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

Interner Bericht

UNICORE - Uniformes Interface für Computer Ressourcen

Mathilde Romberg, Dietmar Erwin

FZJ-ZAM-IB-2001-06

Juli 2001

(letzte Änderung: 10.07.2001)

Preprint: PIK 2/2001 (Praxis in der Informationstechnik und Kommunikation), Saur Verlag, pp. 102-110

UNICORE – Uniformes Interface für Computer Ressourcen

Mathilde Romberg, Dietmar Erwin
Forschungszentrum Jülich GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich
e-mail: [m.romberg, d.erwin]@fz-juelich.de

1. Zusammenfassung

Wissenschaftler aus Universitäten und Forschungseinrichtungen nutzen seit langem Rechner an entfernten Standorten, um komplexe Probleme aus Computational Science und Engineering zu lösen. Da Rechner verschiedener Hersteller zum Einsatz kommen, besteht ein erheblicher Lernaufwand, bevor ein neuer Rechner genutzt werden kann. Benutzer, die für ihre Anwendungen Supercomputer benötigen, sind gezwungen, sich mit den system- und rechenzentrums-spezifischen Gegebenheiten vertraut zu machen. Dieses kann sehr zeitaufwendig werden, insbesondere wenn Rechner in verschiedenen Zentren genutzt werden z.B. für heterogenes Rechnen oder Metacomputing. UNICORE (Uniformes Interface für Computer-Ressourcen) wurde entworfen, um den Benutzern einen einfachen und sicheren Zugang zu verteilten Rechner-Ressourcen zu schaffen.

2. Motivation

Den Anstoß zur Entwicklung eines "seamless Interface" zu Rechnersystemen gaben die Arbeiten zur VESUZ-Machbarkeitsanalyse ([4]), in der die Möglichkeiten und Rahmenbedingungen für einen Verbund der Supercomputer-Zentren in Deutschland untersucht wurden. Eine Voraussetzung für die Realisierung der Empfehlungen der VESUZ-Studie ist, daß die Ressourcen der Supercomputer-Zentren für die Benutzer einfach erreichbar sind. Das heißt, der Aufwand für den Wechsel von einem zum anderen Zentrum muß minimal werden. Derzeit muß ein Benutzer sich zusätzlich zu der eigentlichen Aufgabe, der Lösung eines wissenschaftlichen Problems mit Hilfe des Computers, darum kümmern, welches System von welchem Hersteller mit welchem Betriebssystem wie zu bedienen ist und wie die lokalen Regularien (Benutzernummer, Sicherheitsmechanismen, Scheduling, Datenhaltung, ...) aussehen. Die Schwierigkeiten liegen in den feinen Unterschieden bei den Befehlen der verschiedenen Betriebssysteme und Batch-Subsysteme, auch wenn die meisten UNIX bzw. NQS-Derivate sind, sowie bei den Unterschieden zwischen den Regeln und Gegebenheiten an den Zentren selbst. Neben den unterschiedlichen Benutzerkennungen und Authentisierungen sind es gerade die Unterschiede bei den Nutzungsmodellen, dem Scheduling, der Datenhaltung (wo dürfen welche Mengen von Daten für welchen Zeitraum abgelegt werden) und dem Accounting.
Gleichzeitig wird den Benutzern die Bearbeitung der Rechenprobleme auf der jeweils dafür geeigneten Rechnerarchitektur erleichtert und somit Rechner effizienter genutzt. Denn nicht immer steht das geeignete System lokal zur Verfügung, so daß entfernt stehende Rechner eingesetzt werden müssen; auch ist ein benötigtes Softwarepaket aus Kostengründen nicht immer lokal verfügbar.
UNICORE hat zum Ziel, die Stoßkanten zwischen den Systemen und zwischen den Zentren für den Benutzer zu beseitigen und eine Software-Infrastruktur für einen einheitlichen, intuitiven und sicheren Zugang zu den Zentren zu schaffen. Dieses Ziel beinhaltet für den einheitlichen Zugang zum einen die Definition einer systemunabhängigen Job-Beschreibung zur Formulierung von abstrakten Jobs und zum anderen die Schaffung einer einheitlichen UNICORE Benutzeridentifikation. Für den intuitiven Zugang gehört eine ansprechende Benutzeroberfläche dazu und für die Sicherheit eine tragfähige, auf Standards basierende Sicherheitsarchitektur. Daneben darf UNICORE nur minimale Veränderungen der lokalen administrativen Verfahren wie z.B. Benutzerverwaltung und Scheduling erfordern, um Akzeptanz zu erzielen und eine leichte Integration in die Zentren zu ermöglichen. Weitere Ziele von UNICORE sind, existierende Techniken auszunutzen (z.B. das World Wide Web) und Schnittstellen offen zu definieren, damit UNICORE keine Insel schafft, sondern offen für die Zusammenarbeit mit anderen Lösungen ist. Das Projekt wird einen Prototypen entwickeln, der von den Supercomputer-Zentren genutzt wird und nach dem Projektende zu einem kommerziellen Produkt weiterentwickelt werden soll.

3. Die UNICORE Projekte

Mitte 1997 startete das vom BMBF geförderte Projekt UNICORE (Förderkennzeichen 01 IR 703) mit einer Laufzeit von zwei Jahren. Geförderte Projektpartner waren der Deutsche Wetterdienst in Offenbach (DWD), die Forschungszentrum Jülich GmbH, die Genias Software GmbH, das Konrad Zuse Zentrum für

Informationstechnik in Berlin (ZIB), das Leibniz Rechenzentrum der Bayrischen Akademie der Wissenschaften in München (LRZ), das Paderborn Center for Parallel Computing (PC^2), die Pallas GmbH, das Rechenzentrum der Universität Karlsruhe (RUKA) und das Rechenzentrum der Universität Stuttgart (RUS). Als nicht geförderte Projektpartner waren beteiligt das European Center for Medium Range Weather Forecast in Reading, UK, das Fujitsu European Center for Information Technology in Uxbridge, UK und die Herstellerfirmen Hitachi, HP, IBM, NEC, SGI/Cray, Siemens/Fujitsu und Sun Microsystems. In diesem Projekt wurde die Basisarchitektur von UNICORE entwickelt. Dazu gehört vor allem die Sicherheitsarchitektur, das Abstrakte Job Objekt (siehe Kapitel 5), die Benutzeroberfläche und die Übersetzung der abstrakten, einheitlichen Job-Beschreibung in konkrete Jobs. Die Ergebnisse des Projekts sind zusammengefaßt im gemeinsamen Abschlußbericht der Projektpartner ([6]).
Basierend auf den Erfahrungen und Entwicklungen aus dem ersten Projekt wurde im Januar 2000 das Nachfolgeprojekt 'UNICORE Plus' mit einer Laufzeit von drei Jahren begonnen (BMBF Förderkennzeichen 01 IR 001). An diesem Projekt sind beteiligt DWD, Forschungszentrum Jülich GmbH, ZIB, LRZ, PC^2, Pallas GmbH, RUKA, RUS und ZHR (Zentrum für Hochleistungsrechnen der Technischen Universität Dresden). Die Herstellerfirmen sind nicht mehr als Partner im Projekt vertreten. Sie haben sich mit anderen interessierten Einrichtungen und den Projektpartnern im UNICORE Forum e.V. ([23]) zur weiteren Unterstützung von UNICORE zusammengeschlossen.
Die im UNICORE-Projekt realisierten Funktionen umfassen das Erstellen und Überwachen von Batch-Anwendungen, das heterogene Rechnen in dem Sinn, daß einzelne Job-Schritte in einer festlegbaren Reihenfolge auf verschiedenen Systemen bearbeitet werden, sowie das transparente Bereitstellen der vom Benutzer spezifizierten Daten. In 'UNICORE Plus' wird ein breiteres Spektrum bearbeitet und ein Schwerpunkt auf die Software-Qualität und die Effizienz insbesondere beim Datentransfer gelegt. 'UNICORE Plus' ist in acht Teilprojekte gegliedert, die neben dem Projekt-Management und der Software-Erstellung für die Basiskomponenten die folgenden Themen bearbeiten:

- Werkzeuge für die Systemadministration und Qualitätssicherung sowie Erweiterung und Betrieb der Public Key Infrastruktur,
- Ressourcen-Modellierung,
- anwendungsspezifische Benutzer-Oberflächen,
- Datenmanagement,
- erweiterte Job-Ablauf Kontrolle und
- Metacomputing auf Anwendungsebene.

