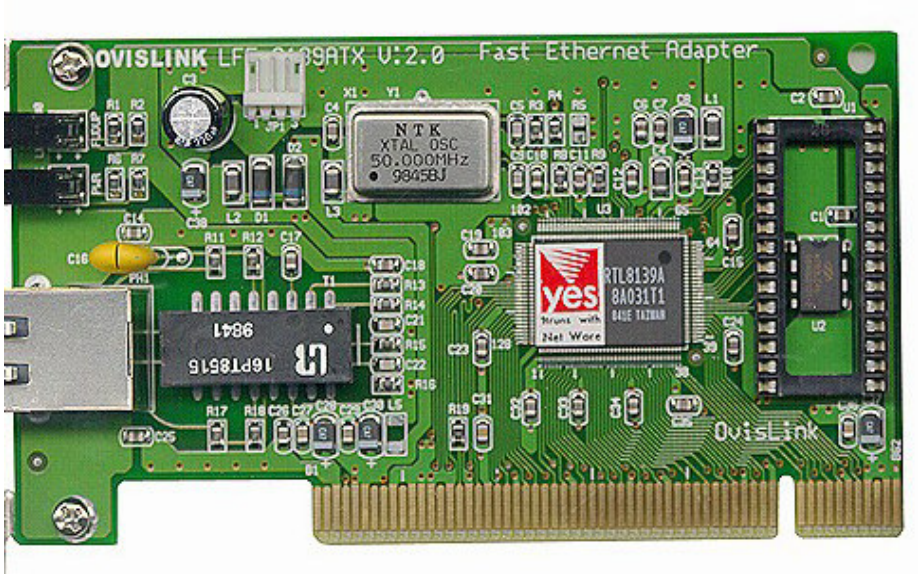
UNIVERSITA' DEGLI STUDI DI FIRENZE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Elaborato per l'esame di
"Informatica Industriale" A.A.1998-99
Prof. A. Fantechi

Andrea Fedeli Sauro Menchetti

Tandem un sistema ad alta disponibilità



Sommario

Introduzione

La Tandem Computers fu fondata nel 1976 per costruire dei sistemi commerciali ad alta disponibilità per l'elaborazione delle transazioni. L'approccio che fu seguito per la realizzazione di tali sistemi fu quello della *ridondanza dinamica*[1] e il Tandem NonStop I fu il primo sistema commerciale disponibile progettato specificatamente per un'elevata disponibilità.

In quegli anni, c'era il bisogno di sistemi tolleranti ai guasti, efficienti e redditizi, per l'elaborazione online delle transazioni. Le architetture di quel tempo erano basate su sistemi multiprocessore in cui i componenti duplicati venivano usati come delle *hot standby*. Il principale difetto di queste architetture era dovuto ai singoli punti di fallimento, come il sistema di alimentazione e i controller dello I/O: un fallimento di tali sottosistemi avrebbe provocato il fallimento di tutto il sistema. Inoltre, tali sistemi non avevano preso in considerazione la possibilità di effettuare la manutenzione online, di costruire un sistema di raffreddamento efficiente insieme ad un sistema di alimentazione distribuito. Tutti questi punti deboli potevano portare ad una perdita di integrità dei dati, con una possibile corruzione del database. Il sistema Tandem si propose di risolvere questi problemi fornendo la possibilità di individuare, sostituire e riconfigurare i componenti guasti senza interrompere il normale funzionamento del sistema, garantendo l'integrità dei dati ed una espansione modulare del sistema. Tutti i singoli punti di fallimento vennero eliminati tramite la duplicazione dei processori, dei dischi, dei percorsi tra i componenti e facendo uso di un sistema operativo basato sui messaggi.

[1] Questi sistemi sono tipicamente composti da più processori ed hanno dei meccanismi di individuazione dell'errore. Quando viene individuato un errore, l'elaborazione viene proseguita in un altro processore.

1.1 Un po' di storia

L'evoluzione dei sistemi ad alta disponibilità può essere riassunta attraverso i tre maggiori venditori commerciali: AT&T, Tandem e Stratus. La AT&T fu la prima a cimentarsi con le applicazioni di tipo telefonico. Il mezzo principale per individuare i guasti è la duplicazione con confronto, con un costo di circa due volte e mezzo quello del sistema non duplicato. La indisponibilità del sistema è di circa 3 minuti all'anno, un buon valore che permette all'utente di avere una bassissima probabilità di incontrare dei problemi durante una chiamata. Le applicazioni telefoniche richiedono un'architettura particolare diversa da quella dei computer usati per applicazioni generali; in particolare, la complessità del sistema è dovuta allo hardware periferico. I quattro maggiori componenti di un sistema telefonico sono l'interfaccia di trasmissione, la rete, i processori del segnale e il controller centrale. La Tabella 1 riporta l'evoluzione dei sistemi della AT&T, includendo il numero di linee telefoniche installate e il modello di processore utilizzato.

Sistema	Numero di Linee	Anno di Introduzione	Numero di Installazioni	Processore
1 ESS	5000 65000	1965	1000	No. 1
2 ESS	1000 10000	1969	500	No. 2
1A ESS	100000	1976	2000	No. 1A
2B ESS	1000 20000	1975	>500	No. 3A
3 ESS	500 5000	1976	>500	No. 3A
5 ESS	1000 85000	1982	>1000	No. 3B

Tabella 1 Evoluzione dei sistemi telefonici della AT&T.

Il sistema 1 ESS fu il primo ad avere un processore con il controllo separato e della memoria per i dati; il processore dello 1A ESS era da 4 a 8 volte più veloce del No. 1; il No. 3A era microcodificato ed in grado di emulare il No. 2; il No. 3B era un processore di tipo generale. Tutti i processori sono totalmente duplicati e il passaggio da un'architettura di tipo dedicato ad una di tipo generale, ha aumentato la complessità del software e del sistema operativo che divengono una parte essenziale nel progetto e nella manutenzione del sistema.

Circa dieci anni dopo l'installazione del primo sistema AT&T, la Tandem Computers progettò un sistema ad alta disponibilità per l'elaborazione online delle transazioni (OLTP). La duplicazione dei vari sottosistemi viene usata non solo per

tollerare i guasti, ma anche per fornire un'espansione modulare del sistema. Tutti i moduli sono duplicati e accoppiati per ridurre la propagazione degli errori: quando un computer viene marcato come primario, l'altro computer, detto di backup, riceve solo delle informazioni periodiche di checkpoint.

La Stratus Computers entrò nel mercato dello OLTP cinque anno dopo la Tandem. Negli anni 80 i microprocessori avevano oramai raggiunto le prestazioni dei minicomputer: sfruttando le loro ridotte dimensioni, fu possibile mettere due microprocessori su una singola scheda, comparando le uscite ad ogni clock. Lo obiettivo principale del sistema Stratus è quello di fornire un servizio ininterrotto senza nessuna perdita dei dati e senza alcuna degradazione delle prestazioni: per questo si fa uso di un'architettura duplicata con componenti self checking. Le schede dei processori e della memoria sono usate in una configurazione pair and spare, in cui un elemento è di scorta e continua l'elaborazione finché non viene rimpiazzato il componente guasto (vedi Figura 1).

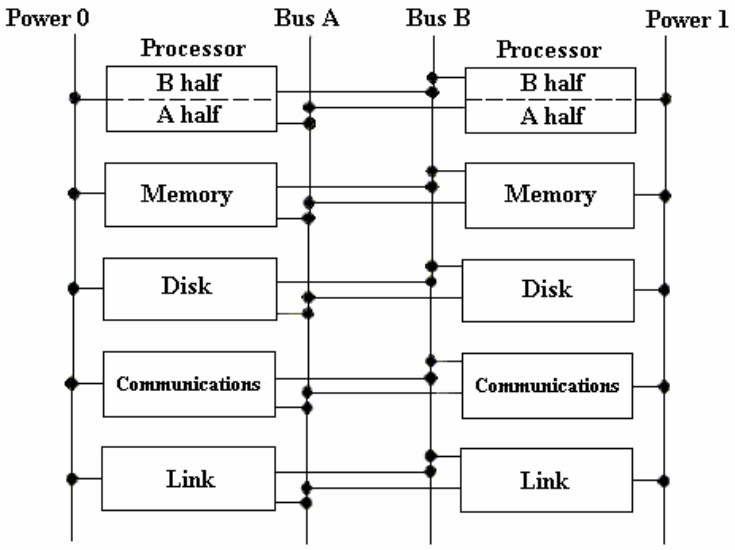


Figura 1 L'architettura pair and spare dello Stratus.

Il sistema operativo esegue dei test per verificare se gli errori avvenuti sono transienti o permanenti: nel caso di errori permanenti, viene fatta una telefonata al centro di assistenza della clientela (CAC). Non c'è bisogno di meccanismi di ripristino software e viene quindi eliminata tutta quella parte complessa di programmazione che riguarda i checkpoint ed i riavvii del sistema. Questo riduce apprezzabilmente i costi annuali per la manutenzione del sistema. La Figura 2 illustra il diagramma di un tipico sistema telefonico.

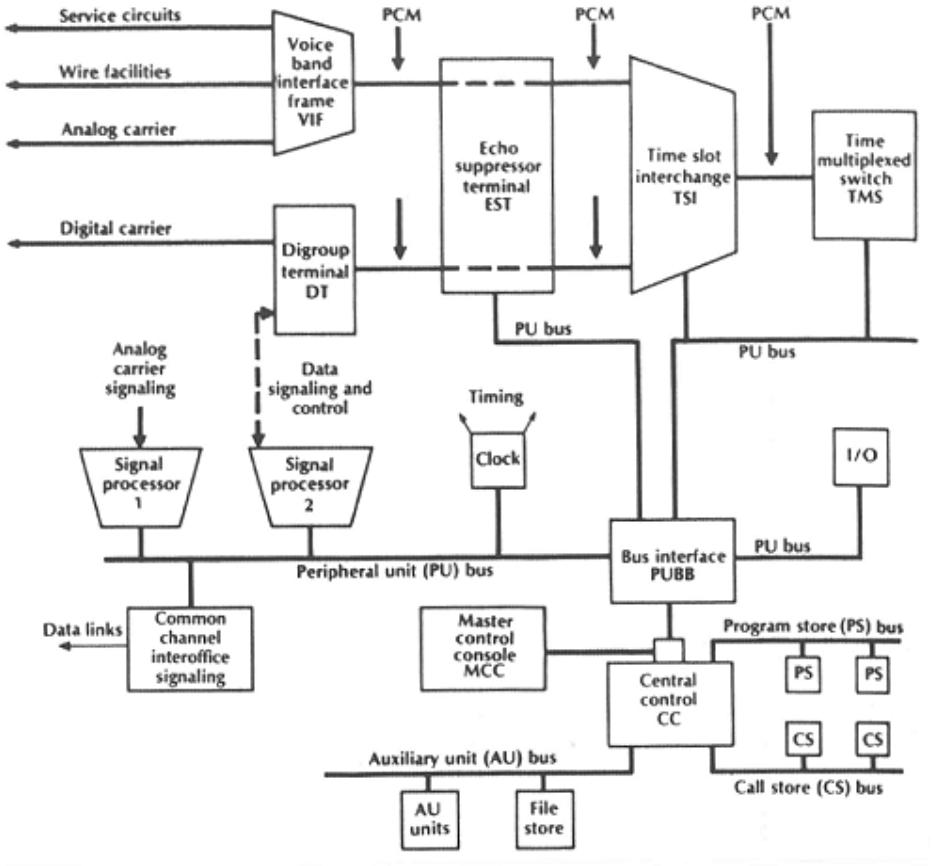


Figura 2 Diagramma di un tipico sistema telefonico.

1.2 Il sistema Tandem

Attorno al 1985 il tasso medio tra fallimenti (MTBF) per sistemi per ben progettati era di circa un fallimento ogni due settimane e il tempo in cui rimanevano guasti era circa di un'ora: tale valore permetteva di raggiungere una disponibilità del 99.6%. Un'analisi delle cause che permettono di raggiungere tali valori indica come lo hardware, il software, l'utilizzo, la manutenzione e l'ambiente contribuiscano a provocare i guasti in misura uguale. Quindi, se lo hardware e il software hanno un MTBF di due mesi, si avrà in media un MTBF di un mese combinando le due cause e di due settimane se inseriamo anche le altre cause di fallimento. L'obbiettivo del sistema Tandem è quello di costruire un sistema con un MTBF di circa 100 anni e questo risultato viene perseguito seguendo alcuni principi di progettazione che possono essere così riassunti:

- *Modularità*: lo hardware e il software sono costituiti da moduli con una granularità molto fine.

- *Fail Fast Operation*: è un meccanismo di risposta rapida ai fallimenti locali basato sull'autodiagnosi di ogni modulo che si ferma immediatamente quando rileva un guasto.
- *Tolleranza ad un singolo guasto*: il sistema è sempre in grado di proseguire la normale attività quando si verifica un singolo guasto, hardware o software. Per poter raggiungere tale obiettivo, occorre duplicare i processori, le memorie e i bus per quanto riguarda lo hardware e i processi per il software: tutto questo permette di raggiungere un tempo medio di riparazione che può essere misurato in millisecondi.
- *Manutenzione online*: i moduli del sistema possono essere controllati, analizzati, riparati e sostituiti senza interrompere il servizio fornito dalla parte restante del sistema: questo vale sia per lo hardware, sia per il software.
- *Interfaccia dell'utente semplificata*: la semplificazione dell'interfaccia dello utente mira a rimuovere i guasti dovuti ad un'errata comprensione del sistema da parte degli operatori.

Hardware del sistema Tandem

Per raggiungere un buon livello di tolleranza ai guasti per quanto riguarda lo hardware, si duplicano i moduli del sistema insieme alle loro connessioni. Un numero sufficiente di moduli hardware replicati per garantire la tolleranza ad un singolo guasto è due: infatti, nel caso in cui uno dei due moduli si guasti, la probabilità che anche l'altro si guasti durante la riparazione del primo è estremamente bassa. In un'architettura con più di due moduli duplicati, in cui anche le connessioni sono replicate, tale probabilità è molto minore di quella di un fallimento dovuto ad altre cause (ad esempio, al software) e può essere quindi trascurata. I principi base per la progettazione dello hardware sono i seguenti:

- *Modularità*: è importante perché permette di sostituire i moduli online e di estendere le capacità di un sistema in modo semplice. Inoltre, se i moduli sono indipendenti, la probabilità che un guasto di uno influenzi il lavoro svolto da un altro è molto bassa.
- *Logica fail fast*: è una tecnica di progettazione dello hardware in cui un componente o funziona correttamente, oppure si ferma: serve ad evitare la corruzione dei dati quando avvengono dei fallimenti.
- *Praticità, durevolezza*: i moduli hardware si dovrebbero guastare il meno possibile, riducendo al minimo la manutenzione; quando proprio è necessario intervenire sul sistema, le operazioni di riparazione non dovrebbero richiedere della capacità e degli strumenti particolari.
- *Costi e rapporto costi/prestazioni*: il cliente non è disposto a pagare dei costi molto elevati per raggiungere un'elevata tolleranza ai guasti e preferisce spesso acquistare dei prodotti meno costosi, sapendo di avere a che fare con sistemi non molto affidabili.

2.1 L'architettura del sistema

L'architettura base del Tandem è riportata in Figura 3: da un punto di vista evolutivo, tutti i sistemi successivi sono stati aggiornati compatibilmente con questo progetto iniziale.

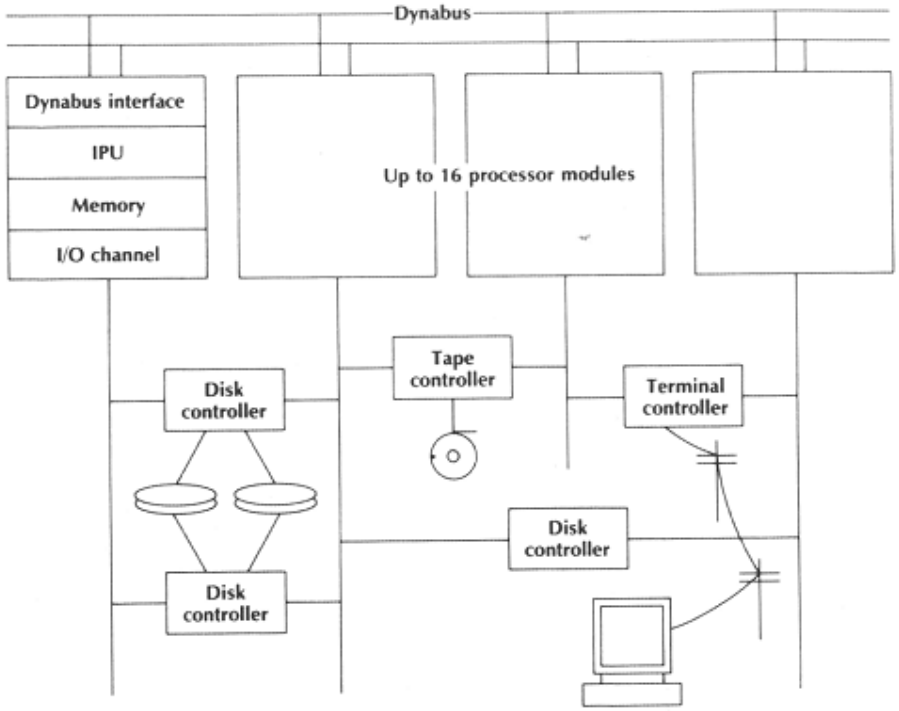


Figura 3 L'architettura originale del sistema Tandem.

Il sistema può contenere da 2 a 16 processori collegati per mezzo di un bus doppio, detto Dynabus Interprocessor Bus: per mezzo di questo bus, i processori possono comunicare tra loro attraverso un meccanismo basato sullo scambio dei messaggi ed il sistema può continuare ad operare anche se si guasta un componente. Ogni processore ha un bus di I/O, una memoria ed una copia del sistema operativo. Quando avviene un guasto in un processore o in un bus di I/O, un meccanismo di commutazione su un altro bus permette di ridistribuire il carico di quel processore su tutti gli altri. I bus di I/O sono collegati a due diversi processori attraverso un controller con due porte che memorizza in un bit quale porta è attiva in un dato momento.

I moduli dei processori rappresentano i componenti più importanti del sistema Tandem: ognuno contiene: una *instruction processing unit* (IPU), una memoria con una copia sua propria del sistema operativo, un canale di I/O e

l'interfaccia con il Dynabus, come illustrato in Figura 4. Il progetto del singolo processore non è molto diverso da quello di un processore tradizionale, a parte l'aggiunta di un meccanismo di controllo degli errori. Ogni processore opera in modo asincrono ed indipendente dagli altri processori e l'interfaccia con il Dynabus è in grado di disconnettere il processore quando viene rilevato un errore da entrambi i bus di cui il Dynabus è composto.

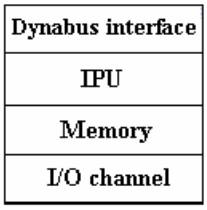


Figura 4 Un modulo del processore.

È stato inoltre considerato attentamente il problema dell'alimentazione: per prevenire un fallimento dovuto a quest'ultima, ci sono due sorgenti di alimentazione. È anche presente una batteria di backup da utilizzare in tutti quei casi in cui avviene un guasto prolungato del sistema di alimentazione principale. Per permettere la manutenzione online, è possibile inserire le schede negli slot a caldo, cioè con l'alimentazione inserita.

Ogni processore fornisce un insieme base di istruzioni; la memoria, inizialmente con un indirizzamento a 16 bit, è passata ad un indirizzamento a 32 bit ed è divisa logicamente in due settori: uno per i dati ed uno per il codice, così come le aree utilizzate dal sistema operativo e dall'utente.

Per assicurare la rilevazione fail fast degli errori, sono stati inseriti nel progetto molti controlli di parità, sia per la trasmissione dei dati, sia per quella dei comandi. Quando un processore rileva di aver commesso un errore, si deve fermare prima di trasmetterlo sui bus di I/O o sul Dynabus. È tuttavia possibile che si abbia un ritardo di alcuni clock da quando avviene un errore a quando viene rilevato, a causa della pipeline interna al processore: tuttavia l'assenza di una memoria condivisa tra i processori permette di tollerare questa latenza. Inoltre, la caratteristica fail fast dei processori permette di evitare la realizzazione di istruzioni di retry quando avvengono degli errori.

2.2 Il Dynabus Interprocessor Bus

È costituito da un due bus indipendenti a cui si accede tramite due controller separati, ognuno con due sorgenti di alimentazione distinte, non facenti parte del processore. Ciascun bus ha 16 linee per i dati ed altre linee per i controlli: tutti i componenti che collegano i processori a ciascuno dei due bus sono fisicamente separati in modo che nessun guasto di un componente possa danneggiare entrambi i bus contemporaneamente. I controller impediscono ad un processore guasto di accedere al bus. Il Dynabus è stato progettato per accogliere le innovazioni future dei processori senza dover essere progettato da capo. Tuttavia, dopo alcune generazioni di processori, è stato introdotto un nuovo Dynabus nel sistema VLX che permette una banda fino a 40 MB/s: tale bus è stato utilizzato anche con le successive versioni dei processori.

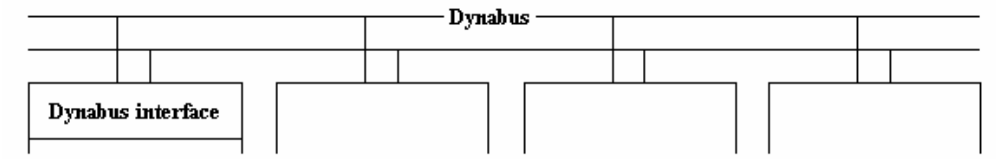


Figura 5 Il Dynabus Interprocessor Bus.

In generale, un bus serve a collegare due processori di cui uno invia i dati e l'altro li riceve. Per trasferire dati sul Dynabus, un processore deve eseguire una apposita istruzione di send in cui vengono specificati il ricevente, il bus utilizzato e il numero di byte che devono essere trasmessi. Possono essere inviati fino a 64 KB con una singola istruzione di send. Il processore che invia i dati continua ad eseguire istruzioni di send finché non è completato il trasferimento dei dati, mentre l'altro processore imprime i dati nella memoria in modo concorrente con la esecuzione di un programma. Vengono attivate delle procedure di ripristino se il trasferimento non viene completato entro un certo intervallo temporale.

2.2.1 Alcune estensioni del Dynabus

Per collegare più sistemi in una rete locale ad alta velocità, nel 1983 è stata introdotta un'estensione del bus a fibra ottica (FOX). Le interconnessioni del sistema FOX a fibra ottica sono illustrate in Figura 6.

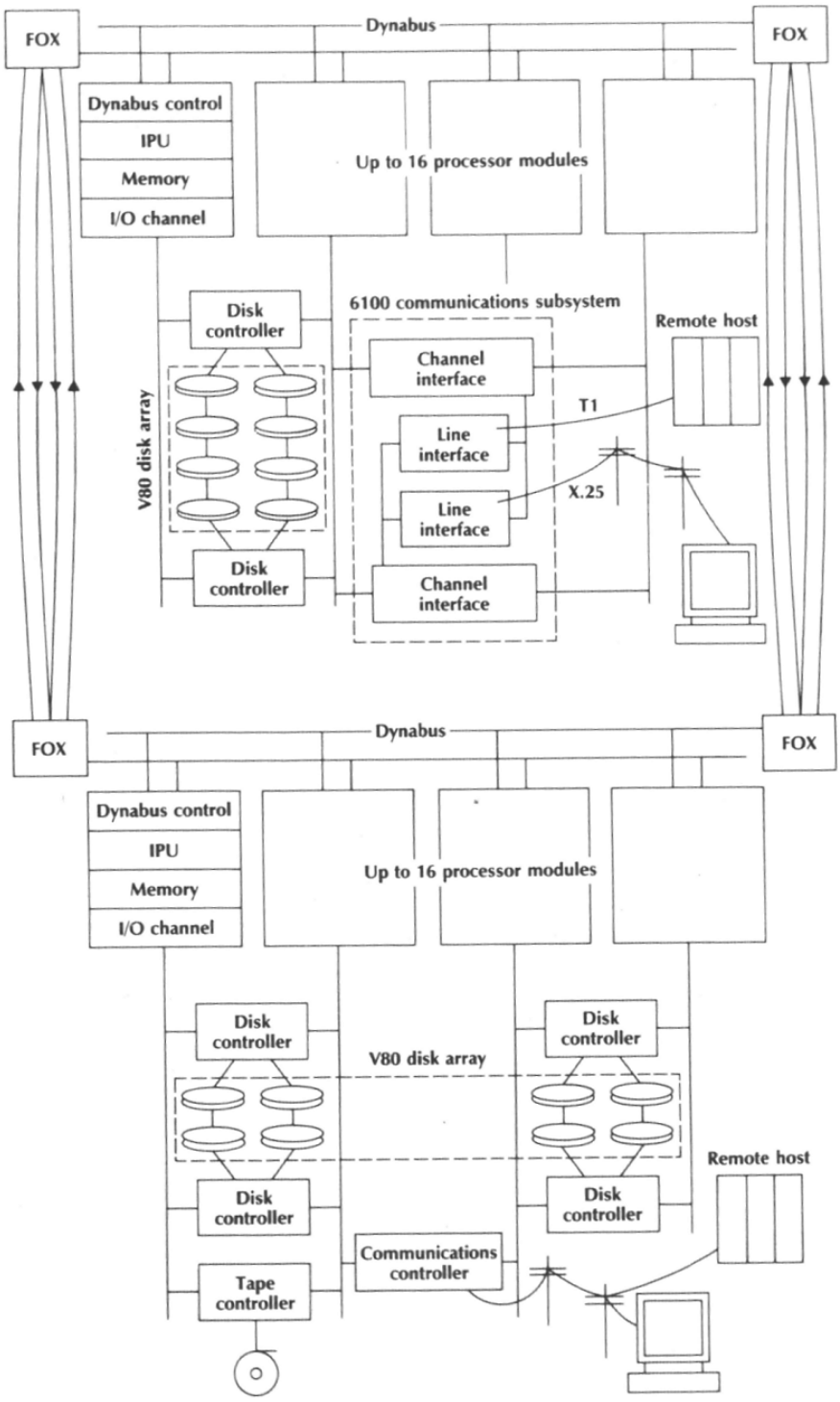


Figura 6 Le interconnessioni del sistema FOX in fibra ottica.

Quest'estensione permette di collegare fino a 14 sistemi ognuno dei quali con al più 16 processori, in una struttura ad anello, per un totale di 224 processori. Nella prima versione la distanza massima tra due nodi adiacenti della rete era di 1

km, salita a 4 km nella versione FOX II. Il collegamento FOX può collegare dei sistemi con processori differenti e far si che ogni nodo possa ricevere o trasmette dati fino ad un massimo di 4 MB/s. La struttura FOX è basata sul modello *store and forward*: quattro fibre connettono un sistema con ognuno dei suoi vicini. In questo modo, ogni bus tra processori è esteso da due fibre ottiche che permettono ai messaggi di muoversi in entrambe le direzioni. La topologia ad anello ha alcuni vantaggi rispetto a quella a stella: infatti, non essendoci un punto di collegamento centrale, si evita che il nodo centrale sia un single point of failure; inoltre l'instradamento è più facile con una struttura ad anello che con una a stella. La banda teorica per una rete ad anello con 14 nodi, ipotizzando che ogni nodo abbia la stessa probabilità di trasmettere, è di 10 MB/s.

L'utilizzo dei collegamenti in fibra ottica ha alcuni vantaggi tra cui:

- non sono soggetti alle interferenze generate dai campi elettromagnetici: si possono così creare dei collegamenti affidabili anche in ambienti molto rumorosi;
- forniscono dei canali di comunicazione con un'alta banda su distanze notevoli: questo permette di collegare molti nodi, evitando la congestione della rete;
- permette di mantenere separati fisicamente i sistemi della rete, fornendo così un elevato grado di isolamento e di protezione contro gli eventi disastrosi.

