Global Virtual Time (GVT) e Approfondimenti sul Time Warp

Gabriele D'Angelo
gda@cs.unibo.it
http://www.cs.unibo.it/~gdangelo

Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna

Sommario

- Global Virtual Time (GVT)
  - Definizione del problema
  - Esempio
- Calcolo asincrono del GVT
- Messaggi transienti
- Simultaneous reporting problem
- Algoritmo di Samadi
- Approfondimenti sul Time Warp

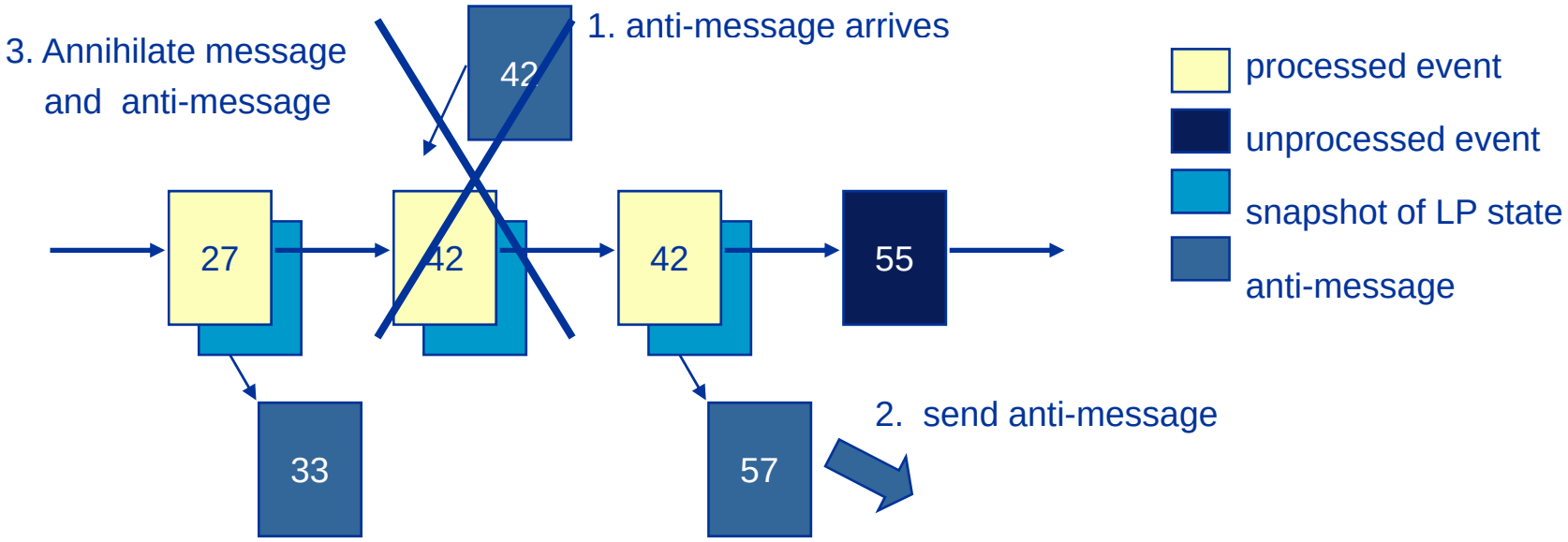
Global Virtual Time (GVT)

- Problemi:
  - Necessità di **fossil collection**: l'algoritmo di sincronizzazione Time Warp *consuma sempre più memoria con il procedere della simulazione* e la creazione di nuovi eventi. È necessario un meccanismo per liberare la memoria utilizzata per gli eventi processati, anti-messaggi e stati del sistema che non sono più necessari
  - Necessità di un meccanismo per le operazioni delle quali non è possibile effettuare un roll-back (ad esempio le operazioni di Input/Output)
- Osservazioni:
  - I LP effettuano roll-back **solo a seguito** della ricezione di messaggi
  - Messaggi "positivi" possono essere generati solamente a seguito di messaggi non processati o parzialmente processati

