TextGrid
Virtuelle Forschungsumgebung
für die Geisteswissenschaften

Architekturskizze eines produktiven Regelbetriebs

(Meilenstein M 4.5.1)

Version 15. Mai 2015
Arbeitspaket 4.5
Verantwortlicher Partner GWDG

TextGrid
Virtuelle Forschungsumgebung für die Geisteswissenschaften

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Projekt: TextGrid - Vernetzte Forschungsumgebung in den eHumanities
BMBF Förderkennzeichen: 01UG1203A
Laufzeit: Juni 2012 bis Mai 2015

Dokumentstatus: Final
Verfügbarkeit: Öffentlich
Autoren:

Fatih Berber, GWDG
Maximilian Brodhun, SUB Göttingen
Stefan E. Funk, DAASI International GmbH
Peter Gietz, DAASI International GmbH
Ubbo Veentjer, SUB Göttingen
Philipp Wieder, GWDG

Einleitung

TextGrid[1] stellt eine produktive Virtuelle Forschungsumgebung für die textorientierten Geisteswissenschaften und zunehmend auch für andere Disziplinen bereit. Ziel der aktuellen Förderphase ist es, diese Umgebung zu verstetigen und nachhaltige Konzepte und Strukturen zu erarbeiten, die Forscher auch in der Zukunft bei ihrer Arbeit bestmöglich unterstützen.

Eine wichtige Säule bei der Verstetigung von TextGrid ist die Middleware, welche die Basis für die Virtuelle Forschungsumgebung darstellt. Hier ist es notwendig, die existierende Architektur und Umsetzung in regelmäßigen Abständen mit den Wünschen der Fachwissenschaftler, den Nutzerzahlen, Dienstgütedaten und aktuellen technischen Entwicklungen abzugleichen und zu bewerten sowie die Resultate im Hinblick auf den technischen Regelbetrieb zu betrachten. Bei einer solchen Betrachtung zeigen sich in der Regel Lücken, die mittels einer sogenannten Gap Analysis zu erfassen sind. Die Kosten und Zeitaufwände für das Schließen dieser Lücken sind anschließend zu erheben und eine Risikoabschätzung einzelner Lücken bezüglich deren Umsetzung zu machen. Ausgehend davon lässt sich eine Roadmap für die Umsetzung formulieren. Grundsätzlich geht es dabei vornehmlich um eine Verstetigung der bisherigen Funktionalität und nicht um die Entwicklung neuer Funktionalitäten.

Im Report R 4.5.2 „Optimierung der TextGrid Architektur und Infrastruktur“ wurden Überlegungen und Strategien zur Ermöglichung der Hochverfügbarkeit von TextGrid dargestellt. Anknüpfend daran soll der Milestone M 4.5.1 – „Architekturskizze“ eines produktiven Regelbetriebs – den Status Quo der TextGrid Architektur darlegen und die Umsetzung des Hochverfügbarkeitskonzepts erläutern. Er beschreibt die durchgeführten Optimierungsschritte in folgender Struktur: Zunächst wird eine Zusammenfassung des Berichts R 4.5.2 gegeben, aufbauend darauf wird der derzeitige Status der TextGrid-Infrastruktur dargelegt, dabei wird vor allem auf die Optimierungen eingegangen, die notwendig sind, um die Infrastruktur-Komponenten in die Hochverfügbarkeit zu überführen. Im anschließenden Abschnitt wird der aktuelle Entwicklungsstand des Repository dargestellt und eine Roadmap präsentiert, die den Weg zur Vollendung der Hochverfügbarkeit der TextGrid-Repository-Infrastruktur beschreibt.

