

# Skalierbarkeit in verteilten Systemen – Architekturen für verteilte Dateisysteme

Markus Brückner (vortrag@slash-me.net)  
<http://www.slash-me.net/work.html>

2004-06-02

## Einführung

### Was ist *Skalierbarkeit*?

Fähigkeit zur Handhabung wechselnder Lasten ohne grundsätzliche Änderung des Systems

Zwei Arten von Skalierbarkeit:

**Last-Skalierbarkeit** Konstantes Systemverhalten über große Lastbereiche hinweg

**administrative Skalierbarkeit** Möglichkeit zur effizienten Verwaltung großer und stark fluktuierender Subjekt- und Objektmengen

## Problemstellungen bei verteilten Dateisystemen

**Nutzeranzahl** Mit steigender Nutzerzahl steigt sowohl die Last im System durch parallele Zugriffe, als auch der Aufwand zur Administration des Systems (speziell bei der Nutzerverwaltung).

**Datenmenge** Größere Datenmengen sowohl in einzelnen Objekten, als auch im System insgesamt führen zu einer erhöhten Netzwerklast. Die Menge der Daten im System korreliert oft mit der Nutzeranzahl, so daß hier beide Faktoren zusammenwirken.

**Zugriffsverhalten** Abhängig vom Nutzungsprofil (wenige langfristige parallele Zugriffe oder viele kurze Operationen) bieten sich unterschiedliche Strategien zur Lastverteilung an. Ein verteiltes Dateisystem sollte die Möglichkeit bieten, detaillierte Anpassungen an das Nutzungsprofil einzelner Objekte vorzunehmen.

**verfügbare Netzbandbreite und -latenz** Bei sinkender Bandbreite und/oder steigender Latenz werden Maßnahmen nötig, die die Performanz des Dateisystems erhalten.

## Mittel zur Steigerung der Skalierbarkeit

**Replikation** Unter Replikation versteht man das mehrfache Vorhalten identischer Kopien bestimmter Daten zum Zweck der Lastverteilung und Erhöhung der Verfügbarkeit. Grundsätzlich ist dabei zu unterscheiden zwischen einfachen *read-only* Repliken, bei denen nur ein Master schreibfähig ist, während von den Kopien nur gelesen werden kann und *read-write* Repliken, bei denen sämtliche Kopien sowohl lesbar als auch schreibfähig sind. Vorteile der Replikation von Daten sind einfache Erweiterbarkeit durch zusätzliche Server sowie lineare Skalierung der Leistungsfähigkeit mit der Anzahl der Serverknoten bei effizienter Verteilung der Anfragen. Nachteilig wirken sich komplexe Synchronisationsalgorithmen bei mehreren schreibfähigen Kopien, ein steigendes Kommunikationsaufkommen zur Sicherstellung der Identität der Repliken sowie ein erhöhter Sicherheitsaufwand aus.

**Caching** Caching ist die für den Anwender transparente Vorhaltung lokaler Kopien der bearbeiteten Daten in einem kleinen, schnellen Zwischenspeicher. Dieses Verfahren wird meist zur Performanzsteigerung, seltener zur Erhöhung der Verfügbarkeit eingesetzt. Vorteilhaft ist die Senkung der Anfragemenge auf dem Server und die Steigerung der Zugriffsgeschwindigkeit. Ein Nachteil ist die notwendige Sicherstellung der Konsistenz zwischen Cache und Server.

**Callbacks** Eine Möglichkeit zur Wahrung der Cache-Konsistenz sind Callbacks. Beim Zugriff auf ein entferntes Objekt gibt der Server ein Versprechen (*callback token*) ab, den Client vor Änderung des Objektes zu informieren. Wird das Objekt nun geändert, wird dieses Versprechen gebrochen (*callback break*) und damit die Daten im Cache des Clients invalidiert. Vorteilhaft ist eine drastische Senkung der Netzlast, da keinerlei Konsistenzprüfung seitens des Clients notwendig ist, solange er im Besitz eines gültigen Callbacks ist. Nachteilig wirkt sich der erhöhte Verwaltungsaufwand auf dem Server aus, der für jedes abgegebene Callback einen Zustand halten muss.

**Hints** Hints sind Informationen, welche bei Korrektheit eine Operation beschleunigen, bei Fehlerhaftigkeit aber nicht zu falschen Ergebnissen führen. Aus diesem Grund können Hints gefahrlos im Cache gehalten werden ohne auf ihre Richtigkeit zu achten. Bei der Verwendung dieser Informationen ergibt sich ihre Validität automatisch. Leider sind nur wenige Informationen in einem verteilten Dateisystem als Hints geeignet. Beispielhaft dafür stehen Ortsinformationen bestimmter Daten. Sind diese richtig, kann der Client sofort auf die Daten zugreifen. Sind sie falsch, sind die entsprechenden Daten nicht zugreifbar und die Ortsinformationen müssen erneut angefordert werden.

**Zentrale Nutzerverwaltung** Bei großen Installationen verteilter Dateisysteme ist eine manuelle Pflege von Nutzerdatenbanken auf jedem Server unmöglich. Daher muss auf ein zentralisiertes System zurückgegriffen werden. Möglichkeiten zur Realisierung eines solchen Systems wären ein zentraler Login-Server, welcher begrenzt gültige *Token* ausstellt, wie bei Kerberos, eine Public Key Infrastruktur oder die automatische Replikation der Nutzerdatenbanken über alle Server.

**Gruppen/Rechtevererbung** Um möglichst einfach die hierarchischen Strukturen einer Organisation im Zugriffsmodell eines Dateisystems nachzubilden, können Gruppen zum Einsatz kommen. Ein Nutzer kann dabei mehreren Gruppen angehören. Desweiteren kann es möglich sein, Gruppen wiederum in weitere Gruppen zu schachteln. Die Rechte eines Nutzers sind dabei immer die Summe aller Rechte der Gruppen, denen er direkt oder indirekt angehört.

**Negative Rechte** In einem weit verzweigten verteilten Dateisystem kann es wünschenswert sein, einem Nutzer Rechte auf ein bestimmtes Objekt zu entziehen, die er aufgrund seiner Mitgliedschaft in einer Gruppe bereits hat. Zu diesem Zweck existieren negative Rechte, welche – verankert in der Access Control List des betreffenden Objektes – von den vorhandenen Rechten eines Nutzers abgezogen werden.