



Technology Scout MediaMosa met meerdere content stores

Versie 1.5
Datum 3 februari 2010

SURFnet/Kennisnet Innovatieprogramma

Inhoudsopgave

INHOUDSOPGAVE	2
SAMENVATTING	4
INLEIDING	6
STORAGETYPEN VOOR MEDIAMOSA	6
<i>Lokale storage</i>	8
<i>Externe storage</i>	8
<i>Externe content</i>	9
<i>Intelligent gebruik</i>	9
STORAGE CONFIGURATIES	10
<i>Samenwerkingsomgeving als eigen storage</i>	10
<i>Instelling met eigen storage</i>	10
<i>Instelling met eigen storage en eigen streamers</i>	11
OPZET PROOF OF CONCEPT	11
<i>Scenario 1: Content uit samenwerkingsomgeving</i>	11
<i>Scenario 2: Instelling met eigen storage</i>	12
<i>Scenario 3: Instelling met eigen storage & streamers</i>	15
STORAGE IN MEDIAMOSA	17
HUIDIGE BEPERKINGEN AAN OPSLAG IN MEDIAMOSA.....	17
UITVOERING SCENARIO 1	18
<i>User stories scenario 1</i>	19
<i>Aanpassingen in MediaMosa ten behoeve van scenario 1</i>	19
UITVOERING SCENARIO 2.....	20
<i>User stories scenario 2</i>	22
<i>Aanpassingen in MediaMosa ten behoeve van scenario 2</i>	23
UITVOERING SCENARIO 3.....	23
<i>User stories van scenario 3</i>	23
<i>Aanpassingen ten behoeve van scenario 3</i>	24
REALISATIE SCENARIO'S	25
UITWERKING SCENARIO 1	25
CMIS	25
<i>Conclusies en aanbevelingen scenario 1</i>	26
UITWERKING SCENARIO 2	26
<i>MediaMosa mountpoints</i>	26

<i>CMIS FUSE implementatie</i>	28
<i>Conclusies en aanbevelingen scenario 2</i>	29
UITWERKING SCENARIO 3	30
<i>Implementatie</i>	30
<i>YouTube API</i>	31
<i>Conclusies en aanbevelingen scenario 3</i>	31
CONCLUSIES EN AANBEVELINGEN	33
STORAGE LOCATIE ALS ABSTRACTIELAAG	33
'INTELLIGENTE' KEUZEN VOOR STORAGE LOCATIES	33
PERFORMANCE ASPECTEN	34

Samenvatting

MediaMosa is open source software waarmee een mediamanagement en distributieplatform kan worden ingericht. Het vormt de basis van tal van succesvolle videodiensten die SURFnet en Stichting Kennisnet aan het onderwijs in Nederland aanbieden.

MediaMosa is software waar eindgebruikers applicaties (EGA) gebruik van kunnen maken voor het opslaan en afspelen van video bestanden. MediaMosa is daarbij voor de eindgebruiker van deze applicaties 'onzichtbaar'.

De huidige opzet van MediaMosa ondersteunt het uploaden van videobestanden naar één storage locatie. Deze technology scout is uitgevoerd om te bezien of het in de toekomst mogelijk zou zijn om video (en ander materiaal) van en naar andere storage locaties te distribueren. Dit maakt het mogelijk om bestanden op basis van bepaalde criteria (zoals kwaliteit of populariteit) op een specifieke storage (die sneller maar duurder of juist trager maar goedkoper is) locatie op te slaan. Door videodata op andere locaties beschikbaar te maken binnen MediaMosa, wordt het ook mogelijk om collecties van andere partners te ontsluiten, zonder dat het materiaal gekopieerd hoeft te worden naar MediaMosa. Voor de bouwers van applicaties op MediaMosa en/of VP-Core en voor de gebruikers hiervan biedt deze uitbreiding mogelijkheden om eenvoudiger eigen collecties te integreren in toepassingen.

Dit rapport is een beschrijving van een proof of concept (PoC) die door Madcap is uitgevoerd. In deze PoC zijn drie scenario's onderzocht op technische (on)mogelijkheden. Alle scenario's leveren werkbare oplossingen op.

In het eerste scenario is het Content Management Interoperability Services (CMIS) protocol toegepast om een koppeling te leggen naar een externe storage locatie. Dit scenario simuleert de situatie waarbij een partnerorganisatie MediaMosa faciliteiten toepast om de functionaliteit en/of capaciteit van haar eigen systeem uit te breiden. Het CMIS protocol blijkt uitstekend geschikt om metadata uit te wisselen, het transparant streamen van externe videobestanden daarentegen is nog niet eenvoudig.

In het tweede scenario wordt het aantal locaties uitgebreid samen met het niveau van integratie. De situatie die in dit scenario wordt nagebootst is die waarbij partners zelf over storage faciliteiten beschikken, Filesystem in Userspace (FUSE) wordt gebruikt als storage abstractie laag. In deze PoC is een FUSE implementatie gerealiseerd met CMIS.

Met FUSE blijkt het goed mogelijk het gewenste niveau van abstractie te bereiken. FUSE zou ook uitstekend geschikt zijn om andere diensten te koppelen (Amazon Simple Storage Service (S3), WebDAV). Bij opschalen van het aantal MediaMosa servers wordt geadviseerd een extra abstractielaag in te bouwen om het beheer in mountpoints en symlinks te vereenvoudigen.

In het derde scenario ten slotte wordt naast de mogelijkheid van externe storage gekeken hoe externe streaming (YouTube) kan worden toegepast in MediaMosa. Deze uitbreiding simuleert de situatie waarbij mediafiles uit andere collecties niet alleen worden opgevraagd via MediaMosa, maar waarbij het afspelen ook uit wordt gevoerd door de systemen van de externe services. YoutubeFS bleek geen bruikbare FUSE implementatie voor dit scenario te zijn, de YouTube API bleek hiervoor uiteindelijk geschikter.

Inleiding

MediaMosa is open source software waarmee een mediamanagement en distributieplatform kan worden ingericht. Het vormt de basis van tal van succesvolle videodiensten die SURFnet en Stichting Kennisnet aan het onderwijs in Nederland aanbieden.

MediaMosa is beschikbaar op www.mediamosa.org. Iedere organisatie die een eigen videoportal wil ontwikkelen, kan MediaMosa gebruiken. Hiermee wordt het voor instellingen of andere organisaties die videodienstverlening willen aanbieden een stuk eenvoudiger om hun videoinfrastructuur in te richten.

MediaMosa is te beschouwen als een software toolkit en biedt tal van (mediamanagement)functionaliteiten die flexibel te gebruiken zijn. Zo kan bijvoorbeeld streaming video worden geïntegreerd met de eigen website, intranet of samenwerkingsomgeving. MediaMosa regelt vervolgens het beheer, de opslag, indexering en het afspelen van videomateriaal.

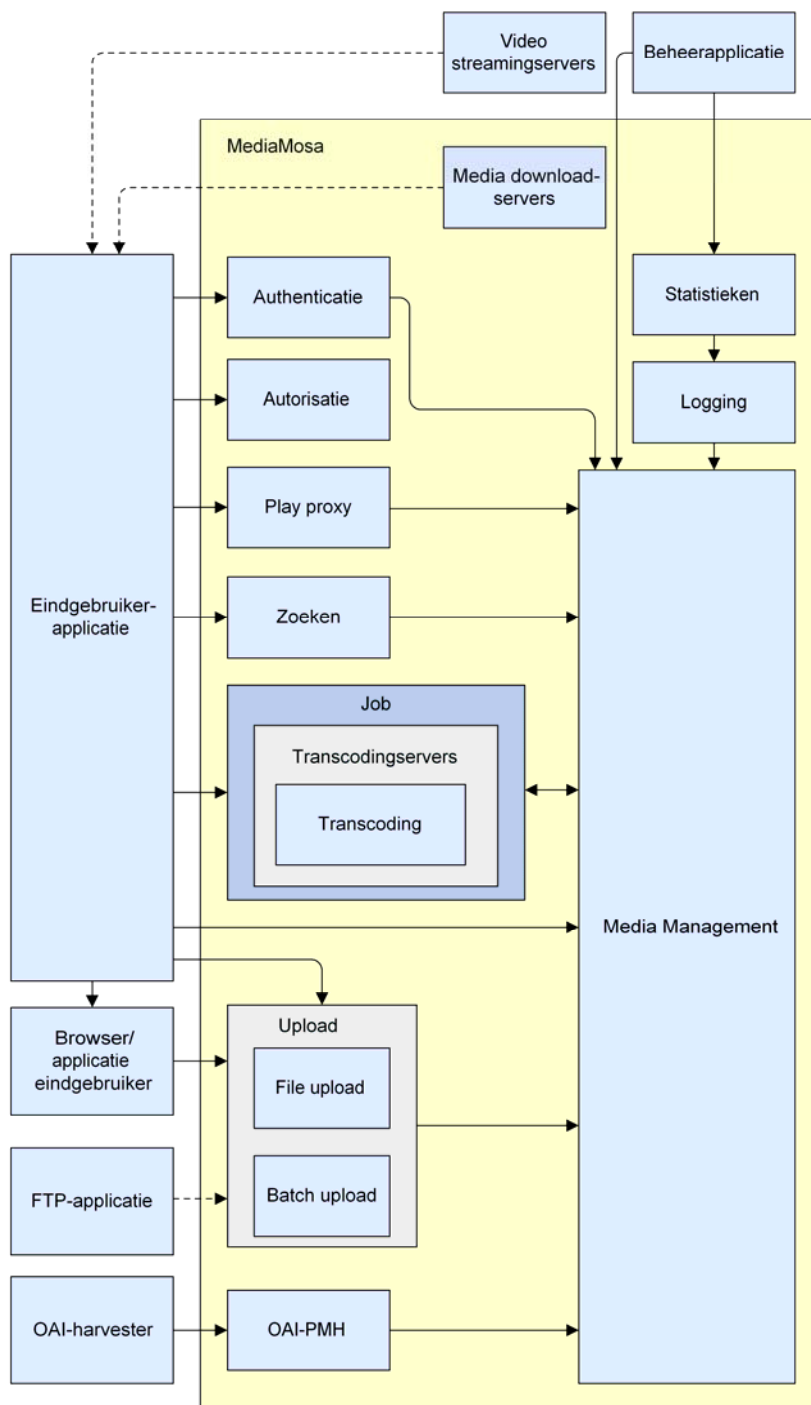
SURFnet en Kennisnet maken voor hun videodiensten zelf ook gebruik van MediaMosa. Diensten zoals SURFmedia en Teleblik zijn aangesloten op het VP-Core videoplatform, waar MediaMosa software wordt gebruikt.