I collegamenti in fibra ottica furono utilizzati in modo diverso con la introduzione del sistema Cyclone, dando vita all'estensione in fibra ottica del Dynabus chiamata Dynabus+. Nel sistema Cyclone i processori sono raggruppati in sezioni, ognuna delle quali può contenere fino a 4 moduli dei processori; questi 4 moduli sono collegati all'interno di una sezione da normali Dynabus, mentre il collegamento tra le sezioni, distanti non più di 50 metri, è fatto in fibra ottica, in modo simile al sistema ad anello FOX. I singoli collegamenti del Dynabus+ hanno una banda di 12.5 MB/s che è comparabile con la banda del Dynabus che è di 20 MB/s. In realtà si ottengono dei risultati migliori in quanto è possibile raggiungere dei picchi di 160 MB/s: questo perché in ogni sezione è presente un controller per il Dynabus di modo che i traffici locali di ogni sezione possono essere gestiti in maniera concorrente su più collegamenti in fibra ottica. Il sistema Dynabus+ è trasparente a qualsiasi livello di software, eccetto al sottosistema per la manutenzione: questo perché il sistema si configura automaticamente all'accensione. Nel caso in cui si verifichi un guasto ad un collegamento in fibra ottica, il sistema si ri-

configura automaticamente, stabilendo dei nuovi percorsi di instradamento e notificando i cambiamenti al sottosistema di manutenzione.

2.3 Le periferiche

Per ottenere un sistema tollerante ai guasti, non è sufficiente che i processori possiedano tale proprietà: il sistema nella sua interezza deve avere dei percorsi doppi, essere modulare e fail fast ed avere anche un buon rapporto prezzo/prestazioni. L'architettura iniziale del sistema non prevedeva uno schema di interconnessione così ricco tra i terminali e le linee di comunicazione. Nella configurazione illustrata nella Figura 6, ci sono molti percorsi per ogni disco. Tipicamente due controller accedono a ciascun disco e ciascun controller è connesso a due canali dei processori. Il software è usato per fare delle copie specchio dei dischi, cioè i dati sono memorizzati su due dischi distinti così che se uno dei due fallisce, i dati sono disponibili sull'altro. In questo modo i dati possono essere ricercati anche se avviene un guasto nel disco, nel controller, nella alimentazione, nel processore o nel canale. L'architettura base permette di configurare il sistema di I/O così da avere percorsi multipli per ogni dispositivo di I/O: se i controller e le periferiche hanno le porte duplicate, ci sono quattro percorsi per ogni dispositivo. Se poi i dischi hanno una copia specchio di loro stessi, ci sono otto percorsi che possono essere usati per leggere e scrivere dati. Per applicazioni particolarmente critiche, si utilizzano due terminali fisicamente vicini, connessi a due differenti controller. Una caratteristica comune delle periferiche è quella di mescolarne differenti tipi per ottenere un sistema adatto per l'applicazione da realizzare.

Il sottosistema di comunicazione 6100 è stato introdotto nel 1983 ed ha contribuito a ridurre l'impatto dei fallimenti nel sistema di comunicazione. Il 6100 è formato da due unità di interfaccia di comunicazioni con porte doppie (CIU) che comunicano con il bus di I/O di due differenti processori (vedi Figura 7). Delle unità singole di interfaccia di collegamento (LIU) sono connesse ad entrambi i CIU e ai cavi di comunicazione o dei terminali. Con questa sistemazione, i fallimenti delle CIU sono completamente trasparenti e i fallimenti delle LIU hanno il solo effetto di perdere la linea da loro controllata. Un ulteriore vantaggio è che ogni LIU può essere utilizzata con differenti protocolli per supportare diversi am-

bienti di comunicazione. Il sottosistema di comunicazione 6100 è configurato per accettare fino a 45 LIU ognuna delle quali supporta fino a 19 Kb/s di comunicazioni asincrona o 64 Kb/s di comunicazione sincrona. Un'alimentazione ridondante e dei ventilatori offrono un margine ulteriore di tolleranza ai guasti e permettono la sostituzione di un componente durante il funzionamento del sistema.

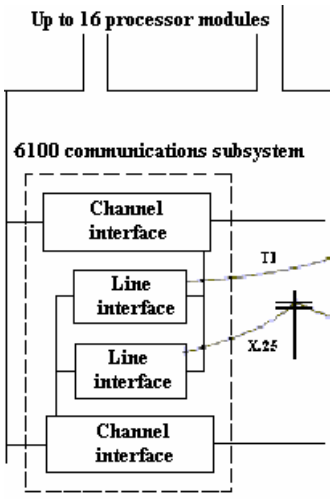


Figura 7 Il sottosistema di comunicazione 6100.

Tra le periferiche del sistema troviamo il sottosistema dei dischi, progettato anch'esso in modo modulare. Nell'elaborazione online delle transazioni è spesso desiderabile riuscire a realizzare degli incrementi della capacità del disco e delle sue prestazioni in modo indipendente l'uno dall'altro. Una volta che un sistema può tollerare i fallimenti singoli, gli effetti del secondo ordine assumo un peso considerevole nel tasso totale dei fallimenti del sistema. Una categoria di guasti critica è la combinazione di un guasto hardware con un errore umano durante l'attività di diagnosi e riparazione. Il sistema dei dischi V8 del 1984 ha ridotto la probabilità di tali guasti composti, riducendo e semplificando l'attività umana. Nel progettare dei sistemi tolleranti ai guasti, è molto importante tenere basso il prezzo delle periferiche, ancora più che nei sistemi tradizionali. Alcune parti del sottosistema delle periferiche devono essere duplicate, anche se questo non comporta niente in termini di prestazioni. Per il mirroring dei dischi ad esempio, si ottiene uno spreco di tempo in lettura ed un raddoppio netto in quello di scrittura. Inoltre il mirroring raddoppia il costo dei dischi per megabyte immagazzinato. Per ridurre tali costi è stato introdotto nel 1986 il disk drive XL8. Questo sistema ha 8 dischi in una singola cabina per una capacità complessiva di 4.2 GB. Come per il sistema

V8, possono essere usati dei dischi nella stessa cabina per il mirror, risparmiando così il costo di un altro sistema ed anche spazio nella ambiente. Inoltre come per il V8, si hanno dei bassi costi di manutenzione. Il sistema V80 rimpiazzò il V8 nel 1988, con una capacità di 2.1 GB distribuita su 8 dischi (vedi Figura 8). Il progetto del V80 estende la capacità e l'affidabilità del V8 con un'interfaccia verso i driver completamente checked. Inoltre il progetto riduce di un fattore 5 il numero di cavi esterni e connettori tra l'unità di controllo e quella di immagazzinamento.

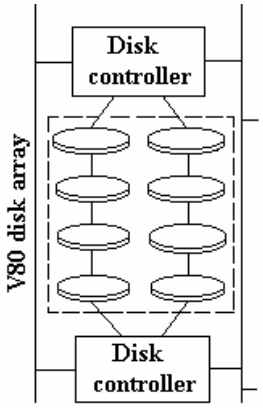


Figura 8 Il sistema dei dischi V80.

L'interfaccia del disk drive è sta costruita basandosi sull'emergente standard industriale IPI 2 con controllo di parità sui dati e sugli indirizzi per assicurare l'integrità dei dati e dei comandi; la precedente architettura SMD effettuava il controllo di parità solo sui dati. L'architettura IPI 2 consente un maggior transfer rate e riduce il numero degli interrupt. Una connessione radiale tra il controller ed i drive permette di eliminare le possibili interazioni tra i drive che generalmente si hanno nelle comuni strutture a bus. La riduzione del numero di cavi è ottenuta inserendo la logica di controllo all'interno della cabina stessa. Nel 1989 lo XL80 ha rimpiazzato lo XL8. In più il sistema XL80 contiene sensori per la temperatura dell'aria, per il controllo dell'alimentazione e per la ventilazione; queste informazioni sono verificate periodicamente dal sottosistema di manutenzione della cabina ed inviate al controller delle periferiche quando si verifica un'eccezione.

I controller delle periferiche devono essere fail fast in modo simile ai processori stessi. Non devono alterare i dati su nessuno dei loro bus anche nel caso in cui si verifichi un guasto e se possibile devono segnalare il malfunzionamento al processore. Nella progettazione è stato enfatizzato l'aspetto della rilevazione degli errori. Per esempio, nel controller del nastro ci sono due processori Motorola

68000 con dei circuiti di comparazione per la rilevazione degli errori. Tutta la logica del controller è totally self checked e ci sono pure dei chip anch'essi self checked per rilevare gli errori in quella parte di logica aggiunta ad hoc nel controller. Oltre a tutto questo, il sistema operativo inserisce un checksum sui dati che viene poi ricalcolato e verificato in lettura. Il controller su una singola piastra che supporta i dischi V80 e XL80, usa una tecnologia CMOS VLSI. È gestito da un doppio Motorola 68010 che fornisce degli strumenti precisi per la rilevazione degli errori e l'isolamento dei guasti. L'evoluzione della tolleranza ai guasti per le periferiche nei sistemi Tandem è riassunta nella Tabella 2.

Year	Product	Contribution
1976	NonStop I	Dual ported controllers, single fault tolerant I/O system
1977	NonStop I	Mirrored and dual ported disks
1982	INFOSAT	Fault tolerant satellite communications
1983	6100 communications subsystem	Fault tolerant communications subsystem
1983	FOX	Fault tolerant, high speed, fiber optic LAN
1984	V8 disk drive	Eight drive, fault tolerant disk array
1985	3207 tape controller	Totally self checked VLSI tape controller
1985	XL8 disk drive	Eight drive, high capacity/low cost, fault tolerant disk array
1986	TMDS	Fault tolerant maintenance system
1987	CLX	Fault tolerant system that is 98% user serviceable
1988	V80 storage facility	Reduced disk cabling and fully checked disk interfaces
1988	3120 disk controller	Totally self checked VLSI disk controller
1989	XL80 storage facility	Reduced disk cabling, fully checked disk interfaces, enviromental monitoring within disk cabinet
1989	Fiber optic interconnect for V80 e XL80	Reduced cabling to a minimum, reduced trasmission errors

Tabella 2 Evoluzione delle periferiche del Tandem.

Evoluzione dei processori nel sistema Tandem

Questa sezione descrive in dettaglio l'evoluzione dei processori e delle periferiche, con particolare riguardo alla tolleranza ai guasti. Come si può vedere dalla Tabella 3, l'architettura dei processori è stata migliorata al passo con la tecnologia. I principali cambiamenti dovuti all'evoluzione tecnologica includono la espansione della memoria virtuale fino a 2 GB nel sistema Cyclone; la introduzione della memoria cache; l'espansione della memoria indirizzabile fisicamente fino a 2 GB; l'inclusione di una cache per le istruzioni separata da quella dei dati; la introduzione di un'architettura superscalare; l'inclusione di una unità indipendente per il fetch delle istruzioni con predizione dinamica. I miglioramenti tecnologici comprendono anche il passaggio ad una memoria centrale di 1 MB di DRAM ed l'evoluzione da Schottky TTL a PAL, a bipolar gate arrays fino a CMOS specializzato.

L'architettura multiprocessore del Tandem permette ad un singolo processore di coprire un ampio raggio di potenza di calcolo. Poiché il sistema è composto da dei processori di varia potenza, si può scegliere quale combinazione si adatta di più per poter raggiungere un certo livello di elaborazione. Per esempio, l'utente può scegliere 4 processori CLX 700 oppure 3 processori TLX o 2 VLX per ottenere la stessa potenza di calcolo. Poiché esiste un'ampia gamma di processori, si può spaziare tra molte applicazioni, da quelle a basso costo fino a quelle molto costose con un alto volume di elaborazione: queste caratteristiche aggiungono un'elevata flessibilità al sistema. La seguente tabella riassume le principali caratteristiche dei processori del sistema Tandem.

	NonStop I 1976	NonStop II 1981	TXP 1983	VLX 1986	CLX 600 1987	CLX 700 1989	Cyclone 1983	CLX 800 1991
Processore								
MIPS/IPU	0.7	0.8	2.0	3.0	1.0	1.5	10.0	2.2
Istruzioni	173	285	285	285	306	306	306	306
Tecnologia	MSI	MSI STTL	MSI, Fast PAL	ECL, Gate array	Custom 2µ CMOS	Custom 1.5µ CMOS	ECL, Gate array	Custom 1µ CMOS
Cycle time	100 ns	100 ns	83 ns	83 ns	133 ns	91 ns	45 ns	61 ns
Microstore		8K × 32b	2 livelli: 8K × 40b, 4K × 84b	10K × 120b, dual	14K × 56b	14K × 56b	8K × 160b std, 8K × 160b pairs	14K × 56b
Cache			64 KB, direct map	64 KB, direct map	64 KB, direct map	128 KB, direct map	2 × 64 KB, direct map	192 KB, direct map
Gates	20K	30K	58K	86K	81K	81K	275K	81K
Schede	2	3	4	2	1	1	3	1
Processo-ri/sistema	2 16	2 16	2 16	2 16	1 6	2 8	2 16	2 16
Memoria								
Virtuale	512 KB	1 GB	1 GB	1 GB	1 GB	1 GB	2 GB	1 GB
Fisica	2 MB	16 MB	16 MB	256 MB	32 MB	32 MB	2 GB	32 MB
Per scheda	64 KB 384 KB	512 KB 2 MB 4 MB	2 MB 8 MB	8 MB 16 MB 48 MB	4 MB (on processor board) 8 MB	8 MB (on processor board) 8 MB	32 MB 64 MB	32 MB (on processor board)
Max schede	2	2	4	2	1	1	2	0
Cycle time	500 ns/2B	400 ns/2B	666 ns/8B	416 ns/8B	933 ns/8B	637 ns/8B	495 ns/16B	414 ns/8B
I/O								
Bus speed (MB/s)	2 × 13	2 × 13	2 × 13	2 × 20	2 × 20	2 × 20	2 × 20	2 × 20
Channel speed	4 MB/s	5 MB/s	5 MB/s	5 MB/s	3 MB/s	4.4 MB/s	2 × 5 MB/s	4.4 MB/s

Tabella 3 Sommario dell'evoluzione dei processori del sistema Tandem.

3.1 Il processore NonStop I

Il processore NonStop I introdotto nel 1976, è costituito da una IPU a 16 bit, dalla memoria principale, dall'interfaccia con il Dynabus e da un canale I/O, come illustrato in Figura 9. Fisicamente la IPU, il canale I/O e l'interfaccia Dynabus sono montati su due differenti piastre ognuna con circa 300 pacchetti di circuiti integrati realizzati in logica Schottky TTL. Il modulo viene visto dallo utente come un processore a 16 bit, capace di utilizzare la memoria virtuale ed in grado di supportare la multiprogrammazione.

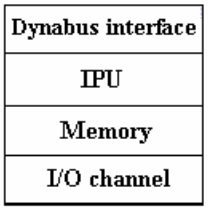


Figura 9 Il modulo del processore del NonStop I.

La IPU è un processore microprogrammato consistente di una unità di esecuzione con la ALU, lo stack, gli shift register e il program counter; un microprogramma con 1024 parole di 32 bit memorizzate in una ROM; delle mappe per la traslazione degli indirizzi in grado di supportare i dati e il codice del sistema e i segmenti del codice e dei dati dell'utente; una memoria principale fino a 512 KB; un chip di memoria da 96 KB con correzione di errore singolo e rilevamento di errore doppio; una batteria di riserva in modo da non perdere i dati in memoria principale fino a 4 ore. Il cuore del sistema di I/O è il canale di I/O. Nel NonStop I, tutte le operazioni di I/O sono fatte attraverso il DMA. Il canale è microprogrammato e permette il trasferimento dei dati in parallelo all'esecuzione di un programma: il sistema di priorità assicura sempre una corretta gestione degli accessi allo I/O prima che la IPU o il Dynabus vi facciano accesso. Il massimo transfer rate è di 4 KB.

I controller dual port forniscono un'interfaccia tra il canale di I/O del NonStop I e una varietà di dispositivi periferici che utilizzano interfacce distinte. Mentre i controller dello I/O sono molto differenti, c'è sempre un tratto comune in tutti loro che varia a seconda del dispositivo con cui si collegano. Ogni controller contiene due porte di canale di I/O indipendenti che sono fisicamente separate l'una dall'altra, così che nessun altro chip di interfaccia può simultaneamente causare il fallimento di ambedue le porte. Logicamente, soltanto una delle due porte è attiva. L'altra porta viene utilizzata soltanto quando un guasto corrompe la prima. Un bit di proprietà, come illustrato nella Figura 10, indica quale porta è la primaria. Il bit di proprietà viene cambiato soltanto quando il sistema operativo invia un comando di Take Ownership I/O. Eseguendo questo speciale comando, il controller scambia il primario con il secondario ed invia a se stesso un segnale di reset. Ogni tentativo di usare un controller senza averne la proprietà provoca una violazione che viene segnalata. Se un processore verifica il malfunzionamento di un determinato controller sul suo canale di I/O, può inviare un comando Disable

Port che disconnette logicamente il processore da quella porta. Questo tuttavia non modifica il bit di stato del controller. In questo modo, se il problema è effettivamente nella porta, può essere utilizzato il percorso alternativo, ma se il problema riguarda la parte comune del controller, il bit di proprietà non viene modificato sull'altro processore.

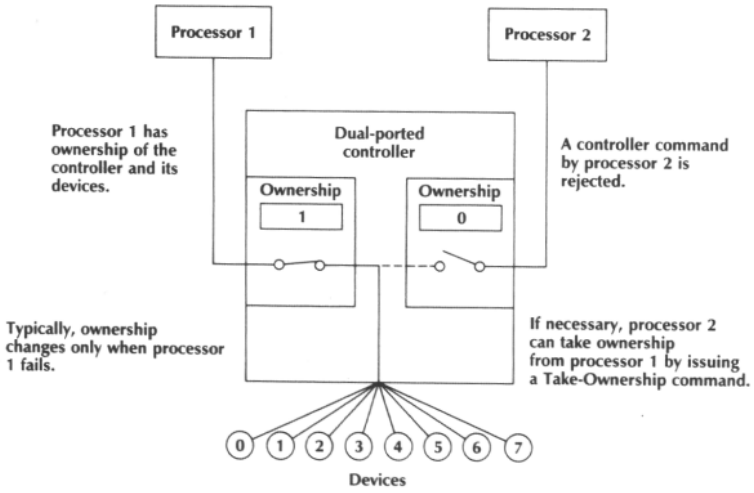


Figura 10 La struttura di un controller dual port nel processore NonStop I.

Facciamo adesso alcune considerazioni sulla tolleranza ai guasti dello I/O. L'interfaccia del canale di I/O consiste di un bus di 2 byte di dati più altri segnali di controllo. Tutti i dati trasferiti sul bus subiscono il controllo di parità in entrambe le direzioni e gli errori sono segnalati attraverso gli interrupt. Un timer watch dog nel canale di I/O verifica se è stato indirizzato un controller inesistente o se un controller ha smesso di rispondere durante una sequenza I/O. La parola che contiene il numero dei byte trasferiti nel canale contiene 4 bit di stato, di cui uno per la protezione. Quando questo bit è attivo, dal dispositivo sono permessi solo i trasferimenti in output. Poiché i controller sono collegati con 2 canali indipendenti, è molto importante che la parola con il conto dei byte trasferiti (word count), l'indirizzo del buffer e la direzione del trasferimento siano controllati dal processore invece che dal controller. Se queste informazioni fossero tenute nel controller, un singolo fallimento potrebbe far fallire i due processori ad esso collegati. Immaginiamo infatti che la word count sia localizzata nel controller e che il conteggio non proceda per un qualche errore durante un input. Sarebbe dunque possibile sovrascrivere il buffer e rendere le tabelle del sistema prive di significato: l'errore si sarebbe propagato sull'altro processore fino a che non si scopre che il primo processore non sta più funzionando. Altre condizioni di errore che il ca-

nale controlla, sono le violazioni del protocollo di I/O, i tentativi di trasferire su pagine non esistenti, alcuni errori di memoria non corretti in altro modo e degli errori di parità della mappa.

3.2 Il processore NonStop II

Il NonStop II è una estensione compatibile del NonStop I. I principali cambiamenti dal NonStop I sono l'introduzione di uno schema di indirizzamento a 32 bit e un processore per il trasferimento dei dati diagnostici (DDT). Il DDT è un microprocessore separato incluso come parte di ciascun modulo che fornisce due distinte funzioni: permette la comunicazioni tra il modulo e il processore di servizio OSP che verrà descritto più avanti e che supporta funzioni sia operative sia di manutenzione, come le routine diagnostiche; verifica lo stato della CPU, dell'interfaccia Dynabus, della memoria, e del processore I/O e segnala ogni errore allo OSP. Il software del NonStop II è un aggiornamento compatibile con il sistema NonStop I. Così i programmi applicativi scritti per il NonStop I possono essere lanciati anche sul nuovo sistema.

3.2.1 La memoria

Le schede di memoria del il NonStop II possono contenere 512 KB, 2 MB o 4 MB di memoria. Fino a 4 di queste piastre, in qualsiasi combinazione, possono essere installate su un processore per un massimo di 16 MB di RAM. Dunque un sistema di 16 processori accetta fino a 64 piastre di RAM per un totale di 256 MB di memoria. Il tempo di accesso della memoria è di 400 ns. Ogni parola della memoria è lunga 22 bit, sei dei quali sono utilizzati per la correzione di errori singoli e la rilevazione di errori doppi. Il codice a correzione d'errore controlla anche l'indirizzo inviato dalla IPU per assicurare che gli accessi alla memoria siano corretti. Lo schema per l'indirizzamento della memoria virtuale introdotto dal processore NonStop II viene usato in tutti i processori successivi: converte un indirizzo a 16 bit in uno a 32 bit. Questo indirizzamento è reso possibile da un apposito set di istruzioni ed è basato sui segmenti che contengono da 1 a 64 pagine di 2048 byte. Ciascun processore può indirizzare fino a 8192 segmenti per un totale di 1

GB di spazio virtuale. Il set di istruzioni permette l'utilizzo di due metodi di indirizzamento: standard ed esteso. Quello standard a 16 bit garantisce un accesso ad alta velocità nello ambito dell'esecuzione di un programma. Quello esteso permette di accedere allo intero spazio virtuale ad alcuni processi privilegiati. In particolare, esistono due tipi di indirizzamento esteso: assoluto e rilocabile. Il primo permette l'accesso a tutto lo spazio byte per byte, ma il suo uso è riservato solo ad alcuni processi speciali, come il sistema operativo; il secondo permette l'accesso all'intero spazio assegnato al processo e ai suoi segmenti di dati privati contenenti fino a 127.5 MB. Per permettere un'efficace traslazione dell'indirizzo fisico a quello virtuale, ogni NonStop II possiede una mappa di 1024 registri ad alta velocità.

3.2.2 La struttura della IPU

La IPU è stata implementata utilizzando una logica Schottky TTL, con un ciclo per microistruzione di 100 ns. Sono state aggiunte istruzione per il supporto dell'indirizzamento a 32 bit. Opzionalmente è disponibile anche un insieme di istruzioni per calcoli in virgola mobile che eventualmente può essere integrato nel chip standard. I set di istruzioni sono implementati su microcodici in una memoria ad alta velocità che ha 8K di parole a 32 bit caricabili dalla memoria e 1K utilizzabile solo in lettura. La prima parte della memoria ad alta velocità viene inizializzata al caricamento del sistema operativo: prima che questo accada, il sistema esegue una serie di routine per verificare che il processore funzioni correttamente. I percorsi per i dati interni al processore ed i registri vengono sottoposti ad un controllo di parità per assicurare l'integrità dei dati. La IPU è caratterizzata da una pipeline a due livelli che permette il fetch dell'istruzione successiva durante la esecuzione dell'istruzione corrente.

3.2.3 Il processore di I/O

Ogni modulo dei processori contiene un processore separato dedicato alle operazioni di I/O. Poiché il processore di I/O opera indipendentemente dalla IPU, i trasferimenti sono estremamente efficienti e richiedono soltanto un minimo di intervento da parte della IPU. Ogni controller dei dispositivi di I/O è bufferizzato, e

permette il trasferimento di dati tra la memoria principale e il buffer del controller alla velocità dei trasferimenti della memoria stessa. I trasferimenti I/O hanno una lunghezza massima di 64 KB. I canali di I/O ad alta velocità usavano tecniche burst multiplexed di accesso diretto alla memoria per raggiungere un transfer rate massimo di 5 MB/s. Così l'insieme di tutti e 16 i moduli permette un transfer rate di 80 MB/s. Il processore di I/O è in grado di gestire fino a 32 controller. A seconda del tipo, i controller possono supportare fino a 8 periferiche, quindi possono essere connesse ad un processore fino a 256 unità. I controller dei dispositivi di I/O sono dispositivi intelligenti e questo permette di far risparmiare alla CPU molto tempo per varie routine.

3.3 Il processore NonStop TXP

Il processore NonStop TXP è stato introdotto nel 1983 per fornire un adeguato supporto all'indirizzamento a 32 bit già presente nel sistema NonStop II che aveva esteso quello del NonStop I a 16 bit. L'esistente banda di 5 MB/s del canale di I/O e i 26 MB/s del Dynabus offrono una banda più che sufficiente per un processore due o tre volte più veloce. L'impacchettamento esistente ha uno slot libero per sviluppi futuri e l'alimentazione può essere riconfigurata per gestire un processore con consumo elettrico maggiore. Il NonStop TXP è stato progettato in modo da essere di facile manutenzione ed efficiente da testare. I registri dei dati e di controllo sono implementati con degli shift register. Questo permette di accedere ai registri in modo più veloce ed efficiente poiché il tester può direttamente osservare e comandare molti punti di controllo. È stato progettato un solo tester per le quattro schede IPU e per la scheda di memoria che costituiscono il processore.

I principali problemi di progettazione riguardarono la creazione di una nuova microarchitettura che permettesse l'esecuzione delle istruzioni a 32 bit ad una velocità ben maggiore con soltanto il 33% di circuiti stampati in più. Il progetto, eliminando alcune caratteristiche che non sono decisive per l'efficienza e trovando dei nuovi modi per risparmiare area sulla piastra, incluse strategie sull'uso di PAL ed un insolito schema di controllo a più livelli. I miglioramenti di velocità ottenuti nel NonStop TXP sono dunque ottenuti da una combinazione di vantaggi nell'architettura e nella tecnologia.

3.3.1 L'architettura del processore TXP

La architettura NonStop TXP usa percorsi doppi a 16 bit, pipeline a 3 livelli, accesso parallelo alla memoria a 64 bit ed una grande cache di 64 KB per processore. Ulteriori guadagni nelle prestazioni sono stati ottenuti aumentando il supporto hardware per l'indirizzamento a 32 bit. La tecnologia della macchina include PAL a 25 ns, chip di RAM statica a 45 ns e la logica Fairchild Advanced Schottky Technology (FAST). Con questi componenti ad alta velocità e con una riduzione nel numero dei livelli logici in ogni percorso, fu utilizzato un clock di 12 MHz (83.3 ns per microistruzione). Il processore NonStop TXP è implementato su 4 grandi piastre utilizzando la logica FAST ad alta velocità, delle PAL e della SRAM ad alta velocità. Ogni modulo contiene da uno a quattro schede di memoria ed ognuna include fino a 8 MB di memoria a correzione d'errore. Un sistema NonStop TXP con 16 processori può contenere fino a 256 MB di memoria.

La disposizione dei data path doppi aumentò ancora le prestazioni attraverso un parallelismo aggiunto, come mostrato nella Figura 11. Una operazione nella ALU poteva essere eseguita in parallelo con un'altra operazione fatta da uno dei moduli speciali. Tra questi moduli speciali venne aggiunta una seconda ALU per eseguire le moltiplicazioni e le divisioni, un barrel shifter, un vettore di 4096 registri, un timer ed un controller degli interrupt. Altri moduli fornivano le interfacce tra la IPU e il bus di sistema, il canale di I/O, la memoria principale ed un processore diagnostico.

