Methoden der Simulation auf Parallelrechnern unter SIMUL_R

von R. Ruzicka, Wien

Die Simulationssprache SIMUL_R wurde auf verschiedenen Multi-Chip-Rechnern mit unterschiedlichen Wegen der Parallelisierung implementiert. SIMUL_TR ist eine auf Transputern basierende Implementierung mit Parallelisierung in Teilmodellen. SIMUL_XR geht den Weg der Parallelisierung über dem Parameterraum, was sich besonders für Parametervariationen eignet (starten von Simulationsläufen "im Hintergrund").

1. Einleitung

Die Simulationssprache SIMUL_R für kontinuierliche und diskrete Systeme ist aufgrund ihrer offenen Systemarchitektur und ihrer Basissprache C sehr einfach auf neuen Computersystemen implementierbar [1].

Die Möglichkeit der Verwendung mehrerer Modelle in einem Simulationsprogramm und des alternativ sequentiellen und/oder parallelen Berechnens der Teilmodelle bietet schon auf Single-Chip-Computern für den Benutzer die Methoden der parallelen Simulation [2].

2. Parallele Simulation

Die Komplexität heutiger Probleme führt oft zu sehr großen Wartezeiten bei deren Simulation und verhindert manchmal ganz die Lösung von Simulationsaufgaben.

Da Rechner mit höherer Leistung oft unerschwinglich sind, liegt der Weg nahe, sich der Kapazität mehrerer (eventuell auch kleinerer) Computer, zu bedienen, also parallel zu rechnen.

Zwei Wege der Parallelisierung bieten sich natürlicherweise an:

- die Parallelisierung mehrerer Module eines dynamischen Systems. Dies ist insbesondere dann von Nutzen, wenn ein System (etwa eine Fabrik) sehr einfach eine parallele Struktur von Teilmoduln erkennen läßt (etwa die einzelnen Maschinen).

- die Parallelisierung über dem Parameterraum, d.h. ein System wird als ganzes auf einem Prozessor berechnet; es wird jedoch parallel mit unterschiedlichen Parametern auf mehreren Prozessoren gerechnet.

Beide Zugänge haben Vor- und Nachteile.

Bei der ersten Methode wird bei einem einfachen Simulationslauf aufgrund der *Modellteilung ein zeitlich besseres Resultat* erreicht. Möchte man ein Modell alleine auf einem Prozessor wie beim zweiten Weg simulieren, so bleibt die Kapazität anderer Prozessoren ungenützt.

Umgekehrt erreicht man die *optimale Nutzung* der vorhandenen Prozessoren nur beim zweiten Weg, da hier die Kommunikation zwischen Modellen wegfällt.

Während beim ersten Verfahren zu jedem Rechenschritt Daten ausgetauscht werden müssen, was im Extremfall (je nach Modell) zum Verlust sämtlicher Zeitgewinne aufgrund einer höheren Anzahl von verwendeten Prozessoren führen kann, ist im zweiten Fall eine beinahe lineare Effizienzsteigerung mit der Anzahl der Prozessoren festzustellen (im ersten Fall ist natürlicherweise für ein konkretes Modell die sinnvolle Anzahl der eingesetzten Prozessoren von der Modularisierbarkeit des Modells abhängig).

3. SIMUL_TR - die SIMUL_R-Transputer-Version

In einem Projekt wurde SIMUL_R auf den Transputersystemen T2400 von Impuls (dieses wird an einen PC angeschlossen) und Thema-Workstation von HEMA (vereinigt PC und modular erweiterbare Transputer-Karten) zur Version SIMUL_TR implementiert [3].

Hierbei hilft die Multi-Modell-Fähigkeit von SIMUL_R: Modelle werden als parallel zu berechnend festgelegt und laufen nun wirklich simultan ab.

3.1 Das SIMUL_TR-System

Abbildung 1 zeigt eine Beispiel-Konfiguration für ein SIMUL_TR-System (diese Konfigurationen sind nicht fest vordefiniert, sondern lassen sich an die jeweilige Hardware - Prozessoren und Kommunikationskanäle - optimal anpassen).

Der Runtime-Interpreter mit der Benutzerschnittstelle arbeitet auf einem PC. Das heißt, der Benutzer arbeitet unter der gewohnten PC-Umgebung, erstellt hier sein Modell und verwendet die von SIMUL_R her bekannten Befehle. Sämtliche Datenspeicherung wird auf dem PC vorgenommen.