Global Virtual Time (GVT)

- GVT: Minimo time-stamp (TS) tra **tutti** i messaggi non processati o parzialmente processati, ad un dato Wall-Clock-Time *t*
- Il calcolo del GVT è banale qualora sia abbia a disposizione uno *snapshot istantaneo del sistema*. Calcolare il minimo TS tra:
  - Eventi non processati e anti-messaggi all'intero di ogni LP
  - Messaggi transienti (ovvero i messaggi spediti prima del tempo *t* e che verranno ricevuti in seguito)
- **La memoria associata agli eventi con TS pari al GVT non può essere liberata**, perché il GVT potrebbe essere uguale al TS di un anti-messaggio che non è ancora stato processato. E quindi l'anti-messaggio potrebbe provocare il roll-back di eventi con TS esattamente uguale al GVT

Esempio



È necessario mantenere gli eventi con TS pari al GVT

Nell'esempio assumiamo che il GVT sia pari a 42, ci sono 2 eventi processati con uguale TS. Il primo è cancellato da un anti-messaggio con TS pari al GWT

Global Virtual Time (GVT)

- Abbiamo detto che:
  - GVT: Minimo time-stamp (TS) tra **tutti** i messaggi non processati o parzialmente processati, ad un dato Wall-Clock-Time *t*
  - Il calcolo del GVT è banale qualora sia abbia a disposizione uno snapshot istantaneo del sistema (minimo tra eventi in ogni LP e messaggi transienti)

- Meccanismi di computazione del GVT:
  - Algoritmi **sincroni**: i LP interrompono l'esecuzione degli eventi ogni volta che il calcolo del GVT ha inizio (e solitamente per tutta la durata del calcolo)
  - Algoritmi **asincroni**: i LP proseguono nell'esecuzione e nella schedulazione degli eventi, il calcolo del GVT in questo caso procede in "background"

