Virtualisierung in Embedded: Fortschritte und Herausforderungen

Virtualisierung revolutioniert den Embedded-Bereich, bleibt aber aufgrund technischer Herausforderungen noch selten umgesetzt. Die dritte AURIX™ Microcontroller-Generation von Infineon verspricht nun eine native Unterstützung. Erfahren Sie, was Virtualisierung bedeutet, welche Vorteile sie bietet und welche Hürden überwunden werden müssen, um diese Technologie effizient einzusetzen.

Was versteht man eigentlich unter Virtualisierung, welche Vorteile erhofft man sich dadurch, und vor welchen Herausforderungen stehen die Entwickler diesbezüglich?

Virtualisierung ist die Fähigkeit, virtuelle Maschinen zu kreieren, die sich so verhalten, als wären sie real. Wenn man davon ausgeht, dass auf einer solchen virtuellen Maschine ein Betriebssystem läuft, spricht man in diesem Zusammenhang häufig von einem Gastbetriebssystem. Ein solches Gastbetriebssystem ist also zu Gast, aber wo?

Nun, es läuft im Kontext einer Software, die allgemein als virtueller Maschinen-Manager oder im Embedded-Bereich gerne als Hypervisor bezeichnet wird. Ein Hypervisor stellt also die virtuelle Umgebung zur Verfügung, in der ein Gastbetriebssystem läuft.

Je nach Anwendungsfall kann der Grad der Virtualisierung unterschiedlich sein. Eine vollständige Virtualisierung sämtlicher Ressourcen ist aber weder nötig noch kosteneffizient.

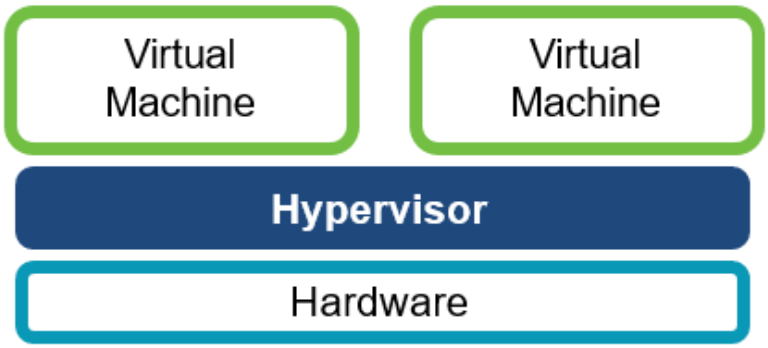

Abbildung 1: Virtualisierung – allgemeines Konzept

Welchen Vorteil bringt eine Virtualisierung?

Schauen wir als Beispiel auf die Automobilindustrie. Die Anzahl der Steuergeräte (ECUs) ist über die Jahre sehr stark gestiegen, was zu höherer Komplexität und damit auch höheren Kosten führt. Ein Ansatz, Kosten zu senken, läuft über eine höhere Integration. Es sei erwähnt, dass in den vergangenen Jahren diese erhöhte Integration über das Verwenden von Multicore-Controllern forciert worden ist.

Dies spielt nach wie vor eine Rolle, und Virtualisierung kann ein nächster Schritt zu noch höherer und einfacher zu erreichender Integration sein.

Aber braucht es zur Integration von Software unbedingt einen Hypervisor? Nicht unbedingt. Gute Software-Architekturen erlauben auch die relativ einfache Erweiterung (sprich Integration) von zusätzlicher Software. Man kann hierbei allerdings einige potentielle Nachteile ausmachen. Heterogene Systeme lassen sich unter Umständen nicht ohne weiteres integrieren. Auch könnten unterschiedliche Safety- und/oder Security-Forderungen der zu integrierenden Komponenten zu größerem Aufwand führen – bis hin zur Anpassung der Architektur selbst.

Genau hier setzt der Hypervisor an. Durch das Verwalten von mehreren virtuellen Maschinen können die ursprünglichen Software-Systeme jeweils „unverändert" auf einer individuellen virtuellen Maschine laufen und sind so voneinander getrennt. Dem Hypervisor obliegt es also nun, für Abstraktion, Trennung, Verwaltung von Ressourcen etc. zu sorgen. Natürlich kostet auch ein Hypervisor Ressourcen und Geld, was sich allerdings über die Wiederverwendbarkeit und einfachere Integration von komplexen Systemen rechnet.