In deze Technology Scout is in het kader van het SURFnet/Kennisnet Innovatieprogramma een onderzoek uitgevoerd naar de mogelijkheid om meerdere stores aan te bieden vanuit MediaMosa. Dit om te bezien of het in de toekomst mogelijk zou zijn om video (en ander materiaal) van en naar andere storage locaties te distribueren, waarmee het ook mogelijk zou worden om videocollecties van onderwijsinstellingen zelf te ontsluiten.

Dit rapport beschrijft de uitgangssituatie, definieert drie scenario's voor het onderzoeken van storage mogelijkheden en geeft een aantal aanbevelingen voor het uitbreiden van de storage mogelijkheden binnen MediaMosa. Op basis hiervan zijn in een proof of concept (PoC) een aantal scenario's getest, waarvan in dit rapport ook verslag is gedaan.

Storagetypen voor MediaMosa

Binnen de MediaMosa architectuur (zie Figuur 1 Componenten MediaMosa backend) speelt de Media Management component in het kader van deze technology scout een centrale rol. Deze component is verantwoordelijk voor het beheer en daarmee voor de opslag van bestanden binnen MediaMosa.



Figuur 1 Componenten MediaMosa backend

In de huidige implementatie van MediaMosa (en VP-Core) wordt alleen de mogelijkheid van lokale (filesysteem gebaseerde) storage ondersteund.

De voor MediaMosa relevante storage typen kunnen globaal in drieën worden gedeeld. De volgende paragrafen behandelen deze vormen van opslag.

Lokale storage

In het lokale storage scenario wordt een 'traditioneel' opslagmedium gebruikt, zoals een disk, DAS, NAS of SAN. Kenmerk is dat die media hetzij rechtstreeks aan de MediaMosa omgeving gekoppeld zijn (disk, DAS), hetzij de opslag voorziening zich in hetzelfde lokale netwerk van de MediaMosa omgeving bevindt (NAS, SAN). Er zijn diverse methodieken beschikbaar zoals SCSI, ATA, NFS, CIFS, iSCSI en FiberChannel om het opslagmedium te koppelen.

Ongeacht de gekozen methodiek is het opslagmedium als een zogenaamd mountpoint beschikbaar. Dit mountpoint is een fysiek koppelpunt wat onderdeel uitmaakt van het filesysteem van een MediaMosa server, waardoor het zich gedraagt als een lokale schijf. De diverse manieren van opslag hebben onderling ieder voor en nadelen¹, echter deze verschillen zijn in het kader van dit onderzoek niet relevant, omdat het hier om het koppelen van meerdere storage systemen aan MediaMosa gaat, en niet om de huidige storage van MediaMosa.

Geen van de oplossingen is in staat rechtstreeks, zonder tussenkomst van een streamer, content aan gebruikers uit te leveren. In dit scenario 'kijken' zowel de streamer als de MediaMosa applicatie naar hetzelfde filesysteem, waardoor beiden wel de beschikking hebben over dezelfde content.

Externe storage

Voor het opslaan hoeft de data niet perse lokaal te staan, maar kan ook gebruik worden gemaakt van 'Storage-as-a-service'. Hierbij wordt storage (capaciteit) aangeboden als dienst, waarbij de gebruiker alleen betaald voor het daadwerkelijk gebruik van de storage. Hoe de aanbieder zijn opslag infrastructuur (technisch) in elkaar heeft gezet is voor de gebruiker niet zichtbaar.

De meest bekende aanbieder is momenteel de Amazon Simple Storage Service (Amazon S3)², maar ook bijvoorbeeld Google, Microsoft en diverse kleinere partijen bieden iets vergelijkbaars. Vanwege het karakter van de aangeboden dienst is de storage voorziening altijd extern ten opzichte van de MediaMosa omgeving.

Kenmerkend voor dit soort diensten is dat ze voor het gebruik van de storage geen traditionele koppelvlakken, zoals in 'Lokale Storage' beschreven, aanbieden. Wel bieden ze vaak FTP, WebDAV of browser gebaseerde interfaces aan. Een relatief nieuwe trend is het aanbieden van een softwarematige 'Application Programmers Interface' (API). Het is daarmee mogelijk middels bijvoorbeeld webservices opdrachten te geven om content te plaatsen, verplaatsen, hernoemen, verwijderen, etc., zie bijvoorbeeld de S3 API³

1 Zie bijvoorbeeld: 'Storage Networking Fundamentals', Marc Farley, Cisco Press, 2005

2 <http://aws.amazon.com/s3/>

3 <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=123>

FTP en WebDAV zijn beide open standaarden en zouden dus relatief eenvoudig in MediaMosa geïmplementeerd kunnen worden. Om gebruik te kunnen maken van een API aanbieder zal MediaMosa normaliter de specifieke API van die leverancier moeten implementeren in de code die de interacties met content afhandelt. Het is niet mogelijk een zogenaamd 'mount point' voor deze vormen van storage aan te maken.

Hoewel het implementeren van een leverancier specifieke API in MediaMosa mogelijk is, is er momenteel geen (open) standaard beschikbaar om te koppelen met dit soort diensten. Dit betekent dat potentieel voor iedere nieuwe storage dienst een API geïmplementeerd zal moeten worden.

De in de op afstand (of wel: 'in de cloud') opgeslagen content benaderen met een streamer is een stuk lastiger dan in het eerder beschreven 'Lokale Storage' scenario, omdat streamers normaliter juist alleen van traditionele opslagmethodieken uit gaan. Om bij de content te komen zal ook de streamer de leveranciersspecifieke API moeten implementeren.

Externe content

Tenslotte zijn er externe diensten die zowel storage als streaming capaciteiten bieden. Het bekendste voorbeeld hiervan is Youtube⁴. Youtube biedt de mogelijkheid middels een API content te plaatsen en te beheren⁵. Daarnaast wordt een API aangeboden om content af te spelen. Naast Youtube bestaan er diverse andere aanbieders van commerciële streaming faciliteiten, zowel browser based als middels 'echte' streamers of CDN (Content Delivery Network) functionaliteit. Ook een instelling kan in principe zelf over streamer faciliteiten beschikken.

MediaMosa kan standaard omgaan met externe links zoals die uit Youtube. Het is echter nog niet standaard mogelijk dergelijke diensten als repository van MediaMosa zelf te gebruiken. Daarnaast zal de afscherming van de content afhankelijk zijn van de afscherming zoals die door de betreffende service wordt geïmplementeerd.

Intelligent gebruik

MediaMosa (1.7.2) gaat op het moment van schrijven uit van de aanwezigheid van een lokale storage oplossing en kan slechts met één fysiek mountpoint omgaan. Binnen MediaMosa is vervolgens middels configuratie aan te geven welk 'type' content op welke plaats (directory) op dat fysieke mountpoint terecht moet komen.

4 <http://www.youtube.com>

5 <http://code.google.com/intl/nl/apis/youtube/overview.html>

Er kunnen dus één of meerdere statische spelregels gedefinieerd worden om het opslaan van de content te regelen. Om zinvol met meerdere storage oplossingen om te gaan zullen er twee zaken aan MediaMosa aangepast moeten worden:

- 1) De mogelijkheid met meer dan één fysiek mountpoint te kunnen werken.
- 2) De mogelijkheid om dynamische spelregels te introduceren. Hierbij kan gedacht worden aan bijvoorbeeld het opslaan van content op een bepaalde plaats op basis van populariteit.

Daarnaast zal MediaMosa, om (2) zinvol te kunnen implementeren, kennis moeten hebben van de gebruikseigenschappen van de content, en deze als basis kunnen gebruiken voor dynamische spelregels.

