WissGrid

Spezifikation

Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten

WissGrid Dienste Framework[1]

Autoren	Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten
Editoren	Andreas Aschenbrenner, Timo Henne, Jens Ludwig, Angelika Reiser
Datum	15. April 2011
Dokument Status	in Bearbeitung
Dokument Version	0.1.1

Änderungen

Version	Date	Name	Brief summary
0.1.0	27.08.2010	Ludwig	Template
0.1.1	28.03.2011	Henne	Abstract, Meeting notes, overview

[1]This work is created by the WissGrid project. The project is funded by the German Federal Ministry of Education and Research (BMBF).

Zusammenfassung

Dieses Dokument beschreibt die Anforderungen an das WissGrid Dienste Framework (WDF) zur Kommunikation zwischen Diensten und Repositories. Neben der generellen Struktur und den Aufgaben des WDF werden die Themen Scheduling, Allokation von Daten und Rechenleistung behandelt. Anhand der Ergebnisse von Recherchen und Erfahrungen aus anderen Projekten werden geeignete Komponenten sowie die darüberhinausgehenden notwendigen Anpassungen beschrieben. Schließlich wird am Beispiel eines Anwendungsfalles im Projekt TextGrid eine prototypische Umsetzung dieser Anforderungen mittels der ausgewählten Komponenten skizziert.

Inhaltsverzeichnis

1 Einleitung

Das WissGrid Dienste Framework (WDF) dient zur Ausführung und Koordinierung der LZA-Dienste sowie der Kommunikation zwischen Diensten und Repository. Obwohl das WDF eine zentrale Rolle in der Einbettung von LZA-Diensten in ein Repository hat, ist es eine von dem Repository unabhängige Komponente. Als solches ist das WDF generisch und auch zur Kommunikation mit mehreren, unterschiedlichen Repositorien ausgelegt.

Eine grundlegende Unterscheidung in der Koordination der Daten und der LZA-Dienste liegt in der Frage: *Daten zu den Diensten, oder Dienste zu den Daten?* Eine Entscheidung hierüber hängt einerseits ab von Performance-Fragen (Wie groß sind die Daten? Wie rechenintensiv ist die Ausführung des Dienstes auf den spezifischen Datenformaten?), andererseits aber auch von Sicherheitsfragen (Dürfen externe/fremde Dienste innerhalb eines vertrauenswürdigen Langzeitarchivs ausgeführt werden?) und letztlich der Architektur des Repositories (Lässt die Softwarearchitektur und die (verteilte?) Hardware die Ausführung von LZA-Diensten zu?). Daher muss jede Community diese Entscheidung für sich treffen. Im Rahmen des WissGrid-Projektes wird aufgrund von Sicherheitsbedenken, und mit dem Ziel einer möglichst generischen Lösung nur das Muster „Daten zu den Diensten“ verfolgt. Partnerprojekte (z.B. AstroGrid) experimentieren aber auch mit Community-spezifischen Ansätzen für „Dienste zu den Daten“, und die dort gesammelten Erfahrungen werden perspektivisch in WissGrid einfließen.

Dieses Dokument stellt eine Fortführung des Deliverables 3.4.2 zur LZA-Spezifikation [10] dar. Die folgenden Abschnitte beschreiben die Aufgaben (Abschnitt 2) und die Architektur (Abschnitt 3) des WDF, sowie Anforderungen und mögliche Lösungen für die erforderliche Scheduling-Komponente (Abschnitt 4). Abschließend findet sich in Abschnitt 5 eine Spezifikation des ersten Anwendungsfalles TextGrid-Masseningest. Die Schnittstelle zu den Repositorien ist im WissGrid-Deliverable 3.5.2 [9] in Kapitel 2 beschrieben.

2 Aufgaben des WDF

Das WDF koordiniert die Kommunikation zwischen einem Repository und vorhandenen LZA-Diensten. Dieser Abschnitt konzentriert sich auf LZA-Dienste, die als Grid-Dienste genutzt werden. Die Nutzung von Web Services für LZA wie dem Planets Interoperability Framework [5] ist auf einer höheren Applikationsschicht anzusiedeln, und deren Integration in das WDF wird derzeit nicht avisiert.