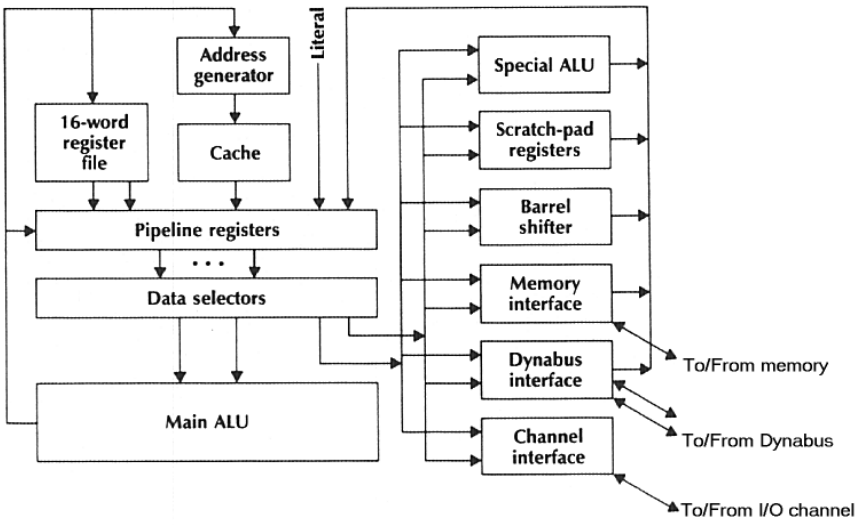


Figura 11 Data path paralleli nel processore TXP.

3.3.2 La selezione degli operandi

La selezione degli operandi per la ALU principale e per i moduli speciali è fatta in due tempi. In un primo tempo i dati vengono presi da un file di un registro dual ported o da registri esterni e collocati in 2 dei 6 registri. Durante lo stesso ciclo, i registri delle altre 4 pipeline vengono caricati con un dato della cache, con una costante, con il risultato della precedente operazione ALU e con il risultato della precedente operazione di un modulo speciale. In un secondo tempo uno dei sei registri viene selezionato per ognuno degli ingressi della ALU principale ed un altro viene selezionato per l'operando di ogni modulo speciale. L'esecuzione della selezione dei registri in due tempi riduce molto il costo dei moltiplicatori e della memoria di controllo. I percorsi doppi per dati a 16 bit tendono a richiedere meno cicli di un singolo percorso a 32 bit quando manipolano dati a 16 bit, mentre richiedono più cicli quando manipolano quantità a 32 bit. Una somma a 32 bit richiede 2 cicli invece di uno, ma gli altri data path sono liberi di usare i due cicli per qualche altra operazione a 32 bit o due operazioni a 16 bit. Delle misure eseguite su delle applicazioni di elaborazione delle transazioni dimostrano che la frequenza di operazioni a 32 bit è piccola in confronto allo spostamento o manipolazione dei dati, i quali sarebbero stati gestiti meglio da un data path doppio che non da un singolo a 32 bit. La maggior parte delle istruzioni includono un parallelismo sufficiente per lasciare che il microcodice faccia un uso efficiente di entrambi i data path. Per controllare il grande utilizzo del parallelismo nel NonStop TXP, occorre un ampio spazio di controllo, che è infatti di oltre 100 bit. Per ridurre il numero di moduli di RAM necessari, la memoria di controllo è divisa in una memoria di controllo verticale di 8K con parole di 40 bit e in una memoria di controllo orizzontale di 4K con parole di 84 bit. La memoria di controllo verticale controlla il primo insieme delle microistruzioni della pipeline e include un campo che indirizza la memoria di controllo orizzontale, i cui campi controllano il secondo insieme della pipeline. Le linee di microcodice che richiedono gli stessi o simili controlli orizzontali, possono condividere le stesse entrate nella memoria di controllo.

Diversamente dai sistemi basati su microprocessori che hanno un microcodice fisso nella ROM, il NonStop TXP lo ha implementato nella RAM così che possa essere cambiato con normali aggiornamenti software e di modo da poter

sempre aggiungere nuove istruzioni più efficienti. Poiché le istruzioni sono caricare in una pipeline, il processore TXP può eseguire le sue istruzioni più veloci in appena due cicli di clock (167 ns). Il processore può anche eseguire frequenti caricamenti di istruzioni in appena tre cicli di clock.

3.3.3 La memoria cache

Ciascun NonStop TXP aveva una cache di 64 KB contente dati e codice. Un sistema di 16 processori NonStop TXP può avere quindi 1 MB di cache. Per decidere come organizzare la cache, sono state effettuate varie misure su un NonStop II utilizzando un particolare monitor opportunamente progettato. Le misure dimostrano che il maggior numero di centri nella cache risulta con una semplice cache grande che non una più piccola e più complessa. Tipicamente il numero di centri nella cache oscilla tra il 96% e il 99%. Qualora il dato non sia presente nella cache, la situazione viene gestita in una routine firmware, invece che con il solito meccanismo di aggiungere una macchina a stati particolare e dei data path dedicati. A causa dei grandi risparmi che si ottengono con la cache hardware, questa dovrà essere sulla stessa piastra, così come i data path primari. La cache viene indirizzata a 32 bit con l'indirizzo virtuale, invece che con l'indirizzo fisico, così da eliminare ulteriori traslazioni tra l'indirizzamento virtuale e fisico. Tale traslazione, necessaria per riempire la cache quando una transazione non la centra, è gestita da una tabella cache che mantiene il mapping di 2048 pagine di 2 Kb ciascuna (vedi Figura 12).

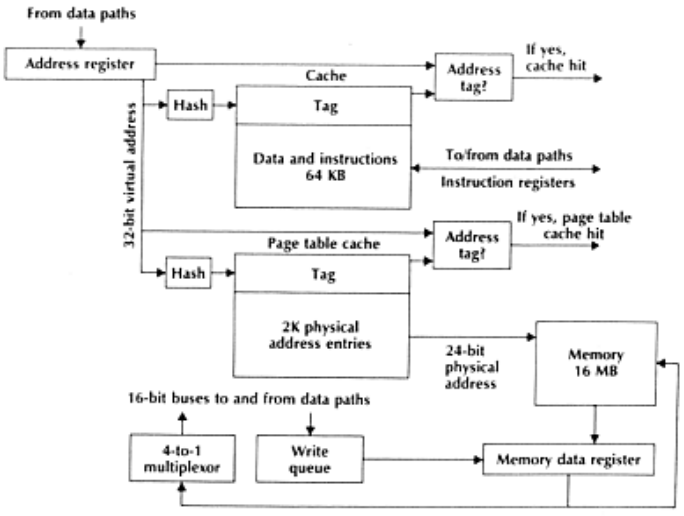


Figura 12 Accesso alla memoria nel processore TXP.

3.4 Il processore VLX

Il processore VLX combina i vantaggi della tecnologia VLSI con le caratteristiche fault tolerant dei suoi predecessori. L'obiettivo principale di tale processore è di ridurre i costi di manutenzione del sistema. Questo processore utilizza una logica ECL per implementare le sue strutture duali e altre estensioni al sistema TXP come 64 KB di cache a mappatura diretta con blocchi di 16 byte, un indirizzamento della memoria fisica di 256 MB, fino a 96 di memoria con schede di memoria da 48 MB, un monitor online della temperatura e della alimentazione ed una pipeline per le istruzioni a 4 livelli che supporta l'esecuzione delle istruzioni in un singolo clock. Nel VLX, il sistema per la manutenzione e la diagnostica del Tandem (TMDS) sostituisce il processore delle operazioni e dei servizi usato nel NonStop II e nel TXP. Più avanti viene descritto in dettaglio il sistema TMDS.

La cabina del VLX, mostrata in Figura 13, è divisa in quattro sezioni: la parte superiore della scheda, quella inferiore, la sezione di raffreddamento e quella di alimentazione. La prima contiene fino a quattro processori, ognuno con il suo canale di I/O e la sua memoria. La seconda parte contiene fino a 24 schede di circuiti stampati per i controller di I/O dove ogni controller contiene da 1 a 3 schede di circuiti stampati. La sezione dei raffreddamento costituita da quattro ventilatori, crea un flusso di aria laminare tra le schede. Infine l'ultima sezione contiene fino a quattro moduli di alimentazione. Queste cabine possono essere combinate tra loro.

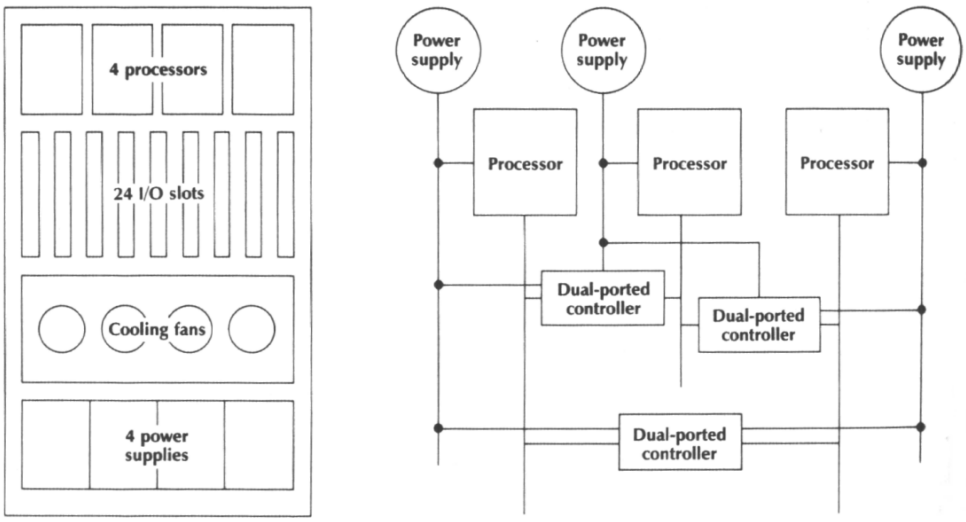


Figura 13 La cabina del sistema VLX e il sistema di distribuzione dell'alimentazione.

I chip del VLX contengono fino a 20000 circuiti e generano dei moduli che sono fino a 3 volte più densi del TXP. La maggior densità permette di aggiungere alcune funzionalità che migliorano il controllo degli errori e la correzione dei guasti, così come anche le prestazioni. Facendo ciò, è possibile ridurre il numero di componenti e di interconnessioni, aumentando quindi le prestazioni e la affidabilità. Il VLX utilizza la logica ECL per migliorare le prestazioni interne e la logica TTL per le funzioni di I/O. Ogni processore VLX contiene 31 porte ECL/TTL distribuite su solo due moduli.

La IPU del VLX utilizza un indirizzamento nativo a 32 bit e 64 bit per i trasferimenti con la memoria principale: tutto questo serve ad incrementare il numero di transazioni eseguibili su un singolo processore, a muovere grandi quantità di dati e a ridurre i costi per transazione. I componenti di riserva nella memoria cache permettono di far fronte ai guasti singoli rimpiazzando il componente guasto con uno di scorta senza chiamare il servizio di manutenzione.

I mainframe tradizionali hanno dello hardware dedicato alla rilevazione degli errori che permette anche di ripetere le istruzioni dopo un fallimento. Questo hardware viene utilizzato per migliorare la disponibilità e per ridurre i costi di manutenzione. L'architettura del Tandem non richiede istruzioni di retry per incrementare la disponibilità ed i processori possono essere fail fast. Il processore VLX è il primo processore Tandem che incorpora un tipo di retry hardware, soprattutto per ridurre i costi di manutenzione.

Nel processore VLX, la maggior parte dei data path e dei circuiti di controllo è costituita da porte ad alta densità che sono estremamente affidabili. Queste caratteristiche fanno si che la SRAM ad alta velocità nella cache e la memoria di controllo costituiscono i maggiori apporti all'inaffidabilità del processore. Sia la cache, sia la memoria di controllo, sono progettate per eseguire istruzioni di retry per gli errori intermittenti ed entrambe hanno una RAM di riserva che può essere utilizzata per continuare le operazioni in caso di un fallimento della RAM. Esistono delle operazioni che garantiscono che ci sia sempre una copia valida della cache nella memoria principale: un errore di parità della cache indica la non correttezza di un dato che quindi viene ripreso dalla memoria principale. Un apposito codice tiene conto del numero di questi errori e, qualora questo ecceda una certa soglia, attiva la RAM di riserva. La memoria di controllo del VLX contiene due copie identiche per permettere ad ogni accesso che richiede due cicli di partire su

dei cicli diversi. La secondo copia della memoria di controllo viene anche usata per ritentare gli accessi in caso di un fallimento nella copia primaria. Di nuovo, il microcodice attiva una RAM di riserva quando viene raggiunta una certa soglia di errori.

Ogni processore ha un'interfaccia per la diagnosi basata su microprocessore che assicura il corretto funzionamento del processore prima della attivazione del sistema operativo. Delle diagnosi pseudocasuali forniscono una copertura di alto livello ed un breve tempo di esecuzione. La correttezza delle operazioni del processore viene verificata prima di iniziare l'elaborazione. Sono anche presenti un controllo di parità su tutti i data path, la correzione di un errore di un singolo bit e l'individuazione di errori di due bit sulla memoria dei dati, così come la individuazione di errori di un singolo bit sugli indirizzi. Le linee del bus di controllo vengono testate per individuare gli errori di linea e dei test di consistenza hardware sono usati su tutto il sistema.

3.5 Il processore CLX

L'obbiettivo principale del progetto del CLX è di fornire all'utente un sistema di progettazione modulare tollerante ai guasti con dei bassi costi di servizi e di manutenzione: il sistema CLX è stato anche progettato per venire incontro alla esigenza di sistemi distribuiti a basso costo. È basato su un chip set particolare in tecnologia CMOS. Il primo CLX 600 introdotto nel 1987 faceva uso di una logica CMOS a 2µ; in seguito si è passati a 1.5µ per il CMOS 700 e a 1.0µ per il CLX 800. Il CLX 700, introdotto appena diciotto mesi dopo il CLX 600, forniva delle prestazioni migliori del 50%; un incremento simile è stato raggiunto dal CLX 800 descritto di seguito.

3.5.1 L'architettura del processore CLX

Tutti i processori CLX hanno un'architettura simile che integra le caratteristiche dei tradizionali dei minicomputer e dei microprocessori VLSI ad alte prestazioni. Questa progettazione ibrida include alcune strutture nuove, come un vettore di RAM statica che può essere utilizzata in tre differenti modi: come memoria di

controllo, come cache per i dati e come cache per le tabelle delle pagine. Il processore include anche dei robusti meccanismi di fault checking per garantire la integrità dei dati. Nel sistema NonStop, le operazioni hardware fail fast sono essenziali per garantire la tolleranza ai guasti. Le operazioni fail fast richiedono che tutti i guasti vengano sempre rilevati e che il processore sia bloccato prima che il guasto si sia propagato. Il modulo CLX utilizza molte strategie per il controllo degli errori e per garantire un'elevata copertura dai guasti.

Una cabina pienamente equipaggiata di un sistema CLX contiene due piastre di processori, sei controller di I/O, cinque disk drive da 145 MB fino a 1 GB ed una unità a nastro. Sono inclusi nella cabina un'alimentazione duplicata e un sistema di raffreddamento. L'intero sistema ha delle dimensioni, delle richieste di alimentazione e di rumore compatibili con quelle di un normale ufficio. Per espandere il sistema, si possono aggiungere delle cabine alla configurazione base. L'architettura del CLX è mostrata in Figura 14.

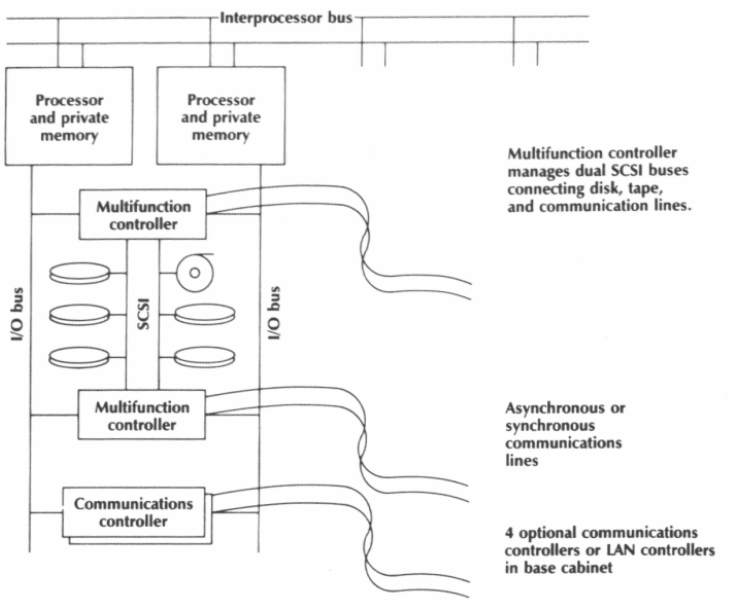


Figura 14 L'architettura del sistema CLX.

Come in tutti gli altri sistemi Tandem, ciascun CLX comunica con gli altri processori attraverso due interprocessor bus (IPB). Ogni bus opera in modo sincrono su dati di 16 bit e fornisce una banda massima di 20 MB/s. I due bus trasferiscono i dati in modo indipendente l'uno dall'altro, fornendo così una banda di 40 MB/s per un massimo di 8 processori. I processori comunicano con i dispositivi di I/O sia attraverso un bus locale di I/O, sia attraverso un IPB verso un altro proces-

sore e verso il suo bus locale. Il bus locale di ciascun processore permette un trasferimento massimo dei dati a 4.4 MB/s ed un massimo di 16 controller. Come con ogni altro processore, i controller sono collegati a due porte distinte e possono essere guidati da entrambi i processori a cui sono attaccati.

La logica del processore risiede in sei chip CMOS e fa si che il processore e la memoria principale possano essere realizzati su una singola piastra. Un diagramma a blocchi del processore appare in Figura 15. I due chip IPU sono identici, formano un modulo pienamente self checking e forniscono la piena operatività della unità IPU; lavorano insieme su un singolo banco di SRAM di cui si è parlato in precedenza.

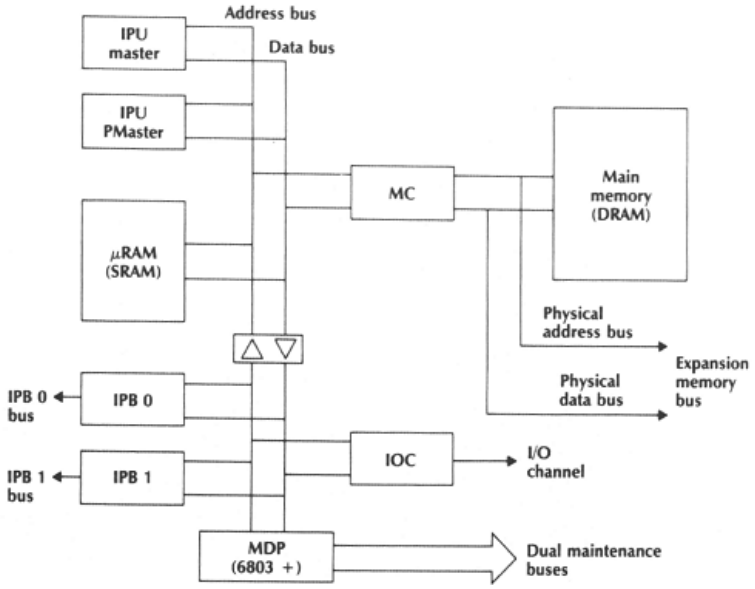


Figura 15 Diagramma a blocchi del processore CLX.

Il CLX utilizza un controller multifunzione (MC) basato sul microprocessore Motorola 68010 per controllare le interfacce SCSI che supportano fino a cinque disk drive ed un nastro. Lo MC carica il kernel del proprio sistema operativo real time che coordina in modo indipendente i comandi per i dischi e per il nastro, le comunicazioni sincrone e asincrone e i compiti di manutenzione remota. Attraverso i bus di manutenzione, le informazioni diagnostiche e di manutenzione possono raggiungere il pannello di controllo, i monitor di sistema, i processori e gli MC. Un software diagnostico avanzato ed un'attenta progettazione di tutte le unità sostituibili permette agli utenti di occuparsi del 98% dei componenti guasti. I chip dello MC includono la logica ECC (SEC/DED) e possono essere installati fino a 32 MB di DRAM sulla medesima piastra. Questi chip contengono un buffer FIFO

per tenere i dati da trasmettere e da ricevere dalla DRAM principale e supportano anche il trasferimento ad alta velocità di blocchi da memoria a memoria. Nella memoria del sistema, lo ECC con la parità degli indirizzi codificata fornisce un controllo a tutti i data path della memoria. In aggiunta, delle macchine a stati ridondanti sono contenute nei chip della memoria e nella logica esterna di generazione dei segnali RAS/CAS. Le transizioni di stato di queste macchine sono codificate in registri CRC le cui uscite sono comparate. La struttura risultante produce una alta copertura ai guasti per dati e per la parte di controllo della memoria principale.

Il chip IOC contiene i latch dei dati e la logica per controllare un bus burst multiplexed asincrono di I/O. Il bus di I/O è controllato principalmente dalla IPU, ma può anche gestire i trasferimenti DMA e la selezione senza l'intervento di nessun microcodice. Il bus contiene anche una logica per gestire la priorità per servire equamente le varie periferiche. Ogni processore ha anche un chip IPB e contiene due code di 16 parole ciascuna, una per l'ingresso e una per l'uscita . Queste code funzionano con una macchina a stati realizzata su chip e servono per inviare e ricevere pacchetti di messaggi in modo asincrono rispetto all'esecuzione. I chip IOC e IPB hanno una protezione di parità sulle linee dei dati e dei comandi a cui sono collegati. Inoltre sono protetti da un checksum software per garantire la integrità dei loro rispettivi bus. L'ultimo componente del processore è un processore diagnostico e di manutenzione basato sul Motorola 6803 che fornisce un controllo totale del processore principale, così come i percorsi diagnostici e di rilevazione degli errori attraverso i bus di manutenzione.

3.5.2 L'architettura della IPU

L'architettura della IPU del CLX riassume le caratteristiche che si trovano nelle architetture dei minicomputer e dei microcomputer. L'interfaccia esterna del chip IPU è simile a quella di un microprocessore VLSI. Infatti include un bus degli indirizzi, un bus dei dati e un bus di stato per vari segnali come le richieste di interruzione, i cicli di attesa della memoria e i comandi del bus three state. Le caratteristiche dei minicomputer si ritrovano nella dimensione dei bus che è di 18 bit per gli indirizzo e di 60 bit per i dati.

L'interfaccia del chip IPU unisce molti bus che dovrebbero essere normalmente separati in un'architettura a minicomputer. In particolare, un ciclo del bus del CLX può eseguire alcune delle seguenti funzioni: accesso alla memoria di controllo del microcodice, accesso alla cache dei dati e delle istruzioni, accesso alle pagine delle tabelle della cache, accesso alla memoria principale e a dei moduli speciali come lo IPB, lo IOC e lo MDP. Questa fusione riduce i costi del processore, diminuendo le parti di SRAM ed il loro supporto logico associato, nonché riducendo il numero di piedini necessari per il chip IPU. Tuttavia se tale fusione non viene effettuata attentamente, si rischia di degradare notevolmente le prestazioni. Per ridurre la banda richiesta sui bus e per minimizzare l'impatto sulle prestazioni, i progettisti hanno impiegato una grande varietà di tecniche riportate di seguito:

1. La micro ROM sul chip è il mezzo principale per ridurre l'impatto della fusione dei bus. La micro ROM contiene 160 parole di microcodice che hanno lo stesso formato del microcodice del chip. Questa ROM è indirizzata dal PC del microcodice oppure attraverso un indice esplicito specificato nelle linee precedenti del microcodice. L'indirizzamento del PC del microcodice viene usato per implementare i cicli più interni dei trasferimenti dello IPB e dello IOC, le routine di riempimento della cache e i movimenti dei blocchi della memoria. L'indice esplicito viene usato per le sequenze brevi del microcodice comune. Queste linee si sovrappongono alle altre linee sequenziali del microcodice esterno. L'uso di queste linee della ROM non crea conflitti con altre microoperazioni.
2. La cache viene indirizzata virtualmente e questo riduce il numero di accessi alle pagine delle tabelle; così si riduce la banda richiesta dalla micro RAM condivisa. Analogamente l'uso di comandi in blocchi per il controller della memoria, riduce il numero di comandi per riempire la cache e il numero di spostamenti di blocchi. Infine, l'utilizzo di comandi a più alto livello per lo IPB e lo IOC riduce i comandi di trasferimento necessari per trasmettere e ricevere dati da questi dispositivi. La micro ROM sul chip, insieme a queste caratteristiche, riduce gli svantaggi derivanti dall'uso di un singolo bus dal 50 al 12 per cento.
3. La principale alternativa alla micro ROM usata nel CLX è uno schema di emulazione nel quale un sottinsieme delle istruzioni è implementato interamen-

te dalla ROM interna e le rimanenti istruzioni sono emulate da una sequenza di istruzioni più semplici. Lo schema con la micro ROM ha due vantaggi rispetto alla tecnica di emulazione. Primo, fornisce delle prestazioni più elevate quando la quantità di spazio della ROM è limitata relativamente al numero di istruzioni che devono essere implementate. Secondo, l'invio di ogni istruzione alla memoria di controllo riscrivibile esterna, abilita gli errori del microcodice della ROM ad essere corretti esternamente anche se con una perdita delle prestazioni.

Il chip IPU stesso è coperto da uno schema di duplicazione e confronto. Questo schema minimizza la quantità di logica interna richiesta per un alto grado di copertura e massimizza l'utilizzo degli elementi di libreria esistenti nei sistemi CAD. L'implementazione dello schema della IPU è illustrata in Figura 16. Lo schema del CLX migliora la copertura dei guasti rispetto ad altri schemi di duplicazione e confronto fornendo un accoppiamento dei dati e delle uscite di parità. Un chip, detto data master, controlla tutte le uscite dei dati, mentre un altro chip detto parity master, controlla tutte le uscite di parità. Questa azione assicura che le uscite dei chip e che la logica di controllo siano attive e che gli errori latenti nella logica di controllo non possono portare ad un fallimento doppio non rilevato. Le uscite di parità della IPU coprono anche le linee dei dati e degli indirizzi collegando la IPU ad altre parti del processore e alla micro RAM.

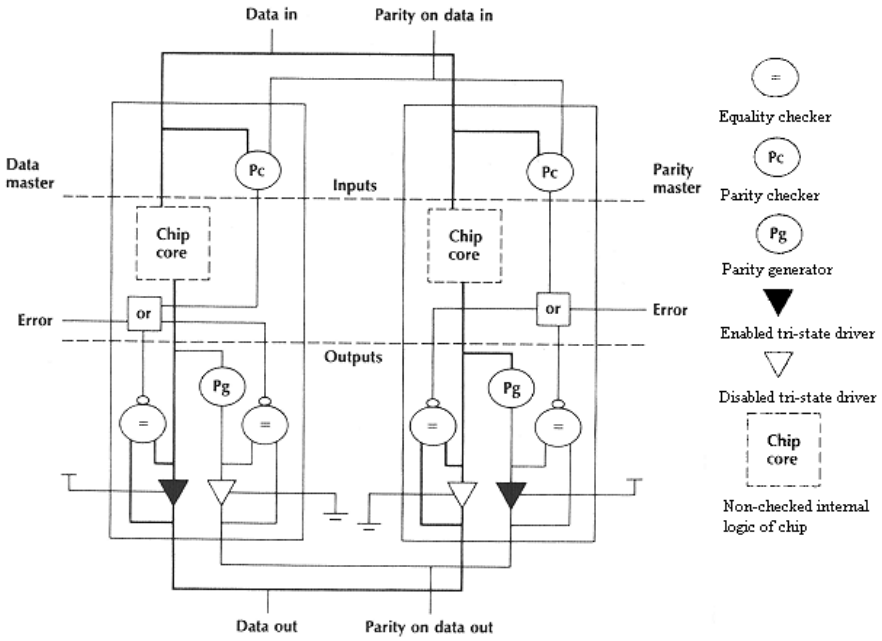


Figura 16 Duplicazione e confronto della IPU nel processore CLX.

3.6 Il processore Cyclone

L'obiettivo del progetto del sistema Cyclone, introdotto nel 1989, fu quello di migliorare significativamente le caratteristiche del suo predecessore, come, ad esempio, le prestazioni, riducendo parallelamente i costi. Il processore Cyclone ha delle prestazioni tre volte superiori a quelle di un VLX, pur mantenendo una completa compatibilità a livello di codice oggetto. L'incremento delle prestazioni è dovuto alla più alta frequenza di clock ed alla nuova microarchitettura che ha reso possibile l'esecuzione di due istruzioni per ciclo di clock: questa architettura è stata chiamata *superscalare*. Altri miglioramenti sono dovuti al parallelismo dei data path e a dei nuovi metodi di progettazione della cache e della memoria principale.

