RAM-Dateisysteme

Christoffer Kassens

26. September 2012

Inhaltsverzeichnis

1 Überblick

Schon immer hat es im Computer eine Hierarchie der Speichermedien gegeben. Diese entstand und ist heute immer noch gültig auf Grund der Tatsache, dass Geschwindigkeit und Kosten im Gegensatz zu einander stehen. Schnelle Speichereinheiten sind durchaus recht komplexe Systeme und daher in der Herstellung zu teuer, als dass man sie als Massenspeichermedien nutzen kann. Daher hat man in Abstufungen langsamere aber größere Speichermedien genutzt, um beide Nachteile zu kompensieren. So werden für die langfristige Speicherung von Daten Festplatten benutzt, die zwar sehr langsam, dafür aber sehr groß sind. Wenn es aber um die Verarbeitung oder das Ausführen von Daten geht, werden diese dafür in schnellere Speichereinheiten geladen. Dabei ist der RAM quasi eine Schnittstelle zwischen den flüchtigen und schnellen Speichereinheiten und den festen und langsamen. Es ist die erste Einheit, die nach der Festplatte kommt und ihre Daten nur flüchtig speichert, dafür aber auch deutlich schneller ist. Es ist die erste Komponente wenn es darum geht Daten für die Datenverarbeitung bereitzustellen. Außerdem hatte der RAM den Vorteil, dass er, wenn auch deutlich kleiner als die Festplatte, für einen schnellen Speicher noch recht groß ist. Häufig hat man die Größe des RAM auch so groß dimensioniert, dass er nicht vollständig genutzt wird. Daher hat man sich überlegt, wie man diese extrem schnelle Speichereinheit so zu nutze machen kann, dass man die doch sehr langsamen Festplatte möglichst umgehen kann. Grade im Serverbereich, wo es durchaus sehr wichtig sein kann möglichst schnell eine Antwort zu generieren, ist es sehr interessant ausschließlich den RAM zu nutzen, damit man nicht auf die Festplatte erst warten muss. Für solche Probleme wurden RAM-Dateisysteme entwickelt. Sie haben den RAM dahingehend präpariert, damit dieser als Dateisystem und damit wie eine Festplatte genutzt werden kann. In diesem Dokument wollen wir uns dieses Konzept genauer betrachten und uns ansehen in wie weit es heute genutzt wird.

2 Abgrenzung

In diesem Dokument wollen wir uns mit dem Thema RAM-Dateisysteme widmen. Da das Thema aber sehr komplex und nicht in diesem Rahmen vollständig behandelt werden kann, werde ich mich auf ein Teilgebiet beschränken. In erster Linie werde ich mich eher auf die konzeptionellen Ebenen des Themas beschränken. Dabei wird die historische Entwicklung in diesem Bereich erläutert, da diese wichtig ist um zu verstehen, warum RAM-Dateisystem so funktionieren, wie sie es heute tun. Auch werde ich auf die Einsatzgebiete von solchen Systemen eingehen und in wie weit sie heute noch genutzt werden. Ich werde aber nicht darauf eingehen können, wie solch ein Dateisystem im Konkreten technisch funktioniert, da das Thema sehr komplex ist und tiefer gehende Erfahrungen mit dem RAM und seinen Arbeitsweise benötigt werden. Es wird daher lediglich abstrakt darauf eingegangen wie ein RAM-Dateisystem arbeitet.

3 Geschichte

In den Jahren 1979/80 entwickelte Jerry Karlin in England die erste Software für ein RAM-Laufwerk. Es nannte sich Silicon Disk System und wurde zu kommerziellen Zwecken weiterentwickelt und von dem Unternehmen JK Systems Research vermarktet. Die Motivation hinter der Entwicklung war, dass man mehr RAM nutzen wollte als eine CPU zu dieser Zeit direkt adressieren konnte. Dazu wurde der überschüssige Speicher wie ein Laufwerk angesprochen und dieses hat sich entsprechend wie eines verhalten.
Das erste mal vertrieben wurde die Silicon Disk 1980 im CP/M Betriebssystem. Später wurde sie auch in MS-DOS eingebaut. Auf Grund der begrenzten Adressierung in der Apple II Serie und in den Commodore Computern wurden auch dort häufig RAM-Laufwerke eingebunden. Bei Apples ProDOS wurde sogar automatisch beim Betriebsstart ein RAM-Laufwerk in "/RAM" angebunden. Was damals als Möglichkeit zusätzlichen RAM zu Nutzen anfangen hat, entwickelte sich bis heute weiter, so dass moderne Betriebssysteme immer noch in der Lage sind RAM-Laufwerke anzulegen. Heute ist aber weniger die Adressierung die Motivation dazu, sondern in erster Linie der deutliche Geschwindig-

keitsgewinn bei RAM-Laufwerke.
Natürlich hat sich das Prinzip der RAM-Disk im Lauf der Zeit weiterentwickelt. So wurde später ein Dateisystem entwickelt, dass den Virtuellen Speicher nutzt, um ein Teil des RAM anzusprechen. Solch eine Entwicklung nannte sich "ramfs", abgeleitet von "RAM file system". Während eine RAM-Disk den RAM direkt angesprochen und diesen als Laufwerk bereitgestellt hat, war hier die Idee, dass ein Dateisystem den Virtuellen Speicher angesprochen und so mountbar gemacht hat. Somit arbeitet dieses Dateisystem auf einer höheren Abstraktionsebene.
Eine weitere Entwicklung von Sun war "tmpfs". Es basierte auf ramfs und sollte die Möglichkeiten des Dateisystems verbessern und die Nutzung sicherer bzw. stabiler machen. Das erste Mal wurde das Dateisystems in SunOS 4 1987 implementiert. In den Linux Kernel aufgenommen wurde es das erste Mal in der Version 2.4.

4 Dateisysteme

In diesem Kapitel wollen wir uns nun die verschiedenen Implementationen von einem RAM-Laufwerk anschauen. Ich werde dabei die drei Systeme "RAM-Disk", "ramfs" und ttmpfsnnutzen, da diese am meisten verwendet werden. Dabei gehe ich auf die Merkmale, Einsatzgebiete und deren Motivationen ein und vergleiche die Systeme auf Basis dieser Aspekte. Ich werde dabei in chronologischer Reihenfolge vorgehen, da diese Systeme sich gegenseitig motiviert und beeinflusst haben.

4.1 RAM-Disk

Eine RAM-Disk ist ein Bereich des RAM, welcher durch eine Software wie ein Laufwerk ansprechbar gemacht wurde. Damit wird der RAM aus einem Primären Speicher ein Sekundärer Speicher.
Es gab verschiedene Motivationen, warum man eine Software entwickelt hat, die aus dem RAM ein Laufwerk macht. Die größte Motivation war, dass damals die CPU nur sehr wenig RAM direkt ansprechen konnte. Deswegen wollte man dafür sorgen, dass

der überschüssige RAM irgendwie genutzt werden konnte. Um das erreichen zu können, war eine sehr maschinennahe Programmierung notwendig, die später auch in den Kernel integriert wurde. Die ersten Kernel in denen solch eine Software integriert wurde, war der Commodore und der Apple II. Später kamen auch MS-DOS und AmigaOS dazu. Bei diesen Betriebssystemen war die Verwendung einer RAM-Disk primär wegen der stark limitierten Speicheradressierung beliebt. Eine anderen Motivation für die Entwicklung von RAM-Disk war, dass man eine Möglichkeit haben wollte ein Betriebssystem starten zu können, ohne eine Festplatte ansprechen zu müssen, wie es zum Beispiel bei eingebetteten Systemen der Fall ist. Allerdings wird heute dazu der RAM mit anderen Tools nutzbar gemacht, auf die ich später noch eingehen werde.
Die Funktionsweisen sind bei den RAM-Disks weitgehend identisch. Bei der Initialisierung wird dabei eine Größe angegeben, um festzulegen wie viel Platz die RAM-Disk haben soll. Diese Größe behält die Disk dann für ihre gesamte Lebensdauer und belegt den Speicher. Für die Speicherverwaltung wird der Bereich, der für die RAM-Disk verwendet wird, so behandelt, als wenn er in Benutzung wäre. Aus diesem Grund hat die Speicherverwaltung auch keine Möglichkeit, die Daten in irgendeiner Form zu manipulieren. Das bedeutet, dass dieser Bereich im RAM von Anfang an belegt ist, auch wenn noch keine Daten auf dem Laufwerk sind. Außerdem kann die Speicherverwaltung auch nicht eingreifen, wenn der RAM voll ist, z.B. in dem es nicht benötigte Daten aus der RAM-Disk swapt[1]. Fällt die Größe der RAM-Disk zu groß aus, kann dies dazu führen, dass der Virtuelle Speicher sehr schnell voll wird und unter Umständen sogar der Rechner abstürzt. Da die Daten in der RAM-Disk außerdem nicht als Arbeitsspeicher nutzbar sind, sondern über das Laufwerk erreichbar sind, müssen diese Daten für die Verarbeitung noch ein mal in den RAM geladen werden. Die Performance muss aber nicht besonders darunter leiden, da ein Kopieren von Speicher zu Speicher ("memory-to-memory copy") sehr schnell geht. Viel gravierender ist da eher die Redundanz in dem Speicher. Nach der Initialisierung der RAM-Disk wurde aber nur ein neues Laufwerk erstellt. Dieses muss

[1] "swappen" kommt aus dem Linux-Umfeld und bedeutet, dass Daten aus dem RAM auf die Festplatte ausgelagert werden. Siehe auch: http://de.wikipedia.org/wiki/Swapping

noch mit einem Dateisystem formatiert werden. Häufig wurden dafür FAT16, FAT32 oder ext2 verwendet. Natürlich kann auch jedes andere Dateisystem verwendet werden.

4.2 ramfs

Die RAM-Disk war damals schon ein sehr statisches System, welches nicht wirklich flexibel an verschiedene Anforderungen angepasst werden konnte. Außerdem wurde es mit der Entwicklung von größeren Adressbussen möglich, dass die CPU ausreichend RAM adressieren konnte. Aus diesem Grund wurde später eine Software entwickelt, die den RAM mit einem Dateisystem formatiert und so wie ein Laufwerk ansprechbar macht.
Während die RAM-Disk sehr maschinennah arbeitete, um den RAM anzusprechen, nutzt ramfs dazu den Virtuellen Arbeitsspeicher. Es arbeitet auf einer deutlich höheren Abstraktionsschicht, als es die RAM-Disk getan hat. Da es für Unix-Systeme entwickelt wurde, hat es die internen Caching-Mechanismen genutzt und diese über ein Dateisystem ansprechbar gemacht.
Der größte Vorteil von ramfs ist, dass es eine dynamische Größe hat und immer nur so groß ist, wie es auch Daten hat. Daher ist es auch nicht notwendig die Größe beim Erstellen zu wissen. Werden neue Daten in das Laufwerk kopiert, so hat ramfs dafür neuen Speicher alloziert und entsprechend bei Löschung von Daten diesen Bereich wieder freigegeben. Allerdings wird der reservierte Bereich, ähnlich wie bei der RAM-Disk, als ïn Benutzungmmarkiert, so dass die Speicherverwaltung auch hier nicht die Möglichkeit hat Daten notfalls zu swappen[2]. Werden also unbedacht Daten auf das virtuelle Laufwerk gespeichert, so wächst es unaufhaltsam, bis es den gesamten Speicher belegt. In solch einem Fall hat das Betriebssystem keine Möglichkeiten mehr Platz im Speicher zu schaffen und es stürzt irgendwann ab.
Ein weiterer, nicht zu verachtender Vorteil liegt in der Nutzung des Linux-Caching-Mechanismus. Die Daten werden effektiv genauso im RAM abgelegt, wie es die Speicherverwaltung auch tun würden. Das versetzt das Betriebssystem in die Lage die Daten in dem virtuellen Laufwerk genauso zu nutzen, als wenn sie im Speicher läge. Dadurch

[2] vgl. http://en.wikipedia.org/wiki/Virtual_memory#Pinned.2FLocked.2FFixed_pages

bleibt das eigentlich unnötige memory-to-memory-Kopieren erspart und Daten liegen nicht mehr redundant im RAM.

4.3 tmpfs

Dieses Dateisystem ist die Weiterentwicklung von ramfs. Es ist zur Zeit das aktuellste System in der Unix Welt, wenn es darum geht den RAM wie ein Laufwerk anzusprechen. Ursprünglich entwickelt wurde es von Sun für SunOS, hat später aber auch den Sprung auf Linux-Systeme geschafft. Heute wird es nicht nur verwendet um z.B. Server zu beschleunigen, sondern auch beim Starten eines Rechners oder in Embedded Systemen ohne Festplatte.
Grundlegend sollte tmpfs die Probleme von ramfs beseitigen und es so stabiler machen. So hat tmpfs ebenfalls eine dynamische Größe. Man kann es aber auf eine maximale Größe begrenzen, um so einen Absturz des Systems zu verhindern. Darüber hinaus unterstützt tmpfs auch die Swapmechanismen, so dass das Betriebssystem in der Lage ist, unbenutzte Daten zeitweise auf den Swap zu legen. Diese Sicherung schützt zwar den Rechner vor dem Absturz, kann aber auch dazu führen, dass Daten unnötig auf eine Festplatte geschrieben werden und so der Performancegewinn verloren geht. In den meisten Fällen ist der Verlust der Performance vertretbar, um so z.B. den Server davor zu schützen, dass der RAM überfüllt wird. Es sei aber noch darauf hingewiesen, dass Daten auch nur geswapt werden können, wenn auch eine entsprechende Partition angelegt wurde. Ist dies nicht der Fall, weil zum Beispiel gar keine Festplatte angeschlossen ist, muss man auch nicht befürchten, dass Daten von der Festplatte geladen werden müssen.

5 Anwendungsgebiete

Anwendungsgebiete gibt es für RAM-Dateisysteme viele. Der simpelste Fall ist das Anlegen eines temporären und schnellen Speichers. Dazu formatiert man den RAM einfach und mountet den Bereich als Laufwerk. Es gibt aber noch weitaus spannendere Fälle in denen solche Software eingesetzt wird. Eines dieser Anwendungsgebiete findet sich in

den Systemstarts von Linuxsystemen. In den meisten Fällen heutiger Systeme wird bei einem Systemstart ein Laufwerk angelegt um so den Boot-Prozess flexibler und schneller zu gestalten. Aus technischer Sicht sind hier zwei Tools zum Einsatz gekommen: initrd und initramfs. Im Folgenden werde ich die Funktionsweise und Grundkonzepte dieser zwei Tools erläutern und gehe dabei auf die Motivationen hinter der Entwicklung ein.

5.1 initrd

Dieses Tool war die erste Entwicklung, um während des Bootprozesses den RAM als Laufwerk nutzbar zu machen. Die Idee dabei ist, dass ein Abbild eines Dateisystems mit den benötigten Dateien für den Systemstart in den RAM geladen werden. Auf diese Weise erspart man sich sehr viele Lesezugriffe auf die Festplatte, die den Systemstart verlangsamen würden.
Der Name ist die Abkürzung für "initial ramdisk" und lässt schon vermuten, dass im Hintergrund eine RAM-Disk erstellt wird. In der Tat wird diese auch erstellt und mit ext2 formatiert. Anschließend wird das Image auf diese Disk gelegt und als root-Verzeichnis gemountet. Dann führt der Kernel "/linuxrc" aus und wartet anschließend darauf, dass dieser wieder zurückkehrt. In dieser Ausführung findet dann der Bootprozess statt. So werden damit notwendige Treiber geladen und letztlich das richtige Verzeichnis gemountet. Nach der Rückkehr führt der Kernel den Startprozess weiter fort, in dem er nun "/sbin/init" ausführt. Damit ist der Vorgang komplett und das System betriebsbereit.
Wie oben Beschrieben ist zumindest in der Linux-Welt eine RAM-Disk veraltet und durch effizientere Tools ausgetauscht. Daher ist auch "initrd" veraltet und durch ein System ausgetauscht, dass die aktuellen Tools verwendet.

5.2 initramfs

Mit diesem Tool wurde die veraltete ïnitrdäbgelöst. In diesem Tool wurde einerseits durch Anpassungen der Startvorgang flexibler gemacht und zum Anderen wurden die Schwächen einer RAM-Disk beseitigt. Gerade für den Einsatz in Eingebettete Systeme bietet es Vorteile, die den Einsatz dort vereinfachen.

Die Abkürzung von "initramfs" bedeutet "initial ram filesystem", wobei hier aber nicht auf das verwendete Dateisystem hingewiesen wird. Intern nutzt es nämlich "tmpfs" um den RAM zu formatieren. Auch hier wird dann ein Image auf das neue Laufwerk kopiert. Anschließend wird allerding "/init" ausgeführt, dass dann für den weiteren Prozess zuständig ist. Der Kernel geht davon aus, dass damit der gesamte Startvorgang abgedeckt ist und führt danach keine anderen Dateien mehr aus. Daher muss "/init" selber dafür sorgen, dass, wenn nötig, "/" gemountet und "/sbin/init" ausgeführt wird.
Was zunächst vielleicht erst wie ein Nachteil klingt, weil nun "/init" das eigentliche Laufwerk mounten muss, ist eigentlich als Vorteil gedacht. Auf diese Weise kann der Bootvorgang deutlich flexibler gehalten werden. Während man früher zwangsweise "/sbin/init" ausführen musste, kann man nun auch eine andere Datei ausführen, oder das so formatierte System einfach beibehalten. Gerade eingebettete Systeme müssen damit nicht extra ein neues Laufwerk mounten, nur weil das Tool danach noch eine weitere Datei ausführen will.

6 Fazit

Schauen wir uns nun noch einmal die Features der jeweiligen Dateisysteme tabellarisch an:

	RAM-Disk	**ramfs**	**tmpfs**
Dateisystem	X	√	√
Größe	fest	dynamisch erweiterbar	dynamisch erweiterbar, kann limitiert werden
Voller Speicher	Wirft Fehlermeldung, Rest RAM kann genutzt werden	Erweitert den Speicher; ist RAM voll ggf. Absturz	Erweitert den Speicher; bis Limit erreicht wird, dann Fehlermeldung
Swap	X	X	√

Hier erkennt man auch deutlich die historische Entwicklung von der RAM-Disk bis hin zu richtigen RAM-Dateisystemen wie "tmpfs". Angefangen von einer Software, die sehr

maschinennah den RAM angesprochen hat, wurde das Konzept dahingehend abstrahiert, dass heutige Software den virtuellen Speicher nutzen. Durch diese Entwicklung hat zumindest das Konzept Einzug in alle größeren Betriebssysteme gefunden, wenn auch nicht überall gleich stark.
Die Tools "ramfs" und "tmpfs" sind, wie oben bereits erklärt, Entwicklungen für das Unix-Umfeld. Gerade im Linux-Kernel ist "tmpfs" kaum noch wegzudenken. Bei jedem Systemstart wird mit Hilfe von "initramfs" der RAM damit formatiert und so beschleunigt. Ferner ist man mit "tmpfs" in der Lage seine temporären Dateien schneller abrufbar zu machen, in dem man es für "/tmp" nutzt. Für man diese Idee weiter und nutzt es sogar für "/" und schiebt so das gesamte Betriebssystem in den RAM, so kann man auf diese Weise die Performance des Servers deutlich steigern.
Unter Windows lassen sich solche Features kaum finden. Eine konkrete Realisierung von "tmpfs" gibt es dort nicht. Man kann lediglich ein Vergleich zu dem Konzept der "Temporären Dateien" ziehen. So werden Dateien, die mit dem Flag "FILE_ATTRIBUTE_TEMPORARY" sowie "FILE_FLAG_DELETE_ON_CLOSE" gesetzt sind, im Virtuellen Speicher abgelegt und nur auf die Festplatte kopiert, wenn der RAM voll ist. Damit ist man zwar in der Lage schnell auf diese Daten zugreifen zu können, aber gar einen Server vollständig im RAM auszuführen versucht man hier hier vergebens. Um sein RAM wirklich mit einem Dateisystem zu formatieren muss man auf das veraltete Konzept der RAM-Disk zurückgreifen und das mit einer Software von einem Drittanbieter.

7 Quellenverzeichnis

- http://de.wikipedia.org/wiki/RAM-Disk
- http://en.wikipedia.org/wiki/RAM_drive
- http://wiki.ubuntuusers.de/RAM-Disk_erstellen
- http://en.wikipedia.org/wiki/Tmpfs
- http://www.solarisinternals.com/si/reading/tmpfs.pdf

- http://cs3.ist.unomaha.edu/štanw/papers/86-vnode.pdf
- http://www.mjmwired.net/kernel/Documentation/filesystems/ramfs-rootfs-initramfs.txt
- http://www.mjmwired.net/kernel/Documentation/filesystems/tmpfs.txt
- http://en.wikipedia.org/wiki/Initramfs#Initramfs_in_comparison_with_initrd
- http://de.wikipedia.org/wiki/Initramfs