Die Transputer sind untereinander und/oder mit dem PC verbunden. Auf jedem läuft ein Kommunikationstask, der sämtlichen Datenaustausch übernimmt, und ein Simulationstask, der je nach Kommando durch den PC ein oder mehrere Teilmodelle simuliert.

Der Benutzer braucht (abgesehen von der bei der Installation durchzuführenden Konfiguration des Systems) nur die Anzahl der vorhandenen Prozessoren zu wissen und sich nicht weiter um die dahinter steckende Hardware zu kümmern.

3.2 Simulieren unter SIMUL_TR

SIMUL_TR ist ein nur zum Teil compiliertes Simulationssystem:

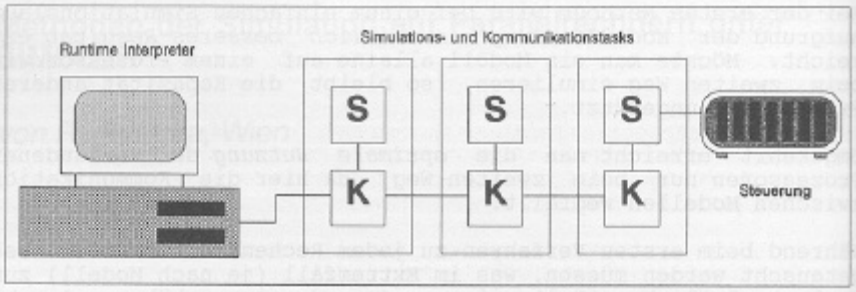

Abbildung 1 Das SIMUL_TR-System.

Wenn ein neues Modell erstellt ist und simuliert werden soll, wird es zuerst vom SIMUL_TR-Compiler übersetzt. Dieser erzeugt ein C-Programm für die Transputer, die dann mit dem übersetzten und gelinkten Programm geladen werden, und eine Definitionsdatei für den Runtime-Interpreter, der mit dessen Hilfe intern neue Datenstrukturen aufbaut.

Beispiel 1:

Drei Modelle (z.B. Module eines Systems) mod_1, mod_2, mod_3 sollen gleichzeitig ablaufen. Dies wird in SIMUL_TR durch

```
mstart mod_1:1, mod_2:4, mod_3:5;
```

(mod_1 auf Transputer 1, mod_2 auf 4, mod_3 auf 5) erreicht. Alle Probleme der Kommunikation (also wann welche Daten auszutauschen sind) werden automatisch und dynamisch selbständig von SIMUL_TR gelöst.

3.3 SIMUL_TR und die Umwelt

Da Transputer im wahrsten Sinne des Wortes sehr kommunikativ sind, liegt es nahe, andere Geräte an diese anzuschließen. Dies wird in SIMUL_TR auf Softwareseite durch spezielle Schnittstellen unterstützt. Abbildung 1 zeigt als Beispiel eine Steuerung.

4. SIMUL_XR - die parallele X-Windows-Version

SIMUL_XR ist die X-Windows-Implementierung von SIMUL_R. Auf Parallelrechnern unterstützt sie parallele Simulation über dem Parameterraum.

4.1 Window-Technik

SIMUL_XR bietet eine Oberfläche unter UNIX und X-Windows bzw. PostScript (z.B. auf COGENT XTM; dies ist ein Parallelrechner unter einer UNIX-ähnlichen Oberfläche, dessen parallele Konzeption unter Linda [4] für Prozessoren unterschiedlichster Bauart geeignet ist).

SIMUL_XR bietet die Möglichkeit zur Verwendung mehrerer Fenster zur Darstellung von Zeichnungen und Bewegungsabläufen. Dadurch können Zeichnungen sehr einfach "aufgehoben" und später verglichen werden.

Runtime-Interpreter-Befehle erlauben sämtliche Fensteroperationen, wie Öffnen, Schließen, Verschieben, Vergrößern/Verkleinern, wodurch automatisiert Simulationsexperimente durchgeführt und ansprechende Präsentationen geschaffen werden können.

4.2 Das parallele SIMUL_XR-System

SIMUL_XR bietet parallele Simulation im Sinne der ehenden zweiten Definition.

