WissGrid

Dokumentation

Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten

Formatkonvertierung NetCDF → ASCII und Einbindung als Grid-Job[1]

Autoren	Arbeitspaket 3: Langzeitarchivierung von Forschungsdaten
Editoren	Rathmann
Datum	14. April 2011
Dokument Status	Entwurf
Dokument Version	0.2.2

Änderungen

Version	Datum	Name(n)	Kurzinfo
0.1.0	14.03.2011	Henne, Panzer, Samadi	Template
0.2.0	07.04.2011	Rathmann	Software-Dokumentation
0.2.1	11.04.2011	Rathmann	Ergänzungen
0.2.2	14.04.2011	Rathmann, Fritzsch	Ergänzungen

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

Inhaltsverzeichnis

1 Anforderungen

- Formatkonvertierung NetCDF → ASCII
  - ASCII-Ausgabe als zweidimensionales Array von Zahlen
  - importierbar in Microsoft Excel
  - Anleitung zum Datenimport in Excel
- Einbindung als Grid-Job im C3Grid (1)

Im Rahmen von WissGrid ist eine Formatkonvertierung NetCDF → ASCII zur Verfügung gestellt und im C3Grid als Workflow (Grid-Job) zugänglich gemacht worden. Die Anforderungen wurden zuvor gemeinsam von WissGrid und dem Climate Service Center (CSC) (2) des Helmholtz-Zentrums Geesthacht entwickelt. Das CSC hat als Serviceeinrichtung für die Klimafolgenforschung gute Kontakte zur Community und konnte uns Auskunft geben, was in der Klimafolgenforschung gebraucht wird.

Die Klimafolgenforschung beschäftigt sich mit den Folgen des Klimawandels auf der Erde, insbesondere mit den künftigen Auswirkungen auf Ökosysteme, Gesundheit, Landwirtschaft und Wirtschaft (z.B. Versicherungswesen, Tourismus, Küstenschutz). Die Community nutzt dafür in der Regel vorhandene Klimadaten nach. Aus dem CSC haben wir erfahren. dass sehr häufig Ergebnisse von Klimamodellrechnungen in Microsoft Excel eingeladen werden, vorzugsweise als zweidimensionales Array. Neben der eigentlichen Formatkonvertierung wurde deshalb auch eine schriftliche Anleitung für den Datenimport in Excel gewünscht.

Im C3Grid sind viele Ergebnisse von Klimamodellrechnungen kostenlos zugänglich. Dort liegen die Daten überwiegend in den Binärformaten NetCDF (3) und GRIB (4) vor, die in der Klimaforschung weit verbreitet sind. In der Klimafolgenforschung werden die Daten — wie oben erwähnt — hingegen oft in einem Format benötigt, das sich in Excel importieren lässt. Ein Format, in dem der Import in der gewünschten Forma als Array gelingt, ist spaltenformatiertes ASCII, d.h. Zahlenkolonnen im einfachen Textformat. Eine Formatkonvertierung von GRIB nach NetCDF ist im C3Grid bereits vorhanden. Diese Konvertierung arbeitet ohne Informationsverlust, so dass es ausreicht, eine Konvertierung von NetCDF nach ASCII zur Verfügung zu stellen. Die Konvertierung von GRIB nach ASCII ist dann als Hintereinanderausführung GRIB → NetCDF → ASCII durchführbar.

2 Design

2.1 Das Konvertierungsskript ascii.py

Das Modul für die Konvertierung NetCDF → ASCII ist als Python-Skript realisiert worden, welches seinerseits die folgende Standard-Software aufruft:

- `ncdump` aus der NetCDF-Library (3) (Hintergrundinformation in Abschnitt 6.1)
- CDO (Climate Data Operators) (5)
- `zip` (Archivierung mit Kompression)

Die beiden ersten Produkte nehmen die eigentliche Formatkonvertierung vor. Dementsprechend können NetCDF 3 bzw. 4 Classic konvertiert werden.

