Ausarbeitung von MARS (Maintainable Real-Time System)

Konzepte von Betriebssystemkomponenten

Matthias Felix
filo@andariel.informatik.uni-erlangen.de

3. juli 2006

1 Einführung

1.1 Echtzeitsysteme

Ein Echtzeitsystem ist nicht, wie umgangssprachlich häufig verwendet, ein Konzept, bei dem man die Verzögerungszeit (nahezu) nicht merkt. Je nach Problem kann es auch Stunden oder Tage dauern, bis eine Reaktion erwartet wird. Das Wichtige ist, dass es nach einem äußeren oder inneren Ereignis in einer vorher fest vorgegebenen Zeit die korreckte Antwort auch unter Volllast bringt! Dabei kommt es nicht so sehr darauf an, dies mit einer möglichst hohen Performance zu erledigen.

Man unterscheidet vor allem zwischen zwei Arten von Echtzeit:

- Weiche Echtzeit:

  Hierfür gibt es viele verschiedene Definitionen, die alle darin übereinstimmen, dass man einen Näherungswert angibt, nach welcher Zeit das Ergebnis vorliegen sollte. Es hat aber keine schwerwiegenden Folgen, wenn diese Zeit auch beträchtig überschritten wird. Da dies nicht schwer zu realisieren ist, wird dies kaum weiter behandelt.

- Harte Echtzeit:

  Sollte hierbei einmal das Zeitintervall zwischen einer Anfrage bzw. einem Ereignis und die daraus resultierende Antwort einmal überschritten werden, können schwerwiegende Folgen auftreten, die bis zum Ausfall des gesamten Systems führen können. Dies muss natürlich unter allen Umständen verhindert werden.

Harte Echtzeit kann man mit einem ereignisgesteuertem oder einem zeitgesteuertem System realisieren.

- Ereignisgesteuert:

  Es wird sofort auf ein von außen eintretendes Ereignis reagiert, wobei die momentan laufende Verarbeitung unterbrochen wird. Dies führt zu einer schnellen Abarbeitung. Sollten allerdings mehrere Einflüsse auf einmal auftreten, kann es zu Komplikationen kommen. Es könnte etwa ein Ereignis niemals abgearbeitet werden, da die neu Ankommenden dieses dauernd wieder vertreiben. Daher kann man die Bearbeitungszeit nicht mit Sicherheit vorhersagen.

- zeitgesteuert: es wird nach einem vorher festgelegtem Zeitplan auf jedes Ereignis reagiert. Auch wenn der Planungsaufwand hierbei deutlich höher ist, kann man dafür das maximal benötigte Zeitfenster im vorhinein angeben. Daher verwendet auch MARS diese Art der Steuerung.

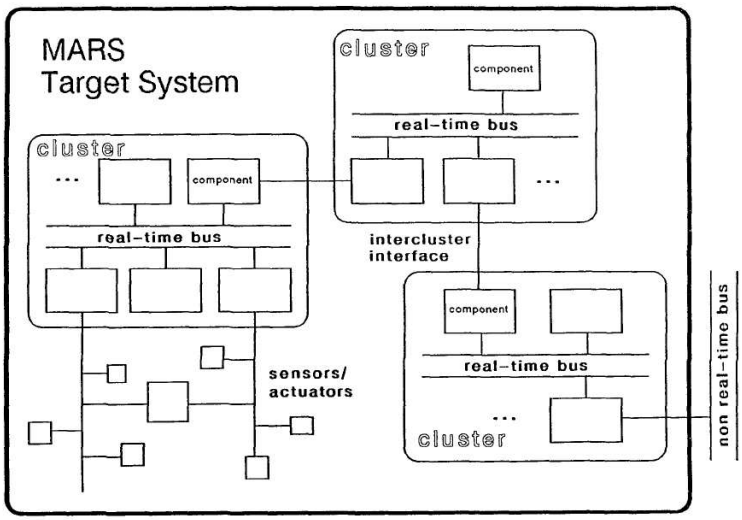
2 MARS (Maintable Real-Time System)

MARS ist ein fehlertolerierendes Echtzeitsystem für prozesskontrollierende Anwendungen. Es war das erste System, das als Hauptziel die exakte Einhaltung der harten Echtzeit hatte, und auch unter Volllast komplett deterministisch sein wollte.

