# Fragebogen und allgemeine Informationen zum Hardware-Sizing

## Version 2.0

| | |
|---|---|
| **Version** | **2.0** |
| **Status** | **in process** |
| Abteilung | Professional Services |
| Autor/ Autoren | Dr. Raphael Richter |

# Inhaltsverzeichnis



# 1 Einleitung

FirstSpirit ist ein Enterprise-Produkt, entwickelt und modelliert für den Einsatz in massive Mehrbenutzer-Szenarien. Bei der Planung der Einsatzszenarien ist die Frage der Dimensionierung (Sizing) unter dem Aspekt der Skalierbarkeit im dauerhaften Betrieb maßgeblich.

Ein Dimensionierungsprozess stellt sich üblicherweise wie folgt dar:

- Erhebung von Projektdaten nach Fragebogen
- Bestimmung der Dimensionierungs-Kennzahlen
- Auswahl der Architektur
- Auswahl der Hardware

Die folgenden Kapitel dienen zur Information und Datenerhebung, um zu einer fundierten Hardware-Sizing-Aussage zu kommen.

Aufgrund der vielen (insbesondere projektabhängigen) Faktoren, welche den Leistungsbedarf beeinflussen, ist es nur schwer möglich, Kalkulationsformeln für CPU-, Festplattenplatz- oder RAM-Bedarf bereit zu stellen, so dass für eine fundierte Aussage zum Bedarf und der notwendigen Konfigurationen immer Architekten oder Consultants der e-Spirit AG beim Hardware-Sizing mit einbezogen werden sollten.

Da die FirstSpirit Architektur eine strikte Trennung zwischen Redaktions- und Live-System vorsieht, kann ein Hardware-Sizing FirstSpirit seitig immer nur die Dimensionierung des Redaktions-Systems (FirstSpirit Master-Server und ggf. Generierungs-/Backup- oder Preview-Slaves) betreffen.

Aussagen zur Dimensionierung des Live-Systems werden in der Regel nicht getroffen.

Werden für die Anbindung einer oder mehrerer externer Datenbanken dedizierte Datenbank-Server in Betracht gezogen, so verweisen wir hier auf die Release-Notes der Datenbank-Hersteller, was die Dimensionierung der Datenbank-Server angeht.

# 2 Erfassung spezifischer Daten

Die folgenden Fragen sollen zur Erfassung der wichtigsten Rahmen-Parameter dienen, die zur Ermittlung eines Hardware-Sizings notwendig sind.

1. Wie hoch ist die Gesamtanzahl der Nutzer innerhalb des Firmen-Intra- bzw. Extranets?

   _____

2. Wie hoch ist die erwartete Gesamtanzahl an FirstSpirit-Nutzern (named)?

| Benutzer-Typ | Anzahl |
|---|---|
| Redakteur | |
| (Projekt-)Administrator | |
| Entwickler | |

3. Wie hoch ist die erwartete Gesamtanzahl an gleichzeitig arbeitenden FirstSpirit-Nutzern (concurrent)?

| Benutzer-Typ | Anzahl |
|---|---|
| Redakteur (JavaClient) | |
| Redakteur (WebEdit) | |
| (Projekt-)Administrator | |
| Entwickler | |

4. Wie hoch ist die erwartete Anzahl an Projekten auf dem FirstSpirit-Server?

   _____

5. Spezifizieren Sie die erwartete durchschnittliche Anzahl von Seiten und Medien pro FirstSpirit-Projekt und das erwartete monatliche Wachstum

| Typ | Anzahl | Größe | mon. Wachstum in % |
|---|---|---|---|
| Medien | 500 | 5MB | 0,5 |
| Seiten | 10.000 | 60K | 0,5 |

6. Bestehen Abhängigkeiten zwischen den Projekten?

   ☐ Nein
   ☐ Ja

   Wenn ja, welche FirstSprit-Module/-Funktionalitäten werden dazu genutzt?

   ☐ CorporateContent
   ☐ CorporateMedia
   ☐ CorporateDatabase
   ☐ lesend eingebundene Datenquellen
   ☐ abhängige Deployments

7. Werden externe Datenbanken zur Verwaltung von Daten über FirstSpirit interne Datenquellen oder lesend eingebunden?

   ☐ Nein
   ☐ Ja

   Wenn ja, welche, und wie ist die erwartete Größe der in den Datenbanken verwaltenen Daten?

| Datenbank-Typ | Art | Größe |
|---|---|---|
| mySQL | r/w | x GB |
| Oracle | r | y GB |
| DB2 | r/w | z GB |
| … | … | … |

8. Welche Veröffentlichungs-Arten werden innerhalb der Projekte genutzt?

   ☐ (Zeitgesteuerte) Komplett-Veröffentlichung
   ☐ (Zeitgesteuerte) Teil-Veröffentlichungen
   ☐ Manuell ausgeführte Teil-Veröffentlichungen
   ☐ projektspezifische Freigabe-Workflows mit anschließender Veröffentlichung
   ☐ projektspezifische Veröffentlichungs-Queue

9. Ungefähre Anzahl von parallel zum Redaktionsbetrieb gleichzeitig laufenden Veröffentlichungen
   ___ Veröffentlichungen mit durchschnittlich jeweils ___ Seiten/Medien

10. Ungefähre Anzahl von parallel zum Redaktionsbetrieb gleichzeitig laufenden Projekt-Backups
    ___ Snapshot-Backups
    ___ inkrementelle Backups
    ___ differentielle Backups

11. Wieviele und welche Umgebungen werden vorgehalten?

    ☐ Entwicklung
    ☐ Test
    ☐ Produktion

12. Soll oder kann Virtualisierungs-Software eingesetzt werden?

    ☐ Nein
    ☐ Ja

    Wenn ja, welche?
    _____

13. Muss bestehender Content aus einem oder mehreren Alt-Systemen migriert werden?

    ☐ Nein
    ☐ Ja

14. Wenn ja, geben Sie die folgenden Kennzahlen an:

| System | Objekt-Typ | Anzahl | Gesamt-Größe |
|---|---|---|---|
| x1 | Seiten/Artikel | y1 | z1 GB |
| x2 | Medien | y2 | z2 GB |

15. Sollen externe Applikationen in die Redaktions-Umgebung integriert werden?

    ☐ Nein
    ☐ Ja

16. Wenn ja, welche?

    _____
    _____
    _____
    _____
    _____
    _____

17. Teilen Sie uns weitere Informationen mit, von denen Sie denken, dass diese wichtig für die Definition der Hardware-Anforderungen sein könnten.

