Anwendung von verteilem PEARL zur ausfallsicheren Datenerfassung.

E. Heilmeier, P. Holleczek, M. Trautner

Regionales Rechenzentrum Erlangen
der Universität Erlangen-Nürnberg

1. Einleitung

Die Erfassung unwiederbringlicher Daten erfordert ausfallsichere Rechnerkonfigurationen. Ausfallsicherheit kann durch redundante Hardware erreicht werden.

In einer Zeit der billigen Mikrorechner bietet es sich an, die zur Erfassung vorgesehenen Rechner mehrfach auszulegen und als Verteiltes System zu betrachten.

Verteiles PEARL [1] unterstützt solche Rechnerkonfigurationen und erlaubt eine bequeme Programmierung. Das soll am Beispiel der Erfassung von Telefongesprächsdaten an der Universität Erlangen gezeigt werden.

2. Steigerung der Zuverlässigkeit

Nach [2] soll unter Betriebszuverlässigkeit verstanden werden, daß der Rechner weder etwas Falsches tut, noch daß er die von ihm erwartete Funktion überhaupt nicht ausführt.

Daraus ergeben sich drei unterschiedliche Forderungen:

1. Der Rechner soll keine falschen Befehle an den Prozeß oder falsche Anweisungen an das Bedienpersonal geben.
2. Unwiederbringliche Daten sollen zerstörungssicher gespeichert werden.
3. Der Rechner soll nicht länger als eine vom Prozeß zulässige Zeit außer Funktion sein.

Der erste Punkt erfordert eine zuverlässige Programmierung mit möglichst vielen Korrektheitsüberprüfungen. Die wichtigsten Anordnungen sollten auf mehreren unabhängigen Wegen berechnet werden und nur dann erteilt werden, wenn sie in allen Fällen übereinstimmen.

Um die zweite Forderung zu erfüllen, ist es ratsam, alle wichtigen Daten auf zwei unabhängigen Datenträgern zu speichern. Probleme, die sich daraus ergeben, werden in der Literatur unter dem Begriff der "Datensicherheit" behandelt.

Redundante Systeme sind in der Lage, den Ausfall von Komponenten so aufzufangen, daß eine korrekte Arbeitsweise weiterhin aufrecht erhalten werden kann, gegebenenfalls mit verminderter Leistungsfähigkeit.

Eine Steigerung der Zuverlässigkeit ist durch

- Fehlerintoleranz und/oder
- Fehlertoleranz

zu erreichen.

Fehlerintoleranz, die Möglichkeit Fehler von vornherein zu vermeiden, wird hardwaremäßig durch Qualitätssteigerung der Technologien und bezüglich der Software durch Verfahren der Programmverifikation ermöglicht.

Techniken der Fehlertoleranz gehen von der "pessimistischen" Annahme aus, daß Fehler

dennoch nie ganz zu vermeiden sind und daher auch zur Laufzeit des Systems aufgefangen werden müssen.

Fehlertoleranztechniken mit Redundanz sind geeignet, auf Hardwareausfälle und Programmierfehler zu reagieren. Nach [3] lassen sich zwei Arten von Hardware-Redundanztechniken unterscheiden:

- statische Hardware-Redundanz und
- dynamische Hardware-Redundanz.

Statische Hardware-Redundanz gewährt eine Fehlermaskierung. Ein Systemausfall wird somit derart verhindert, daß eine ununterbrochene Fortsetzung der Ausführung gewährleistet ist. Statische Redundanz deshalb, weil trotz interner Ausfälle das Systemverhalten nach außen "statisch" erscheint.

Kennzeichen der statischen Hardware-Redundanz ist die sogenannte "funktionsbeteiligte Redundanz". Darunter versteht man eine Redundanz, bei der die zusätzlichen Mittel nicht nur ständig in Betrieb, sondern auch an der vorgesehenen Funktion beteiligt sind.

Merkmal der dynamischen Redundanz ist, daß das System bei einem Ausfall einer Systemkomponente aktiv ("dynamisch") reagiert und Ersatzleistung bereitstellt.

Bezüglich dynamischer Hardware-Redundanztechniken unterscheidet man zwischen
- "stand-by"-Verfahren und
- "fail-soft"-Verfahren.

Bei der "stand-by"-Strategie wird bei einem Ausfall des aktiven Moduls automatisch auf einen Reservemodul (engl. spare) umgeschaltet. Es handelt sich hierbei um eine sogenannte "nicht funktionsbeteiligte Redundanz", da die zusätzlichen technischen Mittel erst bei einer Störung die vorgesehene Funktion übernehmen. Nach [3] sind aber auch Varianten der "stand-by"-Technik möglich, bei denen die redundanten Module ständig mitlaufen und im Fehlerfall nur die Ausgangssignale umgeschaltet werden.

