Server-Setup

Daniel Rohde | Computer- und Medienservice | d.rohde@cms.hu-berlin.de

Eine Zope/Plone-Installation hängt stark von den Anforderungen ab, die diese Installation erfüllen soll. Anhand der an der Humboldt-Universität zu Berlin verwendeten Konfiguration sollen die damit gelösten, aber auch die dadurch neu entstandenen Probleme hier näher betrachtet werden. Einige Aspekte dieser Probleme und Lösungen treffen nicht nur die Server-Administratoren, sondern auch die Entwickler und Autoren von Zope/Plone.

Soll eine Vielzahl von Web-Auftritten auf einer zentralen Zope/Plone-Installation gehostet werden, stößt man recht schnell an die Grenzen dieses Web-Content-Management-Systems.[1, 2] Vor allem die eher schlechte Performance bei vielen parallelen Zugriffen sorgt dafür, dass man die Anfragen und damit die Last auf viele Server verteilen muss. Das allein genügt aber oft nicht. Der nächste Schritt ist es, die häufig abgefragten Seiten, Bilder und Stylesheets zu cachen, statt sie von Zope/Plone ständig neu generieren oder aus der Datenbank holen zu lassen, wobei der Cache von Zope/Plone selbst natürlich schnell überfordert ist. Zudem soll das ganze System auch ausfallsicher sein, was den Hardware-Einsatz nochmals erhöht und letztlich neue Probleme generiert.

Zope/Plone-Administratoren stoßen im laufenden Betrieb auf eine ganze Reihe weiterer Probleme. So ist das System bekannt für seine Speicherlöcher, d. h. ohne regelmäßiges „Durchstarten“ des Zope-Clients läuft der RAM voll, was das System zum „Swappen“ bringt und damit die Server in die Knie zwingt.

Wenn Autoren Seiteninhalte ändern und gleichzeitig lesend darauf zugegriffen wird, fällt schlagartig die Performance, da die schreibenden Zugriffe die lesenden blockieren. Durch häufiges Editieren von Seiten wächst die Datenbank-Datei rapide (jede neue Version vergrößert die Datenbank), was die Zugriffe auf Inhalte ebenfalls ausbremst. Mal abgesehen davon, dass man die Datenbank-Dateien auch noch in endlicher Zeit ins Backup-System – also auf Band – bekommen muss. Ein regelmäßiges Packen – also Entfernen alter Content-Versionen – und ein Verteilen von Inhalten auf mehrere File-Storages ist dabei unausweichlich, wobei beim Packen genügend Platten-Kapazität vorhanden sein muss, da die Datenbank-Datei beim Packen kopiert wird. Manche Dateisysteme bekommen übrigens Performance-Probleme, wenn Datei-Größen die Zwei-Gigabyte-Marke sprengen.

Nicht neu ist auch, dass Software fehlerbehaftet ist. Manchmal bleiben Zope-Clients hängen, weil sie durch Ausnahmefehler in einen undefinierten Zustand geraten. Das macht eine Überwachung zwingend. Je mehr Plone/Zope-Produkte im Einsatz sind, desto mehr kann auch schiefgehen.

Wie man alle diese und die neuen Probleme, die durch die Lösungen entstehen, löst, ohne zu verzweifeln, soll nun anhand der Zope/Plone-Installation erläutert werden, die am Computer- und Medienservice der Humboldt-Universität zu Berlin betrieben wird.

Um zu verstehen, warum wir dieses und kein anderes Setup gewählt haben, muss man die Anforderungen kennen, die an das HU-Setup gestellt wurden:

- Virtuelle Hosts: Es sollen viele Web-Auftritte (Sites) mit eigenen Adressen gehostet werden.
- Isolation: Die einzelnen Sites sollen sich nicht gegenseitig stören, z. B. durch individuelle und/oder fehlerhafte Plone-Produkte.
- Performance: Viele parallele Zugriffe bei gleichzeitig geringer Antwortzeit (<2s) sollten möglich sein.
- Ausfallsicherheit: Bei Ausfall eines Servers soll trotzdem die Erreichbarkeit aller Sites gewährleistet bleiben.
- Backup: Bei versehentlicher oder mutwilliger Löschung von Inhalten sollte die Wiederherstellung sichergestellt sein.