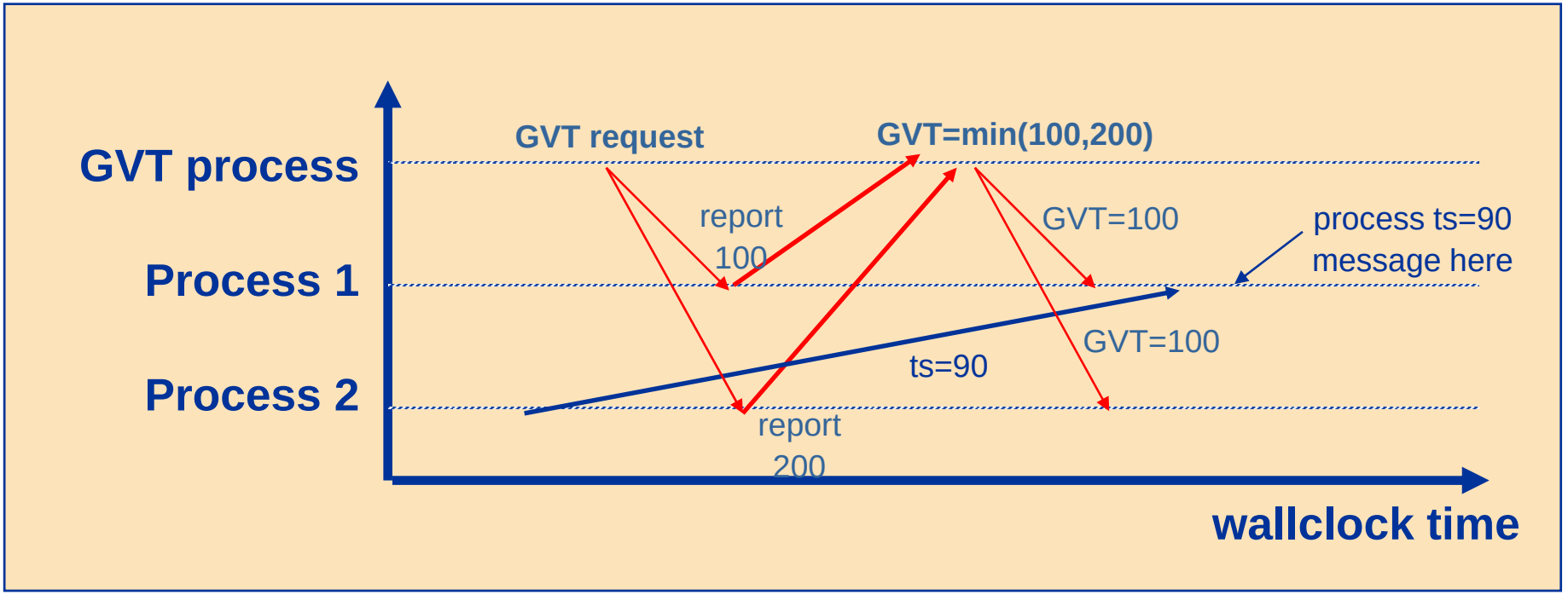
Calcolo asincrono del GVT

- Algoritmo (**sbagliato**) per il calcolo **asincrono** del GVT:
  - Un LP “controllore” inizia la computazione del GVT mandando in broadcast una richiesta di calcolo del GVT
  - Ciascun LP, ricevuta la richiesta, calcola il minimo locale e riporta al controllore il valore ottenuto
  - Il controllore determina il minimo globale e lo comunica a tutti gli altri LP per mezzo di un ulteriore broadcast

- Problemi:
  - Questo algoritmo non tiene conto dei **messaggi transienti** (messaggi già spediti ma non ancora ricevuti al momento del calcolo)
  - I differenti processori riportano il loro minimo locale, ma questi potrebbero riferirsi a momenti diversi rispetto al Wall-Clock-Time, causando un calcolo errato del GVT (simultaneous reporting problem)

Messaggi transienti

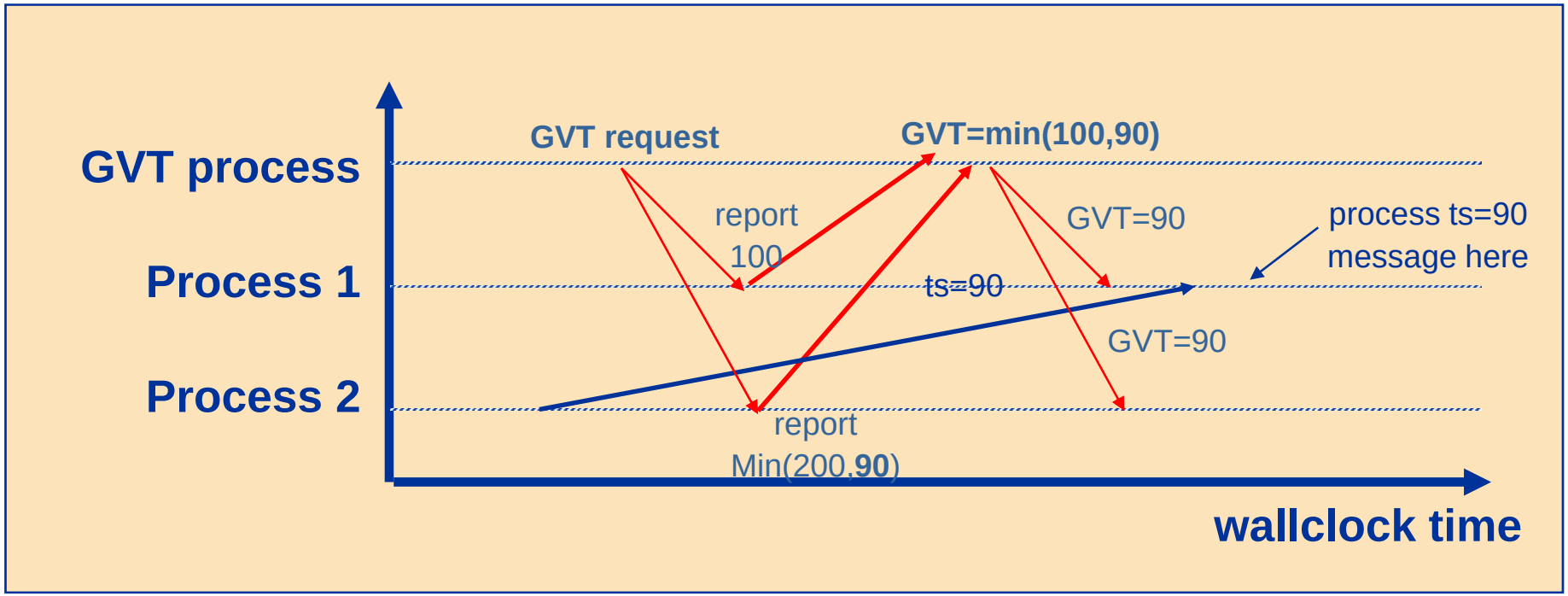
- Definizione:
  - Un messaggio viene definito transiente se è stato spedito ma non ancora ricevuto dalla destinazione
- Problema:
  - Se non si tiene conto dei messaggi transienti il calcolo del GVT sarà errato