Das Projekt will mit diesen Arbeiten prototypisch ein High-Performance-Computing-Grid für die Forscher aus der Bundesrepublik erstellen.

4. Verwandte Projekte

Die Ideen, verteilte Computerressourcen zu einem Metacomputer zusammenzuschließen und effizient zu nutzen sowie Endanwendern einen einfachen Zugang zu verteilten Ressourcen zu ermöglichen, beschäftigt die Wissenschaft schon eine geraume Weile. Das Spektrum der zu lösenden Fragestellungen reicht von Pogrammiermodellen und deren Unterstützung über Scheduling, Quality of Service und Sicherheit bis zu Benutzerschnittstellen. In der Vielzahl der Projekte gibt es zum einen die Ansätze, Benutzern einen einfachen Zugang zu Rechen-Ressourcen über anwendungsspezifische Benutzeroberflächen zu ermöglichen. Damit können Benutzer allein über das Vokabular der Anwendung ihre Rechenaufträge formulieren und müssen sich nicht mit den Begriffen der System-Hardware und -Software auseinandersetzen. Ein Beispiel für ein Projekt aus diesem Bereich ist WebSubmit vom NIST ([15]). Es entwickelt ein einheitliches Web-basiertes Interface zu Anwendungen (z.B. Gaussian 94) auf verschiedenen heterogenen Rechnern.
Programmiermodelle für verteilte Anwendungen, Scheduling, Quality of Service und andere Themen werden im Bereich der Grid Computing Projekte behandelt. Dabei wird ein *computational grid* definiert als eine Hardware- und Software-Infrastruktur, die einen verläßlichen, konsistenten und kostengünstigen Zugang zu Hochleistungsrechner-Ressourcen ermöglicht ([10]). Die Idee dahinter ist, daß in Zukunft der Zugang zu Rechenkapazität, Programmen und Daten so einfach wird wie der Zugang zum Stromnetz (Power Grid). Zwei der wohl bekanntesten Projekte in diesem Bereich sind Legion von der Universität in Virginia ([13]) und Globus vom Argonne National Laboratory et al ([11]). Legion entwickelt ein vollständig objekt-orientiertes Meta System, das als Basis für Metacomputing-Entwicklungen dient. Das Ziel ist, eine einzelne kohärente virtuelle Maschine bereitzustellen, die skalierbar, einfach zu programmieren, fehlertolerant und sicher ist und die dabei die Autonomie der Zentren erhält. Im Globus-Projekt wird ein Metacomputing Toolkit für Grid-Anwendungen erstellt. Das Toolkit umfaßt verschiedene Dienste unter anderen Ressourcen-Zuweisung und Prozeßverwaltung, Kommunikation, den Datenzugriff und Security. Mit diesen Software-Servern können verschiedene, komplexere Metacomputing-Dienste aufgebaut werden wie z.B. Scheduler oder Werkzeuge für parallele Anwendungen. Beide Projekte zielen primär auf Metacomputing auf Anwendungsebene ab. Sie schließen eine Vielzahl verteilter Systeme zusammen, um eine sehr große Anwendung parallel bearbeiten zu können. Der Benutzer muß

dazu die Anwendung dem unterstützten Programmiermodell und den benötigten Diensten entsprechend anpassen. Parallele Anwendungen stehen auch beim Projekt Hypercomputing der Universität Rostock ([14]) im Vordergrund. Der entwickelte Hypercomputer akquiriert temporär nicht benutzte Workstation-Ressourcen in einem Weitverkehrsnetz zur nebenläufigen Bearbeitung von parallelen Anwendungen.
Die Vielzahl der Konzepte und Lösungen hat Grid Computing Interessierte dazu veranlaßt, sich in der nordamerikanischen Grid Forum Initiative ([12]) und im Europäischen Grid Forum ([8]) zusammenzuschließen, um Standards für Grid-Systeme zu vereinbaren. Die Initiativen wollen gemeinsam die Weiterverbreitung und die Entwicklung von ‚best practices', Implementierungs-Richtlinien und Standards betreiben und dabei auf existierenden Systemen aufbauen. Insgesamt soll eine integrierte Grid Architektur entwickelt werden, die der größer werdenden Grid-Gemeinde als Richtschnur für weitere Forschungs- und Entwicklungsaktivitäten dient.

5. Grundlegende Definitionen

Zum Verständnis der UNICORE-Architektur muß zunächst geklärt werden, welche Modelle dem System zugrunde liegen. Ein Benutzer erstellt einen UNICORE-Job (**Ujob**) für ein bestimmtes Zielsystem (**Vsite**) an einem Rechenzentrum, das UNICORE anbietet (**Usite**). Der Ujob kann aus mehreren Job-Schritten bestehen. Ein **Job-Schritt** ist entweder eine Task, die in einen Batch-Job für der Zielsystem übersetzt wird, oder ein Teil-Job, der für eine andere Vsite bestimmt ist und selbst wieder aus Job-Schritten besteht. Job-Schritte können voneinander abhängen. Zur Zeit werden nur sequentielle Abhängigkeiten unterstützt, d.h. eine Task bzw. ein Job-Schritt wird nur dann ausgeführt, wenn alle Vorgänger-Tasks bzw. Vorgänger-Job-Schritte erfolgreich ausgeführt worden sind. Voneinander unabhängige Teile können quasi parallel ausgeführt werden.
Jedem Ujob ist ein temporäres Job-Verzeichnis (**Uspace**) zugeordnet. Dieses wird vor dem Start eines Jobs an einer Vsite angelegt und nach dem Job-Ende wieder gelöscht. Alle Dateien, die ein Job benötigt, müssen in den Uspace importiert werden. Daten, die nach dem Ende eines Jobs weiterhin benötigt werden, müssen in einen permanenten Speicherbereich (z.B. ein Daten-Archiv) exportiert werden. Der Benutzer gibt jeweils nur die Dateien an, Import und Export werden von UNICORE transparent für den Benutzer durchgeführt.
Ujobs werden in einer einheitlichen Art und Weise unabhängig vom gewählten Zielsystem formuliert und als Abstract Job Object (**AJO**) zwischen den UNICORE-Komponenten transferiert. Das AJO repräsentiert damit einen UNICORE-Job bzw. Teil-Job. Es ist genauso rekursiv strukturiert wie ein Ujob und enthält die Beschreibung des Abhängigkeitsgraphen. Darüber hinaus enthält es Protokollelemente für die Kommunikation zwischen den UNICORE-Komponenten: Schickt eine Komponente ein AJO mit einem gewissen Inhalt (z.B. führe den enthaltenen Job aus), so wird das zugehörige Antwort-AJO (z.B. erfolgreich/nicht erfolgreich ausgeführt) zurück erwartet.