In diesem Fall kommt man also insbesondere b rameter-variationen und Optimierungen schneller ans Zi ieselben Modelle werden mit unterschiedlichen Parametern a schiede-nen Prozessoren gestartet - dies ermöglicht linearen Zeitgewinn mit der Anzahl der Prozessoren!

Es können aber auch unterschiedliche Modelle (z.B. verschiedene Modellierungen ein und desselben dynamischen Systems) parallel gestartet und damit öfter Modellvergleiche durchgeführt werden.

Abbildung 2 zeigt die Struktur von SIMUL_XR auf Parallelrechnern. Der Runtime-Interpreter wird wie eine UNIX-Shell verwendet, um Simulationstasks zu starten.

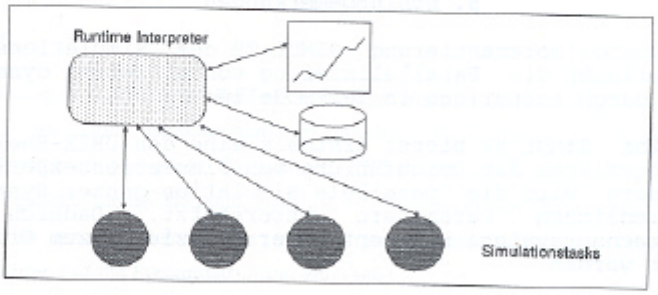

Abbildung 2 Die Struktur von SIMUL_XR auf Parallelrechnern.

Beispiel 2:

Im Sinne von UNIX kann einfach durch

```
start & ;
```

ein Simulationslauf *im Hintergrund* gestartet werden. Wenn dieser beendet ist, holt man seine Resultate ab.

Die in Abbildung 2 beschriebenen Simulationstasks und die für sie bestimmten bzw. von ihnen berechneten Daten können durch Setzen der Systemgröße *task* angesprochen werden.

Wird das erste Mal ein Simulationstask angesprochen, so wird er automatisch auf einem Prozessor (dessen Nummer angegeben oder selbständig von SIMUL_XR vergeben werden kann) gestartet.

Spezielle Befehle ermöglichen die Ausgabe von Informationen über Tasks, das Abbrechen von im Hintergrund gestarteten Simulationsläufen und der Simulationstasks selber, sowie das Zusammenführen von in unterschiedlichen Simulationstasks berechneten Daten, um damit gemeinsame Berechnungen durchführen und einheitliche Zeichnungen erstellen zu können.

Beispiel 3:

Im folgenden wird eine Parametervariation für die Werte 0, 0.1, ..., 1 durchgeführt und 11 Tasks gestartet. Der *wait*-Befehl mit dem Argument -1 wartet, bis alle Simulationsläufe beendet wurden. Am Ende werden die Resultate abgeholt und angezeigt.

```
task=0;
#for x0=0,1,0.1#
   send x0;   start &;   task = task+1;
#end
wait -1;
#for task=0,10#
   receive x;   disp task,x;
#end
```

5. Schlußbemerkungen

Die Transputerimplementierung SIMUL_TR des Simulationssystems SIMUL_R erlaubt die Parallelisierung von einzelnen dynamischen Systemen durch Aufteilung in Teilmodelle.

Die Version SIMUL_XR bietet erstmals eine den UNIX-Shells ähnliche Oberfläche zur Durchführung von Simulationsexperimenten, insbesondere wird die parallele Simulation ganzer Systeme mit unterschiedlichen Parametern unterstützt. Dadurch können Parallelrechnersysteme mit optimaler Effizienz zum Simulieren verwendet werden.

6. Literatur

[1] R. Ruzicka: SIMUL_R - Eine Simulationssprache mit speziellen Befehlen zur Modelldarstellung und -analyse, Proc. des 5. Symposium Simulationstechnik, Aachen, Sept. 1988, Springer, S. 212-217

[2] R. Ruzicka: Environments for SIMUL_R, Proc. of the 3rd European Simulation Congress, Edinburgh, Sept. 1989, S. 268-274

[3] Inmos: Transputer Reference Manual, Prentice-Hall, 1988

[4] A. Ahuja, N. Carriero, D. Gelernter: Linda and Friends, IEEE Computer, Vol. 19(1986), Nr. 8, S. 26-34