3.6.1 L'architettura del processore

Il processore è realizzato su tre schede 18"×18", mentre una quarta scheda contiene da 32 a 64 MB di memoria principale. Una seconda scheda di memoria opzionale può consentire un'espansione fino a 128 MB di memoria principale per processore. Come il VLX, il Cyclone usa una memoria di controllo che permette due cicli di clock per accesso. La memoria di controllo è implementata in una SRAM CMOS 16K×4, montata verticalmente su un doppio strato di ceramica sulle schede principali. L'architettura superscalare del Cyclone fu necessaria a causa del fatto che il VLX eseguiva molte delle istruzioni frequenti in un singolo ciclo di clock e l'obbiettivo di migliorare così tanto le prestazioni non poteva essere raggiunto basandosi ancora su istruzioni eseguite in un solo ciclo di clock. Infatti, aumentare troppo la frequenza avrebbe portato ad una minore affidabilità e a costi molto maggiori. Così il Cyclone ebbe bisogno di rompere la barriera di una istruzione per ciclo di clock. Nei picchi, il Cyclone esegue due istruzioni per ciclo di clock. Per fare ciò, il processore incorpora una unità indipendente asincrona per il fetch delle istruzioni (IFU), delle grandi memorie cache separate per le istruzioni e i dati, una pipeline profonda ed un meccanismo di predizione dinamica dei salti. Un diagramma a blocchi del processore Cyclone è in Figura 17.

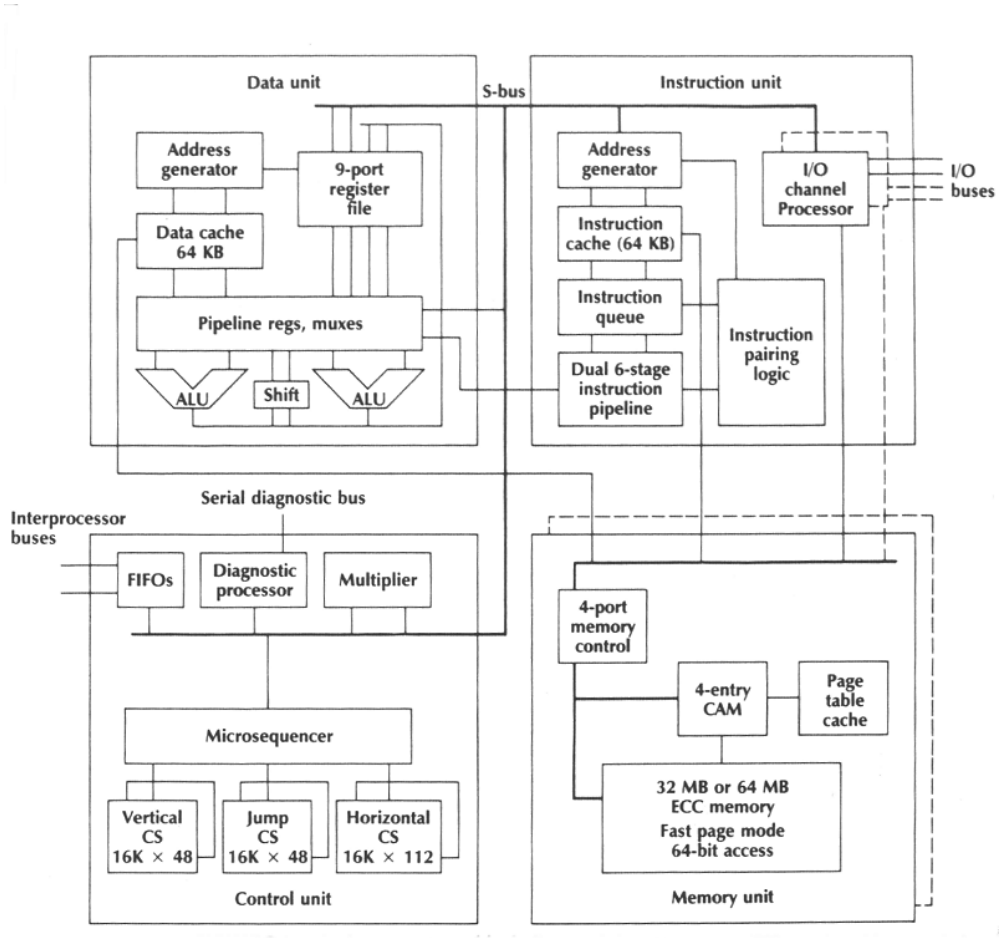


Figura 17 Diagramma a blocchi del processore Cyclone.

Per aumentare la connettività e la banda, il processore Cyclone permette la connessione fino a 4 canali di I/O per processore, mentre i precedenti processori Tandem ne avevano solo uno. Due canali sono forniti sulla piastra principale, mentre gli altri due sono disponibili su una piastra opzionale. Il numero massimo di schede in un sistema Cyclone è dunque sei: tre processori, due memorie ed una opzionale per lo I/O. La tecnologia usata è una combinazione della logica ECL per la velocità, CMOS per l'alta densità e TTL per le interfacce standard e per i bus. I gate array ECL, sviluppati congiuntamente con Advanced Micro Devices (AMD), sono l'equivalente di circa 5000 porte standard ed hanno 155 pin che possono essere programmati ad uno ad uno per le interfacce TTL ed ECL.

Il sistema Cyclone fa un uso esteso di interconnessioni in fibra ottica, che hanno anche il pregio di aumentare l'affidabilità. Il Dynabus+ è già stato descritto in precedenza. Inoltre il sistema Cyclone utilizza dei collegamenti in fibra ottica anche tra controller dei dischi e le unità disco stesse.

3.6.2 L'unità per il fetch delle istruzioni

La IFU opera indipendentemente dal resto del processore: è in grado di eseguire il fetch di due istruzioni per ciclo di clock dalla cache delle istruzioni, le decodifica per capire se sono candidate per una esecuzione parallela e presenta l'indirizzo di partenza del microcodice dell'istruzione singola o della coppia di istruzioni alla unità di controllo e all'unità per l'esecuzione dei dati. La IFU segue anche la esecuzione delle istruzioni di salto e la gestione delle eccezioni. Ci possono essere fino a 16 differenti istruzioni in un certo istante ad un dato livello di esecuzione. I dettagli della IFU sono illustrati nella Figura 18.

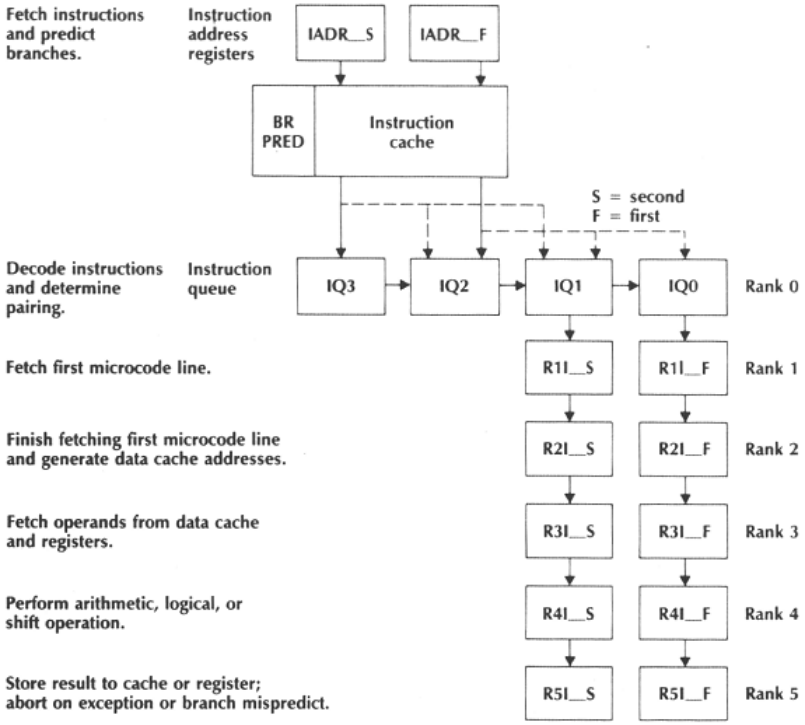


Figura 18 Unità per il fetch delle istruzioni (IFU) nel processore Cyclone.

Il processore Cyclone usa un meccanismo di predizione dinamica dei salti condizionati. Questo meccanismo è basato sull'idea che quando un'istruzione di salto viene incontrata ripetutamente, viene quasi sempre o quasi mai eseguita. Viene quindi aggiunto ad ogni istruzione un bit extra nella cache delle istruzioni: questo bit registra la direzione effettivamente presa dal salto nell'ultima esecuzione dell'istruzione nella cache. Quando un salto viene caricato di nuovo dalla cache delle istruzioni, la IFU predice che il salto sceglierà lo stesso percorso della volta precedente di modo da continuare a precaricare le istruzioni di quel salto.

Quando poi il salto entra nella coda di esecuzione, il microcodice verifica se la predizione era corretta. Se è così va tutto bene, altrimenti il microcodice segnala alla IFU di annullare tutto il lavoro di caricamento dopo quell'istruzione di salto e di caricare le istruzioni lungo l'altro cammino. Dei modelli hanno dimostrato che le predizioni saranno corrette tra l'85% e il 95% delle volte. Il risultato è un costo medio da 1.3 a 1.9 cicli per istruzioni di salto. In aggiunta, poiché le predizioni dei salti avvengono nella coda di prefetch, i salti possono essere eseguiti in coppia con la precedente istruzione o sequenzialmente con l'istruzione seguente.

3.6.3 Altre caratteristiche del processore Cyclone

Il data path del Cyclone usa due ALU a 16 bit, simili a quelle del TXP e del VLX, ma con due grandi differenze:

- le due ALU sono connesse con i registri in modo molto generico. Questa interconnessione è necessaria per l'esecuzione di molte coppie di istruzioni, ma è anche molto utile nel migliorare le prestazioni di molte istruzioni complesse, così come di quelle multiciclo;
- in aggiunta, le due ALU possono essere collegate insieme in modo da eseguire un'operazione aritmetica a 32 bit in un solo ciclo di clock.

La cache per le istruzioni e quella per i dati sono capaci di caricare due parole adiacenti di 16 bit in un solo ciclo, indipendentemente dall'allineamento. Questa caratteristica, assieme all'esecuzione parallela delle istruzioni, ai registri con nove porte, alla possibilità della ALU di gestire 32 bit e alla pipeline profonda, permette l'esecuzione di una istruzione di caricamento a 32 bit e di una istruzione aritmetica a 32 bit in un singolo ciclo di clock. Inoltre nel processore Cyclone gli indirizzi virtuali sono direttamente inviati alla memoria principale e lì è presente un meccanismo per la traslazione dell'indirizzo virtuale in quello reale qualora l'indirizzo inviato non trovi una immediata corrispondenza nella memoria. Poiché la traslazione non viene fatta frequentemente, è possibile implementare le pagine della cache in delle CMOS SRAM relativamente lente ma dense.

Il sequencer del Cyclone è simile a quello del VLX in quanto usa le due copie della memoria di controllo per permettere l'accesso in due cicli. Inoltre per permettere l'uso di alcune parti di CMOS RAM più lente e più dense, le due copie forniscono un backup l'una per l'altra. Nel caso di un errore nel caricamento di

una parola dalla memoria di controllo, il banco rimanente viene automaticamente attivato. Se l'errore non è grave, uno dei due banchi può essere rinfrescato dallo altro. Nel caso di un fallimento grave, viene attivata una RAM di riserva. Parte della memoria di controllo è duplicata ancora una volta (quattro copie dunque). Questa duplicazione permette ad entrambi i possibili percorsi di un microcodice di essere caricati simultaneamente, così da minimizzare la penalizzazione per la scelta del microcodice.

3.6.4 La tolleranza ai guasti nel sistema Cyclone

L'approccio del Cyclone la diagnosi dei è similare a quello del VLX, ma lo migliora in molti aspetti. I test di copertura delle routine diagnostiche microprogrammate sono stati enormemente aumentati ed è stato aggiunto un ulteriori supporti per i test a scansione pseudocasuale. Tutti assieme questi cambiamenti migliorano la capacità automatica online di diagnosi dei guasti e indicano velocemente qual è l'unità responsabile del guasto che deve essere sostituita. In modo simile al VLX, il Cyclone è implementato in porte ECL, anche se di densità maggiore. Poiché è stata aumentata la densità ed è stato aumentato il clock rate, queste porte dissipano fino a 11 watt. Dunque per abbassare la temperatura senza utilizzare dei sistemi a raffreddamento a liquido, il Cyclone utilizza una particolare tecnica di raffreddamento ad aria, detta ad urto. Invece di far circolare l'aria sopra tutto il circuito stampato, questa viene convogliata, attraverso una apposita griglia, verso i componenti più caldi della scheda. Il risultato che si ottiene è di far operare i dispositivi ad una temperatura di giunzione di circa dieci gradi inferiore a quella del VLX, migliorando ancora l'affidabilità complessiva del sistema.

Se lo hardware del processore rileva un errore che non può essere recuperato, si spegne automaticamente da solo in due cicli di clock, prima di trasmettere il dato sbagliato sull'interprocessor bus o sul canale di I/O. L'errore viene registrato in uno o più dei circa 300 registri per l'identificazione dell'errore, permettendo un isolamento veloce del guasto da parte di uno dei 500 meccanismi hardware per le rilevazione degli errori presenti in ogni processore. Come il VLX, il Cyclone include dei dispositivi RAM di riserva per la cache e per la memoria di controllo. Questi dispositivi di riserva sono attivati automaticamente nel caso in cui avvenga un fallimento del dispositivo primario.

Il controllo di parità è utilizzato per rilevare gli errori singoli. La parità è propagata attraverso dispositivi che non alterano i dati, come le memorie, i segnali di controllo, i bus e i registri. La predizione della parità è usata sui dispositivi che alterano i dati, come le unità aritmetiche e i contatori: è basata strettamente sugli ingressi dei dati e della loro parità e non è collegata alle uscite del dispositivo che potrebbe essere guasto. Così ad esempio, se un sommatore esegue una somma sbagliata, il predittore della parità calcolerà la parità corretta, segnalando quindi l'errore. Il moltiplicatore hardware è protetto da una nuova tecnica simile alla ripetizione dell'operazione con gli operandi shiftati (RESO). Dopo ogni moltiplicazione, viene avviata una seconda moltiplicazione con gli operandi scambiati di cui uno shiftato. Il microcodice compara i due risultati ogni volta che il moltiplicatore ne ha bisogno o prima che ogni dato lasci il processore. A differenza di altre implementazioni di RESO, non ci sono delle perdite nelle prestazioni, perché i cicli dei test sono eseguiti concorrentemente ad altri passi di esecuzione non correlati.

Il sistema Tandem Integrity S2

Il sistema Tandem Integrity S2 fu progettato per coprire quella parte di mercato che richiedeva un sistema operativo aperto basato su standard ed un meccanismo di tolleranza ai guasti implementato in hardware (ad esempio, l'industria delle telecomunicazioni). La maggior parte dei sistemi tolleranti ai guasti tipicamente usava dei sistemi operativi proprietari come il sistema NonStop del Tandem, il quale fornisce una efficiente tolleranza ai guasti attraverso il suo hardware fail fast ed il suo sistema operativo proprietario Guardian. Purtroppo la gran parte delle applicazioni commerciali esistenti sono state scritte per sistemi basati su UNIX.

L'obiettivo principale nel progettare il sistema Integrity S2 fu quello di fornire un sistema online tollerante ai guasti durevole e pratico per l'utente, basato sul sistema operativo UNIX. Fu richiesta la portabilità delle applicazioni a livello sorgente insieme ad un supporto per un bus periferico che aderisse agli standard industriali. In aggiunta, nessun fallimento dello hardware avrebbe dovuto corrompere i dati immagazzinati o manipolati dal sistema. Infine, fu riconosciuto che il sistema operativo avrebbe potuto rappresentare un punto singolo di fallimento.

Lo Integry S2 combina quindi numerose tecniche hardware e software appropriate per un mercato commerciale dominato dagli standard che richiede dei sistemi tolleranti ai guasti. La tolleranza ai guasti è stata realizzata senza compromettere l'interfaccia di programmazione, il sistema operativo o le performance del sistema.

4.1 L'architettura del sistema

Il sistema è suddiviso in un insieme di unità sostituibili dal cliente (CRU), progettate per essere sostituite durante il normale funzionamento del sistema, permettendo così la rimozione e l'inserimento online delle CRU guaste e di sostituzione. Un'illustrazione della struttura della macchina è mostrata nella Figura 19.

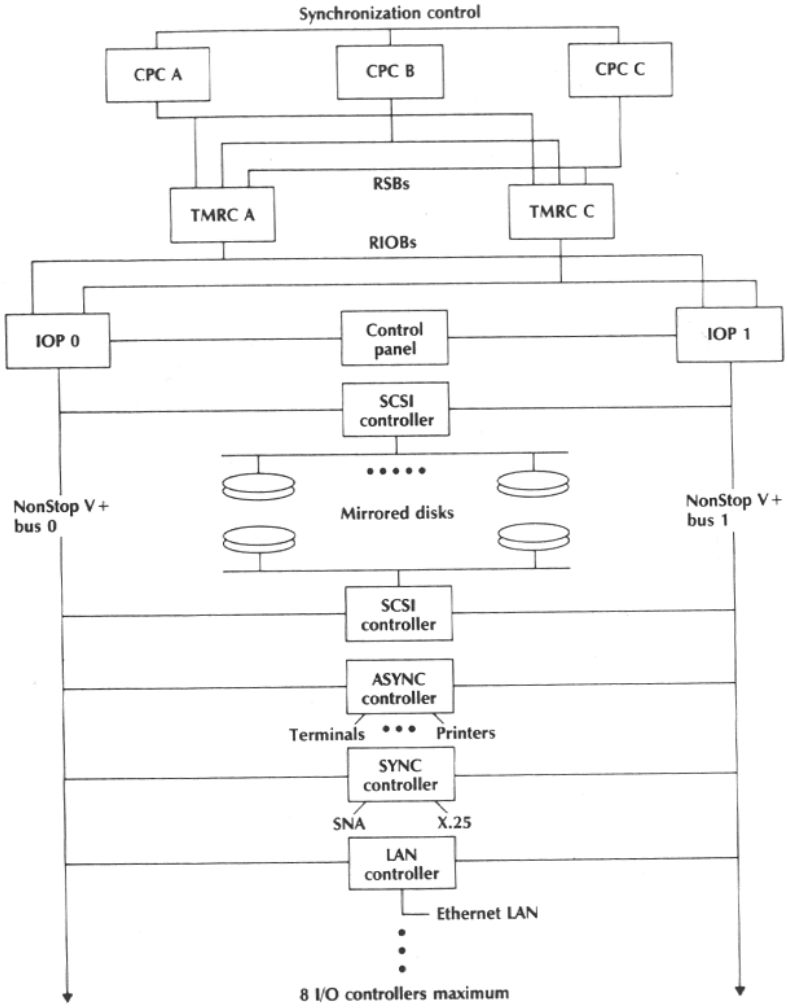


Figura 19 L'architettura del sistema Tandem Integrity S2.

Il cuore del sistema è organizzato in una struttura triplicata costituita da delle unità di elaborazione con memoria locale, disposte su tre processori centrali (CPC). Due controller con ridondanza modulare triplice (TMRC) forniscono una grande memoria principale secondaria (detta memoria globale) e servono come legame per le operazioni di I/O della macchina. Le CPC sono connesse ai TMRC attraverso il reliable system bus (RSB). Una doppia pacchettizzazione di in-

put/output (IOP) fornisce da un lato una interfaccia per un insieme di bus industriali di I/O (VME) e dall'altro una interfaccia verso il TMRC. Le interconnessioni tra gli IOP e i TMRC sono chiamate il reliable system bus di I/O (RIOB). Gli IOP sono i condotti attraverso i quali fluisce tutto lo I/O della macchina e da essi nasce un bus chiamato NonStop V+ che è una variante ad alta integrità dei dati del popolare bus VME. Dei controller VME comuni sono connessi al sistema attraverso un modulo di interfaccia verso il bus (BIM), che fornisce un percorso dual ported dai controller delle periferiche ad ogni IOP.

4.1.1 La struttura di una CPC

Ogni CPC è composta da un oscillatore a 33 MHz che guida un processore R2000 e un coprocessore in floating point R2010, da 128 KB di cache di istruzioni e di dati, da della memoria locale e da una interfaccia RSB. Inoltre una CPC contiene un DMA usato per trasferire blocchi di dati tra la memoria locale e la memoria secondaria, un minimo di 8 MB di DRAM con della logica di protezione per la scrittura e una interfaccia RSB. Il meccanismo del DMA trasferisce pacchetti tra la memoria locale e quella globale e accumula un checksum dei dati durante il trasferimento che viene utilizzato dagli strati superiori del sistema software per fornire un checksum finale per il trasferimento del disco.

4.1.2 Il TMRC e lo IOP

Il TMRC contiene fino a 128 MB di memoria globale e delle interfacce verso le CPC attraverso gli RSB e verso gli IOP attraverso gli RIOB. Il TMRC contiene anche dei registri per il meccanismo delle interruzioni e presenta una vista uniforme degli interrupt ai processori, avendo anche il compito di servire come deposito per le cause delle eccezioni. Un ruolo centrale che il TMRC ha nelle operazioni del sistema è quello di votare le transazioni del RSB. Tutte le transazioni dei processori che sono esterne alle CPC, sono votate bit a bit e il risultato dello OR di questi valori indica se un modulo CPU è in errore. Il circuito di votatura è duplicato e comparato ed ogni errore di self check blocca la scheda. Durante il funzionamento del sistema, un TMRC è disegnato come primario e l'altro di backup: quello primario fornisce i dati nel caso di un'operazione di lettura ed entrambi i

TMRC adempiono a tutte le operazioni di scrittura. Tutti i registri e le RAM statiche dei TMRC sono protette da un bit di parità pari per ogni byte di dati (bit per byte parity). La memoria è organizzata in word e la parità pari della parola di indirizzo del dato è mappata tramite una funzione hash in ogni byte di parità dei dati per individuare i fallimenti di indirizzo nella memoria del controller. La memoria non volatile (EPROM) del TMRC non è protetta dalla parità ma da un checksum implementato in software. Tutti i data path in ogni TMRC sono protetti dalla parità pari e mappati tramite una funzione hash nei quattro byte di dati. Le macchine a stati sono protette usando la parità o la duplicazione. Degli scrubbers, implementati nel sistema operativo, sono usati per individuare e correggere se possibile gli errori latenti nella memoria locale e globale.

Come il TMRC, lo IOP è stato progettato per essere una CRU fail fast self checking. Il sistema ha due IOP ognuno dei quali può supportare il pieno complemento dei controller delle periferiche. Se un IOP fallisce, il software predispone i controller delle periferiche ad adattarsi all'altro IOP ed il sistema continua le sue operazioni. Il data path nello IOP è testato usando tecniche analoghe a quelle descritte per il TMRC. Tutte le transazioni del bus periferico inizializzate dal controller si risolvono in transazioni NonStop V+ che sono traslate in transazioni del bus RIOB dallo IOP. Per evitare che dei controller delle periferiche in errore scrivano o leggano delle celle non appropriate di memoria globale, lo IOP contiene una RAM di accesso e validazione (AVRAM). Questa AVRAM è direttamente mappata sulla cache che trasla gli indirizzi VME virtuali in indirizzi fisici RIOB. Durante il processo di traslazione, lo IOP testa il bit di permesso nelle entrate della AVRAM per vedere se è permesso al controller specifico della periferica di leggere o scrivere l'appropriato indirizzo fisico del RIOB.

4.1.3 La sincronizzazione delle CPC

Il clock dei tre moduli di elaborazione non ha una relazione di fase fissa mantenuta dal sistema. I processori operano indipendentemente, ma sono tenuti in fase logica per mezzo di una logica proprietaria di sincronizzazione. Ci sono due differenti domini temporali che sono rilevanti per la sincronizzazione: *virtuale* e *fisico*. Il tempo virtuale è misurato dal passaggio delle istruzioni in una data CPC. I moduli di elaborazioni indipendenti sono progettati per eseguire lo stesso flusso di i-

struzioni in tempo virtuale. Questi flussi di istruzioni procedono finché il complesso delle elaborazioni ha bisogno di accedere ad una risorsa al di là dei limiti della CPC. Tutte le risorse non facenti parte di una CPC richiedono la generazione di transazioni nello RSB che sono votate dal TMRC. Le operazioni di lettura votate portano inerentemente i processori ad un allineamento più vicino in tempo fisico in quanto c'è una singola copia logica dei dati. Poiché la macchina può operare nei limiti del sottosistema formato dalla cache e dalla memoria locale per lunghi periodi di tempo, serve un altro meccanismo di sincronizzazione. L'andamento dei processori è monitorato in ogni CPC da un insieme di contatori che sono incrementati come la pipeline della macchina avanza. Periodicamente (ogni 2047 istruzioni) ognuno dei processori viene fermato finché tutti gli altri processori non sono arrivati al punto di sincronizzazione o finché non scade un certo timeout. Inoltre l'arbitro del bus locale sulle CPC assicura che tutte le macchine eseguano le letture e le scritture nello stesso ordine temporale.

4.1.4 Altri aspetti del sistema

Anche il sistema di alimentazione è progettato per essere tollerante ai guasti; infatti sono fornite delle batterie per il supporto continuo alle operazioni del sistema durante i fallimenti dell'alimentazione. Inoltre dei redundant bulk supplies guidano delle linee indipendenti a 36 V_{DC} per la protezione contro i bulk failures e delle batterie ridondanti guidano delle linee a 24 V_{DC} per la protezione contro i fallimenti delle batterie stesse. I convertitori DC CD usano queste quattro linee indipendenti per produrre il voltaggio DC necessario richiesto da una specifica CRU.

Le tecniche escogitate per fornire una precisa presentazione delle eccezioni a ciascun processore includono il conteggio delle istruzioni. Come la pipeline del processore avanza, un certo numero di contatori sono incrementati. Nel sistema corrente, gli interrupt possono essere presentati ogni 64 istruzioni. Il processo di raggruppare, distribuire, votare e presentare le eccezioni ad uno specifico modulo di un contatore di istruzioni garantisce che tutti i processori eseguiranno l'eccezione allo stesso tempo virtuale.

4.2 Il sistema software

L'architettura del sistema software è basata su una implementazione standard industriale di UNIX, in quanto uno degli obiettivi base del progetto era quello di usare un kernel UNIX esistente per il sistema operativo. Una importante aggiunta fatta al kernel è quella di un sistema per la gestione della memoria in due livelli: infatti, la memoria principale del sistema è suddivisa in locale e globale. Questo sistema di memoria virtuale tratta la memoria locale come una alleata di quella principale, con funzione di sostegno alla paginazione; inoltre, le pagine della memoria globale saranno trasferite nel disco se la memoria globale non è in grado di soddisfare le richieste. Nella memoria locale vi risiedono il testo, la maggior parte dei dati e gli stack per i processi del kernel e vi possono risiedere anche quelli degli utenti.

