auch gewollte Anteile enthalten, nämlich wenn sich die Task für eine definierte Zeit schlafen legt.

Je nach Auslastung des Systems ergibt sich eine minimale und eine maximale Reaktionszeit für die Rechenzeitanforderung i ($t_{Rmin,i}$ und $t_{Rmax,i}$).

2.1.3 Beschreibungsgrößen der Systemsoftware

Die Systemsoftware ist maßgeblich für das Zeitverhalten des Realzeitsystems mitverantwortlich. Sie wird unter anderem durch die Latenzzeiten (Verzögerungszeiten) gekennzeichnet. Diese können unterschiedliche Ursachen haben. Folgende Latenzzeiten werden unterschieden:

- Interrupt-Latenz
- Task-Latenz
- Kernel-Latenz
- Preemption Delay (Verdrängungszeit)

Die Interrupt-Latenzzeit t_{LIsr} ist die Zeit, die zwischen dem Auftreten eines Interrupts und dem Start der zugehörigen Interrupt-Service-Routine vergeht.

Unter der Task-Latenzzeit t_{LTask} versteht man die Zeit, die zwischen dem Auftreten eines Ereignisses – das kann ebenfalls ein Interrupt sein – und dem Start der zugehörigen Task vergeht.

Betriebssysteme werden bezüglich ihres Zeitverhaltens auf einer definierten Hardware primär über die Interrupt-Latenzzeit und die Task-Latenzzeit durch die Hersteller gekennzeichnet.

Kernel-Latenz $t_{LKernel}$ ist die Zeit, die eine Task vor oder während ihrer Ausführung warten muss, weil systembedingt und zumeist innerhalb des Kernels andere Codesequenzen bevorzugt abgearbeitet werden. Gründe für die Kernel-Latenz stellen somit Unterbrechungssperren und die vor parallelem Zugriff geschützten Bereiche innerhalb von Systemfunktionen respektive Systemcalls dar (siehe auch Abschnitt 4.3).

Verdrängungszeiten $t_{LPreempt}$, auch Preemptionzeiten genannt, entstehen, wenn ein wichtiges Ereignis eine gerade laufende Codesequenz unterbricht.

2.2 Realzeitbedingungen

Die fristgerechte Bearbeitung von Rechenzeitanforderungen ist dann garantiert, wenn die erste und die zweite Realzeitbedingung erfüllt sind: die Auslastungs- und die Rechtzeitigkeitsbedingung. Welche Auswirkun-

gen das Nichteinhalten der Rechtzeitigkeitsbedingung hat, wird in Abschnitt 2.2.3 erläutert.

2.2.1 Gleichzeitigkeit und Auslastung

Jeder im System vorhandene Rechnerkern stellt eine Rechenleistung zur Verfügung, bei der innerhalb eines Zeitfensters eine endliche Anzahl von Befehlen abgearbeitet werden kann. Wird eine Task aktiv, beansprucht sie einen Teil der Rechenleistung.

Abhängig von der Auftrittshäufigkeit einer Rechenzeitanforderung i und der damit anfallenden Ausführungszeit $t_{E,i}$ ist der Rechnerkern (Prozessor) ausgelastet. Die Auslastung ρ durch eine Rechenzeitanforderung ergibt sich damit als Quotient aus notwendiger Verarbeitungszeit und Prozesszeit:

$$\rho_i = \frac{t_{E,i}}{t_{P,i}}$$

Im Worst Case ist die Auslastung durch die Rechenzeitanforderung i durch Gleichung 2-2 gegeben.

Gleichung 2-2
Auslastung im Worst Case

$$\rho_{max,i} = \frac{t_{Emax,i}}{t_{Pmin,i}}$$

Ein Realzeitsystem muss in der Lage sein, die auftretenden Rechenzeitanforderungen in der Summe bearbeiten zu können (oft auch als Forderung nach *Gleichzeitigkeit*, Concurrency, bezeichnet). Mathematisch bedeutet dies, dass die Gesamtauslastung für jeden Rechnerkern kleiner als 100% (also 1) sein muss. Auf Mehrkernmaschinen kann jeder einzelne der *c* CPU-Kerne bis zu 100% belastet werden, sodass in diesem Fall die Gesamtauslastung kleiner als c mal 100% sein muss (Gleichung 2-3).