    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____

# 3 Weiterführende Informationen

## 3.1 Hardware-Architektur / Begrifflichkeiten

Die Backend (CMS) Hardware-Architektur sieht im Allgemeinen die folgenden Systemkomponenten (größtenteils als physikalische Hardware) vor:

- **FirstSpirit Master Server**
  Auf diesem läuft der eigentliche FirstSpirit Server als Java-Anwendung
- **FirstSpirit Content Generation Server**
  Auf diesen Server können Generierungs-Aktionen ausgelagert werden. Er benötigt Lesezugriff auf das Repository des FirstSpirit Master Servers und Schreibzugriff auf das Generierungsverzeichnis
- **FirstSpirit Enterprise Backup Server**
  Auf diesen Server können Backup-Aktionen ausgelagert werden. Er benötigt Lesezugriff auf das Repository des FirstSpirit Master Servers und Schreibzugriff auf das Backup-Verzeichnis
- **FirstSpirit Frontend Server**
  Auf diesen Server wird der Application Server (Tomcat, Websphere, …) für die Preview und die http-Client-Server-Kommunikation ausgelagert
- **FirstSpirit Datenbank Server**
  Auf diesem befindet sich die für die internen Datenquellen verwendete Datenbank
- **FirstSpirit Webserver Frontend**
  Auf diesem wird ein dem FirstSpirit Frontend Server vorgelagerter Webserver (Apache, …) ausgelagert. Er dient meist als Loadbalancer für die dahinter liegenden **FirstSpirit Frontend Server.** Auch wird er eingesetzt, wenn z.B. PHP in der Preview des FirstSpirit-Servers genutzt werden soll

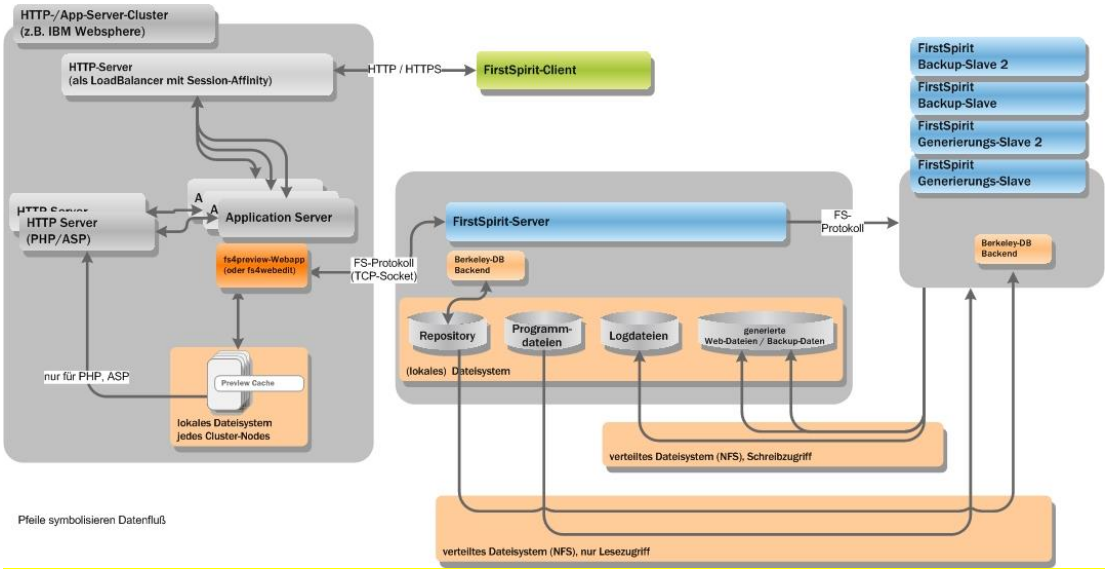


**Abbildung 1 Allgemeine Übersicht einer FirstSpirit Lastverteilungs-Architektur**

## 3.2 Allgemeine Informationen

Der FirstSpirit-Server ist für den Einsatz in Mehrprozessor-System konzipiert und nutzt die verfügbaren Prozessoren bzw. Rechenkerne effizient aus. Der Einsatz von Mehrprozessor-Systemen wird daher in jedem Fall empfohlen. Auch der Einsatz einer 64Bit-fähigen Hardware ist unbedingt ratsam, auch wenn diese zunächst nicht benötigt wird (falls ein 32Bit-Betriebssystem verwendet wird).

## 3.3 Dateisystem/Festplatten

Da der FirstSpirit-Server fast alle Daten in einem sehr effizienten, dateisystembasierten Repository verwaltet, ist dafür zu sorgen, dass ein möglichst leistungsfähiges Festplattensubsystem verwendet wird.

Die beste Performance wird in der Praxis mit Fibre-Channel RAID-Systemen mit viel Cache-RAM (2 GB Cache) und einem großen RAID6-Verbund im SAN erzielt.

Interne RAID-Systeme sind ebenfalls möglich. Ein Einsatz von NAS-Systemen und NFS sollte nur im Einzelfall und nach sehr sorgfältiger Planung und Performance-Analyse erfolgen.

NFS für den lesenden Zugriff eines Content Generation Servers bzw. Enterprise Backup Servers auf das FirstSpirit-Repository und den schreibenden Zugriff auf das Generierungsverzeichnis bzw. das Backup-Verzeichnis wird jedoch in vielen

Umgebungen problemlos eingesetzt.

Speziell beim Einsatz von FirstSpirit in einem verteilten System ist zu beachten, dass ausreichende Netzwerkbandbreite zwischen den einzelnen Serversystemen zur Verfügung steht.

