Dateisysteme

Im Clustersystem gibt es drei Speichersysteme. Jedes Speichersystem stellt eines der für Ihre Arbeit wichtigen Dateisysteme zur Verfügung: $HOME, $BIGWORK und $PROJECT.

Die ersten beiden Dateisysteme – $HOME und $BIGWORK – sind auf allen Knoten im Cluster montiert. $PROJECT ist nur auf den Login- und dem Transferknoten verfügbar. Einige relevante Eigenschaften der Dateisysteme sind aus Abbildung figure 1 ersichtlich.

Fig. 1: Cluster file systems with specifications

Verwendung der drei wichtigen Dateisysteme

$HOME Ihr Home-Verzeichnis ist das Verzeichnis, in welchem Sie sich normalerweise direkt nach dem Login auf der Kommandozeile befinden. Hier sollten nur kleine und besonders wichtige Dateien gespeichert werden, deren Neuerstellung z.B. nach einem Datenverlust besonders viel Zeit kosten würde, also Schablonen für Job Setups, wichtige Scripte, Konfigurationsdateien für Software etc.

$HOME wird von nur einem Server bereit gestellt, der per Gigabit Ethernet angebunden ist. Im Vergleich zu $BIGWORK ist dieses Dateisystem sehr langsam, und es ist insbesondere nicht für die Bearbeitung datenintensiver Aufgaben geeignet. Wenn Sie $HOME überlasten, merken das alle anderen Nutzer auch.

Wichtig: Wenn Sie in diesem Verzeichnis ihre Quota (siehe folgender Abschnitt) überschreiten, – und das passiert schnell, wenn einer Ihrer Rechenjobs große Datenmengen schreibt und Sie versehentlich oder gar (tun Sie das nicht!) absichtlich ins $HOME schreiben – können Sie sich nicht mehr grafisch mit Tools wie z.B. X2Go einloggen und müssen dann zuerst Daten löschen, bevor es weiter geht.

In Ihren Rechnungen sollten Sie gar nicht auf Daten in $HOME zugreifen, weil das aufgrund der oben beschriebenen technischen Gegebenheiten dazu führen kann, dass Ihr Job zur Ausführung erheblich länger benötigt und dabei andere Nutzer bei der Arbeit stört.

Ebenso sollte darauf geachtet werden, alle etwa automatisch von Anwendungssoftware gesetzten Verzeichnisse – oft sind dies temporäre Verzeichnisse – auf $BIGWORK zeigen zu lassen. Und schließlich sollten sie auch NICHT versuchen, in einer Rechnung mit einem symbolischen Link zwischen $BIGWORK und $HOME die Gegebenheiten kreativ zu umgehen. Das nützt nämlich nichts, die Folgen sind beinahe identisch, wie wenn man einfach direkt das Falsche täte. Verwenden Sie zum bequemen Zugriff in einer Rechnung besser die Umgebungsvariable $BIGWORK – und ansonsten nichts, was sich in Ihrem $HOME befindet. Siehe dazu auch die Übung in Kapitel 1.5.

Wichtig: Never WORK at HOME. Tun Sie’s nicht. Versuchen Sie nicht, $HOME als Backupverzeichnis für Ihre Rechenergebnisse zu missbrauchen - dafür ist es einfach nicht geeignet. Wir haben praktisch jede Woche zwei neue Anfragen von Nutzern, die sich selbst Probleme erzeugt haben, weil sie das nicht beachtet haben – versuchen Sie, nicht dazu zu gehören.

Durch die Begrenzung des Speicherplatzes und die implizite Beschränkung auf langsamveränderliche Daten ist es dem System möglich, ein tägliches Backup der Daten im zu $HOME erstellen.

$HOME ist auf allen Knoten im Cluster montiert.

$BIGWORK Ihr Arbeitsverzeichnis. Hier steht eine vergleichsweise große Menge Speicherplatz zur Verfügung. Weil die Datenmengen so groß sind und sich die Daten relativ schnell ändern, ist es nicht möglich, ein automatisches Backup zu erstellen. In diesem Verzeichnis findet der Großteil Ihrer Arbeit statt, alle Rechnungen sollten grundsätzlich hier ausgeführt werden. ist per InfiniBand und somit im Vergleich zu deutlich schneller angebunden, und es wird von einer ganzen Gruppe von Servern bereit gestellt, um die Leistung weiter zu steigern. Eine ebenfalls verwendete Bezeichnung ist Scratch.

$BIGWORK ist auf allen Knoten im Cluster montiert.

$PROJECT Ihr Projektverzeichnis. Jedem Projekt wird ein Projektverzeichnis zugeordnet, auf welches alle Mitglieder eines Projekts Lese- und Schreibzugriff haben unter `/project/<your-cluster-groupname>` (die Umgebungsvariable $PROJECT zeigt auf diesen Pfad). Um die Daten einzelner Nutzeraccounts so zu speichern, dass der Überblick erhalten bleibt, wird empfohlen, sich unter dem jeweiligen Account ein eigenes Unterverzeichnis namens $PROJECT/$USER zu erstellen und dieses mit den passenden Zugriffsrechten (z.B. `mkdir -m 0700`) zu versehen.

