Ausgewählte Kapitel eingebetteter Systeme

Statische Ablaufplanung

Hauptseminar
SS 2006

Matthias Bott
12.07.2006

Inhaltsverzeichnis

1 Einleitung

Ablaufplanung beschäftigt sich mit der Zuteilung von Aufgaben zu Betriebsmitteln. Ein Scheduling Algorithmus ist ein Regelwerk, das festlegt, welcher Aktivitätsträger an einem bestimmten Moment das Betriebsmittel belegen wird. Ein Betriebsmittel kann, je nachdem in welchem Umfeld man Ablaufplanung betreibt, ein Prozessor, Speicher oder andere Geräte sein.

Je nach Problemfeld können an ein Verfahren zur Ablaufplanung verschiedene Anforderungen gestellt werden. Unterschieden wird, ob für die Ausführung der Tasks ein oder mehrere Betriebsmittel zur Verfügung stehen. Zudem können Tasks verschiedene Eigenschaften besitzen, die bei der Ablaufplanung berücksichtigt werden müssen.

Tasks können

- Datenabhängigkeiten besitzen.
- unterbrechbar sein.
- periodisch sein.
- zeitliche Beschränkungen besitzen.
- Prioritäten zugeordnet sein.

Grundsätzlich gibt es zwei Unterschiedliche Ansätze Ablaufplanung zu betreiben:
Ablaufplanung kann statisch oder dynamisch erfolgen. Genauer wird darauf im nächsten Abschnitt eingegangen.

Unterschieden wird Ablaufplanung auch nach dem Zeitpunkt der Erstellung des Ablaufplans. So gibt es die Möglichkeit, Ablaufplanung offline vor oder online während der eigentlichen Programmausführung zu betreiben.

2 Statische vs. dynamische Ablaufplanung

2.1 Abgrenzung

Bei statischer Ablaufplanung besitzt der Algorithmus zur Erstellung des Ablaufplans vollständige Informationen über die Menge der Tasks und ihre Eigenschaften.

Folgende Informationen über einen Task müssen vorhanden sein:

- Bereitzeit
- maximale Ausführungszeit (*WCET*)
- Frist (*Deadline*)
- Periode

Der Algorithmus erzeugt aus dieser Menge einen vollständigen Ablaufplan in tabellarischer Form, der nicht mehr verändert wird. Der Plan enthält feste Angaben, wann die einzelnen Aufgaben ausgeführt werden. Die einer Aufgabe durch den Algorithmus der Ablaufplanung zugeteilte Prozessorzeit ist gleich seiner maximalen Ausführungszeit (*WCET*).
Statische Ablaufplanung erfolgt typischerweise offline vor der eigentlichen Programmausführung.
Im Gegensatz dazu besitzt der Algorithmus bei dynamischer Ablaufplanung lediglich die Informationen zu der aktuellen Menge an Tasks. Diese Menge ist dynamisch, das heißt, sie kann sich während der Laufzeit ändern. Dynamische Ablaufplanung erfolgt deshalb online während der eigentlichen Programmausführung.

2.2 Abarbeitung statischer Ablaufpläne

Der statische Ablaufplan wird während der Laufzeit von einem Dispatcher abgearbeitet.
Im Normalfall werden alle Aufgaben fristgerecht ausgeführt. Unvorhergesehene Ausnahmen wie synchrone Programmunterbrechungen, die zum Beispiel durch Berechnungsfehler ausgelöst werden, dürfen nicht zu einem Verpassen von Fristen führen.

Der Ablaufplan wird durch eine Tabelle repräsentiert. Ein Tabelleneintrag definiert eine Einplanungsentscheidung an einem bestimmten Zeitpunkt. Er kann eine Adresse bzw. eine Identifikation des Arbeitsauftrags oder ein Ruheintervall einer Aufgabe darstellen. In jedem Eintrag befindet sich zu dem noch ein Wert, welcher den Zeitpunkt der nächsten Planungsentscheidung angibt. Wird die Aufgabe vom Dispatcher eingelastet, wird ein Zeitgeber auf diesen angegebenen Zeitpunkt gestellt. Der Zeitgeber aktiviert den nächsten Tabelleneintrag. Liegt für periodische Aufgaben ein zyklischer Ablaufplan vor, wird nach dem letzten Eintrag in der Tabelle wieder der erste Eintrag ausgeführt.