Beim Einsatz der Berkeley DB als Storage-Backend für die Datenverwaltung in FirstSpirit wird ein "eingebettetes" Storage-Backend verwendet. D. h. der FirstSpirit-Server greift über die eingebettete Berkeley DB direkt auf die Datenbank-Daten im Dateisystem zu.

Die Implementierung der Berkeley DB ist dafür zuständig, die zu speichernden Inhaltsdaten auf effiziente Weise zu organisieren. Berkeley DB arbeitet dabei mit Container-Dateien im Dateisystem, die eine konfigurierbare Größe (per Default 10MB) haben. Die innere Struktur dieser Container-Dateien wird von der Berkeley DB verwaltet, d. h. "kleine" Inhaltsobjekte werden so lange in einer bestehenden Container-Datei abgelegt, bis diese gefüllt ist. Dann wird die nächste Container-Datei (komplett!) angelegt und Stück für Stück gefüllt.

Dadurch wird eine zu starke Fragmentierung des Dateisystems vermieden und gleichzeitig die Dateisystem-Zugriffe minimiert. Da die Berkeley-DB-Implementierung eine gleichzeitige Nutzung der Datenbank-Dateien von unterschiedlichen Prozessen/Systemen unterstützt (Einschränkung: 1 x Read/Write, n x ReadOnly), eignet sich diese Lösung ideal für ein verteiltes System auf Basis eines gemeinsamen SAN-Dateisystems.

### 3.3.1 Backup-Szenarien mit Snapshots

Wenn man FirstSpirit 24/7 betreibt ist es absolut notwendig, Filesystem-Snapshots für den Backup-Prozess machen zu können.

Ansonsten ist die Datenkonsistenz innerhalb der Berkeley-DB (filesystembasiertes Repository von FirstSpirit) nicht gewährleistet, denn ein Backup-Vorgang kann Minuten bis Stunden dauern. Wenn währenddessen Inhalt eingegeben wird, ist die Berkeley-DB nachher inkonsistent.

Dies könnte zwar repariert werden, aber nicht automatisch.

Werden interne Datenquellen in einer externen Datenbank (z.B. Oracle) gehalten, so sollte das dateibasierte Backup möglichst zeitgleich mit (aber nie nach) einem DB-Backup gemacht werden, so dass eine Komplett-Wiederherstellung des Systems konsistent möglich ist. Falls die Oracle-DB so konfiguriert ist, dass sie ständig innerhalb eines Intervalls von z.B. 1 min. wiederherstellbar ist, was bei großen

Systemen üblich ist, ist ein konsistenter Stand zwischen filesystembasiertem Backup und den Daten innerhalb der Oracle wieder herstellbar.

Filesystem-Snapshots können im Prinzip auch im laufenden Betrieb ohne Herunterfahren des FirstSpirit Servers gemacht werden, da aufgrund der Kürze der Zeit, die dieses in Anspruch nimmt, das Risiko einer Dateninkonsistenz (insbesondere im Content-Repository) relativ klein ist und vermutlich (falls sie auftritt) von der Berkeley-DB beim Hochfahren eines über ein Backup wieder hergestellten Servers selbst repariert werden könnte. Optimalerweise sollte jedoch der Server während eines Snapshots heruntergefahren werden.

Das Vorgehen in Kombination mit einem TSM (Tivoli Storage Manager) z.B. wäre demnach wie folgt.

Um Dateninkonsistenzen zu vermeiden, wird optimalerweise zunächst der FirstSpirit-Server heruntergefahren. Dann wird ein Snapshot des Filesystems erstellt und der FirstSpirit-Server wieder gestartet.

Dies minimiert die Downtime des Servers insbesondere da das Herunter- und Hochfahren des FirstSpirit Servers nicht viel Zeit in Anspruch nimmt.

**Wichtig:** Sollte - was in hochverfügbaren 24/7 Umgebungen oft gefordert ist - selbst eine (bei FirstSpirit erwartungsgemäß sehr kurze) Downtime des Servers für die Dauer eines Filesystembackups bzw. -Snapshots nicht akzeptabel sein, so gibt es in FirstSpirit alternativ die Möglichkeit, den Server in einen Backup-Modus zu versetzen.

Dies geschieht über die Methode

```
serverBackupPrepare(long prepareTimeOut,long backupTime);
```

Diese versetzt den Server nach der Zeit **`prepareTimeOut`** in Millisekunden in einen Ruhezustand der Dauer **`backupTime`** und hat gegenüber dem Herunterfahren des Servers folgende Vorteile:

- aktive Benutzer-Sessions gehen nicht verloren,
- aber es werden Client-Requests geblockt, laufen aber nicht auf einen Fehler
- Neuanmeldungen am System sind innerhalb von **`backupTime`** nicht möglich
- laufende Generierungsprozesse werden nicht beendet, aber es werden innerhalb von **`backupTime`** auch keine neuen gestartet

- sollten zu diesem Zeitpunkt Berkeley-DB-Cleaning-Prozesse laufen, so werden diese beendet

Vor Ablauf von backupTimeOut kann der Backup-Modus durch den Aufruf der Methode **`serverBackupDone()`** beendet werden.

**Wichtig:** Der Wert des Parameters **`backupTimeOut`** sollte 90s nicht überschreiten, da die aktiven Clients (auch die Generierungs-Slaves) drei Server-Calls in Abständen von 30s vornehmen, bevor sie ihre Session beenden, weil der Server die Requests nicht annimmt. Diese Zeitspanne sollte aber ausreichend sein, um einen Dateisystem-Snapshot durchzuführen. Wird backupTimeOut > 90s gewählt, so besteht die Gefahr, dass ClientSessions beendet werden bzw. Verbindungen zu Generierungs-Slaves verloren gehen.

Der Filesystem-Snapshot kann demnach dann während des Ruhezustands des Servers gemacht werden.

