FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Interoperabilität von Grid-Systemen am Beispiel von UNICORE und Globus

Michael Rambadt

FZJ-ZAM-IB-2002-07

1. Auflage

(letzte Änderung: Mai 2002)

Zusammenfassung

In den vergangenen Jahren hat es einen großen Fortschritt in der Entwicklung von Grid Technologien gegeben. In zahlreichen Projekten arbeiten Wissenschaftler von Universitäten und Forschungszentren weltweit an Verbesserungen im Grid-Computing. Auch wenn diese Projekte viele unterschiedliche Schwerpunkte setzen, haben alle das gemeinsame Ziel, Anwendern Zugang zu entfernten Computerressourcen zu ermöglichen. Dem Anwender bieten sich interessante Möglichkeiten, wenn es gelingt, die verschiedenen Projekte miteinander zu einer mächtigen Grid-Architektur zu verbinden. Dieses Ziel wird in der vorliegenden Arbeit am Beispiel von *UNICORE* (Uniform Interfaces to Computer Ressources) und *Globus* implementiert.

Sowohl UNICORE als auch Globus stellen eine Grid-Infrastruktur zur Verfügung, die es Anwendern ermöglicht, auf verteilte Ressourcen zuzugreifen. Beide Systeme bieten dem Benutzer eine Single-sign-on-Umgebung, deren Sicherheit auf x509v3-Zertifikaten aufbaut.
Globus, das unter der Leitung vom Argonne National Laboratory (USA) entwickelt wurde, bietet dem Benutzer zahlreiche Dienste an, die flexibel zur Grid-Entwicklung eingesetzt werden können. UNICORE schafft eine integrierte Lösung, welche einheitlichen Zugriff auf verteilte Computerressourcen ermöglicht. Es wird von einem Konsortium von deutschen Universitäten, Forschungszentren und Software Firmen entwickelt. UNICORE wurde vom Bundesministerium für Bildung und Forschung (BMBF) gefördert.

Das Ziel der hier vorgestellten Arbeit ist es, mit UNICORE und Globus eine kombinierte Grid-Lösung zu entwickeln, die Vorteile beider Systeme vereinigt. In dieser Lösung wird UNICORE als Klient fungieren, mit dessen Hilfe auf Globus-Ressourcen zugegriffen werden kann. Dieser Klient soll zunächst Job-Übermittlung, Statusabfragen, Datenaustausch und Ergebnissicherung ermöglichen. Es dürfen dabei weder an Globus, noch an UNICORE Veränderungen in der Architektur vorgenommen werden. Diese Vorgabe wird durch eine in dieser Arbeit entwickelte gemeinsame Schnittstelle erreicht.

Ein wesentlicher Aspekt ist die Abbildung der UNICORE- auf die Globus-Sicherheitsmechanismen. Zunächst verwenden beide System verschiedene Zertifizierungsbehörden mit unterschiedlichen Zertifizierungsvorschriften. Zudem unterscheiden sich die Authentisierungstechniken beider Systeme grundlegend. Während UNICORE jeden Teil des Jobs mit dem privaten Schlüssel des Benutzers signiert, arbeitet Globus mit Proxy-Delegation. Um beide Konzepte miteinander verbinden zu können, muß das Interface in der Lage sein, ein temporäres Globus-Proxy-Zertifikat zu erzeugen, das dem aus dem UNICORE-Job gebildeten Globus-Job angehängt wird.

Diese Arbeit gibt eine detaillierte Beschreibung der implementierten Schnittstelle zwischen UNICORE und Globus, die die oben formulierten Anforderungen erfüllt.

Die Funktionalität des Prototyps konnte erfolgreich auf der SC2001-Konferenz in Denver präsentiert werden.

UNICORE and Globus as examples of Grid System Interoperability

There has been a substantial progress in developing Grid technologies in the recent years. At universities and research centers world-wide scientists work on the evolution of Grid computing. Even if the way differs in many cases one of the principal goals of all those projects is the same: to give users access to distributed resources. Different projects focus on different aspects and it is only natural to combine them.

Both Globus and UNICORE provide a Grid infrastructure which gives users access to distributed resources. Both infrastructures offer a single sign on environment with X.509 based security. Globus which is developed under the leadership of Argonne National Laboratory and the University of Southern California can be characterised as a toolkit that allows the development of Grid application using a set of Globus services. UNICORE represents an integrated solution focusing on uniform access to distributed computing ressources. UNICORE is developed by a consortium of German universities, research laboratories, and software companies. It is funded by the Ministry for Education and Research (BMBF).

The goal of the work presented here which has been formed through a cooperation between Argonne National Labaratory and Forschungszentrum Jülich is to bridge between the two worlds combining the seamlessness of UNICORE with the flexibility of Globus. In this work UNICORE acts as the client to access selected Globus resources, allowing for job submission, status queries, data staging and output retrieval while Globus represents one of many UNICORE target systems.

In addition, the objective was to realize the goal without changes to Globus and UNICORE. This is achieved through an developed common interface.

One of the significant subjects is to map the UNICORE security mechanisms onto Globus ones. First of all, UNICORE and Globus use certificates issued by different Certificate Authorities with different policies. Both architectures deal with different authentication mechanisms: UNICORE guarantees the integrity of jobs by signing each part of a job with the user's certificate while Globus uses proxy delegation. For the mapping of the two concepts the interface has to obtain a Globus proxy certificate for the user to attach it to the Globus job it has generated from the UNICORE job. This work will give a detailed description of the established interface.

The functionality of this prototype has been demonstrated successfully on SC2001 in Denver.

Inhaltsverzeichnis

1 Motivation **1**

1.1 Rechnen auf verteilten Ressourcen 1

1.2 Struktur der Arbeit 2

2 UNICORE **5**

2.1 Motivation und Ziele 5

2.2 Die UNICORE Funktionen 5

2.3 Die UNICORE Architektur 7

2.3.1 Klient 7

2.3.2 Gateway 8

2.3.3 Network Job Supervisor 8

2.3.4 Target System Interface (TSI) 9

2.4 Die UNICORE-Sicherheitsmechanismen 10

3 Globus **13**

3.1 Motivation und Ziele 13

3.2 Die Globus Architektur 13

3.2.1 Globus Resource Allocation Manager (GRAM) 14

3.2.2 Globus Access to Secondary Storage (GASS) 15

3.2.3 Globus-Sicherheit 16

4 Die UNICORE Globus Kopplung **19**

4.1 Zielsetzung 19

4.2 Das Prinzip der UNICORE -Globus Job-Übermittlung 20

4.3 Der UNICORE - Globus Zertifizierungsmechanismus 21

4.4 Die Implementation 21

4.4.1 Erweiterungen im UNICORE TSI 22

4.4.2 Job Request Interface (JRI) 23

4.4.3 UNICORE Globus Zertifizierungsserver 27

4.5 Aufruf der ETSI-Komponenten 29

5 Status **31**

6 Ausblick **35**

Literaturverzeichnis **35**

Abbildungsverzeichnis

1.1 Das GRID am Beispiel einer Sternenwarte der NASA 1

2.1 Die UNICORE-Architektur 6
2.2 Der UNICORE-Klient 8
2.3 Der UNICORE-Server 9
2.4 Das Target System Interface 10

3.1 Das Globus-Schichtenmodell 14
3.2 Die GRAM-Architektur 15
3.3 Das Globus-Delegationsprinzip 17
3.4 Abbildung einer globalen auf eine lokale Benutzerkennung im GRAM-Gatekeeper 18

4.1 Enhanced Target System Interface (ETSI) 20
4.2 Implementierte und modifizierte Module des ETSI 22
4.3 Funktionsabhängigkeitsdiagramm 23
4.4 Verwendete UNICORE Parameter und deren Funktion 24
4.5 Verwendete RSL-Parameter und deren Funktion 24
4.6 UNICORE Globus Umsetzungstabelle 25
4.7 Grundgerüst der UNICORE und der Globus Ausgabe im UNICORE Klienten . . . 26
4.8 Das UNICORE Globus Zertifizierungsprinzip 27

5.1 Auswahl des Globus-Zielsystems mit dem UNICORE-Klienten 31
5.2 Job-Erstellung mit der Skript-Task 32
5.3 Job-Submission zu dem ausgewählten Globus-Zielsystem 33
5.4 Standard-Ausgabe des ausgeführten Jobs im UNICORE-Klienten (1.Teil) 34
5.5 Standard-Ausgabe des ausgeführten Jobs im UNICORE-Klienten (2.Teil) 34

Kapitel 1

Motivation

1.1 Rechnen auf verteilten Ressourcen

Bereits seit Anfang der 90er Jahre gibt es Entwicklungen, die sich mit dem Rechnen auf verteilten Computerressourcen befassen. Die Idee, mehrere an verschiedenen Orten verteilte Großrechner zu einem System zu verknüpfen, um so eine noch größere Rechenleistung für vielfältige Problemstellungen zu erhalten, faszinierte nicht nur Wissenschaftler. Man sah die Möglichkeit, sehr komplexe Aufgabenstellungen schneller und besser berechnen zu können. Nachdem vor allem durch die gewachsene Geschwindigkeit und Kapazität von Computernetzen die technischen Voraussetzungen geschaffen waren, konnten konkrete Arbeiten begonnen werden.

In diesem Zusammenhang wurde in Anlehnung an das elektrische Energienetz, das *Electrical*

Abbildung 1.1: Das GRID am Beispiel einer Sternenwarte der NASA

Power Grid, der Begriff *Computational Power Grid*, kurz *Grid* [1], geprägt.
So einfach wie man den Strom aus der Steckdose bezieht, ohne die sich dahinter verbergende Infrastruktur oder den Erzeugungsort kennen zu müssen, so einfach versuchte man Rechenleistung aus einer Art Kommunikationssteckdose zu erhalten. Die Idee des Grids ist, sämtliche zur Lösung eines aufwendigen Problems benötigten Ressourcen von verschiedensten Orten der Welt heranzuziehen und dem Anwender zur Verfügung zu stellen.

Auch wenn es noch ein weiter Weg bis zur Verwirklichung dieser Vision ist, so hat es in den vergan-

genen Jahren doch große Fortschritte in der Grid-Entwicklung gegeben. Es ist mittlerweile möglich, über Hochgeschwindigkeitsnetze verschiedenartige Rechner entfernter Institutionen und mit ihnen auch ihre Speicher- und Datenressourcen zu bündeln, um komplexe Aufgaben schnell und sicher berechnen zu können.

Es gibt zahlreiche Beispiele für wissenschaftliche Anwendungen und Simulationen, die auf einer Grid-Infrastruktur basieren. Abbildung 1.1 zeigt die Idee des Grids am Beispiel einer Sternwarte der NASA, mit der Gravitationswellen und schwarze Löcher erforscht werden sollen. Die über Parabolspiegel und Satelliten erhaltenen Daten sind an verschiedenen Standorten in unterschiedlichen Formaten gespeichert. Für die Auswertung und weitere Berechnung dieser Daten müssen Wissenschaftler auf alle benötigten Informationen zugreifen können, wo auch immer diese abgelegt sind. Zudem fließen in die Interpretation der gewonnenen Informationen bereits zuvor an verschiedenen Orten berechnete Ergebnisse aus Datenbanken ein. Zur Verarbeitung der Daten sind dann verschiedene Großrechner mit in der Regel unterschiedlichen Standorten nötig. Schließlich führt auch die persönliche weltweite Kommunikation unter den Forschern zu einer erfolgreichen Interpretation. Alle diese Ressourcen sollen im Grid gebündelt werden.

Ein weiteres Beispiel für eine Grid-Anwendung ist das in den USA laufende Projekt *NEESGrid* [2], in dem Forscher versuchen, über ein weitverzweigtes Netz von Sensoren, Beobachtungsstationen und Satelliten Erdbeben vorherzusagen. Auch hier ist es notwendig, auf alle verfügbaren Daten zugreifen zu können und verteilte Ressourcen zur Berechnung nutzen zu können.

Realisierungen solcher Anwendungen benötigen jedoch Infrastrukturen, die dem Anwender den Zugang zu einem Grid ermöglichen. Eine solche Grid-Infrastruktur stellen sowohl Globus [3] als auch UNICORE [4] zur Verfügung. Globus, das unter der Leitung vom Argonne National Laboratory (USA) entwickelt wurde, bietet dem Benutzer zahlreiche Dienste an, die flexibel zur Grid-Entwicklung eingesetzt werden können. UNICORE schafft eine integrierte Lösung, welche einheitlichen Zugriff auf verteilte Computerressourcen ermöglicht. Es wird von einem Konsortium von deutschen Universitäten, Forschungszentren und Software Firmen entwickelt. UNICORE wurde vom Bundesministerium für Bildung und Forschung (BMBF) gefördert[1].

Das Ziel der hier vorgestellten Arbeit ist es, mit UNICORE und Globus eine kombinierte Grid-Lösung zu entwickeln, die Vorteile beider Systeme vereinigt.
In dieser Lösung wird UNICORE als Klient fungieren, mit dessen Hilfe auf Globus-Ressourcen zugegriffen werden kann. Dieser Klient soll zunächst Job-Übermittlung, Statusabfragen, Datenaustausch und Ergebnissicherung ermöglichen. Ein weiterer wesentlicher Aspekt ist die Abbildung der UNICORE- auf die Globus-Sicherheitsmechanismen. Es dürfen dabei weder an Globus, noch an UNICORE Veränderungen in der Architektur vorgenommen werden.
Diese Vorgabe wird durch eine in dieser Arbeit entwickelte gemeinsame Schnittstelle erreicht.

1.2 Struktur der Arbeit

Die vorliegende Arbeit geht in Kapitel zwei und drei zunächst auf die Grundideen der UNICORE- und der Globus-Architektur in Bezug auf Job-Übermittlungstechniken und Sicherheitsmechanismen ein. Als Kern der Arbeit wird anschließend in Kapitel vier die Entwicklung der UNICORE Globus Kopplung und deren Funktionalität vorgestellt. In diesem Kapitel wird das Prinzip und die Realisierung der Job-Übermittlung von UNICORE nach Globus erklärt. Es erläutert, wie Statusabfragen, Datenaustausch und Ergebnissicherung implementiert worden sind. Im Kapitel vier wird auch auf die Abbildung der unterschiedlichen Sicherheitsmechanismen beider Architekturen mittels eines implementierten Zertifizierungsservers eingegangen.

[1] Förderungszeitraum: 1.8.1997 - 31.12.1999, Förderkennzeichen: 01 IR 703

Die Kapitel fünf und sechs schließen die Arbeit mit der Beschreibung des aktuellen Status und einem Ausblick auf die zukünftige Entwicklung der UNICORE Globus Kopplung ab.

Kapitel 2

UNICORE

2.1 Motivation und Ziele

Mit steigender Komplexität wissenschaftlicher Anwendungen haben die Nutzer von Rechenleistung vor allem Bedarf an Supercomputer-Ressourcen. Höchstleistungsrechner sind an verschiedenen Standorten in Deutschland verfügbar. Die Architektur dieser Rechner ist jedoch üblicherweise herstellerspezifisch, so daß ein Benutzer auch mit verschiedenen Zugangsmechanismen und Programmen konfrontiert wird, die er für eine effektive Nutzung der Systeme beherrschen muß. Deshalb haben die Supercomputer-Zentren gemeinsam mit den Nutzern und den Herstellern beschlossen, einen einheitlichen, sicheren und intuitiven Zugang zu den Rechnern zu entwickeln. Aus dieser Überlegung ist das Projekt UNICORE entstanden, das vom BMBF 1997 zur Förderung bewilligt wurde. Mit einem ersten Prototyp wurde die Tragfähigkeit des Konzepts 1999 nachgewiesen.

In den Projekten *UNICORE Plus*[1] und *EUROGRID*[2] werden weitere Funktionen entwickelt, die zur Produktionsreife des Systems führen sollen. Zeitgleich mit der Entwicklung von UNICORE hat sich für die von UNICORE behandelte Problemstellung der Begriff Grid-Computing (siehe Kapitel 1) etabliert.

Das UNICORE-System wird im Forschungszentrum Jülich, im Konrad-Zuse-Zentrum Berlin, im Rechenzentrum der Universität Stuttgart, beim Paderborn Center for Parallel Computing, im Leibniz Rechenzentrum in München, beim Deutschen Wetterdienst in Offenbach, dem Rechenzentrum der Universität Karlsruhe und dem Zentrum für Hochleistungsrechnen der Technischen Universität Dresden getestet und eingesetzt.

2.2 Die UNICORE Funktionen

UNICORE bietet dem Anwender eine Reihe von Funktionen, die den Zugriff auf verteilte Ressourcen vereinfachen. Dazu gehören insbesondere [5]:

- systemunabhängige Job-Beschreibung (*Abstract Job Object (AJO)*; siehe Kapitel 2.3.1)
- interaktive Erstellung eines Jobs mit einer grafischen Benutzeroberfläche (GUI) (siehe Kapitel 2.3.1),
- die Submission des Jobs zu einem oder mehreren Rechenzentren,
- volle Job-Kontrolle durch den Benutzer über die GUI,
- Statusabfragen zum Job-Ablauf,
- Ergebnissicherung,

[1] UNICORE Plus wird vom BMBF gefördert, Förderkennzeichen 01-IR-001

[2] EUROGRID wird von der Europäischen Union gefördert, Förderkennzeichen IST-1999-20247

- automatisches Speichern von Dateien (beispielsweise von Ausgabedateien),
- sicherer Zugriff auf entfernte Daten,
- einmalige Benutzerauthentisierung mit einem UNICORE-Zertifikat,
- Zertifikatabbildung auf eine existierende lokale Benutzeridentität auf dem Zielsystem,
- Benutzerautorisierung auf dem Zielsystem,
- Berücksichtigung von lokalen Sicherheitsbestimmungen des Zielsystems.