Storage configuraties

In deze Technology Scout (TS) zijn drie configuraties onderscheiden. De selectie van de configuraties is tot stand gekomen in overleg met de Technisch Product Manager van VP-Core.

Samenwerkingsomgeving als eigen storage

Veel instellingen gebruiken een samenwerkingsomgeving voor het delen van content. Een deel van deze content bestaat uit materiaal dat via MediaMosa efficiënter gedistribueerd kan worden dan middels downloads. Veel samenwerkingsomgevingen hebben standaard een protocol beschikbaar om op file niveau de content te benaderen.

Het uitgangspunt is om content uit een eenvoudige samenwerkingsomgeving te ontsluiten via MediaMosa via een relatief nieuwe open standaard genaamd Content Management Interoperability Services (CMIS⁶). Uit het rapport 'Collaboration Infrastructure'⁷ is gebleken dat dit een van de weinige echt open standaarden is op het gebied van content uitwisseling.

Instelling met eigen storage

Deze configuratie komt overeen met 'Externe storage zonder streaming capaciteiten'. Een instelling heeft wel eigen storage capaciteit, maar wil verder gebruik maken van de centrale faciliteiten van MediaMosa. Belangrijkste businesscase is hierbij vaak de wens een deel van de content 'in huis' te hebben.

6 http://en.wikipedia.org/wiki/Content_Management_Interoperability_Services

7 [http://www.surfnet.nl/Documents/indi-2009-07-020%20\(Rapport%20Collaboration%20Infrastructure\).pdf](http://www.surfnet.nl/Documents/indi-2009-07-020%20(Rapport%20Collaboration%20Infrastructure).pdf)

In deze configuratie dient de MediaMosa omgeving dus toegang te hebben tot de storage omgeving van de instelling. Uitdaging is daarbij dat een methodiek op basis van een traditioneel koppelvlak (bijv. NFS of iSCSI) niet eenvoudig goed te beveiligen is. Ook zijn de protocollen ongeschikt om bij afstanden groter dan ongeveer 50 km voldoende te functioneren.

Eerder onderzoek binnen SURFnet heeft aangetoond dat ook het gebruik van lichtpaden hiervoor geen oplossing bieden. Het lijkt daarom zinvol alleen webgebaseerde API's te gebruiken omdat deze naar verwachting minder te kampen hebben met dit soort problemen.

In de PoC zullen een aantal configuraties worden getest:

- een WebDAV server op enige afstand van de MediaMosa test omgeving
- een CMIS server op enige afstand van de MediaMosa test omgeving
- Amazon S3 storage gekoppeld aan MediaMosa

Instelling met eigen storage en eigen streamers

Instellingen met zowel eigen storage en met eigen streamers zullen zeer beperkt de MediaMosa functionaliteit willen gebruiken. Daarnaast zullen niet alle functionaliteiten van MediaMosa beschikbaar kunnen zijn. Met name afscherming is een probleem, aangezien MediaMosa niet langer kan instaan voor de correcte afscherming van de content, omdat de externe storage locatie daarvoor verantwoordelijk is.

In de PoC zal worden uit gegaan van een situatie waarbij afscherming van de content niet nodig is. Door vervolgens de goed gedocumenteerde API's van Youtube te gebruiken kan voor de PoC de situatie van een instellingen met eigen streamers worden nagebootst.

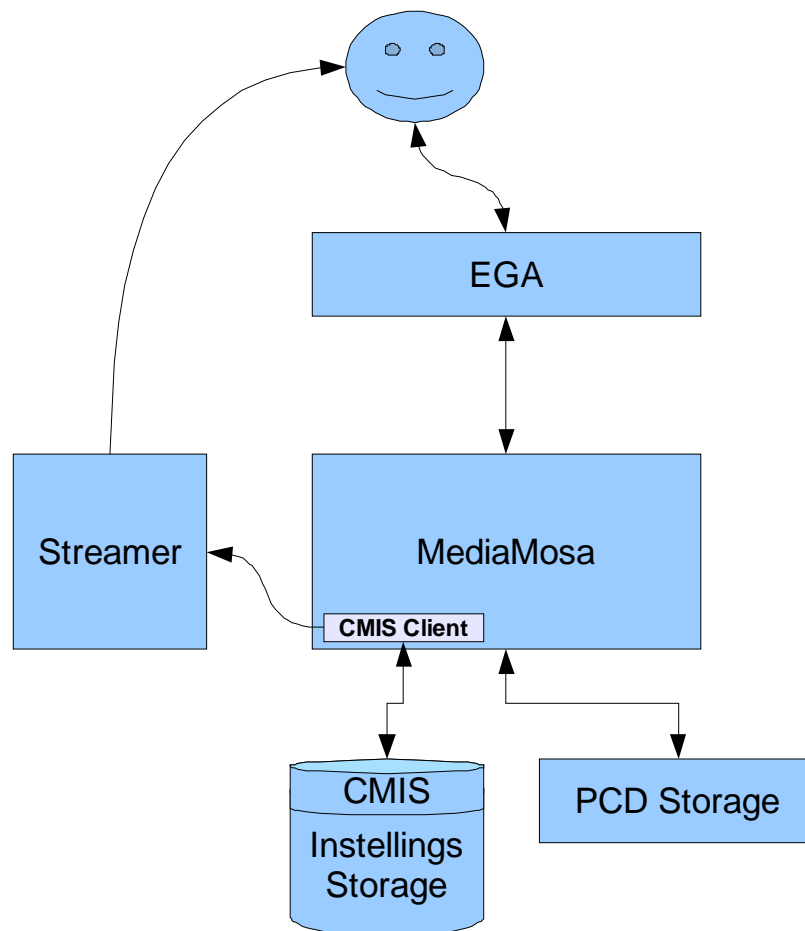
Opzet Proof of Concept

Om gestelde use cases te kunnen realiseren wordt een PoC uitgevoerd met MediaMosa. Hierin wordt een aantal scenario's uitgewerkt, waarbij de in de configuratie beschrijving voorgestelde scenario's worden getest.

Scenario 1: Content uit samenwerkingsomgeving

Als voorbeeldsamenwerkingsomgeving zal de open source omgeving Alfresco (www.alfresco.com) worden gebruikt. Deze omgeving biedt alle faciliteiten van een standaard samenwerkingsomgeving, en heeft daarnaast een CMIS API beschikbaar.

Figuur 2: Externe content op basis van CMIS geeft een overzicht van de beoogde opzet voor scenario 1.



Figuur 2: Externe content op basis van CMIS

Scenario 1 introduceert een CMIS client in MediaMosa. Deze is reeds beschikbaar voor Drupal. Activiteiten in dit scenario:

- Implementatie CMIS client in MediaMosa⁸ (Deze module is nog in ontwikkeling);
- Installatie Alfresco server

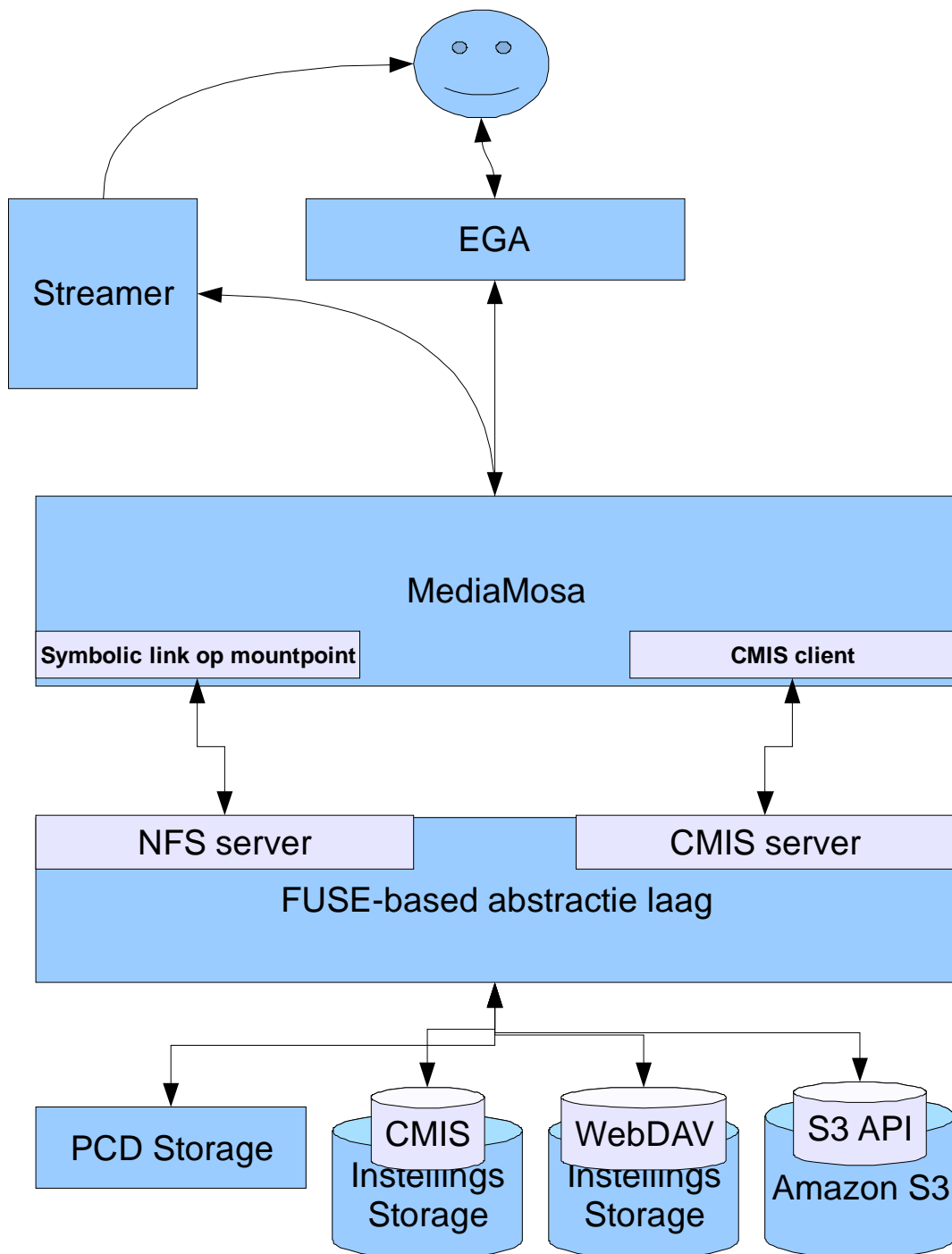
Scenario 2: Instelling met eigen storage