Neben dem "stand-by"-Verfahren gehört auch die "fail-soft"-Strategie zu den dynamischen Hardware-Redundanztechniken. Diese Methode ermöglicht abgestufte Fehlertoleranz, indem bei der Störung eines Moduls oder eines Prozessorausfalls bereits vorhandene Module gleichen Typs die Aufgabe des defekten mit übernehmen.

Damit in einem dynamischen hardware-redundanten System ein Ausfall toleriert werden kann, müssen folgende Punkte automatisch erledigt werden:

a) Diagnose:
   Der Ausfall eines Moduls muß erkannt und der fehlerhafte Modul lokalisiert werden.
b) Rekonfiguration:
   Ersatzleistung muß bereitgestellt und die durch den Fehler betroffenen Aufgaben müssen neu verteilt werden.
c) Wiederanlauf:
   Die Programme, die durch den Ausfall unterbrochen wurden, müssen mit konsistenten Daten wiederaufgesetzt werden.

[4] unterscheidet bezüglich Diagnoseverfahren zwischen

- Selbstdiagnose:
  Selbsttestprogramme erkennen Fehler innerhalb eines Bausteins.
- Nachbarschaftsdiagnose:
  Um eine automatische Fehlerbehandlung durchführen zu können, müssen in einem Multiprozessorsystem auch die übrigen Prozessoren über einen Ausfall informiert sein.
- Systemweite Selbstdiagnose (verteilte Diagnose):
  Das fehlertolerante Multiprozessorsystem muß in der Lage sein, fehlerhafte Systemkomponenten automatisch zu erkennen und zu lokalisieren.

Bei einem von [3] geführten Vergleich zwischen statischen und dynamischen Hardware-Redundanztechniken, bieten die statisch redundanten Systeme gegenüber der dynamischen Redundanz mehr Vorzüge. Dynamisch redundante Systeme haben den Vorteil, daß sie um eine vergleichbare Verfügbarkeit zu erzielen, weitaus weniger redundante Hardware benötigen.

3. Ausfallsichere Datenerfassung mit redundanten Rechnern

An einem konkreten Beispiel soll gezeigt werden, wie mit einfachen (Hardware-) Mitteln und einer komfortablen Programmiersprache die Zuverlässigkeit von Datenerfassungseinheiten gesteigert werden kann.

3.1. Allgemeines

Bei der Erfassung von Daten aus technischen Prozessen kann davon ausgegangen werden, daß Erfassungsaufgaben von den Verarbeitungsaufgaben getrennt auf verschiedene Prozessoren untergebracht sind. Will man sich vor Ausfällen schützen ("Datensicherheit"), besteht ein erster Schritt darin, lediglich den Erfassungsteil redundant auszulegen.

Gegen folgende Fehler will man sich dabei schützen:

- Ausfall der Verbindung zwischen Erfassungs- und Verarbeitungsrechner,
- Ausfall eines Erfassungsrechners.

Eine Maßnahme gegen den ersten Fall ist, den Erfassungsrechner mit eigener Speicherkapazität zu versehen, um den Ausfall zumindest überbrücken zu können.

Eine Maßnahme gegen den zweiten Fehler ist, den Erfassungsrechner mit seinem lokalen Speicher mehrfach auszulegen.

Selbst wenn die Umschalteinheit nur einfach vorhanden ist, kann jede der folgenden Komponenten einmal ausfallen, ohne den Betrieb zu beeinträchtigen:

- Erfassungsrechner,
- lokaler Speicher,
- Verbindung zum Verarbeitungsrechner.

3.2. Die Erfassung von Telefongesprächsdaten

Für die Abrechnung der Telefongebühren an der Universität wurden bisher die wichtigsten Kenndaten jedes Gesprächs auf Lochstreifen aufgezeichnet. Alle Telefonapparate der Universität, die Anschlüsse der Privatpatienten der Universitätskliniken eingeschlossen, sind an die Telefonzentrale, einer Einrichtung der Universität, angeschlossen. Hier wird für abgehende Ferngespräche über ca. 50 Amtsleitungen eine Verbindung zum Telefonnetz der Deutschen Bundespost hergestellt. Kennzeichnende Daten eines abgeschlossenen Telefongesprächs wie die Nummer des Anrufers, die Nummer des Empfängers, Datum und Uhrzeit des Gesprächsbeginns, Einheiten etc. wurden bisher über drei angeschlossene Lochstreifenstanzer auf Lochstreifen ausgegeben (siehe Abb. 1).