Die oben aufgeführten UNICORE-Funktionen sind Bestandteil der UNICORE Globus Kopplung und werden in den folgenden Kapiteln genauer beschrieben.

Weitere Funktionen, wie

- Ausführen voneinander abhängiger Teile eines Jobs auf mehreren Zielsystemen,
- Dateiexport vom lokalen System auf das Zielsystem,
- Dateiimport auf dem Zielsystem,
- Dateitransfer zwischen verschiedenen UNICORE-Rechenzentren und
- Unterstützung von anwendungsspezifischen Schnittstellen (*Plugins*), wie CPMD, Nastran, Fluent und Star-CD [6]

werden in der vorliegenden Arbeit nicht verwendet, da dafür Änderungen in der Globus-Architektur vorgenommen werden müßten. Diese Funktionen sind detailliert in [7] beschrieben. Mit Hilfe

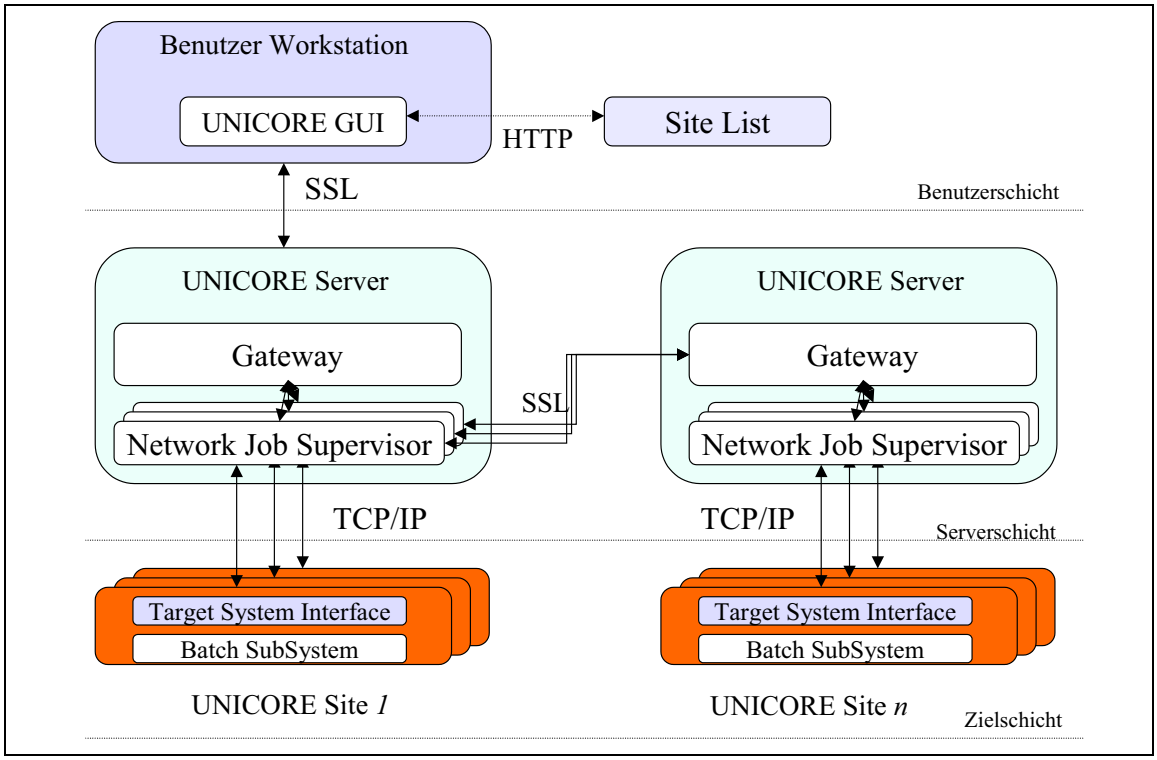

Abbildung 2.1: Die UNICORE-Architektur

der grafischen Oberfläche von UNICORE ist der Benutzer in der Lage, Jobs zu erstellen. Ein Job enthält neben dem Job-Skript mit dem Programmcode zusätzliche Informationen über benötigte Ressourcen, wie Prozessorzeit, Anzahl der Prozessoren und den angeforderten Speicherplatz. Das gewünschte Zielsystem (*Vsite*) bei einem UNICORE-Rechenzentrum (*Usite*) wird vom Benutzer ausgewählt, während UNICORE die für das Zielsystem benötigten jeweiligen Parameter automatisch hinzufügt. Ein Job kann aus mehreren Teilen bestehen, die UNICORE in einer Job-Gruppe zusammenfasst. Jeder Teil kann auf unterschiedlichen Zielsystemen abgearbeitet werden. Wenn für die Job-Erstellung entfernte Daten benötigt werden, teilt der Benutzer UNICORE den Ort mit, an

dem diese sich befinden. Die angegebenen Daten werden von UNICORE zunächst transparent für den Benutzer zum Zielsystem, auf dem die Berechnung stattfindet, transferiert.

Der Status des abgeschickten Jobs kann vom Benutzer jederzeit abgefragt werden. Wenn die Job-Berechnung beendet ist, werden die Ergebnisse und eventuell aufgetretene Fehlermeldungen an den UNICORE-Klienten gesendet.

2.3 Die UNICORE Architektur

Wie in Abbildung 2.1 [8] zu sehen ist, basiert UNICORE auf einer Dreischichtenarchitektur, bestehend aus Benutzer-, Server- und Zielschicht. Der UNICORE-Klient läuft auf dem Arbeitsplatzrechner des Benutzers, die beiden Server-Komponenten *Gateway* und *Network Job Supervisor (NJS)* auf einem dedizierten Rechner. Das *Target System Interface (TSI)* ist die Komponente, die auf dem Zielsystem die Verbindung zwischen UNICORE und dem speziellen Batch-Subsystem bzw. dem Betriebssystem herstellt.

2.3.1 Klient

Der UNICORE-Klient ermöglicht dem Benutzer, mit Hilfe einer Java basierten, grafischen Oberfläche, Jobs zu erstellen und zu verwalten. Eine Komponente des Klienten ist der *Job Preparation Agent (JPA)*. Das JPA ist das grafische Benutzerinterface von UNICORE zur Erstellung, Abstraktion und Submission der Jobs. Über den JPA kann ein Job aus

- der *Script Task* zur Verwendung bereits bestehender Job-Skripte,
- der *Compile-Link-Run Task* zum Einbinden neuer Anwendungen,
- der *Transfer Task* zum korrekten Datentransfer zwischen Job-Schritten und
- der *Job-Gruppe* zur Abarbeitung zusammenhängender Jobs

zusammengesetzt werden [9]. Ein aus mehreren Teilen bestehender Job wird durch den JPA mit Hilfe eines Abhängigkeitsgraphen zu einer Job-Gruppe zusammengefügt. Zusätzlich wählt der Benutzer mit dem JPA das gewünschte Zielsystem aus, auf dem der Job ausgeführt werden soll, läßt den von ihm erstellten Job syntaktisch prüfen, definiert Ressourcenanforderungen an das Zielsystem und submittiert den Job an ein *Gateway* (siehe Kapitel 2.3.2). Wenn ein Job erstellt worden ist, wird dieser in ein sogenanntes *Abstract Job Object (AJO)* übersetzt. Das AJO abstrahiert den Job mit einheitlichen und systemneutralen Spezifikationen, um Anwendungen auf entfernte Systeme mit unterschiedlichen Hardware- und Software-Architekturen submittieren zu können. Das AJO wird mit dem privaten Schlüssel des Benutzerzertifikats signiert. Wenn ein Job aus mehreren für verschiedene *Usites* bestimmten Teilen besteht, dann wird jeder dieser Teile in ein AJO umgewandelt und einzeln signiert. Die Adressen der *Usites* sind Bestandteil des AJO. Informationen über die verfügbaren Zielsysteme erhält der Klient aus einer auf einem zentralen Server liegenden Datei.

Der *Job Monitor Controller (JMC)* steuert als eine weitere Komponente des Klienten die Statusabfragen und die Ergebnisbearbeitung.

Nach Beendigung des Benutzerauftrags können die Ergebnisse der Berechnung und eventuell aufgetretene Fehlermeldungen über die Benutzeroberfläche angefordert werden.
Abbildung 2.2 [8] zeigt den Aufbau des UNICORE-Klienten.

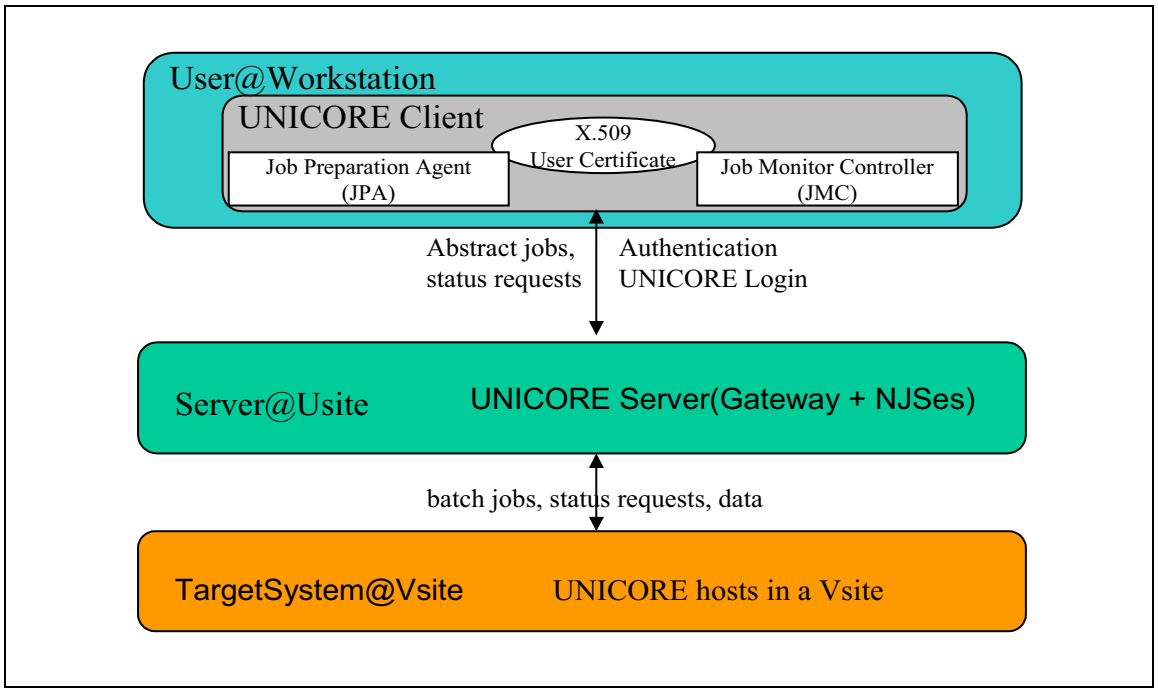

Abbildung 2.2: Der UNICORE-Klient

2.3.2 Gateway

Das *Gateway* ist ein wichtiger Bestandteil der UNICORE-*Sicherheitsarchitektur* (siehe Kapitel 2.4) und bildet zusammen mit dem im folgenden Kapitel beschriebenen *Network Job Supervisor (NJS)* den UNICORE-Server. Es hat die Aufgabe, einen Benutzer anhand seines Zertifikats zu authentisieren. Versuche, ohne ein gültiges Zertifikat auf UNICORE-Ressourcen zuzugreifen, werden bereits vom Gateway abgewiesen. Für jedes Rechenzentrum wird ein Gateway eingerichtet als einheitliches Zugangsportal zu den Zielsystemen.

Das vom Klienten erstellte AJO erreicht über eine verschlüsselte *Secure Socket Layer (SSL)*-Verbindung [10] das Gateway. Bei erfolgreicher Authentisierung des Benutzers wird das AJO über Sockets zum zuständigen NJS weitergeleitet.

2.3.3 Network Job Supervisor

Der *Network Job Supervisor (NJS)* ist die eigentliche Schnittstelle zwischen dem UNICORE-Klienten und den UNICORE-Zielsystemen. Die Aufgaben des NJS sind [9]:

- Transfer der Job-Daten von dem lokalen System in ein lokales Benutzerverzeichnis auf dem Zielsystem (*Uspace*),
- Abbildung der Benutzeridentität auf den Unix-Login-Namen des Zielsystems,
- Übersetzung der abstrakten Job-Beschreibung in einen konkreten Job für das Zielsystem (*Inkarnation*),
- Submission von Job-Teilen vom NJS zu anderen *Usites*, bzw. zu anderen NJS auf derselben *Usite*,
- Synchronisation von Jobs und Job-Teilen, damit diese in der richtigen Reihenfolge bearbeitet werden und die benötigten Ergebnisse in neue Job-Teile als Parameter einfließen.

Der NJS kommuniziert auf der einen Seite mit dem Gateway und auf der anderen Seite mit dem jeweiligen *Target System Interface (TSI)* (siehe Kapitel 2.3.4). Er erzeugt Anfragen, die vom TSI ausgeführt werden. So veranlasst der NJS zunächst, daß ein Uspace-Verzeichnis auf dem Zielsystem angelegt wird. In den Uspace werden die Job-Daten vom UNICORE-Klienten transferiert.

Wenn Job-Teile von verschiedenen Zielsystemen benötigt oder an diese verschickt werden, regelt der NJS auch die Kommunikation zwischen den jeweiligen Uspaces. Nach Beendigung des Jobs wird der Uspace im Auftrag des NJS wieder gelöscht.

Mit Hilfe der *UNICORE User Database (UUDB)* wird das systemunabhängige *UNICORE login (Ulogin)* aus dem Benutzerzertifikat auf das entsprechende lokale *Unix login (Xlogin)* abgebildet. [11].

Der NJS übersetzt unter Verwendung einer *Inkarnations Datenbank (IDB)* die abstrakte Beschrei-

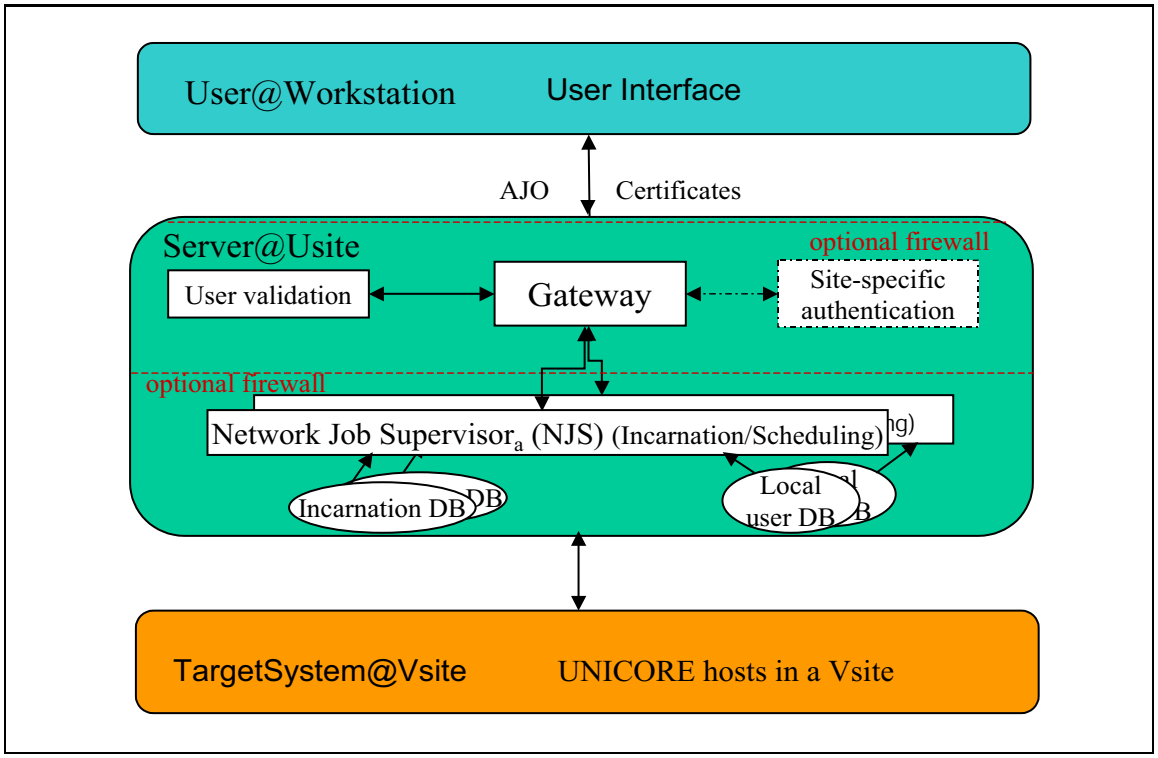

Abbildung 2.3: Der UNICORE-Server

bung des Jobs, wie sie im AJO kodiert ist, in konkrete Anweisungen für das Zielsystem. Die IDB enthält globale, die jeweilige Architektur des Zielsystems betreffende Informationen wie z.B. das verwendete Dateisystem oder die Anzahl der zur Verfügung stehenden Knoten. Nicht für das Zielsystem bestimmte AJOs werden unausgepackt an die zugehörigen Gateways weitergeleitet. Mit dem im AJO enthaltenen Abhängigkeitsgraphen synchronisiert der NJS die Job-Teile und den Datentransfer [7]. Zudem überwacht der NJS den Job-Status und gibt diesen über das Gateway an den Benutzer zurück.
Abbildung 2.3 [8] zeigt den UNICORE-Server mit Gateway und NJS.