Die Gesamtauslastung (mit c = Anzahl der CPU-Kerne und n = Anzahl der Rechenzeitanforderungen) ergibt sich schließlich aus der Summe der Auslastungen der einzelnen Tasks (Gleichung 2-3).

Gleichung 2-3
Gesamtauslastung

$$\rho_{ges} = \sum_{j=1}^{n} \frac{t_{Emax,j}}{t_{Pmin,j}} \leq c$$

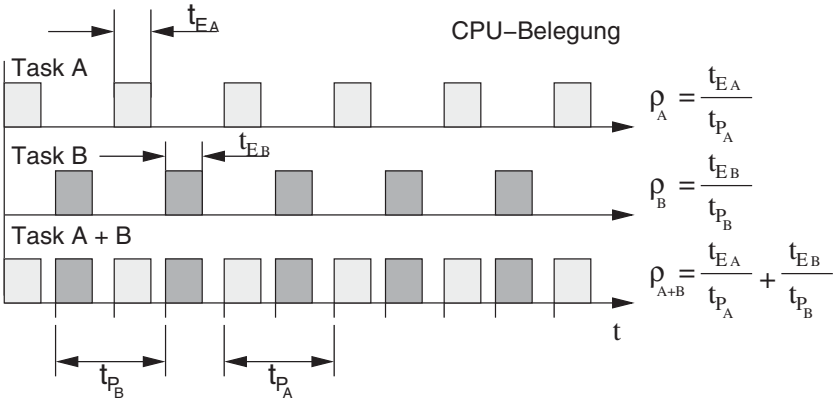


Abbildung 2-10
Auslastung

Abbildung 2-10 verdeutlicht die Auslastung für ein Singlecore-System grafisch. Ein Realzeitrechner bearbeitet zwei Rechenzeitanforderungen A und B mit jeweils einer eigenen Task: Task A und Task B. Sei die Verarbeitungszeit der Task A $t_{E,A}$ = 0.8 ms und die Prozesszeit $t_{P,A}$ = 2 ms ergibt sich eine Auslastung des Rechners durch die Task A von 40%.

Ist die Verarbeitungszeit der Task B ebenfalls $t_{E,B}$ = 0.8 ms und hat diese auch eine Prozesszeit von $t_{P,B}$ = 2 ms, ergibt sich die gleiche Auslastung wie die des Rechenprozesses A von ρ_B = 40 %.

Die Gesamtauslastung des Rechners der beide Rechenprozesse bearbeitet, beträgt $\rho_{ges} = \rho_A + \rho_B$ = 80 %.

Erste Realzeitbedingung

Die Auslastung ρ_{ges} eines Rechensystems muss kleiner oder gleich der Anzahl der Rechnerkerne sein.

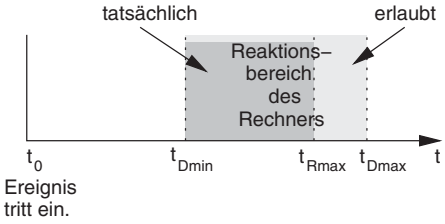
Die Auslastungsbedingung ist eine *notwendige*, aber keine *hinreichende* Bedingung. Für eine fristgerechte Bearbeitung ist es also notwendig (Grundvoraussetzung), dass die Auslastungsbedingung erfüllt ist; ohne sie geht es nicht. Andererseits reicht die Erfüllung der Auslastungsbedingung nicht aus, um die fristgerechte Bearbeitung garantieren zu können. Hierfür muss auch die zweite Realzeitbedingung erfüllt sein.

2.2.2 Rechtzeitigkeit

Neben der Korrektheit und der Vollständigkeit der Ausgangsdaten ist die Rechtzeitigkeit beziehungsweise Pünktlichkeit (Timeliness) der Daten von maßgeblicher Bedeutung. Stehen die Daten zu früh (vor $t_{Dmin,i}$) oder zu spät (nach $t_{Dmax,i}$) zur Verfügung, sind sie wertlos. Nur wenn sie dem technischen Prozess innerhalb des von diesem vorgegebenen Zeitfensters $t_{Dmin,i}$ und $t_{Dmax,i}$ zur Verfügung stehen, kann das gesamte Systemverhalten als korrekt angesehen werden.

