La tolleranza ai guasti

Software tollerante ai guasti

Docente:

William Fornaciari

Politecnico di Milano

fornacia@elet.polimi.it

www.elet.polimi.it/~fornacia

Sommario

- N_version Programming
- Recovery Block
- La tolleranza ai guasti nei sistemi operativi
- La tolleranza ai guasti nelle basi di dati
- La tolleranza ai guasti nelle reti

N-Version Programming

- Corrisponde a NMR per HW tollerante ai guasti
- N versioni sviluppate indipendentemente e compilate con compilatori diversi
  - Se N=2 possibile la sola individuazione dell'errore. La correzione non è intrinseca
  - Se N≥3 è possibile individuazione e correzione mediante voting
- Definire checkpoint di controllo sulle uscite
- Non sempre utilizzabile se sono possibili diversi output corretti e accettabili

Recovery Blocks

- Corrisponde alla ridondanza dinamica per HW tollerante ai guasti
- Consta di tre elementi SW.
  - **Routine primaria**: esegue routine critica
  - **Test di accettazione:** controlla le uscite della routine
  - **Routine alternativa**: realizza la stessa funzione della routine primaria.
- In pseudocodice: **ensure** <test di accettazione>
  by <routine primaria>
  else by <routine secondaria>
  else error

La tolleranza ai guasti nei sistemi operativi

- Un possibile schema: tre tabelle che danno classificazione, localizzazione e azione da intraprendere per ogni errore:
  - valore errore (errore interno, errore CRC, time-out, …)
  - valore locazione(rete, memoria, disco, …)
  - valore azione (aspetta e riprova, ignora errore, termina immediatamente, …)
- In ambiente monotasking (DOS), girando un solo task alla volta la gestione è particolarmente semplice
- In multitasking può ad esempio succedere che si verifichi un errore di I/O anche se nessun processo lanciato dall'utente sta facendo I/O

La tolleranza ai guasti nei sistemi operativi
Esempio: OS/2

- Sfrutta le tre tabelle appena viste
- Caso di mancato inserimento di un dischetto. Può essere che nessun processo stia facendo I/O, ma che il buffering stesso di OS/2 generi l'errore
- Gira in background un *demone* chiamato Hard Error Daemon, che monitora questo tipo di errori
- In caso di Hard Error, si attiva il demone che:
  - fa partire un processo che visualizza l'errore e attende input
  - blocca il processo responsabile, in attesa di risposta
  - dà la risposta al kernel che riattiva il thread incriminato. Se l'errore si ripresenta, si reitera tutto

La tolleranza ai guasti nei sistemi operativi
Esempio: OS/2

Valore errore	Codice	Descrizione
1	OUTRES	Risorsa esaurita
2	TEMPSIT	Situazione temporanea
3	AUTH	Autorizzazione fallita
4	INTRN	Errore interno
5	HRDFAIL	Errore hardware
6	SYSFAIL	Errore del sistema
7	APPERR	Probabile errore dell'applicazione
8	NOTFND	Oggetto non trovato
9	BADFTM	Errore del formato dati o della chiamata
10	LOCKED	Risorsa o dato bloccato
11	MEDIA	Errore dato CRC
12	ALREADY	risorsa/azione già intrapresa/fatta/esistente
13	UNK	Non classificato
14	CAN'T	Azione impossibile
15	TIME	Time-out

La tolleranza ai guasti nei sistemi operativi
Esempio: OS/2

Valore locazione	Codice	Descrizione
1	UNK	Sconosciuto
2	DISK	Disco di risorsa ad accesso casuale
3	NET	Rete
4	SERVDEV	Periferica o risorsa ad accesso seriale
5	MEM	Memoria

Valore azione	Codice	Descrizione
1	RETRY	Ritentare immediatamente
2	DLYRET	Aspetta e riprova
3	USER	Errore dell'utente: ottenere nuovi input
4	ABORT	Termina in modo normale
5	PANIC	Termina immediatamente
6	IGNORE	Ignora l'errore
7	INTRET	Ritenta dopo l'intervento dell'utente

La tolleranza ai guasti nei sistemi operativi
Esempio: MacOS

- Il *Gestore di errori* classifica l'errore secondo tabelle simili a quelle di OS/2
- Il Gestore visualizza una finestra di errore con le opzioni *stop* e *resume*
  - selezionando *stop* il Gestore termina il processo che ha generato l'errore
  - selezionando *resume* il Gestore:
    - recupera lo stato precedente del sistema
    - invoca la procedura di resume, delegata all'applicazione
- Il Gestore è normalmente chiamato dal sistema operativo, non dall'applicazione.