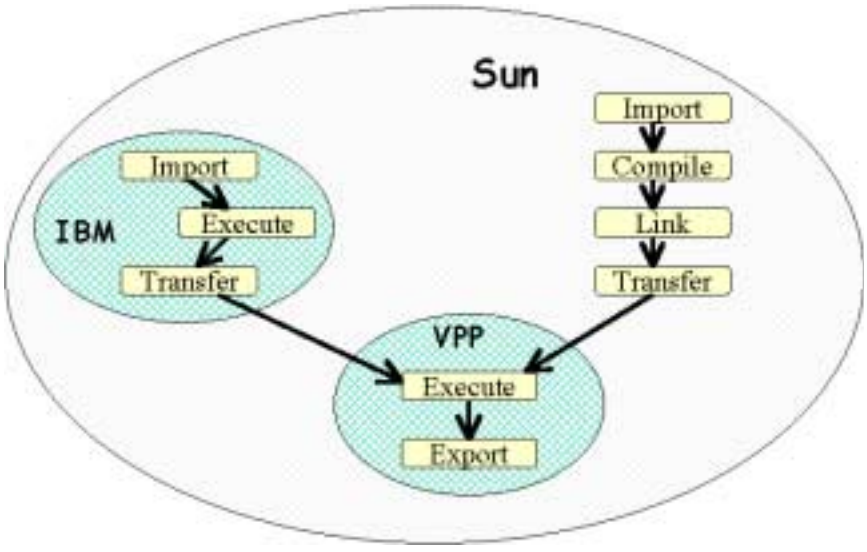

Abbildung 1 Beispiel für ein Abstraktes Job Objekt (AJO)

Folgendes Beispiel (siehe Abbildung 1) soll den Aufbau eines AJO und damit den eines Ujobs verdeutlichen. Das äußere Oval enthält das gesamte AJO, welches für das primäres Zielsystem Sun bestimmt ist. Hier soll eine Cross-Compilation für ein anderes Zielsystem durchgeführt werden, welche aus den Tasks *Import* (importiere Daten in den Uspace), *Compile* (Kompiliere Quelldateien), *Link* (Linke die Objekte) und *Transfer* (Transferiere das erzeugte ausführbare Programm) besteht. Unabhängig von diesem Schritt werden auf einem anderen Zielsystem (IBM) die Eingabedaten vorbereitet. Dieses AJO (linkes mit IBM gekennzeichnetes Oval) enthält die

Schritte *Import* (Importiere Ursprungsdaten), *Execute* (Bearbeite Daten) und *Transfer* (Transferiere aufbereitete Daten). Auf einem dritten Zielsystem (VPP) sollen nun, wenn die beiden Vorgängerschritte erfolgreich beendet worden sind, die vorverarbeiteten Daten mit dem erzeugten ausführbaren Programm verarbeitet (*Execute*) und die Ergebnisse auf permanenten Speicherplatz exportiert (*Export*) werden.

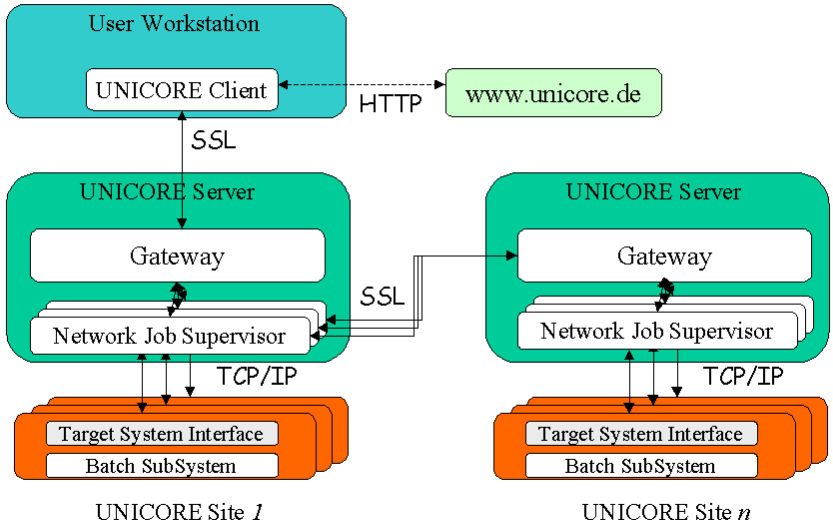

Abbildung 2 Die UNICORE Drei-Ebenen-Architektur

6. Die UNICORE Architektur

Die Architektur besteht aus Benutzer-, UNICORE- und Systemebene. Abbildung 2 gibt einen Überblick über die Komponenten und ihr Zusammenspiel. Auf der Benutzerebene, der Workstation des Benutzers, läuft das graphische Interface zur Erstellung und Kontrolle von UNICORE-Jobs. Von dem zentralen Server www.unicore.de wird die Liste der verfügbaren UNICORE Sites abgefragt. Über das SSL-Protokoll (Secure Socket Layer, siehe [19]) wird die UNICORE Server Ebene angesprochen. Diese besteht aus einem Gateway, welches die Benutzer-Authentifizierung durchführt, und Network Job Supervisor (**NJS**) Servern, die jeweils die Aufträge (Jobs, Statusabfragen) für eine Vsite bearbeiten. Auf der System-Ebene sorgt die Komponente Target System Interface (**TSI**) für die Schnittstelle zum lokalen Betriebssystem und Batch-Subsystem. Enthält ein AJO wie z.B. das in Abbildung 1 Teil-AJOs für andere Zielsysteme an anderen Usites, so werden diese über SSL an die entsprechenden Gateways übergeben.
Einen detaillierteren Blick auf die UNICORE-Architektur zeigt Abbildung 3, insbesondere ist hier die Sicherheitsarchitektur (siehe auch [1],[2]) dargestellt. Den Kern der Sicherheitsarchitektur bilden die X.509V3 Zertifikate der Benutzer, Gateways und NJS Server. Das Benutzer-Zertifikat wird verschlüsselt im Heimatverzeichnis des Benutzers auf dem Arbeitsplatzrechner abgelegt und von der Benutzer-Interface-Anwendung verwaltet. Das SSL-Protokoll als Bestandteil der Sicherheitsarchitektur sorgt zusammen mit den X.509-Zertifikaten für die gegenseitige Authentifizierung der beteiligten Komponenten und für die sichere Kommunikation ([9]). Der öffentliche Teil des Benutzer-Zertifikats ist gleichzeitig die einheitliche, zentrenübergreifende UNICORE-Benutzeridentifikation, die zunächst vom Gateway geprüft wird. Das Gateway akzeptiert nur Anfragen von Klienten mit gültigem UNICORE-Zertifikat.
Das Benutzer-Interface bietet den Job Preparation Agent (**JPA**) zum Erstellen und Submittieren von Jobs und den Job Monitor Controller (**JMC**) zur Job-Kontrolle und Output-Bearbeitung an. Beide Komponenten liefern eine einheitliche Sicht auf die in UNICORE zur Verfügung stehenden Zielsysteme. Über die Oberfläche werden die abstrakten Jobs bzw. Status-Abfragen als AJOs erzeugt und weitergeleitet. Um zu verhindern, daß auf dem Weg zum Ziel-NJS ein AJO unberechtigt manipuliert werden kann, wird jedes AJO (d.h. auch alle Teil-AJOs) mit dem privaten Schlüssel des Benutzers signiert.
Das Gateway hat die Aufgabe der Benutzer-Authentifizierung. Für den Fall, daß eine Usite zusätzliche Authentifizierung ihrer Benutzer für den Zugang z.B. über Securid cards oder DCE benötigt, sorgt das Gateway dafür, daß die notwendigen Informationen vom Benutzer abgefragt und vom Interface in das AJO geschrieben werden.
Das AJO wird vom Gateway an den primären Network Job Supervisor weitergegeben. NJS hat die Aufgabe, das für ihn bestimmte AJO auszupacken (mit Hilfe des öffentlichen Schlüssels des Benutzers), es zu analysieren und nicht für ihn bestimmte enthaltene AJOs zeitlich dem Abhängigkeitsgraphen entsprechend an die zugehörigen

Gateways unverändert weiterzuleiten. Für ihn bestimmte Teile versieht er mit der lokalen Benutzernummer über das **userid mapping.** Dieses ermittelt die zum Benutzer-Zertifikat gehörende Userid aus der lokalen Benutzerdatenbank. Dann übersetzt NJS die abstrakt formulierten Jobs in konkrete Batch-Jobs für das Zielsystem (**Incarnation**) und gibt sie dem Abhängigkeitsgraphen entsprechend an das TSI weiter. Desweiteren stellt der NJS die Ressourcen-Informationen zu der von ihm verwalteten Vsite zur Verfügung. Diese wird vom Benutzer-Klienten dazu verwendet, den User bei der Job-Erstellung so zu unterstützen, daß nur ein für das Zielsystem geeigneter Job zusammengestellt werden kann.