Für den einen oder anderen Zope/Plone-Admin stellt sich hier sicher auch die Frage, was alles nicht gefordert wurde:

- Portal-Features:
  Wir benötigten keine individuellen Benutzer-Sichten auf Plone-Content, d. h. es müssen sich „nur“ Autoren einloggen. Alle andere (anonymen) Benutzer bekommen immer die selbe Ansicht.
- Link-Kontrolle:
  Das Zope/Plone-Feature, das automatisch tote Links erkennt, war für uns nur innerhalb einer Site interessant. Site-übergreifende Link-Konsistenz war nicht zwingend.

Zope/Plone-Server-Setup der HU

Die Abbildung 1 gibt eine grobe Übersicht über Komponenten der Zope/Plone-Installation am Computer- und Medienservice der HU wieder. Als Betriebssysteme setzen wir Debian (Zeo-Client/DB-Server) und Ubuntu (Load-Balancer) Linux ein.

Vor den Zeo-Clients sind zwei Server geschaltet, die im Load-Balancing-Betrieb Anfragen auf die Cluster-IP (webmania.cms.hu-berlin.de) entweder an den lokal laufenden Apache oder auf den Apache der Slave-Node weiterleiten. Dabei kommt der Linux Virtual Server (IPVS) zum Einsatz, der im Round-Robin-Verfahren (Algorithmus: rr) eingehende Pakete auf den Ports 80 und 443 an localhost oder an die Slave-Node routet (Direct Routing: dr). [3]

Der Load-Balancer selbst ist geclustert, d. h. bei Ausfall des Masters übernimmt der Slave die Master-Funktionalität (Load-Balancing, Heartbeat). [4] Sollten der Slave oder einzelne Dienste auf dem Master ausfallen, erkennt der Keepalived-Dämon diesen Ausfall und konfiguriert IPVS entsprechend um.