In dit scenario wordt een abstractielaag gecreëerd tussen de verschillende storage oplossingen en MediaMosa. Deze abstractielaag implementeert aan de 'onderzijde' de verschillende storage oplossingen, en levert deze als mountpoints aan MediaMosa. Als basis hiervoor kan FUSE (Filesystems in Userspace⁹) gebruikt worden.

8 http://drupal.org/project/cmیس_alfresco

9 <http://FUSE.sourceforge.net/>

FUSE is een open source ontwikkeling die het mogelijk maakt een programma te implementeren als ware het een filesystem. Op die manier is het dus mogelijk een (webservice) API te implementeren die zich kan gedragen als een traditioneel mountpoint. FUSE behoort reeds enige tijd tot de standaard pakketen die als onderdeel van de Linux kernel worden uitgeleverd en is aantoonbaar betrouwbaar en stabiel. Figuur 3: Externe content op basis van CMIS geeft een overzicht van de beoogde opzet.



Figuur 3: Externe content op basis van CMIS

Met het introduceren van de abstractielaag wordt het aantal wijzigingen aan MediaMosa beperkt, daarnaast wordt het eenvoudiger om gebruik te maken van verschillende soorten storage, omdat er voor FUSE reeds tientallen implementaties van storage systemen gerealiseerd zijn.

Zo zijn WebDAV en S3 implementaties reeds beschikbaar (NB: de betrouwbaarheid van deze implementaties dient wel nader onderzocht te worden).

Voor CMIS is er nog geen FUSE implementatie beschikbaar. Wel zijn er in diverse talen open source CMIS client implementaties beschikbaar, die waarschijnlijk goed bruikbaar zijn als voorbeeld.

Activiteiten in dit scenario:

- wordt MediaMosa aangepast om met meerdere mountpoints om te kunnen gaan;
- worden eenvoudige dynamische spelregels geïntroduceerd om de content over de verschillende mountpoints te kunnen distribueren;
- wordt een abstractielaag gerealiseerd met een standaard Linux gebaseerde server omgeving, of op basis van een open source SAN/NAS oplossing zoals OpenFiler¹⁰ of FreeNas¹¹;
- worden op basis van reeds beschikbare FUSE implementaties WebDAV¹² en S3¹³ geïmplementeerd in de abstractielaag;
- wordt een WebDAV omgeving geïnstalleerd, op basis van Apache WebDAV;
- wordt een FUSE client implementatie voor CMIS ontwikkeld. Bij voorkeur gebeurt dat op basis van PHP zodat deze code later ook door MediaMosa gebruikt kan worden om met EGA's te communiceren. Een PHP interface laag voor FUSE is al beschikbaar¹⁴, net als een php client voor CMIS¹⁵.

Scenario 3: Instelling met eigen storage & streamers

Scenario 3 bouwt voort op het tweede scenario en implementeert een koppeling met Youtube¹⁶. Het plaatsen van de content gebeurt dan ook op dezelfde wijze als in scenario 2. Afspelen van een media file kan (wanneer de file zowel binnen MediaMosa als op Youtube wordt geplaatst) via de eigen MediaMosa streamer of via YouTube. De keuze voor de afspelmethode kan door een EGA bepaald worden en verband houden met afwegingen over kwaliteit, bandbreedte, prijs en/of afscherming.

De Youtube api's ondersteunen zowel opslag als afspeelmogelijkheden. Daarom zullen zowel de FUSE abstractielaag en MediaMosa zelf direct met deze API interactie hebben. De EGA krijgt via MediaMosa een play url aangeboden die niet meer door voor de streamers van MediaMosa, maar voor en door Youtube is gegenereerd.

10 www.openfiler.com

11 www.freenas.org

12 <http://noedler.de/projekte/wdfs/>

13 <http://code.google.com/p/s3fs/>

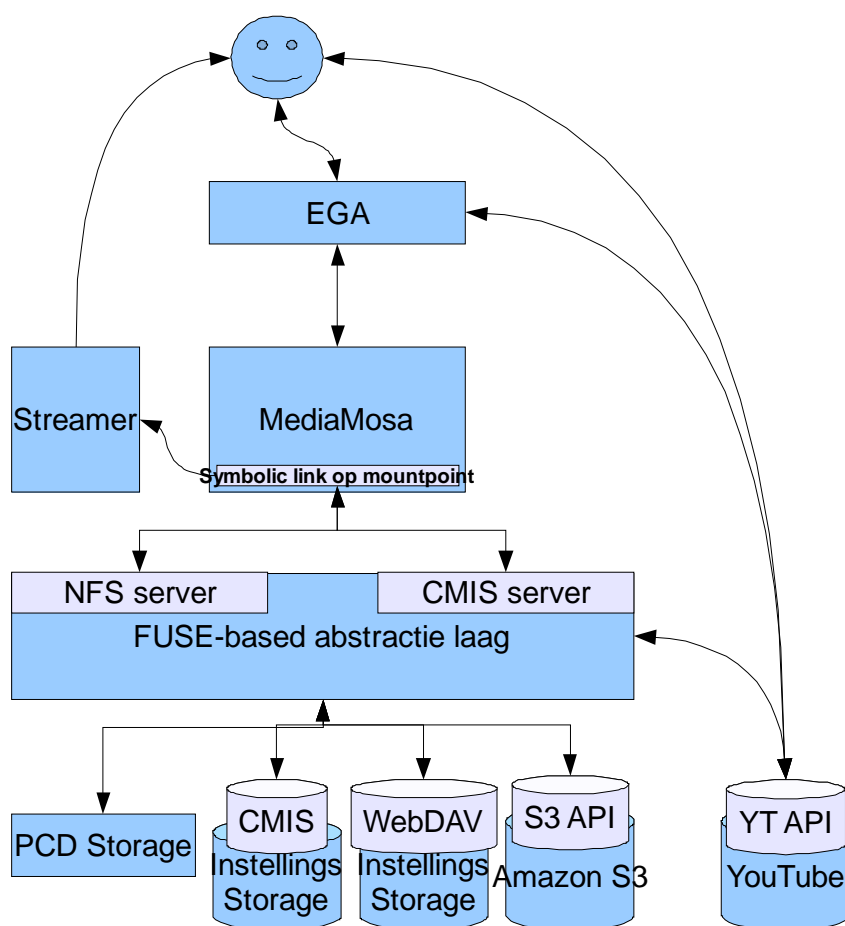
14 <http://pecl.php.net/package/FUSE/>

15 http://drupal.org/project/cmیس_alfresco

16 www.youtube.com

Voor zowel EGA als eindgebruiker schernt MediaMosa de koppeling naar achterliggende storage en afspeelmogelijkheden af. In dit YouTube scenario zal een EGA de standaard upload call aanroepen, het verschil is dat MediaMosa een extra play url zal teruggeven. Eindgebruikers zullen pas bij het afspelen van een filmfragment het verschil opmerken (wanneer de YouTube streamer wordt toegepast).

Figuur 4: MediaMosa met daaraan YouTube als opslag en afspeel omgeving geeft een overzicht van de beoogde uitbreiding van het eerder gerealiseerde.



Figuur 4: MediaMosa met daaraan YouTube als opslag en afspeel omgeving

Om dit scenario te realiseren dient de al aanwezige FUSE abstractie laag te worden uitgebreid met `youtubefs`¹⁷, een FUSE plugin voor Youtube. De storage kan opnieuw worden aangeboden als mountpoint aan MediaMosa. Daarnaast zal MediaMosa enkele API calls moeten implementeren om de play urls van YouTube op te kunnen vragen.