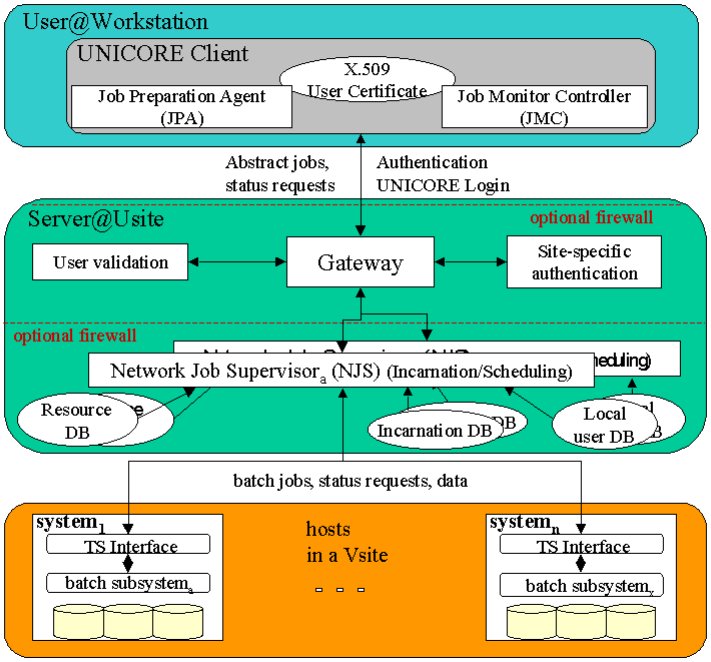

Abbildung 3 Detaillierte Architektur

Die UNICORE Server Ebene ist derart gestaltet, daß sie mit Firewalls zusammen eingesetzt werden kann. Mit SSL wurde ein Standard-Protokoll gewählt und das Gateway akzeptiert nur Verbindungen zu einem definierten Port. Damit kann diese Verbindung von außen gefahrlos von einer Firewall akzeptiert werden. Gateway und NJS sind nicht-privilegierte Java-Applikationen, die über feste, vom Rechenzentrum definierbare Sockets kommunizieren. Damit ist es auch möglich, eine Firewall zwischen Gateway und NJS Servern einzurichten.
Aufgabe des Target System Interface (TSI) ist es, den Uspace zu verwalten, die Jobs zum lokalen Batch Subsystem weiterzuleiten und regelmäßig den Job-Status zu überprüfen. Die Kommunikation zwischen TSI und NJS erfolgt über eine feste Socket-Verbindung, die von Seiten des TSI aufgebaut wird. Das Target System Interface läuft als privilegierter Prozeß, um für den Benutzer handeln zu können.
Die Sicherheitsarchitektur wird durch eine Public Key Infrastructure unterstützt, die für die Erstellung und Verwaltung der UNICORE Zertifikate sorgt. Die zugehörige „Certification Authority“ (CA) ist beim LRZ angesiedelt, alle anderen UNICORE-Zentren fungieren als „Registration Authority“ (RA). Die Zertifizierungs-Policy basiert auf den vom DFN-PCA (Deutsches Forschungsnetz – Policy Certification Authority) herausgegebenen Empfehlungen ([5]).

7. Die Benutzerfunktionen

Dem Benutzer wird durch die oben beschriebene Architektur eine einheitliche Benutzer-Schnittstelle zu (Super-) Computern angeboten. Unabhängig vom jeweiligen Zielsystem werden Ressourcen-Anforderungen und Kommando-Optionen über eine graphische Oberfläche angegeben. Der JPA erzeugt aus den Angaben den abstrakten Job (das AJO) und später wird dieser vom NJS in die konkreten Befehle und Optionsangaben für das vom Benutzer angewählte Zielsystem übersetzt. Das X.509-Zertifikat des Benutzers dient als einheitliche UNICORE-Benutzeridentifikation. Der Prototyp unterstützt derzeit die Erstellung und die Überwachung von Batch-Anwendungen. Die UNICORE-Jobs können aus mehreren Teilen aufgebaut werden, die asynchron auf verschiedenen Rechnern an verschiedenen Rechenzentren ablaufen können. Der Benutzer kann die Teile mit sequentiellen Abhängigkeiten verbinden (siehe Abbildung 4).

Das in UNICORE verwendete Datenmodell beinhaltet, daß für jeden UNICORE-Job ein Job-Directory (Uspace) angelegt wird, welches das Basisdirectory während der Job-Ausführung auf dem Zielsystem ist. Benötigte Daten müssen jeweils in diesen Datenraum importiert bzw. aus ihm exportiert werden, wenn die Daten über das Job-Ende hinaus erhalten bleiben sollen. Zwischen Job-Schritten, die auf unterschiedlichen Zielsystemen ablaufen sollen, müssen die zu transferierenden Dateien in speziellen Transfer-Tasks spezifiziert werden. UNICORE sorgt für den Datentransfer, der transparent für die Benutzer abläuft.

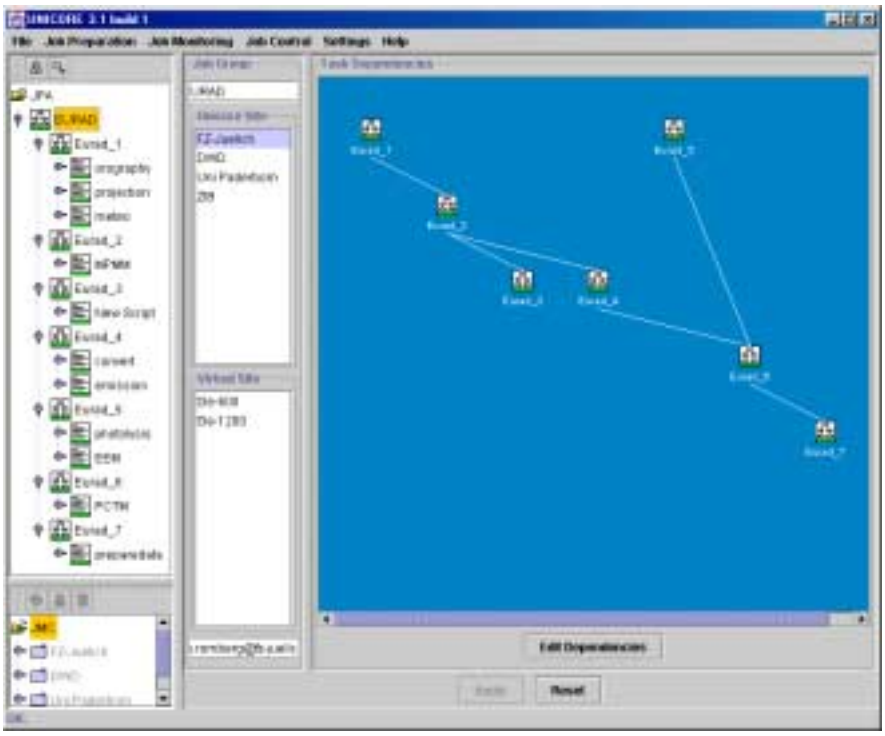

Abbildung 4 UNICORE-Job für das EURAD Modell

7.1. Erstellen von Jobs

Über den Job Preparation Agent (JPA), das graphische Benutzerinterface zur Erstellung und Submission von UNICORE-Jobs, können derzeit Jobs aus den folgenden Elementen zusammengesetzt werden:

- Script Task, über die bereits bestehende Job-Scripts einbezogen werden können,
- Compile-Link-Run Task für das Einbinden neuer Applikationen,
- Transfer Task für den Datentransfer zwischen Job-Schritten und
- Job-Gruppe, die rekursiv Teil-Jobs für ein anderes Zielsystem enthält.