Parallel zum Load-Balancer läuft ein Apache, der URLs aller eingehenden An-

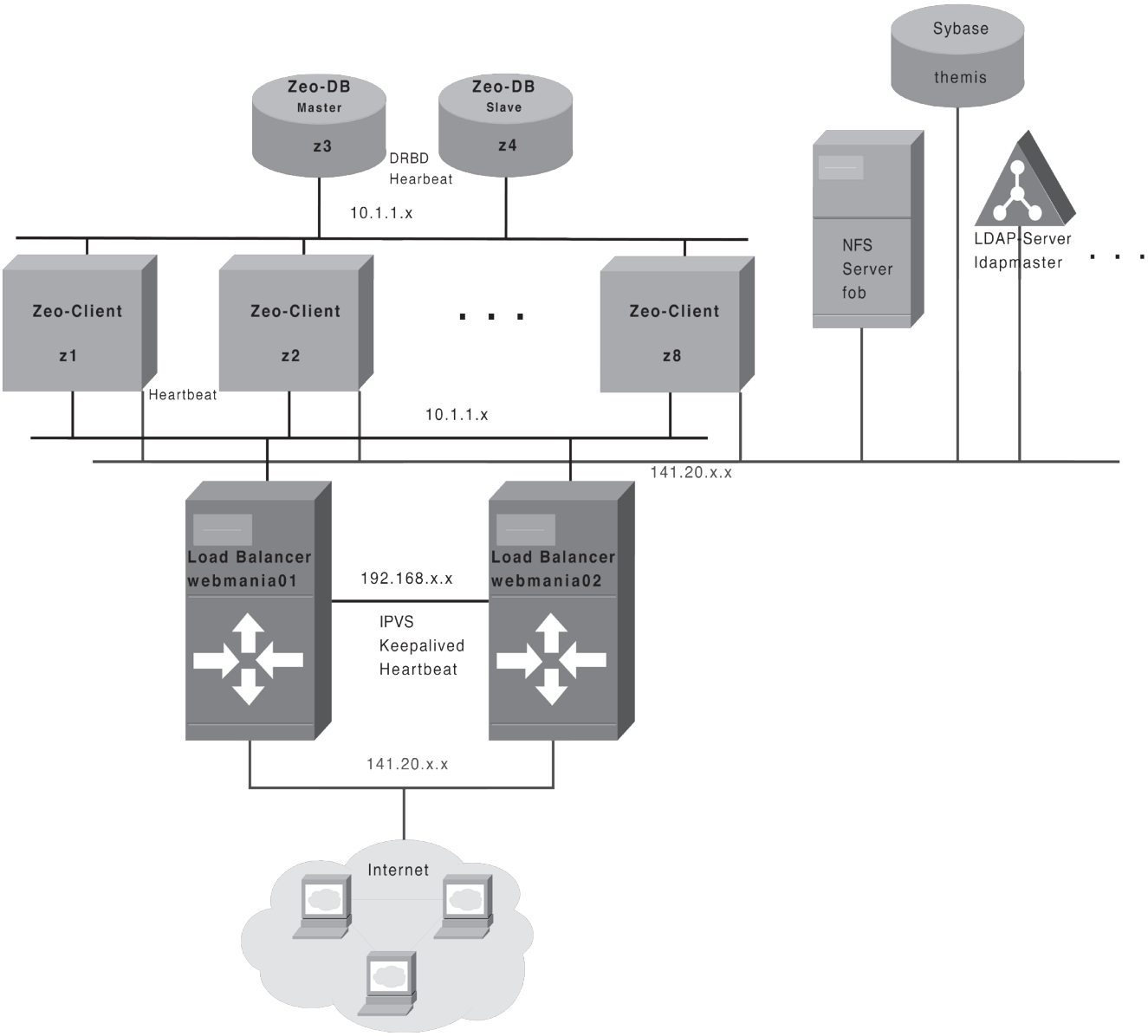

Abb. 1: Überblick

fragen mittels Rewrite-Rules in Virtual Host Monster-URLs umschreibt und dann die Anfragen an den ebenfalls auf dem Load-Balancer laufenden Squid weiterreicht (siehe Abb. 2). Der Apache-Server verwendet ausschließlich Rewrite-Regeln und keinerlei Virtual-Host-Konfiguration, um alle ca. 180 Web-Adressen zu bedienen.

Zusätzlich läuft auf jedem Zeo-Client-Server ein Apache, der über die Standard-Webports 80 (http) und 443 (https) erreichbar ist und wie auf dem Load-Balancer über Squid auf den Port-umschreibenden Apache zugreift, um einzelne Cluster-Knoten testen zu können. Außerdem lässt sich im Notfall die Load-Balancer-Cluster-IP-Adresse auf eine oder mehrere Zeo-Client-Server umlegen (DNS-Round-Robin), was gerade bei Totalausfall des Load-Balancers nützlich ist.