Messaggi transienti: una soluzione

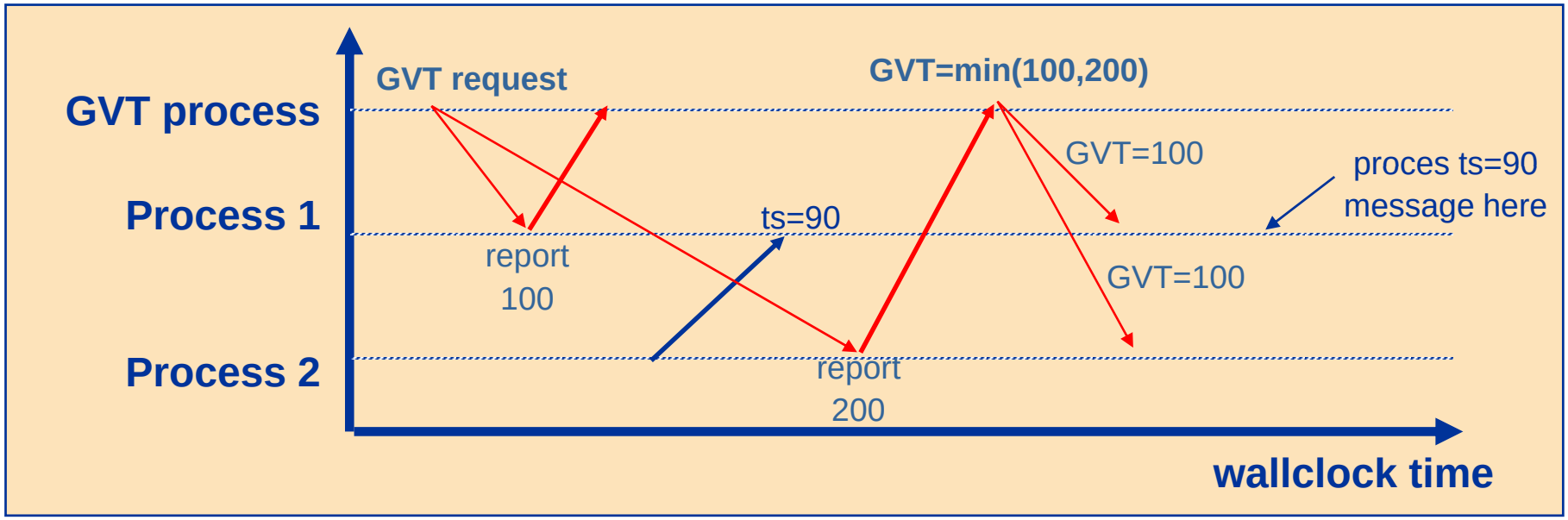
- Approccio:
  - Assicurare che ogni messaggio venga preso in considerazione **da almeno un LP**, durante la computazione del GVT
- Algoritmo:
  - Per ogni messaggio ricevuto viene inviato un acknowledgment (ACK)
  - Il mittente riporta il minimo tra tutti i messaggi inviati e che non hanno ancora ricevuto un ACK
  - Il ricevente è responsabile di considerare **tutti i messaggi ricevuti**