17 <http://code.google.com/p/youtubefs/>

Storage in MediaMosa

Huidige beperkingen aan opslag in MediaMosa

In de huidige opzet van MediaMosa wordt er uitgegaan van één mountpoint naar een filesystem voor het opslaan van mediabestanden. Op dit filesystem wordt een standaard directory structuur aangemaakt om het aantal bestanden binnen een directory te beperken.

In de database van MediaMosa wordt in de 'current_mount_point' tabel informatie over dit mountpoint vastgelegd. De structuur van deze tabel is weergegeven in Tabel 1 current mount point tabel.

Kolom	Type	Key	Omschrijving
current_mount_point	varchar	primary	Het huidige san/nas mountpoint waarnaar nieuwe bestanden worden weggeschreven.

Tabel 1 current mount point tabel

De tabel 'media file' is de representatie van een daadwerkelijke file op het SAN-NAS. In Tabel 2 Media file tabel worden de velden van deze tabel getoond. In deze tabel wordt verwezen naar het Mount point in de eerste tabel.

Kolom	Type	Key	Omschrijving
mediafile_id	varchar	primary	Uniek id van de mediafile
asset_id	varchar	fk naar <i>asset.asset_id</i>	Het asset waartoe deze file behoort
app_id	uint		Het app id van de ega
owner_id	varchar(255)		De eigenaar van de file
group_id	varchar(255)		de groep
is_original_file	enum		Is dit de originele (geuploade) file?
filename	varchar(1000)		Bestandsnaam op het SAN-NAS.
uri	varchar(1000)		URI naar extern bestand
sannas_mount_point	varchar(1000)		Waar is het betreffende sannas gemount
transcode_profile_id	uint		eventueel gekozen profile_id
tool	varchar(10)		eventueel gekozen tool voor het transcoderen
command	varchar(1000)		eventueel gekozen parameters om mee te geven aan de tool
file_extension	char(5)		file extensie van het doel bestand. (container)
testtag	enum		Betreft het testdata?
is_protected	enum		Heeft dit bestand toegangsbeperking?
created	timestamp		Wanneer aangemaakt.
changed	timestamp		Wanneer veranderd.

Tabel 2 Media file tabel

Hoewel deze structuur in principe geschikt is om meerdere mountpoints op te slaan, is dit niet uitgewerkt in de huidige implementatie van MediaMosa. De tabel structuur kent ook geen mogelijkheden voor het ondersteunen van andere koppelingen naar achterliggende storage. Tenzij deze zoals FUSE in combinatie met een symbolic link op logisch file systeem niveau zich niet onderscheiden van een 'normaal' NFS mountpoint.

Uitvoering scenario 1

In dit scenario wordt een MediaMosa instantie gebruikt om de opslag- en afspeelmogelijkheden van een CMS in gebruik bij een partner uit te breiden en te verbeteren. Voor het emuleren van het partner CMS zal in de PoC een Alfresco instantie worden opgezet.

Om niet voor ieder partnersysteem een aparte interface te implementeren, is gekozen om een generieke koppel interface tussen CMS systemen, CMIS, toe te passen. Bovendien is er een Drupal module beschikbaar die een koppeling ondersteunt met Alfresco via CMIS.

Het scenario kan in twee varianten worden uitgevoerd:

- Variant 1: in deze variant worden media bestanden die door MediaMosa afgespeeld gaan worden, verplaatst vanuit het CMS naar de lokale MediaMosa storage.
- Variant 2: hier wordt alleen de streaming functionaliteit van MediaMosa toegepast, die files rechtstreeks afspeelt vanaf het CMS (via CMIS).

De beperking van één mountpoint speelt daarom in dit scenario geen rol, omdat media bestanden die door MediaMosa verwerkt gaan worden, of in de eigen storage worden opgeslagen, of in het CMS.

In het geval van variant 1 moet worden uitgezocht welke CMIS functionaliteit voor het overhalen van bestanden van het CMS naar MediaMosa zal worden toegepast. Er kan een CMIS push vanuit het CMS naar MediaMosa worden toegepast bij het uploaden van een bestand in het CMS of een CMIS synchronisatie geïnitieerd door MediaMosa.

Voor variant 2 moet na worden gegaan in hoeverre streamers geschikt zijn om rechtstreeks CMIS te benaderen om een bestand af te spelen. Als deze functionaliteit (nog) niet wordt ondersteund door de VP-Core streamers, is het niet mogelijk deze variant in de PoC te testen.

Het onderliggende CMS ontvangt in beide scenario's uiteindelijk van MediaMosa een stuk HTML code die gebruikt kan worden om de link naar de streamer via de play proxy te embedden. Een eindgebruiker in het CMS wordt bij het afspelen van een film doorgelinkt naar de MediaMosa play proxy.

Opgemerkt dient te worden dat de tweede variant het MediaMosa principe schendt dat MediaMosa eerst een mediafile analyseert, voordat deze verwerkt wordt. De toepassing hiervan garandeert dat media bestanden binnen MediaMosa een zekere technische kwaliteit bezitten en biedt garanties bij het uitleveren van deze bestanden.

In variant twee moet het onderliggende CMS worden vertrouwd inzake de technische metadata van een bestand. In de PoC situatie zal dit geen probleem zijn, maar voor praktijktoepassingen zal een alternatieve 'analyse op afstand' methode moeten worden gevonden.

In de PoC zal Alfresco¹⁸ worden toegepast als 'partner CMS'. Alfresco is gekozen omdat dit een volwassen CMS is met een degelijke CMIS ondersteuning. Daarbij is dit CMS beschikbaar als open source wat voordelen biedt rond kosten en (community) ondersteuning.

User stories scenario 1

Om scenario 1 volledig uit te kunnen voeren, zullen de volgende user stories worden ingebracht in de PoC:

- gebruiker kan video fragment(en) in CMS (Alfresco) uploaden
- synchronisatie:
 - MediaMosa maakt assets aan voor gevonden documenten in CMS
 - MediaMosa haalt attachment bij documenten op en maakt media file aan
 - Eindgebruiker speelt video af (wordt doorgelinkt naar MediaMosa streamer)
 - MediaMosa streamer speelt via CMIS koppeling video af

Aanpassingen in MediaMosa ten behoeve van scenario 1

De belangrijkste aanpassing aan MediaMosa voor dit scenario is de koppeling via CMIS met Alfresco. Voor deze koppeling zal gebruik worden gemaakt van de Drupal CMIS Alfresco module¹⁹. Deze module wordt ondersteund door zowel de Drupal als de Alfresco community.

18 <http://www.alfresco.com/>

19 http://drupal.org/project/cmیس_alfresco

Een belangrijke beperking van de huidige versie van deze module is dat er (nog) geen ondersteuning is voor binaries. Alleen metadata over objecten en tekstuele inhoud kan worden uitgewisseld tussen de systemen. Voor de PoC is dit onvoldoende.

Daarnaast is de opzet van de Drupal CMIS module dusdanig dat CMIS objecten worden gekoppeld aan Drupal nodes. In MediaMosa wordt geen gebruik gemaakt van nodes. Dit betekent dat ook in het geval van de Alfresco koppeling er in MediaMosa een extra mapping nodig zal zijn. De benodigde code zal indien nuttig als open source beschikbaar worden gemaakt binnen het MediaMosa project.

Deze twee zaken betekenen dat er in de PoC rekening moet worden gehouden dat er voor het koppelen met Alfresco ontwikkelwerk nodig zal zijn.

Uitvoering Scenario 2

In het tweede scenario wordt een middleware laag, FUSE, geïntroduceerd tussen MediaMosa en een viertal achterliggende storage locaties. In MediaMosa kan op twee manieren een koppeling naar FUSE worden gemaakt: via een NFS mountpoint of via CMIS.

NFS mountpoints zijn POSIX²⁰ compliant methode om met FUSE verbinding aan te maken. Voor iedere onderliggende storage wordt een mountpoint aangemaakt. De NFS naar NFS (PCD) verbinding zal zonder problemen horen te werken, omdat dit een POSIX-POSIX verbinding is waarbij de systemen zich ook dicht bij elkaar bevinden (netwerk technisch gezien).

De NFS verbinding met de niet POSIX storage systemen (webdav, cmis, S3) zal onderzocht moeten worden of dit een werkbare methode is. In de systeem configuratie van de PoC bevinden de systemen zich wel dicht bij elkaar, waardoor problemen met response tijden en andere netwerk problemen beperkt zullen zijn. NFS kent geen echte mechanismen of concepten om hier mee om te gaan en zou kunnen constateren dat bestanden of de hele storage voorziening niet meer beschikbaar is.

De CMIS interface gaat uit van een webbased verbinding naar achterliggende storage en kent mechanismen om om te gaan met lange wachttijden en (tijdelijke) uitval van storage.

20 <http://en.wikipedia.org/wiki/POSIX>

In de PoC zal onderzocht gaan worden in hoeverre de NFS/FUSE combinatie geschikt is voor een configuratie met decentrale storage. Door NFS properties en service niveau instellingen en/of caching op FUSE niveau zou een NFS kunnen blijven voldoen.