Das Python-Skript `ascii.py` erwartet als Input eine NetCDF-Datei. Die NetCDF-Datei wird auf zwei Wegen in ASCII konvertiert:

1. In Spalten fester Breite. Dabei wird mit Hilfe des CDO `splitname` zunächst für jede Datenvariable eine eigene NetCDF-Datei erzeugt, die nur noch diese eine Datenvariable enthält. In einem zweiten Schritt werden die zu dieser Datenvariable gehörenden Daten als Zahlenkolonnen fester Breite in eine ASCII-Datei geschrieben. Dies wird durch Anwendung des CDO `outputf` auf jede der mit `splitname` erzeugten NetCDF-Dateien durchgeführt. Die Daten sind in Spalten formatiert und diese durch ein oder mehrere Leerzeichen voneinander getrennt. Die Zahl der Spalten ist identisch mit der Zahl der Werte derjenigen Koordinatenvariable, die in der Definition der Datenvariable an letzter Stelle steht. Die Zahl der Spalten `dim` wird vom Unterprogramm `lastdim` geliefert und an `outputf` übergeben. Als Input benötigt `lastdim` den Header der jeweiligen CDL-Datei, der mittels `ncdump -h` erzeugt wird.
2. Mit Hilfe von `ncdump` als komplette CDL-Datei (Hintergrundinformation in Abschnitt 6.1)

Die auf beiden Wegen erzeugten Dateien werden zusammen mit zwei Hilfedateien in ein zip-Archiv gepackt und komprimiert.

2.2 Einbindung als Grid-Job

Die oben beschriebene Formatkonvertierung wurde im C3Grid installiert, weil im C3Grid Archive eingebunden sind, die von der Klimafolgenforschung genutzt werden.

Im C3Grid bieten sich zwei Möglichkeiten der Einbindung als Grid-Job an:

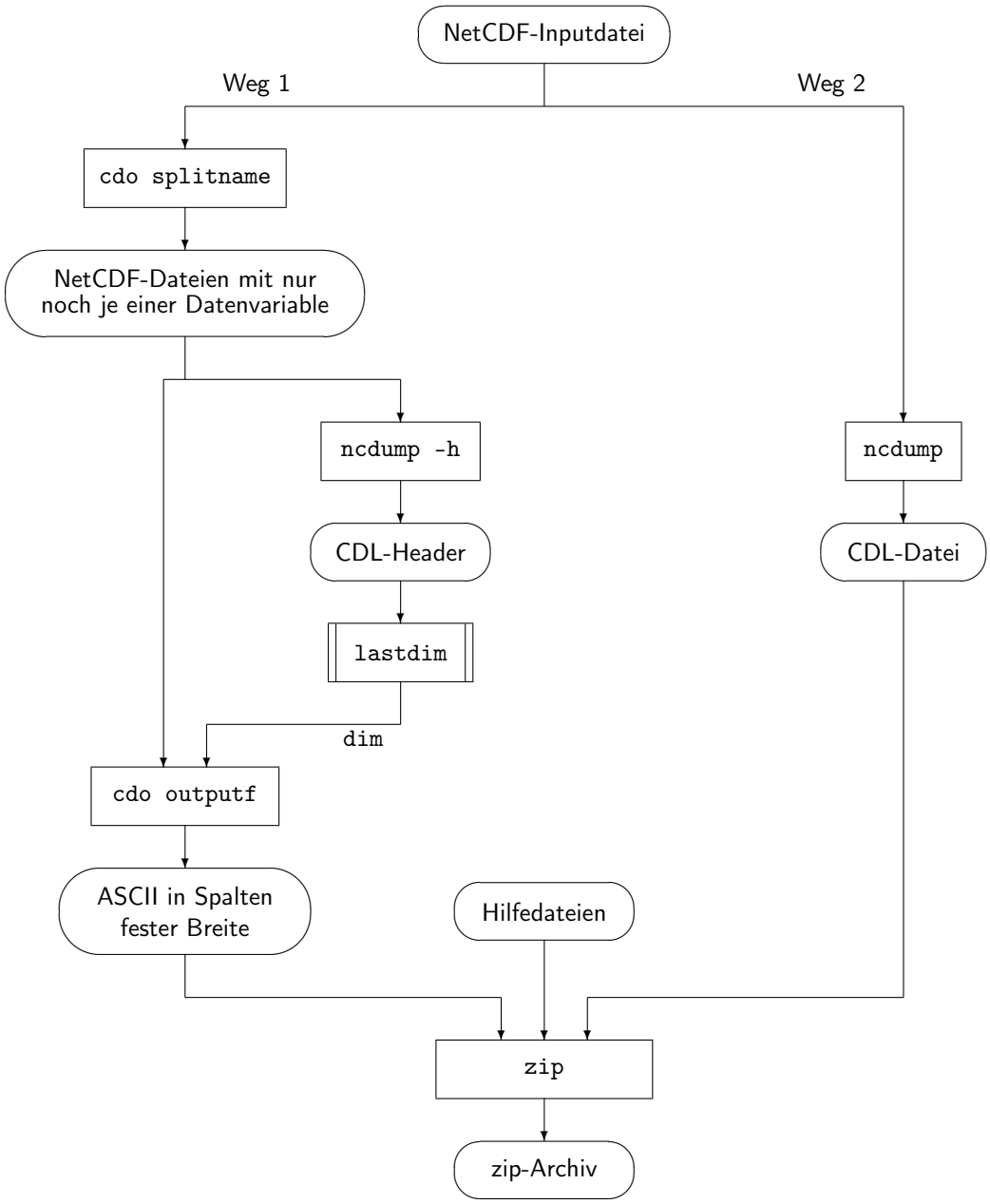


Abbildung 1: Flussdiagramm für `ascii.py`

1. Installation auf den Compute-Nodes: In diesem Fall würde das Grid-Scheduling einen Compute-Node aussuchen, der unter nur geringer Last steht. Dort würde die Formatkonvertierung dann ausgeführt werden. Zuvor müssten die Daten von Archiv zum Compute-Node transportiert werden.

2. Die Konvertierung findet beim Datenprovider statt, egal wie groß die Compute-Last dort gerade ist. Da sich dort auch die Daten befinden, entfällt der Transport vor der Konvertierung.

Der mit der Konvertierung NetCDF → ASCII verbundene Rechenaufwand ist relativ gering. Da zudem die im Archiv befindlichen NetCDF-Darteien meist sehr groß sind, wurde der zweiten Variante der Vorzug gegeben.

Mit GNDMS (Generation N Data Management System) (7) (8) besitzt C3Grid einen Datenmanagementdienst, in dessen Datenprovider-Schnittstelle sich Formatkonvertierungen ohne großen Aufwand einhängen lassen. In dieser Weise wurde verfahren, um die Formatkonvertierung NetCDF → ASCII als Grid-Job zur Verfügung zu stellen. Das Konvertierungsskript `ascii.py` wird dabei von einem Wrapper-Skript aufgerufen, welches die Schnittstelle zwischen GNDMS und lokaler Software beim Datenprovider ausfüllt und Größenschätzung, Holen, Ausschneiden und Formatkonvertierung der angeforderten Daten sowie die Erzeugung zugehöriger technischer Metadaten steuert[2].

Nutzer benötigen so nur einen Grid-Job für Bereitstellung und Formatkonvertierung. Das kommt den Bedürfnissen der Klimafolgenforschung entgegen, deren Nutzer die gewünschten Daten in der Regel nicht selbst haben, sondern erst beschaffen müssen.

Über eine Größenschätzung wird sichergestellt, dass die angeforderten Daten nicht zu voluminös für den abschließenden Download sind. Größenschätzung und Metadaten-Erzeugung wurden angepasst. Da zip-komprimiertes ASCII etwas weniger Platz beansprucht als NetCDF, das Skript `ascii.py` Daten aber zweimal ausgibt (spaltenformatiert und als CDL), wurde in die vorhandene Größenschätzung ein Faktor 2 eingeführt:

$$\text{zip-Dateigröße} = 2 \cdot \text{NetCDF-Dateigröße} + 1.1 \text{ MB}$$

1.1 MB ist die Größe einer PDF-Datei, die die Anleitung zum Import in die Tabellenkalkulationsprogramme Excel und OpenOffice enthält und zusammen mit den Daten ins zip-Archiv gepackt wird.