2.3 Typische Anwendungsgebiete

Dynamische Ablaufplanung ist in vielerlei Sicht flexibler als statische. Jedoch hat die statische Planung auch einige Vorteile.

Bei statischer Ablaufplanung sind genaue Informationen über Ausführungszeiten und Berechnungszeiten aller Prozesse vorhanden. Nach erfolgreicher Erstellung ist deshalb sichergestellt, dass alle Bedingungen (z.B. Fristen) bei der Abarbeitung eingehalten werden.

Jedoch kann die Voraussetzung, dass vollständige Informationen über alle Prozesse schon bei der Planung verfügbar sein müssen, nicht von allen Systemen erfüllt werden. Realistisch ist dies trotzdem für vielerlei Echtzeitsysteme, die in eine wohl definierte Umgebung eingebettet sind.

3 Strategien zur Erstellung statischer Ablaufpläne

Statische Ablaufpläne werden typischerweise offline erstellt. Es können deshalb komplexe Algorithmen dafür eingesetzt werden. Im nächsten Abschnitt möchte ich zwei Ansätze vorstellen.

3.1 Finden eines statischen Ablaufplans durch Suchen

Gegeben:

- Periodische Tasks
- Alle Eigenschaften (z.B. Bereitzeit, Berechnungszeit, Frist, ...) der Tasks sind bekannt

Randbedingung:
Es wird von nicht unterbrechbaren Tasks ausgegangen.

Gesucht:
Gültiger Ablaufplan, der sämtliche Tasks unter Berücksichtigung aller Beschränkungen plant.

3.1.1 Erstellung des Suchbaums

- Es wird jede mögliche Abfolge der Tasks in einem Suchbaum abgebildet.
- Die Wurzel des Baums ist der Ausgangspunkt der Suche, der leere Ablaufplan.
- Ausgehend von der Wurzel stellt jede Kante eine weitere Planungsentscheidung (Planung eines Tasks zu einem bestimmten Zeitpunkt) dar.
- Jeder Pfad zu einem inneren Knoten weist auf einen unvollständigen, jeder Pfad zu einem Blatt auf einen vollständigen Ablaufplan.
- Blätter stellen nicht unbedingt gültige Ablaufpläne dar.

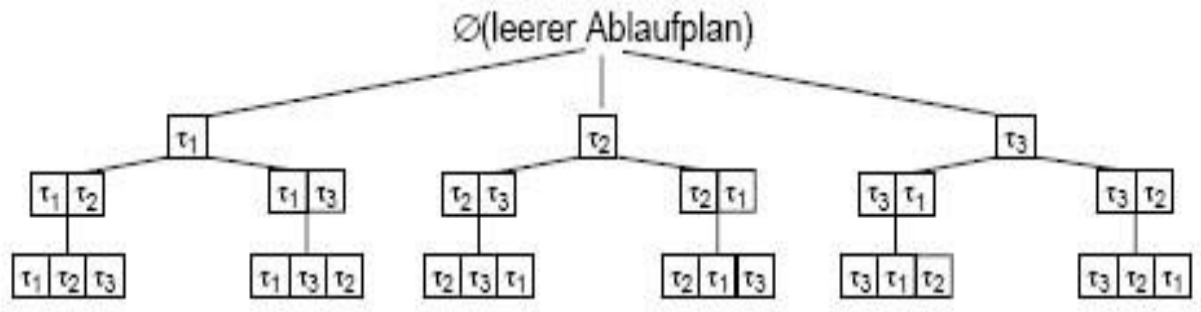


Abbildung 1: Beispiel eines Suchbaums

3.1.2 Suche nach gültigen Ablaufplänen

Algorithmus:

1. Gehe von der Wurzel des Suchbaums aus, also von dem leeren Ablaufplan
2. Wähle ein noch nicht besuchtes Kind aus
3. Überprüfe, ob der unvollständige Ablaufplan, den der Pfad repräsentiert, ein gültiger Ablaufplan ist

   gültiger Ablaufplan:

   Ist der aktuelle Knoten ein Blatt des Baumes?