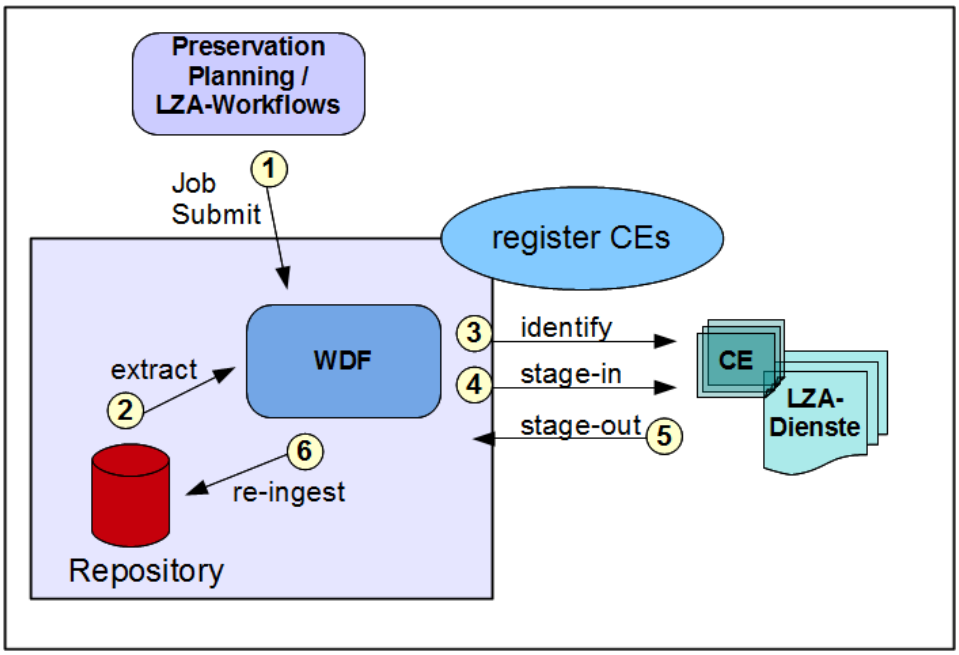

Abbildung 1: Interaktion zwischen Diensten und Repository durch WDF

In Abbildung 1 ist die Kommunikation zur Ausführung einer typischen LZA-Aufgabe schematisch dargestellt. Die Hauptkomponenten sind hierbei

- der Auslöser einer LZA-Aufgabe, wobei hier zwischen dem organisatorischen, durch Menschen gesteuerten Planungsvorgang und Beschluss (z.B. die Preservation Planning Einheit eines Forschungsarchivs) einerseits, und der technischen Durchführung (z.B. durch Trigger) andererseits zu unterscheiden ist
- das Repository, in dem die Daten vorliegen
- der WDF-Dienst, der meist eng mit einem Repository verknüpft ist und auch in einer gemeinsamen Sicherheitsdomäne angesiedelt ist
- die LZA-Dienste, die auf Computing Elementen (CEs) installiert sind und über Grid-Mechanismen angesprochen werden können

Die Beschreibungen von LZA-Diensten und der zugehörigen CEs werden hierbei explizit beim WDF zur Nutzung registriert.

Der Ablauf einer LZA-Aufgabe wird über das WDF koordiniert und beinhaltet folgende Schritte:

1. **Submit:** Annahme der LZA-Aufgabe, z.B.:
   - Konvertiere alle TIFF Daten, die älter als 2 Jahre sind, in ein JPEG 2000 Format
   - Extrahiere aus eingehenden Bilddateien alle Metadaten, welche Erstellung und Bearbeitung betreffen
2. **Extract:** Filterung der gefragten Daten aus dem Repository *oder* Annahme von Daten aus einem **Ingest**
3. **Identify:** Auswahl einer verfügbaren CE, auf der der gefragte Dienst installiert ist
4. **Stage-In:** Transfer der Daten (im Batch) auf das CE
5. **Stage-Out:** Annahme der konvertierten Daten, nach Ablauf des Jobs; ggf. Zwischenspeicherung der Daten
6. **Re-Ingest:** Rückführung in das Repository, gegebenenfalls unter neuerlicher Vergabe von Metadaten und Erstellung einer neuen Objekt-Version

3 Architektur

In dieser ersten Version der Spezifikation wird eine vorläufige Skizze des WDF vorgestellt, welche im weiteren Projektverlauf unter Klärung der insbesondere in den Abschnitten 4 und 5 aufgeworfenen Fragen fortgeschrieben und konkretisiert wird.

Der Einsatz des WDF ist an mehreren unterschiedlichen Stellen denkbar:

- aus der Administration des Repositories (Archival Information Update, OAIS) heraus, wenn eine Routine LZA-Aufgabe im laufenden Betrieb erfüllt werden muss, z.B. wenn alle Objekte in einem veralteten Format in ein aktuelles Format migriert werden
- beim Einbringen neuer Objekte in das Repository („Ingest“), z.B. zur Validierung von Dateiformaten
- beim Zugriff auf Repository-Inhalte („Access“), z.B. bei einer Migration-on-Access LZA-Strategie, bei der die Inhalte als Originale oder in einem Standardformat im Repository vorgehalten werden, und nur beim Zugriff in ein vom Nutzer gewünschtes Format überführt werden
- für Nutzer-gesteuerte Workflows, in denen LZA-Dienste benötigt werden

Um diese generische Einsetzbarkeit zu erreichen, ist das WDF als unabhängige Komponente konzipiert, die aber physisch „nahe“ an einem Host-Repository installiert wird

(a) um maximale Datenübertragungsraten für die oft umfangreichen zu transportierenden Datenmengen zu erreichen, und auch

(b) um ein Repository und ein zugehöriges WDF in einer gemeinsamen Sicherheitsdomäne zu behalten und damit Rechtefragen zu vereinfachen.