Messaggi transienti: una soluzione



Simultaneous reporting problem

- Valori errati di GVT possono essere calcolati quando i processi ricevono la richiesta di calcolo GVT in momenti diversi



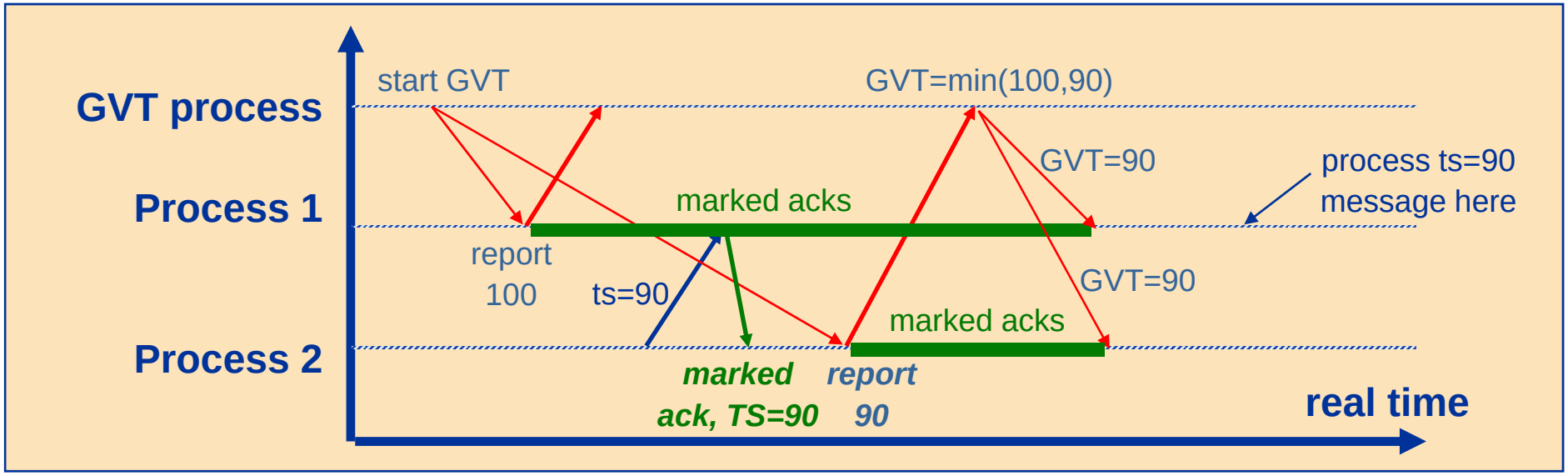
- Il processo 1 non considererà il messaggio con time-stamp 90
- Il processo 2 assume che sarà il processo 1 a considerarlo
- L'approccio basato su ACK dei messaggi è sufficiente?
  - NO, è necessario marcare gli ACK che sono stati spediti DOPO che il minimo locale è stato riportato

Algoritmo di Samadi

- Approccio:
  - Spedisci un ACK per ogni messaggio (e anti-messaggio) che viene ricevuto
  - Marca tutti gli ACK che sono stati spediti **DOPO** che il processo ha comunicato il proprio minimo locale

- Algoritmo:
  - Il controller invia in broadcast la richiesta di computazione del GVT
  - Ogni processo riporta il minimo time-stamp tra: **1)** *i messaggi locali*, **2)** *i messaggi spediti ma senza ACK*, **3)** *gli ack marcati che sono stati ricevuti*
  - I seguenti messaggi di ACK spediti dal processo saranno marcati, fino a quando un nuovo GVT non sarà stato ricevuto
  - Il controller determina il minimo dei time-stamp ricevuti e comunica in broadcast il nuovo GVT