Zusammenfassung des Berichts R 4.5.2

Der Bericht R 4.5.2 handelt im Wesentlichen von der Planung und Umsetzung eines Hochverfügbarkeitskonzepts für die Infrastruktur des TextGrid Repository. Dabei wurden relevante Komponenten der TextGrid-Infrastruktur identifiziert und mögliche Strategien zur Realisierung erläutert. Mittels Aufwandsabschätzung wurde bis zum Produktivbetrieb eine Roadmap aufgestellt, mit dem Ziel, die Status-Quo-Architektur inkrementell zu einer Hochverfügbarkeits-Architektur auszubauen. In den folgenden

1 http://www.textgrid.de/

Abschnitten wird näher auf den aktuellen Status der einzelnen Komponenten eingegangen.

Status der TextGrid-Infrastruktur

Der Status der TextGrid-Infrastruktur ergibt sich aus der Kombination der Server-Architektur und der einzelnen Services, die in TextGrid verwendet werden. Dieser Abschnitt beschreibt zunächst die grundlegende Server-Architektur und im Folgenden die Services. Dabei wird Bezug auf den aktuellen Status und das zukünftige Umsetzungskonzept genommen.

Server-Architektur

Als Basis für das Hochverfügbarkeitskonzept wurden, wie in R.4.5.2 erwähnt, zwei performante Virtuelle Maschinen (VMs) aus dem ESX-Cluster der GWDG eingerichtet. Für die Hochverfügbarkeit werden diese mittels Puppet-Skripten vollautomatisch konfiguriert.

Für den Test-Betrieb befinden sich zwei weitere VMs in der GWDG-Compute-Cloud im Aufbau, welche schon durch Puppet-Skripte konfiguriert werden. Für einen speziellen Test der RDF-Datenbank BlazeGraph wurde noch eine dritte Test-VM erstellt, die dann auch für den Produktivbetrieb benötigt wird.

Die erste VM mit dem Alias *textgrid-esx1* befindet sich bereits im Produktivbetrieb. Leistungsdaten, die durch das ESX-Cluster automatisch erfasst werden, bestätigen die Spezifikation der VM. Die zweite VM mit dem Alias *textgrid-esx2* befindet sich derzeit noch nicht im Produktivbetrieb, da noch nicht alle produktiven geclustert bzw. in die Hochverfügbarkeit überführt werden konnten.

Zunächst ist die volle Funktion der Produktiv-Maschine auf die Test-VMs aus der Compute-Cloud abzubilden, was dank der Puppet-Konfiguration kein großer Aufwand sein wird. Anschließend kann dann mit geringem Aufwand der Load-Balancer konfiguriert werden. Nach erfolgreichem Einsatz des Load-Balancers kann abschließend der Load-Balancer gegen die Produktiv-Maschinen konfiguriert werden. Der Load Balancer sorgt ebenfalls dafür, dass im Falle eines Serverausfalls alle Anfragen an den jeweiligen zweiten Server geleitet werden.

Hochverfügbarkeitskonzept

Für die Hochverfügbarkeit in TextGrid werden die Kernkomponenten mit ihren unterliegenden Datenbanken betrachtet. Somit sind für die Hochverfügbarkeit die Komponenten TG-crud, TG-auth, TG-noid und TG-search relevant. Damit zusammenhängend müssen die Datenbanken LDAP, ElasticSearch, Berkeley DB und die RDF-Datenbank für eine Hochverfügbarkeitsanwendung konfiguriert werden.

Bei Ausfall einer Datenbank-Instanz soll der vollständige Zugriff des Datenbestands dennoch gewährleistet sein. Das heißt, dass sowohl Lese-, als auch Schreibrechte vorhanden sind.

Das Hochverfügbarkeitskonzept sieht vor, dass zwei parallel laufende Server im ESX-Cluster der GWDG laufen. Diese beiden Server werden mittels der Software Puppet konfiguriert, so dass eine automatische und gleichmäßige Konfiguration durchgeführt werden kann. Durch diese redundante Architektur können nicht nur die Daten und die Services höher verfügbar gemacht werden, sondern auch Rechenlast auf die verschiedenen Server verteilt werden. Für letzteren Aspekt können Load-Balancer hinzugefügt werden, die konfigurierbar die Anfragen entgegen nehmen und die Rechenlast auf die Software und somit auf die unterliegenden Datenbanken verteilen. Durch diese Prozessierung wird lediglich die Datenbankinstanz angesprochen, die verfügbar ist und die Schreibrechte im Cluster besitzt.