Somit wird perspektivisch für jedes Repository ein WDF installiert. Um dem WDF die Kommunikation mit verschiedenen Repositories zu erlauben, die auf unterschiedlichen Software-Plattformen basieren, ist die Definition der Schnittstelle zwischen dem Repository und dem WDF (siehe Deliverable 3.5.2 [9], Kapitel 2), wie auch die Schnittstelle zwischen dem WDF und den Diensten als Standard zu definieren.

Die Registrierung der verfügbaren LZA-Dienste auf den CEs wird im Rahmen von WissGrid voraussichtlich lediglich hart-kodiert. Perspektivisch wird die Architektur jedoch so offen ausgelegt, dass die vom Partnerprojekt WisNetGrid [16] geplante Service Registry an das WDF angedockt werden kann. Geprüft wird auch, in welchen Fällen CEs und ihre zugehörigen LZA-Dienste einer expliziten „Trust Zone“ (z.B. mit einem dezidierten LZA-Server mit starken Sicherheitsrestriktionen, statt einem öffentlichen Grid-Server) zugewiesen werden müssen, um das Vertrauen der Nutzer zu stärken.

4 Scheduling-Aufgaben im WDF

Eine Hauptaufgabe des WDF liegt in der Verwaltung und Steuerung von im Grid verfügbaren Diensten. Um die Ausführung und Abarbeitung dieser Dienste effizient zu gestalten, ist ein sogenannter Scheduler erforderlich, welcher die verfügbaren und geeigneten Computing Elements (CEs) identifiziert und sowohl den ggf. erforderlichen Datenfluss von den Storage Elements (SEs) zu den CEs als auch die Dienstausführung steuert und überwacht. In diesem Abschnitt werden die konkreten Anforderungen an einen solchen Scheduler benannt und mit den Spezifikationen von existierenden Schedulern abgeglichen.

4.1 Anforderungen an Scheduler

Die vom Scheduler konkret zu verrichtenden Aufgaben hängen zum Teil von den jeweils auszuführenden Diensten ab. Je nach Dienst muss ebenfalls entschieden werden, ob ein Scheduling überhaupt notwendig und sinnvoll ist. So sind zum Beispiel im Anwendungsfall des TextGrid-Masseningests die Parallelisierung der Dienstausführung sowie die Berücksichtigung des Datentransferaufwandes („data awareness“) entscheidende Kriterien, welche den Einsatz eines Schedulers erforderlich machen. In anderen Fällen (z.B. Ingest eines einzelnen Objekts) kann die Dienstausführung jedoch ggf. ohne Scheduling direkt auf dem Rechner des Zielrepositories erfolgen. Innerhalb des WDF muss somit eine Instanz in jedem Fall darüber entscheiden, ob und in welchem Umfang der Scheduler eingesetzt wird. Eine zu klärende Frage besteht somit darin, ob diese Entscheidung vom verwendeten Scheduler selbst getroffen werden kann, oder ob hierfür eine Erweiterung des Schedulers bzw. eine eigenständige Komponente im WDF erforderlich ist.

Eine andere Frage betrifft die vom WDF zu speichernden Informationen über die CEs im Grid. Eine Anforderung für die sinnvolle Verteilung der Dienstausführungen im Grid ist zunächst, dass sämtliche verfügbaren CEs beim WDF registriert werden, einschließlich der bei ihnen installierten Dienste. Zu erwägen ist weiterhin, ob darüberhinaus auch durchschnittliche oder garantierte Transferraten von und zu den CEs gespeichert werden können und sollen. Diese Informationen könnten dann im WDF direkt mit den Informationen über die für einen Prozess (Ingest, Access, etc.) notwendigen

Dienste sowie den Anforderungen an selbige verknüpft und dem Scheduler zur Verfügung gestellt werden. Als einzige vom WDF bei den CEs direkt zu erfragende Information verbliebe dann die aktuelle Auslastung der CEs hinsichtlich Speicher und Rechenkapazität.

Um eine möglichst generische Struktur des WDF zu gewährleisten, welche mit verschiedenen Repository-Systemen und Grid-Ressourcen zusammenarbeiten kann, sollten bei der Wahl eines Schedulers alle möglichen Fälle berücksichtigt werden.