Ein neuer UNICORE-Job wird im JPA über die Auswahl *New Job* im *File*-Menü erstellt. Mit dieser Funktion erzeugt der Benutzer den äußeren Rahmen für den UNICORE-Job, der allgemeine, für den gesamten Job relevante Information wie

- das Zielsystem, das aus einer Liste der in UNICORE verfügbaren Zielsysteme ausgewählt wird,
- den Jobnamen und
- die email-Adresse des Benutzers, an die Nachrichten vom System geschickt werden sollen,[1]

enthält. Im GUI wird der Job-Aufbau durch ein Baum-Symbol im linken Teil des Fensters angezeigt versehen mit dem gewählten Jobnamen. Die benötigten allgemeinen Informationen gibt der Benutzer im mittleren Teil des Fensters an. Der rechte Teil des GUI-Fensters ist zunächst leer. Über das *Job Preparation*-Menü kann der Benutzer die Elemente auswählen, die auf dieser Ebene des Jobs hinzugefügt werden sollen. Bei Auswahl einer Job-Gruppe kann der Benutzer die Job-Attribute des UNICORE-Jobs für diesen Job-Teil ändern, z.B. ein anderes Zielsystem angeben. Wählt der Benutzer die Script-, Transfer- oder die Compile-Link-Run Task aus, so werden die Komponenten-Symbole auf der linken Seite des GUI-Fensters mit Verbindung zum zugehörigen UNICORE-Job- oder Job-Gruppen-Element hinzugefügt. Rechts erscheinen die Element-Symbole einer Job-

[1] Über *Settings* können Standardeinstellungen für diese Werte gesetzt werden.

Ebene zunächst ohne Struktur. Hier werden die Symbole in einem gerichteten Graphen positioniert, wenn der Benutzer die Abhängigkeiten zwischen den Komponenten definiert.
Für die Elemente Script-, Transfer- und Compile-Link-Run Task sind die benötigten Ressourcen zu spezifizieren. Zu den Ressourcen gehören:

- die Anzahl der Nodes,
- die Anzahl der Prozessoren pro Node,
- der Hauptspeicherplatzbedarf,
- die CPU-Zeit und
- der Plattenplatzbedarf[1].

Hierbei werden jeweils die minimal und maximal möglichen Werte für das Zielsystem mit angezeigt, so daß der Benutzer keine Werte außerhalb des zulässigen Bereichs anfordern kann und auch sofort sieht, ob ein angewähltes Zielsystem die benötigten Ressourcen überhaupt anbietet.
Jedes Job-Komponenten-Symbol ist mit einem Farbfeld versehen, über das angezeigt wird, ob alle für die Submission notwendigen Informationen vorhanden und für das Zielsystem passend sind. Dabei steht rot für *noch nicht ok* und grün für *ok.* Nur wenn alle Komponenten und damit der ganze UNICORE-Job grün gekennzeichnet sind, kann der Job submittiert werden; der entsprechende Befehl wird erst dann im Interface aktiviert und ausführbar. Das hat den Vorteil, daß nur Jobs, die von den Zielsystemen auch akzeptiert werden, submittiert werden können. Der Benutzer kann Fehler frühzeitig erkennen und beheben.
Die Abbildung 4 zeigt die graphische Oberfläche des JPA mit einem bereits erstellten Job. Auf der linken Seite sieht man die gesamte Job-Struktur, während in der Mitte die zur Job-Gruppe gehörenden allgemeinen Angaben und auf der rechten Seite der Graph der Elemente mit ihren sequentiellen Abhängigkeiten auf der obersten Job-Ebene gezeigt werden. Das Beispiel ist eine komplexe Anwendung aus dem Eurad-Projekt ([7], Europäisches Ausbreitungs- und Depositionsmodell), das sich mit der (Schad-) Stoff-Ausbreitung in der Atmosphäre befaßt. Die Anwendung besteht aus einer Reihe von Schritten, die zum Teil auf Vektorrechnern (CRAY T90) und zum Teil auf massiv-parallelen Systemen (CRAY T3E) abläuft. Bisher werden alle Schritte sukzessive gestartet und überprüft bevor der jeweils nächste Schritt gestartet wird. Über UNICORE kann in Zukunft die ganze Anwendung in einem Job zusammengefaßt werden und automatisch ablaufen und damit dem Benutzer Synchronisations- und Überwachungsarbeit abnehmen.

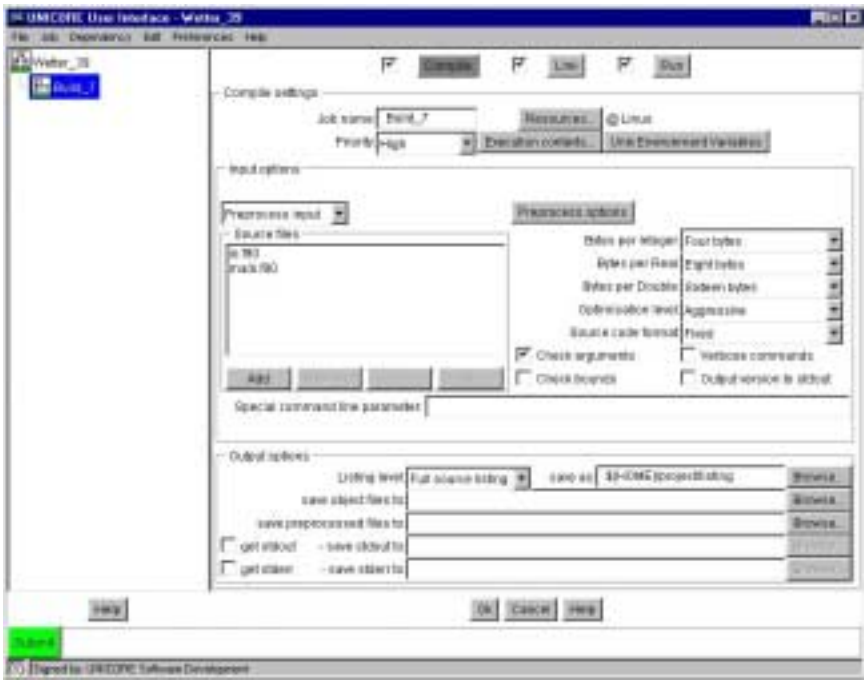

Abbildung 5 Beispiel für das Compile User Interface

Ein Blick auf die Compile-Link-Run Task zeigt insbesondere die Mächtigkeit von UNICORE (siehe Abbildung 5, Abbildung 6)[2]. Diese Job-Komponente dient dazu, neue Anwendungen zielsystemunabhängig zu erstellen. Sie

[2] Die Graphiken zeigen noch die UNICORE Version, in der das Benutzer-Interface über ein signiertes Java-Applet realisiert wurde.