[2] Hintergrundinformation in Abschnitt 6.2. Im Ablaufdiagramm (Abb. 4) ist die eingehängte Formatkonvertierung grün markiert.

3 Technische Umsetzung

Typ: Unix-artiger Befehl

Programmiersprache: Python

Output-Files im zip-Archiv:

`all.cdl`	CDL-Datei
`readme.txt`	Informationen zum Inhalt der gelieferten Dateien
andere `.txt`-Dateien	Daten der gleichnamigen Variablen in spaltenformatiertem ASCII
`readme_office.pdf`	Anleitung zum Einladen von in Zahlenkolonnen formatierten Daten in Excel und OpenOffice

Temporäres Verzeichnis:

`ascii.py` legt auf `/tmp` ein temporäres Verzeichnis unter einem Namen an, der dort noch nicht in Gebrauch ist. Vor Erreichen des Programmendes wird das Temporärverzeichnis wieder gelöscht.

Log-Dateien:

Im Falle einer fehlerhaften NetCDF-Eingabedatei schreibt `ascii.py` eine Fehlermeldung in die Datei `error.txt`. Diese wird nur im Fehlerfall erzeugt und befindet sich mit im zip-Archiv. Bei der Abarbeitung als Grid-Job schreiben viele Grid-Komponenten Log-Dateien, zu denen der Nutzer jedoch keinen Zugang hat.

Tests:

Zum Test der Formatkonvertierung und der Einbindung als Grid-Job wurden sechs über das C3Grid-Portal (9) verfügbare Dateien geholt, konvertiert und in Excel und OpenOffice importiert.

4 Installation und Benutzung von ascii.py

4.1 Systemanforderungen

Sie brauchen ein unixartiges Betriebssystem. Getestet wurde `ascii.py` mit Linux-Kernel 2.6.16.27-0.9-xen

4.2 Installation

Welche andere Software muss zuerst installiert werden?

Bevor Sie `ascii.py` nutzen können, müssen die folgenden Pakete installiert sein:

- ncdump of the NetCDF library (getestet mit NetCDF-Library 3.6.2)
- CDO (Climate Data Operators) (getestet mit Version 1.4.5.1)
- zip (getestet mit Info-ZIP 2.31)
- Python (getestet mit Version 2.4.2)

Wie installiere ich ascii.py?

1. Entpacken Sie das zip-komprimierte Paket:
   - `unzip netcdf2asc.0.3.zip`
2. Kopieren Sie `ascii.py` und `readme_office.pdf` in Ihr Arbeitsverzeichnis (oder ändern Sie Zeile 315 in `ascii.py`, wenn Sie `readme_office.pdf` woanders ablegen wollen)

4.3 Getting Started

Wie kann ich ein Testbeispiel laufen lassen?

1. Kopieren Sie die NetCDF-Testdatei `test.nc` in Ihr Arbeitsverzeichnis. Achtung: `ascii.py` löscht die NetCDF-Inputdatei automatisch nach erfolgter Konvertierung!
2. In einer Shell tippe
   - `python ascii.py test.nc`
3. Das erzeugte zip-Archiv `test.ascii.zip` sollte mit dem im Paket mitgelieferten übereinstimmen.

Wie kann ich ascii.py starten?

Verwenden Sie anstelle von `test.nc` Ihre zu konvertierende NetCDF-Datei und verfahren Sie wie beim Test.

5 FAQ

Was ist ascii.py?
`ascii.py` ist ein Formatkonvertierer, der eine NetCDF-Datei in ASCII umwandeln kann.

Für wen ist ascii.py gedacht?
Für jeden, der NetCDF 3 oder 4 Classic in ein ASCII-Format konvertieren will, das von Microsoft Excel gelesen werden kann.

Kann ascii.py nur im Grid verwendet werden?
`ascii.py` kann auch stand-alone verwendet werden.

Wie kann ich ascii.py bekommen?
`ascii.py` ist im Paket netcdf2asc enthalten. Dieses können Sie per Download bekommen.

Ich will ascii.py im Grid einsetzen. Wo kann ich GNDMS bekommen?
Bitte wenden Sie sich an das Zuse-Institut Berlin (ZIB). Die GNDMS Datenprovider-Software kann über (7) heruntergeladen werden. Dies ist jedoch nur ein Teil von GNDMS.