Unabhängig von der konkreten Implementierung des WDF sollte der eingesetzte Scheduler folgende Kernaufgaben erfüllen können:

- **Stage-In:** Verteilung und Transfer der Daten auf die CEs
- Aufruf der Dienste auf den CEs
- **Stage-Out:** Annahme der Dienst-Ergebnisse und Weiterleitung an Nachbearbeitung
- Überwachung der Transfers und Dienst-Ergebnisse auf Fehler, sowie Behandlung dieser Fehler (z.B. Re-Scheduling)

In Abhängigkeit vom eingesetzten Scheduler und dessen Anpassbarkeit sowie vom Leistungsumfang der übrigen Repository-Komponenten ist zu entscheiden, ob einige der folgenden Aufgaben ebenfalls vom Scheduler übernommen werden können:

- **Submit:** Annahme der Daten vom Producer (s. OAIS-Modell)
- **Extract:** Extrahieren der erforderlichen Daten aus dem Repository
- Entscheidung über Einsatz und Umfang von Scheduling
- **Identify:** Ermittlung der geeigneten CEs einschließlich Berechnung und Abwägung des Transfer- und Rechenaufwands
- Aufteilung der Daten in für Transfer und Dienstausführung geeignete Pakete (bei Masseningests oder -access)

Sinnvollerweise nicht direkt vom Scheduler, sondern von anderen Komponenten zu übernehmende Tasks sind hingegen:

- Ermittlung der für den jeweiligen Prozess auszuführenden Dienste sowie der Anforderungen an die CEs hinsichtlich Transferrate und Rechen- sowie Speicherkapazität
- Aufruf der Nachbearbeitungs-Prozesse (PID-Vergabe, Suchindex-Erstellung etc.)
- **(Re-)Ingest:** Speicherung bzw. Aktualisierung der Daten im Repository

4.2 Existierende Scheduler

In diesem Abschnitt werden einige gebräuchliche Scheduler anhand ihrer eigenen Spezifikationen kurz vorgestellt und im Hinblick auf die in Abschnitt 4.1 aufgeführten Anforderungen hinsichtlich ihrer Eignung für den Einsatz im WDF bewertet.

4.2.1 GridWay

Dieser Scheduler stammt aus der Globus Community [2] und integriert sich folglich nahtlos in das weit verbreitete Globus Toolkit. GridWay [8] wird unter der Bezeichnung „Workload Manager“ zur Verwaltung von Diensten und Ressourcen im Grid geführt. Die hervorstechendsten Merkmale dieses Schedulers sind die umfangreiche Fehlerbehandlung, die Fähigkeit zu dynamischem Scheduling inklusive Migration sowie die Unterstützung von Scheduling Policies und voneinander abhängigen Diensten. Eine Berücksichtigung von data awareness ist hier noch nicht implementiert, allerdings existiert ein erster Prototyp hierzu [12].

4.2.2 CSF

Das Community Scheduler Framework (CSF) [1] ist ebenfalls ein Produkt der Globus Community und wurde bereits in der Globus Toolkit Version 3.0 integriert. Es handelt sich hierbei um eine Open-Source Implementierung von verschiedenen Grid-Services, welche zusammen die Funktionalität eines Grid-Meta-Schedulers umfassen. Für die konkrete Anwendung im Grid kann das CSF als Baukasten für sogenannte Community Scheduler genutzt werden, welche dann auf die jeweilige Anwendung und Grid-Struktur der entsprechenden Virtuellen Organisation zugeschnitten sind. Das CSF bietet eine Abstraktionsebene für Konzepte wie „Job“, „Ressourcen-Reservierung“ oder „Scheduling“ und forciert die Einhaltung von Standards wie des OGSI-Agreements [6]. Konkrete Mechanismen zur Berücksichtigung des Datentransferaufwands sind hier jedoch nicht erkennbar.

4.2.3 GWES

Der Generic Workflow Execution Service (GWES) [7] wurde vom Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik (FIRST) entwickelt. Er verwendet eine eigens entwickelte Beschreibungssprache (GWorkflowDL), welche auf High-Level Petri-Netzen basiert und eine umfassende Beschreibung und Modellierung von Workflows erlaubt. GWES wird zur Beschreibung von Workflows bereits in mehreren Grid-Projekten angewendet, darunter auch TextGrid [14]. Die Unterstützung von data allocation scheint sich der Spezifikation nach jedoch auf voneinander abhängige Dienste innerhalb eines Workflows zu beschränken und keine grundsätzlichen Erwägungen bei der Auswahl der dienstausführenden Ressourcen in Abhängigkeit vom Speicherort der Daten zu beinhalten.

4.2.4 WSS

Der Workflow Scheduling Service (WSS) ist eine Scheduler-Entwicklung des Projektes C3Grid [3]. Er wurde für die direkte Interaktion zwischen Benutzer, Repository und Diensten entwickelt und wird im Rahmen von C3Grid von Benutzern direkt über ein Portal angesprochen. Seine Aufgaben liegen in der Koordination von Task-Ausführungen im Grid sowie im Planen von Datenzugriffen, wobei er eng mit dem C3Grid Data Management System (DMS) zusammenarbeitet. In diesem Zusammenhang wird auch das Problem der Datenallokation betrachtet.
Um die Verwaltung von gegenseitig abhängigen Diensten zu unterstützen, wurde hierbei eine eigene, auf der standardisierten Job Specification and Description Language (JSDL) aufbauende Sprache zur Dienstebeschreibung entwickelt, die Workflow Specification Language (WSL). Die Überwachung

von Dienstausführungen wird jedoch laut Spezifikation nicht vom Scheduler WSS selbst geleistet, sondern vom C3Grid-Portal koordiniert.