Voor MediaMosa betekent dit scenario met vier achterliggende storages (PCD, WebDav, CMIS, S3) die via FUSE worden ontsloten en via CMIS of NFS gekoppeld, dat er acht mountpoints benodigd zijn. De huidige opzet met één mountpoint zal moeten worden uitgebreid .

Een ander verschil tussen de koppelmethode (NFS versus CMIS) is de mapping tussen MediaMosa entiteiten en die in de koppellaag.

Bij het gebruiken van een symbolische link met NFS, vindt de mapping op filesysteem niveau plaats. Voor MediaMosa is er (qua mapping) feitelijk geen verschil of een mediafile zich lokaal dan wel op een storage bevindt. Bij het benaderen en manipuleren van het bestand zal MediaMosa identieke commando's op NFS nivo toepassen.

De huidige Drupal CMIS implementatie gebruikt een algemene CMIS module in combinatie met een specifieke module voor een CMIS compliant CMS. Een dergelijke module verzorgt de mapping tussen objecten uit een specifiek CMS en Drupal nodes. Er zijn implementaties voor Alfresco en Knowledge Tree. Voor de mapping tussen de in de PoC gewenste systemen (WebDav, S3) en MediaMosa/Drupal zullen deze modules nog moeten worden gevonden of gebouwd.

In de PoC zal gekeken worden naar hoe de koppeling van de streamer naar achterliggende storage in te richten. De meeste streamers kennen geen functionaliteit om met een CMIS koppeling te werken. De streamers verwachten dat het af te leveren videobestand zich op een eigen lokaal filesysteem bevindt.

In het geval van de NFS methode is dit geen probleem. Net zoals het geval is met MediaMosa, lijkt het voor de streamer alsof de file zich 'logisch' lokaal bevindt. In de PoC kan onderzocht worden in hoeverre dit mechanisme bestand blijkt tegen de eis van hoge throughput die voor streaming geldt.

De rol en mogelijkheden van de FUSE laag zullen bij deze userstory worden meegenomen. FUSE kan optreden als cache en zo ervoor zorgen dat de streamer kan werken met een file op afstand. Aangezien de afstand over de netwerken in de PoC situatie relatief klein is zal een voldoende prestatie te verwachten zijn.

In het geval van de via CMIS gekoppelde storages, zijn er alternatieve scenario's mogelijk. MediaMosa zou ook als een cache kunnen optreden en bestanden naar een lokaal filesysteem kunnen overhalen (weer een reden voor een mounpoint/storage locatie). Daar klaar gezet kan de streamer deze zoals in de huidige opzet uit serveren.

Een tweede alternatief is wanneer het bestand op de storage locatie rechtstreeks benaderbaar is via een uri en de streamer streaming over http ondersteunt. In dit geval kan de streamer rechtstreeks de storage benaderen, zonder gebruik te maken van de MediaMosa/CMIS/FUSE stack.

Een variant hierop is het gebruik maken van de WOWZA feature (URL) om rechtstreeks vanaf een S3 store te kunnen streamen. Dit is uitbreiding op de standaard WOWZA streamer die eventueel kan worden toegepast.

User stories scenario 2

Voor dit scenario zijn er acht 'technische' user stories gedefinieerd voor het koppelen van MediaMosa met de achterliggende storages:

- MediaMosa maakt via FUSE laag koppeling met PCD storage als NFS mountpoint.
- MediaMosa maakt via FUSE laag koppeling met Webdav storage als NFS mountpoint.
- MediaMosa maakt via FUSE laag koppeling met CMIS storage als NFS mountpoint.
- MediaMosa maakt via FUSE laag koppeling met S3 storage als NFS mountpoint.
- MediaMosa maakt via FUSE laag koppeling met PCD storage als CMIS interface.
- MediaMosa maakt via FUSE laag koppeling met Webdav storage als CMIS interface.
- MediaMosa maakt via FUSE laag koppeling met CMIS storage als CMIS interface.
- MediaMosa maakt via FUSE laag koppeling met S3 storage als CMIS interface.

Daarnaast zullen de volgende user stories worden uitgevoerd in de PoC:

- Functioneel beheerder stelt te gebruiken storage locatie in.
- Eindgebruiker load een media file up naar MediaMosa.
- MediaMosa slaat file op bij ingestelde storage locatie via FUSE laag.
- Eindgebruiker speelt media file af .
- MediaMosa speelt file af via FUSE laag.

Aanpassingen in MediaMosa ten behoeve van scenario 2

Door de toepassing van meerdere storage locaties in dit scenario zal MediaMosa aangepast moeten worden. De huidige implementatie waarbij één mountpoint als storage ingang wordt gebruikt is niet bruikbaar. Hiervoor zal de Media Management component moeten worden aangepast.

Zoals hierboven beschreven zal er in MediaMosa op twee manieren gekoppeld worden met de FUSE laag. Voor de NFS methode is het uitbreiden van het aantal mountpoints binnen MediaMosa voldoende om met meerdere storage locaties te kunnen werken.

Voor de koppelingen via CMIS zal een abstractielaag moeten worden geïmplementeerd. Deze laag zal de mapping tussen CMIS objecten en de MediaMosa objecten, zoals assets en mediafiles, verzorgen.

In de PoC zal getest gaan worden in hoeverre de toepassing van de FUSE laag geschikt is voor het afspelen van media bestanden. Door het configureren en testen van de instellingen in deze laag zal onderzocht worden in hoeverre caching voldoende prestaties mogelijk maakt die benodigd zijn voor het afspelen van video's.

Het is de vraag of een rechtstreekse koppeling van de VP-Core streamers via CMIS mogelijk is. Wanneer dit niet het geval is kan het rechtstreeks koppelen van de streamers op de externe storage een alternatief zijn. De WOWZA streamer kent bijvoorbeeld ondersteuning voor het rechtstreeks spelen vanaf S3 storage²¹.

Uitvoering Scenario 3

Scenario 3 is een relatief kleine uitbreiding van scenario 2. Voor MediaMosa betekent dit een extra storage locatie plus een uitbreiding op de afspeel mogelijkheden.

User stories van scenario 3

- MediaMosa maakt via FUSE laag koppeling naar YouTube
- Functioneel beheerder stelt te gebruiken storage locatie in (op YouTube locatie)
- Eindgebruiker load een mediabestand up naar YouTube via MediaMosa
- Eindgebruiker speelt een mediabestand af via YouTube

21 <http://aws.amazon.com/solutions/case-studies/wowza/>

Aanpassingen ten behoeve van scenario 3

Wanneer het toepassen van de FUSE laag zoals bedacht voor scenario 2 succesvol is geïmplementeerd, zal het toevoegen van een extra storage locatie binnen MediaMosa niet veel problemen geven. Wel moet op FUSE niveau deze nieuwe koppeling worden aangemaakt en geconfigureerd.

In MediaMosa is er wel impact op de workflow. De YouTube storage wijkt af van de andere typen omdat deze:

- Naast andere storage moet worden toegepast. Een bestand wordt én op een normale storage én op YouTube geplaatst.
- De storage zijn eigen play url kent. Video's zullen daar worden afgespeeld of via MediaMosa, beide opties moeten aan de EGA worden aangeboden.
- Er extra informatie wordt teruggegeven. YouTube slaat niet alleen het bestand op, maar transcodeert deze en biedt een eigen 'play url' voor het film fragment.

Afhankelijk van de resultaten van scenario 2 kan de koppeling met YouTube via CMIS of via NFS of volgens beide methoden worden toegepast. Wel moet er gecontroleerd worden of de extra informatie die YouTube terug geeft via de koppeling terug wordt gegeven. Indien dit niet het geval is moet de hele keten worden uitgebreid met deze functionaliteit.

Realisatie scenario's

Uitwerking scenario 1

CMIS

De CMIS koppeling, naar een Alfresco CMS van Surfnet, is gerealiseerd met een bestaande drupal module (http://www.drupal.org/project/cmیس_alfresco) wat dankzij het pluggable framework van Drupal eenvoudig is te installeren en configureren. Voor de PoC was het niet nodig om hierin aanpassingen te maken.

Volgens "Content Management Interoperability Services. Part I: Introduction, General Concepts, Data Model, and Services"²² worden 4 primaire objecten gedefinieerd in CMIS: Document Objects, Folder Objects, Relationship Objects, and Policy Objects. Naar MediaMosa kan de volgende mapping gemaakt worden:

- Object-type CMIS::documents MM::Asset+mediafile
- Object-type CMIS::Folders MM::collections

Voor Relationship objects zijn in MediaMosa geen corresponderende object typen, Policy objects komen overeen met autorisatie regels, maar vallen verder buiten bestek van de PoC.

De mediafile zelf wordt uiteindelijk opgehaald met de CMIS call `getContentStream()`. Vanaf het moment dat de stream binnenkomt en op de storage wordt opgeslagen is de file af te spelen. Streaming is mogelijk, echter streaming servers krijgen het einde van het tussenresultaat binnen als een EOF (end of file) van de mediafile en stoppen met afspelen.