2.3.4 Target System Interface (TSI)

Das *Target System Interface (TSI)* verbindet den NJS und das Zielsystem mit Sockets. Über diese Verbindung erhält das TSI die vom NJS übersetzten Job-Parameter. Für jeden UNICORE-Parameter stellt das TSI eine zugehörige PERL-Routine bereit, die entsprechende Anweisungen, wie Knoten- und Speichergenerierung, etc. auf dem Zielsystem ausführt. Mit jedem Job schickt der NJS den UNICORE-Benutzernamen an das TSI. Mit dieser Information legt das TSI zunächst den persönlichen Uspace in dem vom NJS mitgeteilten Verzeichnis an. Im Uspace sind sämtliche Jobs eines Benutzers und die zugehörigen Daten abgelegt. So ist eine eindeutige Zuordnung der Daten zum Benutzer gegeben. In dieses Verzeichnis übermittelt das NJS auch das Job-Skript, das den vom TSI auf dem Zielsystem auszuführenden Quelltext enthält.

Statusabfragen des Benutzers werden über das NJS zum TSI weitergeleitet. Das TSI formuliert daraufhin ein *qstat*-Kommando (ein UNIX-Befehl zur Statusabfrage) und setzt dieses auf dem lokalen Batch-Subsystem ab. Nach Beendigung des Jobs speichert das TSI die Ausgabedateien im Uspace und sendet eine entsprechende Nachricht an das NJS.

Das TSI muß für jedes UNICORE-Zielsystem vom Administrator modifiziert werden, damit es die jeweiligen systemspezifischen Anforderungen erfüllt.

Das UNICORE TSI ist eine wichtige Komponente der UNICORE–Globus–Kopplung. Als Bestandteil des *Enhanced TSI (ETSI)* (siehe Kapitel 4) wird es als Schnittstelle zwischen UNICORE und den für die Kommunikation mit Globus benötigten Programmteilen dienen. Abbildung 2.4 [8] zeigt die generelle Funktionsweise des TSI.

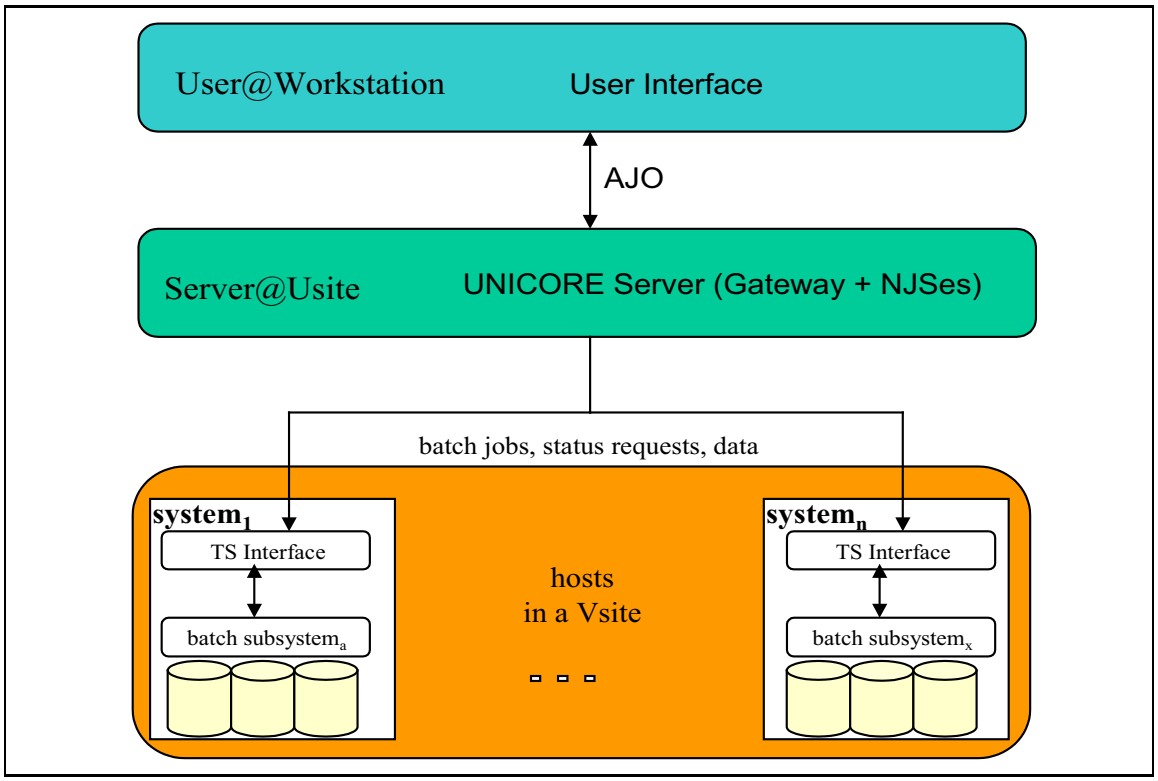

Abbildung 2.4: Das Target System Interface

2.4 Die UNICORE-Sicherheitsmechanismen

Ein wesentlicher Aspekt von UNICORE ist, für den Zugriff auf die entfernten Ressourcen einen hohen Sicherheitsstandard zu gewährleisten. Das UNICORE-Sicherheitsmodell basiert auf der *Public-Key-Infrastructure (PKI)* [12], die *X509v3-Zertifikate* [10] verwendet. Ein X509v3-Zertifikat beinhaltet:

- den Benutzernamen,
- den zugehörigen öffentlichen Schlüssel,
- die Signatur von der *Certification Authority (CA)*, die

Die Signatur der CA belegt, daß das Zertifikat von der CA ausgestellt worden ist, und bürgt für den Benutzernamen und für die Verbindung zwischen Benutzernamen und öffentlichem Schlüssel. Beim PKI-Verfahren wird ein Schlüsselpaar erzeugt, das aus einem öffentlichen und einem privaten Schlüssel besteht. Zum Verschlüsseln einer Nachricht kann der Sender den öffentliche Schlüssel

des Empfängers benutzen (nur dieser kann sie mit seinem privaten Schlüssel wieder entschlüsseln). Dadurch ist gewährleistet, daß nur der Empfänger, für den die Nachricht bestimmt ist, diese auch lesen kann. Eine zweite Möglichkeit ist es, die Daten mit dem privaten Schlüssel des Senders zu verschlüsseln und mit dem öffentlichen Schlüssel des Empfängers zu entschlüsseln. So weiß der Empfänger, daß die Nachricht wirklich von dem Sender stammt, der sich als solcher ausgegeben hat.
UNICORE benutzt die zweite Möglichkeit, um so den Benutzer eindeutig in zu authentisieren. Die öffentlichen Schlüssel aller UNICORE- Benutzer werden mit den X509v3-Zertifikaten auf einem zentralen Server abgelegt.

Durch das UNICORE-Sicherheitsmodell werden folgende wesentliche Aspekte berücksichtigt [8]:

- die Authentisierung,
- die Integrität von Jobs und Daten,
- die Autorisierung und
- Single sign-on.

Mit der Authentisierung werden UNICORE-Benutzer und -Server eindeutig identifiziert. Integrität bedeutet, daß über das Netz verschickte Inhalte, Informationen und der Absender nicht von Unbefugten verändert werden können. Die Autorisierung legt fest, welche Dienste der Benutzer auf dem Zielsystem nutzen darf. Dadurch werden die Ressourcen des Benutzers vor unberechtigten Zugriffen geschützt. Single-sign-on bedeutet, daß der UNICORE-Benutzer nur einmal zu Beginn einer Sitzung sein Passwort eingeben muss, um den privaten Privaten Schlüssel zum Signieren freizugeben. Dadurch wird der Austausch von Passwörtern über das Netz verhindert und der Benutzer braucht sich nur ein Passwort zu merken.

Die Authentisierung beider Verbindungspartner erfolgt im UNICORE über SSL. UNICORE authentisiert den Benutzer, indem das Gateway die vom Benutzer mit dem privaten Schlüssel signierten Daten mit dessen öffentlichen Schlüssel entschlüsselt. Wenn die Daten erfolgreich entschlüsselt werden konnten, ist der Benutzer von UNICORE authentisiert.

Auch die Integrität gewährleistet UNICORE über das SSL-Protokoll. UNICORE signiert jeden Teil des Jobs mit dem privaten Benutzerschlüssel und garantiert so, daß die Daten unverändert beim Gateway ankommen.

Die Autorisierung findet in der UNICORE-UUDB (siehe Kapitel 2.3.3) statt, die den öffentlichen Schlüssel des Benutzers auf den lokalen Benutzer-Account der jeweiligen UNICORE-Site abbildet. Das setzt vorraus, daß der Benutzer zu jedem Zielsystem, auf dem er seine Daten berechnen lassen möchte, einen Zugang besitzt.

Eine am Leibniz Rechenzentrum in München angesiedelte CA erzeugt und verwaltet die UNICORE-Benutzer-Zertifikate. Um ein Zertifikat zu beantragen, muß der Benutzer persönlich bei einer der *Registration Authorities* (RA), die an jedem UNICORE-Rechenzentrum eingerichtet ist, mit seinem Personalausweis vorstellig werden.

Kapitel 3

Globus

3.1 Motivation und Ziele

Etwa zeitgleich zur UNICORE-Entwicklung startete in den USA das Globus-Projekt. Globus ist aus der Kooperation verschiedener Universitäten und Forschungszentren unter der Leitung des Argonne National Laboratory und der University of Southern California hervorgegangen. Das Projektziel war, dem Benutzer Protokolle und Dienste zur Verfügung zu stellen zur

- Unterstützung von Sicherheit im Grid-Computing,
- Anforderung von verteilten Ressourcen,
- Ressourcen-Verwaltung und
- Kommunikation im Grid.

Die Protokolle sind in einer Toolbox zusammengefasst, mit deren Hilfe der Benutzer in der Lage ist, Grid-Anwendungen zu erstellen, die die Globus-Infrastruktur nutzen.

3.2 Die Globus Architektur

Wie in Abbildung 3.1 zu sehen ist, folgt Globus einem Schichtenmodell. Basierend auf grundlegenden lokalen Diensten, wie TCP/IP oder dem jeweiligen Betriebssystem, definiert Globus Kerndienste für die Entwicklung von Grid-Anwendungen. In dieser Schicht sind die zur Erstellung von Grid-Anwendungen benötigten Protokolle zusammengefaßt. Die Kerndienste von Globus sind in *Verbindungsschicht* und *Ressourcenschicht* unterteilt. Die Verbindungsschicht definiert die Sicherheitsarchitektur durch das *Grid Security Infrastructure (GSI)*-Protokoll (siehe Kapitel 3.2.3). Die Ressourcenschicht beinhaltet

- Ressource-Management-Dienste mit dem *Globus Ressource Allocation Manager (GRAM)*-Protokoll (siehe Kapitel 3.2.1)
- Informationsdienste mit dem *Metacomputing Directory Service (MDS)*-Protokoll (siehe Kapitel 3.2.1 und [13]) und die
- Datentransferdienste mit dem *Grid File Transfer Protocol (GridFTP)* und dem *Globus Access to Secondary Storage (GASS)*-Protokoll (siehe Kapitel 3.2.2).

Die Globus-Kerndienste dienen als Schnittstelle zwischen den lokalen und den globalen Diensten, die in der Regel komplexer strukturiert sind und auf die Kerndienste aufbauen. Die oberste Schicht bilden die Grid-Anwendungen. Um Grid-Anwendungen erstellen zu können, benötigt der Benutzer nur Zugriff auf die Globus-Kerndienste. Mit dem Schichtenmodell gewährleistet Globus, daß weitere globale Dienste definiert werden können ohne die zugrundeliegende Infrastruktur ändern zu müssen. In den Abschnitten 3.2.1 - 3.2.3 werden die für die UNICORE–Globus–Kopplung verwendeten Globus-Kerndiente, das GRAM-, das GASS und das GSI-Protokoll, im Detail vorgestellt.

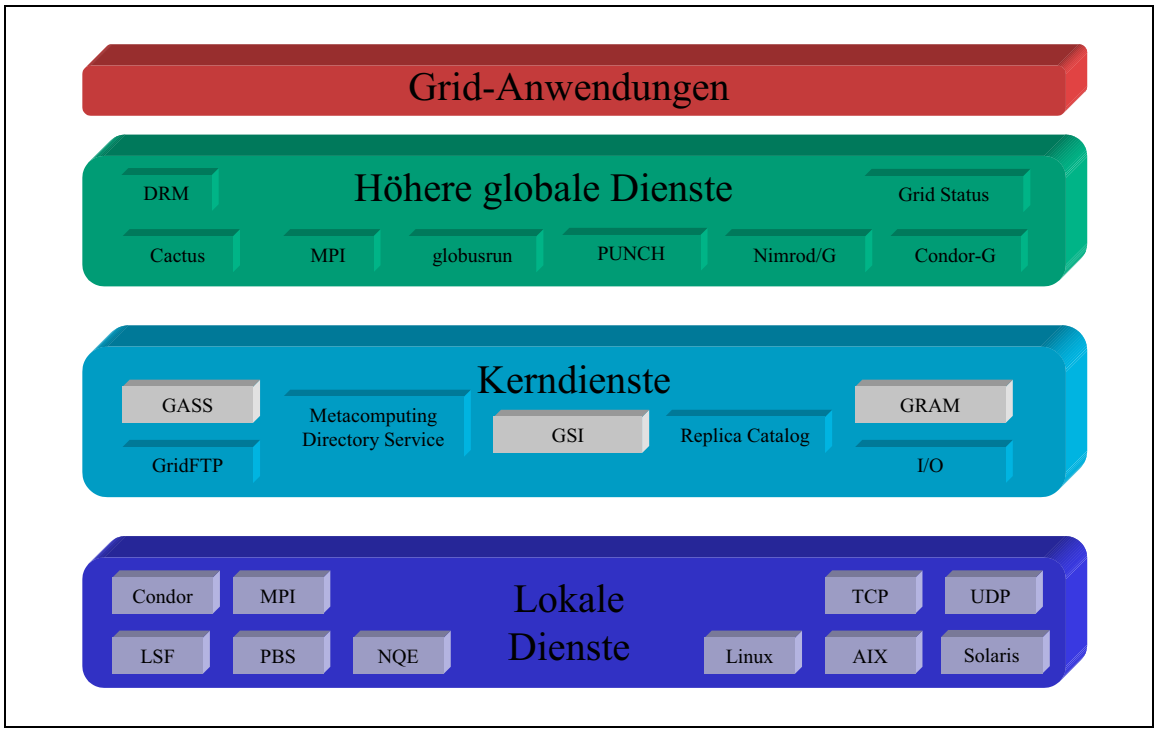

Abbildung 3.1: Das Globus-Schichtenmodell

3.2.1 Globus Resource Allocation Manager (GRAM)

Der *Globus Resource Allocation Manager (GRAM)* [14] stellt dem Benutzer eine Reihe von Routinen zur Verfügung, die folgende Aufgaben haben:

- Anforderung einzelner Ressourcen,
- Erstellen von Prozessen,
- Abstrahieren von Job-Anfragen zur systemunabhängigen Job-Submission,
- Job-Submission,
- Kontrolle des submittierten Jobs,
- Statusabfragen und
- Statusweiterleitung über Rückrufmechanismen.

Der GRAM besteht aus drei Komponenten: dem *GRAM-Klienten*, dem *GRAM-Gatekeeper* und dem *GRAM-Jobmanager*. GRAM-Gatekeeper und GRAM-Jobmanager bilden zusammen den *GRAM Server*. Der GRAM-Klient unterstützt den Benutzer bei der Prozesserstellung, dem Abstrahieren der Job-Parameter, der Job-Submission zu einem Zielsystem und der Statusabfrage. Um die benötigten Job-Parameter systemunabhängig formulieren zu können, muß der Benutzer diese mit Hilfe der *Resource Specification Language (RSL)* formulieren. Die RSL ist ein Bestandteil des GRAM-Protokolls und stellt eine allgemeine Formulierung zum Austausch von Informationen zwischen verschiedenen Komponenten zur Verfügung. Mit der RSL werden zwei Arten von Informationen formuliert: die Ressourcen-Anforderungen wie Art des Zielrechners, Anzahl der Knoten, benötigter Speicher, etc, und die Job-Konfiguration z.B die Angabe des Job-Skriptverzeichnisses und der auszuführenden Argumente.