4.2.5 KOALA

Beim Co-allocating Grid Scheduler KOALA [4], einer Entwicklung der TU Delft im Rahmen des VL-e-Projekts [11], steht die Integration von Daten- und Prozessor-Zuweisungen für datenintensive Grid-Jobs im Vordergrund. Der Scheduler identifiziert dabei zunächst geeignete Ressourcen im Grid, transferiert dann die Daten zu diesen Ressourcen und bemüht sich erst anschließend um die Reservierung von Rechenkapazitäten. Hinsichtlich data-awareness kann KOALA verschiedene Co-Allocation policies verfolgen, so z.B. Load-aware (WF), Input-file-location-aware (CF) oder Communication-aware (CM/FCM/CA) [13]. Die Entwicklung von KOALA scheint jedoch seit dem Projektende 2009 nicht weitergeführt worden zu sein — auch läßt sich keine Dokumentation zur Verwendung dieses Schedulers in anderen Umgebungen finden.

4.2.6 WisNetGrid

Im Projekt WisNetGrid [16] wird an einer generischen Lösung für die Beschreibung von Diensten und Workflows gearbeitet. Ein erstes Ergebnis hierbei ist eine eigene Dienstebeschreibungssprache suprimePDL [15], welche auf dem π-Kalkül basiert und verschiedenste Anforderungen zur allgemeinen Modellierung und Beschreibung von Prozessen abdeckt, u.a. auch Dienstekomposition. Eine Architektur oder gar Anwendung zur Erfüllung von Scheduling-Aufgaben ist hier bisher nicht vorhanden. Auch Fragen der data-awareness werden in WisNetGrid bisher nicht angesprochen.

4.2.7 Zusammenfassung

Aus den in diesem Abschnitt beschriebenen Spezifikationen sowie aus Gesprächen mit Partnern aus anderen Projekten (C3Grid, TextGrid, WisNetGrid) lassen sich folgende Schlüsse ziehen:

- Viele zur Zeit vorhandene Scheduler sind das Ergebnis von akademischen Projekten und als solche hinsichtlich ihrer weiteren Unterstützung nicht gesichert.
- Die konkrete Eignung eines Schedulers bezüglich seiner Anpassungsfähigkeit an verschiedene Grid-Kontexte und -Anforderungen lässt sich nur durch die Implementierung geeigneter Testfälle abschätzen.
- Die Implementierung von data-awareness, also der Berücksichtigung des erforderlichen Datentransferaufwandes bei der Jobverteilung, ist bisher nur wenig verbreitet.
- Der Fokus in WissGrid sollte zunächst auf der genauen Spezifikation von Anwendungsfällen und deren Anforderungen, sowie der Umsetzung dieser Anwendungsfälle mit möglichst weit verbreiteten Schedulern liegen.
- Zur Verwendung in diesen Anwendungsfällen schlagen wir daher zunächst die Scheduler GridWay und WSS, falls erforderlich in einer weiteren Stufe GWES, eine eigene Implementierung basierend auf dem CSF und evtl. KOALA vor.

5 Anwendungsfälle

Als erster Anwendungsfall wird hier zunächst nur der Anwendungsfall TextGrid Masseningest beschrieben. Im weiteren Projektverlauf werden jedoch weitere Fälle für verteilte Dienstausführung, wie z.B. umfangreiche Access-Anfragen, betrachtet.

5.1 TextGrid Masseningest

Bei diesem Anwendungsfall sollen über einen gewissen Zeitraum erstellte Digitalisate des Göttinger Digitalisierungszentrums (GDZ) im Paket in das Grid-Repository geschrieben werden. Hierbei ist eine Formatvalidierung und -extrahierung durch den Dienst JHOVE2 durchzuführen, und die Ergebnisse dieses Dienstes sind mit den Repository-Daten zu verknüpfen. Als minimal ausreichende Performance wird hierfür definiert, ein Paket von der Größe der in einem Monat anfallenden Daten innerhalb eines Monats zu verarbeiten und im Repository zu speichern. Detailliertere Spezifikationen zu den in diesem Anwendungsfall verwendeten Daten finden sich in Anhang A.

5.1.1 Workflow

Im Folgenden werden die erforderlichen Schritte des Anwendungsfalles beschrieben. Dabei ist hier nur der Ablauf nach Ingest Variante A (s. Abschnitt 5.1.2) aufgelistet, Abweichungen für Variante B sind durch Fußnoten kenntlich gemacht.

