Erfahrungen mit einer Programmiertechnik für Verteilte Systeme

P. Holleczek, Universität Erlangen

Einleitung

In Erlangen wurde ein Programmiersystem für Verteile Systeme von Mikroprozessoren entwickelt. Es wurde als Erweiterung der Programmiersprache PEARL realisiert. Über Konzept und Sprachelemente wurde bereits berichtet /FHKK82/, /AFHM82/.

Die Entwicklungen sind abgeschlossen, erste Erfahrungen im Umgang mit dem Programmiersystem liegen inzwischen vor. Außerdem wird der Themenkreis "Verteilte Systeme" verstärkt diskutiert, so auch im Rahmen dieser Veranstaltung /FH83/, /WB83/.

Es soll hier deshalb der Erlanger Ansatz noch einmal aufgegriffen werden. Im Einzelnen sollen das zugrundeliegende Konzept rekapituliert, Komponenten und Funktionen des Programmiersystems vorgestellt, erste Erfahrungen wiedergegeben und Perspektiven aufgezeigt werden.

Konzept

Das am RRZE realisierte Konzept und seine Grundlagen wurde ausführlich in /FHKK83/ dargestellt. Es soll sowohl die gedanklichen Grundlagen für eine Programmierung paralleler Prozesse in Verteilten Systemen liefern, als auch geeignete Sprachkonstrukte, als PEARL-Erweiterung, vorsehen.

Es faßt einige der gängigen Vorschläge der neueren Literatur zusammen und unterstützt in elementarer Form die in /FH83/ als wesentlich genannten Abstraktionsmechanismen und Strukturierungsmerkmale.

Zur Kommunikation von parallelen Prozessen sieht es zusätzlich zu den in PEARL bereits formulierbaren gemeinsamen Objekten einen Botschaftsmechanismus vor. Während das Mittel der gemeinsamen Objekte sinnvollerweise nur Prozessen auf dem gleichen Prozessor vorbehalten sein sollte, ist ein Botschaftsaustausch sowohl zwischen Prozessen auf verschiedenen, als auch auf dem gleichen Prozessor möglich. Dem empfangenden Prozeß kann auf Wunsch ein Wartebereich vorgeschaltet werden, der Botschaften endlich oft puffern kann. Auf diese Weise ist eine "synchrone" (als Redezvous) wie eine "asynchrone" Kommunikation möglich. (Der Sender bzw. der Empfänger muß nun warten, wenn kein Platz im Puffer frei ist, bzw. sich die erwartete Nachricht noch nicht im Puffer befindet.) Zum Austausch von Botschaften dienen die Operationen RECEIVE und TRANSMIT. Botschaften werden anhand ihres Namens unterschieden, sie können beliebige Parameter aufweisen. Die von einem Prozeß insgesamt zu sendenden bzw. zu empfangenden Botschaften müssen, um Konsistenzprüfungen zu ermöglichen, im Task-Kopf vereinbart werden.

Zur Vermeidung von Reihenfolgeproblemen (vergl. "Asynchronitätsabstraktion" in /FH83/) wurden sogenannte "nicht-deterministrische Kontrolloperationen" (guarded statements) eingeführt. Sie erlauben das Reagieren auf Bedingungen, die über "und" und "exklusives oder" verknüpft werden können. Solche Bedingungen können z.B. sein: Das Eintreffen einer erwarteten Botschaft, das Erhöhen einer Semaphore, das Eintreffen eines Interrupts.

Auf das Erfülltsein von Bedingungen kann eine endliche Zeit gewartet werden (durch "Guarded Regions"), es kann aber auch nur ihr Momentanwert abgefragt werden (über "Guarded Commands"). Im ersten Fall kann der aufrufende Prozeß blockiert werden, im zweiten Fall nicht.

Mit Hilfe von Guarded Regions können beispielsweise zeitüberwachte Botschaftsoperationen realisiert werden.

Das Programmiersystem

Das Programmiersystem besteht im wesentlichen aus den Komponenten Compiler, Betriebssystem, Netzwerksystem und Testsystem.

Der portable, in FORTRAN geschriebene, Compiler wurde um Botschaftsmechanismus und Guarded Statements erweitert. Die Erweiterungen konnten durch Hinzufügen weiterer Verarbeitungsschritte realisiert werden. Der Compiler läuft zur Zeit auf verschiedenen Wirtsrechnern (CYBER, VAX).