Pünktlichkeit oder Rechtzeitigkeit darf nicht mit Schnelligkeit verwechselt werden. Allerdings lassen sich Anforderungen an Pünktlichkeit mit schnellen Systemen leichter erfüllen als mit langsamen. Prinzipiell muss ein Realzeitsystem eine Aufgabe in einem vorgegebenen Zeitfenster erfüllt haben – wie es das tut, ist dabei unwesentlich.

Abbildung 2-11
Reaktionsbereich



Gleichung 2-4
Zweite Realzeitbedingung (Rechtzeitigkeitsbedingung)

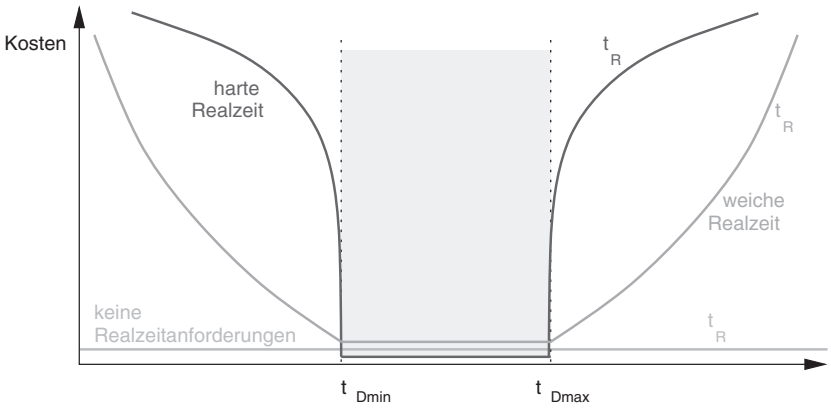
$$t_{Dmin,i} \leq t_{Rmin,i} \leq t_{Rmax,i} \leq t_{Dmax,i}$$

Zweite Realzeitbedingung

Um eine Rechenzeitanforderung i pünktlich (in Realzeit) zu erledigen, muss die Reaktionszeit größer oder gleich der minimal zulässigen Reaktionszeit, aber kleiner oder gleich der maximal zulässigen Reaktionszeit sein. Für alle Rechenzeitanforderungen *i* muss Gleichung 2-4 gelten.
Hierbei handelt es sich um eine notwendige und hinreichende Bedingung. Wenn diese Bedingung für alle Rechenzeitanforderungen i erfüllt ist, ist eine fristgerechte Bearbeitung gewährleistet.

2.2.3 Harte und weiche Realzeit

Abbildung 2-12
Harte und weiche Realzeit als Kostenfunktion



t_{Dmin} minimal zulässige Reaktionszeit

t_{Dmax} maximal zulässige Reaktionszeit

Die Forderung nach Pünktlichkeit ist nicht bei jedem System gleich stark. Während das nicht rechtzeitige Absprengen der Zusatztanks bei einer Rakete zum Verlust derselbigen führen kann, entstehen durch Verletzung der Rechtzeitigkeitsbedingung bei einem Multimedia-System allenfalls Komfortverluste. Vielfach findet man den Begriff *harte Realzeit*, wenn die Verletzung der Realzeitanforderungen katastrophale Folgen hat und daher nicht toleriert werden kann. Systeme, bei denen eine Deadline-Verletzung zunächst nur geringe Zusatzkosten aufwirft, bezeichnen wir als *weiche Realzeitsysteme*. Allerdings legt dieser Begriff nicht fest, wie stark oder wie häufig Deadline-Verletzungen toleriert werden dürfen.

Die Unterschiede bei der Forderung nach Pünktlichkeit werden oft anhand einer Kostenfunktion verdeutlicht. Bei den sogenannten *weichen Realzeitsystemen* bedeutet das Verletzen der Realzeitbedingung einen leichten Anstieg der Kosten. Bei den sogenannten *harten Realzeitsystemen* steigen jedoch die Kosten durch die Verletzung der Realzeitbedingung massiv an.