Die Komponenten der TextGrid-Middleware sind abhängig von zugrunde liegenden Datenbanken. Ohne geclusterte Instanzen dieser Datenbanken können die Komponenten keine Hochverfügbarkeit erreichen. Aus diesem Grund werden zunächst die Datenbanken auf ihre Parallelisierbarkeit beschrieben und der aktuelle Status Quo dargestellt. Aufbauend auf diese Beschreibung wird die Cluster-Fähigkeit der Middleware-Komponenten beschrieben und deren Status Quo dargestellt.

Datenbanken

ElasticSearch

Das Clustering der ElasticSearch Instanzen ist gut konfigurierbar und wurde bereits zwischen zwei Servern getestet. Dabei kann zwischen verschiedenen Rollen der einzelnen Instanzen unterschieden werden:

- Ein *Workhorse-Knoten* übernimmt keine Master-Funktionen und ist nur für die Speicherung und Bereitstellung der Daten verantwortlich.
- Ein *Coordinator-Knoten* kann Master-Funktionen übernehmen und somit die Datenaufteilung zwischen Workhorse- und Masterknoten koordinieren und andere Knoten zum Master ernennen. Ein Coordinator-Knoten speichert jedoch selbst keine Daten.
- Der *Load Balancer*: Weder Master noch Datenfunktionalitäten sind möglich. Diese Art von Knoten nimmt nur Rechenlast von anderen Knoten.

Durch diese Mechanismen gestaltet sich das Management des Clusters einfach und sehr dynamisch. Bei Zuschaltung eines Knotens muss der Cluster nicht komplett abgeschaltet werden.

Sesame

Der RDF-Tripelstore ist wichtig für die Abfrage der Beziehungen zwischen den verschiedenen TextGrid Objekten. Der bisherige Triplestore Sesame ermöglicht kein Clustering von mehreren Instanzen, aus diesem Grund muss für diese Komponente ein Ersatz gesucht werden.

Auf Grund der High-Availability Komponente von BlazeGraph (ehemals BigData) wurde dieser Triplestore im ersten Ansatz für die grundlegenden Anwendungen in

TextGrid getestet. Die API für diese RDF-Datenbank ist dieselbe wie sie auch in Sesame zur Verfügung gestellt wird, so ist eine Einbindung in die TextGrid-Infrastruktur ohne große Umbauten der genutzten Services problemlos möglich.

Die Tests verliefen erfolgreich, sodass der Test auf die Clustering-Fähigkeit vertieft werden konnte. Für dieses Clustering werden drei Knoten benötigt, die eine Konfiguration über Zookeeper erfordern. Dies ist nicht so einfach zu konfigurieren wie beim Clustering von ElasticSearch. Die Anzahl von drei Knoten ist dabei ein Minimum und bedingt zu den zwei produktiven Servern einen weiteren. Die kompletten Tests sind derzeit noch nicht abgeschlossen.

LDAP

Die LDAP-Datenbank kann problemlos geclustert werden, wobei verschiedene Szenarien möglich sind. Für die TextGrid-Infrastruktur und deren HA-Konzept wurde eine Multi-Master-Replikation konfiguriert, bei der zwei LDAP-Server gleichzeitig als sogenannter Master fungieren und als Lese- wie auch Schreibinstanz genutzt werden können. Alle Daten werden automatisch untereinander repliziert. Dies wurde auf den Testservern textgrid-test1 und textgrid-test2 eingerichtet und getestet. Die Konfiguration wurde in die Puppet-Konfiguration eingebaut und auf dem Testserver deployed. Die Konfiguration jeder weiteren Instanz muss im Wesentlichen nur bezüglich eigener Zertifikate für die TLS-Verbindungen und Hostnamen geändert werden.