Über den GRAM-Klienten submittiert der Benutzer den abstrahierten Job zum GRAM-Gatekeeper. Je ein GRAM-Gatekeeper ist auf jedem verfügbaren Globus-Zielsystem als Root-Prozess installiert. Nur wenn dieser auf dem Zielsystem läuft, kann auch ein Job dorthin geschickt werden. Welche Systeme jeweils verfügbar sind, entnimmt der GRAM-Klient den *Metacomputing Directory Service (MDS)*-Daten [13]. Der MDS sammelt alle verfügbaren Informationen über Ressourcen z.B deren aktuelle Erreichbarkeit oder Konfiguration. Die Informationen erhält der MDS vom *Batch-*

Subsystem der jeweiligen Ressource. Das Batch-Subsystem ist eine zielsystemspezifische Komponente, wie beispielsweise *NQE* [15] oder *Condor* [16]. Jeder GRAM-Gatekeeper ist verantwortlich für ein bestimmtes Batch-Subsystem, welches später für die Speicheranforderung und das Starten des Prozesses auf dem jeweiligen Zielsystem verantwortlich ist. Der GRAM-Gatekeeper wartet auf die Job-Submission vom GRAM-Klienten. Er authentisiert den Benutzer und bildet die Benutzeridentität auf eine systeminterne Identität ab. Dabei werden die jeweiligen Sicherheitsanforderungen des Zielsystems berücksichtigt. Der GRAM-Gatekeeper initialisiert für jede Job-Anfrage einen GRAM-Jobmanager. Der GRAM-Jobmanager prüft die vom Benutzer erzeugte RSL-Zeichenkette auf syntaktische Korrektheit und bildet diese auf die lokale Syntax ab. Anschließend wird der Job weiter vom GRAM-Jobmanager an das Batch-Subsystem des jeweiligen Zielsystems geleitet.

Nach der Submission an das Batch-Subsystem pollt der GRAM-Job-Manager den Status des jeweiligen Prozesses. Über einen Rückrufmechanismus (siehe Kapitel 4.4) kommuniziert der GRAM-Jobmanager mit dem GRAM-Klienten und schickt bei Statusänderungen die entsprechende Mitteilung an den GRAM-Klienten. Der Benutzer kann über diese Kommunikationsverbindung den Job im GRAM-Jobmanager abbrechen. Bei Beendigung der Job-Ausführung auf dem Zielsystem wird auch der GRAM-Jobmanager beendet. Abbildung 3.2 zeigt die GRAM-Architektur.

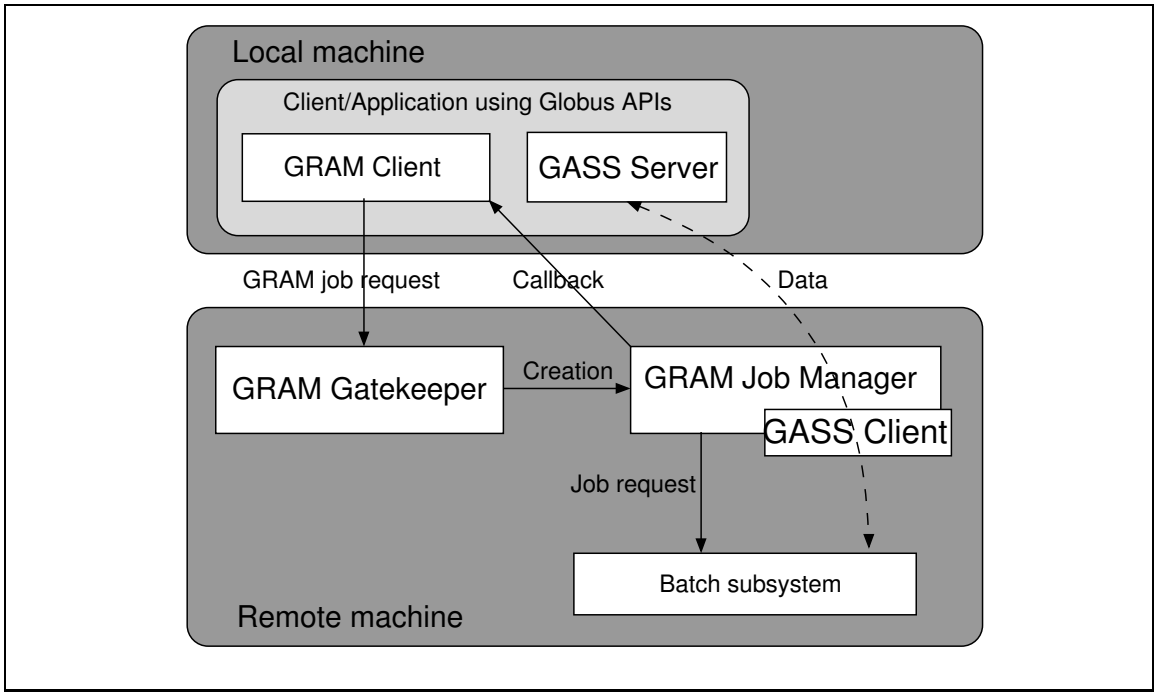

Abbildung 3.2: Die GRAM-Architektur

3.2.2 Globus Access to Secondary Storage (GASS)

Die *Globus Access to Secondary Storage (GASS)*-Komponente [17] wird von GRAM verwendet, um Dateien und auszuführende Programme zu entfernten Systemen zu transferieren und auf entfernte Dateien zuzugreifen. Dafür verwendet GASS ein sogenanntes *Datei-Caching*: ein Speicherbereich (*Cache*) wird angegeben, in dem Kopien von entfernten Dateien abgelegt werden können. Wenn eine entfernte Datei zum Lesen oder Schreiben geöffnet wird, dann wird diese vollständig in den lokalen Speicherbereich kopiert. Die lokale Kopie wird anstelle der entfernten Datei geöffnet und die entsprechenden Aktionen werden ausgeführt. Eine so veränderte Datei wird zurück in den Speicherbereich des entfernten Systems geschrieben. Dieses Vorgehen bleibt dem Benutzer verborgen.

GASS ersetzt die herkömmlichen UNIX-Dateideskriptoren *open* und *close* durch eigene Routinen. Der einzige Unterschied zu den UNIX-Kommandos ist, daß GASS den Routinen eine URL[1] anstatt eines Dateinamens als Parameter übergibt. Die URL enthält

- die Adresse des Rechners, auf dem der GASS-Server läuft,
- den Ort des lokalen Speicherbereichs, den der GASS-Server verwaltet
- und das Protokoll, mit dem auf den Speicherbereich zugegriffen werden kann.

Das GASS-Modul besteht aus dem GASS-Server und dem GASS-Klienten. Der GASS-Server wird vom Benutzer mit dem *globus_module_activate(gass)*-Kommando auf dem lokalen System gestartet. Mit dem zu submittierenden Job wird die GASS-Server-URL zum GRAM-Gatekeeper und von dort aus weiter zum GRAM-Jobmanager gesendet, der den GASS-Klienten startet. Mit Hilfe der GASS-Server-URL ist der GASS-Klient in der Lage, Dateien von oder auf das entfernte System (aus Benutzersicht auf das lokale System) zu kopieren. So werden beispielsweise Ausgabe- und Fehlerdateien eines submittierten Jobs von dem Batch-Subsystem des Zielsystems über den GASS-Klienten zum GASS-Server geschickt. Die Kommunikation zwischen GASS-Server und GASS-Klient findet über eine sichere SSL-Verbindung statt, auf der das *Grid Security Infrastructure (GSI)-Protokoll* (siehe Kapitel 3.2.3) definiert ist.

3.2.3 Globus-Sicherheit

Das Globus-Sicherheitsmodell verwendet die *Grid Security Infrastructure (GSI)* [18]. Die GSI arbeitet wie UNICORE mit einer auf X509v3-Zertifikaten basierenden Public-Key-Technologie. Zunächst erzeugt der Benutzer ein Schlüsselpaar, bestehend aus öffentlichen Schlüssel (*Public Key*) und privaten Schlüssel (*Private Key*). Der private Schlüssel wird mit einem Passwort versehen auf dem lokalen Rechner gespeichert, um ihn vor unberechtigtem Zugriff zu schützen. Um ein Zertifikat zu beantragen, schickt der Benutzer die Zertifikatanfrage, die u.a. den öffentlichen Schlüssel enthält, per Mail an eine Globus *Certification Authority (CA)*. Globus akzeptiert Zertifikate, die von einer beliebigen, vom Administrator eines Globus-Servers registrierten CA erstellt worden sind. Die CA signiert mit entsprechenden Routinen automatisch das Zertifikat und sendet es per Mail zurück zum Benutzer. Globus verlangt im Gegensatz zu UNICORE keine persönliche Benutzeridentifizierung.

Wesentliche Aspekte der GSI sind

- gegenseitige Authentisierung,
- Single-sign-on und
- Autorisierung.

Gegenseitige Authentisierung bedeutet, daß zwei Kommunikationspartner die Identität des jeweils anderen überprüfen. Zur gegenseitigen Authentisierung benutzt die GSI das *Secure Socket Layer (SSL)-Handshake-Protokoll* [10] unter Verwendung von sogenannten *Proxies*. Ein Proxy wird vom Benutzer im GRAM-Klienten mit der Globus-Routine `grid-proxy-init` lokal erstellt und besteht aus einem temporären *Proxy-Zertifikat* (mit einem neuen öffentlichen Schlüssel) und einem temporären neuen privaten Schlüssel. Das Proxy-Zertifikat beinhaltet die Benutzeridentität mit dem Zusatz *proxy*, um anzuzeigen, daß es sich um ein Proxy-Zertifikat handelt und erlaubt Prozessen, im Auftrag des Benutzers Ressourcen anzufordern. Das Proxy-Zertifikat wird mit dem Globus-Benutzerzertifikat und dem privaten Benutzerschlüssel signiert. Dazu muß der Benutzer einmal das Passwort, das den privaten Benutzerschlüssel schützt, eingeben. Proxy-Zertifikate haben eine begrenzte Lebensdauer von standardmäßig acht Stunden.

Der private Schlüssel des Proxies muß lokal gespeichert und darf nicht über das Netz verschickt werden, um die Sicherheit zu gewährleisten. Da das Proxy jedoch nur begrenzt gültig ist, kann der private Proxy-Schlüssel im Gegensatz zu dem privaten Benutzerschlüssel ohne Passwort abgelegt

[1] Uniform Resource Locator (URL)

werden. So kann der private Schlüssel des Proxies zur gegenseitigen Authentisierung vom Benutzer und GRAM-Gatekeeper verwendet werden, ohne daß der Benutzer ein weiteres Mal sein Passwort einzugeben braucht.

Das SSL-Handshake funktioniert wie folgt: Der GRAM-Klient schickt das Proxy-Zertifikat zum GRAM-Gatekeeper. Dieser verwendet den öffentlichen Schlüssel des Benutzers, um die Signatur im Proxy-Zertifikat zu bestätigen. Anschließend wird der öffentliche Schlüssel der CA benötigt, um die Signatur im Benutzerzertifikat zu bestätigen. So wird eine Kette von vertrauenswürdigen Zertifikaten von der CA bis zum Proxy gebildet.

Zudem muß der GRAM-Gatekeeper verifizieren, daß der Benutzer mit dem im Proxy-Zertifikat angegebenen Benutzer wirklich übereinstimmt. Dafür generiert der GRAM-Gatekeeper eine zufällige Zeichenkette und schickt diese zu dem GRAM-Klienten, der die Zeichenkette mit dem privaten Benutzerschlüssel verschlüsselt. Die so veränderte Nachricht wird an den GRAM-Gatekeeper zurückgesendet. Wenn die Zeichenkette dort mit dem öffentlichen Schlüssel des Benutzers wieder entschlüsselt werden kann,dann ist die Benutzeridentität korrekt.
Umgekehrt authentisiert sich der GRAM-Gatekeeper beim GRAM-Klienten: Der GRAM-Gatekeeper schickt dem GRAM-Klienten das Server-Zertifikat. Der GRAM-Klient prüft das Zertifikat und schickt eine Zeichenkette zum GRAM-Gatekeeper. Dieser verschlüsselt die Zeichenkette und sendet sie zurück zum GRAM-Klienten. Wenn der GRAM-Klient in der Lage ist, die Zeichenkette wieder zu entschlüsseln, dann ist die gegenseitige Authentisierung zwischen GRAM-Klient und GRAM-Gatekeeper erfolgreich abgeschlossen.

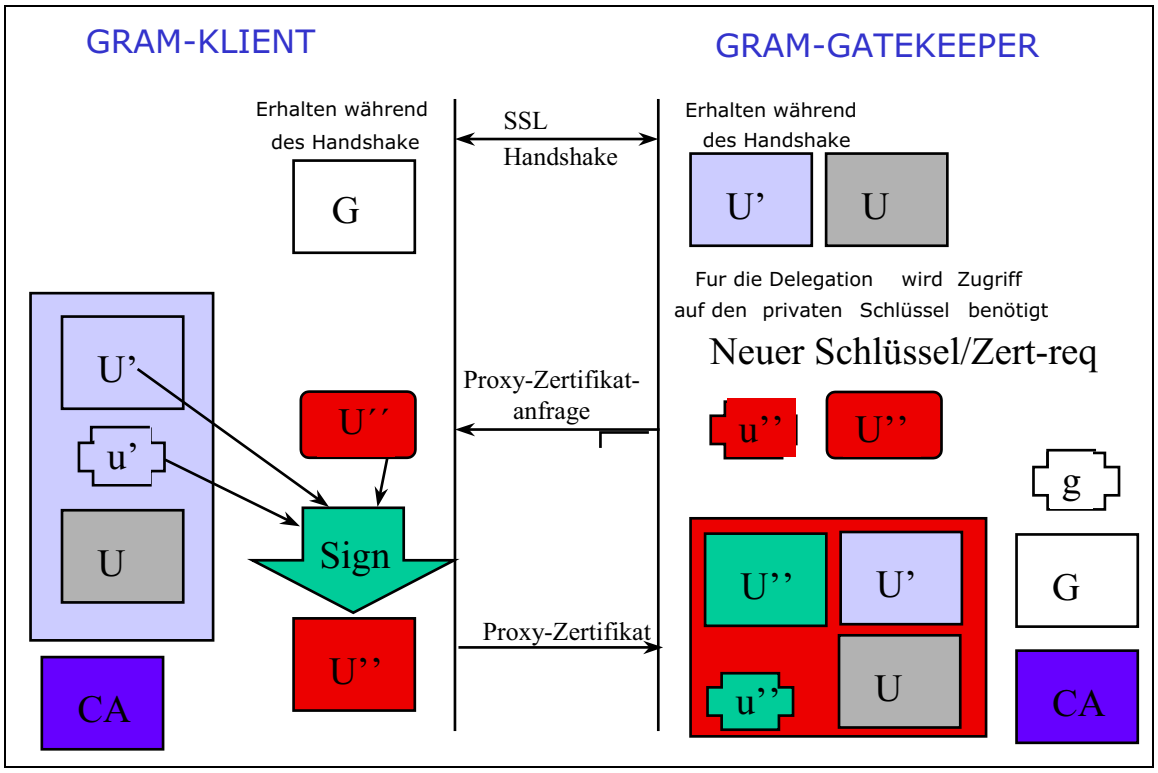

Abbildung 3.3: Das Globus-Delegationsprinzip

Nach der Authentisierung des Benutzers mit dem GRAM-Gatekeeper wendet Globus die sogenannte *Proxy-Delegation* an. Die vom GSI unterstützte Delegation ist eine Erweiterung des standardisierten SSL-Protokolls. Ziel der Delegation ist, die Benutzeridentität auf den GRAM-Gatekeeper abzubilden, bei dem sich der Benutzer zuvor authentisiert hat. Dadurch, daß der GRAM-Gatekeeper die Identität des Benutzers annimmt, kann der Benutzer Grid-Dienste anfordern oder seinen Job zu mehreren GRAM-Gatekeepern senden, ohne Passwörter oder private Schlüssel über das Netz

verschicken zu müssen. Die Delegation wird durch die Globus-Proxies ermöglicht. Das Delegationsprinzip wird beispielsweise auch in *Kerberos* [19] verwendet. Kerberos bietet ebenfalls eine Sicherheitsarchitektur, die dem Anwender *Tickets* anstelle von Proxies ausstellt. Kerberos hat jedoch den Nachteil, daß SSL nicht unterstützt wird.

Das Prinzip der Delegation funktioniert folgendermaßen: Da für die Delegation Zugriff auf den privaten Schlüssel notwendig ist, erzeugt der GRAM-Gatekeeper eines entfernten Systems ein zweites Proxy für den Benutzer. Dieses besteht aus einem neuen privaten Schlüssel, der in diesem Fall auf dem Gatekeeper-Rechner sicher gespeichert wird, und einem neuen noch nicht signierten Proxy-Zertifikat. Der GRAM-Gatekeeper submittiert die Proxy-Zertifikatanfrage an den GRAM-Klienten. Der GRAM-Klient signiert das neue Proxy-Zertifikat mit dem vom GRAM-Klienten zuvor erzeugten Proxy-Zertifikat und seinem privaten Proxy-Schlüssel. Anschließend wird das signierte, neue Proxy-Zertifikat zurück zum Gatekeeper gesendet. Dieses Proxy-Zertifikat enthält ebenfalls die Identität des Benutzers.

Das Prinzip der Delegation ist in Abbildung 3.3 dargestellt. *U* charakterisiert das Benutzerzertifikat, *U'* das vom GRAM-Klienten erzeugte Proxy-Zertifikat mit dem privaten Proxy-Schlüssel *u'*. *G* und *g* sind das Zertifikat und der private Schlüssel des GRAM-Gatekeepers. *U"* und *u"* stellen das vom GRAM-Gatekeeper erzeugte neue Proxy-Zertifikat und den neuen privaten Proxy-Schlüssel dar. Durch die Delegation in Verbindung mit den Proxies wird in Globus das Single-sign-on erreicht.