2.1 Architektur

Die Grundidee von MARS besteht darin, immer von einem stabilen Zustand in einen anderen zu gelangen. Dieser Zustandswechsel wird durch ein Ereignis gestartet, das dann nach dem dafür vorgesehenen Zeitplan abgearbeitet wird und am Ende ein Ergebnis liefert. Da diese Schritte komplex sein können, teilt Mars diese in viele kleine Schritte auf, die dann einfach und stabil zu programmieren sind. Wenn das maximal zur Verfügung gestellte Zeitintervall vorgegeben ist, nennt man es Echtzeitübergang. Mit dieser absolut zeitgesteuerten und periodischen Strategie erreicht MARS auch unter Volllast harte Echtzeitanforderungen und deterministisches Verhalten.

Durch diese Aufteilung besteht ein MARS-System aus vielen verschiedenen Cluster, die aus mehreren Komponenten bestehen, die durch einen Echtzeit-Bus (MARS-Bus) eine hohe innere Anbindung besitzen. Jede Komponente ist ein selbstständiger Computer, der mehrere Aufgaben erfüllen kann. Diese Hardware-Software-Vereinigung muss sicherstellen, dass es ihre Echtzeitaufgaben in der Vorgegebenen Zeit richtig verarbeitet. Außerdem hat jede Komponente noch eine Kopie des MARS-Betriebssystems, um das korrekte Zusammenspiel der Komponenten zu gewährleisten. Solange keine Anfragen an eine Komponente kommen, kann diese zu jeder Zeit vom System genommen und durch Andere ersetzt werden. Es muss nur sichergestellt werden, dass diese mit dem System bei Ereignis Aufnahme und Antwort kooperiert. Im folgendem Bild sieht man angedeutet, wie das System in mehrere Cluster mit ihren Komponenten aufgeteilt ist, die jeweils ihren Echtzeit-Bus haben.



Die einzige Kommunikationsmöglichkeit zwischen verschiedenen Abwicklungsschritten, Komponenten und Clustern sind die Status-Mitteilungen. Diese haben immer eine gewisse Gültigkeitszeit, nach deren Ablauf die Meldung verworfen wird, damit keine veralteten Nachrichten verwendet werden. Diese Ablauffrist existiert, da in einem Echtzeitsystem nicht nur das korrekte Ergebnis interessiert, sondern alles der Zeiteinhaltung unterliegt. Um dies alles synchronisieren zu können, verwenden alle die gleiche Systemzeit, die global für alle Komponenten zur Verfügung steht. Diese Systemzeit wird verwendet, um den Zugang zu dem MARS-Bus zu kontrollieren und den Ablaufzeitpunkt der Nachrichten zu bestimmen u.s.w.

Das MARS-System vollzieht Fehlererkennung durch Selbstkontrolle der Komponenten und durch mehrfaches Schicken der Nachrichten über den Echtzeit-Bus. Weitere Sicherheitsmassnahmen sind der Einsatz von Software-Fehlererkennungsprogrammen beim Kernel, sowie der Einsatz spezieller Hardware. Diese Fehlererkennung ist sehr wichtig, da bei einem Fehler keine Meldung kommt, sondern einfach stillschweigend keine Ausgabe erfolgt, so dass danach keine Möglichkeit besteht, diesen Fehler zu erkennen. Dies wäre wiederum fatal für das gesamte Echtzeitsystem.

2.2 Struktur einer MARS-Komponente

Eine MARS-Komponente ist eine Kombination aus Hardware und Software, die einfach zu pflegen ist. Die Software dafür kann in drei Klassen unterteilt werden:

- Betriebssystemkernel

  Der Kernel läuft im privilegierten Modus der CPU, deren Hauptaufgabe die Resourcenverwaltung und Hardwareabstraktion ist. Die Grundaufgaben sind ähnlich zu einem normalen Kernel, auch wenn beim Implementieren wieder mehr auf Fehlertoleranz und Echtzeit Wert gelegt wurde.

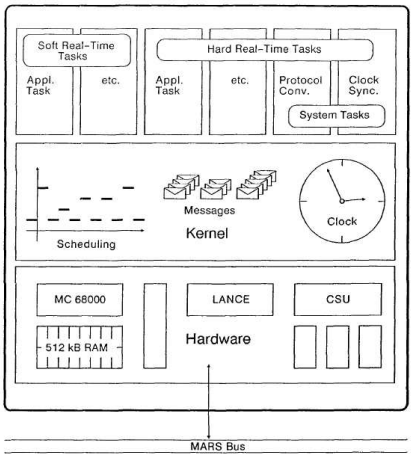
- Verarbeitung von Harter Echtzeit

  Hierunter versteht man das Empfangen, Verarbeiten und Senden von Nachrichten, das wieder in der bei der Entwicklung des Systems vorgegebenen Zeit abgeschlossen werden muss. Das Empfangen ist der Einfluss von außen, auf den mit dem Senden der Nachricht die Antwort geschickt wird. Die meisten Harten Echtzeitverarbeitungen sind Auftragsverarbeitungen. Darunter gibt es aber auch Systemverarbeitungen, wie z.B. Zeitsynchronisation.