Berkeley DB

Die Berkeley Datenbank ist die zugrundliegende Datenbank für den Dienst TG-noid. NOID erzeugt momentan die TextGrid-Identifier (URIs) und verwaltet das interne und externe Locking von TextGrid-Objekten über TG-crud. Da eine Parallelisierung von NOID nicht einfach zu bewerkstelligen ist, werden diese beiden Aufgaben getrennt.

Die Erzeugung von TextGrid-URIs werden weiterhin nun zwei NOID-Instanzen übernehmen, die jeweils mit einen anderen Prefix URIs erzeugen. Anstatt *textgrid:123abc* werden dann z. B. *textgrid:a:123abc* (von NOID auf textgrid-esx1) und *textgrid:b:123abc* (von NOID auf textgrid-esx2) vergeben, was allen beteiligten Services keinerlei Probleme bereiten sollte.

Die Verwaltung des internen und externen Lockings, was bisher durch Schlüssel/Wert-Paare geschah, die über NOID an die URI gebunden waren, wird nun ein Java-internes In-Memory Data Grid[2] – Hazelcast – übernehmen. Dieses wird für alle Locking-Methoden in das URI-Interface von TG-crud implementiert.

Infrastrukturkomponenten

Die Hochverfügbarkeit der Infrastrukturkomponenten bezieht sich auf die Kerndienste von TextGrid (TG-curd, TG-search, TG-auth und TG-noid). Diese Kernkomponenten sind abhängig von den beschriebenen Services, die ebenfalls in die Hochverfügbarkeit integriert werden müssen.

2 http://hazelcast.org/

TG-crud

Die Cluster-Fähigkeit von TG-crud wurde in der Form umgesetzt, dass es lediglich von den unterliegenden Anwendungen abhängig ist. TG-crud bedient die folgenden unterliegenden Dienste und Datenbanken:

- ElasticSearch als Indexdatenbank (direkt)
- BlazeGraph als RDF-Datenbank (direkt)
- LDAP (über TG-auth)
- StorNext als Storage für Daten und Metadaten (momentan über NFS-Mount, später über die DARIAH Storage-API)
- TG-noid (BerkeleyDB über NOID)

Sofern alle diese Dienste/Datenbanken geclustert sind, können zwei TG-cruds parallel laufen und jeweils die verfügbare Datenbank ansprechen.

TG-search

Die Cluster-Fähigkeit von TG-search ist abhängig von der Index-Datenbank Elastic Search und der RDF-Datenbank BlazeGraph. Sind diese beiden Datenbanken in das Hochverfügbarkeitskonzept überführt, ergibt sich daraus auch eine Hochverfügbarkeit von TG-search. Dies bezieht sich im Fall von TG-search auf den Lese-Zugriff der beiden Datenbanken.

TG-auth*

Der Rechteverwaltungsdienst von TextGrid ist ebenfalls von einem unterliegenden Service abhängig. Um eine Hochverfügbarkeit von TG-auth* zu erreichen, muss eine Hochverfügbarkeit des LDAP bestehen. Zwei Instanzen von TG-auth* können problemlos nebeneinander laufen, da der Dienst stateless implementiert wurde. So kann die jeweilige Instanz von TG-auth* mit der jeweiligen Instanz von LDAP kommunizieren.

TG-noid

Damit TG-crud URIs erstellen kann, wird der Kerndienst TG-noid benötigt, welcher wiederum Zugriff auf eine Berkeley Datenbank benötigt. Insofern wäre das Clustering der unterliegenden BerkeleyDB eine wesentliche Anforderung. Wird jedoch das interne und externe Locking von TextGrid-Objekten von NOID unabhängig implementiert, ist ein Clustering der BerkeleyDB unnötig, und es werden zwei voneinander unabhängige NOIDs genutzt, die jeweils eine eigene Datenbank nutzen. Das Locking wird Java-intern über Hazelcast geregelt (siehe auch oben).