besteht intern aus drei Teilen, nämlich Compile-, Link- und Run-Teil, von denen jedes einzeln, beliebige Paare oder alle drei gemeinsam zum Jobaufbau benutzt werden können. Die Auswahl dazu ist oben im rechten Teil des graphischen Interfaces für die CLR-Task in der Form von Checkboxes gegeben. Wählt man im linken Teil des GUI eine CLR-Komponente aus, erscheint auf der rechten Seite des Fensters das Eingabe-Fenster für die Compile Task (siehe Abbildung 5). Der Eingabe-Bereich ist in vier Zonen geteilt: die oben genannten Checkboxes, ein allgemeiner Teil mit Namen-, Ressourcen-, Kontextangaben, u.a., ein Bereich für Input-Daten wie Source-Dateien und Compiler-Optionen und ein Bereich für Output-Daten. Wesentliche Bereiche für abstrakte, systemunabhängige Job-Definitionen, sind die *Resources*, die *Execution contexts*, *Preprocess options* und die Compiler-Optionen wie *optimisation level*, *check arguments*, *check bounds*, *listing level*, und andere. Da gibt der Benutzer zum Beispiel für die Anzahl der benötigten Prozessoren für seinen parallelen Job 12 an, was im AJO zum NJS transportiert wird und dort für ein Zielsystem mit Batch-Subsystem CRAY NQS in die qsub-Option *–l mpp_p=12* übersetzt wird und für ein Zielsystem mit Batch-Subsystem Codine in die qsub-Option *–pe <parallel_environment> 12*. Die Art des parallelen Programmiermodells (z.B. MPI) wählt der Benutzer bei den *Execution contexts* aus. Das wird im obigen Beispiel für CRAY NQS nicht als qsub-Option übersetzt, sondern bei den Bibliotheken und z.B. bei der Programmausführung über mpprun berücksichtigt, während es bei Codine auch in die qsub-Option *–pe mpi 12* übersetzt wird. Analog werden die ausgewählten Compiler-Optionen auf das jeweilige Zielsystem abgebildet – völlig transparent für den Benutzer, egal welches Zielsystem er ausgewählt hat, seine Angaben im JPA sind immer dieselben. So wird die Benutzerangabe *Optimisation level agressive* aus dem Interface bzw. dem AJO vom NJS für den Fortran Compiler auf einer CRAY T3E in die f90 Option *–O3,unroll2* übersetzt, für eine NEC SX-4 in die f90 Optionen *–Wf,-dir par –P auto –C hopt –pi –Wf, -pvctl,fullmsg,vwork=stack,noassume,loopcht=10000000* .
Das GUI für den Run-Teil (siehe Abbildung 6) ist genauso gegliedert wie das für Compile; man kommt zu der Oberfläche, indem man den *Run*-Knopf in der Kopfzeile anschaltet. Dateien, die als Input-Daten angegeben werden, werden in das UNICORE-Job-Directory importiert, Output-Daten werden entsprechend exportiert. Das auszuführende Programm (*Executable*) kommt entweder aus dem Link-Schritt vorher und muß damit nicht angegeben werden, oder wird explizit angegeben und, wenn nötig, importiert.

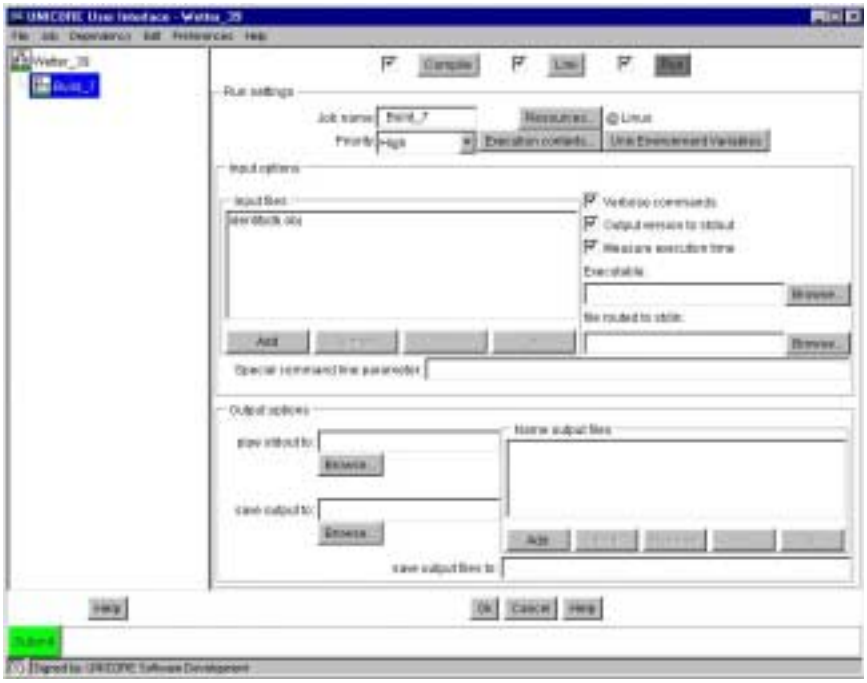

Abbildung 6 Beispiel für das Run User Interface

Sehr wichtig für die Akzeptanz von UNICORE ist es, daß Benutzer auch bereits bestehende Job-Scripts über UNICORE abschicken können. Für diesen Zweck gibt es die Script Task Komponente (siehe Abbildung 7). Neben Jobnamen und Ressourcen-Anforderung gibt der Benutzer das Job-Script an, das entweder lokal auf seiner Workstation liegt oder von einem anderen System importiert werden soll. Das Script kann in dem GUI noch angepaßt und lokal abgespeichert werden.

Unter UNICORE erstellte Jobs können auf der lokalen Workstation gespeichert (über die Auswahl *Save Job* bzw. *Save Job as* im *File*-Menü) und später wieder in den JPA geladen werden (über die Auswahl *Open Job* im *File*-Menü). Diese Jobs können modifiziert werden indem Komponenten gelöscht, hinzugefügt oder abgeändert werden. Zum Beispiel können auch Ressourcen-Anforderungen geändert werden: Ein Benutzer hat bisher Programmentwicklung gemacht und nur kurze Testläufe benötigt. Nun will er für den Produktionslauf die CPU-Zeit und/oder die Anzahl der Prozessoren heraufsetzen. Dieses kann der Benutzer über das *Edit Resources* Panel der entsprechenden Job-Komponente angeben. Soll der UNICORE-Job auf einem anderen Zielsystem ausgeführt werden, wählt der Benutzer auf der betroffenen Job-Ebene die neue Usite und Vsite aus den entsprechenden Listen aus. UNICORE zeigt dem Benutzer alle Ressourcen-Angaben, die durch die Änderung des Zielsystems nun gegebenenfalls nicht mehr passen. Damit hat der Benutzer sofort die Möglichkeit die Angaben der neuen Auswahl anzupassen oder das Zielsystem als nicht geeignet für die Anwendung zu verwerfen und eine andere Lösung zu suchen.