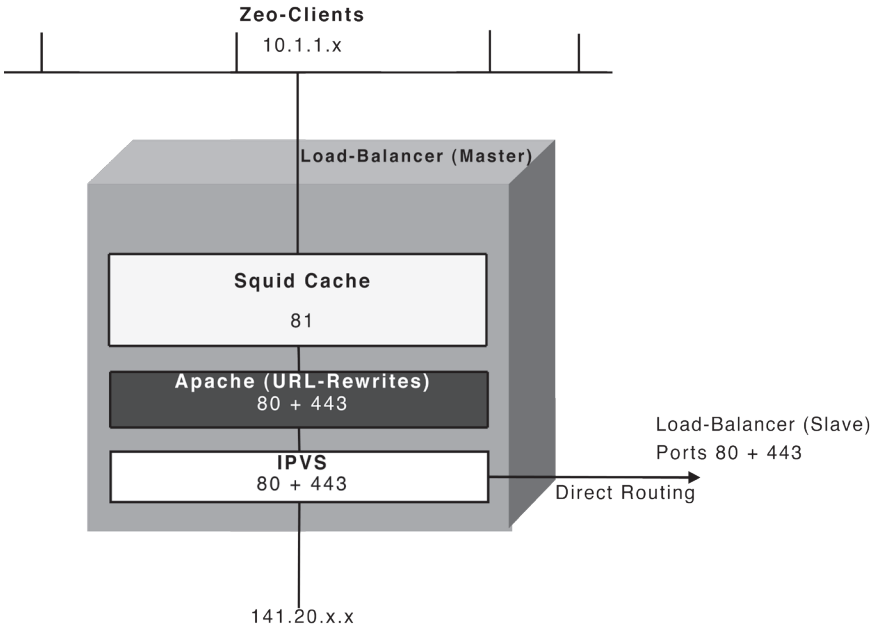

Abb. 2: Load-Balancer

Der Squid selbst leitet Anfragen nicht direkt weiter (always_direct never bzw. never_direct allow all), sondern überreicht diese an sogenannte Cache-Peers. Diese Cache-Peers sind die Apache-Server auf den Zeo-Client-Servern. Die Anfragen werden gewichtet verteilt, da wir unterschiedlich leistungsfähige Hardware bei den Zeo-Client-Servern einsetzen. Der Squid-Cache selbst liegt ausschließlich im RAM (via tmpfs) und belegt pro Cluster-Node circa 30 GB.

Auf den Zeo-Client-Servern läuft jeweils ein Apache, der anhand der Virtual Host Monster-URL den korrekten Port der dazugehörigen Zeo-Client-Instanz bestimmt und an diese die Anfragen weiterreicht (siehe Abb. 3). Auf den Zeo-Client-Servern, die alle identisch konfiguriert sind, laufen für jede Site eigene Instanzen des Zeo-Clients (derzeit ca. 70). Jeder Zeo-Client kommuniziert mit dem Zeo-Datenbank-Server. Außerdem sind die Server paarweise via Heartbeat geclustert (Failover-Konfiguration).

Auf dem Zeo-Datenbankserver läuft für jede Site eine eigene Zeo-Datenbank-Instanz. Die Zeo-Datenbanken liegen auf einem DRB-Device (DRBD=Distributed Replicated Block Device; kurz RAID-1 übers Netz), das sämtliche Schreiboperationen über das interne Netz an den Ausfallserver weiterreicht. Sollte der Datenbank-Server ausfallen, übernimmt automatisch der Ausfall-Server mittels Heartbeat das DRBD-Device und die Cluster-IP-Adresse für den Zeo-Datenbank-Zugriff.

Die gesamte Kommunikation zwischen Load-Balancer, Zeo-Client-Servern und Zeo-Datenbank-Server läuft über ein separates, privates und geswitchtes Netz, das im Moment noch einen „Single Point of Failure“ darstellt, da bei Ausfall des Switches kein alternativer Pfad zwischen den Zeo-Clients und den Zeo-DBs existiert. In den letzten drei Jahren kam es nur zu einem Ausfall und der entstand durch Arbeiten im 19-Zoll-Schrank, in dem sich der Switch befindet – das Stromkabel wurde versehentlich gezogen. Unter Linux könnte man das Problem mit zusätzlichen Netzwerk-Adaptern, separater Switch-Technik und Link-Aggregation (Ethernet Bonding) lösen. [5]

Technische Daten:

- Load-Balancer: 2 x Dell PE R900 (2x1.6GHz Xeon Quad-Core; 48GB RAM; 2x2 76GB SAS-Platten im RAID-1)
- Zeo-DB-Server: 2x Dell PE2850 (2x3.2GHz Xeon; 8GB RAM; 2x148GB SAS im RAID-1; 4x148GB SAS im RAID-5)
- Zeo-Client-Server:
  - 2xDell PE1850 (2x3GHz Xeon + Hyperthreading,: 8GB RAM; 2x73GB SAS im RAID-1)
  - 2xDell PE1850 (2x2.8GHz Xeon Quad-Core; 12GB RAM; 2x146GB SAS im RAID-1)
  - 2xDell PE1950 (2x1.6GHz Xeon Quad-Core; 16GB RAM: 2x146GB SAS im RAID-1)

Unterstützende Systeme:

- NFS-Fileserver für Dateiablage (Binärdateien, wie Bilder, Dokumente usw.) und DB-Sicherung
- LDAP-Server für Benutzerauthentifizierung und Gruppenverwaltung