4.2.1 I core fault

Gli errori che avvengono nel complesso formato dalla memoria e dal processore (CPC e TMRC) sono chiamati core fault. Lo hardware garantisce che il sistema operativo può continuare ad eseguire le istruzioni per effetto della riparazione del sistema, ma il sistema operativo deve identificare il componente guasto ed eseguire le appropriate azioni di ripristino. Un core fault è segnalato ai processori attraverso un interrupt ad alta priorità. A questo punto i processori usano un meccanismo privato di scrittura per distribuire una visione globale della condizione del guasto. Il meccanismo privato di scrittura permette ai processori di immagazzinare dei dati privati non votati possibilmente asimmetrici nella memoria globale senza lanciare il meccanismo di votatura che avrebbe registrato il guasto. Una volta che ognuno dei tre processori ha ottenuto una descrizione comune della causa, una coda di elaborazione delle eccezioni di basso livello segue un meccanismo di analisi deterministica dello stato dello hardware. Questa analisi fornisce una indicazione di un possibile malfunzionamento alla CRU, la quale lancia subito delle azioni dipendenti dal tipo e dalla severità del guasto e dal grafo della configurazione corrente del sistema. Alcune tipiche azioni, in ordine di probabilità, sono l'incremento di un contatore a soglia, la registrazione di un evento nel meccanismo di registrazione degli eventi o per ultimo la rimozione di una CPC o di un

TMRC. I guasti non riguardanti le CPC e i TMRC ma gli IOP o i controller delle periferiche, sono gestiti dallo strato di gestione dei guasti dello I/O. I fallimenti dello IOP sono tipicamente recuperabili. In un dato momento un insieme di controller è collegato ad entrambi gli IOP attraverso il BIM e il sistema può ancora accedere ai controller per mezzo di un percorso alternativo.

4.2.2 Le operazioni di I/O

Il sistema operativo lancia le operazioni di I/O a vantaggio dei processi dello utente e fa in modo che i dati siano mossi verso la destinazione appropriata. Se la sorgente del dato per una operazione di scrittura è in memoria locale, allora è allocato un buffer nella memoria globale ed i dati sono quindi trasferiti dalla memoria locale a quella globale per mezzo dello hardware DMA dedicato a quel compito implementato nelle CPC e nei TMRC. In modo simile, se la destinazione di un'operazione di lettura è nella memoria locale, il trasferimento dei dati eseguito dal sistema operativo usa questo meccanismo di DMA dopo che il dato è stato posto nella memoria globale dal controller della periferica. L'insieme dei servizi che forniscono l'illusione di operazioni fault free al sistema è chiamato subscription based service. Una entrata nel sistema operativo che richiede la notifica di un evento, chiama una funzione per sottoscrivere la occorrenza di quell'evento. Se avviene un evento, viene invocata una funzione specificata dal chiamante con i parametri che dipendono dal particolare tipo di evento.

4.2.3 La reintegrazione delle CPC e dei TMRC

La capacità di riparare il sistema online è un mezzo per raggiungere un'alta affidabilità nei sistemi replicati. Il supporto da parte del sistema operativo per tale attività di riparazione è chiamato reintegrazione. Consideriamo l'integrazione di una CPC. Una CPC appena inserita lancia un power on self test (POST) dalla sua E-PROM locale per verificare le condizioni della scheda. I processori rimanenti si accordano per reintegrare e copiare una piccola parte dello stato locale nella memoria globale. Poi i processori rilasciano un soft reset (una operazione supportata dallo RSB e dal TMRC e quindi votata) il cui risultato è quello di entering reset in tutti e tre le CPC. Le CPC notificano che questa è un'operazione di soft reset e

dopo l'inizializzazione di una parte dello stato locale, trovano il blocco di comunicazione depositato dal sistema operativo nella memoria globale e caricano il program counter e lo stack pointer. A questo punto il sistema operativo ha di nuovo il controllo e procede a memorizzare una routine di copia nella memoria locale di ogni processore. Poi i processori usano il meccanismo del DMA locale globale per muovere le pagine dalla memoria locale a quella globale e viceversa, usando il meccanismo di votatura per portare il contenuto di tutte e tre le memorie locali in accordo. Quando questo processo di copiatura è terminato, riparte la normale esecuzione. Da notare che l'elaborazione è sospesa durante il periodo di copia necessario per la reintegrazione della CPC: si parla di poco più di un secondo in un sistema con 8 MB di memoria locale.

A differenza dei processori, la reintegrazione dei TMRC avviene durante le normali operazioni della macchina e si fa prestare dei cicli dalla macchina in piccoli blocchi che sono controllati dall'amministratore del sistema. La prima fase della reintegrazione del TMRC copia solamente la memoria globale in un buffer della memoria globale e viceversa, usando il meccanismo del DMA locale globale. Durante questa fase, gli IOP credono che il TMRC di sostituzione sia offline. Il TMRC di sostituzione restituisce uno stato valido per tutti gli RSB e i TMRC letti e scritti. Nella seconda fase, il TMRC di sostituzione rimane in una memoria di sola scrittura. I processori bloccano il RIOB, copiano un pacchetto dalla memoria globale a quella locale e viceversa usando il meccanismo del DMA. Poi il RIOB è sbloccato e il processo è ripetuto finché tutta la memoria non è stata ripristinata. Durante l'intero processo di rinascita dello IOP, tutti i TMRC accettano scritture dalle CPC e dagli IOP. Questo garantisce che le memorie sono consistenti dopo che il processo di copia è completato. Il sistema operativo standard UNIX assume che lo hardware e il software siano perfetti. Un fallimento nello hardware o nel kernel produrrà una perdita di servizio incondizionata (un "panico"). Dato questo insieme di vincoli autoimposti, è stato adottato un modello di fault recovery basato sul forward recovery. Il kernel usa check di consistenza come meccanismo di faul detection attraverso circa 1000 ASSERT. Un ASSERT è semplicemente una macro che assicura che un'espressione è vera. Nel kernel standard di UNIX, il fallimento di un ASSERT provoca un panico del sistema. Il ripristino da un fallimento di un ASSERT usa una routine di forward recovery specifica per le asserzioni. Queste routine di ripristino sono guidate da una struttura dati di revisione delle

routine. La struttura dati di revisione delle routine determina la validità e la consistenza delle varie strutture dati.

Una implementazione di una versione di UNIX che è stata dimostrata corretta deve essere ancora prodotta. Un meccanismo affidabile di panico fu implementato con un grande incremento della probabilità che le varie strutture dati residenti nel disco fossero consistenti dopo un reboot da un guasto del sistema operativo non recuperabile. Per realizzare questo, un meccanismo hardware di protezione dalla scrittura è usato per bloccare un numero di strutture dati critiche del kernel durante la procedura di gestione dello stato di panico. Inoltre, sono eseguiti una serie di test di consistenza sulle strutture dati del kernel e solo le strutture dati che passano i vari test di validità sono successivamente utilizzate. I blocchi sporchi nel buffer sono scritti nel disco usando una versione polled del drive del disco. Questo assicura che una minima quantità della struttura del sistema è usata per compiere l'operazione di scrittura. Infine, un'immagine del kernel è scritta nel disco usando delle PROM. L'esperienza mostra che questa procedura incrementa molto la probabilità che il file system sia in uno stato consistente dopo il reboot.

Tecniche di manutenzione

Le tecniche per la manutenzione dello hardware si sono evolute notevolmente negli ultimi dieci anni; c'è stato il tentativo di incrementare la partecipazione degli utenti alla manutenzione, rendendo più facile la risoluzione dei problemi hardware, basandosi inizialmente su test diagnostici online per isolare rapidamente le cause di un fallimento. Nei sistemi successivi si è andati verso la capacità di individuare i fallimenti automaticamente, cioè senza effettuare test, di analizzarli, di riportarli e di tenere nota delle eventuali riparazioni.

Nel sistema NonStop I erano disponibili per ogni processore solo un insieme di spie per comunicare le informazioni di errore e di switch per resettare e ricaricare il processore. Nel 1979, è stata introdotta l'interfaccia diagnostica Diaglink per permettere l'accesso al sistema da luoghi remoti attraverso un modem asincrono. Con Diaglink, il personale a disposizione degli utenti poteva esaminare per via remota il sistema dell'utente per ottenerne lo stato e per eseguire delle procedure diagnostiche per un debug remoto di basso livello.

5.1 Una prima evoluzione: il processore OSP

Il sistema NonStop II rimpiazzò Diaglink con un Operations and Service Processor (OSP). Lo OSP è un microcomputer che comunica con tutti i microprocessori, affiancato da una consolle di manutenzione e fornisce oltre alle capacità del Diaglink, altre caratteristiche addizionali per rilevare i fallimenti, per operare con il sistema in modo remoto ed anche delle operazioni per ottenere lo stato di ogni processore.

Lo OSP comunica con un trasmettitore di dati diagnostici (DDT) incluso come parte di ogni modulo dei processori. Oltre alla comunicazione, il DDT monitorizza lo stato dell'interfaccia del Diaglink, il processore del canale di I/O, la memoria e lo IPU insieme anche a tutti i data path interni. La comunicazione permette all'operatore di diagnosticare facilmente problemi hardware e software: per esempio abilita l'operatore a mettere il processore in modalità single step così da poter controllare il contenuto dei registri passo a passo. Lo OSP include un modem interno per la connessione ad un terminale remoto od ad un altro OSP: questa permette ad un operatore di diagnosticare e possibilmente anche correggere i problemi in modo remoto. Un addetto può per via remota, ad esempio, lanciare delle routine diagnostiche residenti su un OSP locale; alternativamente può scaricare i dati diagnostici dallo OSP remoto a quello locale, resettare in modo remoto e ricaricare i processori visualizzando le informazioni di errore. Nel sistema TXP, lo OSP fu potenziato per includere un modem asincrono, della diagnostica migliorata, più possibilità di operazioni remote e capacità di supporto remoto addizionali.

5.2 Il sistema diagnostico e di manutenzione del Tandem

Un aspetto importante della riduzione dei costi nel VLX è stato il miglioramento delle capacità diagnostiche e di manutenzione attraverso il Tandem Maintenance and Diagnostic System (TMDS) che sostituì lo OSP. Il TMDS ha permesso l'eliminazione del pannello frontale di spie e di switch, semplificando moltissimo l'attività di manutenzione. A differenza dei suoi predecessori, il TMDS opera online senza richiedere risorse significative di sistema ed ha fornito una interfaccia uniforme a molti sottosistemi diagnostici.

Dai primi VLX, la strategia di manutenzione si è evoluta verso il monitoraggio real time di componenti fino ad includere una analisi online automatica dei guasti ed una chiamata automatica per il supporto online dai centri di assistenza remoti. Il TDMS gira sotto il sistema operativo Guardian ed è distribuito automaticamente a tutti i clienti in quanto è compatibile anche con i sistemi NonStop e TLX. Attraverso una diffusa strumentazione, un sottosistema tollerante ai guasti

interno di diagnostica e manutenzione controlla continuamente i processori del sistema, le sorgenti di alimentazione ed i cabinet enviroment, cioè gli armadietti in cui si trova il sistema. Quando il sistema operativo Guardian o il processo di I/O individuano un cambio di stato o un evento irregolare, questo scrive l'indicazione di un evento in un registro degli eventi sul disco: il TMDS esamina ogni indicazione di evento e se sembra opportuno un ulteriore studio, il TMDS attiva un modulo conosciuto come l'analizzatore automatico di guasti. Dunque il TMDS permette sia un test attivo dei componenti, sia un'analisi dei guasti basata su sintomi.

5.2.1 L'analizzatore dei guasti

L'analizzatore dei guasti del TMDS solleva il cliente dal bisogno di una profonda conoscenza dello hardware, dei codici dello stato o di specifici valori di errore. Il TMDS usa algoritmi basati sulla regola "if then" per valutare gli eventi che non si addicono ad una certa base di conoscenza, per automatizzare molte individuazioni degli errori, per l'interpretazione e per riferire i compiti precedentemente richiesti dalla consolle di un operatore. La base di conoscenza contiene un insieme di affermazioni accettabili e dei fattori ambientali. Se l'analizzatore dei guasti trova un evento che cade in un range accettabile, il TMDS salva l'analisi del guasto in un registro locale che non è altro che un catalogo degli eventi del sistema che può aiutare nella risoluzione di problemi futuri. Comunque, se l'analizzatore dei guasti individua un evento che suggerisce un problema potenziale o già attivo, il TMDS trasmette un segnale ad un processore dei servizi tollerante ai guasti chiamato la interfaccia remota di manutenzione (RMI).

5.2.2 L'interfaccia remota di manutenzione RMI

La RMI è composta da dei processori duali Motorola 68000 based che comunicano l'un l'altro e con gli altri sottosistemi attraverso il bus di manutenzione seriale ad un bit. I processori, i controller FOX ed i monitor degli alimentatori sono tutti connessi ai bus di manutenzione. Lo RMI supporta tutte le funzioni del vecchi OSP, ma è una unità più compatta – due schede di circuiti residenti in uno degli armadietti (VLX e Cyclone) o in una parte dello MFC (CLX). Attraverso un protocollo sincrono, uno speciale processo di comunicazione e una richiesta di

password, lo RMI riduce anche di molto il rischio di accessi di utenti non autorizzati al sistema attraverso la parte diagnostica. Quando riceve un segnale che indica un problema, lo RMI mette in allerta l'operatore che si trova nel luogo, nel CLX, nel VLX o nel sistema Cyclone e opzionalmente telefona al Centro di Assistenza Nazionale del Tandem (TNSC). Attraverso lo RMI, l'operatore presente sul luogo o l'analista remoto e gli ingegneri al TNSC possono rivedere il registro degli eventi, lanciare programmi diagnostici, fare test di componenti e isolare e diagnosticare il problema. Nello equipaggiamento più nuovo, queste azioni includono la individuazione di inserimenti fuori specifica, il malfunzionamento dei dischi o dei controller e le ventole di raffreddamento guaste. Il TNSC usa anche dei programmi diagnostici per testare le sorgenti di alimentazione, i clock e le batterie. Se necessario, il TNSC può inviare un ingegnere in loco per risolvere il problema, eventualmente operando le opportune sostituzioni.

5.2.3 Alcuni aspetti avanzati

IL TMDS è stato evoluto con l'inserimento di nuove caratteristiche: una di queste è il metodo built in self test (BIST) che usa dei vettori di test pseudocasuali e lo scan path design. Sul CLX, il test pseudocasuale copre diverse funzioni abituali dello IC, la logica commerciale MSI, il vettore di RAM statica e le loro interconnessioni. Anche il BIST esegue un test funzionale della memoria RAM dinamica principale e della sua logica di controllo. Il test è controllato da un software di manutenzione dei processori, semplificando così lo hardware della scheda del processore dedicato al BIST.

Le tecniche di monitoraggio dell'alimentazione e dell'ambiente sono state migliorate significativamente nel sistema Cyclone: oltre alla verifica di un numero maggiore di componenti, i sensori sono interrogati molto più frequentemente e la maggior parte di essi sono replicati per permettere la differenziazione tra un componente guasto ed un sensore guasto. In aggiunta, il sottosistema di manutenzione del Cyclone può individuare la presenza o l'assenza fisica di molti componenti, come i cavi, le sorgenti di alimentazione, le ventole ed altri ancora. I componenti guasti possono essere facilmente identificati e sostituiti in modo affidabile poiché il sottosistema di manutenzione conosce tutti gli indirizzi logici e le posizioni fisiche di ciascuno di loro.

5.2.4 Un esempio di funzionamento

Il TMDS permette anche agli analisti e agli ingegneri di eseguire programmi diagnostici online per identificare i problemi che l'analizzatore dei guasti non copre. Infatti, molti programmi diagnostici possono essere lanciati mentre il dispositivo che si sta studiando sta funzionando. Nel caso peggiore, c'è bisogno di sospendere solo la attività del dispositivo malfunzionante mentre usando il precedente sistema diagnostico si doveva sospendere il dispositivo e il suo processore di controllo. In ogni caso, il test con il TMDS influenza solo minimamente le prestazioni del sistema: l'elaborazione continua senza essere ostacolata dai test diagnostici a meno che non debba essere esaminato il processore stesso. I seguenti passi illustrano come opera il TMDS se un processo di I/O del nastro individua un evento di errore che coinvolge un'unità nastro.

1. Il processo di I/O del nastro crea subito un evento di errore e lo manda al registro degli eventi del TMDS.
2. Il TMDS invia un segnale all'analizzatore dei guasti.
3. L'analizzatore dei guasti localizza l'errore in una particolare scheda del controller.
4. L'analizzatore dei guasti scrive delle informazioni addizionali sul registro degli eventi, specificando la FRU probabile, l'indirizzo del controller e il codice di errore finale. Tutte queste azioni sono eseguite in qualche secondo.
5. Dopo aver completato l'analisi, il TMDS telefona al TNSC.
6. Il TNSC telefona al TMDS per verificare l'analisi.
7. Il TNSC invia un operatore per sostituire la scheda del controller.
8. Il TMDS regista la sostituzione nel registro degli eventi.

Il registro degli eventi del TMDS e il resoconto generato dall'intervento remoto sono archiviati in un database di supporto centralizzato: questo contiene la storia delle richieste di servizio, delle diagnosi e delle azioni di supporto per lo hardware e per il software. Questo è utile in quanto degli esperti periodicamente analizzano il database per poter migliorare la manutenzione del sistema.

Il Sistema Operativo Guardian

Nella precedente discussione dell'evoluzione del sistema Tandem, il lettore attento avrà visto che non si è parlato affatto di moduli hardware tolleranti ai guasti. Infatti, uno dei principali criteri di progetto per lo hardware del Tandem è quello di renderlo *fault intolerant*. Qualsiasi modulo hardware che non funzioni correttamente dovrebbe individuare il problema e più velocemente possibile sospendere l'attività. Ci sono un po' di eccezioni a questa regola, come la memoria principale con un codice error correcting, ma il progetto fondamentale del Tandem è quello di connettere hardware fail fast con del software tollerante ai guasti.

La tolleranza ai guasti normalmente implica la ridondanza dello hardware. Mentre questo è vero per il sistema Tandem, il costo della rete addizionale verso gli utenti è stato tenuto particolarmente basso a causa delle molte caratteristiche innovative del sistema Tandem. In molti casi, ognuno dei moduli ridondanti esegue del lavoro utile seguendo la propria direzione ed eccetto quando si verifica un guasto, contribuisce alla funzione del sistema. I moduli guasti possono essere sostituiti mentre il sistema è attivo. L'effetto sulla rete di un fallimento dello hardware è, al peggio, un breve periodo di prestazioni leggermente degradate. Un sistema con una quantità normale di capacità in eccesso sopravviverà ad un fallimento senza produrre un effetto evidente. La chiave per una tolleranza ai guasti senza alcuna forma di ridondanza inutilizzata è il sistema operativo Guardian. Le seguenti sezioni descrivono i vari componenti di Guardian, dal kernel (gestione della memoria, dei processi e del sistema dei messaggi) al gestore delle transazioni, al supporto per la rete fino al NonStop Structured Query Language (SQL). Ognuno dà un importante contributo non solo alla funzionalità ma anche alla tolleranza ai guasti del sistema Tandem.

La tolleranza ai guasti sarebbe di poca importanza senza l'integrità dei dati: il commercio richiede un'accurata registrazione dei dati e un database inconsistente è spesso peggio dell'assenza del database stesso. Comunque, un organismo commerciale che dipende dal suo sistema di computer dovrebbe essere in grado di potenziare tale sistema velocemente almeno quanto i suoi affari.

6.1 Introduzione al sistema operativo Guardian

Il sistema operativo Guardian ha due responsabilità primarie: (*a*) fornire dei servizi generali per i suoi clienti e per i processi e in particolare fornire un mezzo efficiente ed affidabile di comunicazione tra di loro; (*b*) mantenere una vista consistente del sistema mentre permette ad ogni processore di eseguire indipendentemente i propri compiti in piena autonomia. Una differenza fondamentale tra Guardian e gli altri sistemi è il livello di integrazione molto alto del software. Benché ci sia l'usuale stratificazione della funzionalità del software, questi strati sono legati insieme in modo stretto e interdipendenti. Questo approccio ha i suoi costi, ma l'efficienza risultante, accoppiata con un alto livello di funzionalità, è unica nell'industria dei computer. Ci sono molti componenti software che contribuiscono alla tolleranza ai guasti, alla integrità dei dati, alla espandibilità ed alla funzionalità di base del sistema Tandem. In questa sezione, daremo uno sguardo generale agli elementi di Guardian che differenziano Tandem dagli altri sistemi.

Il kernel include il solito supporto per la gestione dei processi, della memoria, della comunicazione tra i processi, dei nomi, del tempo, dello I/O periferico, della sincronizzazione tra processi e del debugging. In aggiunta, il kernel individua i fallimenti di ogni processore, del bus tra i processori e del canale di I/O ed esegue l'appropriato ripristino. Sono usate diverse tecniche innovative per sincronizzare i processi indipendenti e per fornire una vista consistente del tempo e dello stato del sistema, nonostante i fallimenti e i ritardi di comunicazione. Infine, il kernel supporta la gestione delle coppie di processi che rappresentano la chiave di volta della tolleranza ai guasti hardware e software.

Il file system nasconde la natura distribuita del sistema dai processi delle applicazioni e presenta una vista convenzionale dei dispositivi periferici di I/O. La comunicazione con i dispositivi di I/O e tra i processi è compiuta senza tenere

conto della locazione della risorsa, sia che si trovi nello stesso processore, sia che si trovi in un altro processore dello stesso sistema o in un processo in un sistema remoto. Il file system fornisce anche un meccanismo di checkpoint e di retry per nascondere gli effetti dei fallimenti hardware e software dai processi delle applicazioni.

I processi di I/O gestiscono i dispositivi periferici e reagiscono ai fallimenti dei componenti dirigendo l'accesso sui percorsi di lavoro e sui dispositivi e poi notificando agli operatori e agli ingegneri a disposizione della clientela il bisogno di riparare o di sostituire il componente guasto. I processi di I/O ricevono dei messaggi dai processi delle applicazioni attraverso il file system ed eseguono le operazioni richieste sul dispositivo fisico. Dal punto di vista del programmatore delle applicazioni, il processo di I/O e il dispositivo che gestisce sono indistinguibili. Ci sono dozzine di processi di I/O, ognuno progettato per gestire una particolare classe di dispositivi.

Il processo che gestisce il disco è probabilmente il componente singolo più importante del sistema. Garantisce l'integrità dei dati e fornisce un accesso affidabile ai dati nonostante i fallimenti del processore, del canale, del controller o del mezzo di comunicazione. Supporta efficacemente i dischi mirrored, facendo un buon uso dei percorsi di accesso duale ai dati. Supporta quattro tipi di strutture di file, così come le tabelle SQL, il partizionamento dei file, gli indici alternati, la sicurezza dei dati e il cacheing della memoria principale sia in lettura che in scrittura. Supporta la gestione totale delle transazioni, includendo i due protocolli *two fase locking* e *two fase commit*. Infine, ma non di minore importanza, può eseguire quelle parti delle query SQL che richiedono che i dati siano locali ad un disco, riducendo molto il traffico dei messaggi per molte applicazioni.

La procedura di facilitazione della gestione delle transazioni (TMF) coordina l'elaborazione degli accessi al disco e gli aggiornamenti, garantendo la atomicità, la consistenza, l'isolamento e la persistenza. Un'applicazione può, in maniera molto semplice, richiedere aggiornamenti multipli del database in molte locazioni fisiche, anche migliaia di miglia distanti, ed essere sicura che tutti o nessuno di loro saranno eseguiti; durante una transazione, le altre applicazioni avranno sempre una vista consistente del database, non potendo vedere solo una parte degli aggiornamenti e non gli altri. Il TMF protegge il database dai fallimenti totali del mezzo di comunicazione, includendo la perdita di una coppia di dischi mir-

rored, attraverso la tecnica del dumping online e del roll forward del registro delle transazioni. Il TMF supporta anche la facilitazione della duplicazione remota del database, la quale può velocemente recuperare la perdita di un'intera facilitazione della gestione delle transazioni.

Il monitor di elaborazione delle transazioni fornisce un metodo flessibile per la gestione delle applicazioni degli utenti in un sistema distribuito: questo distribuisce automaticamente i processi delle applicazioni, detti server, ai processori disponibili e, nel caso di un fallimento di un processore, ridistribuisce le applicazioni ai processori rimanenti. Qualsiasi computazione che è persa o compromessa dal fallimento, è riavviata automaticamente dopo essere stata riportata al suo stato iniziale: questo procedimento viene detto *roll back*. La programmazione fatta dagli utenti è diretta e non richiede l'esecuzione di operazioni speciali per raggiungere la tolleranza ai guasti.

Il processo di controllo e di gestione della rete Expand gestisce la comunicazione tra il sistema e la rete di altri sistemi Tandem. L'utente di un nodo della rete vede il resto come un'estensione del sistema locale. Le richieste per una autonomia locale potrebbe imporre l'accesso alle barriere e i ritardi di comunicazione potrebbero imporre dei ritardi nelle prestazioni; d'altra parte, è facile gestire le applicazioni e i database sia distribuiti sia locali.

6.2 La gestione del processore

L'inizializzazione del sistema avviene quando un processore viene *caricato a freddo* da un disco ad esso attaccato; qualsiasi processore può essere usato per questa operazione finché è connesso al disco con una copia del file immagine del sistema operativo. Questo file immagine contiene il codice del kernel e i processi del sistema che sono lanciati automaticamente quando viene caricato ogni processore. Una volta che un processore è stato caricato, viene usato per caricare gli altri processori attraverso lo IPB. Tutti i processori ad esclusione del primo possono essere caricati in parallelo. Quando un processore è guasto o quando è rimosso per la manutenzione, può essere ricaricato da un altro processore. Non c'è nessuna differenza sostanziale tra il caricamento iniziale di un processore e il caricamento successivo eseguito dopo che è stato riparato. Un'operazione di ricaricamento di

un processore non interferisce con le altre operazioni delle applicazioni dei processi. Ogni processore riceve una copia identica del kernel ed altro software che si trova a livello di sistema, ma la sua configurazione dipendente dai dispositivi periferici che gli sono connessi. Ogni processore lancerà gli appropriati processi di I/O per gestire i dispositivi collegati.

6.3 La sincronizzazione dei processori

Una volta che il software e la configurazione di un processore sono stati caricati, si passa attraverso una fase di sincronizzazione con gli altri processori. In breve, questa sincronizzazione consiste nel trasmettere una copia consistente dello stato del sistema ad ogni processore: una volta che questo è stato eseguito, il processore diventa un'entità pienamente indipendente e non c'è qundi nessun processore master. Benché il sistema di computer Tandem sia stato progettato per tollerare ogni singolo guasto sia hardware che software, qualche volta possono avvenire dei fallimenti multipli, anche attraverso errori multipli dello hardware - improbabili -, errori software - probabili -, errori delle operazioni - più probabili - o durante i periodi di manutenzione. Anche quando avvengono dei guasti multipli, solo una parte del sistema diviene inutilizzabile finché non sono riparati e reintegrati nel sistema. Vengono forniti due meccanismi diversi per operare con i guasti multipli: la sincronizzazione dei processori e le transazioni. In aggiunta ad un semplice algoritmo per individuare i fallimenti del processore, Guardian possiede anche tre algoritmi più complessi per assicurare che tutti i processori di un sistema abbiano una vista in accordo della configurazione del processore, dei dati replicati e del tempo. Ognuno di questi algoritmi richiede un minimo di comunicazione ed è sufficientemente robusto senza richiedere i costi e la complessità della soluzione del problema dei Generali Bizantini.

6.3.1 Il protocollo I'm-Alive

Una volta che due o più processori sono stati caricati, devono continuamente testarsi l'un l'altro. Poiché i processori sono stati progettati fail fast, rispondono ad un errore interno interrompendo completamente l'attività e rifiutandosi di comu-

nicare con il mondo esterno finché non ricevono un segnale esplicito che sono pronti per essere ricaricati. Così è compito dei processori rimanenti individuare un fallimento attraverso l'assenza di una qualunque risposta dal processore guasto.

Usando il protocollo "I'm-Alive", ogni processore trasmette un breve messaggio ad ognuno degli altri processori compreso se stesso, almeno una volta ogni secondo, attraverso uno dei due IPB: il messaggio a se stesso verifica se il bus di ingresso e quello di uscita stanno funzionando. Ogni due secondi, ogni processore esegue un test per vedere se ha ricevuto almeno un messaggio da ogni altro processore: ogni messaggio mancante implica che un processore non è in salute ed è probabilmente guasto. Il protocollo I'm-Alive viene illustrato in Figura 20.

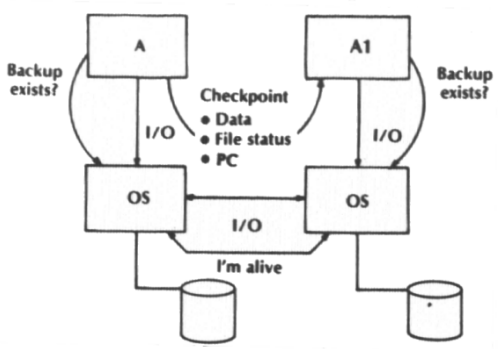


Figura 20 Il protocollo I'm-Alive.