Naar aanleiding hiervan is gekozen om in de verdere uitwerking het bestand eerst op de MediaMosa storage op te slaan, alvorens met behulp van de streaming servers de bestanden te kunnen aanbieden. Voor de proof of concept is hiervoor 1 asset opgenomen (zie asset `CtOZvwgrWVtuH0byvRkMGVi7`²³), met een aanpassing in de playproxy module van MediaMosa, dat als het bestand nog niet bestaat, het bestand ophaalt mbv `getContentStream()`, en het op de storage opslaat. Met deze implementatie zal het eerste afspeelverzoek vertraging opleveren (afhankelijk van de grootte van het bestand), maar daarna goed afspelen.

22 "Content Management Interoperability Services. Part I: Introduction, General Concepts, Data Model, and Services" . Version 0.5. 76 pages. Copyright © EMC Corporation, IBM Corporation, Microsoft Corporation. (zie: <http://xml.coverpages.org/cmیس.html>).

23 <http://web.mediamosa.surfnet.nl/poc/asset/CtOZvwgrWVtuH0byvRkMGVi7>

Conclusies en aanbevelingen scenario 1

Enkele conclusies en aanbevelingen:

- De CMIS module van Drupal voldoet;
- Omdat afspelen via de MediaMosa streaming server moet lopen, is een filecopy nodig, dit is een performance issue;
- Momenteel wordt pas begonnen met streamen als de filecopy compleet is. Dit is geen noodzakelijke uitwerking, echter streaming servers stoppen als ze een 'end of file' tegen komen. Wat hier nodig is een streaming adapter voor de streaming servers die het bestand naar de streaming server kan streamen. Nadere studie is hier nodig.
- Bij het eerste ophalen van het bestand is de bepaling van de technische metadata nog niet uitgevoerd; MediaMosa beslist op basis van deze technische metadata welke streaming server het bestand kan serveren. Momenteel is het bepalen van deze metadata nog een 'job' met enige verwerkingstijd. In dit scenario wordt een vaste streaming server gekozen. Zie ook de oplossing in het volgende scenario.
- CMIS ondersteunt stills middels het object "cmis:thumbnail". Aangezien ook MediaMosa de notie stills kent zou dit in de synchronisatie meegenomen kunnen worden (mits de CMIS server dit ook ondersteunt).

Uitwerking scenario 2

MediaMosa mountpoints

Momenteel (versie 1.7) werkt MediaMosa met 1 storage mountpoint. Deze mountpoint heeft de volgende interne structuur:

```
data/1  
data/...  
data/Z  
data/transcode/  
ftp/  
tickets/
```

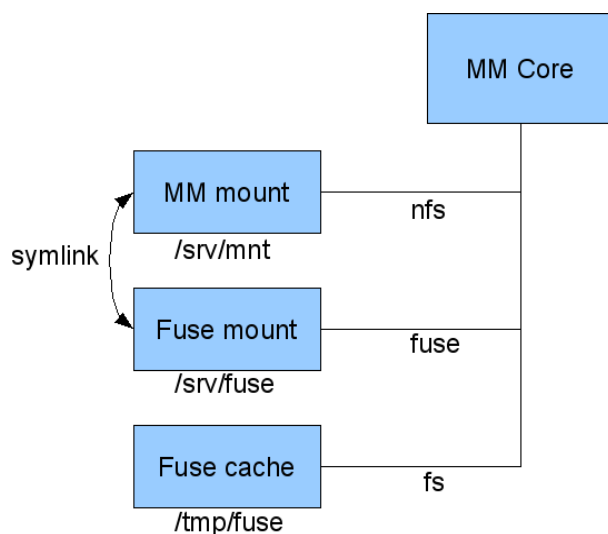
De bestanden worden opgeslagen in de data/ directory, de overige zijn ondersteunende directories die MediaMosa nodig heeft en verwacht. Als deze structuur niet aanwezig is, leveren allerlei processen fouten op.

Een FUSE mount volgt deze structuur uiteraard niet, een aantal mount points naast de bestaande zou er als volgt uit kunnen zien:

```
/mnt/sannas  
/mnt/surfnet_alfresco  
/mnt/youtube
```

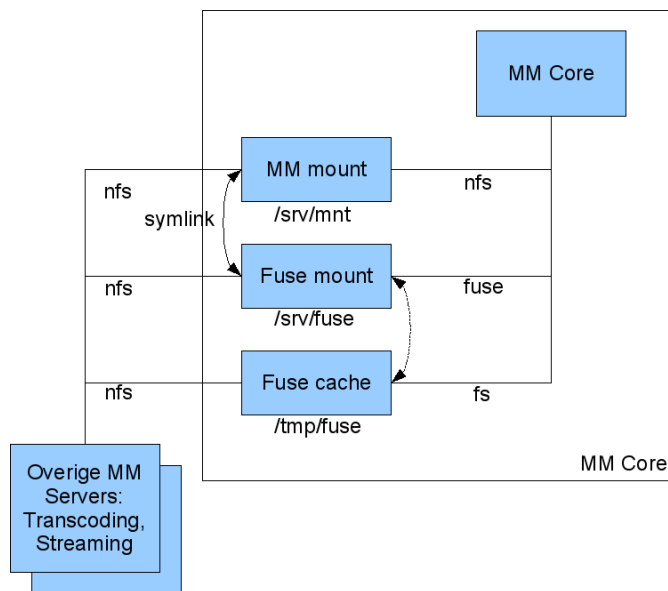
Elke FUSE implementatie organiseert de files onder de FUSEmount op zijn eigen manier. Hieraan gerelateerd moeten alle betrokken servers (transcoding, streamer) ook dezelfde bestanden kunnen raadplegen.

Het is mogelijk om MediaMosa uit te breiden met een mountpoint beheer, zoals besproken in de eerdere hoofdstukken. Een implementatie heeft ook consequenties voor de streaming servers, die in de huidige opzet nog kunnen volstaan met weinig logica. Gezien de omvang van deze implementatie is er in de PoC gekozen voor een symlink constructie van de structuur van MediaMosa met een verwijzing naar de juiste FUSEmountpoints. Dit heeft als voordeel dat de bestaande MediaMosa code hiervoor niet hoeft te worden aangepast (zie figuur 5).



Figuur 5 Mount points met FUSE

De overige servers kunnen dan middels een extra NFS mount naar dezelfde directories worden gemount. Een groot voordeel is dat de externe FUSE-mount niet op alle servers hoeft te worden ingesteld, maar dat ze via een standaard, en eenvoudiger te configureren, NFS mount gekoppeld kunnen worden (zie figuur 6).



Figuur 6 Mount points met FUSE over meerdere servers

Een meer wenselijke oplossing lijkt een aanpassing in de FUSE implementaties, die de MediaMosa directory structuur simuleert, dit zou in een nadere studie onderzocht kunnen worden.

CMIS FUSE implementatie

De gekozen CMIS/FUSE implementatie is een combinatie van een C programma voor de basis file system requests (bv ls), en een PHP programma die een koppeling maakt met de Drupal CMIS module. Opstarten van het CMIS FUSE programma levert een mount point op naast de bestaande SAN/NAS directory:

```
/mnt/sannas
/mnt/surfnet_alfresco
```

Daarnaast is een synchronisatie routine gerealiseerd dat van de CMIS koppeling de assets/mediafiles ophaalt en in MediaMosa plaatst. De gesynchroniseerde assets zijn te zien (en af te spelen) op de demo site van de PoC²⁴ (Alle assets met description die beginnen met "/mnt/FUSE").

24 <http://poc.mediamosa.surfnet.nl/asset>

De synchronisatie is geïmplementeerd in een sync routine, die alle beschikbare assets en mediabestanden van Alfresco ophaalt en importeert in MediaMosa. De synchronisatie is om deze reden traag bij overhalen van grote bestanden, echter daarna heeft het afspelen geen verdere beperkingen meer.

Conclusies en aanbevelingen scenario 2

De koppeling van MediaMosa en Alfresco door middel van CMIS heeft enkele haken en ogen, er valt uitstekend een mapping te maken van CMIS objecten naar representatieve objecten in MediaMosa, ook het delen van informatie stuit op weinig bezwaren. Echter het delen van de videobestanden, vooral bij grote omvang, stuit op beperkingen. Met behulp van CMIS valt wel uitstekend een synchronisatie routine te schrijven, maar een on demand in sync houden van data is vrij complex.

De huidige oplossing heeft een basis caching mechanisme ingebouwd. Bestanden worden bij synchronisatie opgehaald om te worden geanalyseerd. Hierbij worden ze in een temporary directory geplaatst, en worden verwijderd op het moment dat de cache time verloopt. Het lijkt logischer om de bestanden in de MediaMosa mount te plaatsen op moment van synchronisatie, dit kan echter een impact hebben op de gebruikte storage van de standaard mount point. En er is in dit geval geen sprake meer van een cache gezien het permanente karakter van de opslag.

Het synchronisatie proces haalt alle bestanden op om ze te kunnen analyseren. De analyse is een vereiste voor MediaMosa omdat het op basis van de analyse gegevens de juiste streaming server kan serveren. Het is mogelijk om de functie `getContentStream()` met een limit en offset aan te roepen, voor de analyse zijn alleen de eerste paar kilobyte voldoende voor een geslaagde analyse. (Nog niet geïmplementeerd in de Drupal CMIS module).