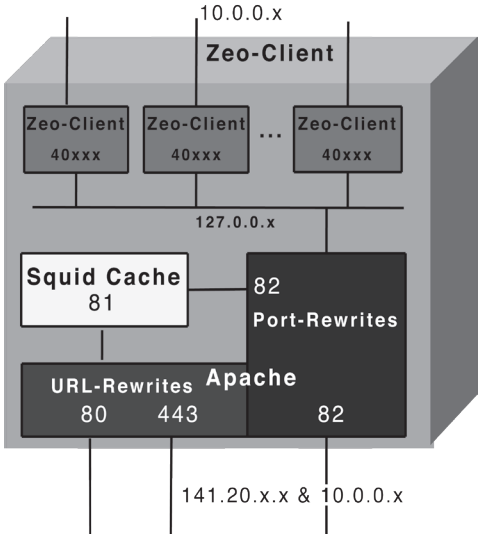

Abb. 3: Zeo-Client

- TSM-Server für Datensicherung und Wiederherstellung (TSM=Tivoli Storage Management)
- Nagios-Server für die Überwachung der Erreichbarkeit und Ressourcen (RAM, Plattenplatz etc.)

Cluster-Probleme

Die Isolation der Web-Sites ist nicht unproblematisch. Da über 70 Instanzen parallel laufen, nehmen sich diese gegenseitig Ressourcen wie Arbeitsspeicher und CPU-Zeit weg. Außerdem verliert man die Site-übergreifende Link-Prüfung, was nur durch zusätzliche Prüfsoftware oder durch Auswertung von Web-Statistiken ausgeglichen werden kann. Tote Links, die in Zope-Instanzen referenzieren, verursachen nicht unerhebliche Performance-Probleme.

Der Einsatz von Load-Balancern und Fail-Over-Clustern schafft neue Probleme und Mehraufwand:

- Es werden redundante Netzwerk-Verbindungen (separate Leitungen und Switch-Technik) zwischen den Cluster-Nodes benötigt.
- Durch den Einsatz von IPVS, Keepalived, DRBD und Heartbeat entsteht zusätzlicher Konfigurationsaufwand. Diese Software-Produkte haben auch Eigenheiten und Bugs.
- Sämtliche Konfigurationsdateien (Apache, Squid, Zeo-Clients) müssen bei Änderungen auf alle Knoten synchronisiert werden.
- Dasselbe gilt für Cron-Jobs zur Überwachung, beispielsweise von Speicherlimits.
- Sämtliche Zope/Plone-Produkte und -Installationen müssen auf allen Zeo-Clients identisch sein.
- Für das Speichern von Binär-Dateien (File System Storage für Bilder, Dokumente etc.) muss ein von allen Zeo-Clients gemeinsam genutztes Verzeichnis zur Verfügung stehen, d. h. es ist eine NFS-Fileserver-Anbindung nötig.
- Die vielen Zeo-Client-Instanzen dürfen nicht alle gleichzeitig gestartet werden, was gerade bei Zope/Plone-Produkt-Änderungen manchmal nötig ist, da sonst der System-Load locker die 40 sprengt.
- Backup-Zeiten sollten nicht zu stark überlappen, um Performance-Einbrüche zu vermeiden.
- Web-Statistiken müssen über alle Apache-Logs generiert und die Statistik-Seiten wiederum auf die Load-Balancer-Nodes verteilt werden.
- Das Cachen von Zope/Plone-Sites generiert sowohl auf der Server- als auch auf der Client-Seite ebenfalls Probleme (siehe Artikel „Caching“ von Frau Lanyi und Herrn Swiatek).
  Wenn beispielsweise Zope bzw. Plone keine korrekten Cache-Informationen im HTTP-Response-Header ausliefert, kann sowohl Squid als auch jeder Web-Browser mehr Cachen als gut ist – was gerade Autoren nervt, da Änderungen unsichtbar bleiben – bzw. gar nichts cachen. Letzteres wirkt sich extrem negativ auf die Performance des Clusters aus.

Sicherheitskonzept, Serverüberwachung und Backup

Nicht nur über die Ausfallsicherheit muss man sich beim Betrieb eines solchen Clusters Gedanken machen. Die Server müssen auch vor Angreifern geschützt werden, die die Systeme „cracken“ wollen, um sie als Dateiablage oder als Ausgangspunkt für Angriffe auf andere Server-Systeme im Internet zu missbrauchen.