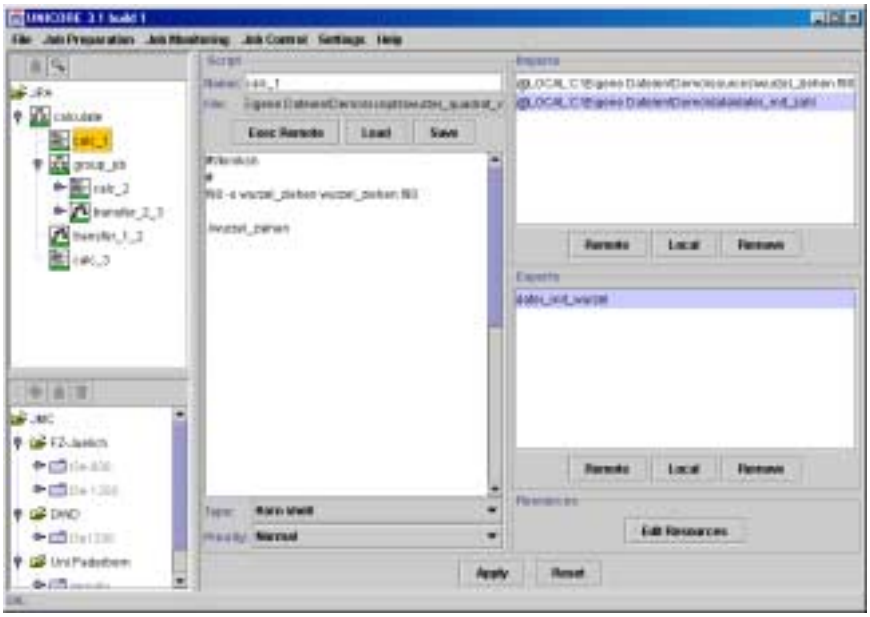

Abbildung 7 Beispiel für das Script Task User Interface

7.2. Überwachen von Jobs

Der Job Monitor Controller (JMC) ist der Teil des UNICORE-Benutzer-Interfaces, über das der Status der UNICORE-Jobs überwacht wird. Die Statusanzeige belegt die untere Hälfte des linken Teilfensters. Dort wird die Liste der Usites mit ihren Vsites und den vom Benutzer an UNICORE abgeschickten und noch aktiven Jobs angezeigt. Über Farben wird der jeweilige Job-Status des UNICORE-Jobs und seiner Komponenten verdeutlicht. Grün bedeutet dabei *erfolgreich beendet*, rot steht für *mit Fehler beendet* bzw. *killed*, gelb zeigt *executing* an, blau *queued* und grau bedeutet *undefined*. Im Fall von *executing*, ein Job wird aus UNICORE Sicht gerade ausgeführt, wird der Farbbalken zweigeteilt und in der zweiten Hälfte wird der Status aus Sicht des Batch-Subsystems auf dem Zielrechner gezeigt: blau für *queued* und gelb für *running*. Im rechten Teilfenster ist die Jobstruktur der auf der linken Seite aus einem Job ausgewählten Ebene zu sehen mit den definierten Abhängigkeiten. Darüber wird klarer, warum z.B. eine Komponente des Jobs noch *queued* ist: Mindestens eine Vorgängerin im Abhängigkeitsgraphen wurde noch nicht beendet.
Als Beispiel ist in Abbildung 8 der detaillierte Status einer Script Task dargestellt. Auf dieser Ebene erhält der Benutzer für den auf dem Zielsystem ausgeführten Job die Informationen aus der Standard-Ausgabe- und der Standard-Fehler-Datei, die er bei Bedarf abspeichern kann.
Neben der Überwachung des Job-Status kann der Benutzer UNICORE-Jobs löschen, wobei dann automatisch alle zugehörigen Job-Komponenten auf den Zielsystemen gelöscht werden. Je nach Anforderung bleiben die Standard-Dateien erhalten oder werden sofort gelöscht.

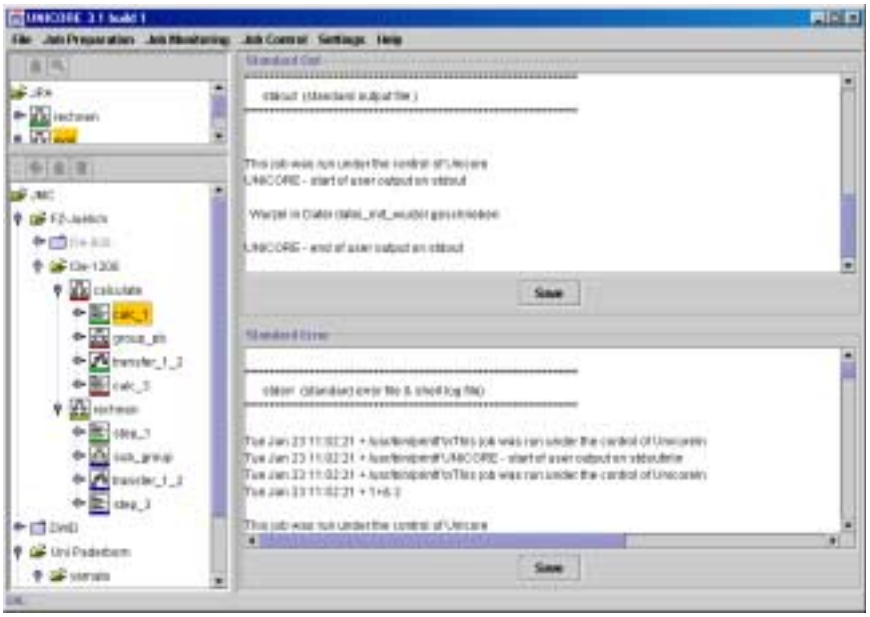

Abbildung 8 Beispiel für die Job-Status Anzeige

8. Status und Ausblick

Im 'UNICORE Plus' Projekt wird in der ersten Phase die gegenüber dem Vorgänger-Projekt geänderte Architektur implementiert. Während sich im UNICORE-Projekt die Komponenten um einen https-Server und ein bestehendes Batch-System als Grundlage für den Network Job Supervisor gruppierten ([17]), sind nun alle Komponenten speziell für die UNICORE-Anforderungen neu entwickelt. Sie sind dadurch schlanker und für das Projekt leichter wartbar. Bis auf das Target System Interface, das in Perl geschrieben ist, sind alle Komponenten Java-Applikationen. Zur Zeit (Januar 2001) laufen die neuen Komponenten für die Zielsysteme Cray T3E, Siemens hpcLine und Fujitsu VPP beim DWD, Forschungszentrum Jülich, LRZ, PC² und ZIB. Die Integration der Zielsysteme Hitachi SR 8000, IBM SP2 und NEC SX-5 erfolgt in den nächsten Wochen. An Benutzerfunktionen bietet das jetzige System die in Kapitel 7 beschriebenen:

- Aufbau eines Jobs mit mehreren sequentiell abhängigen oder unabhängigen Schritten für verschiedene Zielsysteme;
- Datentransfer über „data streaming" zwischen Benutzerworkstation und Zielsystem und zwischen den verschiedenen Zielsystemen;
- Script, Compile-Link-Run und Transfer Task;
- Sequentielle Abhängigkeiten;
- Job-Status Anzeige über Farben;
- Bereitstellen von Standard-Output und –Error Dateien.

Für die Public Key Infrastructure wurde die Policy festgelegt und die Certification Authority beim LRZ in München aufgebaut. Die Prozeduren für die Registration Authorities werden zur Zeit entwickelt.

Das UNICORE System (Version 3.0) soll neben den Zentren der Projekt-Partner auch bei den Partnern des geplanten Europäischen Projekts EUROGRID ([3]) zum Einsatz kommen. Für EUROGRID ist geplant, UNICORE als Grundlage für die Entwicklung von Grid-Anwendungen einzusetzen.

Im Projektverlauf gibt es zwei weitere Meilensteine jeweils zur Mitte des Kalenderjahres, zu denen dann eine neue UNICORE System-Version mit jeweils erweiterter Funktionalität fertiggestellt wird und an den Zentren zum Einsatz kommt. Die Weiterentwicklungen kommen aus den 'UNICORE Plus' Teilprojekten, die in Kapitel 3 aufgeführt sind. Im Bereich der anwendungsspezifischen Benutzer-Oberflächen wird für die CPMD Anwendung (Car-Parrinello Molecular Dynamics) ein Interface entworfen, über das automatisch die benötigte Konfigurationsdatei erzeugt und das zugehörige AJO erstellt und submittiert wird. Der UNICORE Server wird dahingehend erweitert, daß CPMD-Jobs für das ausgewählte Zielsystem inkarniert werden können. Parallel dazu laufen Entwicklungen, die bereits bestehenden graphischen Oberflächen für FLUENT, NASTRAN und STAR-CD als anwendungsspezifische Interfaces in UNICORE zu integrieren. Darauf aufbauend wird in einem nächsten Schritt ein generisches Interface entworfen, über das sich neue Anwendungspakete leicht integrieren lassen.