Bei der Autorisierung verfolgt Globus ein ähnliches Prinzip wie UNICORE. Jeder GRAM-Gatekeeper verwaltet eine Datenbank, den sogenannten *gridmap file*, der die Abbildung von der globalen auf die lokale Benutzerkennung des Zielsystems vornimmt und damit die für den Benutzer verfügbaren Anwendungen festlegt. Globus bezeichnet die globale Benutzerkennung als *Distinguish Name*. Für einen Distinguish Name verlangt Globus mindestens die Angabe des Herkunftslandes, des Projekts und des Benutzernamens. Zusätzliche Angaben, wie der Name der Firma oder der Abteilung, sind optional. Der Inhalt einer solchen Abbildungstabelle kann wie in Abbildung 3.4 dargestellt aussehen [20].

Distinguish Name	**lokaler Benutzername**
"/C=DE/O=UNIGlo/CN=Michael Rambadt"	rambadt
"/C=US/O=Globus/CN=Ian Foster"	itf
"/C=DE/O=UNICORE/CN=Dietmar Erwin"	zam123
"/C=US/O=Globus/CN=Carl Kesselman"	u14543

Abbildung 3.4: Abbildung einer globalen auf eine lokale Benutzerkennung im GRAM-Gatekeeper

Die Abbildung des Distinguish Name auf den lokalen Benutzernamen setzt voraus, daß für den Benutzer bereits ein Account auf dem jeweiligen System existiert.

Kapitel 4

Die UNICORE Globus Kopplung

Aus den vorangegangenen beiden Kapiteln, die die Architektur von UNICORE und Globus detailliert beschreiben, ergeben sich Probleme für die Kopplung beider Systeme. Auch wenn die Zielsetzungen von UNICORE und Globus oftmals gleich sind, unterscheidet sich die Art, wie diese Ziele erreicht werden, erheblich voneinander.
Die für die Kopplung von UNICORE und Globus wesentlichen Unterschiede sollen hier noch einmal zusammenfassend verdeutlicht werden:

UNICORE abstrahiert den Job mit einem Java basierten *Abstract Job Object (AJO)* (siehe Abschnitt 2.3.1), Globus dagegen mit der *Ressource Specification Language (RSL)* (siehe Kapitel 3.2.1). Globus versteht die AJO-Darstellung von UNICORE nicht, während UNICORE die Globus-RSL nicht interpretieren kann. Für die Job-Übermittlung von UNICORE nach Globus ist es jedoch notwendig, daß die UNICORE Job-Parameter in die Globus-RSL übersetzt werden, und später Statusrückgaben von Globus in UNICORE verstanden werden.

Ein weiterer wesentlicher Aspekt, der bei der Kopplung von UNICORE und Globus beachtet werden muß, ergibt sich aus den unterschiedlichen Sicherheitstechniken beider Systeme, die aufeinander abgebildet werden müssen.

Die oben aufgezählten grundsätzlichen Unterschiede zwischen UNICORE und Globus führen zu der in den folgenden Kapiteln beschriebenen Implementation der gemeinsamen Schnittstelle.

4.1 Zielsetzung

Für die UNICORE Globus Kopplung sind folgende grundlegende Entwicklungen vorzunehmen:

- Die Übersetzung von UNICORE Aufgaben, wie
  - Job-Übermittlung,
  - Statusabfragen und
  - Ergebnissicherung

  in die entsprechenden Globus-Darstellungen und
- die Abbildung von UNICORE Benutzerzertifikaten auf temporäre Globus-Zertifikate.

Diese Funktionalitäten sollen ohne Änderungen in der jeweiligen Architektur erreicht werden, um Beeinflussungen in laufenden Projektentwicklungen von UNICORE und Globus zu vermeiden.

Die oben formulierten Anforderungen werden durch eine Erweiterung des UNICORE *Target System Interface (TSI)* (siehe Kapitel 2.3.4) erreicht. Das daraus resultierende ***E**nhanced **T**arget **S**ystem **I**nterface (ETSI)* ist die Schnittstelle zwischen UNICORE und Globus und wird von UNICORE wie

eines von vielen Zielsystemen behandelt, während es für Globus die Rolle eines Klienten annimmt. Das ETSI besteht aus vier Hauptkomponenten:

- *Job Request Interface (JRI)*,
- *Certification Authority Interface (CAI)*,
- *Proxy Certification Authority (Proxy-CA)* und
- *Target System Interface (TSI)*.

Die ersten drei Komponenten wurden in dieser Arbeit vollständig entwickelt, während beim schon vorhandenen TSI Erweiterungen vorgenommen werden mußten.

Die Funktionalität des Enhanced Target System Interface ist in Abbildung 4.1 verdeutlicht und wird in den folgenden Kapiteln erläutert.

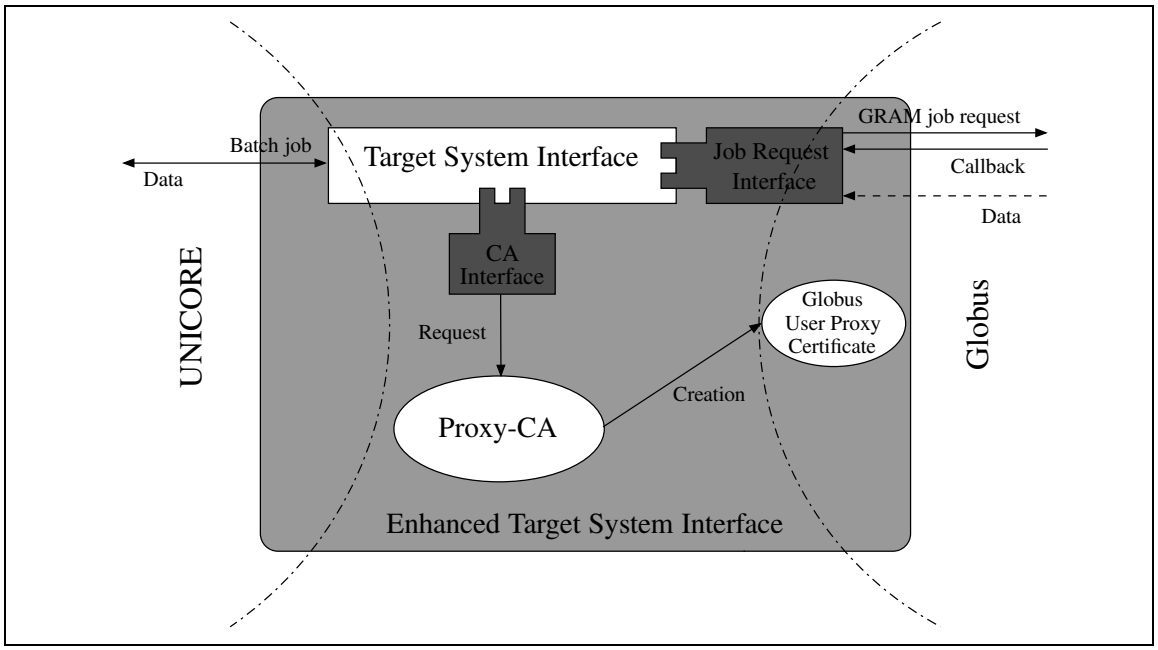

Abbildung 4.1: Enhanced Target System Interface (ETSI)

4.2 Das Prinzip der UNICORE -Globus Job-Übermittlung

Die Job-Übermittlung von UNICORE nach Globus wurde wie folgt gelöst: Der Benutzer erzeugt mit Hilfe des UNICORE-Klienten einen UNICORE-Job und wählt als *Usite* (siehe Kapitel 2.2 ein Globus-Zielsystem aus, auf dem der Job verarbeitet werden soll. Der als AJO formulierte Job wird zunächst an das Gateway gesendet und von dort zum Network Job Supervisor (NJS). Das NJS übersetzt das AJO in systemspezifische Anforderungen, die als Parameter an das Target System Interface (TSI) weitergeleitet werden. Bis zu diesem Zeitpunkt durchläuft der Job die Standardroutinen von UNICORE. Das TSI kommuniziert im folgenden jedoch nicht mit einem lokalen Batch-Subsystem, wie in UNICORE, sondern mit dem *Certificaton Authority Interface (CAI)* und dem *Job Request Interface (JRI)*. Das CAI ist für die Abbildung des UNICORE Benutzernamens auf ein Globus Zertifikat verantwortlich, während das JRI den Datenaustausch zwischen UNICORE und Globus ermöglicht und somit auch für die Job-Submission vom TSI zu einem Globus Zielsystem verantwortlich ist. In den Kapiteln 4.3 und 4.4.3 wird der in dieser Arbeit entwickelte Zertifizierungsmechanismus detailliert beschrieben.

Nachdem der UNICORE-Job das JRI erreicht hat, werden die zuvor vom NJS aus dem AJO aus-

gepackten UNICORE Job-Parameter erneut (diesmal mit der Globus-Darstellung) abstrahiert, um so nach Globus submittiert werden zu können. Dazu werden die UNICORE Job-Parameter auf die Globus *Ressource Specification Language (RSL)* (siehe Kapitel 3.2.1) abgebildet. Dadurch ist Globus in der Lage, den ursprünglichen UNICORE Job richtig zu interpretieren. Wie die Übersetzung konkret gelöst worden ist, wird im Abschnitt 4.4.2 beschrieben.
Es würde im Prinzip ausreichen, den UNICORE-Job direkt mit der Globus-RSL zu abstrahieren. Da aber keine Änderungen in der Architektur von Globus und UNICORE vorgenommen werden dürfen, muß in diesem Fall der Umweg über die Abstrahierung durch das AJO und zusätzlich durch die RSL in Kauf genommen werden.

Der so aus dem UNICORE-Job erstellte Globus-Job wird zu einem GRAM-Gatekeeper gesendet, der für die Weiterleitung des Jobs zu einem Batch-Subsystem sorgt (siehe Kapitel 3.2.1). Dort wird der Job verarbeitet und die Ergebnisse anschließend zurück zum JRI geschickt, das diese an UNICORE weiterleitet. Mit Hilfe der grafischen Oberfläche von UNICORE ist der Benutzer in der Lage, auf die Ergebnisse zuzugreifen, und diese weiter zu verarbeiten.

4.3 Der UNICORE - Globus Zertifizierungsmechanismus

UNICORE und Globus bieten dem Benutzer sicheren Zugriff auf entfernte Ressourcen. Die Sicherheitskonzepte von UNICORE und Globus basieren zwar beide auf einer standardisierten Public-Key-Technologie unter Verwendung von X509v3 Zertifikaten, die Authentisierungsmechanismen der Systeme unterscheiden sich jedoch grundsätzlich. UNICORE signiert jeden Teil eines Jobs mit dem Benutzer-Zertifikat. Dieses Vorgehen garantiert die Integrität des Jobs und identifiziert den Benutzer, der den Job zu einem Zielsystem sendet (siehe Kapitel 2.4). Globus hingegen arbeitet mit Proxy-Delegation (siehe Kapitel 3.2.3).

Für die UNICORE Globus Kopplung muss es möglich sein, ein Globus Zertifikat für einen UNICORE Benutzer automatisch zu erstellen. Globus bietet hierzu die Verwendung der *MyProxy*-Bibliothek [21] an. Die MyProxy-Software erlaubt Globus-Benutzern, die keinen Zugriff auf die Globus *Grid-Sicherheits-Infrastruktur (GSI)* (siehe Kapitel 3.2.3) haben, dennoch Globus-Ressourcen zu nutzen. Über einen MyProxy-Server, auf dem die GSI installiert ist, werden für den Benutzer die benötigten Proxies erstellt. Damit wird zwar die Forderung, Globus Proxies für einen UNICORE Benutzer zu generieren, erfüllt, jedoch bedingt die Einbindung der MyProxy-Software Änderungen im UNICORE Klienten. Da aber eine der Anforderungen dieser Arbeit ist, keine Veränderungen an den bestehenden Architekturen vorzunehmen, kann die Myproxy-Software nicht zur Lösung verwendet werden.

In dieser Arbeit wurde eine andere Lösung entwickelt, ohne auf die grundlegenden Sicherheitsanforderungen, wie Authentisierung, Integrität und Autorisierung zu verzichten. Dieser Mechanismus basiert auf der Implementation eines eigenen Zertifizierungsservers. Dazu wird der *Proxy-CA-Server* zur Erstellung und Verwaltung von Globus Benutzerzertifikaten und Globus Proxies (siehe 3.2.3) auf einem dedizierten System eingerichtet. Die Kommunikation zwischen TSI und Proxy-CA erfolgt über die CAI-Komponente. Eine detailierte Beschreibung der Funktionalität der für die UNICORE Globus Kopplung verwendeten Zertifizierung wird in Kapitel 4.4.3 gegeben.

4.4 Die Implementation

Die UNICORE Globus Kopplung basiert auf den zum Zeitpunkt der Implementation aktuellen Versionen von UNICORE (Version 3.5) und Globus (Version 1.7). Für die Entwicklung des Enhanced TSI sind folgende programmiertechnischen Entscheidungen getroffen worden:

- Die Software wird in PERL und in C geschrieben, da das UNICORE TSI in PERL, die Globus Schnittstellen jedoch in C programmiert sind.
- Der Datenaustausch zwischen JRI, CAI, TSI und Proxy-CA wird über Sockets implementiert, da sowohl die UNICORE als auch die Globus Komponenten intern über Sockets kommunizieren.

Weder in der UNICORE noch in der Globus Architektur sollen Veränderungen vorgenommen werden. Eine Ausnahme stellt das UNICORE TSI dar, das in UNICORE für jedes Zielsystem verändert werden muß (siehe Kapitel 2.3.4). Somit bietet es sich an, das UNICORE TSI auch in der UNICORE Globus Kopplung für die Entwicklung des Enhanced Target System Interface zu erweitern.

Alle in dieser Arbeit programmierten bzw. zu verändernden Funktionen sind deswegen Bestandteil des implementierten Enhanced Target System Interface.

Die Funktionalität der UNICORE Globus Kopplung basiert auf den in Abbildung 4.2 beschriebenen Komponenten und den zugehörigen Modulen. Die Abhängigkeiten der Module und Funktionen untereinander zeigt Abbildung 4.3.
Die Funktionsweise der einzelnen Module wird in den folgenden Kapiteln beschrieben.

Komponente	Modul	Funktion
TSI	uniglo	Schnittstelle zwischen UNICORE, CAI und JRI
JRI	job_mapping	wandelt den UNICORE Job in eine Globus RSL um
	file_manager	regelt den Dateitransfer zwischen UNICORE und Globus
	job_submission	Übermittlung eines Jobs vom ETSI nach Globus
	log_messages	protokolliert alle auftretenden Meldungen und Fehler
CAI	connect_to_ca	Schnittstelle zwischen TSI und Proxy-CA
Proxy-CA	proxy_ca	erzeugt und verwaltet ein Globus Benutzerzertifikat und ein Globus Proxy-Zertifikat

Abbildung 4.2: Implementierte und modifizierte Module des ETSI

4.4.1 Erweiterungen im UNICORE TSI

Das TSI muß in UNICORE für das jeweilige Zielsystem modifiziert werden. Auch in der UNICORE Globus Kopplung werden im UNICORE TSI notwendige Änderungen vorgenommen. Die bereits im UNICORE TSI implementierten PERL-Funktionen dienen als Schnittstelle zu UNICORE. Das C-Modul *uniglo* baut die Schnittstellen zum JRI und zum CAI auf. Das UNICORE TSI und die Erweiterung durch das Modul uniglo kommunizieren über Sockets miteinander. Dadurch wird der Datenaustausch zwischen den PERL- und den C-Routinen erreicht, und die hohe Portabilität des ETSI auf andere Maschinen gewährleistet. Beim Start des Enhanced TSI wird zunächst eine Socket-Verbindung zwischen TSI und NJS aufgebaut. Das Target System Interface erwartet UNICORE Benutzer-Anfragen, wie `submit`, `status`, oder `retrieve output`. Bei jeder `submit`-Anfrage wird zunächst die CAI-Komponente aktiviert (siehe Kapitel 4.4.3). Erst nachdem das CAI seine Aufgaben abgearbeitet hat, startet uniglo das JRI für die Job-Übermittlung nach Globus. So wird erreicht, daß Globus von UNICORE wie eines von vielen UNICORE Zielsystemen angesehen wird.