Üblicherweise werden bei uns alle Server gehärtet: [6]

- Unnötige Software wird deinstalliert. Alle andere Software wird sicher konfiguriert.
- Offene Ports werden durch sichere Konfigurationen in der Software selbst oder durch den TCP-Wrapper geschützt. Unnötig offene Ports werden geschlossen. Eine Firewall ist daher unnötig.
- Als root-Passwörter werden sehr lange (mindestens 15 Zeichen), mit einer Vielzahl von Sonderzeichen, Zahlen und Klein-/Großbuchstaben versehene Wörter gewählt. Der SSH-Zugang erfolgt üblicherweise mittels hinterlegtem Public-Key.
- Gegebenenfalls werden Virenscanner installiert, die das System regelmäßig überprüfen und deren Signaturen mehrmals täglich aktualisiert werden.
- Betriebssystem-Updates, aber auch Software-Updates werden bei Bedarf eingespielt.

Bis auf die Apache-Installationen, die die Web-Ports 80 und 443 bedienen, sind alle Port-Bindungen von Apache, Squid, Zeo-Client und Zeo-DB auf private und damit nicht geroutete Netze gelegt. Dasselbe gilt für alle Cluster-Produkte wie Heartbeat. Sie sind aus dem HU-Netz und damit aus dem Internet nicht erreichbar. Durch die Wahl der Ports schützt zusätzlich die Firewall der HU vor Zugriffen von außen. Sämtliche Dienste laufen mit eingeschränkten Benutzerrechten, d. h. nicht mit root-Rechten. Da Ports oberhalb der 1024 verwendet werden, werden auch /etc/services-Einträge benötigt, um sicherzustellen, dass nicht die normalen, dynamischen Netzverbindungen die Server-Ports belegen.

Um Ausfälle und Angriffe schnell zu erkennen, ist eine Server-Überwachung lebensnotwendig. Neben Nagios, mit dem die Erreichbarkeit von Diensten, die Speicher-, Platten- und CPU-Auslastung und Prozesse überwacht werden, loggen auch alle Server zentral auf einem Loghost. Die Logdaten werden sofort automatisch auf Auffälligkeiten überprüft und die zweimal am Tag erstellten Log-Statistiken helfen ebenfalls, Probleme und Angriffe zu erkennen. Lüfter, Netzteile, RAM, Betriebs-Temperaturen und andere Hardware-Komponenten werden von einem im Server integrierten Controller überwacht und bei Störungen, Ausfällen usw. automatisch per E-Mail gemeldet.

Da Zeo-Client-Instanzen gerne Arbeitsspeicher auffressen, wird der Speicher-Verbrauch aller Instanzen stündlich überwacht und der übermäßige Konsum durch Neustart gestoppt. Leider geraten die Instanzen auch gern in undefinierte Zustände, beispielsweise durch Ausnahme-Fehler. Ein Überwachungsskript, das jede Instanz kontrolliert, indem die Ausgabe der Start-Seite mittels MD5-Summe überprüft wird, hilft dabei zuverlässig, derartige Probleme zu erkennen. Es müssen aber bei zwei auf anderen Servern laufenden Instanzen andere MD5-Summen zustande kommen (Mehrheits-

entscheid) und der letzte Neustart lang genug her sein, um ein Stopp/Start auszulösen.

Sämtliche Server werden durch das zentrale Backup der HU gesichert. Leider lässt sich die Zeo-DB nicht wie jede Datenbank sichern, d. h. dass man nicht einzelne „Ordner“ oder Content-Objekte aus dem Backup wiederherstellen kann, sondern nur die komplette Datenbank. Man benötigt daher auch immer eine separate Zope-Installation, an die man eine wiederhergestellte Datenbank hängen kann, um dann auf einzelne Objekte zugreifen zu können. Damit ein schneller Zugriff auf die letzten drei gesicherten Zeo-Datenbanken gewährleistet ist, werden diese zusätzlich auf dem NFS-Server abgelegt.

Hilfsskripte

Bei uns kommen eine ganze Reihe von Skripten zum Einsatz, die die Arbeit mit dem Cluster und die Überwachung vereinfachen. Einige hatte ich schon erwähnt, aber der Übersicht halber möchte ich hier mal eine kleine Liste liefern.