   Ja: Gültiger Ablaufplan gefunden!

   Füge Ablaufplan in eine Liste, die alle gefundenen gültigen Ablaufpläne beinhaltet, ein

   Abbruchkriterium erfüllt?

   Ja: Gehe zu Schritt 5

   Nein: Gehe zu Schritt 4

   Nein: Gehe zu Schritt 2

   Ablaufplan nicht gültig:

   Gehe zu Schritt 4
4. Gehe so viele Schritte zurück bis ein Knoten gefunden wird, dessen Kinder noch nicht alle besucht sind und gehe anschließend zu Schritt 2
5. Wähle aus Liste der gefundenen Ablaufpläne einen optimalen Plan aus

3.1.3 Problemfelder

Überprüfung der Gültigkeit des Ablaufplans

Überprüfe, ob zeitliche Eigenschaften eines jeden schon geplanten Tasks berücksichtigt werden:

Bereitzeit ≤ Startzeit

Startzeit + WCET < Frist

Abbruchkriterium

Es ist möglich, das Abbruchkriterium so zu wählen, dass es immer erfüllt ist. Dann wird nach dem ersten gefundenen, gültigen Ablaufplan die Suche beendet.

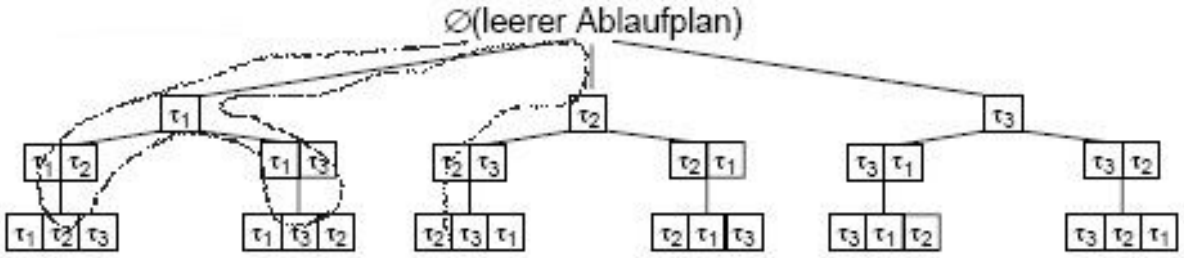


Abbildung 2: Darstellung eines möglichen Ablaufs der Tiefensuche

3.2 Tabu-Suche

Die Tabu-Suche ist

- iterativ.
- ein heuristisches Optimierungsverfahren.
- für komplexe Probleme geeignet.

3.2.1 Grundidee

Ausgegangen wird von einer Initial-Lösung. In jedem Durchgang wird der beste Ablaufplan einer Nachbarschaft des aktuellen Ablaufplans ausgewählt. Die Nachbarschaft wird durch eine Nachbarschaftsfunktion erzeugt und muss mindestens ein Element enthalten. Die Struktur der Nachbarschaft ändert sich von Iteration zu Iteration. Damit keine Zyklen entstehen, werden die betrachteten Lösungen der Nachbarschaft in eine Tabu-Liste eingefügt. Lösungen, die in der Tabu-Liste eingetragen sind, dürfen in späteren Iterationen nicht betrachtet werden. Um lokale Minima zu überwinden, sollten auch Lösungen akzeptiert werden, die nicht besser sind als der aktuelle Ablaufplan.

3.2.2 Algorithmus

1. Starte Algorithmus mit einer Initial-Lösung
2. Erzeuge eine Nachbarschaft durch die Nachbarschaftsfunktion
3. a) Selektiere die beste Lösung der Nachbarschaft
   (Lösungen, die in der Tabu-Liste stehen, scheiden als mögliche Lösung aus)
   b) Bewerte Lösung mit einer Bewertungsfunktion und entscheide, ob Tabu-Suche beendet wird
4. Trage die gewählte Lösung in die Tabu-Liste ein
   Fahre mit Schritt 2 fort

3.2.3 Problemfelder

Größe der Tabu-Liste

Ist die Größe der Tabu-Liste zu klein gewählt, kommt es zur Zirkulation zwischen mehreren „besten“ Lösungen. Zu groß darf diese jedoch auch nicht gewählt werden, weil der Algorithmus sonst sehr viel Speicher verbraucht.