Der Prozess stellt sich dann wie folgt dar und kann z.B. als Abfolge von drei aufeinanderfolgenden Skript-Aktionen innerhalb eines Server-Auftrags erfolgen:

- Aufruf der Methode **`serverBackupPrepare(…)`** mit geeigneten Parametern **`prepareTimeOut`** und **`backupTime`**
- Erstellen eines Dateisystem-Snapshots
- Aufruf der Methode **`serverBackupDone()`**

Je nach Anzahl der Projekte und Aktivität auf dem Server kann die Zeit variieren, die der Server benötigt, um in den Ruhezustand versetzt zu werden.

Sollte dies innerhalb der vorgegebenen Zeit **`prepareTime`** nicht erfolgt sein, so wirft die Methode eine **`TimeOutException`**.

Es sollte also serverabhängig im Vorfeld getestet werden, welcher ein geeigneter Wert für **`prepareTimeOut`** ist. Der Wert **`backupTime`** sollte hinreichend groß gewählt werden, insbesondere, da nach dem Erstellen des Snapshots der Ruhezustand über den Aufruf der Methode **`serverBackupDone()`** vor Ablauf von **`backupTime`** beendet werden kann.

Eine Beispielkonfiguration für die Steuerung des FileSystem-Snapshots über einen FirstSpirit-Serverauftrag könnte wie folgt aussehen.

Es wird innerhalb der Auftragsverwaltung der FirstSpirit-Server-Eigenschaften ein

Auftrag „Snapshot-Backup“ erstellt, der zeitgesteuert konfiguriert werden kann.

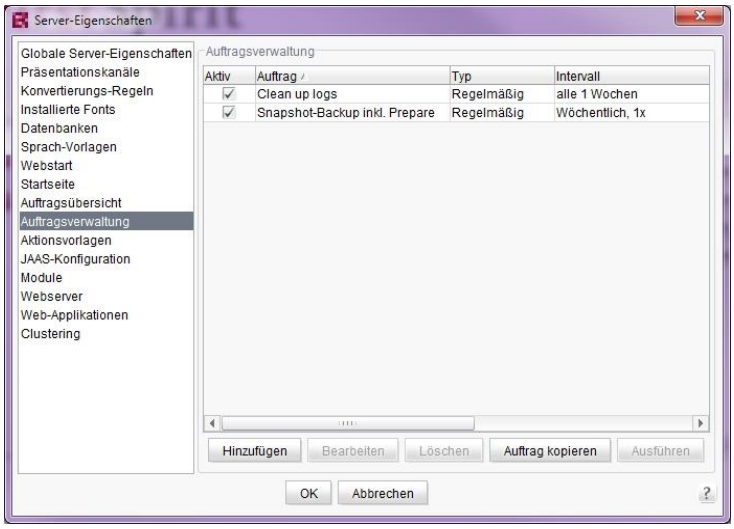


**Abbildung 2 FirstSpirit Server-Auftrag Snapshot-Backup**

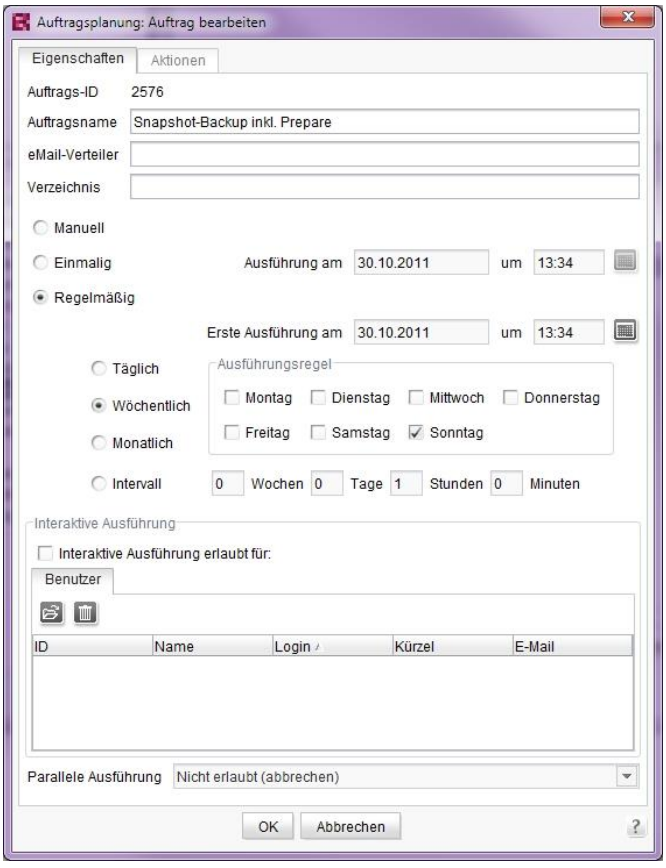


**Abbildung 3 FirstSpirit Server-Auftrag Snapshot-Backup - Zeitsteuerung**

Als Aktion wird in diesen Auftrag eine Skriptaktion konfiguriert, welche durch die

Parameter **`prepareTimeOut`** und **`backupTime`** steuerbar ist.