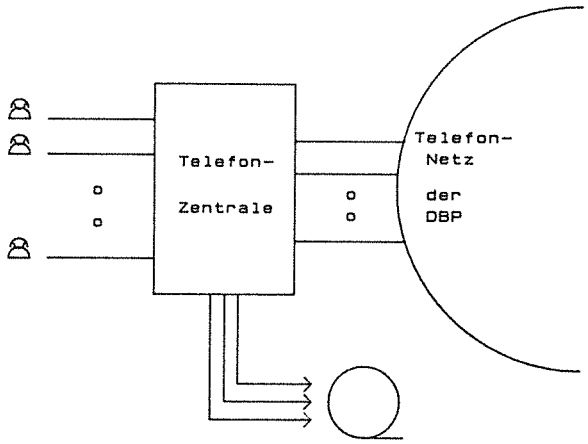

Abb. 1 Konventionelle Erfassung der Telefongespraechsdaten

Da die Verwendung eines Lochstreifens als Aufzeichnungsmittel überholt ist, bot sich als Datenträger die Diskette an. Zum anderen sollte der Datentransport automatisch erfolgen.

Die Kenndaten abgeschlossener Telefongespräche sollen somit anstatt auf Lochstreifen auf Floppy gespeichert und täglich zum Verwaltungsrechner übertragen werden. Für die Gebührenerfassung, Speicherung und Übertragung der Daten zum Verwaltungsrechner sind Mikrorechner vorgesehen.

Die Kenndaten abgeschlossener Telefongespräche kommen von drei unabhängigen Leitungen an. Bevor sie zur Speicherung auf Floppy weitergegeben werden können, müssen die parallel ankommenden Datensätze in sequentielle Form gebracht werden. Die gespeicherten Kenndaten werden automatisch zu vorgegebenen Zeitpunkten oder auf Wunsch des Bedienperso-

nals an den Verwaltungsrechner gesendet. In monatlichen oder vierteljährlichen Abständen wird am Verwaltungsrechner die Abrechnung der Gesprächsdaten durchgeführt.

Hauptziel ist, die Ausfallsicherheit des Gesamtsystems zu erhöhen, um eine kontinuierliche Erfassung und zuverlässige Speicherung der Daten zu gewährleisten.

3.2.1. Lösungsansatz

Für die "on-line"-Erfassung, Speicherung und Übertragung der Kenndaten abgeschlossener Telefongespräche ist ein Mehrrechnersystem, bestehend aus vier Mikrorechnern vom Typ Z80, eingesetzt (siehe A b b . 2). Insgesamt sind für die Aufnahme drei unabhängige Datenleitungen vorgesehen, um bei Hochbetrieb die ankommenden Daten quasi gleichzeitig erfassen und speichern zu können.

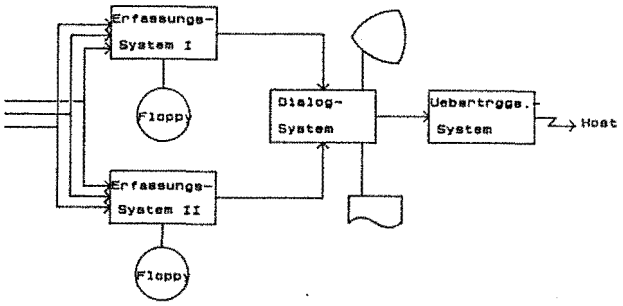

Abb. 2 Ausfallsichere Konfiguration zur Erfassung der Telephongespraechsdaten

Ein Z80-System, das speziell für die Erfassung vorgesehen ist, nimmt die ankommenden Daten entgegen und speichert diese in einer Datei auf Diskette. Gleichzeitig eintreffende Datensätze müssen vor der Weiterbearbeitung erst sequentialisiert werden.

Damit bei einem eventuellen Ausfall des Erfassungssystems (z.B. Stromausfall, Programmfehler, Spurfehler auf der Diskette) die Datenerfassung nicht zum Erliegen kommt, ist ein zweites Erfassungssystem vorhanden, das synchron zum anderen die ankommenden Daten erfaßt und auf einem eigenen Datenträger abspeichert. Mit dem Einsatz des redundanten Systems wird eine erhebliche Verbesserung der Betriebssicherheit erzielt. Eine wichtige Voraussetzung für den Anschluß einer redundanten Hardware-Einrichtung ist, daß beide Systeme von verschiedenen Stromquellen versorgt werden. Die Bedienung der Anlage ist von einer speziellen Komponente des Gesamtsystems, dem Dialogsystem, aus möglich.