Normalmente ogni processore dovrebbe immediatamente dichiarare che un processore è guasto e procedere a cancellare tutte le comunicazioni in sospese con quel processore. L'esperienza mostra che ci sono casi rari nei quali un processore potrebbe essere soltanto un po' in ritardo nell'inviare il suo messaggio I'm-Alive. Questa situazione di solito accade quando il ripristino da un fallimento di tensione o di altri errori di priorità alta si impadronisce momentaneamente un processore. Poiché gli intervalli del protocollo I'm-Alive non sono sincronizzati tra i processori, un tardivo I'm-Alive potrebbe derivare da un processore che è stato dichiarato guasto da alcuni processori ma non da altri. Un tale caso, chiamato *split brain*, potrebbe portare ad un buco di integrità nel database. Così è stato realizzato un algoritmo di raggruppamento che gestire questi casi con un piccolo smembramento finché è possibile. In pratica, viene data una seconda chance al processore lento. Ogni processore individua comunque che manca un messaggio I'm-Alive e tutti i processori, compreso quello sospetto se attivo, scambiano dei messaggi per deci-

dere quali processori non sono guasti. Dopo due round di messaggi broadcast, viene presa una decisione a cui partecipano tutti i processori.

6.3.2 La sincronizzazione temporale

Un sistema OLTP è progettato per registrare e controllare gli eventi del mondo reale nel momento in cui avvengono. Una parte importante dell'informazione elaborata è il tempo corrente. Per di più, è importante che la sequenza nella quale avvengono gli eventi possa essere ricostruita dal *timestamp* associato ad ogni evento. Benché sia chiaramente difficile coordinare i clock in un sistema largamente distribuito, i tentativi iniziali di sincronizzare il tempo in un sistema Tandem locale hanno dimostrato che anche questo non è un problema semplice. Nonostante non sia un gran problema portare i clock ad un certo numero di secondi l'uno dallo altro, la sincronizzazione di un sistema multiprocessore richiede che non può essere mandato nessun messaggio dal processore A, con tempo *t* di clock, che dovrebbe arrivare al processore B prima che il clock di B abbia raggiunto il tempo *t*. Infatti, se così fosse, il tempo scorrerebbe all'indietro. Un nuovo algoritmo [Nellen, 1985, 1986] permette di scambiare dei pacchetti aggiustati temporalmente, da un processore all'altro; ogni processore, non solo aggiusta il suo clock con il tempo medio di tutti gli altri processori, ma calcola anche l'errore relativo del suo clock rispetto alla media e lo aggiusta continuamente. Questo algoritmo assicura che il tempo medio non fluttui molto quando un processore fallisce ed è rimpiazzato da un processore con un differente velocità di clock. Questo provvedimento per un clock esterno può essere esteso all'algoritmo di sincronizzazione locale per i sistemi Tandem geograficamente distribuiti. I clock elettronici che monitorizzano un tempo standard diffuso a tutti, possono tenere sincronizzati i sistemi Tandem a meno del tempo che gli serve per comunicare l'un l'altro.

6.3.3 Il protocollo di aggiornamento globale

Una certa parte di informazione, osservando la tabella di controllo della destinazione descritta più avanti, è replicata e deve essere identica in ogni processore. Gli aggiornamenti dell'informazione replicata comunque, nascono da processi e processori multipli. La consistenza degli aggiornamenti dipende dall'*atomicità* che

richiede: che ogni aggiornamento sia completato in un tempo massimo; che siano aggiornate tutte le copie replicate o nessuna di loro; che tutti gli aggiornamenti avvengano in modo seriale. Nel sistema Tandem, l'atomicità degli aggiornamenti è garantita dal protocollo di aggiornamento globale. Tutti gli aggiornamenti sono eseguiti dal kernel di Guardian così che i processi ad alta priorità non possono ritardare il completamento di un aggiornamento in un certo tempo massimo. Tutti gli aggiornamenti devono essere prima mandati ad un processo locker che assicura che avvengano tutti in modo seriale e che ognuno di loro sia propagato a tutti i processi, anche se il processo che lo ha originato fallisce, includendo i fallimenti simultanei dei processi locker e di aggiornamento.

6.4 La gestione dei processi

Un processo è un'entità eseguibile indipendentemente che consiste principalmente di un codice di programma condivisibile, di memoria privata e di un vettore dello stato del processo. Il vettore di stato del processo include il program counter, i registri, i privilegi, la priorità e un timer in microsecondi che viene incrementato solo quando il processo è in esecuzione. Una volta che un processo è in esecuzione in un processore, esso rimane in quel processore finché non è terminato. Ogni processo nel sistema ha un unico identificatore o nome per mezzo del quale gli altri processi possono comunicargli su un'ampia rete. I messaggi sono indirizzati ad un processo attraverso il suo identificatore, e così lo identificatore fornisce un metodo tollerante ai guasti su ampia rete per indirizzare i processi.

Guardian mantiene una tabella di controllo della destinazione che è identica in tutti i processori del sistema. Questa tabella contiene i nomi dei processi con le loro locazioni, cioè il numero del processo e del processore così che i messaggi per un processo possono essere instradati in modo efficiente. Lo scheduling dei processi usa un meccanismo di priorità pura con *preemption*. Una cura considerevole è stata presa per evitare il problema dell'inversione di priorità nel quale un client a bassa priorità fa una richiesta ad un server ad alta priorità, promuovendo così il suo lavoro ad una priorità più alta. Questo problema è unico per i sistemi basati sui messaggi e deve essere risolto per fornire un meccanismo di scheduling con una priorità globale.

6.4.1 Le coppie di processi

La chiave per una tolleranza ai guasti software ed hardware è costituita dalle coppie di processi. Una coppia di processi è formata da un processo *primario* che esegue tutto il lavoro e da un processo di *backup* che è passivo ma che è pronto a sostituire quello primario quando questo fallisce (vedi Figura 21). Questo arrangiamento è simile ad avere un processore in standby o un sistema di computer eccetto che per il fatto che se viene arrangiato in modo opportuno i costi del processo di backup sono molto inferiori a quelli del processo primario.

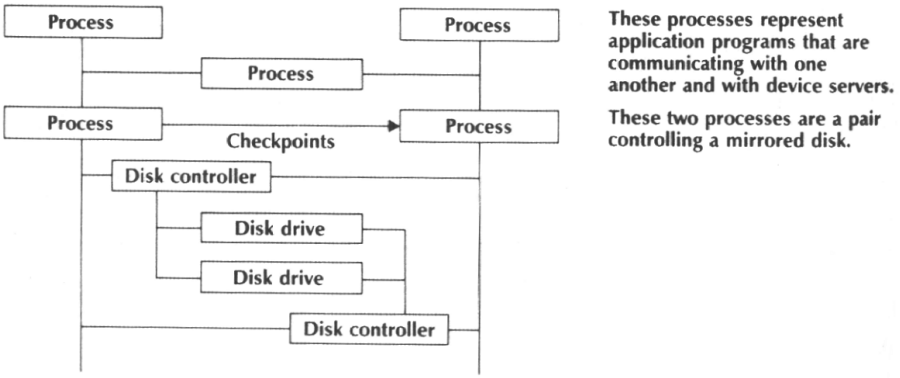


Figura 21 Diagramma della struttura del sistema.

Una coppia tipica di processi parte nello modo in cui partono i processi ordinari: come un singolo processo eseguito in un singolo processore. Il processo che notifica al sistema operativo che vorrebbe creare un clone di se stesso, usa lo stesso file di programma ma in un altro processore. Anche il secondo processo parte in modo ordinario, ma con due piccole differenze:

1. il secondo processo riceve un messaggio iniziale che lo informa che è un processo di backup; il processo si mette allora in condizione passiva, ed accetta messaggi solo dal processo primario o da Guardian per notificare che o il processo primario o il suo processore sono falliti;
2. il processo primario e quello secondario condividono un singolo nome e per ogni nome la tabella di controllo della destinazione registra i processi primari e di backup; tutte le comunicazioni alla coppia di processi sono inviate al processo primario.

Mentre viene eseguito il processo primario, quello di backup sta in una condizione passiva. Quando si giunge in un punto critico, il processo primario manda un messaggio di *checkpoint* a quello di backup.

6.4.2 I messaggi di checkpoint

I messaggi di checkpoint hanno due forme differenti ed una coppia di processi normalmente ne userà una ma non entrambe. Per il software delle applicazioni viene fornita una forma semplice di checkpoint. Il checkpoint copia lo stato del processo selezionato, dopo aver scelto attentamente i punti di rilevamento, dal processo primario a quello secondario. L'aggiornamento dello stato del processo di backup viene eseguito solamente dal software del sistema che è al comando del processo primario; il processo di backup è completamente passivo. In un checkpoint, di solito immediatamente prima o dopo alcune importanti operazioni di I/O, il processo primario e quello secondario sono funzionalmente identici. Se il primario fallisce, quello di backup entra in esecuzione a partire dal punto in cui era avvenuto l'ultimo checkpoint. Per la maggior parte del software di sistema, come i processi di I/O, i messaggi di checkpoint contengono degli aggiornamenti funzionali per lo stato del processo. Il processo di backup deve interpretare questi messaggi ed eseguire gli appropriati aggiornamenti alle sue proprie strutture dati locali; questo viene chiamato un *active backup*. Benché il codice del programma sia identico, il contenuto della memoria può essere totalmente differente. Un tipico checkpoint potrebbe indicare che un file è stato aperto o chiuso ma, nel caso del processo del disco, anche la maggior parte degli aggiornamenti per i file importanti potrebbe coinvolgere un checkpoint.

6.4.3 I vantaggi di un backup attivo

L'approccio del backup attivo è più complicato ma ha diversi vantaggi. Vengono trasmessi meno dati in ogni checkpoint e l'informazione che il backup può ottenere da altre sorgenti, come ad esempio un file su disco, non avrà bisogno di subire il checkpoint. Per quanto riguarda la tolleranza ai guasti software, il backup attivo è migliore perché il backup deve gestire il suo proprio stato ed è meno probabile che gli errori nel processo primario contaminino quello di backup. Poiché la coppia di processi condivide un solo nome, appare a tutti gli altri processi come un singolo processo. Quando gli viene mandato un messaggio, il sistema dei messaggi consulta automaticamente la tabella di controllo della destinazione per determinare quale processo è il primario e manda il messaggio a quella locazione. Quan-

do il processo primario fallisce per qualche ragione, quello di backup diviene il primario, la tabella del controllo della destinazione viene cambiata e tutti i messaggi sono diretti al nuovo processo primario. Il nuovo processo primario ha già ricevuto i checkpoint che descrivono i creatori correnti del processo, così che quei creatori non devono fare niente per ristabilire la comunicazione. Ci sono delle potenziali race condition nelle quali una coppia di processi server ha eseguito delle richieste e non ha replicato al client perché è avvenuto un fallimento. Quando la richiesta in uscita viene rimandata al nuovo processo primario, esso potrebbe eseguire la stessa operazione una seconda volta, probabilmente causando una inconsistenza. Questo problema viene eliminato associando una sequenza di numeri con ogni richiesta e facendo in modo che il nuovo processo primario faccia delle risposte duplicate alle richieste già elaborate. Problemi simili potrebbero avvenire se una coppia di processi client eseguisse un checkpoint e poi facesse una richiesta ad un server. Se avvenisse un fallimento, il client di backup lo rileverebbe ed eseguirebbe delle richieste duplicate al server. Queste richieste sono gestite usando lo stesso schema di numerazione della sequenza. Durante le normali operazioni, i numeri della sequenza sono usati per l'eliminazione dei duplicati e per la individuazione dei messaggi persi nel caso di errori di trasmissione.

6.5 La gestione della memoria

Ogni processo ha uno spazio virtuale di indirizzi che contiene il suo programma e uno stack dei dati del processo. Un programma è composto da del *codice utente* e da una *libreria utente* opzionale. La libreria dell'utente si trova in un file del codice oggetto che può essere condiviso dai processi che eseguono dei file di programma di codice utente differenti. Tutti i processi in un processore che eseguono lo stesso programma condividono la memoria per il codice oggetto. Lo stack dei dati è privato ad un processo e non può essere condiviso. I processi possono allocare dei segmenti di memoria estesa addizionali che hanno una taglia massima di 127.5 MB ciascuno. Un processo può accedere concorrentemente al suo stack dei dati ed ad un segmento di memoria estesa. Se sono allocati dei segmenti multipli di memoria estesa, il processo deve esplicitamente richiedere che un particolare segmento sarà usato quando è richiesto.

I dati nei segmenti di memoria estesa possono essere condivisi con gli altri processi in due modi differenti:

1. un segmento di sola lettura è un modo di accedere al contenuto di un file come se fosse in memoria principale, usando la memoria virtuale per caricare la informazione richiesta. Tali segmenti possono essere condivisi tra tutti i processi in tutti i processori, anche se esistono delle copie multiple dei dati in dei processori differenti;
2. un segmento di lettura scrittura può essere condiviso solo dai processi che si trovano in un singolo processore, in quanto sarebbe impraticabile aggiornare copie multiple in processori differenti.

La condivisione dei dati di lettura e scrittura tra i processi delle applicazione degli utenti viene scoraggiata, così che viene mantenuto il contenimento dei guasti ed il carico del sistema può essere distribuito dai processi in cui è stato messo ai processori inattivi. Guardian non fornisce il controllo della concorrenza tra i processi necessario per la coordinazione degli aggiornamenti nella memoria condivisa se non attraverso i messaggi. Il gestore della memoria di Guardian supporta la memoria virtuale usando la paginazione su richiesta.

6.6 Guardian: un sistema operativo basato su messaggi

La prima responsabilità del sistema operativo richiede che ogni processore stabilisca e mantenga una comunicazione con gli altri processori del sistema. La continua disponibilità dello IPB è un presupposto fondamentale, poiché i processori devono coordinare molte operazioni e notificare l'un l'altro i cambi nello stato del sistema. Se due processori non sono capaci di comunicare l'un l'altro per un certo periodo di tempo, è probabile che abbiano delle viste inconsistenti dello stato del sistema; uno dei due deve essere riavviato. Così un IPB duplicato e tollerante ai guasti è una richiesta importante. Eccetto per le funzioni di basso livello di spedizione dei processi, per la trasmissione dei messaggi e per l'elaborazione degli interrupt, tutto il lavoro nel sistema è gestito da uno o più processi. I processi del sistema gestiscono la creazione e la terminazione dei processi, la memoria virtuale, la sicurezza, la misura delle prestazioni, la gestione degli eventi, la diagnostica e il debugging. I processi di I/O gestiscono l'accesso ai dispositivi periferici

e sono responsabili di occuparsi dei componenti guasti. I processi delle applicazioni e delle utility dirigono le operazioni del sistema verso alcuni compiti utili.

I messaggi sono il metodo primario di interazione tra i processi: alle applicazioni il sistema sembra costituito da un solo processore convenzionale programmato con dei linguaggi di programmazione convenzionali come il Pascal, il C e il Fortan. I processi interagiscono usando un protocollo client server tipico dei protocolli di chiamata delle procedure remote che sono comuni nelle workstation server LAN oggi. Questa architettura client server è ben accetta oggi, ma 15 anni fa era considerata insolita.

La tolleranza ai guasti viene raggiunta attraverso la duplicazione dello hardware e del software. L'accesso ai dispositivi di I/O è fornito da coppie di processi formate da un processo primario e da un processo di backup. Il processo primario deve inviare le informazioni sullo stato al processo di backup che può così rilevare se avviene un fallimento. Le richieste a questi dispositivi sono inviate usando il nome del processo logico così che la richiesta è inviata sempre al processo primario corrente. Il risultato è un insieme di primitive e di protocolli che permettono il ripristino e l'elaborazione continuata anche in presenza di fallimenti del bus, del processore, del controller di I/O o dei dispositivi di I/O. Inoltre queste primitive fornisco l'accesso a tutte le risorse del sistema da ogni processo del sistema.

6.6.1 La gestione dei messaggi

Ora che è stato trattato il problema base di mantenere la consistenza tra i processori distribuiti, possiamo addentrarci nell'esaminare le prestazioni del lavoro utile. Come in tutti i sistemi moderni, Guardian supporta l'esecuzione concorrente di molti processi indipendenti. Quello che distingue Guardian, è la sua forte dipendenza dai messaggi per coordinare le operazioni dei processi. I messaggi sono essenziali per le operazioni del software di sistema e per le applicazioni dell'utente. Sia lo hardware sia il software sono stati ottimizzati per facilitare l'invio dei messaggi tra i processori e tra i processi. La forte dipendenza dai messaggi, preferita ad altri meccanismi di comunicazione e di sincronizzazione, è stata molto importante nel progetto di un sistema che è distribuito e facilmente espandibile. Le applicazioni degli utenti finali possono facilmente crescere aggiungendo semplice-

mente più processori, più dischi ed altre periferiche, senza modificare il software o ricaricare il sistema.

Data questa struttura basata sui messaggi, le applicazioni sono inconsapevoli della configurazione fisica del sistema. Un'applicazione accede direttamente alle periferiche connesse in modo locale e in modo remoto, esattamente nello stesso modo; i messaggi sono scambiati con il gestore delle periferiche; anche il processo di I/O non sa se il richiedente si trova nello stesso processore o in un altro. L'efficiente meccanismo dei messaggi è stato anche un elemento chiave nello implementare la tolleranza ai guasti. Il software a livello di sistema usa i messaggi per fare un checkpoint dell'informazione critica per l'integrità dei dati e per garantire la continuazione delle operazioni durante un fallimento. Le applicazioni sono di solito inconsapevoli che è avvenuto un fallimento in quanto i messaggi possono essere automaticamente reinstradati da un componente guasto ad uno funzionale.

6.6.2 I messaggi tra i processi

Il protocollo base del sistema dei messaggi segue il modello requestor server. Un processo manda un messaggio ad un altro processo insieme ad una richiesta e aspetta la risposta. Il secondo processo controlla la richiesta e risponde con un risultato come illustrato in Figura 22. Tutto questo viene considerato un singolo messaggio e può essere visto come una procedura di chiamata remota. Naturalmente, questo meccanismo è adeguato anche per semplici datagrammi o per trasferire un nucleo di dati, fino a 60 KB per messaggio, in entrambe le direzioni. I processi multi threaded requestor possono avere molti messaggi in sospesi negli insiemi dei processi server. I processi multi threaded server possono accettare richieste multiple e rispondere ad ognuna in un certo ordine desiderato. Qualsiasi processo può lavorare concorrentemente sia ad un cliente sia a ad un server. Per i processi delle applicazioni, l'accesso al sistema dei messaggi avviene attraverso il file system, che fornisce una certa protezione dall'abuso dei meccanismi privilegiati e semplifica l'interfaccia per inviare i messaggi. I processi del sistema generalmente usano direttamente le primitive del sistema dei messaggi, in quanto forniscono delle prestazioni migliori.

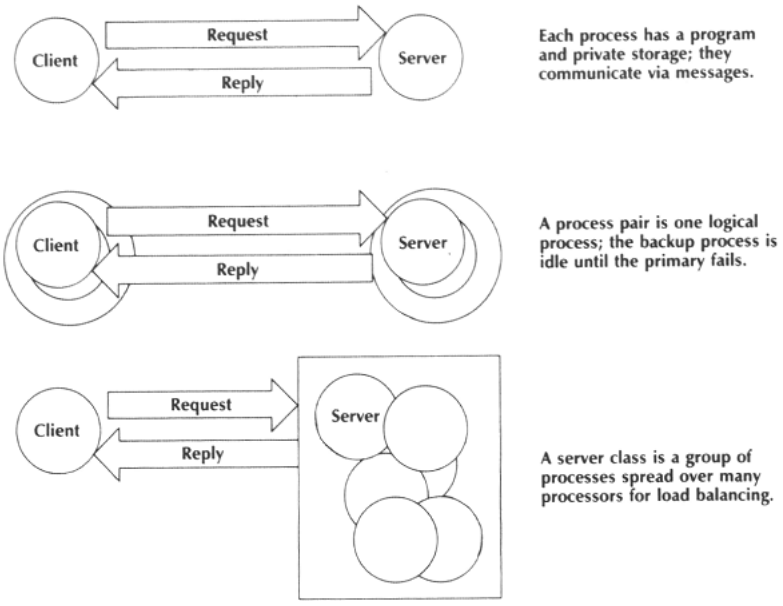


Figura 22 Modello requestor server nel sistema operativo Guardian.

Il più semplice uso dei messaggi riguarda i processi delle applicazioni che accedono ai dispositivi periferici; il programmatore non sa in generale che sono coinvolti i messaggi, così come il programma fa una semplice lettura o scrittura dei dati in un file. Se, per esempio, il file è un file su disco, allora le richieste di lettura/scrittura invieranno un messaggio al gestore del disco, conosciuto come il processo del disco. È semplice per un processo mascherarsi come un gestore dei dispositivi. Un esempio di questo è il processo di printing spooler: le applicazioni inviano dei messaggi in uscita ad uno spooler che immaganizza i dati per la successiva stampa; lo spooler poi invia i dati in uscita, di nuovo attraverso i messaggi, ad un reale processo fisico di stampa. Il programmatore delle applicazioni non ha bisogno di sapere se il sistema dei messaggi instrada i messaggi ad un processo spooler o direttamente ad un processo di I/O che controlla una stampante fisica; deve essere cambiato solo il nome del processo per scegliere il processo destinazione. Un uso dei messaggi più interessante è nello strutturare le applicazioni come una rete di processi comunicanti.

6.6.3 I messaggi e la tolleranza ai guasti

Nel sistema Tandem, i messaggi sono il meccanismo fondamentale per raggiungere la tolleranza ai guasti.

- Il *protocollo di comunicazione* per i bus tra i processori tollera ogni singolo errore del bus durante l'esecuzione di una primitiva del sistema dei messaggi. Un fallimento di comunicazione avverrà solo se il processo mandante o ricevente o entrambi fallissero. Gli errori del bus che avvengono durante una operazione del sistema dei messaggi sono corretti automaticamente senza coinvolgere i processi di comunicazione.
- Il *meccanismo della coppia di processi*, come descritto nella sezione successiva, fornisce un accesso tollerante ai guasti ai dispositivi periferici nonostante i fallimenti del processore, del canale o del software. Un messaggio di richiesta che è stato elaborato dal componente guasto, viene automaticamente rimandato al componente di backup; l'applicazione non è mai consapevole di quello che è avvenuto e non ha bisogno di prendere nessun provvedimento per esso.

I movimenti della memoria, piuttosto che per i bus tra i processori, sono usati per la comunicazione tra i processi nello stesso processore, ma non c'è un'apparente differenza con i processi di comunicazione. In aggiunta ai messaggi tra i processi, Guardian implementa anche dei semplici messaggi di controllo che servono per la comunicazione tra i kernel dei processori. I messaggi di controllo vengono usati come base per il protocollo del sistema dei messaggi e come un meccanismo non costoso per mantenere la sincronizzazione tra i processori.

Guardian e lo IPB sono molto ottimizzati per l'elaborazione dei messaggi tra i processi, specialmente per i messaggi brevi di 2 KB o meno. Un messaggio tra i processi in processori differenti è solo leggermente più costoso di un messaggio tra i processori. Infatti, un messaggio tra i processori di solito impiega un tempo più corto di un messaggio tra i processi, poiché i processi mandante e ricevente possono andare in parallelo.

Un vantaggio finale del sistema dei messaggi è il suo supporto trasparente per le reti short haul e long haul. Eccetto che per gli inevitabili ritardi di comunicazione, il client e il server non possono individuare nessuna differenza apparente nell'accedere ad un file del disco locale (o ad un processo o ad una stampante) o di un disco remoto. I protocolli del sistema dei messaggi e del file system sono esattamente gli stessi nel caso remoto e nel caso locale.

6.7 La tolleranza ai guasti software

I sistemi in cui la tolleranza ai guasti è basata solamente su meccanismi hardware possono essere progettati per un'alta affidabilità e per continuare a funzionare anche in presenza di fallimenti di componenti hardware. Sfortunatamente, un'alta percentuale dei fallimenti dei sistemi di computer è dovuta al software. A differenza di quando si lavora con componenti hardware, è possibile sviluppare un software perfetto, libero dai difetti e a prova di fallimento. È solo una questione di costi per il fabbricante e di disturbo per il cliente, che deve aspettare più a lungo prima che il prodotto software sia rilasciato. Nel mondo commerciale, la clientela chiede continuamente un flusso continuo di nuovo software e di miglioramenti a quello vecchio. Inoltre si chiede che questo sia fatto velocemente, più velocemente di una competizione, ed ad un prezzo ragionevole. I nuovi sistemi software sono inevitabilmente più funzionali e più complessi dei sistemi chi rimpiazzano. L'uso della programmazione strutturata e dei linguaggi di più alto livello non hanno eliminato gli errori software poiché hanno permesso la costruzione di programmi sempre più grandi e complicati. I metodi per migliorare la qualità del software, come l'ispezione del codice e le tecniche di testing strutturato, sono efficaci ma riducono soltanto il numero degli errori, non li eliminano del tutto. Quindi, in pratica, anche mettendo una cura particolare nel processo di sviluppo del software, i guasti software sono inevitabili. Infatti, come detto precedentemente, i fallimenti software sono tipicamente più comuni di quelli hardware.

La tolleranza ai guasti software porta indirettamente ad una migliore qualità del software e dell'integrità dei dati. Nel Tandem, i programmatori di sistema sono invogliati a fare numerosi test di consistenza e, se viene individuato un problema, a fermare il sistema: il software del sistema Tandem ha probabilmente una delle più alte densità di istruzioni per fermare i processori in ambito industriale. I programmatori di sistema sanno che, per la maggior parte di tutti i problemi di consistenza, i processi di backup continueranno a fornire il servizio all'utente. Questi test di consistenza hanno due effetti diretti:

1. Quando viene individuata una contaminazione delle strutture dati del sistema, il processore viene immediatamente fermato, riducendo la possibilità che venga contaminato anche il database.

2. Tutti gli errori significanti in un sistema software divengono molto visibili e poiché l'intero stato del processore viene ghiacciato e copiato quando viene individuato un errore, è più facile scoprire la causa di un errore. Così gli errori che minano la stabilità del sistema e l'integrità dei dati sono trovati e corretti molto velocemente. Il risultato è un software di più alta qualità ed un numero inferiore di fallimenti che devono essere tollerati.

Le coppie di processi garantiscono un'esecuzione tollerante ai guasti dei programmi. Tollerano ogni singolo guasto hardware e la maggior parte dei guasti software transienti. La maggior parte dei guasti nella produzione del software sono transienti, in quanto gli errori di programmazione più semplici sono eliminati durante il testing. Le coppie di processi permettono ai programmi fail fast di continuare l'esecuzione tramite il processo di backup quando il baco software è transiente.

6.8 I processi di I/O

La maggior parte dei processi possono essere eseguiti in un qualsiasi processore. L'eccezione è costituita dai processi di I/O perché gestiscono dei dispositivi ed ogni dispositivo è connesso a due processori attraverso i controller dual ported del dispositivo. Quando viene configurato un dispositivo, una coppia di processi di I/O viene automaticamente definita per quel dispositivo; un membro della coppia risiede in ogni processore connesso al dispositivo. I parametri di configurazione possono indicare quale processore rappresenta il percorso di accesso normale primario, ma questa scelta può essere modificata dall'operatore per il testing e per l'armonizzazione delle prestazioni. Nel caso di un fallimento di un processore, del canale o di una porta del controller, il processo che ha ancora accesso al dispositivo ne assumerà il controllo e tutte le richieste saranno indirizzate ad esso.

Quando avviene una richiesta per un'operazione come l'apertura o la chiusura di un file, il processo primario manda questa informazione al processo di backup attraverso il sistema dei messaggi. Questi checkpoint assicurano che il processo di backup abbia tutte le informazioni di cui ha bisogno per assumere il controllo del dispositivo. Le coppie di processi forniscono un'interfaccia uniforme

per accedere ai dispositivi di I/O e a tutte le altre risorse di sistemi ampi. Questo accesso è indipendente dalla funzione eseguita nei processi, dalla loro locazione o dalla loro implementazione. Nella coppia di processi il sistema dei messaggi è usato per fare checkpoint dei cambiamenti dello stato così che il processo di backup può assumere il controllo quando avviene un fallimento. Una coppia di processi per un disco mirrored è mostrata in Figura 21.