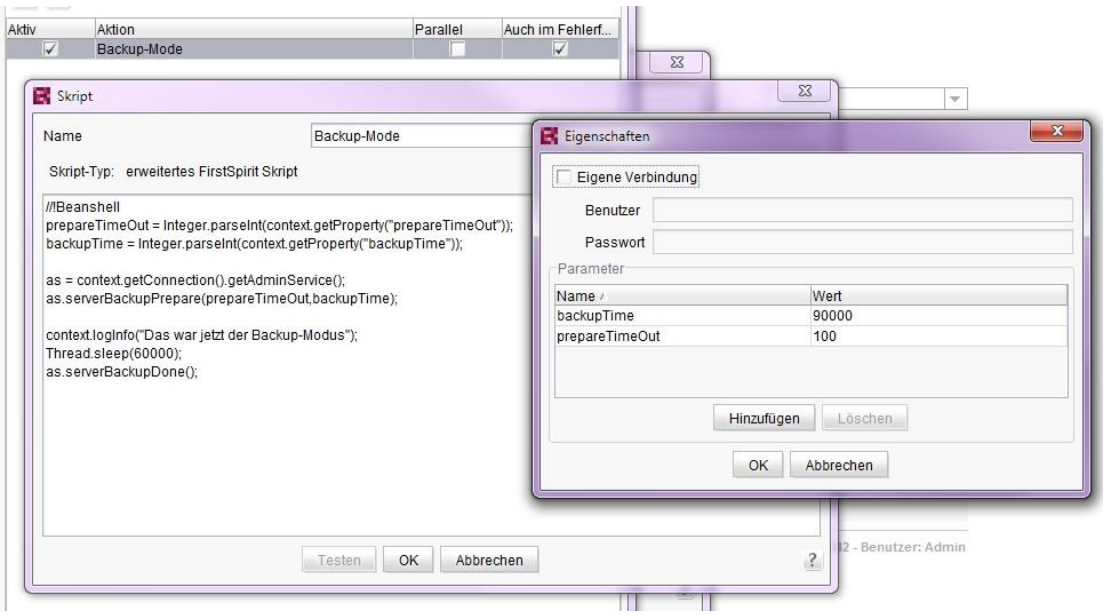


**Abbildung 4 FirstSpirit Server-Auftrag Snapshot-Backup - Aktion Backup Mode**

Wichtig zu bemerken ist, dass der Backup-Modus nicht für eine „klassische“ Filesystemsicherung genutzt werden sollte, da diese erfahrungsgemäß zu lange dauert.

Der TSM öffnet dann den erzeugten Snapshot im Read-Only-Modus und macht darauf das Backup, während auf dem laufenden System weitergearbeitet werden kann.

Dieses Verfahren sollte für die Komplett-Sicherung und -Wiederherstellung des FirstSpirit-Servers genutzt werden, da es die schnellste Möglichkeit der Wiederherstellung des Systems bietet. Die Laufzeit einer Komplett-Wiederherstellung des FirstSpirit-Servers über das Modul Enterprise Backup ist nicht so klar abschätzbar und sollte möglichst zur Wiederherstellung einzelner Projekte innerhalb des FirstSpirit-Servers genutzt werden. Das wird man aber in einem Backup-Konzept noch detaillierter auszuführen haben.

Optimal ist demnach ein snapshotfähiges Filesystem.

Unter SLES 11 steht dies direkt aber nur in Form des btrfs zur Verfügung, welches zur Zeit noch nicht als stabil genug für FirstSpirit angesehen werden muss. Veritas VxFS unter SLES 11 wäre möglich, ist aber ein externes Produkt.

Deshalb wird der Snapshot auf Ebene des Logical Volume angelegt (Volume

Snapshot) und als Dateisystem ein normales ext4 (oder ext3) verwendet.

*Beispiel:*

lvcreate -s -L 1G -n snapshot01 /dev/vg1/firstspirit

Der Snapshot kann dann readonly eingehangen und von der Backup-Software (z.B. dem Tivoli Storage Manager [TSM]) gelesen werden:

mount -o ro /dev/lvm/snapshot01 /backup

Anschließend wird der Snapshot wieder entfernt:

umount /backup

lvremote /dev/vg1/snapshot01

**Wichtig:** Der oder die Preview Server benötigen keinen Zugriff auf das zentrale Filesystem. Sie kommunizieren mit dem FirstSpirit Master Server über den Socket-Port, um Informationen aus dem Repository zu bekommen. Die Preview Server legen pro Webanwendung Dateien im sog. Preview-Cache im lokalen Filesystem ab. Auf dieses muss auch von keinem anderen Ort zugegriffen werden. Es sei denn, man nutzt einen vorgelagerten http-Server für die Preview zur Auslieferung von PHP- oder ASP-Dateien. Diese werden vom Preview-Server im Preview-Cache abgelegt. Zur Auslieferung sendet der Preview-Server einen Request an einen http-Server, der diese dann zurückliefern können muss.

Zu Backup-Strategien in Zusammenhang mit dem FirstSpirit Modul Enterprise Backup, um Server-Konfigurationen bzw. einzelne Projekte zu sichern und zurückzuspielen, s. [3], Kapitel 6.

### 3.3.2 Anbindung eines zentralen Filesystems

Das gängigste und bei den meisten der FirstSpirit-Kunden eingesetzte Szenario ist die Anbindung eines zentralen Filesystems über NFS. Hierbei empfehlen wir aus Performancegründen den NFS-Server softwareseitig auf den FirstSpirit-Master-Server zu installieren, da die schreibenden Zugriffe dann "lokal" erfolgen.

Der Betrieb des FirstSpirit-Master-Servers über NFS (also an einem NAS) ist möglich, wenn die Kombination aus externem NFS-Server und NFS-Client unter SLES 11 technisch einwandfrei ist und sich fehlertolerant gegenüber kurzen Netzausfällen verhält, also keinen I/O-Error beim Herausziehen des Ethernetkabels produziert, sowie eine vergleichbare Leistung bezüglich Durchsatz und Latenzzeit

pro Datenblock bietet wie ein lokales Dateisystem auf einen SAN-Blockdevice.

Mindestvoraussetzung ist der NFS-Mount-Parameter "hard" sowie ein funktionierender rpc.lockd, so dass die Betriebssystemfunktion flock() 100% funktioniert.

Als externer NFS-Server kann ein Linux-basierter NFS-Server nicht empfohlen werden.

Stattdessen z.B. Solaris oder ein Fileserver der Herstellers NetApp, der zusätzlich den Vorteil hätte, ein snapshotfähiges Dateisystem integriert zu haben. Vor Einsatz von NFS für den FirstSpirit-Master-Server sollten Lasttests des Dateisystems in Verbindung mit simulierten Netzausfällen und Lasttest unter FirstSpirit durchgeführt werden. Zu den Lasttests unter FirstSpirit kann e-Spirit Skripte als Unterstützung bereitstellen.