1. **Submit:** Auslösen des Ingest Vorganges durch das GDZ
   a) Annahme des Ingest-Vorgangs durch den TextGrid Server
   b) Transfer der Daten vom GDZ zum TextGrid-Server*
   c) Modellierung der angelieferten Daten gemäß dem TextGrid-Objektmodell bzw. den unterstützten Import-Profilen (z.B. DFG-Viewer METS)*
2. **Lokale Verarbeitung I:** *
   a) Überprüfung der Metadaten
   b) Auflösen der Referenzen zwischen Dateien
3. **Ingest:** Schreiben der Daten in den Grid-Speicher*
   a) Schreiben in iRODS Server
   b) ggf. Benachrichtigung und Aktualisierung von Fedora mittels Callback
4. **Dienstausführung** Formatvalidierung sowie Extrahierung der tech. Metadaten mit JHOVE2
   a) **Identify:** Ermittlung von verfügbaren CEs, auf welchen der gefragte Dienst installiert ist, einschließlich
      i. (geschätztem) Rechenaufwand
      ii. Verfügbarkeit von (temporärem) Speicher
      iii. (erwartetem) Transferaufwand

und darauf basierend Auswahl der CEs und ggf. Aushandeln/Reservieren von Rechenzeit

b) **Extract:** Extrahieren der benötigten Daten aus dem Repository und Aufteilen in geeignete Pakete†
c) **Stage-In:** Transfer der Daten auf die CEs†
d) Ausführung des Dienstes (JHOVE2) auf den jeweiligen CEs
e) **Stage-Out** Annahme des Ergebnisses und Weiterleitung zur Verarbeitung (Schritt 5b)

5. **Lokale Verarbeitung II:**
   a) Vergabe von PIDs (Persistent Identifier)
   b) Aktualisieren der Metadaten
   c) Erstellung des Suchindexes
   d) Aktualisieren der Rechte
6. **Notify:** Benachrichtigung über Abschluß des Ingest-Vorgangs bzw. über evtl. aufgetretene Probleme

Insgesamt ist das WDF für die Aufgaben in Schritt 4 zuständig. Weiterhin wird davon ausgegangen, dass der verwendete Scheduler, wie in Abschnitt 4.1 beschrieben, mindestens die Schritte 4c, 4d und 4e initiiert und koordiniert. Alle weiteren Schritte werden entweder vom TextGridServer, Fedora oder iRODS übernommen.

5.1.2 Mögliche Ingest-Verfahrensweisen im Anwendungsfall TextGrid Masseningest

Je nach Struktur des Repositories und Lage der Dienste und Daten lassen sich zwei generelle Varianten des Ingests unterscheiden, welche im Diagramm in Abbildung 2 dargestellt sind und im Folgenden erläutert werden. Rote Linien kennzeichnen hierbei Kommunikationswege zwischen Komponenten, bei denen der Transfer von in das Repository einzuspielenden Daten stattfindet.

Ingest Variante A: Der Fokus dieser Ingest-Variante liegt auf dem initiierenden Submit(1) an TextGrid CRUD (Create, Retrieve, Update, Delete), welches direkt den Fedora-Server anspricht. Fedora prozessiert zunächst die erhaltenen Daten(2) durch Überprüfung der bereits enthaltenen Metadaten sowie durch Auflösen von eventuellen Referenzen zwischen den Dateien. Wenn möglich parallel hierzu startet Fedora die Anfrage(4) an das WDF zur Identifizierung(4a) der für die angefragten/anfallenden Dienste in Frage kommenden CEs. Nach erfolgter lokaler Verarbeitung beginnt Fedora mit dem sequentiellen Ingest in iRODS (3a). Nach erfolgreichem Ingest (oder evtl. auch schon parallel hierzu) erfolgt dann durch den Scheduler des WDF das Extrahieren(4b) der Daten aus dem iRODS-Repository, das Stage-In(4c) zu den identifizierten CEs, das dortige Verarbeiten(4d) der Daten durch die Dienste sowie das Stage-Out(4e) der Ergebnisse an Fedora und die abschließende Verarbeitung der Dienstergebnisse(5) sowie die Aktualisierung der Rechte. Abschließend wird

*Bei der Ingest Variante B (s. Abschnitt 5.1.2) findet der Ingest nach iRODS bereits vor der **Lokalen Verarbeitung I** statt. Sämtliche weiteren Schritte werden dann per Callback an Fedora oder über ein Messaging-System — ausgehend von iRODS — angestoßen.

†Die Schritte 4b und 4c sind je nach Verteilung der Daten auf den CEs nur bedingt erforderlich, was in Abbildung 2 durch die gestrichelten Linien verdeutlicht wird.

