing GVT**
  - » ***simultaneous reporting problem*: different processors report their local minima at different points in wallclock times, leading to an incorrect GVT value**

The Transient Message Problem

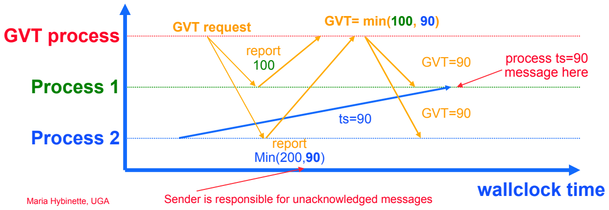
- **Transient message: A message that has been sent, but has not yet been received at its destination**
- **Erroneous values of GVT may be computed if the algorithm does not take into account transient messages**



Transient Messages: A Solution

Approach: Ensure every message is accounted for by at least one processor when GVT is being computed:

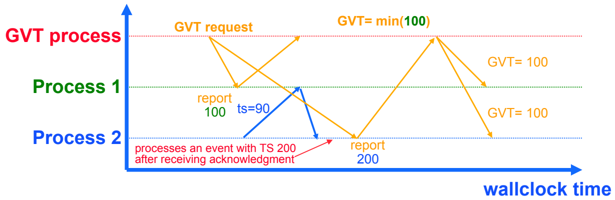
- **Send an acknowledgement message for each message.**
- **Sender reports minimum of any unacknowledged messages.**
- **Receiver takes responsibility as it receives message.**



Sender is responsible for unacknowledged messages

Simultaneous Reporting Problem

Erroneous values of GVT may be computed when processes receive GVT request at different point in time.

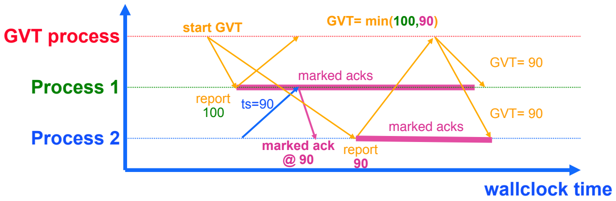


- **Process 1 doesn’t account for time stamp 90 message**
- **Process 2 assumes process 1 will account for the message**
- **Do message acknowledgements solve this problem?**
  - » **No, at least not by themselves**
  - » **Solution: Mark acks that are sent after local min has been reported**

Samadi’s Algorithm

Approach: Send an ack for each event & anti-message received, mark acks after the processor has reported its local minimum

- **Controller broadcast “start GVT” message**
- **Each processor reports minimum time stamp among among (1) local messages, (2) unacknowledged sent messages, (3) marked acks that were received**
- **subsequent acks sent by process are marked until new GVT is received**
- **controller computes global minimum as GVT value, broadcasts new GVT.**



Summary

- Global Virtual Time
  - Similar to lower bound on time stamp (LBTS)
    - Time Warp: GVT usually not as time critical as LBTS
    - Asynchronous GVT computation highly desirable to avoid unnecessary blocking
- Samadi Algorithm
  - Transient message problem: Message acknowledgements
  - Simultaneous reporting problem: Mark acknowledgements sent after reporting local minimum
  - Requires acknowledgements on event messages