Roadmap zur Vollendung

Um die Hochverfügbarkeit in TextGrid zu erreichen, wurden in verschiedenen Treffen technologische Möglichkeiten besprochen und Zeitpläne erstellt. Als Zusammenfassung hat sich folgender Zeitplan ergeben.

Aktion	Aufwand	Datum
Test-Server textgrid-test1 und textgrid-test2 konfigurieren und freischalten		Erledigt
LDAP-Replikation	4 Stunden Multi-Master-Konfiguration, 4 Stunden für Integration in Puppet	Erledigt
ElasticSearch	1 Stunde Datenbank-Konfiguration, 1 Stunde Integration in Puppet	Erledigt
Sesame (Umzug auf BlazeGraph)	5 Tage Installation, Konfiguration und Integration in Puppet	Ende Mai
TG-crud	Abhängig von TG-noid und der StorNext Einbindung	Mitte Mai
TG-noid	Zwei Wochen	Mitte Mai
TG-auth*		Erledigt
StorNext	Unklar	Nach TextGrid in DARIAH*
Konfiguration des Load-Balancers	3 Tage	Ende Mai
Gesamtsystem zusammenführen und testen (funktional und Lasttests)		Ende Mai
HA von Testsystemen auf Produktivsystem deployen		Nach TextGrid in DARIAH

Tabelle 1: Aufwandsabschätzung der einzelnen Aktionen, um das TextGrid Repository in ein Hochverfügbarkeitskonzept zu überführen

Entwicklungsstand des TextGrid-Repository

Das TextGrid Repository wird derzeit einigen Neuerungen unterzogen, die sowohl Funktionsumfang als auch die Usability umfassen. Dies betrifft die TextGrid-Website (textgrid.de) wie auch den Repository-Browser, über den man die Inhalte des TextGrid-Repository komfortabel über das Internet recherchieren und anzeigen kann (textgridrep.de).

Durch die Integration des TextGrid-Repositoriums in ein Liferay-Portal wird eine einheitliche Wartbarkeit der Web-Anwendungen ermöglicht. Sowohl die TextGrid-Website, als auch das TextGrid-Repository wird dann über dieses Portal präsentiert. Als neue Funktionen sind dabei beispielsweise Facetten hinzugekommen, um eine komfortable Suche und ein angenehmes Browsen durch die Daten des TG-rep zu ermöglichen. Diese müssen zunächst in TG-search implementiert werden, unterliegende Datenbanken wie z. B. ElasticSearch unterstützen Funktionen wie Facettierung Out-of-the-Box. Weitere Aspekte der Usability (z.B. intuitivere Website-

* Diese Funktionalität wird nach Ende der Projektlaufzeit, im Kontext der Zusammenlegung mit DARIAH, realisiert.

Steueurng) wurden durch die Task Force „TextGrid-rep Usability“ ausgearbeitet, welche im Zuge der Neugestaltung umgesetzt werden.

Roadmap zur Vollendung

Aktion	**Datum**
Aufsetzen eines Liferay-Servers	Erledigt
Erstellen eines Liferay- Themes	Ende Mai
Erstellen des Web-Designs	Mitte Mai
Produktivschaltung der neuen Repository Webanwendung	Ende Mai
Weiterentwicklung des OAI-Services (Set- und ResumptionTokens)	Ende Mai
Facettenfunktion in TG-search	Mitte Mai
Dokumentation und Hilfetexte	Mitte Mai