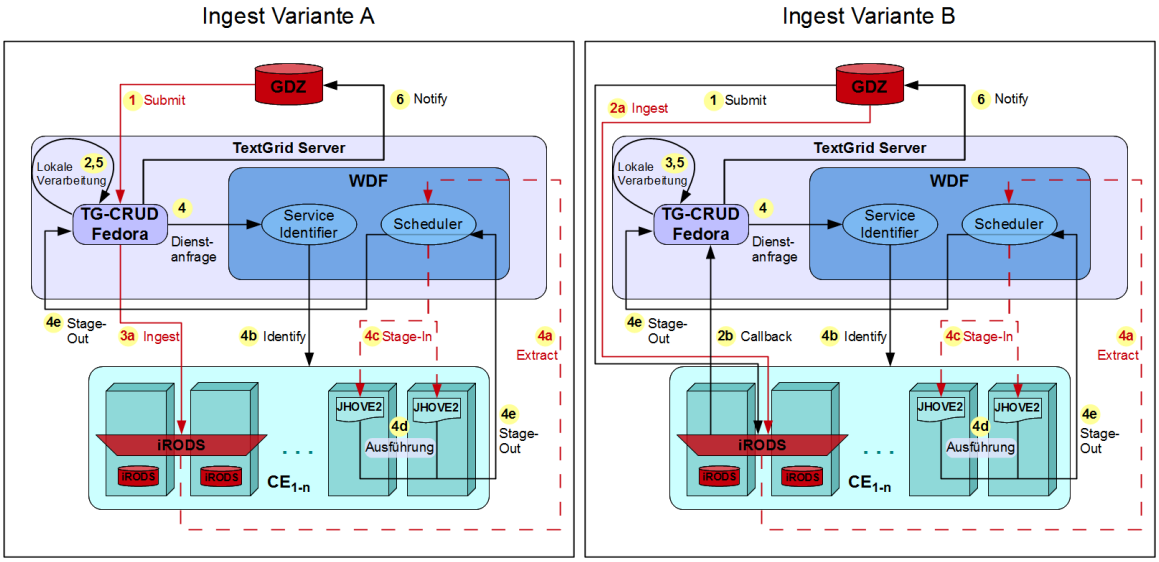

Abbildung 2: Mögliche Ingest Varianten im Anwendungsfall

das GDZ über den Abschluß benachrichtigt(7).
Zusammenfassend läßt sich diese Variante wie folgt beschreiben:

1. GDZ initiiert Ingest über TextGrid-CRUD an Fedora
2. Fedora leitet Ingest-Anfrage an WDF weiter, wo Formatvalidierung wie unter 5.1.1 beschrieben durchgeführt wird
3. Parallel hierzu werden Daten sequentiell im iRODS-Server gespeichert

Ingest Variante B: Bei dieser Ingest-Variante erfolgt der Submit(1) und sowie Ingest(2a) unter Umgehung von Fedora direkt an den iRODS-Server. Mittels Callback-Aufrufen(2b) wird nach dem Ingest (oder parallel hierzu) Fedora über die neuen Daten informiert und initiiert parallel zur lokalen Verarbeitung(3) wiederum die Identifizierung(4a-e) und Abarbeitung der Dienste, wie bereits unter **Ingest A** beschrieben. Die abschließenden Schritte (5-7) entsprechen ebenfalls der Ingest Variante A.

1. GDZ initiiert Ingest direkt in iRODS-Server
2. iRODS informiert Fedora per Callback über Ingest
3. Fedora beginnt mit Verarbeitung der vorhandenen Metadaten und stößt parallel hierzu Identifizierung und Ausführung der Dienste an

Bei beiden Varianten ist prinzipiell die Ausnutzung von Lokalität möglich, dass also die Dienste auf genau den CEs ausgeführt werden können, wo die Daten bereits vorhanden sind, um den erforderlichen Datentransfer zu reduzieren. Die Schritte **Extract** und **Stage-In** sind daher nur für die Fälle durchzuführen, in denen die Daten nicht auf den CEs liegen, wo die Dienste ausgeführt werden sollen. Hierfür werden allerdings in beiden Fällen Informationen über die physischen Speicherorte

benötigt, welche entweder über Fedora oder direkt bei iRODS abgefragt werden müssen.

Für die Ingest Variante A spricht, dass sich durch die Vorverarbeitung von Fedora evtl. die auszuführenden Dienste und somit geeignete CEs für den Ingest (und somit die Ermöglichung lokaler Diensteverarbeitung) sinnvoller identifizieren lassen. Die Ingest Variante B hingegen hat ihre Vorteile darin, dass zum einen die von iRODS selbst angelegte Dateistruktur deutlich sinnvoller strukturiert ist als bei einem Ingest über Fedora, zum anderen der Ingest durch die direkte Speicherung in iRODS parallelisiert werden könnte, während Fedora jedes Objekt sequentiell abarbeitet. Welche Variante schließlich effizienter arbeitet, hängt zum Großteil von den Möglichkeiten zur Parallelisierung der anderen Arbeitsschritte ab, welche sich erst bei der Implementierung zeigen werden. Auch hier gilt somit, dass die Entscheidung über den Einsatz der Varianten letztendlich von jeder Community selbst getroffen werden muss und das WDF beide Varianten unterstützen sollte.