**Wichtig:** Der FirstSpirit-Server verwendet die Unix-Systemfunktion flock() um einige Steuerungsdateien für den exklusiven Zugriff abzusichern. Die Kombination aus Betriebssystem und Dateisystem muss daher flock() anbieten. Auf lokalen Dateisystemen wird diese Funktion von allen von FirstSpirit unterstützten Betriebssystemen bereitgestellt. Auf verteilten Dateisystemen (z. B. GFS2, OCFS, VXFS oder NFS) muss vor der Installation die Dokumentation des jeweiligen Anbieters bezüglich flock() auf dem verwendeten Dateisystem überprüft werden. Unter Solaris und AIX ist z. B. flock() auf NFS immer verfügbar, unter Linux erst seit Kernel 2.6.12 auf NFS.

**Wichtig:** Bei einer großen Anzahl von Projekten auf dem FirstSpirit Server muss ggf. die maximal mögliche Anzahl gleichzeitig geöffneter Filehandles unter dem Benutzerkonto des FirstSpirit-Servers angepasst werden. Da Filehandles auch TCP-Sockets umfassen, die für die Client-Server-Kommunikation notwendig sind, muss der Parameter für die erwartete Anzahl gleichzeitiger Client-Zugriffe ausreichend groß gewählt werden. Jedes FirstSpirit-Projekt belegt bis zu 200 geöffnete Dateien.

### 3.3.3 Cluster-Failover und Architektur

Zur Umschaltung von der Original-Hardware, auf der der FirstSpirit-Master läuft, auf eine Ersatz-Hardware ist ein Cluster-Manager notwendig, der einen Hardware-Ausfall zunächst erkennt, dann die defekte Hardware ausschaltet und alle abhängigen Dienste auf anderen Servern herunterfährt.

Abhängige Dienste für FS sind: Preview-Server, Generierungs-Slaves, NFS-Clients der Generierungs-Slaves.

Sobald die Reserve-Hardware läuft, und der FS-Master-Server wieder hochgefahren ist, werden die abhängigen Dienste wieder eingeschaltet.

Zusätzlich steuert der Cluster-Manager ein sogenanntes Fencing-Device. Das ist eine Hardware, die entweder die Stromversorgung eines Servers trennt, oder nur die SAN oder NAS-Anbindung (Fibre Channel oder Ethernet).

Sie sorgt dafür, dass zu jeder Zeit immer nur genau ein Server auf das Storage-Device zugreifen kann, also niemals, auch nicht bei Fehlbedienungen, Original- und Reserve-System gleichzeitig auf das FirstSpirit-Dateisystem zugreifen können.

Da somit der singuläre Zugriff auf das Dateisystem des Master-Server sichergestellt ist, kann dort ein normales Dateisystem eingesetzt werden, eben ext4 auf Logical Volume Manager (LVM).

Ein Cluster-Manager ist z.B. die Red Hat Cluster Suite for Red Hat Enterprise Linux http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/index.html
Vergleichbares wird von Novell für SLES 11 ebenfalls angeboten, nämlich die SUSE Linux Enterprise High Availability Extension http://www.novell.com/documentation/sle_ha/

Zum Thema Fencing siehe dort das Kapitel 15.0 Storage Protection.

Zur Anbindung der Generierungs-Slaves könnte theoretisch auch ein Cluster-Filesystem statt NFS verwendet werden.

Auch der FirstSpirit-Master würde dann direkt auf diesem Cluster-Filesystem arbeiten.

Das wäre OCFS2 unter SLES 11 oder GFS2 unter Red Hat. Der Betrieb eines Cluster-Filesystems ist aber wesentlich aufwendiger als der Betrieb über NFS und ist für FirstSpirit nicht zwingend notwendig.

Bei Verwendung eines Cluster-Filesystem müsste beispielsweise das Fencing-Device jeden Generierungs-Slave-Server ebenfalls kontrollieren.

**Wichtig:** Fällt ein FirstSpirit Server aus, so erhalten die angemeldeten Benutzer beim nächsten Versuch, eine Verbindung zum Server aufzumachen, eine Meldung, dass die Verbindung zum FirstSpirit Server unterbrochen ist. Die Benutzer haben dann keine Möglichkeit mehr, weiter zu arbeiten, sondern nur noch, den FirstSpirit Client zu schließen. Alle seit der letzten Zwischenspeicherung bis zum Zeitpunkt des Erkennens des Verbindungsverlusts gemachten Änderungen gehen verloren.

## 3.4 Vertikale und horizontale Skalierbarkeit

### 3.4.1 Exkurs Garbage Collection

„Garbage Collection“ bezeichnet einen automatisierten Prozess des Freigebens von Objekte die nicht mehr vom Programm referenziert werden. In anderen Programmiersprachen wie z.B. C oder C++, wird keine Garbage Collection genutzt, d.h. dynamisch allokierter Speicher muss explizit freigegeben werden. In der Programmiersprache Java wird die Freigabe von nicht mehr referenzierten Objekten durch das Garbage Collection System automatisch durchgeführt. Somit steht der freigegebene Speicher im weiteren Programmverlauf wieder zur Verfügung.

#### 3.4.1.1 Vorteile

Die „Last“ der expliziten Freigabe von Speicher wird dem Programmierer abgenommen, somit werden zwei potentielle Fehlerquellen beseitigt,

- zum einen das zu frühe Freigeben von noch referenzierten Objekten,
- zum anderen gar keine Freigabe von Ressourcen.

#### 3.4.1.2 Nachteile

Das automatische Einsammeln von nicht mehr referenzierten Objekten muss zur Laufzeit durchgeführt werden. Das heißt es entsteht ein Overhead (eine gewisse Last), der durch das explizite Anfordern und Freigeben von Speicher nicht anfällt. Dieser anfallende Overhead spielt in „normalen“ Anwendungen keine große Rolle, in Echtzeitsystemen wie FirstSpirit kann dies aber zu einem Problem werden, so das die Anwendung für einige Sekunden oder sogar mehrere Minuten nicht mehr nutzbar ist.