Tabelle 2: Aufwandsabschätzung für Neuerungen innerhalb des TextGrid Repository

Architekturskizze

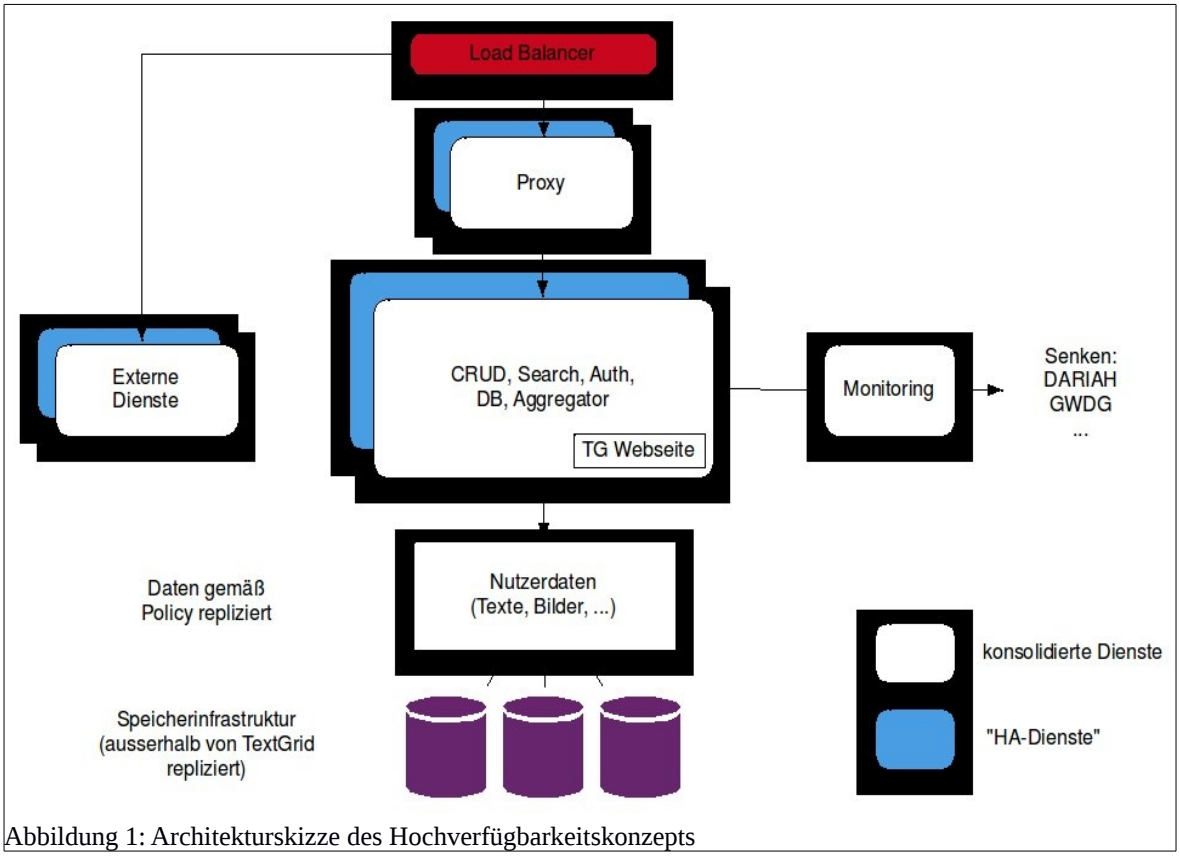

Abbildung 1: Architekturskizze des Hochverfügbarkeitskonzepts

Abbildung 1 zeigt die geplante und in diesem Dokument beschriebene Architektur von TextGrid unter Verwendung der Hochverfügbarkeits-Elemente.

In einem ersten Schritt wurden bereits die Server umgestellt, sodass Server aus dem ESX Bereich der GWDG genutzt werden. Ebenfalls wurde das Monitoring in das DARIAH-Monitoring integriert.

Nutzung und Überführung der TextGrid-Dienste in die DARIAH-DE Infrastruktur

Die Überführung der TextGrid-Infrastruktur nach DARIAH-DE ist weitestgehend abgeschlossen:

- Die TextGrid-Produktivserver wurden in die GWDG ESX-Cluster überführt und sind so hochverfügbar. Die TextGrid-Testserver wurden in der GWDG-Compute-Cloud erstellt und für Tests der HA-Komponenten und der neuen Puppet-Konfiguration genutzt. Alte Services und Server wurden weitestgehend abgeschaltet und werden spätestens bis zum Projektende alle abgeschaltet sein (einige sind noch für Testzwecke notwendig).