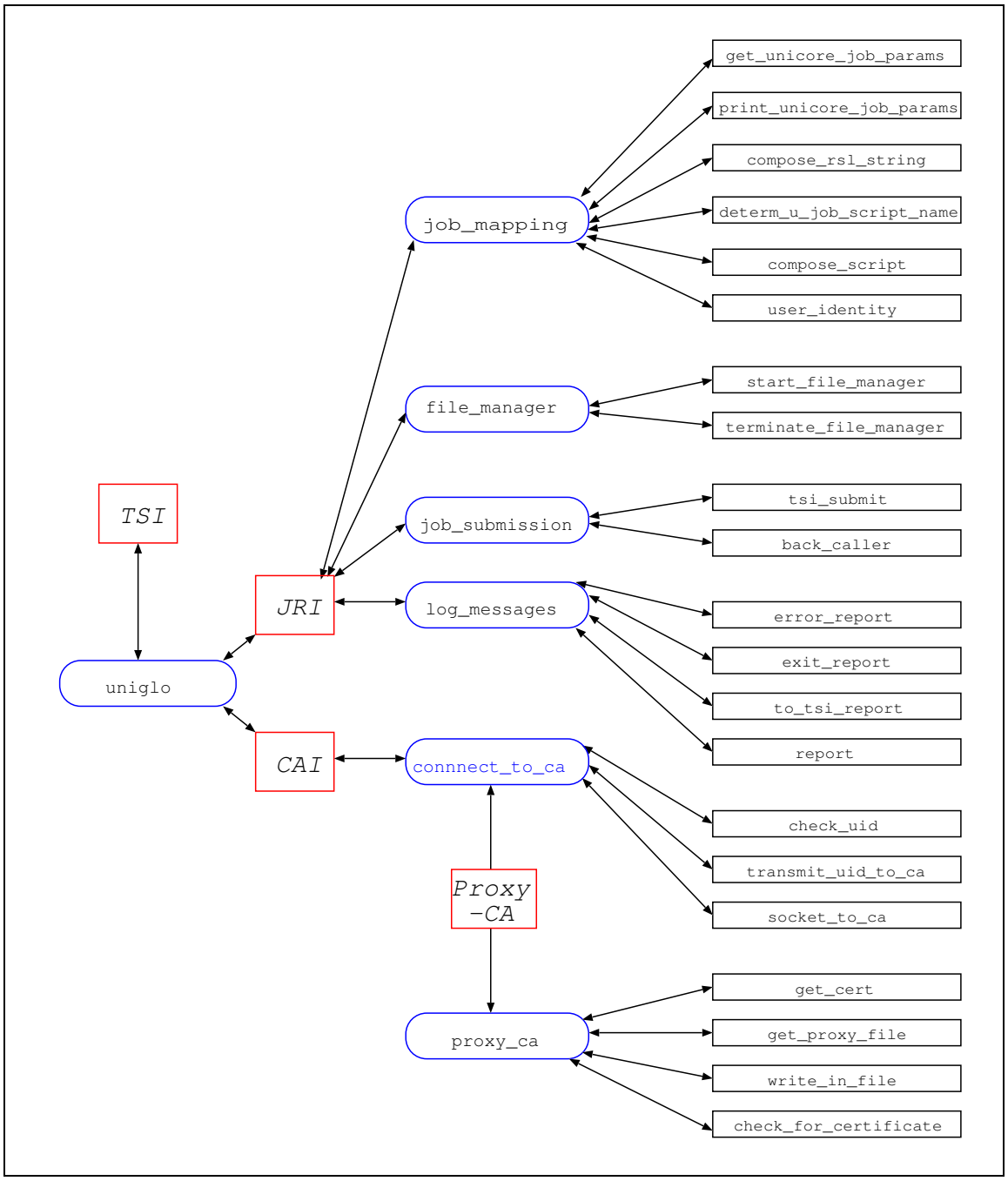

Abbildung 4.3: Funktionsabhängigkeitsdiagramm

4.4.2 Job Request Interface (JRI)

Das Job Request Interface ist die Schnittstelle zwischen TSI und Globus und implementiert

- die Übersetzung eines UNICORE Job-Formats in die RSL-Notation,
- die Job-Übermittlung zu einem Gram-Gatekeeper,
- die Abfrage, Weiterleitung und Rückgabe des Jobstatus,
- die Ergebnissicherung nach Beendigung der Job-Berechnung.

Beim Eintreffen einer `submit`-Anfrage für ein Globus Zielsystem leitet das TSI diese zusammen mit den UNICORE Job-Parametern über das uniglo-Modul zum JRI weiter. Das `submit`-Kommando bewirkt im JRI

- die Abbildung von UNICORE Parametern auf die RSL,

- die Initialisierung des *Globus Access to Secondary Storage (GASS)-Servers* und des *Globus Resource Allocation Manager (GRAM)-Klienten* (siehe Kapitel 3.2.2 und 3.2.1), sowie
- die Einrichtung eines Rückrufmechanismus, über den Globus mit dem JRI kommunizieren kann (siehe Kapitel 3.2.1).

Die für die Umwandlung in die RSL erforderlichen Funktionen sind in dem Modul *job_mapping* zusammengefasst. Zunächst extrahiert die Funktion `get_unicore_job_params` sämtliche UNICORE Parameter aus dem vom NJS ausgepackten AJO und speichert diese mit dem zugehörigen Inhalt mit `print_unicore_job_params` im JRI. Für die UNICORE Globus Kopplung werden die in Abbildung 4.4 aufgeführten UNICORE Parameter benötigt:

UNICORE Parameter	**Funktion**
#TSI_IDENTITY	enthält den UNICORE Benutzernamen
#TSI_SUBMIT	enthält die Submissionsanfrage für das Zielsystem
#TSI_USPACE_DIR	enthält den Pfad des Jobs
#TSI_JOBNAME	enthält den Job-Namen
#TSI_OUTCOME_DIR	enthält das Verzeichnis für die Ausgabe und Fehlermeldungen

Abbildung 4.4: Verwendete UNICORE Parameter und deren Funktion

Die übrigen UNICORE Parameter werden zwar nicht für diese Arbeit benötigt, können aberfür eine spätere Erweiterung der UNICORE Globus Kopplung verwendet werden.

Abbildung 4.5 zeigt die zur Job-Übermittlung notwendigen Globus RSL-Parameter.

RSL-Parameter	**Funktion**
("executable")	enthält den Pfad der Globus Shell zur Job-Ausführung
("rsl_substitution")	enthält die URL-Adresse des GASS-Servers und einen freien Port
("arguments")	enthält den Pfad und den Namen des Job-Skripts
("stdout")	enthält den Pfad für das Ausgabeverzeichnis
("stderr")	enthält den Pfad für das Fehlerausgabeverzeichnis

Abbildung 4.5: Verwendete RSL-Parameter und deren Funktion

Die Funktion `compose_rsl_string` aus dem Modul job_mapping füllt die zuvor beschriebenen RSL-Parameter mit dem von UNICORE im *Abstract Job Object (AJO)* (siehe Kapitel 2.3.1) übermittelten Inhalt. Um eine eindeutige Zuordnung zwischen dem Globus Zielsystem und Job Request Interface herzustellen, wird in `rsl_substitution` die URL-Adresse des im JRI zu startenden GASS-Servers angegeben (siehe Kapitel 3.2.2). Die RSL-Parameter `rsl_substitution` und `executable` werden unabhängig von UNICORE erzeugt, während die übrigen aus den entsprechenden Parametern TSI_JOBNAME, TSI_OUTCOME_DIR und TSI_USPACE_DIR gewonnen werden. TSI_JOBNAME beinhaltet den vom Benutzer gewählten Skriptnamen. UNICORE erweitert diesen Namen intern mit einer Zahlenkombination, um Eindeutigkeit zu gewährleisten. An das TSI wird jedoch nur der nicht erweiterte Dateiname weitergegeben. Den vollständigen Skriptnamen ermittelt die Funktion *determ_u_job_script_name* mit Hilfe von Shell-Kommandos, die über

eine Pipe abgesetzt werden. Um den Benutzer später mit den korrekten Ausgabeinformationen versorgen zu können, erstellt die Funktion *compose_script* anschließend aus dem UNICORE Jobskript eine neue UNICORE Globus Datei in dem gleichen Verzeichnis. TSI_JOBNAME bildet zusammen mit dem in TSI_USPACE_DIR stehenden Pfad den RSL-Parameter `arguments`. Den Pfad für die Standardausgabe und die Fehlermeldungen entnimmt Globus den RSL-Parametern `stdout` und `stderr`, die mit Hilfe von TSI_OUTCOME_DIR gebildet werden.

Tabelle 4.6 zeigt zusammenfassend die für die Anbindung notwendigen RSL-Parameter mit den zugehörigen UNICORE Parametern.

UNICORE Parameter	**RSL-Parameter**
#TSI_USPACE_DIR: /Uspace/dir/	("executable" = "Uspace_dir")
	("rsl_substitution" = $("GASS_URL")"*Adresse : Port*")
#TSI_USPACE_DIR: /Uspace/dir/ #TSI_JOBNAME: UNIGlo1	("arguments" = $("GASS_URL")"/Uspace/dir/UNIGlo1")
#TSI_OUTCOME_DIR: /Ausgabe/dir/	("stdout" = $("GASS_URL") "/Uspace/dir/stdout") ("stderr" = $("GASS_URL") "/Uspace/dir/stderr")

Abbildung 4.6: UNICORE Globus Umsetzungstabelle

Nach der Umwandlung in die RSL initialisiert das JRI den GRAM-Klienten und den GASS-Server. In der UNICORE Globus Kopplung ermöglicht der GASS-Server[1] den Transfer des auszuführenden Skripts vom JRI zu einem Globus Zielsystem und der Ausgabe- und Fehlerdateien von Globus zurück zum JRI. Der GASS-Server wird in dem Modul *file_manager* durch Globus-Routinen, die von der Funktion `start_file_manager` gestartet werden, aufgerufen. Zunächst aktiviert die Globus-Funktion `globus_module_activate` den GASS-Server. Anschließend werden die benötigten Optionen für den Betrieb des GASS-Servers aus der Globus GASS-Bibliothek gewählt. Für die UNICORE Globus Kopplung reicht es aus, Lese- und Schreibrechte für den vom GASS-Server verwalteten Speicherbereich zu definieren. Damit ist der GASS-Klient in der Lage, das Jobskript auf das Zielsystem zu kopieren und die Ergebnisdateien zum lokalen System zurück zu schicken. Die Funktion `globus_gass_transfer_listener_get_base_url` ermittelt die URL des GASS-Servers (GASS_URL), welche Bestandteil aller erzeugten RSL-Zeichenketten ist.

In dem Modul *job_submission* findet die Job-Übermittlung von UNICORE nach Globus statt. Dazu wird der GRAM-Klient und ein Rückrufmechanismus benötigt (siehe Kapitel 3.2.1). Der GRAM-Klient, der die Übermittlung der RSL-Parameter zu dem gewählten GRAM-Gatekeeper ermöglicht, wird in *tsi_submit* durch die Globus-Funktion `globus_module_activate` aufgerufen. Während die Parameter von dem GRAM-Klienten über den GRAM-Gatekeeper zum GRAM-Jobmanager gesendet werden, nehmen die Statusinformationen den umgekehrten Weg vom GRAM-Jobmanager zurück zum JRI. Aus diesem Grund registriert sich das JRI nach der Initialisierung des GRAM-Klienten und des GASS-Servers mit der Globus Routine `globus_gram_client_callback_allow` zu einem Rückrufmechanismus mit Globus. Ein dieser Routine übergebener Parameter ist die Funktion `back_caller`, welche zum einen dem entfernten Jobmanager ermöglicht, den GRAM-Klienten zu kontaktieren, und zum anderen die Auswertung der von Globus nach der Job-Submission zurückkommenden Statusinformationen übernimmt. Zunächst jedoch sendet der GRAM-Jobmana-

[1] siehe auch Kapitel 3.2.2

ger die eigene Identifikation über die Rückrufverbindung zum JRI. Das JRI benötigt diese Information, um eine Abbildung der Globus Jobkennung auf die UNICORE Jobkennung zu erreichen. Die Identifikation des Jobmanagers wird in einer Tabelle gespeichert und erlaubt, sämtliche Ergebnis- und Statusinformationen dem zugehörigen Job zuzuordnen.

Mit der Globus-Funktion `globus_gram_client_job_request` wird die Job-Submissionsanfrage zu dem ausgewählten Globus Zielsystem geschickt, während das zugehörige Job-Skript vom GASS-Klienten beim GASS-Server angefordert und auf das Zielsystem transferiert wird. Der Benutzer ist zwar in der Lage, den Job über das JRI abzubrechen, oder anzuhalten, ansonsten jedoch findet die Job-Verarbeitung unabhängig von UNICORE unter der Kontrolle von Globus statt. Immer dann, wenn sich der Status ändert, sendet Globus den neuen Status über die Rückrufverbindung automatisch zurück zum JRI. Der Globus Status wird anschließend auf die entsprechende UNICORE Darstellung abgebildet. UNICORE und Globus unterscheiden vier verschiedene Stati:

- *PENDING*: der Job wartet auf Ausführung,
- *ACTIVE* : der Job wird verarbeitet,
- *FAILED* : der Job ist zwar beendet, aber ein Fehler ist aufgetreten, oder
- *DONE* : der Job konnte erfolgreich verarbeitet werden.

Bei einer Statusnachfrage durch den Benutzer wird nicht, wie in UNICORE, ein `qstat`-Kommando zu einem lokalen Batch-Subsystem abgesetzt, sondern die aktuellen Einträge werden in der Tabelle gefiltert und die den jeweiligen Job betreffenden Informationen zum UNICORE Klienten zurückgeschickt.

Das Sichern der Ergebnisse und die Weiterleitung an den UNICORE Klienten ist die letzte Station in der UNICORE Globus Jobübermittlung. Nach Beendigung der Job-Ausführung schickt Globus die Ausgabedateien über den GASS-Klienten zum GASS-Server im JRI und speichert diese in die zuvor angelegten Ausgabe-Dateien.
UNICORE bindet bei der Job-Verarbeitung in die Ausgabe einen allgemeiner Informationstext ein. Das Ausgabeformat von einem UNICORE- und einem Globus-Job soll für den Benutzer einheitlich erscheinen. Deshalb fügt die Funktion `compose_script` (aus dem Modul job_mapping) in den zu submittierenden Job ein Shell-Skript ein, das die benötigten Informationen für die Globus Ausgabe erzeugt. Abbildung 4.7 zeigt das Grundgerüst der UNICORE und der Globus Ausgabe im UNICORE Klienten.

UNICORE Pro/Epilog	**Globus Pro/Epilog**
This script was created and executed by Unicore	This script was created by UNICORE and executed by Globus
UNICORE - start of user output on stdout	Globus - start of user output on stdout
UNICORE Standard Ausgabe	*UNIGLO Standard Ausgabe*
UNICORE - end of user output on stdout	Globus - end of user output on stdout

Abbildung 4.7: Grundgerüst der UNICORE und der Globus Ausgabe im UNICORE Klienten

Alle bei der Job-Übermittlung von UNICORE nach Globus enstehenden Nachrichten und Fehlermeldungen protokolliert das Modul *log_messages* in einer Datei, auf die der Benutzer Zugriff hat. Die Funktion `error_report` wird aufgerufen, wenn ein Fehler in der Programmausführung aufgetreten ist. Die Job-Verarbeitung ist von dieser Art Fehler normalerweise nicht betroffen. Bei einem nicht erfolgreichen Start des GRAM-Klienten oder bei einer fehlgeschlagenen Job-Submission wird der Job jedoch beendet und die Informationen über den Jobabbruch durch die Funktion `exit_report` protokolliert. Informationen über den laufenden Betrieb wertet die Funktion `report` aus.

Meldungen, die in die UNICORE-Informationsdatei geschrieben werden sollen, werden der Funktion *to_tsi_report* übergeben. In der für die UNICORE Globus Kopplung entwickelten Bibliothek `uniglo_errors` sind sämtliche Fehlermeldungen aufgeführt, die während der Job-Übermittlung auftreten können.

Nachdem die Job-Submission beendet ist, werden der GASS-Server und der GRAM-Klient von den entsprechenden Globus-Routinen in der `terminate_file_manager` geschlossen, während die ETSI-Komponenten (JRI, CAI, TSI und Proxy-CA) auf den nächsten Job warten.