Die Quota (Speicherplatzbeschränkung) eines Projektverzeichnisses beträgt normalerweise 10 TB. Das Verzeichnis steht nur auf den Login- und dem Transferknoten zur Verfügung, was bedeutet, dass Sie es nicht in Jobs auf den Rechenknoten benutzen können (verwenden Sie dafür bitte $BIGWORK). Auch ein Kopieren zwischen $BIGWORK und $PROJECT ist nur auf den Login- und – bevorzugt, da für solche Aufgaben vorgesehen – dem Transferknoten möglich. Dort besteht zwischen den beiden Dateisystemen eine schnelle Verbindung. Der Projektspeicher ist ein eigener, von $BIGWORK getrennter und schneller Speicher mit einer breitbandigen Anbindung (Lustre, Infiniband). Er ist für die langfristige Ablage von Ausgangs- oder Ergebnisdaten im Rechennetz gedacht und so konfiguriert, dass die Daten physikalisch doppelt gespeichert werden. Aufgrund der Datenmenge gibt es allerdings kein gesondertes Backup.

Wichtig: Regelmäßiges Sichern Ihrer Daten von auf dem -Speicher oder auf die Rechner des Instituts ist unerlässlich, da als Scratch-Speichersystem konzipiert ist.

Quota und grace time - Kontingente und "Gnadenfristen"

Auf den Speichersystemen wird Ihnen bzw. Ihrer Nutzerkennung jeweils nur ein Teil des gesamten Speicherplatzes zur Verfügung gestellt. Dieses Kontingent wird mit dem englischen Begriff bezeichnet. Es wird unterschieden in und . Das ist eine obere Grenze. Es kann nicht mehr Speicherplatz verbraucht werden, als durch das vorgegeben. Das hingegen darf für einige Zeit – die Gnadenfrist – überschritten werden. Überschreitet der Speicherplatzverbrauch den durch das vorgegebenen Wert, so beginnt die . Das ist die Zeit, für die man das erlaubt überschreiten darf. Nach Ablauf dieser Frist muss der Speicherplatzverbrauch unter die Schwelle des gesenkt werden, ansonsten ist kein weiteres Schreiben auf das Speichersystem möglich. Durch Unterschreiten des wird die für das betroffene Dateisystem und die jeweilige Grenze zurückgesetzt.

Der Quotamechanismus dient dazu, Nutzer und System vor Fehlern anderer zu schützen, den einzelnen Nutzern zur Verfügung stehenden Speicherplatz zu begrenzen und das System möglichst performant zu halten. Wenn es z.B. möglich wäre, dass ein einzelner Account durch einen Programmierfehler oder auch aus Unwissen das gesamte vollschreiben könnte, hätte das negative Folgen für alle Nutzer. Generell sollten nicht mehr benötigte Dateien gelöscht werden. Ein niedriger Füllstand sichert insbesondere bei einen optimalen Betrieb. Die Abfrage von eigenem

Speicherplatzverbrauch und erfolgt durch folgende Befehle – siehe auch die Übung in Kapitel 1.5.

- **fquota** Zeigt den Speicherplatzverbrauch und von $HOME an.
- **lquota** Zeigt den Speicherplatzverbrauch und von BIGWORK und $PROJECT an.

Wichtig: Ist Ihr Speicherkontingent auf $HOME ausgeschöpft, scheitert jede grafische Anmeldung. Eine Anmeldung per ssh (ohne -X) ist weiterhin möglich.

$BIGWORK's Dateisystem Lustre und stripe count

Wichtig: Alle Angaben in diesem Abschnitt gelten auch für den $PROJECT-Speicher

Technisch betrachtet besteht $BIGWORK aus mehreren Einzelkomponenten, die zusammen das Speichersystem ergeben, dargestellt in Abbildung [fig:bigwork]. Grundsätzlich kann $BIGWORK verwendet werden, ohne an den Standardeinstellungen etwas zu verändern. Unter Umständen kann es jedoch sinnvoll sein, den sogenannten anzupassen. Dadurch kann besonders bei großen Dateien und in parallelen Rechnungnen, welche auf unterschiedliche Teile einer Datei zugreifen, eine höhere Leistung erzielt werden und das Gesamtsystem wird ausgeglichener ausgelastet, was der gesamten Nutzerschaft zugute kommt.

Auf $BIGWORK werden die Daten auf OSTs, , gespeichert. Jedes OST besteht seinerseits aus einer Anzahl von Festplatten. Standardmäßig werden Dateien, unabhängig von ihrer Größe, auf einem einzigen OST gespeichert. Dies entspricht einem von eins. Der gibt an, auf wie viele OSTs eine Datei aufgeteilt wird, bildlich beschrieben also, in wie viele Streifen eine Datei gedanklich aufgeteilt wird. Durch die Aufteilung auf mehrere OSTs können höhere Zugriffsgeschwindigkeiten erreicht werden, da die Schreib- und Lesegeschwindigkeiten mehrerer OSTs und somit einer Vielzahl von Festplatten genutzt werden. Gleichzeitig sollte man möglichst nur große Dateien so verteilen, da sich die Zugriffszeiten auch erhöhen können, wenn zu viele kleine Anfragen ans Dateisystem gestellt werden. Je nach persönlichem Nutzungsszenario muss man also ein wenig experimentieren, um herauszufinden, welche Einstellung die besten Ergebnisse erzielt.