Algoritmo di Samadi



Algoritmo di Samadi

- Vantaggi:
  - Relativamente semplice
- Svantaggi:
  - L'algoritmo è centralizzato
  - Richiede un messaggio di ACK per ogni messaggio che viene ricevuto (implica un raddoppio dell'overhead per ogni messaggio presente nel sistema!)
- Considerazioni:
  - Ogni quanto tempo è necessario / opportuno effettuare il calcolo del GVT?
  - Da quali fattori è determinato l'overhead dovuto al calcolo del GVT?

Altri algoritmi

- Bellenot (1990)
  - Utilizza una topologia ad anello
  - Il LP che inizia la computazione spedisce un token sull'anello che segnala l'inizio del calcolo del GVT
  - Un secondo giro del token è utilizzato per determinare il minimo tra i valori locali, il token funge da mezzo di trasporto per la determinazione del minimo
  - Un terzo giro del token è utilizzato per propagare il nuovo valore di GVT

- Mattern (1993)
  - Basato su tecniche per la creazione di snapshot distribuiti (tagli consistenti)
  - Asincrono, non necessita di messaggi di acknowledgement
  - Piuttosto complesso

Fossil collection

- Batch fossil collection
  - Dopo aver determinato il valore di GVT, scandisce le liste interne ad ogni LP per determinare quale memoria può essere liberata (es. eventi da deallocare, stati ecc) ed esegue le operazioni di I/O che sono in attesa
  - Questo procedimento può essere estremamente costoso e viene eseguito con la stessa frequenza del calcolo del GVT

- On-the-fly fossil collection
  - Ogni evento processato viene immediatamente inserito nella “free memory” list, ma con un timestamp associato
  - Quando il LP ha la necessità di allocare nuova memoria, accede alla free list ma prima di procedere verifica il timestap dello slot di memoria disponibile
  - Se il timestamp è maggiore del GVT attuale scarta quello slot e ne richiede un altro (varie implementazioni ottimizzate del meccanismo sono possibili)

Time Warp, meccanismi avanzati: lazy cancellation

- Motivazioni
  - L'esecuzione di un LP seguente ad un roll-back potrebbe portare alla generazione (e conseguente spedizione) degli stessi messaggi dell'esecuzione "originale" (in certi casi questo sarà molto frequente)
  - In questo caso l'idea è quella di NON cancellare i messaggi originali
- Meccanismo
  - Il roll-back NON provoca la spedizione immediata degli anti-messaggi
  - Dopo il roll-back si procede con la compuazione
  - Gli anti-messaggi vengono spediti SOLAMENTE SE la computazione **non** produce gli stessi messaggi dell'esecuzione "originale"
- Effetti
  - Questo meccanismo può evitare dei roll-back che effettivamente sono inutili

Time Warp, meccanismi avanzati: lazy cancellation

- Problemi
  - È necessario confrontare i messaggi relativi all'esecuzione "originale" e quella successiva. NON è sufficiente confrontare i time-stamp, TUTTO il contenuto del messaggio deve essere verificato. Questo implica un notevole overhead
  - Viene aggiunto un ritardo, potenzialmente molto significativo, nella cancellazione di esecuzioni che sono sbagliate. Durante questo ritardo gli altri LP procedono nell'esecuzione sprecando risorse
  - Un quantitativo ancora maggiore (rispetto alla versione standard) di memoria è necessario
- Prestazioni
  - Come sempre dipende dalla simulazione, dalla piattaforma di esecuzione e dall'efficienza dell'implementazione, ma sono possibili vantaggi rilevabili
- Inoltre... è possibile pensare ad una tecnica di lazy re-evaluation, quindi relativa alla computazione invece della comunicazione?

Global Virtual Time (GVT) e Approfondimenti sul Time Warp

Gabriele D'Angelo
gda@cs.unibo.it
http://www.cs.unibo.it/~gdangelo

Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna