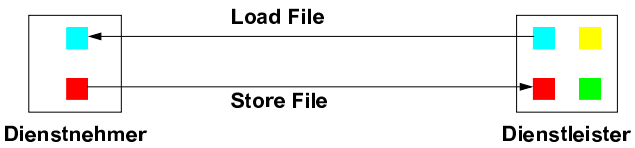
3.3 Verteilte Dateisysteme

Architektur

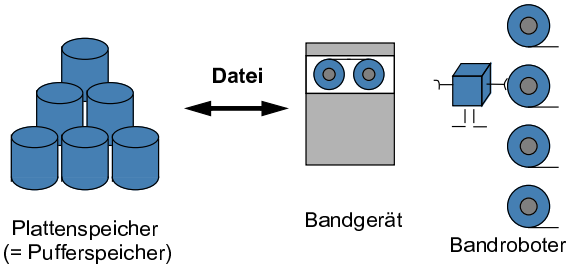
Dateidienst-Interface

- Verlagerungsmodell (upload/download model)
  - Ganze Dateien werden vom Dienstleister zum Dienstnehmer transferiert und dort bearbeitet



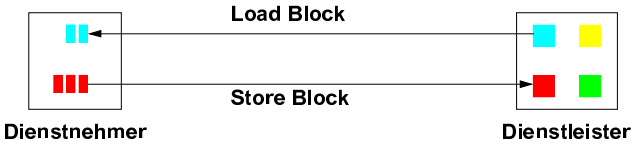
  - Nach Beendigung der Arbeit werden sie zum Dienstleister zurücktransferiert

- Typisch für Massenspeichersysteme, z. B. Unitree



Ebene i-1:	Plattenspeicher
Ebene i:	Magnetbandroboter
Granulat:	Datei
write on hit:	write back
write on miss:	allocating
Kohärenz:	keine besonderen Maßnahmen erforderlich

Fernzugriffsmodell (remote access model)



- Nur die benötigten Blöcke werden zum Dienstnehmer transferiert
- Nicht mehr benötigte Blöcke werden zurücktransferiert
- Weiterer Gestaltungsspielraum:
  write back/through?
  allocating?
  Kohärenz? (Unix: Prozessorkohärenz)
- Beispiel: NFS (Network File System von Sun)

Führung von Zustandsinformation

Zustandslose Dienstleister (stateless servers)

- Nach Bedienung eines Auftrags wird keine Information über diesen Auftrag gehalten
- Der Auftrag muß alle zu seiner Abwicklung notwendigen Informationen mitbringen (vollständige Dateinamen oder äquivalente Information, Lese-/Schreibzeiger, Zugriffserlaubnis, ...)
- Nachrichten werden dadurch länger
- Für jeden Aufruf muß erneut die Bestimmung des physikalischen Speicherorts vorgenommen werden
- Bessere Beherrschbarkeit transienter Ausfälle des Dienstleisters
- Keine Tabellen notwendig
  ⇒ Keine Beschränkung der Zahl von Dienstnehmern
- Koordinierung von Zugriffen nicht möglich
  ⇒ Besonderer Koordinator erforderlich

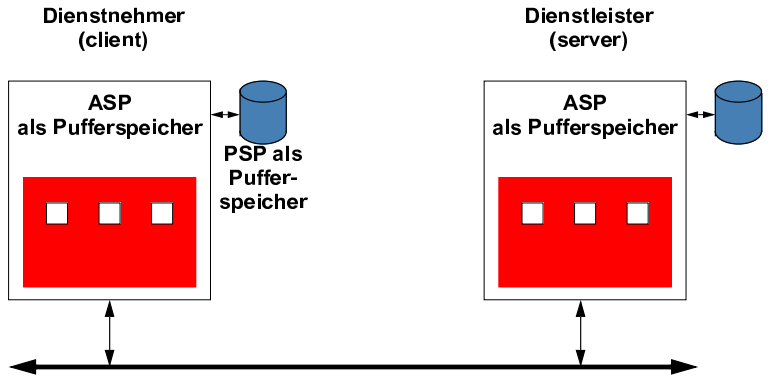
Zustandsbehaftete Dienstleister (stateful servers)

- Für jeden Dienstnehmer wird Information über den Bearbeitungszustand seiner geöffneten Dateien gehalten
- Nur der Auftrag zum Öffnen einer Datei benötigt den vollständigen Dateinnamen

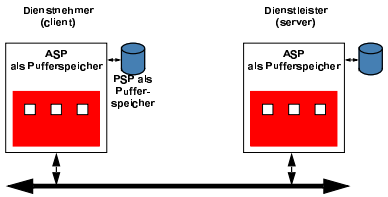
  ➡ Dienstnehmer erhält Kennung zur (fälschungssicheren) Benennung des zuständigen Dateideskriptors
- Weitere Lese-Schreibaufträge müssen nur diese Benennung mitliefern

  ➡ Lokalisierung der Information nicht von Grund auf neu vorzunehmen
- Schwieriger Wiederanlauf; erfordert Kooperation mit den Dienstnehmern

Grundsätzliche Möglichkeiten der Plazierung gepufferter Blöcke



Vor- und Nachteile von Pufferung



Dienstleister:

Pufferung im ASP reduziert E/A

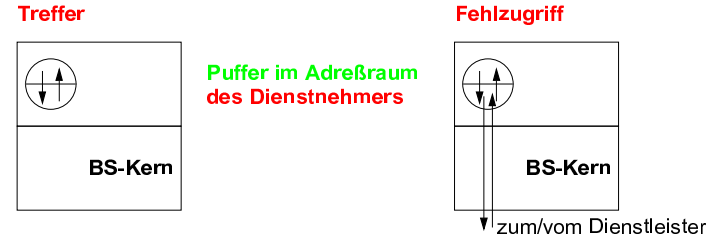
Dienstnehmer:

Pufferung beim Dienstnehmer reduziert E/A und Netzwerkverkehr
Pufferung erzeugt Konsistenzprobleme

Pufferungsgranulate:

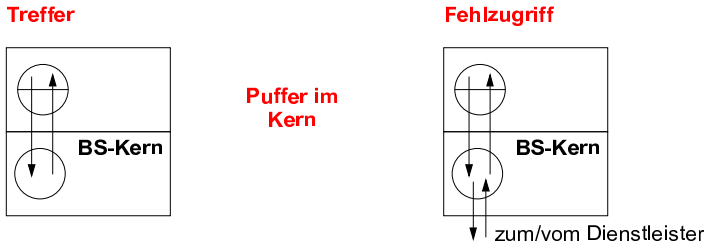
Dateien oder Dateiblöcke?

Pufferung im Adreßraum des Dienstnehmers



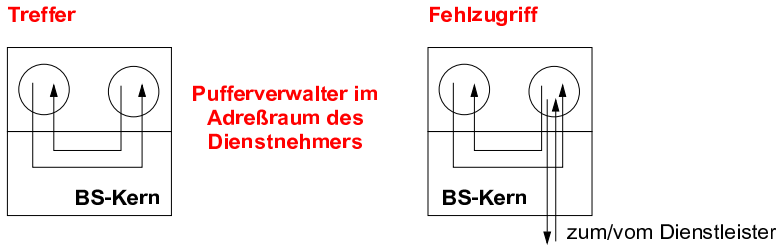
- Geringer Overhead
- Vorteilhaft, wenn Prozesse die gleiche Datei mehrfach öffnen/schließen
- Puffer wird bei Terminierung des Prozesses automatisch freigegeben

Pufferung im Adreßraum des BS-Kerns



- Jeder Zugriff ist mit einem Kernaufruf verbunden
- Puffer überleben Prozeßterminierung

Pufferung mit Pufferverwalter im Adreßraum des Dienstnehmers



- Kern frei von DFS-Code
- Puffer überlebt Prozeßterminierung
- Gepufferter Block kann im Rahmen eines Demand Paging ausgelagert werden!