4.4.3 UNICORE Globus Zertifizierungsserver

Der UNICORE Globus Zertifizierungsserver besteht aus dem *Certification Authority Interface (CAI)* und der *Proxy-CA*. Das CAI unterhält eine Datenbank, in der die UNICORE Benutzernamen, die zugehörigen Globus Benutzer- und Proxy-Zertifikate abgelegt sind und stellt die Verbindung zwischen TSI und Proxy-CA her. Die in dem CAI implementierten Funktionen formulieren die Zertifikatanfragen für einen UNICORE Benutzer und leiten diese zur Proxy-CA weiter. Diese Funktionen sind in dem Modul *connect_to_ca* implementiert worden. Das UNICORE Globus Zertifizierungs-

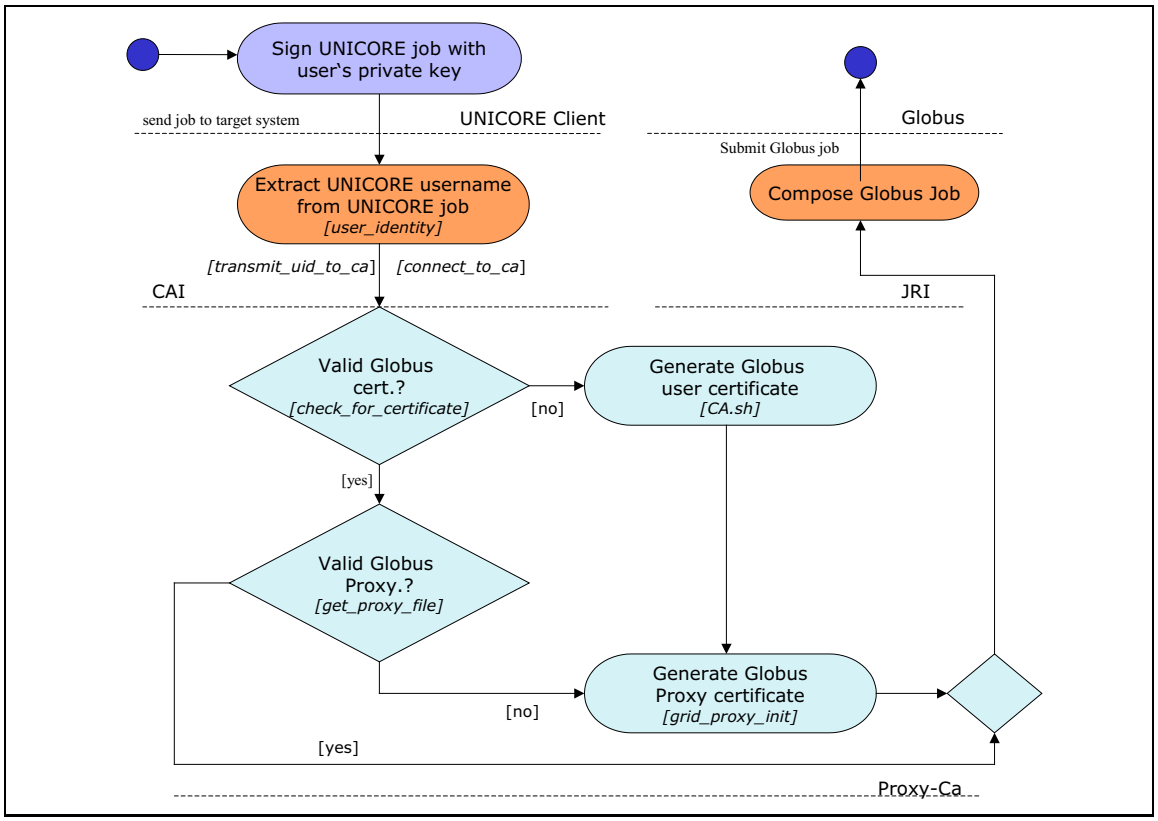

Abbildung 4.8: Das UNICORE Globus Zertifizierungsprinzip

prinzip ist in Abbildung 4.8 dargestellt und funktioniert wie folgt: Der UNICORE Job, der an ein Globus-Zielsystem submittiert werden soll, ist mit dem privaten Schlüssel des UNICORE Benutzers signiert. Zunächst authentisiert das UNICORE Gateway den Benutzer mit dessen öffentlichem UNICORE Schlüssel. Anschließend ist der Benutzer berechtigt, Jobs über UNICORE zu submittieren. Um Jobs nach Globus submittieren zu können, muß der Benutzer sich zusätzlich gegenseitig mit dem GRAM-Gatekeeper von Globus authentisieren. Das Prinzip der UNICORE und Globus internen Authentisierung ist in den Kapiteln 2.4 und 3.2.3 erläutert.

Im folgenden wird die für die UNICORE Globus Kopplung benötigte Abbildung des UNICORE

Benutzernamens auf ein Globus-Benutzerzertifikat beschrieben:
Für jede `submit`-Anfrage aus dem UNICORE Klienten extrahiert die Funktion `user_identity` in job_mapping den UNICORE Benutzernamen aus dem UNICORE Parameter TSI_IDENTITY. Da in UNICORE jeder Benutzername nur einmal vergeben wird, und auch auf dem jeweiligen Globus-Zielsystem die Namen eindeutig sind, sind Verwechslungen ausgeschlossen.

Das Modul connect_to_ca initialisiert mit `socket_to_ca` eine Socket-Verbindung zwischen CAI und Proxy-CA und sendet mit der Funktion `transmit_uid_to_ca` den UNICORE Benutzernamen zur Proxy-CA. Die Proxy-CA basiert auf der SSLeay-Bibliothek [22], die von Globus für die Einrichtung von eigenen CAs propagiert wird. Das in dieser Bibliothek u.a. implementierte Shell-Skript `CA.sh` und die Konfigurationsdatei `ssleay.cnf` sind für die UNICORE Globus Kopplung modifiziert worden.

Mit der Funktion `check_for_certificate` prüft die Proxy-CA, ob bereits ein Globus-Zertifikat für den Benutzer vorhanden ist. Falls noch kein Globus Benutzerzertifikat existiert, erstellt die Proxy-CA ein neues. Dabei führt `CA.sh` die eigentliche Zertifizierung durch. Dieses Skript wird von der Proxy-CA in der Funktion `get_cert` aufgerufen. `CA.sh` erzeugt für den übermittelten UNICORE Benutzernamen einen privaten Schlüssel und ein Globus Benutzerzertifikat, das den öffentlichen Schlüssel beinhaltet. Das Benutzerzertifikat und der private Schlüssel, mit denen später das Globus Proxy-Zertifikat signiert wird, werden von der Proxy-CA lokal in der Datenbank gespeichert.

In der UNICORE Globus Kopplung erzeugt die Proxy-CA, und nicht wie in Globus der Benutzer, den privaten Schlüssel. Der Benutzer ist also nicht im Besitz des Passworts, mit dem er auf den privaten Schlüssel zugreifen kann. Er hätte aber auch gar keine Möglichkeit das Passwort einzugeben, da er keine Eingaben, die die UNICORE -Globus Job-Übermittlung intern beinflußen, über den UNICORE Klienten absetzen kann. Der private Schlüssel muß also ohne Passwort auf dem Zertifizierungsserver abgespeichert werden. Dadurch erhöht sich zwar prinzipiell die Gefahr, daß der private Schlüssel von Unbefugten gestohlen wird. Diese Gefahr wird jedoch dadurch minimiert, daß die CAI und die Proxy-CA für die UNICORE Globus Kopplung auf einem dedizierten, sicheren Rechner eingerichtet ist.

Um das Globus Benutzerzertifikat mit den jeweiligen Benutzerinformationen zu versehen, werden bestimmte Parameter benötigt. Diese werden in der Konfigurationsdatei ssleay.cnf gesetzt. Die SSLeay-Routinen sind von den Autoren so programmiert worden, daß der Benutzer sämtliche Parameter zur Erzeugung des *Distinguish Name* (siehe Kapitel 3.2.3), wie die Länderkennung, den Projekt- und den Benutzernamen, über die Standardeingabe selbst erzeugen muß. Auch hier ergibt sich das Problem, daß der Benutzer die entsprechenden Eingaben nicht über den UNICORE Klienten formulieren kann. Daher ist diese Implementierung für die UNICORE Globus Kopplung nicht zweckmäßig.

Die SSLeay-Routinen sind in dieser Arbeit so modifiziert worden, daß die Parameter nunmehr automatisch und ohne weitere Eingaben des Benutzers erzeugt werden. Die Parameter Länderkennung und Projekt sind fest vorgegeben. Der Benutzername, für den das Globus Benutzerzertifikat erstellt werden soll, wird bei jeder Kontaktierung der Proxy-CA mit der Funktion `write_in_file` an die entsprechende Stelle in dem SSLeay-Konfigurations-Skript geschrieben.

Wenn ein neues Globus Benutzerzertifikat erstellt worden ist, dann wird es von der Proxy-CA signiert. Zusätzlich muß auch ein Globus-Proxy für den Benutzer generiert werden. Das Globus-Proxy wird mit der in `get_proxy_file` aufgerufenen Globus-Routine `grid_proxy_init` (siehe Kapitel 3.2.3) erzeugt. Das Proxy-Zertifikat wird anschließend von dem Globus Benutzerzertifikat signiert. Die Gültigkeitsdauer des Proxy-Zertifikats (von standardmässig acht Stunden) kann vom Proxy-CA-Administrator im CAI beliebig gesetzt werden.

Auch das Proxy-Zertifikat wird in der Datenbank abgelegt. Auf dieses Proxy-Zertifikat greift die Globus Grid-Security Infrastructure (GSI) bei der Authentisierung mit dem GRAM-Gatekeeper (siehe Kapitel 3.2.3) zu.

Falls für den Benutzer bereits ein Globus-Zertifikat in der Datenbank vorhanden ist, braucht die Funktion `check_for_certificate` nur noch die Gültigkeit des Globus-Proxies zu prüfen. Wenn es abgelaufen ist, dann wird nach dem oben beschrieben Verfahren ein neues Globus-Proxy für den Benutzer generiert und ebenfalls an das JRI weitergeleitet.

4.5 Aufruf der ETSI-Komponenten

Für die UNICORE Globus Jobübermittlung müssen verschiedene Komponenten gestartet werden. Während der Benutzer nur wie gewohnt den UNICORE Klienten zu starten braucht, muss der Administrator des ETSI-Servers gewährleisten, daß der UNICORE Server bestehend aus Gateway und NJS, sowie das Enhanced TSI permanent zur Verfügung stehen. Da der Zertifizierungsserver auf einem dedizierten Rechner läuft, muss das CAI, welches intern die Proxy-CA initialisiert, zusätzlich vom Administrator dieses Rechners gestartet werden. Die Globus-Komponenten werden vom JRI aufgerufen.

Kapitel 5

Status

Der Prototyp der UNICORE Globus Kopplung ist so implementiert worden, daß er die in Kapitel 4.1 aufgestellten Anforderungen erfüllt: in dieser Arbeit ist ein Mechanismus implementiert worden, der die UNICORE-Architektur mit der Globus-Architektur koppelt. Die UNICORE Globus Kopplung ermöglicht dem Benutzer, einen Job im UNICORE-Klienten zu erstellen, diesen zu Globus zu submittieren und dort berechnen zu lassen. Der Benutzer ist in der Lage, den Job-Status über den Klienten abzufragen. Auch die Ergebnisse der Berechnung werden an den Benutzer zurückge-

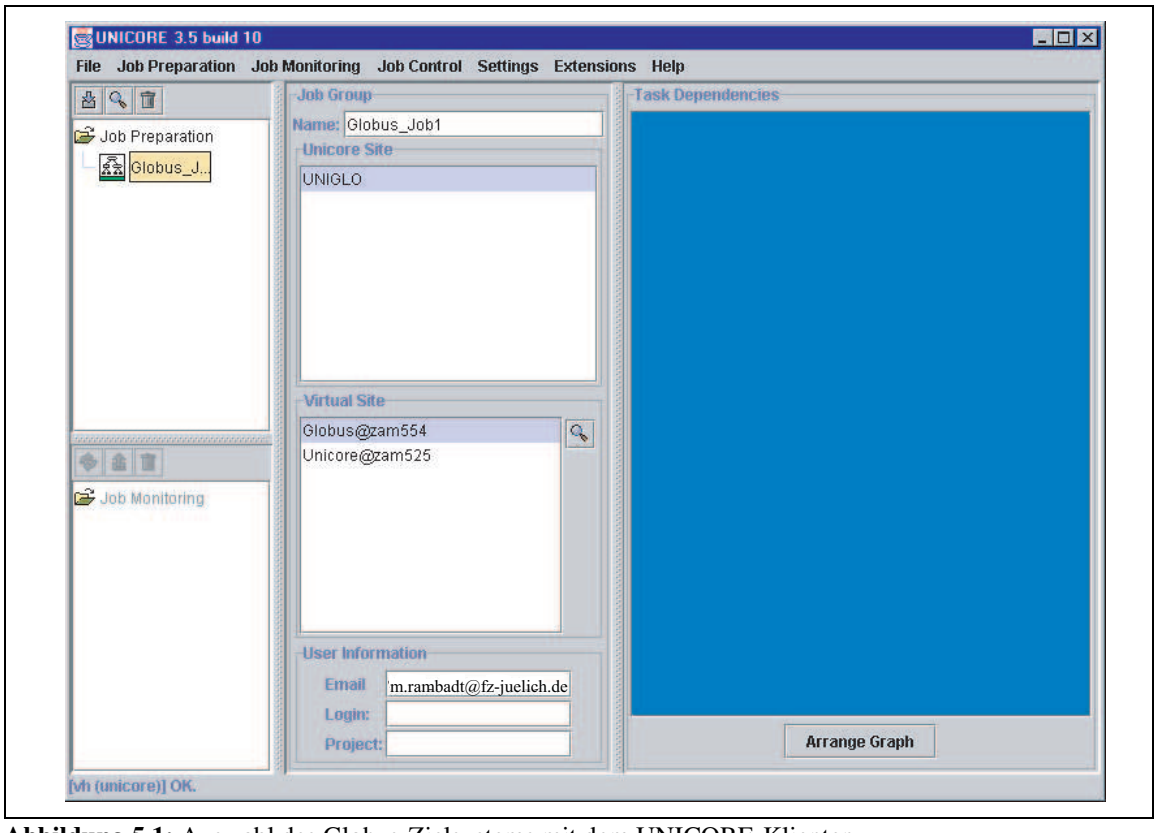

Abbildung 5.1: Auswahl des Globus-Zielsystems mit dem UNICORE-Klienten

sendet. Diese Funktionalitäten basieren auf der automatischen Generierung von Globus Zertifikaten und Globus-Proxies für den UNICORE-Benutzer.

Für den UNICORE-Benutzer stellt sich Globus als ein weiteres UNICORE-Zielsystem dar, das er im UNICORE-Klienten für die Job-Submission auswählen kann. Die internen Abbildungen der UNICORE-Parameter auf die Globus-Parameter und die Abbildung der Benutzeridentität auf ein Globus-Zertifikat bleiben dem Benutzer verborgen. Somit braucht der Benutzer keine zusätzlichen Einstellungen im UNICORE-Klienten vorzunehmen, um die Funktionalitäten der UNICORE Globus Kopplung nutzen zu können. Der Benutzer muß lediglich dafür sorgen, daß er einen Zugang zu dem Globus-System hat, zu dem er seinen Job submittieren möchte. Außerdem müssen sein *Distinguish Name* (siehe Kapitel 3.2.3) und die lokale Benutzeridentität vom Administrator des Globus-Rechners in den *gridmap file* eingetragen werden. Da Globus aber wie ein UNICORE-Zielsystem behandelt wird, und der Benutzer sich in jedem Fall bei den jeweiligen UNICORE-Zielsystemen in der *UNICORE User Database (UUDB)* (siehe Kapitel2.3.3) registrieren muß, ist er mit diesem Umstand bereits vertraut.

Die Abbildungen 5.1 - 5.5 zeigen die erfolgreiche UNICORE Globus Kopplung an einem Beispiel. Es wird eine Benutzersitzung simuliert.

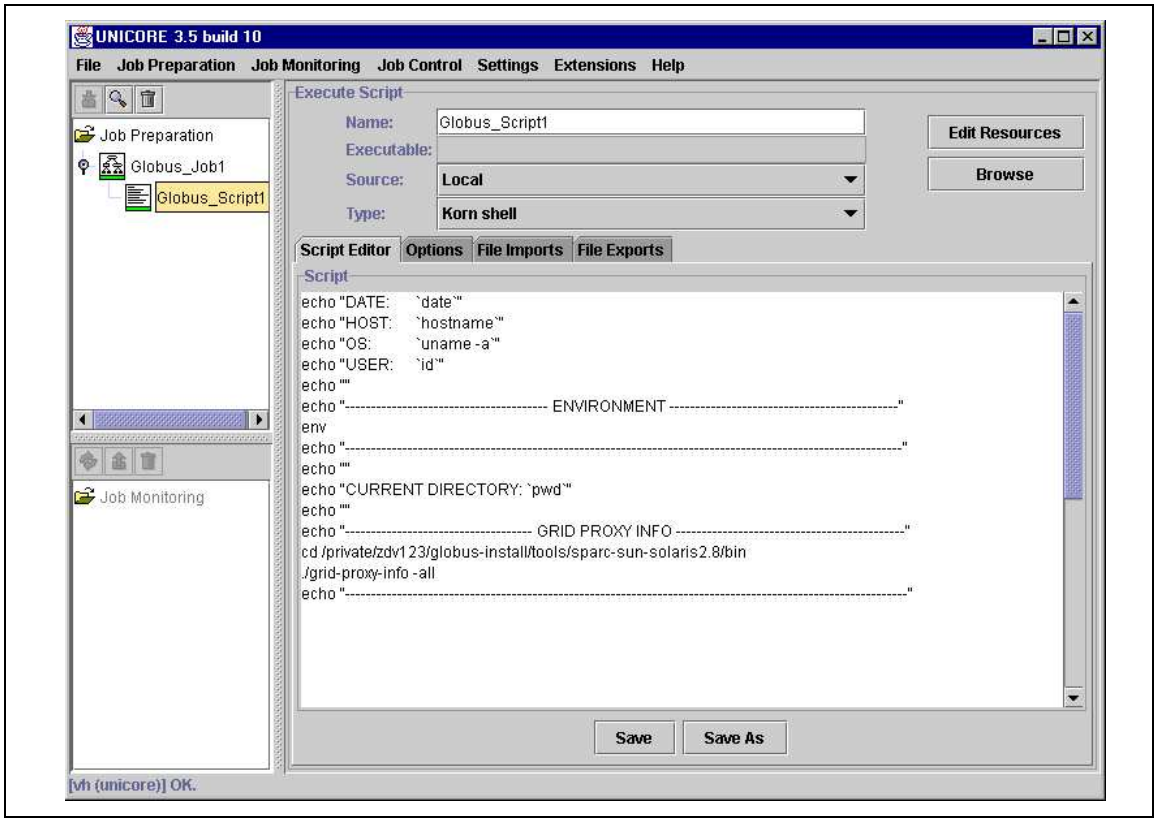

Abbildung 5.2: Job-Erstellung mit der Skript-Task

Die Job-Erstellung erfolgt mit der grafischen Oberfläche vom UNICORE-Klienten. In Abbildung 5.1 ist zu sehen, daß der Benutzer den zu erstellenden Job *Globus_Job1* nennt. Anschließend wird das Rechenzentrum (bzw die UNICORE Site), wo der Job verarbeitet werden soll, ausgewählt. Die gewählte UNICORE-Site *UNIGLO* enthält in diesem Fall zwei Zielsysteme, auf die der Job submittiert werden kann. Der Job soll auf dem Globus-Zielsystem Globus@zam554 berechnet werden.