Wichtig: Wer mit Dateien arbeitet, welche größer als 1 GB sind, und bei denen Zugriffszeiten z.B. im Rahmen von parallelen Rechnungen wesentlich zur Gesamtdauer einer Rechnung beitragen, sollte über eine Anpassung des nachdenken – siehe Kapitel 1.6.

Der *stripe count* wird als Zahl der zu verwendenden OSTs gesetzt, wobei -1 für alle verfügbaren OSTs steht. Es bietet sich an, ein Verzeichnis unterhalb von $BIGWORK anzulegen, welches einen von -1 erhält und dort z.B. alle Dateien, welche größer als 100 MB sind, zu speichern. Für kleine Dateien – erheblich kleiner als 100 MB – ist der voreingestellte *stripe count* von eins besser und auch ausreichend.

Wichtig: Um für bestehende Dateien den anzupassen, müssen diese kopiert werden, siehe Kapitel 1.7. Ein Verschieben mit `mv` reicht nicht aus.

$TMPDIR

Innerhalb eines Jobs steht unter der Variable $TMPDIR ein Verzeichnis auf den Knoten, welche am Job teilnehmen, lokal zur Verfügung. Wenn Speicherplatz lokal benötigt wird sollte man dieses Verzeichnis

benutzen.

Wichtig: Nach Ende des Jobs werden alle in $TMPDIR gespeicherten Dateien automatisch gelöscht.

Wer denkt, auf $TMPDIR könnte grundsätzlich schneller geschrieben werden als auf $BIGWORK, sollte dies unbedingt durch Tests verifizieren. $TMPDIR bietet sich für temporäre Dateien an bei Anwendungssoftware, welche zwingend ein gesondertes temporäres Verzeichnis voraussetzt.

Übung: Dateisysteme benutzen

```
# where are you? lost? print working directory!
pwd

# change directory to your bigwork/project/home directory
cd $BIGWORK
cd $PROJECT
cd $HOME

# display your home, bigwork & project quota
checkquota

# make personal directory in your group's project storage
# set permissions (-m) so only your account can access
# the files in it (0700)
mkdir -m 0700 $PROJECT/$USER

# copy the directory mydir from bigwork to project
cp -r $BIGWORK/mydir $PROJECT/$USER
```

Fortgeschrittene Übung: stripe count setzen

```
# get overall bigwork usage, note different fill levels
lfs df -h

# get current stripe settings for your bigwork
lfs getstripe $BIGWORK

# change directory to your bigwork
cd $BIGWORK

# create a directory for large files (anything over 100 MB)
mkdir LargeFiles

# get current stripe settings for that directory
lfs getstripe LargeFiles

# set stripe count to -1 (all available OSTs)
```

```
lfs setstripe -c -1 LargeFiles

# check current stripe settings for LargeFiles directory
lfs getstripe LargeFiles

# create a directory for small files
mkdir SmallFiles

# check stripe information for SmallFiles directory
lfs getstripe SmallFiles
```

Use newly created LargeFiles directory to store large files

Fortgeschrittene Übung: stripe count anpassen

Falls Sie nicht wissen, wie groß die Dateien werden, welche Ihre Anwendung erzeugt, können Sie *stripe size* nachträglich setzen. Als erstes wird eine 100 MB große Datei erstellt.

```
# enter the directory for small files
cd SmallFiles

# create a 100 MB file
dd if=/dev/zero of=100mb.file bs=10M count=10

# check filesize by listing directory contents
ls -lh

# check stripe information on 100mb.file
lfs getstripe 100mb.file

# move the file into the large files directory
mv 100mb.file ../LargeFiles/

# check if stripe information of 100mb.file changed
lfs getstripe ../LargeFiles/100mb.file

# remove the file
rm ../LargeFiles/100mb.file
```

Um den stripe zu ändern, müssen Dateien kopiert (`cp`) werden. Verschieben (`mv`) reicht nicht aus.

Erste Methode:

```
# from within the small files directory
cd $BIGWORK/SmallFiles

# create a 100 MB file
dd if=/dev/zero of=100mb.file bs=10M count=10
```

```
# copy file into the LargeFiles directory
cp 100mb.file ../LargeFiles/

# check stripe in the new location
lfs getstripe ../LargeFiles/100mb.file
```

Zweite Methode:

```
# create empty file with appropriate stripe count
lfs setstripe -c -1 empty.file

# check stripe information of empty file
lfs getstripe empty.file

# copy file "in place"
cp 100mb.file empty.file

# check that empty.file now has a size of 100 MB
ls -lh

# remove the original 100mb.file and work with empty.file
rm 100mb.file
```