Benutzeranwendungen sollen auch dadurch besser unterstützt werden, daß zusätzlich zu sequentiell abhängigen Job-Schritten auch Abhängigkeiten für Job-Ketten (while/repeat) und Verzweigungen (if-then-else/case)

einbezogen werden. Damit können z.B. Simulations-Testreihen automatisiert werden. Weiterhin wird die parallele Ausführung von Anwendungsprogrammen bzw. einzelner Anwendungen auf verschiedenen Systemen unterstützt. Letzteres ist Voraussetzung für die Entwicklungen im Bereich Metacomputing. Hier stehen Arbeiten zu Mechanismen zur verteilten Ausführung von Anwendungsprogrammen und dem Scheduling im Vordergrund. Zur Performance Analyse der Metacomputing-Anwendungen werden Werkzeuge zur Visualisierung ihres Laufzeitverhaltens erstellt.
Das Ressourcen-Modell im UNICORE-Projekt beinhaltete eine kleine Gruppe statischer Ressourcen. Das neue Modell ist XML-basiert und wird erheblich erweitert, zum einen zur Abbildung von dynamischen Ressourcen wie der Systemauslastung und zum anderen zur Beschreibung einer umfassenden Menge von Hardware und Software-Ressourcen ([16]). Insbesondere wird das Modell die Anforderungen des Metacomputing (z.B. Co-Scheduling) berücksichtigen.
Ein Schwerpunkt der Entwicklungen liegt beim Datenmanagement: Große Datenmengen (mehrere Giga-Bytes) müssen für UNICORE-Jobs bzw. zwischen Job-Schritten transportiert werden. Das bezieht auch den Anschluß an Archivsysteme sowie an Systeme des verteilten Datenmanagements wie HPSS mit ein. Darüberhinaus wird eine Benutzer-Schnittstelle entwickelt, die Filetransfer-Funktionen unter Verwendung der UNICORE Sicherheitsmechanismen unabhängig von einem UNICORE-Job anbietet.
Im UNICORE-Projekt haben die Themen Software-Verwaltung, Qualitätssicherung und Administrierbarkeit gegenüber den Benutzer-Funktionen zurückgestanden. Die Akzeptanz von UNICORE in einem Rechenzentrum, welches das System einsetzen will, hängt aber ganz entscheidend von diesen Punkten ab. Die dazu notwendigen Tools wie z.B. für Software-Maintenance, Konfiguration, Job-Verfolgung und Fehleranalyse werden nun im 'UNICORE Plus' Projekt schrittweise entwickelt.
Die Stärke des UNICORE-Systems liegt in der Abstrahierung der Benutzer-Jobs im AJO gepaart mit einer auf Standards aufbauenden Architektur. Durch ersteres kann den Benutzern ein seamless Interface geboten werden und damit der problemlose Wechsel zwischen Rechenzentren und Zielsystemen. Dieses wurde bereits im UNICORE-Projekt erreicht und erfolgreich demonstriert. Die inhärente Sicherheit und die weiteren Entwicklungen im 'UNICORE Plus' Projekt wird eine leistungsfähige Infrastruktur für das Grid Computing geschaffen, die den Anforderungen von Benutzern und Rechenzentren gleichermaßen gerecht wird.

9. Referenzen

[1] Jim Almond, *Seamless Computing: Einheitlicher Zugriff auf Supercomputer über das Web*, FOKUS Praxis Information und Kommunikation, Band 16, Hans-Werner Meuer (Hrsg.), Supercomputer 1998, K.G.Saur Verlag München

[2] Jim Almond, Dave Snelling, *UNICORE: uniform access to supercomputing as an element of electronic commerce*, Future Generation Computer Systems FGCS Vol. 15 (1999), Numbers 5-6, October 1999, pp.539-548

[3] Application Testbed for European GRID Computing (EUROGRID), IST-1999-20247, http://www.eurogrid.org

[4] Arbeitsgruppe zur Erstellung der VESUZ-Machbarkeitsanalyse, *Verbund der Supercomputer-Zentren in Deutschland –eine Machbarkeitsanalyse–*, Oktober 1997, BMBF-Förderkennzeichen 01 IR 602/9

[5] DFN-PCA http://www.pca.dfn.de/dfnpca

[6] Dietmar Erwin (Editor), *UNICORE – Uniformes Interface für Computer Ressourcen*, Gemeinsamer Abschlußbericht des BMBF-Verbundprojekts 01 IR 703, ISBN 3-00-006377-3

[7] Das EURAD Projekt an der Universität zu Köln, http://www.uni-koeln.de/math-nat-fak/geomet/eurad/index.html

[8] European Grid Forum http://www.egrid.org

[9] Jalal Feghhi, Jalil Feghhi, Peter Williams, *Digital Certificates*, Addison Wesley,1998

[10] Ian Foster, Carl Kesselman *Computational Grids* in The Grid: Blueprint for a New Computing Infrastructure by Ian Foster and Carl Kesselman (Eds.), Morgan Kaufman Publishers, 1998

[11] Ian Foster, Carl Kesselman *Globus: A Metacomputing Infrastructure Toolkit* in Intl. J. Supercomputer Applications, 11(2): 115-128, 1997

[12] Grid Forum http://www.gidforum.org

[13] Andrew S. Grimshaw, Wm. A. Wulf, and the Legion team *The Legion Vision of a Worldwide Virtual Computer* in Communications of the ACM Vol.40, No.1, January 1997, pp.39-45

[14] U. Kleinau, J. Schulz, D. Tavangarian, *Eine Architektur für Hochleistungs-Datenverarbeitung in Weitverkehrs-Workstation-Netzwerken*, 15. GI/ITG Fachtagung Architektur von Rechnersystemen (arcs'99), Jena, 4.-7.10.1999

[15] Ryan McCormack, John Koontz, and Judith Devaney *WebSubmit: Web-based Applications with Tcl* in NISTIR 6165, June, 1998

[16] A. Reinefeld, H. Stüben, W. Baumann, *Models for Specifying Distributed Computer Resources in UNICORE*, in ISThmus 2000, Research and Development for the Information Society, Conference Proceedings, Instytut Informatyki, Politechnika Poznánska, Poznan, 2000. ISBN 83-913639-0-2
[17] Mathilde Romberg, *The UNICORE Architecture: Seamless Access to Distributed Resources*, in Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing, August 1999, pp.287-293
[18] Mathilde Romberg, *The UNICORE Grid Infrastructure*, Conference Proceedings of the SGI Users' Conference (SGI2000), Kraków, Poland, 11.-14.10.2000, pp.144-153, ISBN 83-902363-9-7
[19] Secure Socket Layer (SSL) Beschreibung http://developer.netscape.com/docs/manuals/security/sslin/index.htm
[20] Dave Snelling, *The Abstract Job Object: An Open Framework for Seamless Computing*, Vortrag beim 1.DATORR-Treffen, 8.10.98, Argonne National Laboratory, http://www.fz-juelich.de/unicore/DARR.ps.gz
[21] UNICORE Projekt http://www.fz-juelich.de/unicore
[22] 'UNICORE Plus' Projekt http://www.fz-juelich.de/unicoreplus or http://www.unicore.de
[23] UNICORE Forum e.V. http://www.unicore.org