Nützliche Skripte bei Änderungen an Konfigurationen und Installationen:

- Erzeugen neuer Zope-Instanzen
  - automatische Bestimmung freier Ports und anschließende Registrierung in /etc/services
  - Anlegen der Zeo-DB
  - Anlegen der Zeo-Client-Dateien und Verteilen auf die Cluster-Nodes
  - Starten der Zeo-DB und Zeo-Client-Instanzen auf allen Nodes
- Prüfen und Verteilen der Apache-Konfiguration auf alle Cluster-Nodes (manuell nach Bedarf)
- Synchronisation von Zope/Plone-Produkten auf alle Nodes (erfolgt manuell nach Änderungen)
- Verteilen neuer Squid-Konfiguration (manuell nach Bedarf)

Skripte zur Überwachung:

- Stündliche und tägliche Überwachung des Speicherlimits aller Instanzen (läuft als cron-Job auf allen Zeo-Client-Servern).
- Überwachung der Erreichbarkeit und Checksummen aller Instanzen (läuft stündlich via cron auf dem Zeo-Datenbank-Server).

Weitere Skripte:

- Skript zum Löschen einzelner URLs aus dem Squid-Cache aller Squid-Installationen im Cluster
- Angepasstes Portal-Squid-Produkt aus dem CacheFu-Paket zum Löschen von Squid-Cache-Inhalten auf allen Squid-Servern
- Generierung von Web-Statistik-Seiten mit AWstats und Verteilen der statischen Statistik-Seiten auf die Load-Balancer-Nodes
- Skript zum Durchstarten von Sybase-Datenbank-Verbindungen in einzelnen Zeo-Client-Instanzen (via cron mehrmals täglich – das ist nötig, da die Sybase-DB-Anbindung in Zope fehlerhaft arbeitet und bei Verbindungsabbrüchen keine neuen Verbindungen zur DB aufgebaut werden.)

Diese Liste von Skripten ist unvollständig und die Quellcodes stellen wir gerne auf Anfrage zur Verfügung.

Statistik

Web-Statistiken können als Argumentationshilfen dienen, wenn es beispielsweise um Browser-Anpassungen von Webseiten geht (Wieso soll sich noch jemand wegen selten benutzter Browser die Mühe von Style-Anpassungen machen? usw.) oder wenn mal neue Hardware beschafft werden muss (viel zu viele Zugriffe ...). Sie helfen, tote Links zu finden oder bei der (Um-)Gestaltung von Einstiegsseiten, da auch Statistiken über Einstiegs- und Ausstiegsseiten („Entry“/“Exit“ Pages) generiert werden. Für unsere Web-Statistiken setzen wir AWstats ein, das recht umfangreiche Informationen liefert. Die Web-Statistiken selbst sind frei zugänglich (Link am Ende des Artikels). [7, 8]

Neben Web-Statistiken lassen wir auch System-Aktivitäten („sar“) mitloggen, um Lastspitzen des Betriebssystems besser analysieren zu können und um Engpässe zu erkennen, z. B. Mangel an CPU-Kapazitäten oder Probleme beim I/O-Durchsatz.

Das zentrale Logging unserer Server erlaubt uns, Statistiken über die System-Meldungen zu generieren. Die Häufigkeiten von Einträgen einzelner Schweregrade (Severities) von Anwendungen (Facilities) helfen uns bei der Problem-Erkennung.

Literatur und Links

[1] *Zope*. http://www.zope.org/, 2008

[2] *Plone*. http://www.plone.org/, 2008

[3] *Linux Virtual Server (IPVS)*. http://www.linuxvirtualserver.org/, 2008

[4] *Linux-HA (heartbeat/DRBD)*. http://www.linux-ha.org/, 2008

[5] *Linux Ethernet Bonding Driver HOWTO*. http://www.kernel.org/, Kernel Source Tree, Documentation/networking/bonding.txt, 2008

[6] Daniel Rohde: *Linux-Sicherheit*. cms-journal 30, http://edoc.hu-berlin.de/browsing/cms-journal/, Juni 2008

[7] *AWstats*. http://www.awstats.org/, 2008

[8] *Web-Statistiken zentraler Webserver der HU*. http://www.hu-berlin.de/allwebstats, http://www.hu-berlin.de/centralwebstats/, 2008