Konsistenz

Write-Through

- Jede Modifikation wird sofort an den Dienstleister weitergeleitet
- Pufferung sehr effektiv für Lesen, aber nicht für Schreiben
- Wenn die Pufferung Prozeßterminierung überlebt, muß Gültigkeit mit dem Dientstleister abgeglichen werden (z. B. durch Versionsnummern)

Delayed Write

- Notieren, daß modifiziert wurde, aber kein sofortiges Informieren des Dienstleisters
- Alle Modifikationen in regelmäßigen Abständen (z. B. 30 Sekunden) zum Dienstleister senden ⇒ effizienter
- Reduziert Transporte für temporäre Dateien, die modifiziert, gelesen und getilgt sein können, bevor der Dienstleister informiert werden muß.
- Klare Semantik wird aufgegeben zugunsten größerer Effizienz
  ⇒ was andere Prozesse lesen ist zeitabhängig

Write on close

- Anpassung an Sitzungs-Semantik und Rückschreiben zum Dienstleister alle 30 Sekunden, nachdem die Datei geschlossen wurde
  ⇒ getilgte Dateien werden nicht zurückgeschickt!
- Modifikationen können verloren gehen, wenn zwei Prozesse die gleiche Datei modifizieren

Centralized controller

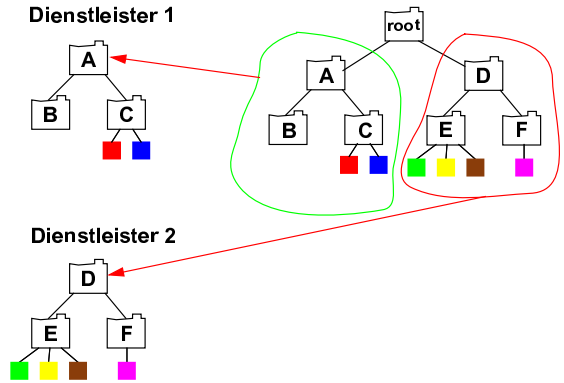
- Führt Buch über alle geöffneten Dateien und die zugehörigen Dienstnehmer
- Kollidierende Aufrufe können auf drei Arten behandelt werden:
  - Aufruf ablehnen
  - Aufruf in Warteschlange aufnehmen
  - Aufruf genehmigen, aber alle evtl. betroffenen Dienstnehmer veranlassen die Pufferung des betreffenden Blocks gegebenenfalls zu invalidieren und im weiteren nicht puffern
    ⇒ unverlangte Nachrichten an Dienstnehmer
- Skaliert nicht und ist nicht robust gegen transiente Rechnerausfälle

Beispiel: Network File System (NFS)

McKusick, M. K.; Bostic, K.; Karels, M. J.; Quarterman, J. S.: The Design and Implementation of the 4.4BSD Operating System. Addison-Wesley, 1996, 580 pages.

- Von vielen Herstellern für Unix und MS-DOS unterstützt
- Hardwareunabängig
- Architektur:
  - Dienstgeber exportieren Dateien
  - Dienstnehmer montieren sie in ihren Dateibaum
  - Inanspruchnahme von Diensten über Fernaufrufe (RPC, XDR)

Architektur



Montageprotokoll

- Dienstnehmer sendet an den Dienstgeber den Pfadnamen des Verzeichnisses, das in seinem eigenen Verzeichnis montiert werden soll.
- Wenn der Pfadname gültig ist und das Verzeichnis exportierbar ist, gibt der Dienstleister ein *file handle* zurück.

  file handle = (file system type, disk, directory's i-node number, security information)