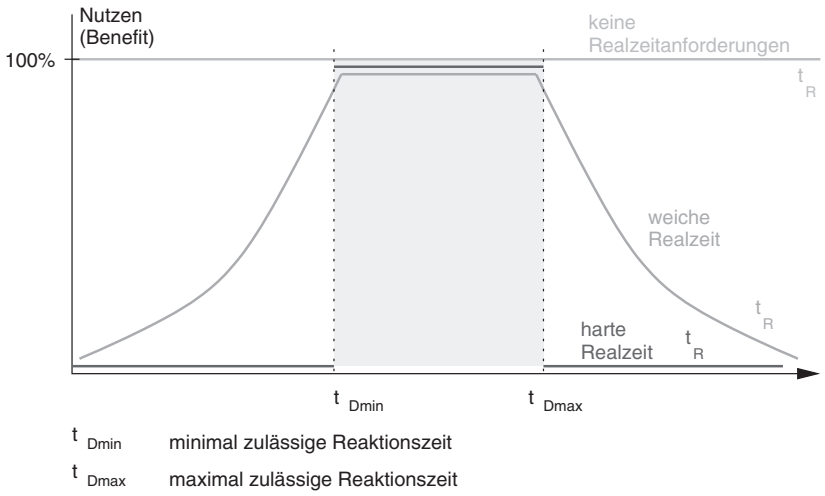


Abbildung 2-13
Nutzenfunktion

Eine andere Art der Darstellung ist die sogenannte Benefit Function (Nutzenfunktion). Hier wird der *Nutzen* der Reaktion des Realzeitsystems auf die Rechenzeitanforderung über der Zeit aufgetragen. Bei Systemen ohne Realzeitanforderungen ist der Nutzen unabhängig von dem Zeitpunkt der erfolgten Reaktion, also stets bei 100%. Bei einem harten Realzeitsystem ist nur dann ein Nutzen vorhanden, wenn die Reaktion innerhalb des durch t_{Dmin} und t_{Dmax} aufgestellten Zeitfensters erfolgt. Außerhalb des Zeitfensters ist der Nutzen null. Bei weichen Realzeitsystemen ist auch außerhalb des Zeitfensters ein Nutzen gegeben. Wie

stark dieser Nutzen ist, hängt jedoch vom jeweiligen System ab und kann nicht allgemein definiert werden.

2.3 Systemaspekte

Um Realzeitsysteme mit vernünftigem Aufwand realisieren zu können, muss das eingesetzte Betriebssystem einige Voraussetzungen erfüllen. Erstens muss es in der Lage sein, zusammengehörige Codesequenzen (Tasks) in mehrere Abschnitte zu zerteilen und Stück für Stück abzuarbeiten (Preemptibility). Zweitens muss die Systemsoftware Mechanismen zur Verfügung stellen, mit denen die Abarbeitungsreihenfolge unterschiedlicher Codesequenzen zeitlich festgelegt werden kann. Drittens schließlich muss die Systemsoftware Mechanismen zum Umgang mit Ressourcen anbieten. Durch Parallelverarbeitung kommt es nämlich schnell zu Ressourcenkonflikten und sogenannten kritischen Abschnitten.

2.3.1 Unterbrechbarkeit

Unter Unterbrechbarkeit (Preemptibility) versteht man die Eigenschaft beziehungsweise Anforderung an eine Codesequenz, die Abarbeitung in mehrere Abschnitte aufteilen zu können. Diese Abschnitte werden dann in der korrekten Reihenfolge, aber eben mit Unterbrechungen, in denen andere Verarbeitungen stattfinden, ausgeführt. Hierzu ein Beispiel:

Ein Messwerterfassungssystem, realisiert auf einer Singlecore-Hardware, soll im Abstand von 1 ms kontinuierlich Messwerte aufnehmen. Dazu benötigt die zugehörige Task eine Rechenzeit von 500 µs. Jeweils 100 Messwerte ergeben einen Datensatz, der vorverarbeitet und zur Archivierung weitergeleitet wird. Dazu ist eine Rechenzeit von 40 ms notwendig. Die Auslastung ergibt sich zu 0,5 + 0,4 = 0,9 = 90% Die Auslastungsbedingung (erste Realzeitbedingung) ist also erfüllt.

Bei linearer Abarbeitung der Aufgabe ohne Unterbrechbarkeit ergibt sich die in Abbildung 2-14 dargestellte Rechnerkernbelegung. Der zum Zeitpunkt 101 ms auftretende Messwert kann bereits nicht mehr erfasst werden. Die zweite Realzeitbedingung ist nicht erfüllt: Es handelt sich um kein Realzeitsystem.