Wie kann ich als Nutzer Zugang zu den Daten des C3Grid bekommen?
Sie brauchen ein Nutzerkonto für das C3Grid-Portal (9) und die CERA-Datenbank (10). Nutzerkonten für das C3Grid-Portal vergibt der Portalverwalter. Bis Herbst 2011 sind neue Nutzerkonten für das C3Grid-Portal jedoch nur projektintern erhältlich, da sich C3Grid bis dahin in einer Phase des Umbaus befindet.

Kann ascii.py mehrere NetCDF-Dateien auf einen Befehl hin konvertieren?
Nein

Warum wird die NetCDF-Inputdatei am Ende der Konvertierung gelöscht?
`ascii.py` wurde für das Grid entwickelt. Dort sind stehen gebliebene Dateien unerwünscht.

Lässt sich die automatische Löschung der NetCDF-Inputdatei abstellen?
Das ist möglich. Der Löschbefehl steht in Zeile 253 von `ascii.py`. Löschen sie diese Zeile oder kommentieren Sie sie aus. Die Formatkonvertierung wird dadurch nicht beeinflusst.

Woher bekomme ich die zu den Daten gehörigen Gitterkoordinaten?
Schauen Sie in die Datei `all.cdl`, die im Zip-Archiv mitgeliefert wird. Im Abschnitt `variables` befinden sich die Variablendefinition, in Abschnitt `data` die zugehörigen Zahlenwerte. Die Koordinatenvariablen sind diejenigen, bei denen die Indexmenge genau so heißt wie die Variable selbst, z.B. `lon(lon)`. Die zu einer Datenvariable gehörigen Gitterkoordinaten setzen sich aus den Koordinaten zusammen, die in der Definition der Datenvariablen in Klammern aufgeführt sind, z.B. `var(time, lat, lon)`.

Wie sind die Daten angeordnet (allgemein)?

Die Reihenfolge in den `.txt`-Dateien und in `all.cdl` entspricht der lexikalischen Anordnung der Indexmenge. Die Indexmenge einer Datenvariable ist das kartesische Produkt der Indexmengen derjenigen Koordinatenvariablen, die in der Definition der Datenvariable stehen.

Wie sind die Daten in den .txt-Dateien konkret angeordnet?

Die `.txt`-Dateien enthalten Zahlenkolonnen.

- Die Zahl der Spalten stimmt mit der Zahl der Werte derjenigen Koordinatenvariable überein, die in der Definition der Datenvariable hinten steht. Ist eine Varaible `var` in der NetCDF-Datei z.B. als `var(time, lat, lon)` definiert und gibt es 2 Werte für `time`, 4 für `lat` und 7 für `lon`, so besitzt `var.txt` 7 Spalten.
- Bezogen auf das obige Beispiel stehen in der ersten Spalte von `var.txt` `var`-Werte, die zum ersten `lon`-Wert gehören, in der zweiten Spalte die zum zweiten `lon`-Wert gehörigen. Die Zuordnung folgt der gleichen Reihenfolge, wie sie in der NetCDF-Datei bzw. `all.cdl` für die Koordinatenvariable zu finden ist.
- Bezogen auf das obige Beispiel enthält die erste Zeile in `var.txt` `var`-Werte, die zum ersten Wert von `lat`, die zweite Zeile solche, die zum zweiten Wert `lat`-Wert gehören. Die ersten vier Zeilen lassen sich den vier `lat`-Werten zuordnen, der Reihenfolge in `lat` folgend. Zeile 5 bis 8 sind ebenso zugeordnet, d.h. Zeile 5 enthält wieder `var`-Werte, die zum ersten Wert von `lat` gehören. Die ersten vier Zeilen beziehen sich auf den ersten Wert von `time`, die letzten vier Zeilen auf den zweiten `time`-Wert. Allgemein geschieht die Ausgabe der Werte mittels ineinander geschachtelter Schleifen, wobei die äußerste über die am weitesten vorn in der Variablendefinition stehende Koordinate läuft (im Beispiel `time`).