Hierbei spricht man von einer so genannten „Stop the World“ (Full) Garbage Collection (Full-GC).

#### 3.4.1.3 Optimierung der Systemkonfiguration

Viel Energie ist schon darauf verwendet worden, den Prozess der Garbage Collection zu optimieren, und die Bemühungen sind noch nicht am Ende. Grundlegend kommt es darauf an, die Garbage Collection möglichst so durchzuführen, dass die Leistungsfähigkeit der laufenden Prozesse möglichst wenig

behindert wird, und die Full-GC-Zeiten auf einem erträglichen Niveau bleiben — im besten Fall keine Pausen. Und ähnlich wie verschiedene Menschen unterschiedliche Strategien haben, Ordnung zu halten oder mit der Unordnung zu leben, gibt es auch hier verschiedene Ansätze, die nicht von vornherein als ’richtig’ oder ’falsch’ zu bewerten sind. Durch verschiedene JVM-Parameter können die Strategien der Garbage Collection beeinflusst werden und an spezielle Bedürfnisse angepasst werden. Gerade in großen Anwendungen kann die Optimierung der Garbage Collection zu einem beträchtlichen Performance-Gewinn beitragen.

Um optimieren zu können, ist es wichtig, das Verhalten des Garbage Collectors zu kennen, sowie die Server Infrastruktur, Betriebssystemeinstellungen und anfallenden Belastungen der Anwendung im Betrieb.

### 3.4.2 Vertikale Skalierbarkeit (RAM/CPU)

FirstSpirit unterstützt eine "vertikale Skalierung", d.h. durch das Hinzufügen von Ressourcen, wie mehr CPUs oder eine Erhöhung des Hauptspeichers, ist eine Erhöhung der Systemleistung möglich, da in starkem Umfang Verfahren wie Multithreading und Caching eingesetzt werden.

#### 3.4.2.1 RAM

Einem FirstSpirit-Server sollten als Grundvoraussetzung zunächst mindestens 4GB RAM zur Verfügung stehen. Dies gilt auch für die FirstSpirit-Instanzen, die als Generierungs- oder Backup-Slaves aufgesetzt werden (s. Kapitel 3.4.3).

Bei der Hardware-Dimensionierung bezüglich RAM ist es wichtig zu beachten, dass ein Java-Prozess nicht mehr als 10 GByte Heapsize bekommen sollte, sonst können Probleme mit zu langen Laufzeiten von Full-GCs entstehen, die nicht immer ganz zu vermeiden sind. Dies sind allgemeine Erfahrungen aus dem Java-Umfeld.

FirstSpirit selbst läuft auch mit mehr Heapsize, jedoch sollte dann die Java-VM so detailliert konfiguriert werden, dass nie ein Full-GC während der Arbeitszeit auftritt, denn dieser würde dann mit mehr als 10s zu lange dauern.

Die 10GByte sind allerdings keine feste Grenze, denn die Laufzeit des Full-GCs hängt zunächst einmal von der CPU-Leistung ab und steigt ungefähr proportional mit der Heapsize.

Bei viel verfügbarem RAM auf einem System geht es nicht darum, EINEN großen Heap sondern MEHRERE kleine (sprich mehr Java-Prozesse) zu haben (vgl. []).

Inzwischen gibt es alternative VMs wie z.B. Azul, die auch mit großen Heaps arbeiten. Hierfür gibt es allerdings noch keine „Pilot-Installation“ im Bereich FirstSpirit, demnach auch noch keine offizielle Freigabe über das technische Datenblatt.

Das Betriebssystem und Dateisystem-Cache benötigen dann insgesamt noch ca. 4GByte RAM, so dass 16 GByte RAM eigentlich genügen.

Bei aktueller Hardware ist meistens schon eine 32 GByte – Austattung Standard, was also in vielen Fällen zu empfehlen und ausreichend wäre. Durch Hinzunahme von mehr RAM vergrößert sich u.a. der Dateisystem-Cache, der auf keinen Fall schadet.

#### 3.4.2.2 CPU

Anstelle von mehr RAM, das ab einer bestimmten Größe von FirstSpirit nicht mehr genutzt werden kann, besteht die Möglichkeit, mehr CPU-Cores für den Master- oder die Generierungs-Server vorzusehen.

Allgemein kann man sagen, dass eine Projektgenerierung 1,5 CPU-Cores benötigt. Ein Core für den Generierungs-Thread selbst und 0,5 für den Garbage-Collector.

In einigen Projekten wird es sicher so sein, dass ständig im Hintergrund in Minutenabständen generiert wird, wohingegen in anderen Projekten Generierungen zeitgesteuert so koordiniert werden können, dass keine zeitgleichen Generierungen stattfinden.

Hier ist die konkrete Last-Zeit zu berücksichtigen, denn die Pausen zwischen den Generierungen können meist nicht sinnvoll mit anderen Tasks gefüllt werden.

Es müssen also pro FirstSpirit Server-Instanz die konkreten Projektanforderungen und Deployment-Varianten beleuchtet werden.

Ein FirstSpirit Master Server, der gleichzeitig auch Generierungsaufgaben erfüllt (also ohne ausgelagerte Generierungs-Slaves eingesetzt wird), sollte mit minimal 4 CPUs ausgestattet sein.

Kommt Virtualisierung zum Einsatz (z.B. unter VMWare/ESX), so sollte ein FirstSpirit Master Server, der gleichzeitig auch Generierungsaufgaben erfüllt (also ohne ausgelagerte Generierungs-Slaves eingesetzt wird), mindestens 8 CPU in einer VM haben, denn die Standard-Konfiguration von 4 CPU unter VMWare-ESX reicht in der Regel nicht aus.

### 3.4.3 Horizontale Skalierbarkeit

Es ist möglich, einzelne Funktionskomponenten des FirstSpirit-Servers auf unterschiedliche Computersysteme zu verteilen.