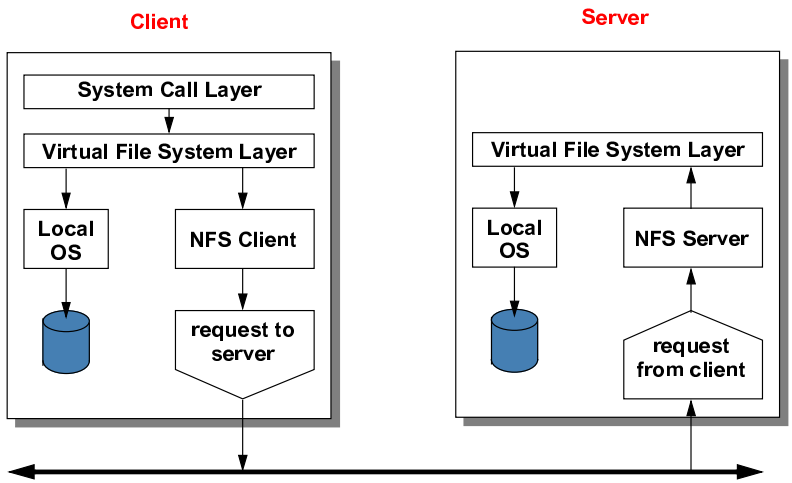
- Automounting (durch den Dienstnehmer): Wenn eine Datei geöffnet wird, werden alle in einer besonderen Tabelle verzeichneten Dienstleister kontaktiert. Montiert wird das erste angebotene Dateisubsystem.

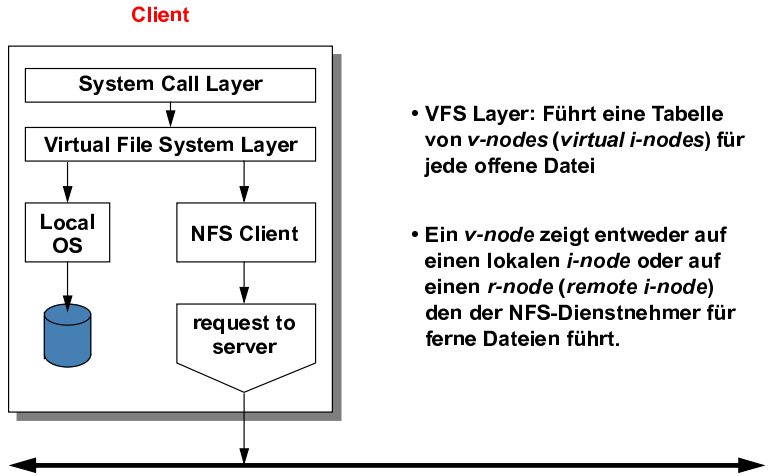
Zugriffsprotokoll

- NFS ist zustandslos
  - ⇒ keine Zustandsinformation über geöffnete Dateien
  - ⇒ es gibt in NFS keinen *open*-Aufruf
  - ⇒ es gibt keine Koordinierungsunterstützung
- Es gibt eine *lookup*-Operation:

  Vor der Bearbeitung einer Datei führt der Dienstnehmer einen Aufruf von *lookup* mit dem Pfadnamen aus. Der Dienstleister gibt ein *file handle* zurück, das bei künftigen Aufrufen benutzt wird.

- Lese- und Schreibaufrufe müssen neben dem *file handle* auch die Angabe enthalten, ab welcher Stelle wieviele Byte gelesen/geschrieben werden sollen.
- Lese- und Schreibaufrufe an NFS wirken atomar.





- VFS Layer: Führt eine Tabelle von *v-nodes* (*virtual i-nodes*) für jede offene Datei
- Ein *v-node* zeigt entweder auf einen lokalen *i-node* oder auf einen *r-node* (*remote i-node*) den der NFS-Dienstnehmer für ferne Dateien führt.

Implementierung von Sun

- Dienstnehmer puffern *i-nodes* und Dateidaten.
- Gepufferte Datenblöcke werden alle 3 Sekunden freigegeben, gepufferte Verzeichniseinträge alle 30 Sekunden.
- Wenn eine gepufferte Datei geöffnet wird, wird der Zeitpunkt ihrer letzten Modifikation überprüft. Gegebenenfalls werden neue Kopien beschafft.
- Alle 30 Sekunden werden alle im Puffer modifizierten Blöcke zurückgeschrieben.
- Der Dienstleister benutzt eigene Pufferungsmechanismen zur Reduktion des Verkehrs mit seinen Plattenspeichern.
- Übertragen werden jeweils 8 KB.
- Wenn die VFS Ebene 8 KB empfängt, fordert sie sofort die nächsten 8 KB an.
- Beim Schließen einer Datei erfolgt ein *write back*.