MediaMosa maakt gebruik van vrij geavanceerde analyse software voor video's. Het verdient aanbeveling om, als dit niet tot resultaten leidt, te kunnen terugvallen op standaard mimetype. In de metadata die door CMIS (`getProperties()`) kan worden aangeleverd kan dan voldoende informatie worden gegeven om een keuze in streaming server te kunnen maken.

In de toekomstige CMIS koppeling moet het ingesteld kunnen worden dat bestanden tijdens de synchronisatie in zijn geheel moeten worden overgehaald (kopieer actie, met synchronisatie mogelijkheid), of dat ze pas bij een afspeelverzoek moeten worden opgehaald (met een extra setting om cache size en expire time in te stellen).

De huidige FUSE implementatie bouwt de connectie met CMIS Alfresco iedere keer opnieuw op. Hier is een optimalisatie gewenst. Voordat de code vrijgeven kan worden aan de Opensource community is het zeer wenselijke om de afhankelijkheid met de Drupal te verwijderen en de CMIS calls in C te implementeren.

Deze Proof-of-Concept haalt de metadata op via de FUSElaag en dus het filesystem (filename, date). Beter is het hiervoor de respectievelijke CMIS functies te gebruiken, om een complete vulling van bijvoorbeeld Dublin Core (Metadata) te kunnen krijgen, en alleen de FUSElaag te gebruiken voor de bestandsfuncties.

De constructies om de verschillende servers met betrekking tot de mounts in sync te houden zijn uiteindelijk vrij gecompliceerd (nfs over FUSE en symlinks waarbij dezelfde configuraties nodig zijn voor alle servers). Het verdient daarom aanbeveling om een extra virtualisatie (Mediamosa vfs) boven op de FUSE layer te bouwen, die deze complexiteit kan reduceren.

Uitwerking scenario 3

Implementatie

YoutubeFS²⁵ is een FUSE implementatie voor YouTube. Na installatie en configuratie zijn de eigen video's en favorieten via een filesystem door te bladeren/bekijken. Een test directory zag er als volgt uit:

```
$ ls -slh favourites/  
115 2009-11-16 14:14 4WD.htm  
115 2009-11-16 14:14 I Can't Get The Printer To Work.htm
```

een cat op een bestand geeft:

```
$ cat 4WD.htm  
<META HTTP-EQUIV="Refresh" CONTENT="1  
URL=http://www.youtube.com/v/xhcVFHZTgew?f=user_favorites&app=yout  
ube_gdata">
```

de URL uit dit .htm bestand bevat een adres naar een flash stream, en niet het video bestand zelf:

```
http://www.youtube.com/v/xhcVFHZTgew?f=user_favorites&app=youtube_g  
data
```

Het videobestand is hiermee niet direct te benaderen.

25 <http://code.google.com/p/youtubefs/>

Concluderend: de YouTube bestanden worden als html bestanden aangeboden, niet als videobestanden zelf; de html bestanden bestaan ook nog eens uit een redirect en zijn dus lastig te verwerken in een eigen website. De conclusie is dat voor de gewenste oplossing YoutubeFS een te beperkte uitwerking van het FUSE protocol is. Het is ook niet eenvoudig om een echte FUSE implementatie voor YouTube te implementeren, omdat de dienst zelf geen download optie aanbied. Een goede implementatie moet dus op basis van de aangeboden streams de FUSE bestandsfuncties implementeren (wel mogelijk gezien de beschikbare youtube download oplossingen).

YouTube API

De use case om met YouTube content verder te kunnen distribueren is dus middels de FUSE implementatie niet te realiseren, om toch de use case verder uit te werken, zonder een FUSE laag gaat het verdere onderzoek van deze PoC verder met YouTube API²⁶, waarmee de YouTubeFS overigens ook gebouwd is.

Synchroniseren met YouTube vindt plaats door een rss feed te raadplegen: <http://gdata.youtube.com/feeds/api/users/foripepe/uploads>. Hiermee kan een pull mechanisme richting MediaMosa gerealiseerd worden. In de PoC is een video-id gekoppeld met een YouTube video, de aanpassing hiervoor is in de playproxy module gerealiseerd²⁷.

De benodigde objectcode wijkt uiteraard ook af van de standaard afspeelobjecten van MediaMosa, in de PoC wordt de html code voor het YouTube object in de playproxy gegenereerd, voor een uiteindelijke oplossing kan deze objectcode in de mediafile tabel worden opgeslagen. Het uploaden van video middels de API was ook eenvoudig te implementeren. Hiermee zou een bestaande asset kunnen worden gekopieerd naar YouTube.

Conclusies en aanbevelingen scenario 3

YoutubeFS biedt voor MediaMosa in dit scenario geen toegevoegde waarde, de API van YouTube echter wel. Hiermee kan relatief eenvoudig (additionele) storage en externe streaming worden gerealiseerd. Ook adviseren we om niet de YoutubeFS beter uit te werken, omdat voor dit scenario de filesystem laag niet essentieel is. Eenmalige push via de API is voldoende, het afspelen is gebaseerd op de externe streaming servers, en dus is een filesystem hiervoor niet nodig.

26 http://code.google.com/apis/youtube/2.0/developers_guide_protocol.html

27 <http://poc.mediamosa.surfnet.nl/asset/A2ZgvnhxTjCZn7uGiQrSFIUC>

Er zijn twee YouTube limieten waar rekening mee gehouden moet worden:

- Bestanden mogen maximaal : "2GB in size and 10 minutes long." zijn.
- Voor API toepassingen is er een limiet van 2000 files per account in totaal.

Deze beperkingen zijn met een betaald account op te heffen.

Gezien de eerdere ervaringen met CMIS zou het in dit kader een interessant idee zijn als er een YouTube CMIS adapter ontwikkeld zou worden. De YouTube API biedt hiervoor een veelbelovende basis.

Conclusies en Aanbevelingen

In dit rapport is een voorstudie gedaan naar de mogelijkheden om MediaMosa uit te breiden met ondersteuning voor meerdere storage locaties.

Storage locatie als abstractielaag

In het eerste scenario is het CMIS protocol toegepast om een koppeling te leggen naar een externe storage locatie. Dit scenario simuleert de situatie waarbij een andere organisatie MediaMosa faciliteiten toepast om de functionaliteit en/of capaciteit van haar eigen systeem uit te breiden. CMIS voldoet in dit scenario voor synchronisatie van Content Management, maar mist functies voor het efficiënt streamen van content.

In het tweede scenario wordt het aantal locaties uitgebreid samen met het niveau van integratie. De situatie die in dit scenario wordt nagebootst is die waarbij organisaties zelf over storage faciliteiten beschikken, FUSE wordt gebruikt als storage abstractie laag. In de PoC is een FUSE implementatie gerealiseerd met CMIS, en is onderzoek gedaan naar de caching mechanisme in Webdav FUSE. De verschillende mountpoints zijn niet eenvoudig te integreren zonder de werking van de streaming servers aan passen. Met behulp van standaardisatie van FUSE naar NFS is een werkbare oplossing gerealiseerd. Het advies is om een extra abstractielaag te implementeren.

In het derde scenario ten slotte wordt naast de mogelijkheid van externe storage gekeken hoe externe streaming (YouTube) kan worden toegepast in MediaMosa. Deze uitbreiding simuleert de situatie waarbij mediafiles uit andere collecties niet alleen worden opgevraagd via MediaMosa, maar waarbij het afspelen ook uit wordt gevoerd door de systemen van de externe services. De YouTube API bleek hier uiteindelijk het meest geschikt.

'Intelligente' keuzen voor storage locaties

De introductie van meerdere stores in MediaMosa impliceert ook een ondersteuning van 'intelligente' keuzes door MediaMosa voor specifieke storage locaties in bepaalde gevallen. De waarde van het gebruik van meerdere storage locaties neemt toe wanneer MediaMosa op basis van configureerbare regels bestanden van een bepaald type of kwaliteit naar langzamere, goedkopere of juist snellere en duurdere storage plaatst. Hoewel binnen deze technology scout dit aspect niet expliciet wordt onderzocht, is het aan te bevelen deze mogelijkheid verder uit te werken.

Een configureerbare workflow zou de intelligente keuze van storage kunnen ondersteunen²⁸. Op basis van metadata, collectiekeuze of een expliciete opgave vanuit een EGA zou een storage locatie (of meerdere) worden gekozen.

Performance aspecten

Een ander aspect van de meervoudige storage is de impact op het streamen. De meeste streamers werken (nog) alleen met bestanden die lokaal beschikbaar zijn. In de PoC bleek dat on demand overhalen van de bestanden problemen oplevert als streaming servers een voortijdig einde tegenkomen. Dit is vooral een probleem bij het skippen van een deel van de video (doorspoelen). Dit is in de PoC opgelost door de bestanden van tevoren integraal over te halen. Een nader onderzoek zou nog gedaan kunnen worden naar een streaming adapter van de files naar de video streaming server.

28 Zie ook rapport Technology Scout Andere Content
(<http://www.surfnetkennisnetproject.nl/resultaten/mediamosa>)