Außer der Kommunikation zwischen Bedienpersonal und Gesamtsystem hat das Dialogsystem ferner die Aufgabe der Überwachung. Ein Ausfall jedes der drei restlichen Komponenten wird von diesem System erkannt und gemeldet.

Für die Übertragung der auf Floppy gespeicherten Telefongesprächsdaten zum Verwaltungsrechner der Universität ist eine eigene Systemkomponente vorgesehen.

Da die Kapazität einer Diskette nicht für die Datenaufnahme eines Monats oder länger ausgerichtet ist, wird eine Übertragung täglich automatisch gestartet.

3.2.2. Maßnahmen zur Steigerung der Zuverlässigkeit

Durch den modularen Aufbau der Funktionen jedes Teilsystems der Anlage ist ein wichtiger Grundstein für die Fehlertoleranz des Gesamtsystems gelegt.

Ein Ausfall der Übertragungskomponente, die im Normalfall nur einmal am Tag aktiviert wird, ist wegen des funktionalen modularen Aufbaus nicht weiter tragisch. Das defekte System kann in der Zwischenzeit repariert und ohne großen Datenverlust bei der Erfassung wieder dazugeschaltet werden.
Wie aus den theoretischen Ausführungen zur Fehlertoleranz im vorigen Kapitel hervorgeht, handelt es sich bei der verwendeten Methode um eine Mischform zwischen statischer und dynamischer Hardware-Redundanz mit "stand-by"-Verfahren, wobei der Trend in letztere Richtung geht.

Der einzige Unterschied zur beschriebenen "stand-by"-Strategie als dynamische Hardware-Redundanz besteht darin, daß im vorliegenden Modell nicht erst bei einem Ausfall das redundante System zugeschaltet wird, sondern, wie als Variante der "stand-by"-

Technik in [3] vorgeschlagen, bei der Datenerfassung ständig mitläuft.

Ein Ausfall der Erfassungssysteme und/oder des Übertragungssystems wird vom Dialogsystem entsprechend der Nachbarschaftsdiagnose [4] erkannt und gemeldet. Eine Störung des Dialogrechners und ein darauffolgender Ausfall eines Erfassungssystems hat zur Folge, daß der zweite Ausfall vom bereits defekten Dialogsystem nicht gemeldet werden kann. In diesem Fall wird die Störung des Dialogsystems und die des Erfassungssystems auf zwei unterschiedliche Leuchtdioden am intakten Erfassungsrechner angezeigt.

Der modulare Aufbau garantiert selbst eine kontinuierliche Datenaufnahme bei einem Ausfall aller Komponenten, bis auf einen Erfassungsrechner, der als einziges funktionsfähiges System die ankommenden Daten annimmt und abspeichert.

3.3. Das Programm

Das Programm wurde in PASS [5], [6] spezifiziert und in Verteiltem PEARL programmiert. Es umfaßt 18 Tasks mit ca. 4500 Quellzeilen und läuft auf vier gekoppelten Z80-Rechnern. Der redundante Teil des Codes beträgt ca. 1100 Zeilen.

Der Erstellungsaufwand mit Entwurf, Spezifikation und Programmierung umfaßte 3/4 Mann-Jahr.

4. Möglichkeiten von Verteiltem PEARL

Die Programmierung von verteilten Systemen stellt hohe Anforderungen an die Programmiersprache. Die Echtzeitprogrammiersprache PEARL gewährleistet als "verteiltes PEARL" [1] eine Kommunikation zwischen Prozessen, die auf demselben als auch auf verschiedenen Prozessoren laufen. Das Konzept basiert auf Botschaftsoperationen und nicht-deterministischen Kontrollanweisungen. Diese Konstrukte sind Erweiterungen von PEARL und sind im PEARL-Compiler und -Betriebssystem für Z80-Prozessoren integriert.

Mit Hilfe des Botschaftsmechanismus können Prozesse eines Prozeßsystems Botschaften austauschen und auf diese Weise miteinander kommunizieren. Im vorliegenden Fall bedeutet das einen Nachrichtenaustausch zwischen den Prozessen an den Erfassungssystemen, dem Dialog- und dem Übertragungssystem. Die Störung eines Prozessors kann insofern sofort festgestellt werden, als eine gesendete Botschaft von diesem nicht angenommen wird.