A Spezifikationen zum Anwendungsfall TextGrid-Masseningest

Kollektionen: Exemplarische Auswahl an Masterfiles des Göttinger Digitalisierungszentrum (GDZ)

Datenquelle: GOOBI (Software) + GWDG (Storage)

Datenvolumen: zunächst exemplarisch im einstelligen GB-Bereich; perspektivisch im Bereich 500+GB/-Monat sowie ca. 100TB an bereits vorhandenen Bestandsdaten

Datenbeschaffenheit: TIFF-Dateien + METS/XML-Kurzbeschreibung + teilweise TEI-Volltextbeschreibung

Rechte in Bezug auf Ingest-Verarbeitung: bisher unproblematisch, da open access

Dienste: JHOVE2

Repository/Datensenke: Fedora-iRODS Server + GWDG Storage

Rechenressourcen: eigene Globusinstallation + evtl. verteilte D-Grid-Ressourcen

Grid-Knoten :

- Göttingen, GWDG
- evtl. später Würzburg, Uni (Replikation)

Zeitbegrenzung: bei Dateien mit ca. 2MB Größe, Gesamtvolumen 500GB:
200000 Dateien im Monat, also 1 Datei in 12 Sekunden

Parallelisierbarkeit: alle Dateien einzeln prozessierbar

Effizienz des bisherigen Verfahrens: noch zu testen

B Scheduler Übersicht

Die folgende Tabelle liefert eine komprimierte Übersicht über die Spezifikationen der in Abschnitt 4.2 beschriebenen Scheduler.

Bez.	GridWay	GWES	WSS	(WisNetGrid)	KOALA	CSF
Name	GridWay	Generic Workflow Execution Service	Workflow Scheduling Service	—	Processor and data co-allocation system	Community Scheduler Framework
Entwickler/ Projekt	Globus	Fraunhofer FIRST	C3Grid	WisNetGrid	TU Delft	Globus
URL	2	3	4	5	6	7
Grid-Kontext/ Middleware	Globus	Globus	Globus	—	JavaGAT	Globus
Beschreibungs-sprache(n)	JSDL	GWorkflowDL	WSDL	suprimePDL	Globus RSL	Globus RSL
Data awareness	Prototyp	nein	ja	nein	ja	(Interfaces)

[b] http://www.gridway.org/
[c] http://www.gridworkflow.org/
[d] http://www.c3grid.de/
[e] http://www.wisnetgrid.org/
[f] http://www.st.ewi.tudelft.nl/koala/
[g] http://www.globus.org/toolkit/docs/4.0/contributions/csf/

Literatur

[1] Globus Alliance. Community scheduler framework. http://www.globus.org/grid_software/computation/csf.php, 2009.

[2] Globus Alliance. Globus alliance. http://www.globus.org/, 04 2011.

[3] C3Grid. C3grid - collaborative climate community data and processing grid. http://www.c3grid.de/, 04 2011.

[4] TU Delft. Koala co-allocating grid scheduler. http://www.vl-e.nl, 04 2009.

[5] A. Farquhar and H. Hockx-Yu. Planets: Integrated services for digital preservation. *Int. Journal of Digital Curation*, 2(2):88–99, 2007.

[6] The Global Grid Forum. Open grid services infrastructure (ogsi). http://www.ggf.org/documents/GFD.15.pdf, 06 2003.

[7] The Grid Workflow Forum. Generic workflow execution service. http://www.gridworkflow.org/snips/gridworkflow/space/GWES, 04 2011.

[8] GridWay. Gridway scheduler. http://www.gridway.org/, 04 2011.

[9] WissGrid Konsortium. Wissgrid-spezifikation: Grid-repository. http://www.wissgrid.de/publikationen/deliverables/wp3/WissGrid-D3.5.2-grid-repository-spezifikation.pdf, 04 2008.

[10] WissGrid Konsortium. Wissgrid-spezifikation: Langzeitarchivierungsdienste. http://www.wissgrid.de/publikationen/deliverables/wp3/WissGrid-D3.4.2-lza-dienste-spezifikation.pdf, 04 2010.

[11] University of Amsterdam. Virtual laboratory for e-science. http://www.vl-e.nl, 10 2008.

[12] A.D. Peris, J. Hernandez, E. Huedo, and I.M. Llorente. Data location-aware job scheduling in the grid. application to the gridway metascheduler. In *Journal of Physics: Conference Series*, volume 219, page 062043. IOP Publishing, 2010.

[13] O. Sonmez, H.H. Mohamed, and D.H.J. Epema. Communication-aware job placement policies for the koala grid scheduler. In *Proc. of the Second IEEE International Conference on e-Science and Grid Computing*, pages 79–87, 2006.

[14] TextGrid. Textgrid - vernetzte forschungsumgebungen für e-humanities. http://www.textgrid.org/, 04 2011.

[15] WisNetGrid. Semantische beschreibungssprache für web-dienste. http://wisnetgrid.org/index.php?option=com_jdownloads&Itemid=0&view=finish&catid=4&cid=6&lang=de, 06 2010.

[16] WisNetGrid. Wissensnetzwerke im grid. http://wisnetgrid.org/, 12 2010.