Wie weit geht die Virtualisierung?

Das lässt sich nicht allgemein beantworten und hängt schlussendlich vom Anwendungsfall ab. Allerdings können wir drei größere Bereiche ausmachen, die typischerweise damit einhergehen: zeitliche Separierung, lokale Separierung und Umgang mit asynchronen Events (z.B. Interrupts). Um eine effiziente Realisierung der Virtualisierung zu erreichen, sind üblicherweise unterstützende hardwarebasierte architektonische Blöcke notwendig. Hier unterscheiden sich die verschiedenen Hardware-Hersteller voneinander, auch wenn die zugrunde liegenden Prinzipien dieselben sind.

Die zeitliche Separierung wird erreicht, indem auch der Faktor Zeit virtualisiert wird. Er kann als Ressource betrachtet werden, die wiederum vom Hypervisor verwaltet wird. Dazu werden typischerweise eine physikalische Zeit (global) und mehrere virtuelle Zeiten (je virtueller Maschine) zur Verfügung gestellt. Die lokale Separierung erfolgt einerseits auf logischer Ebene durch den User/Integrator, muss allerdings überwacht werden können, um im Zweifel immer noch sichere Systeme zu gewährleisten. Hierzu bedient man sich auf Herstellerseite der altbekannten Memory Protection Units (MPUs), allerdings erweitert um eine zweite Schutzebene, die wiederum nur vom Hypervisor kontrolliert werden kann. Die größten Unterschiede zwischen Herstellern sind bei den Interrupts zu sehen. Ein möglicher Ansatz ist, individuelle Interrupts auch an individuelle virtuelle Maschinen zu zuweisen. Je nach Hersteller erfordert dies eine durchgängige Unterstützung der Virtualisierung über alle Komponenten hinweg, die an der Bearbeitung von Interrupts beteiligt sind.

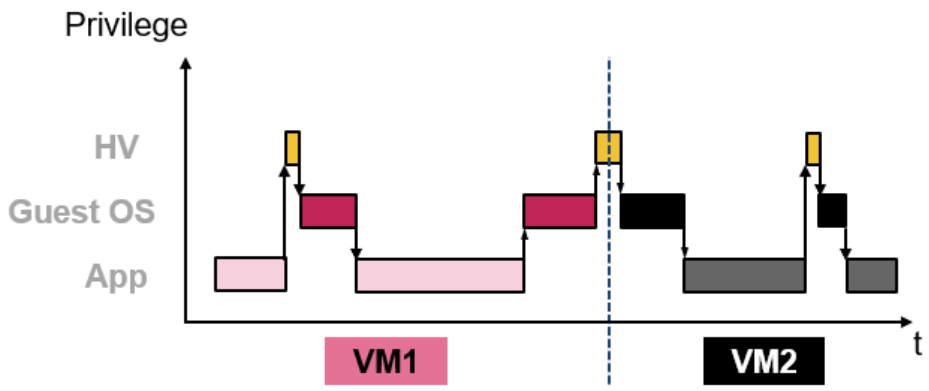

Abbildung 2: Zeit als Ressource

Zu guter Letzt gibt es auch Bereiche, wo eine Virtualisierung praktisch nicht zu erreichen ist. Dies ist insbesondere bei Peripheriemodulen zu sehen. Deshalb bedient man sich hier des sogenannten „Pass-Through“ Ansatzes, bei dem der Hypervisor dem Gastbetriebssystem direkten Zugriff auf die physikalische Ressource gewährt, dies aber wieder über entsprechende Schutzmaßnahmen überwacht.

In unserem neuen AURIX TC4x Crashkurs lernen Sie in 2,5 Tagen alles über Virtualisierung im Embedded-Bereich und entdecken viele weitere Themenbereiche. Jetzt anmelden!

Weiterführende Informationen

1. MicroConsult Fachwissen zum Thema Mikrocontroller
2. AURIX™ TC4xx Crashkurs: 32-Bit Multicore Mikrocontroller-Familie (3G Dritte Generation)
3. MicroConsult Training & Coaching zum Thema AURIX™
4. Alle Trainings & Termine auf einen Blick

Der MicroConsult-Newsletter

Wir informieren Sie mehrmals jährlich über Trends und Best Practices im Embedded Systems Engineering.

Erhalten Sie wertvolles Fachwissen und Tipps aus erster Hand von unseren Embedded-Experten!

Jetzt abonnieren!