6.8.1 Gestione dei dischi: introduzione

Benché il progetto d'insieme di un'applicazione OLTP possa essere molto complesso, è essenzialmente composto di molti semplici server che richiedono una piccola quantità di calcolo ed eseguono un grande numero di accessi e di aggiornamenti al database. Per raggiungere delle alte prestazioni in questo ambiente è necessaria una grande quantità di sofisticazione nel progetto del software di gestione del database. In questa sezione, daremo solo un ampio sommario della miriade di responsabilità e di funzioni del processo del disco, ma esso chiaramente gestisce la maggior parte dei compiti richiesti dai componenti del sistema. Ogni coppia di processi del disco deve gestire una coppia di dischi mirrored, ognuno connesso a due controller che sono a turno connessi ai due canali dei processori. Così ci sono otto possibili percorsi per i dati di cui può fallire qualche componente; anche nei rari casi di un fallimento multiplo, il processo del disco deve tentare di usare tutte le risorse disponibili per fornire un accesso continuo ai dati.

6.8.2 Gestione dei dischi : la cache

Ad ogni processo del disco possono essere assegnati molti megabyte di memoria principale per la cache del disco. Nel 1990, il limite superiore era di 56 MB per disco, ma poi è stato costantemente incrementato insieme alla dimensione della memoria principale. Il processo del disco può servire molte richieste concorrentemente e non è così insolito che debba soddisfare una mezza dozzina di richieste dalla cache mentre sta eseguendo una singola operazione sul disco di I/O. Il caso peggiore per la gestione della cache è un'applicazione che esegue degli accessi casuali ad un database molto ampio. La probabilità che venga riusata una pagina a cui si è fatto un accesso di recente è abbastanza bassa. L'esperienza mostra che

un'applicazione tipica ha di solito al più un tale file in aggiunta a numerosi file più piccoli a cui si fanno accessi in ogni transazione; così il rapporto tra gli accessi alla cache e quelli al dispositivo rimane abbastanza buono. Anche nel caso di file ampi, attraversando una struttura con indice a B Albero per trovare i dati registrati, potrebbero occorrere tre accessi fisici al dispositivo di I/O che possono essere salvati memorizzando sulla cache i blocchi dell'indice.

Benché la cache abbia un ovvio beneficio nel ridurre le letture, la situazione non è così chiara con le operazioni di scrittura. Ci sono molte situazioni in cui gli aggiornamenti dei file su disco sono fatti in blocchi che erano stati aggiornati di recente; se solo una parte di un blocco era sta aggiornata, questo può risparmiare una lettura per ottenere quella parte del blocco non modificata. In modo più significativo, se i blocchi aggiornati possono essere tenuti nella memoria principale e scritti sul disco solo quando conviene, molte scritture su disco possono essere eliminate. Su un sistema convenzionale, non si possono fare queste operazioni perché un fallimento del processore provocherebbe la perdita di un gran numero di aggiornamenti e corromperebbe l'intero database.

Nei sistemi Tandem, potremmo considerare i checkpoint come aggiornamenti del disco al processo del disco di backup, che è molto meno costoso che eseguire un'operazione reale di I/O. Se fallisce un processore, quello di backup dovrebbe assicurarsi che gli aggiornamenti siano stati scritti su disco. Questo approccio comunque, non è abbastanza sicuro in quanto è possibile che per un lungo fallimento dell'alimentazione o per semplice sfortuna avvenga un fallimento multiplo con la perdita di un numero inaccettabile di aggiornamenti del database. Fortunatamente c'è una soluzione a questo problema ma dobbiamo rimandarne la discussione finché non abbiamo parlato dei concetti base delle transazioni.

6.8.3 Gestione dei dischi: il mirroring

I dischi mirrored sono, come il nome indica, l'immagine identica l'uno dell'altro. Potrebbe sembrare che questa sia una situazione in cui la tolleranza ai guasti richieda una risorsa ridondante e costosa che non contribuisce alle capacità del sistema: il valore dell'integrità dei dati in molti casi comunque giustifica il costo dei dischi ridondanti. Fortunatamente, comunque, anche se i dischi ridondanti non contribuiscono alla capacità di memorizzazione, contribuiscono in modo signifi-

cativo alla capacità di elaborazione. Quando i dati devono essere letti dal disco, il processo del disco può usare entrambi i due dischi e di solito userà il disco che offre il tempo di seek più corto. Le richieste di lettura multiple possono essere eseguite in modo concorrente e, se un disco è occupato, può essere usato l'altro. Poiché i dischi doppi offrono dei tempi di seek più corti, supportano un tasso di letture più alto dei dischi ordinari.

Ogni operazione di scrittura deve essere fatta in ognuno dei dischi mirrored e richiede che entrambi i dischi siano posizionati nello stesso cilindro, riducendo quindi la possibilità di avere un seek breve durante la lettura successiva. Di conseguenza, le scritture si disco sono abbastanza più costose delle letture ma, quando eseguite in parallelo, non sono molto più lente di una scrittura su un singolo disco. L'uso appropriato della cache del disco, specialmente quando è protetta dalla gestione delle transazioni, può eliminare gran parte delle scritture su disco senza sacrificare l'integrità dei dati.

Gli utenti che considerano che tutto o solo una parte del database sia una risorsa non critica possono usare i dischi unmirrored su base disk by disk. I dischi moderni sono molto affidabili e anche quando falliscono, è estremamente raro che vengano persi dei dati. Molte attività, come lo sviluppo del software, non sarebbero state molto influenzate dalla rara indisponibilità dei dischi.

6.8.4 Gestionte dei dischi: il partizionamento

Se un'applicazione OLTP ha un alto tasso di transazioni e ogni transazione accede ad un particolare file, il carico sul processo del disco, anche con una cacheing perfetta, potrebbe essere troppo grande per essere sostenuto su un singolo processore. Come con le applicazioni, è necessario distribuire il carico degli accessi al database facilmente tra i vari processori: in generale, una ridistribuzione dinamica del carico non si è resa necessaria e quindi dovrebbe essere facile ridistribuire il carico in modo statico.

Il concetto del partizionamento non è nuovo ma il processo del disco ed il file system cooperano per fornire una soluzione semplice e flessibile. Qualsiasi file può essere semplicemente partizionato emettendo alcuni comandi che specificano gli intervalli di variazione delle chiavi in ogni partizione. Per le applicazioni, il file partizionato appare come un singolo file e il file system ridirige ogni richie-

sta di lettura o di scrittura alla partizione appropriata in modo dipendente dalla chiave del record. Se una partizione diviene troppo piena o troppo occupata, può essere spezzata suddividendo gli intervalli delle chiavi e muovendo i record appropriati nelle nuova partizione.

Le partizioni di un file possono essere distribuite su una rete e così può essere costruito e mantenuto un database distribuito in un modo che è completamente invisibile alla applicazioni, mentre vengono anche mantenute delle prestazioni eccellenti quando si fanno degli accessi ai dati locali. Per esempio, si potrebbero costruire facilmente 50 partizioni da un file, una per ogni stato della America e porle fisicamente in dei sistemi Tandem diversi che si trovano uno in ogni stato. In ogni sistema, i dati locali allo stato potrebbero essere elaborati in modo efficiente ma ogni processo dell'applicazione potrebbe avere l'accesso allo intero database come se fosse un singolo file, soggetto solamente agli inevitabili ritardi di comunicazione.

6.8.5 Gestione dei dischi: il locking

Un'applicazione OLTP potrebbe avere centinaia di server indipendenti che eseguono degli accessi incondizionati allo stesso insieme di file; il processo del disco deve supportare il lock dei file e dei record, sia per gli accessi condivisi che per quelli esclusivi. Il lock può essere ottenuto attraverso la richiesta esplicita di una applicazione o attraverso le operazioni di gestione delle transazioni che automaticamente forniscono l'atomicità e l'isolamento di tutti gli accessi del database di una transazione. Qualcosa di più sulle transazioni appare più avanti.

6.8.6 Gestione dei dischi: il file system

Il file system è un insieme di routine che sono eseguite nei i processi delle applicazioni e nella gestione della comunicazione con i processi di I/O e con gli altri processi delle applicazioni. Per accedere ad un dispositivo di I/O, il file system nasconde la struttura dei processi e dei messaggi e sembra che il programma dell'applicazione faccia una richiesta diretta ad un supervisore locale dello I/O. Cioè, il file system fornisce un'interfaccia per le chiamate delle procedure ai processi remoti di I/O, mascherando il fatto che sono chiamate di procedure remote.

L'applicazione è inconsapevole della natura distribuita del sistema. Per implementare i file partizionati, il file system gestisce automaticamente le richieste. Implementa il buffering, così che molte operazioni sequenziali possono essere soddisfatte con una sola richiesta al processo di I/O.

Il file system implementa il primo livello di sicurezza nel sistema, in quanto non permette ad un'applicazione di mandare un messaggio ad un processo o ad una applicazione di I/O a meno che l'applicazione non si sia precedentemente identificata con un messaggio di Open che contiene un user name autenticato. Se il processo server nega l'accesso all'oggetto, il file system non permetterà che vengano inviati più messaggi (eccetto i messaggi di Open). Il file system gestisce anche i timeout e la ritrasmissione delle richieste; manda i messaggi al processo del server di backup se il primario è fallito. In aggiunta, se il client è una coppia di processi, il file system gestisce il checkpoint dello stato del processo al processo di backup.

6.9 Le transazioni

Il lavoro coinvolto in una elaborazione può essere impacchettato come un'unità atomica usando la Transaction Monitoring Facility (TMF) illustrata in Figura 23 e in Figura 24. Questa capacità permette ad un'applicazione di fare una richiesta di Begin Transaction, di fare numerosi accessi al database e aggiornamenti su file multipli, su dischi multipli e su nodi multipli della rete e poi di fare una richiesta di End Transaction. Il sistema garantisce che il lavoro delle transazioni sia ACID, cioè:

- *Atomic* (atomico): devono essere eseguiti tutti o nessuno degli aggiornamenti del database; per esempio, se una transazione muove del denaro da un conto di bilancio di una banca ad un altro, alla fine non ci dovrà essere una quantità minore o maggiore di denaro nei due conti.
- *Consistent* (consistente): ogni transazione che termina con successo deve preservare la consistenza del database.
- *Isolated* (isolato): gli eventi di una transazione non devono essere eseguiti concorrentemente alle altre transazioni; altrimenti una transazione che fallisce potrebbe non essere resettata al suo inizio.

- *Durable* (persistente): il risultato di una transazione deve sopravvivere a qualsiasi fallimento una volta che gli è stato dato il commit.

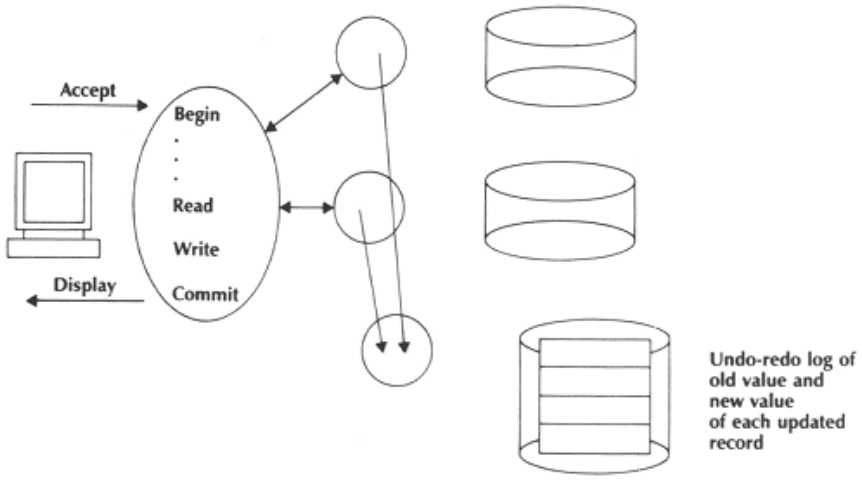


Figura 23 La struttura del TMF, Transaction Monitor Facility.

L'applicazione o il sistema operativo Guardian dovrebbero individuare un problema che compromette la transazione e chiedere la Abort Transaction che farà in modo che tutti gli aggiornamenti del database siano disfatti. Il sistema può gestire migliaia di transazioni concorrenti, tenendole tutte isolate l'una dall'altra. Il programma dell'applicazione non ha a che fare con il protocollo dei lock in quanto le operazioni di locking richieste sono ben definite ed eseguite automaticamente.

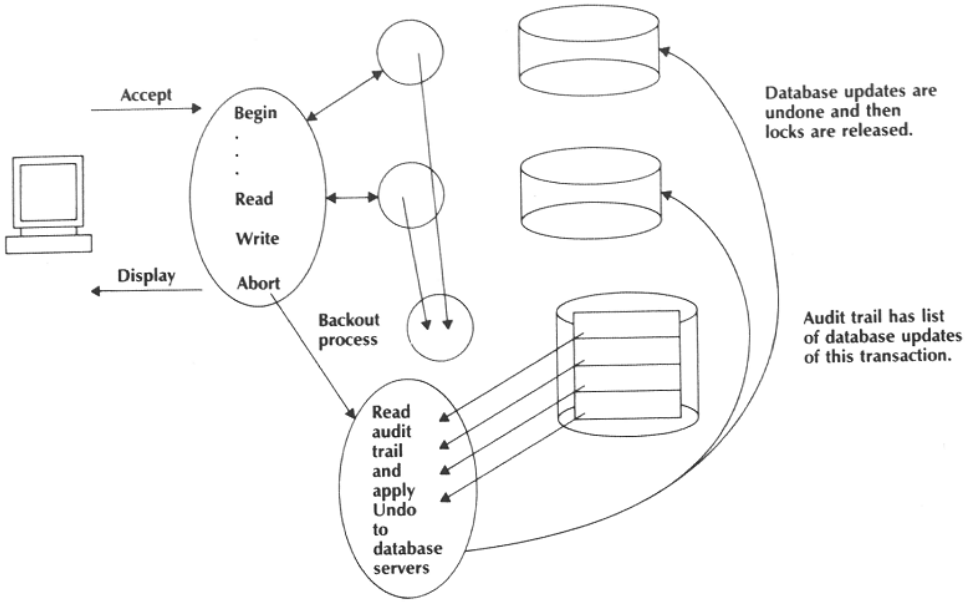


Figura 24 La struttura di una transazione di backout del TMF.

Il lavoro di una transazione può essere distribuito su processi multipli. Come vedremo, è normale strutturare un'applicazione in dei processi client che si rivolgono a numerosi processi server, ognuno progettato per eseguire una parte semplice della transazione. Una volta che il client ha chiesto il Begin Transaction, tutte le richieste ai server sono marcate come appartenenti alla transazione; tutti gli aggiornamenti del database fatti dal server diventano parte di una singola transazione finché il client non fa una richiesta di End Transaction o di Abort Transaction. Quando un client chiede un Begin Transaction, il sistema crea un *unico identificatore della transazione* e lo usa per etichettare tutti i messaggi e tutte le richieste del database. Se un messaggio etichettato viene accettato da un server, tutti i suoi messaggi e tutte le richieste del database ricevono la stessa etichetta. Anche i lock del database acquisiti dal client o dai suoi server sono etichettati con lo identificatore della transazione.

Durante una transazione, i processi del disco mantengono gli aggiornamenti nella loro cache. Se non c'è abbastanza cache disponibile, il processo del disco può aggiornare il database prima che la transazione finisca, ma prima deve generare i record di undo e redo che permettono alla transazione di essere disfatta nel caso che fallisca (abort) e di essere rifatta nel caso che vada a buon fine (commit) ed accada un fallimento più tardi. Quando il client chiede l'End Transaction, ogni processo del disco con degli aggiornamenti per quella transazione deve generare dei record di log di undo e redo (fare e disfare) prima di aggiornare il disco. I record di undo e di redo sono scritti in dei file di log delle transazioni specificatamente progettati chiamati *audit trail*. Normalmente questi file sono su dischi separati dal database e i record di undo e di redo devono essere scritti sul disco degli audit trail prima che il database principale sia aggiornato. Così anche un fallimento totale del sistema non farebbe perdere le transazioni che sono state committed e non renderebbe inconsistente il database. Anche se venisse distrutta una coppia di dischi mirrored, il database e tutte le transazioni sarebbero ripristinate da una copia del disco e del log delle transazioni. Qualsiasi processo che partecipa alla transazione può abortirla unilateralmente. Il sistema implementa il protocollo two phase locking ed usa un protocollo two phase commit nonblocking, grouped e presumed abort.

Potrebbe sembrare che il supporto per le transazioni aggiunga del considerabile overhead alle operazioni di base di accedere e aggiornare il database. Sor-

prendentemente il TMF migliora le prestazioni, mentre consolida la tolleranza ai guasti e l'integrità dei dati. Come detto in precedenza, gli aggiornamenti del database dovrebbero essere scritti immediatamente sul disco per evitare la loro perdita nel caso di un fallimento. Questo incrementa il traffico del disco e allunga il tempo di risposta della transazione. Quando le operazioni del database sono protette dal TMF, una descrizione di tutti gli aggiornamenti viene scritta nello audit trail; non è quindi più necessario per il processo del disco scrivere gli aggiornamenti sul database, eccetto quando è conveniente farlo. Finché i record degli audit trail sono memorizzati in modo affidabile sul disco, l'applicazione può notificare che gli aggiornamenti sono stati registrati in modo permanente e sopravviveranno a qualunque fallimento. Scrivere gli aggiornamenti nello audit trail è considerabilmente più efficiente di scriverli nel database perché molti aggiornamenti di una singola transazione sono bloccati insieme e scritti in una solo operazione di I/O. Per di più, lo audit trail è scritto in modo sequenziale e la scrittura è eseguita con un minimo di posizionamenti, mentre gli aggiornamenti del database sono ad accesso casuale ed implicano numerosi posizionamenti. Infine, il processo del disco esegue meno operazioni di checkpoint per il suo backup poiché gli aggiornamenti non committed non necessitano di essere protetti contro i fallimenti del processore.

Il risultato di questi effetti è che il logging e il recovery del TMF è un salvataggio netto sopra la meno funzionale cache store-thru-disk. Il TMF converte gli accessi casuali al database nella memoria principale in accessi sequenziali, riducendo moltissimo la densità dei trasferimenti di I/O. I benchmark hanno dimostrato che la densità dello I/O può essere ridotta di un fattore di due o tre quando viene impiegato il TMF.

Mentre il sistema Tandem fornisce un'alta disponibilità attraverso una tolleranza ai guasti singoli, il TMF fornisce una tolleranza ai guasti multipli per gli elementi critici del sistema di elaborazione delle transazioni: il database. Benché i fallimenti multipli sono delle rare eccezioni, il costo di una perdita del database è molto alto. Prima dell'introduzione del TMF, i programmatori delle applicazioni si affidavano alle coppie di processi e al forward error recovery per fornire la tolleranza ai guasti. Se veniva individuato un errore, il processo di backup riesumava l'applicazione dall'ultimo checkpoint. Le coppie di processi erano difficili da progettare ed implementare e riducevano la produttività dei programmatori di appli-

cazioni. Le applicazioni implementate con il TMF sono molto più semplici da programmare e raggiungono un livello equivalente di tolleranza ai guasti. Poiché le transazioni implicano un protocollo di locking automatico, è molto più facile mantenere consistente il database.

Le coppie di processi sono ancora un importante concetto e sono usate per il sistema e per i processi di I/O. Sono gli elementi fondamentali su cui il TMF ed altri sistemi software sono costruiti così che la clientela può scrivere delle applicazioni tolleranza ai guasti senza preoccuparsi affatto della tolleranza ai guasti. Nella Tabella 4 sottostante viene riportato un esempio delle transazione eseguite da una coppia di processi per mantenere la consistenza.

Step	Requester SeqNo = 0	Requester Backup SeqNo = 0	Server SeqNo = 0	Server Backup SeqNo = 0
1	Issue request to write record →			
2			If SeqNo < MySeqNo, then return saved status	
3			Otherwise, read disk, perform operation, check point request →	Saves request
4			Write to disk SeqNo = 1 →	Saves result SeqNo = 1
5			Checkpoint result Return results	
6	Checkpoint Results →	SeqNo = 1		

Tabella 4 Esempio delle transazioni di una coppia di processi.

Consideriamo l'interazione di una coppia di processi come illustrato in Tabella 4. Inizialmente tutti i numeri delle sequenze (SeqNo) sono messi a zero. Il richiedente manda una richiesta al server. Se il numero della sequenza è più piccolo di quello della copia locale del server, è accaduto un fallimento e viene restituito lo stato dell'operazione completata. Da notare che l'operazione del richiedente viene fatta una sola volta. Poi il disco viene scritto, viene incrementato ad uno il numero della sequenza e viene fatto un checkpoint dei risultati sul server di backup che incrementa il suo numero di sequenza. I risultati vengono restituiti dal server al richiedente. Infine viene fatto un checkpoint dei risultati sul richiedente di backup che incrementa il suo numero di sequenza. Supponiamo adesso che avvenga un fallimento. Se entrambi i processi di backup falliscono, l'operazione termina con successo. Se fallisce il richiedente dopo che ha fatto la richiesta, il server completerà l'operazione ma non sarà in grado di restituire il risultato. Quando il richiedente di backup diviene attivo, ripeterà la richiesta. Poiché il suo

numero di sequenza è zero, il test del server al passo 2 restituirà il risultato senza eseguire di nuovo l'operazione. Infine, se il server fallisce, il server di backup non fa niente e non completa l'operazione usando le informazioni di checkpoint. Quando il richiedente rimanda la richiesta, il nuovo server (cioè il vecchio server di backup) esegue l'operazione e restituisce i risultati salvati.

6.9.1 Lo SQL del NonStop

Il file system e il server del disco cooperano per elaborare le operazioni sul database fatte per mezzo dello Structured Query Language (SQL) in modo integrato ed efficiente. Il file system gestisce l'elaborazione dello SQL nel client ed esegue tutte le operazioni di alto livello come l'ordinamento, il join e la aggregazione. Il processo del disco comprende i costrutti semplici come le tabelle, i campi e le espressioni ed esegue le operazioni SQL di basso livello sui dati. Così le operazioni possono essere eseguite al livello che garantisce una maggiore efficienza. Per esempio, le istruzioni SQL

```
UPDATE ACCOUNT SET BALANCE = BALANCE + :DEPOSIT-AMOUNT
WHERE ACCOUNT_NUMBER = :ACCOUNT-ID;
```

possono essere elaborate con un solo messaggio al processo del disco. Non c'è bisogno che l'applicazione prelevi l'informazione, la aggiorni e la invii indietro al processo del disco. In un altro esempio, l'istruzione

```
SELECT FIELD_A, FIELD_E FROM TABLEX
WHERE FIELD_A + FIELD_B > FIELD_C;
```

permette che il filtraggio sia eseguito dal processo del disco, minimizzando il trasferimento dei dati all'applicazione.

Lo SQL del NonStop è progettato specificamente per gestire le applicazioni OLTP raggiungendo un buon livello di prestazioni. Poiché è integrato con lo altro software di sistema, può essere usato per le applicazioni OLTP nei sistemi di reti geograficamente distribuiti. Le tabelle SQL possono essere partizionate attraverso i sistemi presenti nella rete. Le applicazioni possono essere eseguite in un nodo della rete mentre accedono e aggiornano i dati di un altro nodo. Per di più, le applicazioni stesse possono essere distribuite. Con lo SQL del NonStop, la tolleranza ai guasti deriva dal meccanismo base delle coppie di processi, dai dischi mirrored e dal sistema geograficamente distribuito, insieme all'autonomia dei no-

di ed al supporto delle transazioni. Tutte le transazioni, locali o nella rete, sono protette dalla consistenza e dal ripristino degli errori.

Dal punto di vista della di tolleranza ai guasti, ci sono due novità circa lo SQL del NonStop. La prima è il progetto dell'autonomia dei nodi. Il sistema è progettato in modo che se il client e il server possono comunicare, allora il client può accedere ai dati. Questa semplice richiesta implica che tutti i metadati che descrivono il file devono essere replicati con il file. Se il file è partizionato tra molti nodi delle rete, l'informazione del catalogo che descrive il file deve essere replicata in tutti questi nodi. La seconda richiesta è che non devono essere permesse delle operazioni amministrative sui dati che usino il database offline. Per esempio, lo utilizzo del dump di un archivio deve essere fatto mentre i dati vengono acceduti, la riorganizzazione dei dati deve essere fatta online e così via. Molti compiti amministrativi non sono ancora online, ma uno dei maggiori degli sforzi correnti è quello di rendere tutti i compiti amministrativi online.

L'SQL permette agli amministratori dei i dati di attaccare dei vincoli di integrità ai dati; questi possono essere dei *vincoli di entità* che limitano i valori che un record può assumere o dei *vincoli di integrità referenziale* che vincolano le relazioni tra i record in tabelle differenti. Mettere i vincoli sui dati è più affidabile che lasciare al programma dell'applicazione il compito di eseguire tali test. Gli aggiornamenti dei dati che violano questi vincoli sono rifiutati.

6.9.2 Il monitor delle transazioni Pathway

Le applicazioni sono strutturate come processi client server. I processi client sono i responsabili della presentazione dei servizi e della gestione dell'interfaccia dello utente. Le interfacce degli utenti spaziano da quelle orientate alle form a quelle tipo posta elettronica o sistemi home banking, a quelle real time dove l'utente si trova in un magazzino automatizzato, in una pompa per il gas o in una centralina telefonica. I server sono programmati per eseguire specifiche funzioni, di solito un insieme di accessi e di aggiornamenti correlati ad un database. Nell'esempio della posta elettronica, un server consulta i nomi mentre un altro è responsabile di instradare i messaggi agli altri nodi della rete e ai gateway. Nell'esempio della pompa del gas, un server fa l'autenticazione mentre un altro fa le fatture. Tipicamente le applicazioni sono strutturate come un gruppo di centinaia di servizi.

Spezzare un'applicazione in requestor e server promuove la modularità del software, permette dei cambi online e la crescita delle applicazioni e sfrutta la architettura dei microcomputer. Con l'avvento dei terminali intelligenti, il client è emigrato verso le workstation e l'architettura client server sta diventando la struttura standard per tutte le applicazioni di elaborazione delle transazioni.

Il progettista dell'applicazione specifica i programmi e i parametri per ogni client e server. I server possono essere programmati in qualsiasi linguaggio, ma i client sono stati programmati tradizionalmente in un dialetto del COBOL chiamato Screen COBOL. Questo linguaggio interpretato gestisce automaticamente le transazioni e le coppie di processi. Nel caso in cui il processo client fallisca per una qualche ragione, il processo di backup lo rileva, riprocessa il messaggio in input se necessario e rilascia il messaggio di output. Questo dà esattamente una volta la semantica all'elaborazione delle transazioni. Lo Screen COBOL solleva il programmatore delle applicazioni dal bisogno di capire come scrivere le coppie di processi. È il modo più comune con cui i clienti ottengono l'integrità dei messaggi. Il monitor dell'elaborazione delle transazioni, Pathway (Figura 25), è il responsabile della gestione delle applicazioni dei requestor e dei server. Crea i requestor e i server alla accensione del sistema, mantiene un database di configurazione che può essere alterato online dai comandi degli operatori e bilancia il carico del sistema creando e cancellando le istanze dei server quando il carico varia e quando i processori vanno e vengono dal sistema.