La tolleranza ai guasti nei sistemi operativi
Esempio: Unix

- Esiste una tabella dei **segnali**, non tutti riguardanti la FT, che informano un processo di un evento capitato. Generati da:
  - hardware
  - kernel
  - altri processi
  - utente
- L'ambiente del processo contiene le informazioni su come il processo deve reagire al segnale:
  - azione di default (di solito terminazione)
  - ignorare il segnale
  - *catch* del segnale (si invoca un Gestore dei segnali)

La tolleranza ai guasti nei sistemi operativi
Esempio: Unix

Segnale	Tipo	Descrizione
1	SIGHUP	Hang-up
2	SIGINT	Interrupt
3	SIGQUIT	Fine
4	SIGILL	Istruzione illegale
5	SIGTRAP	Trap di traccia
6	SIGIOT	Istruzione IOT
7	SIGEMT	Istruzione EMT
8	SIGFPT	Errore di virgola mobile
9	SIGKILL	Terminazione
10	SIGBUS	Errore del bus
11	GIGSEGV	Violazione di segmento
12	SIGSYS	Argomento sbagliato nella chiamata di sistema
13	SIGPIPE	scrittura su un *pipe*, ma nessuna lettura
14	SIGALARM	Errore nel clock
15	SIGTERM	Fine del software
16	SIGUSR1	Segnale utente 1
17	SIGUSR2	Segnale utente 2
18	SIGCLD	Morte di un processo figlio
19	SIGPWR	Errore di alimentazione

La tolleranza ai guasti nelle basi di dati
Transizioni

- Si definisce **transazione** un'unità indivisibile di istruzioni, racchiusa tra i due comandi: Begin Transaction (**bot**) e End Transaction (**eot**). Il codice compreso tra BOT ed EOT deve godere di:
  - atomicità: in caso di errore si deve ripristinare lo stato preesistente all'inizio della transazione stessa (*undo)*
  - consistenza: l'esecuzione di una transazione non deve violare i vincoli di integrità della base di dati
  - isolamento: l'esecuzione della transazione deve essere indipendente dall'esecuzione di altre transazioni
  - persistenza: l'effetto di una transazione andata a buon fine non deve perdersi (redo)
- Comandi speciali: **commit** (tutto OK); **abort** (errore)

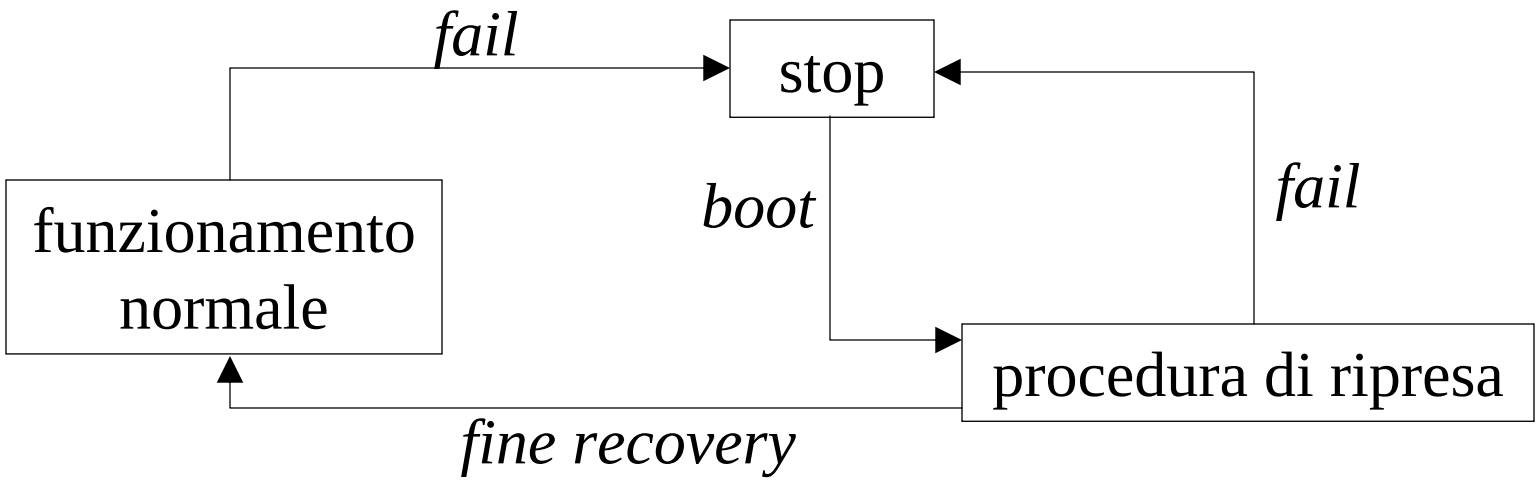
La tolleranza ai guasti nelle basi di dati
Controllo di affidabilità: il *log*

I record del ***log*** possono essere di due tipi:

- Record di **transazione**:
  - **Begin**: riporta l'identificativo della transazione. B(T)
  - **Update**: riporta l'identificativo della transazione, l'identificativo dell'oggetto su cui avviene l'update, e i suoi valori prima e dopo la modifica. U(T,O,BS,AS)
  - **Insert**: come update, senza BS. I(T,O,AS)
  - **Delete**: come update, senza AS. D(T,O,BS)
  - **Commit:** riporta l'identificativo della transazione. C(T)
  - **Abort**: riporta l'identificativo della transazione. A(T)

La tolleranza ai guasti nelle basi di dati
Controllo di affidabilità: il ***log***

- Record di **sistema**:
  - **Dump**: copia completa della base di dati, eseguita in mutua esclusione con altre transazioni. DUMP
  - **Checkpoint**: registra periodicamente le transazioni attive. CKPT(T_1, T_2, … , T_n)
- Modello **fail-stop** in caso di guasto:



La tolleranza ai guasti nelle basi di dati
Controllo di affidabilità: ***warm restart***

- Si attua quando si è persa la sola memoria volatile
- Consiste in quattro operazioni
  - Accesso al log dal fondo, risalendo al primo CKPT
  - Costruzione di due insiemi di transazioni da rifare (REDO, inizialmente vuoto) e disfare (UNDO, inizialmente riempito con le T_k del record CKPT). ripercorrendo in avanti il log, va in UNDO ogni transazione T_s di cui è presente il begin; va in REDO ogni transazione T_r di cui è presente il commit.
  - Si risale, di norma oltre al checkpoint, alla prima azione della transazione più vecchia dei due insiemi.
  - Si applicano undo e redo sulle transazioni dei due insiemi seguendo l'ordine del log.

La tolleranza ai guasti nelle basi di dati
Controllo di affidabilità: ***cold restart***

- Si attua quando, oltre alla memoria volatile, si è persa anche parte della memoria di massa
- Consiste in tre operazioni
  - Accesso al log dal fondo, risalendo al primo DUMP, e si ricopia l'intera base di dati (o la parte deteriorata)
  - Si ripercorre in avanti il log riapplicando sia tutte le azioni sulla base di dati che i commit e gli abort, riportandosi nella situazione antecedente al guasto
  - Si effettua un warm restart

La tolleranza ai guasti nelle reti

- Controllo e correzione possono teoricamente farsi a qualunque livello della pila ISO/OSI)
- Il primo livello (livello fisico) è attualmente molto affidabile, e ci si concentra sui livelli più alti
- Si è spostata l'attenzione dal *byte* al *bit*, per semplificare i protocolli
- La correzione dell'errore viene fatta mediante ARQ (Automatic Repeat reQuest). Ne vediamo tre categorie:
  - Stop and Wait
  - Go back *n*
  - Selective reject

La tolleranza ai guasti nelle reti
Stop and Wait

- Il più semplice protocollo FT nelle reti
- Il trasmettitore, dopo l'invio di un pacchetto, aspetta un riscontro dal ricevente
  - ACK se il pacchetto è stato ricevuto correttamente
  - NAK se il ricevente ha rilevato un errore CRC
- Se non il trasmettitore non riceve riscontro entro un certo tempo (time-out), il pacchetto viene inviato nuovamente
- Per evitare ripetizioni, i pacchetti si possono numerare, purchè la numerazione trovi posto nell'header aggiunto dal protocollo al pacchetto

La tolleranza ai guasti nelle reti
Go back *n*

- Basato sul protocollo Stop and Wait
- Risolve le lunghe attese del protocollo precedente, nel caso di reti con tempi di propagazione molto grandi
- Consente la ritrasmissione successiva di *n* pacchetti consecutivi a partire dall'ultimo consegnato correttamente.
- Si può operare in full-duplex inserendo ACK e NAK negli header di ogni pacchetto (*piggy backing*) o mischiandoli con i pacchetti di dati

La tolleranza ai guasti nelle reti
Selective reject

- Basato sul protocollo Go back *n*
- Il ricevitore conserva in un buffer tutti gli *n* pacchetti ricevuti correttamente, ma fuori sequenza per un pacchetto trasmesso erroneamente
- Il ricevente informa il trasmittente circa quali pacchetti sono arrivati bene e quali no
- Rispetto a Go back *n*:
  - Si introduce un overhead maggiore e si rendono necessarie strutture di memoria aggiuntive (buffer)
  - Si risparmia tempo tempo in ritrasmissioni

La tolleranza ai guasti nelle reti
Protocolli esistenti

- Start Stop: orientato al byte, usato su vecchi terminali asincroni
- BSC: orientato al byte, di IBM. Stop and Wait
- DDCMP: orientato al byte, distingue i pacchetti di dati da quelli di controllo e servizio. Go Back *n*
- HDLC: orientato al bit, è la base dei protocolli commerciali, standard ISO. Go back *n* e Selective Reject
- TCP di 4° livello: insieme al protocollo IP di 3° livello è usato in ambito internet. Go back *n*
- Livelli MAC e LLC IEEE 802: utilizzati per l'accesso multiplo in reti locali