Nach diesen vorbereitenden Schritten erstellt der Benutzer seinen Job mit der *Skript-Task* im *UNICORE Script Editor*, wie in Abbildung 5.2 zu sehen ist. Der Benutzer nennt das Skript *Globus_Script1*. Er möchte verschiedene Shell-Kommandos auf dem Zielsystem ausführen lassen, die

- das aktuelle Datum der Job-Submission,

- den Namen des Globus-Zielsystems, auf dem der Job berechnet werden soll,
- das Betriebssystem des Zielsystems und
- die Umgebungsvariablen des Zielsystems ermitteln.

Ein weiterer Bestandteil des Jobs ist die Globus-Routine `grid-proxy-info`. Diese Routine gibt alle verfügbaren Informationen über das bei der Delegation vom Globus-Zielsystem erzeugte Proxy zurück. Nach der Job-Erstellung erfolgt wie in Abbildung 5.3 gezeigt die Job-Submission

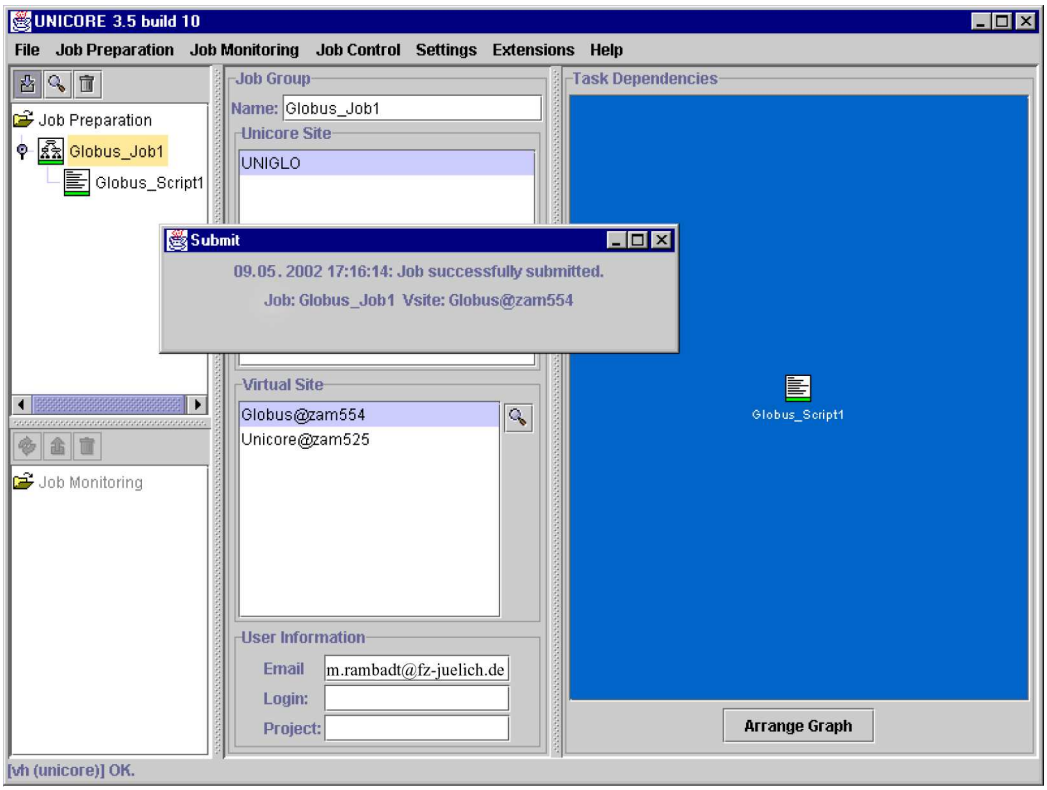

Abbildung 5.3: Job-Submission zu dem ausgewählten Globus-Zielsystem

zu dem Globus-Zielsystem über den *submit-Button*. UNICORE zeigt eine Meldung, wenn die Job-Submission erfolgreich gewesen ist. Der Job wird nun intern von Globus verarbeitet. Die Job-Verarbeitung bleibt dem Benutzer verborgen. Während der Job-Verarbeitung wird der Status der Berechnung ermittelt. Bei erfolgreicher Ausführung des Jobs wird der Status im *UNICORE Job Monitoring*-Fenster grün dargestellt. Eine fehlerhafte Job-Ausführung erscheint rot und ein noch nicht beendeter Job gelb.

Die Abbildungen 5.4 und 5.5 zeigen die Standard-Ausgabe der Job-Verarbeitung. In Abbildung 5.4 sind die Ergebnisse der Shell-Befehle zu sehen. Den Kopf der Ausgabedatei bildet der in Kapitel 4.4.2 erzeugte allgemeine Informationstext.

Das Ergebnis des grid-proxy-init-Kommandos ist in Abbildung 5.5 zu sehen. Als *subject* erscheint der Benutzername, mit den Erweiterungen *proxy* und *limited proxy*, um zu verdeutlichen, daß das Proxy bei der Delegation auf dem Globus-Zielsystem im Auftrag des Benutzers erzeugt worden ist. In der darauffolgenden Zeile wird mit *issuer* die Instanz angegeben, die das Proxy-Zertifikat signiert hat. Das ist in diesem Fall das vom GRAM-Klienten erzeugte Proxy-Zertifikat (siehe auch Kapitel 3.2.3).

Die UNICORE Globus Kopplung ist erfolgreich auf der SC2001-Konferenz in Denver präsentiert worden.

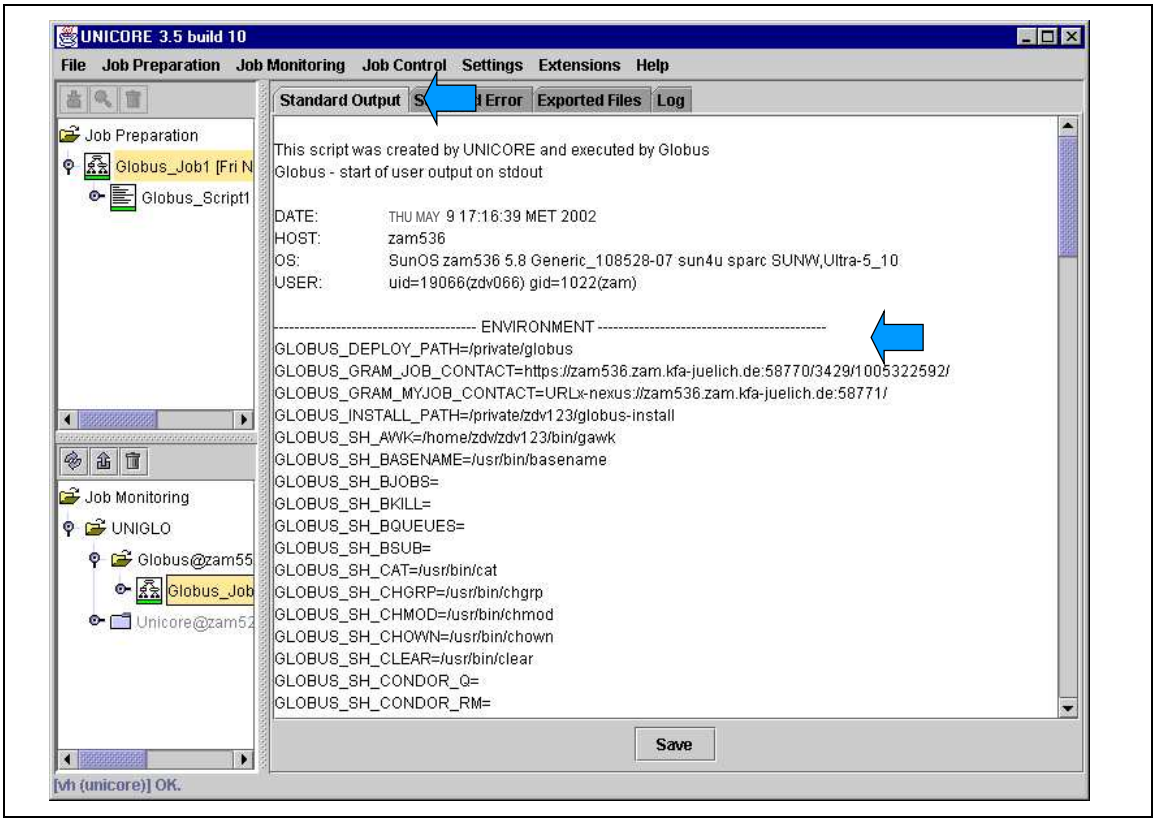

Abbildung 5.4: Standard-Ausgabe des ausgeführten Jobs im UNICORE-Klienten (1.Teil)

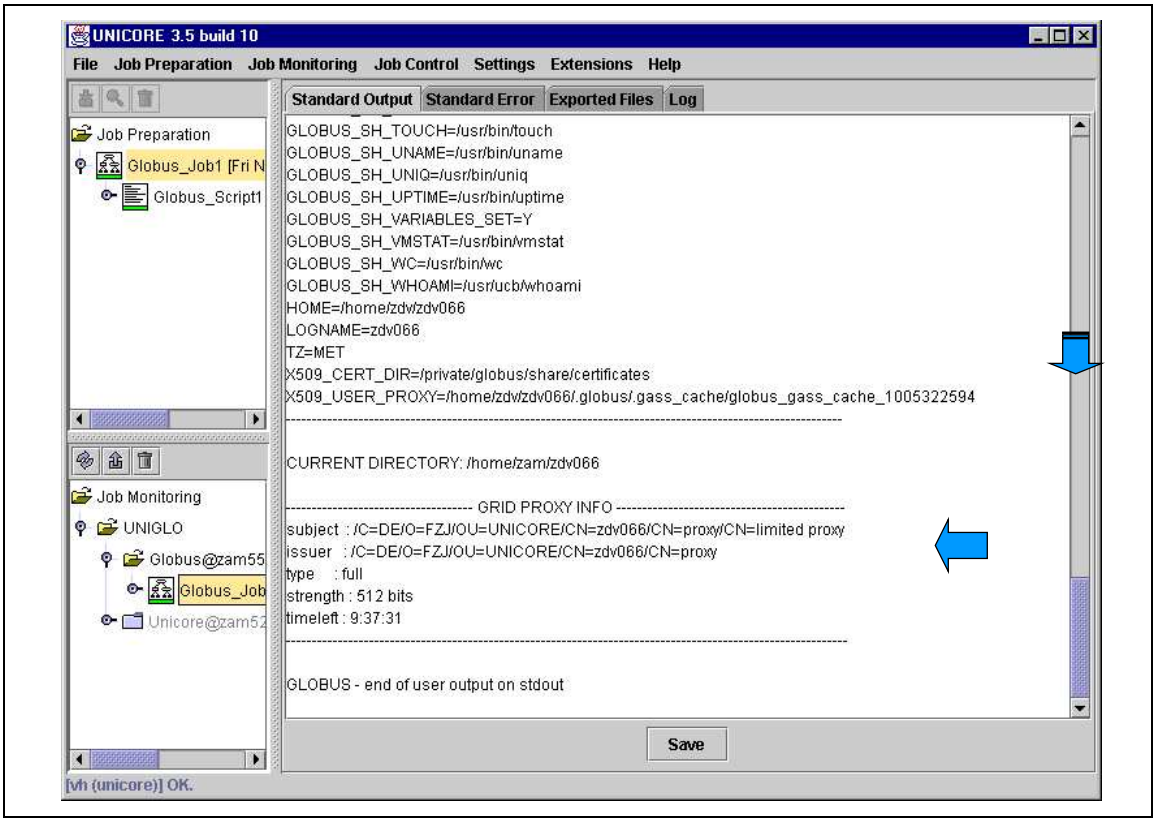

Abbildung 5.5: Standard-Ausgabe des ausgeführten Jobs im UNICORE-Klienten (2.Teil)

Kapitel 6

Ausblick

Wie in Kapitel 5 gezeigt sind die geforderten Funktionalitäten der UNICORE Globus Kopplung entwickelt und implementiert worden, ohne die jeweiligen Architekturen verändert zu haben. Die Entwicklung der UNICORE Globus Kopplung ist jedoch noch nicht vollständig abgeschlossen. Es sind noch Erweiterungen denkbar, die teilweise bereits jetzt schon entwickelt werden. Eine Integration weiterer von Grid-Benutzern geforderten in UNICORE vorhandenen Funktionen (beispielsweise die Unterstützung von mehrstufigen Jobs bzw. von Jobs, die benötigte Teile von verschiedenen Seiten anfordern) erfordert zusätzliche erweiterungen.

Um die Funktionalität des hier vorgestellten Prototyps erweitern zu können, ist das *GRIP-Projekt* [23] (Grid Interoperability Project)[1] Anfang des Jahres 2002 gestartet worden. Dieses Projekt wird von der Europäischen Union mit einer Förderdauer von Januar 2002 bis Dezember 2003 unterstützt. Bei diesem Projekt kooperieren unter der Projektleitung des ZAM verschiedene europäische Forschungszentren zusammen mit den Argonne National Laboratories. Die GRIP-Projekt Aufgabenstellungen sind u.a.

- eine erweiterte Integration von UNICORE und Globus, unter Verwendung der während der Entwicklung dieses Prototypen gewonnenen Ergebnisse und Einsichten. Bei diesem Projekt soll nicht ausgeschlossen werden, daß mögliche Änderungen in den Architekturen von UNICORE und Globus vorgenommen werden, was den Projektpartnern interessante Möglichkeiten für die Entwicklung bietet. So ist beispielsweise denkbar, daß der Zertifizierungsmechanismus in den UNICORE-Klienten integriert wird.
- Erarbeitung von Grid Standardisierungsvorschlägen im Global Grid Forum [24] und Prototypentwicklungen der relevanten Standards.
- Unterstützung von biomolekularen und metereologischen Anwendungen in der erweiterten UNICORE Globus Entwicklung.

Die Ergebnisse der hier vorliegenden Arbeit zeigen, daß unabhängige Grid-Technologien, wie UNICORE und Globus, erfolgreich miteinander kombiniert werden können, was sehr interessant für zukünftige Entwicklungen in dieser Richtung sein wird.

[1] IST-20001-32257

Literaturverzeichnis

[1] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a Future Computing Infrastructure*. 1999.

[2] "NEESGrid." `http://www.neesgrid.org`.

[3] "Globus." `http://www.globus.org`.

[4] "UNICORE." `http://www.unicore.de`.

[5] D. Erwin, "UNICORE and EUROGRID: Grid Computing in Europe," May 2001. Vortrag gehalten auf der Terena Networking Conference 2001 in Antalya, Türkei.

[6] V.Huber, "Supporting Car-Parinello Molecular Dynamics with UNICORE," in *Computational Science, ICCS International Conference San Franzisco*, CA, USA, May 2001.

[7] D. W. Erwin and D. F. Snelling, "UNICORE: A Grid Computing Environment," in *Proceedings of Euro-Par 2001*, pp. 825–834, Springer LNCS 2150, August 2001.

[8] M. Romberg, "The UNICORE Grid Infrastructure." to appear in: Scientific Programming, Special Issue on Grid Computing.

[9] M. Romberg and D. Erwin, "UNICORE: Uniformes Interface für Computer Ressourcen," in *Praxis der Informationsverarbeitung und Kommunikation (PIK) Ausgabe 2/01*, K.G. Saur Verlag, München, April 2001.

[10] J. Feghhi, J. Feghhi, and P. Williams, *Digital Certificates-Applied Internet Security*. 1998.

[11] V.Huber, "UNICORE: A Grid Computing Environment for Distributed and Parallel Computing," in *Parallel Computing Technologies, 6th International Conference, PaCT 2001*, Springer, September 2001.

[12] M. Ashley and P. Neuhaus, *Das Gnu-Handbuch zum Schutze der Privatsphaere*.

[13] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," pp. 365–375, 1997.

[14] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," in *The Grid: Blueprint for a Future Computing Infrastructure*, pp. 259–278, MORGAN-KAUFMANN, 1998.

[15] "Cray-Handbuecher: SG-2148 NQE User's Guide 3.3."

[16] J. Basney, M. Livny, and T. Tannenbaum, "High throughput computing with condor," in *HPCU news, Volume 1*, June 1997.

[17] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A data movement and access service for wide area computing systems," in *Proc. IOPADS'99*, 1999.

[18] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in *ACM Conference on Computers and Security*, pp. 83–91, 1998.

[19] `http://web.mit.edu/kerberos/www/`.

[20] "Globus Toolkit User Tutorial Part 2: Security and Remote Process Creation." Tutorial, May 2001. `http://www.globus.org/training/toolkit-internals/03-Common.pdf`.

[21] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy," in *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[22] "SSLeay documentation." Tutorial. `http://www2.psy.uq.edu.au/ ftp/Crypto/`.

[23] `http://www.grid-interoperability.org`.

[24] "Global Grid Forum." `http://www.gridforum.org/`.