Das Betriebssystem mußte fast vollständig neu entwickelt werden. Es ist soweit modular aufgebaut, daß folgende Versionen konfigurierbar sind:

- Herkömmliches Betriebssystem (für Monoprozessoren).
- Betriebssystem mit Botschaften (für Verteilte Systeme).
- Betriebssystem mit Guarded Statements (für Monoprozessoren).
- Betriebssystem mit Botschaften und Guarded Statements (für Verteilte Systeme).

Alle Versionen enthalten als Spezialfall das herkömmliche PEARL-Betriebssystem. Das Betriebssystem ist in einer Systemsprache formuliert und fast vollständig portabel.

Das Netzwerksystem vereinigt in sich zwei Funktionen. Es enthält einen Ladeteil, der von einem willkürlich bestimmbaren Masterprozessor die Satelliten im Verteilten System mit Programmcode versorgt. Die Satelliten benötigen keinen Hintergrundspeicher (Floppy - Diskette etc.). Je nach Qualität des unterlagerten Transportsystems kann eine gesicherte oder ungesicherte Ladeprozedur gewählt werden.

Das Netzwerksystem enthält außerdem Leitungstreiber, die den Austausch der Botschaften zwischen den Betriebssystemen über die Übertragungs-Hardware abwickelt. Die Leitungstreiber werden automatisch an die jeweilige Hardware-Konfiguration des Verteilten Systems angepaßt. Die Informationen über die Konfiguration werden dem PEARL-Systemteil entnommen. Die Leitungstreiber können ebenfalls wahlweise eine Sicherungsprozedur vorsehen. Das Netzwerksystem ist wie das Betriebssystem in einer Systemsprache formuliert und weitgehend portabel.

Das Testsystem erlaubt den sprachorientierten Umgang mit den üblichen PEARL-Konstrukten (Auskünfte über bzw. Statusänderung von Variablen, Tasks, Semaphoren, Interrupts etc.), das Setzen von Haltepunkten, das Einschalten von Zeilentraces. Alle diese Operationen lassen sich auf einen Monoprozessor als auch in einem Verteilten System durchführen. So ist es beispielsweise möglich, vom Masterprozessor aus auf einem entfernten Prozessor, der über keine Bedienperipherie verfügt, testweise Prozeßinterrupts auszulösen oder einen Zeilentrace anzustoßen.

Das Testsystem gibt es, je nach verfügbarem Speicherplatz, in verschiedenen Versionen:

- Komprimierter oder ausführlicher Kommandointerpreter.
- Komprimierte oder ausführliche Protokollausgabe.

Das Testsystem ist weitgehend portabel (in PEARL und einer Systemsprache formuliert).

Das Programmiersystem baut auf folgender Hardware auf: Während der Compiler zur Zeit auf einem fremden Wirtsrechner läuft, wurden Betriebssystem und die restliche Laufzeitorganisation an den Z80-Mikroprozessor angepaßt. Die Kopplung zwischen verteilten Z80-Systemen erfolgt über private serielle Leitungen, ist aber auf andere Medien, z.B. ein lokales Netz, übertragbar. Der Zugang vom Netz zum Z80-Prozessor erfolgt über E/A (SIO) -Bausteine, die interruptgesteuert betrieben werden.

Erfahrungen

Erste Eindrücke liegen inzwischen zu der Eignung der entwickelten Systemsoftware und zum Einsatz der neuen Sprachkonstrukte vor.

Die Implementation der aufwendigen Sprachkonstrukte (Botschaftsmechanismus und Guarded Statements) gelangen in einem 18 kByte großen Betriebssystem (der Vergleichswert von 5 kByte für das herkömmliche Betriebssystem verdeutlicht den Aufwand für die neuen Konstrukte), das Netzwerksystem benötigt 2 kByte. Das sprachorientierte Testsystem belegt auf einen Monorechner 10 kByte, auf den unbedienten Prozessoren (Satelliten) im Verteilten System 6 kByte.

Bei ersten Einsätzen erwiesen sich folgende

Übertragungsgeschwindigkeiten als sinnvolle obere Grenze: Ca. 120 kBit/sec. in der Ladephase und 19 kBit/sec. in der Kommunikationsphase. Diese Grenzen sind durch die interruptgesteuerte Zeichenverarbeitung in den Leitungstreibern bedingt, die den Zentralprozessor nicht auslasten sollten. Um die dabei wirklich erreichbare Geschwindigkeit bei der Übertragung von Botschaften zu werten, muß berücksichtigt werden, daß auf 64 Oktetts Nutzdaten jeweils noch 18 Oktetts durch das zwischen den Betriebssystemen verwendete Protokoll kommen.