Bewertungsfunktion

Die Bewertungsfunktion kann je nach Schedulingproblem von unterschiedlicher Natur sein. Für die Ablaufplanung von weichen oder festen Echtzeitsystemen könnte man die Anzahl eingehaltener Fristen als Kriterium benutzen.

Nachbarschaftsfunktion

Möglich wäre, einen Task durch Zufall zu selektieren und diesen im Ablaufplan zu verschieben. Jede mögliche Verschiebung des Tasks kann einen Nachbar darstellen. Alle Verschiebungen stellen die Nachbarschaft dar.

3.3 Weitere heuristische Verfahren

Weitere heuristische Optimierungsverfahren, die ebenfalls für Ablaufplanung interessant sind:

- Simulierte Abkühlung (Simulated annealing)

  Es wird ein physikalischer Abkühlungsprozess nachgebildet. Der Algorithmus erzeugt in jeder Iteration eine Nachbarlösung und senkt die Temperatur. Die Temperatur entspricht einer Akzeptanzschwelle. Unterhalb dieser Schwelle wird die Nachbarlösung als neue Zwischenlösung akzeptiert, auch wenn diese schlechter ist als die vorherige Lösung. Somit ermöglicht der Algorithmus, dass die Lösung aus einem lokalen Optimum entfliehen und möglicherweise ein besseres lokales Optimum oder sogar das globale Optimum erreichen kann.
- Evolutionäre Algorithmen

  Eine Anzahl an Individuen werden zufällig erzeugt. Die Besten unter den Individuen werden durch eine Gütefunktion ausgewählt. Diese werden mutiert und miteinander kombiniert, um neue Lösungen für eine neue Generation zu erzeugen.

4 Erweiterte statische Ablaufplanung

Nachfolgend möchte ich auf Erweiterungen hinweisen, die Systeme mit statischer Ablaufplanung flexibler machen. Es wird eine Möglichkeit eingeführt, verschiedene Arbeitsphasen eines eingebetteten Systems gesondert zu behandeln. Anschließend wird die Einschränkung eines statischen Ablaufplans, dass nur periodische Tasks behandelt werden können, aufgeweicht.

4.1 Wechsel des Arbeitsmodus

Ein eingebettetes System kann während der Programmausführung verschiedene Phasen durchlaufen, die sich hinsichtlich der Tasks, die ausgeführt werden sollen, unterscheiden.

Beispiel: Kontrollsystem eines Flugzeugs
Während Start-, Flug- und Landephase sind unterschiedliche Kontrollaktivitäten nötig.

Diese Phasen werden im System als Arbeitsmodi modelliert. Verschiedene Arbeitsmodi eines Systems können auch verschiedene zeitliche Anforderungen besitzen:

- unkritischer Abschnitt
- weiche Echtzeit
- harte Echtzeit

Um solche Arbeitsmodi zu unterstützen, muss für jeden Modus jeweils ein statischer Ablaufplan erstellt werden. Bei einem Wechsel des Arbeitsmodus ersetzt der Plan des neuen Modus den aktuellen Ablaufplan.

Ausgelöst wird ein Moduswechsel von einem Task innerhalb des gerade aktiven Arbeitsmodus. Ein Wechsel wird ebenfalls als ein eigenständiger Arbeitsmodus verstanden, dem Wechselmodus.

Anforderungen an das System:

- Zeitliches Verhalten muss in harten Echtzeitsystemen deterministisch bleiben.
- Voraussetzung dafür ist die Bindung von zeitlichen Beschränkungen an Wechsel der Arbeitsmodi.
- Um die korrekte Ausführung des Systems sicher zu stellen, muss gewährleistet werden, dass ein Wechsel des Arbeitsmodus keinen Einfluss auf Aktivitäten innerhalb der Arbeitsmodi hat.