Im Zusammenhang mit dem Botschaftsmechanismus stehen die von Dijkstra [7] vorgeschlagenen nicht-deterministischen Kontrollanweisungen zur Verfügung. Mit diesen Konstrukten ist es möglich, auf das alternative oder gleichzeitige Eintreffen verschiedener Nachrichten von verschiedenen Prozessen zu warten.

Zur Überwachung der einzelnen Prozessoren ist auf jedem Rechner ein eigener Prozeß verantwortlich, der in vorgegebenen Zeitabständen eine Botschaft an den Überwachungsprozeß eines anderen Prozessors sendet bzw. erwartet. Dadurch, daß ein Prozeß auf eine Botschaft vom anderen Überwachungsprozeß und letzterer auf die Abnahme der gesendeten Nachricht wartet, ist eine gegenseitige Überwachung der Prozesse und somit der Prozessoren gewährleistet.

Durch die Einführung eigener Überwachungsprozesse, die zu fest vorgegebenen Zeitpunkten unabhängig von der Systembelastung die Aktivität aller Prozessoren überprüfen, wird eine Störung selbst dann erkannt, wenn alle Prozessoren "leerlaufen".

Bei der Erfassung der Telefongesprächsdaten werden diese an jedem der Erfassungssysteme in einer vorgesehenen Datei auf der Diskette gespeichert. Da ein Systemausfall oder das Auftreten eines Spurfehlers beim Beschreiben der Floppy nicht vorherzusehen ist, bedeutet dies einen Verlust aller erfaßten und gespeicherten Daten auf der eröffneten Datei. Um diesen Verlust an Daten möglichst gering zu halten, bietet sich als Lösung die Einführung vieler kleiner Dateien an. Jede einzelne Datei wird, sobald sie vollständig beschrieben ist, geschlossen und die Daten bleiben bei einem Fehler sicher erhalten.

Allein die Daten der zum Zeitpunkt des Fehlers eröffneten Datei sind verloren.

5. Erfahrungen

Die Programmierung verteilter Systeme in Echtzeitumgebung stellt - auch beim Einsatz einer höheren Programmiersprache - durchaus noch keine alltägliche Aufgabe dar. Insbesondere bei Problemen, die den Einsatz redundanter Hardware erfordern, sehen sich Bearbeiter vor schwierigen Entwurfsentscheidungen und Implementationshindernissen. Eine Programmiersprache mit einschlägigen Grundfunktionen, wie Botschaftsoperationen mit Zeitüberwachung (z.B. für Nachbarschaftskontrolle), Guarded Statements mit Verundung (z.B. für synchronen Gleichlauf von redundanten Prozessoren) stellt eine begriffliche Grundlage dar. Entscheidende konzeptionelle Hilfe kommt allerdings von einer auf die Möglichkeiten der Programmiersprache abgestimmten Spezifikationstechnik, wie z.B. PASS.

Literatur

[1] Fleischmann, A.; Holleczek, P.; Klebes, G.; Kummer, R.: Synchronisation und Kommunikation verteilter Automatisierungsprogramme. Angewandte Informatik 7/83, 290-297

[2] Bolch, G.: Prozeßautomatisierung mit Prozeßrechnern. Vorlesung. Erlangen 1980

[3] Maehle, E.: Fehlertolerantes Verhalten in Multiprozessoren. Untersuchungen zur Diagnose und Rekonfiguration. Dissertation. IMMD Erlangen 1982

[4] Maehle, E.: Experimente mit parallelen Programmen auf DIRMU Multiprozessor-Konfigurationen. Vortrag im Informatik-Kolloquium der Universität Erlangen-Nürnberg. 1985

[5] Fleischmann, A.; Holleczek, P.; Koch, I.; Kragl, G.: Eine Spezifikationstechnik für Verteilte Systeme. PEARL-Tagung 1984

[6] Andres, C.; Fleischmann, A.; Holleczek, P.; Hillmer, U. Kummer, R.: Eine Methode zur Beschreibung von Kommunikationsprotokollen. GI/NTG-Fachtagung Kommunikation in Verteilten Systemen 1985. Informatik-Fachberichte 95, Springer-Verlag

[7] Dijkstra, E.W.: Guarded commands, non determinacy and derivation of programs. ACM Computing Surveys. August 1975

Dr. Peter Holleczek
Regionales Rechenzentrum
Universität Erlangen-Nürnberg
Martensstraße 1
8520 Erlangen
Tel.: 09131/85-7031