- Verarbeitung von Weicher Echtzeit

  Diese haben kein streng einzuhaltendes Zeitlimit, und werden so normalerweise zu einer Zeit ausgeführt, wenn die CPU wenig beansprucht wird und frei verfügbar ist.

Dies wird nochmal mit folgender Grafik veranschaulicht:



2.3 Nachrichtenweitergabe

2.3.1 Status-Mitteilung

Alle Kommunikationen mit der Umwelt, wie z.B. den Komponenten, oder den Clustern, werden über eine einheitliche Nachrichtenart übermittelt. Sie werden als Status Mitteilungen gesendet. Die Nachrichten werden durch einen clusterweiten eindeutigen Namen identifiziert. Die Nachrichten werden weder verändert noch besitzen die Aufgaben einen exklusiven Zugriff , so dass die in einer Komponente gespeicherte Nachricht beliebig oft von beliebig vielen verschiedenen Aufgaben nahezu zu jeder Zeit gelesen werden können. Es kann nur eine Mitteilung zu einem Nachrichtennamen gültig sein, d.h. wenn eine neue Nachricht verschickt wird, ist die alte nicht mehr gültig.

2.3.2 Struktur einer Nachricht

MARS-Nachrichten haben konstante Länge, einen Standardkopf und einen Standardschluss. Neben der zum Verschicken notwendigen Informationen, wie z.B. Quelladresse und Zieladresse, kommen noch der Name der Nachricht und mehrere Zeitinformationen hinzu, wie z.B. Gültigkeitsdauer, Sendezeit und Empfangszeit. Das Ende enthält eine Prüfsumme, um wieder die Fehlertoleranz zu erhöhen. Die Information kann dann beliebig in der Nachricht enthalten sein. Einzige Vorgabe ist die Länge, da jede Nachricht die gleiche Länge haben muss.

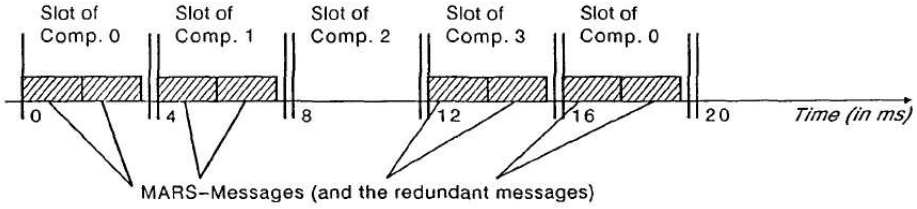
2.3.3 Pufferverwaltung

Da die Anzahl der zu empfangenden Nachrichten und die Anzahl der durch die Komponenten benötigen Zwischenspeicher (z.B. zur nächstmöglichen Zeit die Nachricht zu verschicken) schon zur Entwicklungszeit feststeht, kann man für deren Speicherung genug Zwischenspeicher bereithalten. Kommt eine neue Nachricht an, wird die alte einfach überschrieben, da nur die neueste Information Gültigkeit besitzt.

Die Zwischenspeicherverwaltung wird vom Kernel übernommen. Um eine schnellere Abarbeitung zu ermöglichen und unnötiges Kopieren zu vermeiden, bekommen die Aufgaben in der Komponente nur einen Pointer auf diese Nachricht. Dies hat außerdem den Sinn, dass die Aufgaben immer mit den neuesten Informationen arbeiten. Daher stehen diese Nachrichten selbstverständlich nur zum Lesen bereit, damit keine Aufgabe diese verändern kann.

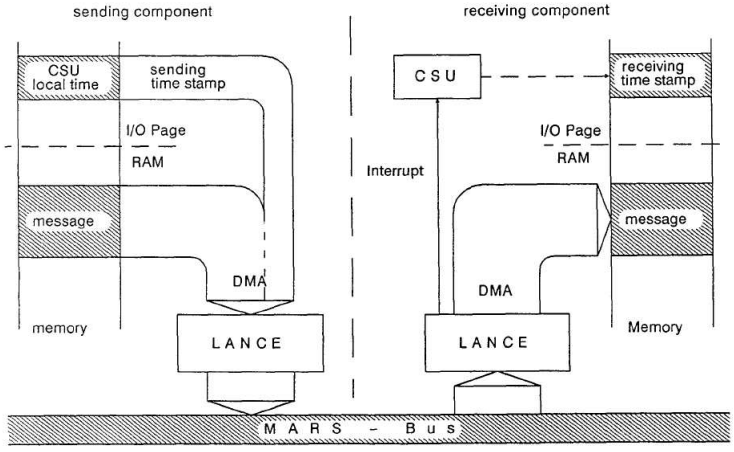
2.3.4 TDMA und Nachrichten-Ablaufplanung