4.2 Behandlung aperiodischer Tasks

Bei statischer Ablaufplanung werden alle Aufgaben als periodische Tasks modelliert.
Es gibt zwei Arten von Aufgaben, die so abgefertigt werden können:

Natürlich-periodische Aufgaben

> Alle Aktivitäten, die von Natur aus periodisch sind, wie zum Beispiel wiederkehrende Kontrollaktivitäten.

Pseudo-periodische Aufgaben

> Alle Aktivitäten, die von Natur aus nicht periodisch sind, aber durch spezielle Tasks also solche modelliert werden können.
>
> Diese Tasks werden an festen periodischen Zeiten statisch geplant. Wird der Task während der Laufzeit ausgeführt, prüft er, ob die Bedingung gilt, die zum Ausführen einer solchen Aktivität führen soll. Ist dies der Fall, wird die Aufgabe ausgeführt.

Ein Beispiel für aperiodische Aktivitäten sind asynchrone Ereignisse (z.B. Interrupts). Aperiodische Aktivitäten, die als pseudo-periodische Aufgaben modelliert sind, plant der Algorithmus der Ablaufplanung statisch im Ablaufplan. Dieser Task wird periodisch ausgeführt und prüft, ob die Bedingung, die zur Ausführung der Aufgabe führt, im aktuellen Moment gültig ist. Asynchrone Ereignisse werden so durch *Polling* abgefragt, was Berechnungszeit verschwendet.

Ein Ansatz Aktivitäten während der Abarbeitung eines statischen Ablaufplans noch mit einzuplanen, ist *slot shifting*. Ein Slot bezeichnet hierbei eine Zeiteinheit, in der Tasks geplant werden können.

Grundidee:

In statischen Ablaufplänen sind ungenutzte Zeitspannen verfügbar und bekannt. Diese können für dynamische Aktivitäten benutzt werden.

Erster Ansatz:

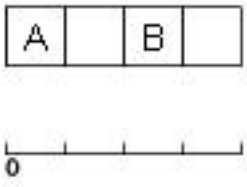
Während dieser Leerlaufzeiten (*idle times*) des statischen Ablaufplans können dynamisch Aktivitäten eingeplant werden, deren Berechnungszeit maximal so groß sein darf wie die Leerlaufzeit.

-> Kombination von statischer und dynamischer Ablaufplanung

Beispiel:

Task	Bereitzeit	Frist	Dauer
A	0	2	1
B	2	4	1

Erzeugter Ablaufplan:

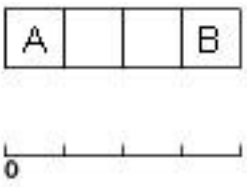


Zum Zeitpunkt 1 und 3 könnten hier dynamische Aktivitäten eingeplant werden.

Erweiterung:

Die statischen Tasks werden im Ablaufplan innerhalb ihrer zeitlichen Beschränkungen so verschoben, dass die Leerlaufzeiten zwischen ihnen maximiert wird.

Wird im Beispiel Task B erst zum Zeitpunkt 3 geplant, entsteht ein Slot von 2 Zeiteinheiten. Die Deadline wird dennoch eingehalten.



5 Literaturverzeichnis

[1] Fohler, G.: Flexibility in Statically Scheduled Hard Real-Time Systems. Dissertation, Technische Universität Wien, 1994

[2] Fohler, G.: Time Triggered and Event Triggered; Off-line Scheduling. Mälardalen University, Sweden, 2004

[3] Lennvall, T: Handling Aperiodic Tasks and Overload in Distributed Off-line Scheduled Real-Time Systems. Licentiate Thesis, Mälardalen University, Sweden, Mai 2003

[4] Bellosa, F: Ausgewählte Kapitel der praktischen Betriebsprogrammierung: Embedded Systems. Universität Erlangen-Nürnberg, 2000

[5] Teich, J: Digitale Hardware / Softwaresysteme - Synthese und Optimierung, Springer, 1997

[6] Liu, C.L.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the Association for Computing Machinery, Vol. 20, No. I, January 1973