- Im Rahmen dieses Umzugs wird auch die Produktivinstanz des TextGrid Repository per Puppet konfigurierbar gemacht. Die Module hierfür kommen direkt aus den Grundmodulen, die in DARIAH-DE entwickelt wurden. Diese wurden für TextGrid angepasst, bzw. konfigurierbar gemacht. Ein Zusammenwachsen der TextGrid- und DARIAH-DE-Repository Services wird sowohl im Quellcode (u. A. TG-crud/DH-crud, TG-publish/DH-publish), als auch in den Repository-Modulen der Puppet-Konfiguration realisiert.[3]

- Das Monitoring wurde komplett für die Produktivserver ESX1 und ESX2 in das Monitoring von DARIAH-DE eingebunden.

- Die AAI-Infrastruktur wurde im Rahmen von DARIAH-DE mit der von TextGrid zusammengeführt. Abschließende Arbeiten werden momentan noch in DARIAH-DE durchgeführt.

- Die Planung der Migration des Storage von StorNext zur DARIAH Storage-API wird noch bis zum Projektende von TextGrid durchgeführt. Diese ist abhängig von den Entwicklungen des DARIAH-DE Repositoriums und deren unterliegenden Speicherkomponenten. Weiterhin ist eine Dauer dieser Datenmigration aufgrund der großen Datenmengen kaum abzuschätzen, so dass die Migration selbst noch in TextGrid begonnen werden kann und möglicherweise in DARIAH-DE abgeschlossen wird.

3Siehe hierzu auch den DARIAH-DE Bericht zum Meilenstein 4.3.2.1: DARIAH-DE Repositorium – Prototyp.
https://dev2.dariah.eu/wiki/display/publicde/Reports+and+Milestones

- Alle Entwicklungs-Tools und die gesamte Continuous Integration für die Software-Entwicklung sind ebenfalls zum DARIAH-DE Developer Portal[4] umgezogen, zu nennen sind hier in erster Linie das GIT Repository, das Projektmanagement sowie das Bugtracking (Chili-Project der GWDG)[5]. Weiterhin nutzt TextGrid einen Jenkins Server und ein Nexus Maven Repository, das von DARIAH-DE zur Verfügung gestellt wird sowie schon seit längerer Zeit das DARIAH-DE Wiki Confluence.

Zusammenfassung

Die Bereitstellung einer Hochverfügbarkeit für TextGrid betrifft die Kerndienste TG-auth, TG-crud und TG-search. Um diese Dienste in die Hochverfügbarkeit zu überführen, ist die Cluster-Fähigkeit der unterliegenden Technologien wichtig. Von diesen sind einige leicht als Cluster aufzubauen, wie zum Beispiel ElasticSearch, andere bedürfen einiges an Arbeit, wie die Berkeley Datenbank für NOID. Im Fall des Triplestores muss die Technologie ausgetauscht werden, da Sesame kein Clustering unterstützt. Auf der Infrastruktur-Ebene wurden zwei hoch performante Server aus dem ESX-Bereich der GWDG bereitgestellt, sowie ein Hardware-Loadbalancer. Durch Puppet kann ein TextGrid Server nun automatisch installiert und konfiguriert werden. Dies ermöglicht die schnellere Bereitstellung eines neuen Servers.

Durch Zusammenlegung des Monitorings, der Speichertechnologie und der AAI konnten TextGrid und DARIAH-DE enger zusammenwachsen.

4Vgl. DARIAH-DE Developer Portal. https://de.dariah.eu/developer-portal
5Vgl. Projektserver der GWDG. https://projects.gwdg.de/projects