Um MARS-Mitteilungen übers LAN zu verschicken, kann man die gewöhnlichen Protokolle nicht verwenden, da diese für die Echtzeit nicht genügen. Daher wird das TDMA-Protokoll (Time Devision Multiple Access) verwendet, um einen kollisionsfreien Verlauf auch unter Volllast zu gewährleisten. Jede Komponente für Harte Echtzeitübertragung hat in gleichen zyklischen Abständen immer wieder eine gewisse Zeit die Leitung für sich alleine. In der Zeit wird die Nachricht zur Fehlerkontrolle zweimal verschickt. (Siehe hierzu die Grafik weiter unten). Der Nachteil dieses Verfahrens ist, dass die Übertragung der Daten relativ lange dauert, wenn außer von wenigen Komponenten keine Daten verschickt werden, da das Transfervolumen für die Komponente nur ein Bruchteil der möglichen Leitungskapazität ist. Dafür hat diese Komponente diesen Bruchteil unter allen Umständen zur Verfügung. Die weiche Echtzeit kann die nicht benötigten Zeitintervalle verwenden, da für diese kein spätester Zeitpunkt festgelegt ist.



2.3.5 Zeitstempel

Jede MARS-Nachricht, die übers Netz verschickt wurde, enthält zwei Felder für den Zeitpunkt, zu dem sie abgeschickt wurde, und für den Zeitpunkt, zu dem sie empfangen wurde. Diese Angaben sind auf drei Mikrosekunden genau. Diese ständigen Zeitstempel werden v.a. zur Synchronisation und Fehleranalyse verwendet. Der Ablauf wird in folgender Zeichnung noch einmal deutlich gemacht.



3 Zusammenfassung

3.1 Fehlertoleranz

Jede Komponente ist intern für sich selbst dafür verantwortlich, dass keine Fehler auftreten. Dies wird mit Software-Fehlererkennung und spezieller Hardware realisiert.

Ansonsten liegt die einzige Gefahr bei den Nachrichten, sei es beim Verschicken, sei es, dass veraltete Nachrichten verwendet werden. Der Versand erfolgt durch zweimaliges Senden der Nachricht; diese erhält zusätzlich eine Prüfsumme am Ende einer jeden Nachricht, so dass bei einer Verfälschung rekonstruiert werden kann, welche die Richtige ist. Dass nicht mit veralteten Mitteilungen gearbeitet wird, verhindert man, indem die Aufgaben immer nur einen Pointer auf die Nachricht bekommen. Desweiteren hat jede Nachricht ihren eigenen Ablaufzeitpunkt, so dass dann erst eine Neue angefordert werden muss. Sollte in dieser Zeit eine neue Mitteilung zugeschickt werden, wird die alte im Puffer überschrieben, so dass die Aufgaben wieder nur die neue Nachricht zur Verfügung haben.

Fehlertoleranz und Fehlererkennung sind enorm wichtig, da das System keine Fehlermeldungen bringt, weil das System davon ausgehen muss, dass die vorliegenden Daten korrekt sind. Würde man dies immer nochmals nachkontrollieren und gegebenenfalls die Nachricht erneut anfordern, könnte man die Echtzeit nicht mehr garantieren.

3.2 Einsatz als Echtzeitsystem

MARS ist ein verteiltes fehlertolerierendes Echtzeitsystem mit hohen Ansprüchen an die Ausfallsicherheit und Zeiteinhaltung. Diese Ansprüche werden auch bei absoluter Volllast sicher eingehalten, indem das System in einzelne Cluster aufgeteilt wird, die dann deutlich einfacher und sicherer zu warten sind. Hierdurch wird dann eine umfangreiche Aufgabe in kleinere Schritte zerlegt.

Der Informationsaustausch geht nur über die Status-Mitteilungen mit einer einheitlichen Struktur. Hierdurch wird die Möglichkeit gegeben ein Ereignis für die Komponenten zu übermitteln und die Antworten wieder in Form einer Nachricht zurückzuschicken.

Durch die Puffer liegen die Nachrichten zu jeder Zeit für die Komponenten bereit, ohne dass sie diese erst jedesmal neu anfordern müssen.

Durch das TDMA-Protokoll können die Nachrichten auch übers Netz in Echtzeit verschickt werden.

4 Literaturverzeichnis

Literatur

[1] A.Damm, J. Reisinger, W. Schwabl, and H. Kopetz. The real-time operating system of MARS. 1989

[2] Real-Time Systems - Hermann Kopetz; Kluwer Academic Publishers