Bei der Verwendung der neuen Sprachelemente in der Anwendungsprogrammierung ergaben sich unerwarteterweise keine Probleme. PEARL-Neulinge (Informatik-Studenten) waren, nach kurzer Einführung, auch angesichts komplexer Probleme in der Lage, korrekte Programme zu schreiben. Wertvolle Hilfe leistete dabei auch das Testsystem.

Zwei Probleme wurden aber offensichtlich:

- Um den Einsatz von Botschaftsmechanismen und Guarded Statements bereits im Vorfeld der Programmierung zu unterstützen, ist ein geeignetes Spezifikationsverfahren vonnöten.

- Das Konfigurieren einer für eine bestimmte Anwendungsklasse geeignete (Betriebs-) Systemsoftware-Version erfordert einen Systemspezialisten und ist nicht durch den Anwendungsprogrammierer vollziehbar.

Perspektiven

Verschiedene Konsequenzen wurden aus den vorliegenden Erfahrungen gezogen. Sie betreffen die Hantierung des Verteilten Systems, den Nachfolgeprozessor und die Spezifikationstechnik.

Die Konfigurierung einer für eine Anwendungsklasse maßgeschneiderten Systemsoftware war verbesserungswürdig. Es wurde deshalb ein Generator in Angriff genommen, der nur die gerade notwendigen Komponenten zusammenstellt. Dies gilt vor allem für das Betriebssystem. So wird aus dem Benutzerprogramm entnommen, ob es sich um eine Ein- oder Mehrrechner-Konfiguration handelt, ob Botschaften verwendet werden etc. Darüberhinaus werden dem Programmierer nur wenige Angaben abverlangt, wie z.B. der Wunsch nach gesicherter Übertragung durch die Leitungstreiber. Auf diese Weise soll der Einsatz eines "Verteilten Programmiersystems" dem Anwendungsprogrammierer möglichst nahe gebracht werden.

Unumgänglich ist außerdem der Übergang auf einen leistungsfähigeren Prozessor. Während bisher, wenn es sich nicht gerade um Einzelanwendungen handelte, in der Regel 8-bit-Prozessoren erschwinglich waren, kommen nun auch 16-bit-Mikroprozessorsysteme in Frage. Es wurde deshalb begonnen, die vollständige Systemsoftware auch an den Mikroprozessor Motorola 68000 anzupassen. Das ist wegen des hohen Anteils portabler Komponenten nicht allzu aufwendig. Die Arbeiten für die Einprozessor-Version sind nahezu abgeschlossen. Wegen des hohen Adressiervolumens des 68000 wird damit der Einsatzbereich des in Erlangen entwickelten Verteilten Programmiersystems auch für komplexe Anwendungen eröffnet.

Zur Unterstützung des Anwendungsprogrammierers wurde ein Spezifikationsverfahren für parallele Prozesse entwickelt, das auch Botschaftsmechanismen und Guarded Statements brücksichtigt. Hierüber wird in Kürze berichtet werden.

Nicht zuletzt soll aber, wie vor Jahresfrist an dieser Stelle bereits geschehen, zu einer Diskussion über PEARL-Sprachmittel für Verteilte Systeme aufgerufen werden, da sich sonst begonnene Einzelentwicklungen zu stark verfestigen.

Literatur

FHKK82 A. Fleischmann, P. Holleczek, G. Klebes, R. Kummer: Ein Mechanismus zur Kommunikation für Verteilte Systeme in PEARL. PEARL-Rundschau 1982, Band 3, Nr. 5

AFHM82 C. Andres, A. Fleischmann, P. Holleczek, W. Mühlbauer: Ein PEARL-Testsystem zum Einsatz in Verteilten Systemen. PEARL-Rundschau 1982, Band 3, Nr. 5

WB83 W. Bockhoff:
Ein Botschaftensystem mit PEARL-Schnittstellen zur Kommunikation in Verteilten Systemen.
Beitrag zu dieser Tagung

FH83 F. Hofmann:
Koordinierung und Verteilte Systeme.
Beitrag zu dieser Tagung

FHKK83 A. Fleischmann, P. Holleczek, G. Klebes, R. Kummer:
Synchronisation und Kommunikation Verteilter Automatisierungsprogramme.
Angewandte Informatik 7/83

Alle weiteren Zitate finden sich in FHKK83.

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