Ein Aspekt dieser "horizontalen Skalierbarkeit" ist die Lastverteilung bei der Generierung der FirstSpirit-Inhalte auf die Mitglieder eines Cluster-Verbundes. Die Aufteilung der Generierung erfolgt auf Auftragsebene. Die Generierungsaktionen (innerhalb eines oder mehrerer Aufträge) können auf Clusterknoten verteilt werden. Dabei wird eine Generierungsaktion jeweils vollständig auf einem Clusterknoten abgearbeitet. Weitere, parallel anstehende Generierungsaktionen können auf weitere Clusterknoten verteilt werden. Die Skalierung der Aktionen über die Clusterknoten erfolgt automatisch.

Neben der Vorschauerzeugung ist die Generierung von Inhalten eine der zeitkritischsten Operationen in einer FirstSpirit Umgebung. Dabei treffen hohe Anforderungen an die Rechenleistung und der Wunsch nach einer möglichst geringen Wartezeit zusammen. Zielsetzung der FirstSpirit-Funktionalität "Clustering" ist eine Erhöhung der Performance in Multi-User-Umgebungen. Zu diesem Zweck wird die rechenintensive Generierung vom Master-Server auf einen (oder mehrere) andere Server (Generation-Server) verlagert.

Skalierungsmöglichkeiten bestehen in der Anzahl der Content Generation Server. Bei vielen parallelen Generierungen wäre z.B. die Empfehlung (da auch in diesem Fall die Grenze von 10GB Heapsize eine Rolle spielt), mehrere Generierungs-Slaves einzusetzen.

Der Einsatz von Generierungs-Slaves macht erst Sinn ab einer Generierungszeit von mehr als 5 Minuten, da sich die Zeit, die der Generierungs-Server benötigt, um ein Projekt zu öffnen, sonst nicht rechnet.

## 3.5 FirstSpirit Frontend Server / Webserver Frontend

Der FirstSpirit-Server ist eine klassische Java-Anwendung, die nicht in einem Application-Server betrieben wird. Die benötigten Standard-Serverdienste für den FirstSpirit-Server sind im Standard-Installationsumfang bereits enthalten.

Über das Open-Source-Produkt „Mort Bay Jetty 6.1/7“ werden ein Webserver sowie eine JSP/Servlet-Engine zur Verfügung gestellt, die als Teil des FirstSpirit-Servers in der gleichen VM ablaufen.

Dennoch kann es wünschenswert sein, bestimmte FirstSpirit-Dienste auf

bestehende Infrastruktur zu verlagern. In diesem Fall kann ein Einbinden externer Softwarepakete notwendig werden.

Für den Einsatz der FirstSpirit-Web-Applikationen wird eine Servlet-Engine benötigt, welche die Servlet-API in der Version 2.4 implementiert.

Das bedeutet, dass anstatt des integrierten Jetty die Verwendung eines oder mehrerer externer Servlet-Engines (z.B. Tomcat 6.0, 7.0 oder Websphere) als externer Application-Server möglich und sogar empfohlen ist.

Ab 20 parallelen Benutzern sollte auf den Einsatz des integrierten Jetty verzichtet und auf eine extern ausgelagerte Servlet-Engine bzw. einen Application Server zurückgegriffen werden.

Pro 100 parallelen Benutzern wird bspw. eine Tomcat-Instanz zur Lastverteilung empfohlen.

Bei der Dimensionierung des RAM-Bedarfs (Java-Heap-Size) ist zu beachten, dass jede FirstSpirit-Benutzer-Session im Tomcat ca. 30 MB Java-Heap belegt.

Zur Verwendung von PHP innerhalb der FirstSpirit Preview- und Staging-Applikationen oder zum Loadbalancing für die Application Server sollte ein Apache Webserver in die Infrastruktur als FirstSpirit Webserver Frontend (s.o.) mit einbezogen werden.

## 3.6 Einsatz von Virtualisierungs-Software

Grundsätzlich empfiehlt e-Spirit den Einsatz von FirstSpirit auf dedizierter Hardware.

Vom Einsatz von FirstSpirit in virtualisierten Umgebungen wird nicht prinzipiell abgeraten, denn Virtualisierung wird auch in anderen Projekten - bei Kunden mit hinreichend Erfahrung im Hinblick auf Virtualisierungs-Umgebungen - erfolgreich in Produktivumgebungen eingesetzt.

Allerdings ist beim Einsatz von FirstSpirit im Umfeld von Virtualisierungsprodukten egal welcher Art (VMWare WS/GSX/ESX, Microsoft Virtual PC, XEN usw.) zu beachten, dass durch die zusätzliche Systemkomplexität Probleme, z. B. im Bereich der Performance, nicht grundsätzlich ausgeschlossen werden können. Daher sollte (speziell im Umfeld produktiv eingesetzter Systeme) vor dem Einsatz einer Virtualisierungslösung die gewählte Systemkonfiguration ausgiebig evaluiert werden. Im Optimalfall sollte die Evaluierung in Zusammenarbeit mit e-Spirit erfolgen, speziell wenn das konkrete Virtualisierungsprodukt e-Spirit nicht zur Verfügung steht.

Zu Test- und Evaluierungszwecken ist eine FirstSpirit-Installation auf der Basis von Virtualisierungslösungen möglich, bei ggf. auftretenden Problemen kann e-Spirit aber keine Unterstützung bei der Konfiguration und Optimierung der Virtualisierungsplattform zur Verfügung stellen. Die Reproduktion von Fehlern kann hier problematisch sein.

### 3.7 Weitere performancerelevante Aspekte

Der Einsatz einer lokalen Virenscanner-Installation kann sich negativ auf die Performance und die Systemstabilität auswirken, außerdem ist ein moderater Log-Level von nicht zu unterschätzender Relevanz.

## 4 Referenzierte Dokumente

| Nr. | Dokument | Beschreibung | Ablageort |
|---|---|---|---|
| [1] | | FirstSpirit Admin Dokumentation | |
| [2] | | FirstSpirit technisches Datenblatt | |
| [3] | | FirstSpirit Dokumentation Enterprise Backup | |
| [4] | | FirstSpirit Dokumentation Installation | |

**Tabelle 1 Referenzierte Dokumente**