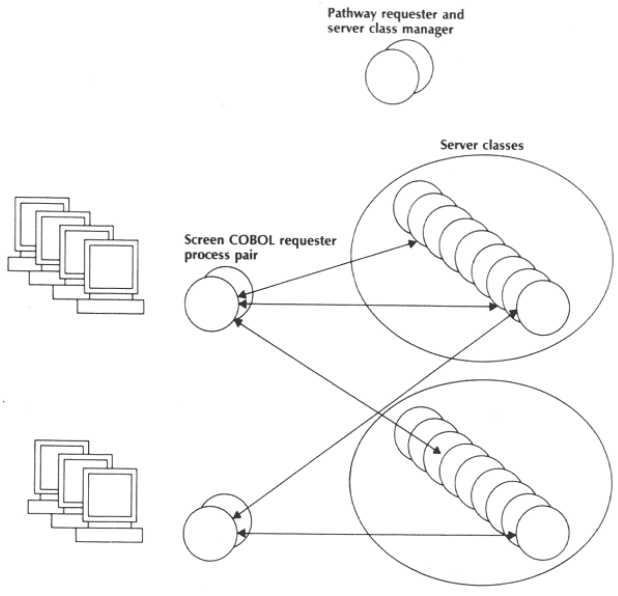


Figura 25 Struttura del Pathway, il monitor dell'elaborazione delle transazioni.

6.10 I processi server

Per ottenere la modularità del software, le computazioni sono spezzate in diversi processi. Per esempio, una transazione che arriva da un terminale passa attraverso un processo line handler (per esempio X.25), un protocollo (per esempio SNA), un processo di presentazione dei servizi per fare la gestione dello schermo, un processo dell'applicazione che gestisce la logica del database e diversi altri processi del disco che gestiscono i dischi, i pool del buffer del disco e gli audit trail delle transazioni. Questo metodo spezza l'applicazione in molti piccoli moduli che rappresentano delle unità di servizio e di fallimento. Se una unità fallisce, la sua computazione passa al suo processo di backup. Se un processo esegue un particolare servizio (per esempio è un name server o gestisce un database particolare), allora è probabile che il lavoro su questo server cresca come cresce il sistema. Gradualmente, il carico su tale processo aumenterà finché esso diverrà un collo di bottiglia. I colli di bottiglia possono essere un impedimento alla crescita lineare delle prestazioni quando vengono aggiunti dei processori. Il concetto di *classe di processi server* è stato introdotto per aggirare il problema dei colli di bottiglia.

Una classe di server è un insieme di processi che eseguono tutti la stessa funzione, tipicamente sparsa sopra diversi processori. Un tale insieme può essere gestito da Pathway. Le richieste sono mandate ad una classe piuttosto che ai membri individuali di una classe. Come il carico aumenta, vengono aggiunti dei membri alla classe. Se fallisce un membro od uno dei processori, la classe dei server migra nei processori rimanenti. Come il carico diminuisce, la classe dei server si restringe. Quindi, le classi dei processi server sono un meccanismo per la tolleranza ai guasti e per la distribuzione del carico in un sistema distribuito. Il progettista dell'applicazione specifica il programma, i parametri, la taglia minima e quella massima e la distribuzione delle classi si processi. Il Pathway legge questo database di configurazione all'avvio del sistema e gestisce le classi dei server, aumentandole quando il carico aumenta e restringendole quando il carico diminuisce.

6.11 Il networking

Un rete proprietaria chiamata Expand ampia il progetto originale basato su 16 processori ad una rete con 4080 processori e 255 nodi: questa generalizza in modo naturale la struttura di Guardian basata sui processi e sui messaggi. Expand usa un packet switched, hop by hop schema di instradamento per muovere i messaggi tra i nodi; in essenza, vengono connessi tutti gli IPB del sistema remoto. Expand è ora ampiamente usato come la chiave di volta per le reti collettive o come una rete intelligente che gioca il ruolo di porta tra le altre reti.

La tolleranza ai guasti e la modularità dell'architettura la rende una scelta naturale per queste applicazioni. Inoltre il software di sistema supporta gli standard come SNA, OSI, MAP, SWIFT, TCP/IP, Named Pipes ad altri. Questi protocolli portano al massimo il sistema dei messaggi e lo estendono. La tolleranza ai guasti fornita dal sistema si estende alla rete. Il software di rete permette di continuare una sessione anche se si rompe il link di comunicazione. Per esempio, SNAX, l'implementazione di Tandem dell'IBM Stystem Network Architecture, fornisce un servizio continuo per mezzo di un checkpoint trasparente nei punti chiave dell'informazione interna che hanno bisogno di operazioni di sostegno . Questo permette a SNAX di mantenere tutte le sessioni attive anche nel caso in cui falliscano un processore o una linea. Provvedimenti simili esistono nel software di sistema di interconnessione aperta.

La tolleranza ai guasti è anche alla base dei prodotti di gestione dei sistemi distribuiti per la gestione globale dei sistemi NonStop e delle reti Expand. Per esempio, con il sottosistema di gestione degli eventi, un processo di registrazione di un evento come una coppia di processi NonStop fornisce un dolce ripristino dai fallimenti singoli di processo. Cioè, se il processo primario di registrazione di un evento fallisce o viene fermato, il processo di backup continua a registrare nel log degli eventi appropriato.

6.12 La protezione dai disastri

Le operazioni di ripristino da un disastro sono generalmente un processo complesso, che richiede molto lavoro, soggetto ad errori, che comunemente im-

piega dalle 12 alle 48 ore. Le possibilità di ripristino da un disastro cambiano molto quando ci si sposta da un ambiente tradizionale di tipo batch al mondo della elaborazione delle transazioni online. La clientela dello OLTP richiede che il ripristino avvenga in una manciata di secondi o di minuti con una piccola o nessuna perdita di transazioni odi dati. Le reti simmetriche e i progetti delle applicazioni basate sulla possibilità della duplicazione remota del database (RDF) danno questo tipo di protezione ai disastri. Le operazioni personali possono usare lo RDF per far ritornare indietro le applicazioni online in pochi minuti dopo un disastro.

Lo RDF memorizza il database in due sistemi tipicamente in duo luoghi geograficamente distinti. Per ogni dato c'è una copia primaria e una di backup dei record. Tutti gli aggiornamenti sono fatti nella copia primaria e i record di log del TMF per il primario sono mandati nel luogo che contiene la copia di backup dei record, dove sono sostituiti alla copia di backup dei dati. L'utente può selezionare una di queste tre opzioni per l'invio di questi record di log:

- **2-Safe:** Non c'è nessuna perdita di transazioni in rilevamento. In questo caso, la transazione è registrata in entrambi i luoghi prima di mandare la transazione al client. Occorre un tempo di risposta leggermente più lungo perché si aggiungono i ritardi della rete.
- **1-Safe:** Una parte delle ultime transazioni in rilevamento potrebbe essere persa perché tale parte non è stata memorizzata del luogo di backup. *1-Safe* ha un tempo di risposta migliore e può essere appropriato se ogni transazione è di poco valore o è facilmente ricostruibile in seguito.
- **Electronic Vaulting:** Il log del sistema è semplicemente trasmesso in un luogo remoto e memorizzato là. Non viene applicato ad un database remoto finché non c'è un disastro reale. Questo è simile allo schema del luogo di standby ma evita i movimenti dei dati dal luogo di standby quando avviene il disastro.

Lo RDF rende possibile il mantenimento di una copia corrente e online del database in un nodo secondario, come illustrato in Figura 26. Il database secondario può essere posto vicino o attraverso la nazione. L'uso dello RDF è completamente trasparente al programmatore delle applicazioni. Ogni applicazione TMF può essere convertita in un'applicazione RDF senza alcun cambio: cambiano solo la configurazione del database e le operazioni delle procedure. Per supportare le sue capacità di backup, lo RDF monitorizza ed estrae informazioni dallo audit file del TMF e invia queste informazioni attraverso la rete Expand ad un processo

RDF corrispondente sul secondo nodo. Questa estrazione di solito impiega alcuni secondi della creazione dello audit file da parte del TMF. Il processo RDF sul secondo nodo riceve le transazioni dello audit e le memorizza nel buffer del disco. Un altro processo RDF sul secondo nodo applica queste transazioni al database, mantenendo così duplicato il database. La seconda copia del database è di solito al corrente del database primario in pochi secondi. L'attività di aggiornamento sul secondo database può essere temporaneamente sospesa senza compromettere l'integrità del processo di backup online. Le transazioni dello audit si accumulano nel buffer del disco nel luogo secondario finché non riprende l'aggiornamento. Allora tutti gli aggiornamenti accumulati sono automaticamente applicati nella sequenza corretta.

Se uno dei luoghi ha bisogno di essere disconnesso per un aggiornamento del software o dello hardware, il carico può essere indirizzato verso altri nodi senza interrompere il servizio.

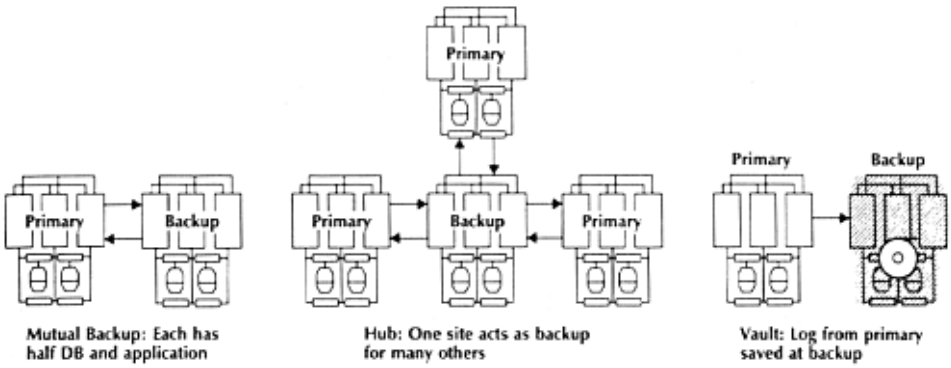


Figura 26: I meccanismi per la duplicazione remota del database.

6.13 Sommario del sistema operativo

Guardian come si vede è un sistema operativo che riesce a sintetizzare in modo efficace i concetti introdotti finori, come i processi e i messaggi.

Attraverso tali astrazioni si riesce a rendere trasparente all'utente la struttura hardware ridondante e sono un punto essenziale per la tolleranza ai guasti. Inoltre forniscono anche l'indipendenza dalla configurazione che è necessaria per permettere al software di sistema e alle applicazioni di girare su sistemi di varia grandezza. Le coppie di processi sono l'estensione naturale del concetto di pro-

cesso per ottenere un'esecuzione in grado di tollerare i guasti software transienti. Le transazioni sono state integrate nel sistema operativo e appaiono come una parte naturale dell'ambiente di esecuzione.

L'estensione del sistema basato sui messaggi ad una rete long haul lo rende geograficamente distribuito. Il sistema operativo fornisce al programmatore delle applicazioni un approccio generale alla comunicazione tra processi ed alla tolleranza ai fallimenti.

Infine, gli obiettivi del progetto del sistema sono stati dimostrati praticamente: dei sistemi con 2 fino a 200 processori sono stati installati e stanno funzionando con applicazioni online. Molti di questi sistemi fanno parte di reti multinodi. Sono stati ripristinati dai fallimenti e sono stati riparati online, con un minimo o nessun impatto sugli utenti del sistema

Conclusioni

La tolleranza ai guasti singoli attraverso l'uso di moduli fail fast e della riconfigurazione, deve essere applicata sia al software sia allo hardware. I processi e i messaggi sono la chiave per strutturare il software in moduli con un buon isolamento dei guasti. Un beneficio collaterale di questa progettazione è che può utilizzare processori multipli e che si presta al progetto di un sistema distribuito. La crescita modulare del software e dello hardware è un effetto collaterale della tolleranza ai guasti. Se il sistema può tollerare la riparazione e la reintegrazione dei moduli, allora può anche tollerare l'aggiunta di nuovi moduli marcati. In aggiunta, il sistema deve tollerare i guasti delle operazioni e ambientali.

La tolleranza ai guasti può essere applicata anche ai sistemi aperti basati su standard. Questi sistemi forniscono il beneficio della tolleranza ai guasti trasparente alle applicazioni al costo di risorse addizionali di hardware e un decremento delle risorse per i guasti software.

Glossario

Affidabilità

Continuità di servizio del sistema. Misura della continuità del servizio corretto fornito. Misura del tempo per giungere ad un fallimento.

Ambiente del sistema

È costituito dagli altri sistemi che interagiscono o interferiscono con il sistema dato.

Benchmark

Insieme di programmi selezionati per la valutazione delle prestazioni di un sistema.

Cache (memoria)

Memoria piccola e veloce che funge da area tampone per i trasferimenti di dati tra dispositivi con velocità molto differenti.

Checkpoint

Operazione che viene svolta periodicamente, con l'obbiettivo di registrare quali transazioni sono attive e di trasferire la cache su disco.

Codice

Mezzo per rappresentare l'informazione che usa un insieme di regole predefinite.

Codici rilevatori di errori

Particolare codifica dei dati che permette di individuare i guasti.

Codice separabile

L'informazione originale è seguita dalla nuova informazione, rendendo il processo di decodifica una semplice rimozione della informazione addizionale.

Compensazione degli errori

Forma di elaborazione degli errori in cui uno stato erroneo contiene ridondanza in modo sufficiente a correggere il servizio fornito.

Comportamento del sistema

Quello che fa un sistema.

Confinamento dei guasti

Limitare gli effetti di un guasto in un'area del sistema.

Copertura

Misura di quanto l'attività di validazione copre le situazioni effettive in cui il sistema si troverà durante la sua vita operazionale.

Crash

Fallimento che conduce ad una persistente mancanza di servizio.

Crescita di affidabilità

La capacità del sistema di fornire il servizio corretto viene migliorata (incremento stocastico dei tempi per giungere ad un fallimento).

DED

Codice in grado di individuare due errori.

Dependability

Credibilità e fiducia che possono essere giustificabilmente riposte sul servizio fornito da un sistema.

Diagnosi dei guasti

L'azione di determinare le cause di un errore che riguardano la sua locazione e la sua natura.

Direct Memory Access

Vedi DMA.

Disponibilità

Proprietà di un sistema di essere pronto per l'uso.

Diversità di progetto

Un approccio alla progettazione di un sistema che fa in modo di fornire dei servizi identici con una progettazione e con delle implementazioni separate.

DMA

Meccanismo per cui un dispositivo periferico scambia dati con la memoria centrale senza coinvolgere l'unità centrale.

Dump

Copia completa della base di dati, che viene normalmente effettuata in mutua esclusione con tutte le altre transazioni quando il sistema non è operativo.

Duplicazione e confronto

Due copie identiche dello stesso modulo sono impiegate per individuare i guasti. Un semplice confronto permette la individuazione dei guasti.

ECC

Codici a correzione d'errore.

Elaborazione degli errori

Le azioni intraprese per eliminare gli errori da un sistema.

Errore

Parte dello stato di un sistema che porta ad un fallimento. Manifestazione di un guasto in un sistema.

Errori coincidenti

Errori prodotti dallo stesso input.

Errore latente

Errore non riconosciuto come tale.

Esecuzione simbolica

Verifica dinamica eseguita con ingressi simbolici.

Fallimento

Deviazione del servizio fornito dalle specifiche del sistema. Transizione da un servizio corretto ad uno non corretto.

Fallimento benigno

Fallimento le cui conseguenze sono dello stesso ordine dei benefici forniti dal sistema.

Fallimento catastrofico

Fallimento le cui conseguenze sono incommensurabilmente più grandi dei benefici forniti dal sistema.

Fallimento con omissione

Fallimento che provoca la cessazione del servizio.

Fallimento consistente

Fallimento percepito nello stesso modo da tutti gli utenti.

Fallimenti di modo comune

Fallimenti che derivano da guasti correlati.

Fallimento di stop

L'attività del sistema, se ce ne è, non è più percepibile dagli utenti e viene fornito in uscita un valore di servizio costante.

Fallimento di valore

Fallimento in cui il valore del servizio fornito non si addice alle specifiche.

Fallimento inconsistente

Fallimento percepito in modo diverso dai vari utenti.

Fallimenti sequenziali

Fallimenti che non avvengono nella stessa finestra temporale.

Fallimenti simultanei

Fallimenti che avvengono nella stessa finestra temporale.

Fallimento temporale

Fallimento in cui la temporizzazione del servizio fornito non si addice con le specifiche.

Fault avoidance

Metodi e tecniche che hanno lo scopo di progettare un sistema privo di guasti.

Fault secure

Un circuito si dice fault secure se per ogni guasto facente parte di un insieme prescritto, il circuito non produce mai un'uscita non corretta per ogni parola di codice in ingresso.

Funzione del sistema

Quello per cui è stato realizzato il sistema.

Funzione di affidabilità

Probabilità condizionata che un componente operi correttamente in un certo intervallo temporale, dato che operava correttamente all'inizio di tale intervallo temporale.

Funzione real time

Funzione che deve essere eseguita in un intervallo temporale finito dettato dall'ambiente.

Grado di severità

Grado delle conseguenze del fallimento sullo ambiente del sistema.

Guasto

Causa aggiudicata o ipotizzata di un errore. Causa di un errore che deve essere evitata o tollerata. Conseguenza di un fallimento di un sistema che ha interagito o interagisce con il sistema considerato.

Guasto accidentale

Guasto che appare o che è creato in modo fortuito.

Guasto attivo

Guasto che produce un errore.

Guasti correlati

Guasti attribuiti ad una causa comune.

Guasto di progetto

Guasto interno provocato dall'uomo.

Guasto dormiente

Guasto interno non attivato dal processo di elaborazione.

Guasto esterno

Guasto risultante da interferenze o interazioni ambientali.

Guasto fisico

Guasto risultante da degli avversi fenomeni fisici.

Guasto hard

Guasto che necessita di un insieme di azioni intraprese per evitare sia di nuovo attivato (vedi passivazione).

Guasti indipendenti

Guasti attribuiti a cause diverse.

Guasto intenzionale

Guasto creato premeditatamente.

Guasto intermittente

Guasto temporaneo interno. Guasto le cui condizioni di attivazione non possono essere riprodotte o che avviene abbastanza raramente.

Guasto interno

Guasto interno ad un sistema.

Guasto operazionale

Guasto che avviene durante il funzionamento del sistema.

Guasto permanente

Guasto la cui presenza non è correlata a delle condizioni puntuali interne o esterne del sistema.

Guasto soft

Guasto per cui non è necessaria la passivazione.

Guasto temporaneo

Guasto che è presente per una quantità limitata di tempo.

Guasto transiente

Guasto esterno fisico temporaneo.

Individuazione degli errori

L'azione di identificare che uno stato del sistema è erroneo.

Individuazione dei guasti

Il riconoscere che qualcosa di inatteso è avvenuto nel sistema.

Integrità

È la condizione di essere corretto.

Intrusione

Guasto operazionale interno intenzionale.

Latenza del guasto

Periodo di tempo arbitrario che può trascorrere prima che sia individuato un guasto.

Locking

È un sistema molto usato nei sistemi operativi per sincronizzare gli accessi concorrenti a risorse condivise.

Logica maliziosa

Guasti di progetto intenzionali.

Manuntenibilità

Misura del servizio non corretto fornita dal sistema. Misura del tempo che occorre per la riparazione dall'ultimo fallimento.

Manutenzione correttiva

Mantenimento o miglioramento della capacità di un sistema di fornire un servizio che si addice con le specifiche durante la sua vita operazionale.

Manutenzione curativa

Manutenzione correttiva che ha lo scopo di rimuovere i guasti che hanno prodotto degli errori che sono stati individuati.

Manutenzione preventiva

Manutenzione correttiva con lo scopo di prevenire l'introduzione o la manifestazione dei guasti.

Mascheramento dei guasti

Il risultato dell'applicazione della compensazione degli errori in modo sistematico, anche in assenza di errori.

MTBF

Tempo medio tra i fallimenti di un sistema.

MTTF

Tempo medio in cui un sistema opera correttamente prima che avvenga un fallimento.

MTTR

Tempo medio richiesto per riparare un sistema.

NMR

Generalizzazione del TMR in cui si usano N componenti duplicati.

Paginazione

Divisione della memoria, di qualunque genere, in blocchi di dimensione fissa.

Pair and a spare

Combina le caratteristiche dello stanby sparing e della duplicazione con confronto.

Parità

Aggiunta di un bit di controllo ad ogni gruppo di bit così che la parola risultante abbia un numero pari (parità pari) o dispari (parità dispari) di "1". I codici di parità sono separabili.

Passivazione di un guasto

Insieme di azioni intraprese affinché un guasto non sia più attivato.

Prevenzione dei guasti

Metodi e tecniche che hanno lo scopo di prevenire l'introduzione o la manifestazione dei guasti.

Previsione dei guasti

Metodi e tecniche con lo scopo di stimare il numero, l'incidenza futura e le conseguenze dei guasti.

Processo

Un processo è un'entità eseguibile indipendentemente che consiste principalmente di un codice di programma condivisibile, di memoria privata e di un vettore dello stato del processo. Il vettore di stato del processo include il program counter, i registri, i privilegi, la priorità e un timer che viene incrementato solo quando il processo è in esecuzione.

Punto di ripristino

Istante temporale nell'esecuzione di un processo da cui lo stato corrente potrebbe essere reinizializzato.

Punto singolo di fallimento

Componente del sistema il cui fallimento conduce ad un fallimento del sistema.

Redo

Operazione che viene svolta quando una o più transazioni correttamente terminate non hanno reso persistente i dati; consiste nel *rifare* tutto il lavoro svolto solitamente a partire dall'ultimo checkpoint.

Reintegrazione

Fase in cui il modulo riparato deve essere reintegrato nel sistema.

Restart

Riesumazione di tutte o di parte delle operazioni dal punto in cui viene individuato il guasto.

Ridondanza

Risorse o informazioni aggiuntive che occorrono per durante il normale funzionamento del sistema.

Ridondanza attiva

Individua l'esistenza dei guasti ed esegue una qualche azione per rimuovere lo hardware guasto dal sistema.

Ridondanza ibrida

Combina la ridondanza attiva con quella passiva.

Ridondanza passiva

Usa il concetto del mascheramento dei guasti per nascondere i guasti e per prevenire che i guasti originino degli errori.

Ridondanza temporale

Riduce la quantità dello hardware ridondante a spese di una maggiore utilizzazione temporale.

Ridondanza triple duplex

Variazione della ridondanza ibrida che combina la duplicazione con confronto e la ridondanza triplice modulare.

Riconfigurazione

Capacità del sistema di riconfigurare i suoi componenti

Rimozione dei guasti

Metodi e tecniche con lo scopo di ridurre la presenza dei guasti.

Riparazione

Fase in cui un componente guasto viene sostituito.

Ripristino all'indietro

Forma di ripristino dagli errori che consiste nel riportare il sistema in uno stato corretto precedentemente occupato.

Ripristino degli errori

Forma di elaborazione degli errori dove uno stato erroneo è sostituito con uno privo di errori.

Ripristino del servizio

Transizione dal fornire un servizio non corretto ad uno corretto.

Ripristino in avanti

Forma di ripristino dagli errori in cui lo stato erroneo viene portato in nuovo stato non precedentemente occupato.

Safety

Proprietà di un sistema di non produrre fallimenti catastrofici. Misura della continuità del servizio corretto o non corretto dopo un fallimento benigno. Misura del tempo per giungere ad un fallimento catastrofico.

SEC

Codice capace di correggere un solo errore.

Self checking (componente)

Componente che include dei meccanismi di individuazione degli errori associati con la sua parte funzionale.

Self testing

Un circuito si dice self testing se per ogni guasto facente parte di un insieme prescritto, il circuito produce in uscita una parola che non fa parte del codice per almeno una parola facente parte del codice in ingresso.

Servizio

Comportamento del sistema come percepito dallo utente del sistema.

Servizio corretto

Servizio fornito in accordo con le specifiche del sistema.

Servizio non corretto

Servizio fornito non in accordo con le specifiche del sistema.

Servizio real time

Servizio che deve essere eseguito in un intervallo temporale finito dettato dall'ambiente.

Sicurezza

Prevenzione degli accessi non autorizzati o della gestione delle informazioni.

Sistema

Entità che ha interagito, che interagisce o capace di interagire con altre entità. Insieme di componenti connessi insieme per interagire tra loro.

Sistema atomico

Sistema la cui struttura interna non può essere individuata oppure non è di interesse e può essere ignorata.

Sistema fail safe

Sistema i cui fallimenti sono o possono solo essere benigni.

Sistema fail silent

Sistema i cui fallimenti sono o possono essere dei crash.

Sistema fail stop

Sistema i cui fallimenti provocano o possono provocare l'inattività del sistema con un valore costante in uscita.

Sistema real time

Sistema che esegue almeno una funzione real time o che fornisce almeno un servizio real time.

Specificazione del sistema

Descrizione delle richieste del sistema.

SQL

Linguaggio di interrogazione standardizzato per basi di dati relazionali.

Stack

Organizzazione della memoria per cui è momentaneamente accessibile solo l'ultimo elemento inserito (politica Last In First Out).

Standby sparing

Un modulo è attivo mentre uno o più moduli servono come riserve.

Stato del sistema

Essere in una certa condizione rispetto ad un insieme di circostanze.

Structured Query Language

Vedi SQL.

Struttura del sistema

Che cosa fa un sistema per fare quello che fa.

Tasso dei fallimenti

Numero medio dei fallimenti di un certo dispositivo o di un sistema in un dato periodo di tempo.

Test

Verifica dinamica eseguita con dei valori in ingresso.

Test basato sui guasti

Test con lo scopo di rivelare delle specifiche classi di guasti.

Test deterministico

Forma di test in cui i pattern sono predeterminati da una scelta selettiva.

Test funzionale

Forma di test dove gli ingressi sono selezionati in accordo a dei criteri correlati con la funzione del sistema.

Test operazionale

Test eseguito per valutare la dependability di un sistema, con un profilo di ingresso rappresentativo delle sue condizioni operazionali.

Test random

Vedi test statistico.

Test statistico

Forma di test in cui i pattern sono selezionati in accordo ad una distribuzione di probabilità nel dominio degli input.

Test strutturale

Forma di test in cui gli ingressi sono selezionati in accordo a dei criteri correlati con la struttura del sistema.

Timestamp

Contatore che viene associato ad ogni evento in un sistema, sia centralizzato che distribuito. Il timestamp può essere generato leggendo il valore dello orologio del sistema al momento in cui è avvenuto l'evento.

TMR

Ridondanza triplice modulare: si triplica lo hardware e si esegue un voto a maggioranza per determinare l'uscita del sistema.

Tolleranza ai guasti

Metodi e tecniche con lo scopo di fornire un servizio che si addice alle specifiche anche in presenza di guasti.

Totally self checking

Un circuito si dice totally self checking se è self testing e se fault secure.

Transazione

Identifica una unità elementare di lavoro svolta da un programma applicativo, cui si vogliono associare particolari caratteristiche di correttezza, robustezza e isolamento.

Trattamento dei guasti

Le azioni intraprese per prevenire che i guasti possano essere riattivati.

Trustability

Capacità del sistema di fornire agli utenti delle informazioni circa la correttezza del servizio.

Undependability

Proprietà di un sistema per cui non si può o non si potrà più riporre fiducia in modo giustificabile sul servizio fornito dal sistema.

Undo

Ricostruzione della situazione esistente all'inizio di una transazione, *disfacendo* il lavoro svolto dalle istruzione della transazione eseguite fino a quel momento a causa di un errore

Utente del sistema

Un altro sistema, fisico o umano, che interagisce con il sistema considerato.

Validazione

Metodi e tecniche per far raggiungere ad un sistema l'abilità di fornire un servizio che si addice alle specifiche.

Verifica

Il processo di determinare se un sistema aderisce a certe proprietà (le condizioni di verifica) che possono essere generali, cioè indipendenti dalla specifica oppure specifiche, cioè dedotte dalla specifica.

Verifica di regressione

Verifica eseguita dopo una correzione, per testare che la correzione non abbia conseguenze non desiderate.

Verifica dinamica

Verifica che coinvolge l'esercizio del sistema.

Verifica statica

Verifica eseguita senza esercitare il sistema.

Indice delle figure e delle tabelle

Bibliografia

[1] D. P. Siewiorek, R. S. Swarz “Reliable Computer System – Design and Evalution”, Digital Press

[2] A. S. Tannebaum “Modern Operating Systems”, Prentice Hall

[3] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone “Basi di dati Concetti, linguaggi e architetture”, McGraw-Hill

[4] A. Albano “Basi di dati Strutture e algoritmi”, Addison-Wesley

[5] D. A. Patterson, J. L. Hennessy “Struttura e progetto dei calcolatori”, Zanichelli