Wie sind die Daten in all.cdl angeordnet?

Im Abschnitt `data` gibt es für jede Variable einen Zahlenblock. Innerhalb dieses Zahlenblocks wird dieselbe Anordnung verwendet wie in den spaltenformatierten `.txt`-Dateien mit dem Unterschied, dass zusätzliche Zeilenumbrüche vorhanden sein können, weil die Zeilenlänge in CDL-Dateien beschränkt ist.

Wie sind die Daten in den .txt-Dateien formatiert?

Die `.txt`-Dateien enthalten Zahlenkolonnen. Die Zahlen werden im Gleitzahlformat `%13.6g` der Programmiersprache C geliefert. Damit beträgt die Spaltenbreite 13 bei sechs Dezimalstellen. Sehr große und kleine Zahlen werden in Exponentialdarstellung ausgegeben.

Wie sind die Daten in all.cdl formatiert?

In der Datei `all.cdl` gibt es für jede Variable einen Zahlenblock im Format CSV (Comma Separated Values). Anders als in den `.txt`-Dateien gibt es möglicherweise zusätzliche Zeilenumbrüche.

Warum werden die Daten zweimal ausgegeben?

Datenvariablen werden als CDL und als `.txt`-Dateien ausgegeben, weil

- sich nur die spaltenformatierten `.txt`-Dateien leicht in Excel importieren lassen. Dass die Zahl der Spalten mit der Zahl der Gitterpunkte in einer Koordinatenrichtung übereinstimmt, soll die Weiterverarbeitung im Tabellenkalkulationsprogramm zusätzlich erleichtern.
- Nur der CDL-Datei kann der Nutzer die Koordinatenvariablen und deren Werte entnehmen.
- Jede syntaktisch korrekte NetCDF-Datei lässt sich mit `ncdump` in CDL konvertieren (Weg 2 in Abb. 1), weil `ncdump` zum NetCDF-Standard gehört. Dagegen hat CDO `splitname` (Weg 1 in Abb. 1) einige mit `ncgen` (siehe Abschnitt 6.1) erzeugte Testdateien nicht verarbeiten können. Dieses Problem ist bei Tests mit Archivdaten aber nicht aufgetreten, sondern nur mit „künstlich" erzeugtem NetCDF, das nicht die CF-Konvention[3] erfüllt.
- Zusätzlich verfügbares Datenformat CSV (Comma Separated Values)

Kann ascii.py die Formatkonvertierung auch leisten, wenn Werte fehlen?

Ja, denn diese Fähigkeit ist erforderlich, wenn z.B. Ozeandaten verarbeitet werden sollen und Gitterpunkte an Land liegen. Im CDL fehlen die Werte dann ebenfalls. In den `.txt`-Dateien sind fehlende Werte durch `1e+20` ersetzt.

Wird die NetCDF-Datei vor der Formatkonvertierung validiert?

Nein, durch `ascii.py` wird nur der Teil des NetCDF-Standards abgeprüft, dessen Nichterfüllung `ascii.py` zum Absturz bringen würde. Wird ein solcher Fehler entdeckt, wird die Formatkonvertierung beendet und eine Fehlermeldung in die Datei `error.txt` geschrieben. `error.txt` wird in einem solchen Fehlerfall mit im zip-Archiv an den Nutzer ausgeliefert.

Welchen Namen bekommt die Output-Datei?

Den Namen der NetCDF-Inputdatei mit dem Unterschied, dass die Extension in `.ascii.zip` geändert ist. Im C3Grid wird eine lange Zeichenkette als Name verwendet, die im C3Grid eindeutig ist.

Meine Frage ist nicht auf dieser Seite. Wo kann ich Antwort bekommen?

Wenden Sie sich an den Entwickler von `ascii.py`, Torsten Rathmann, rathmann(at)dkrz.de

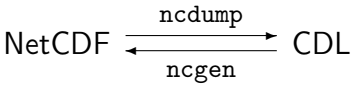
[3] CF steht für „Climate and Forecast" und ist eine Konkretisierung des Standard-Datenformates NetCDF für die Bedürfnisse der Klimaforschung (11).

6 Hintergrundinformationen

6.1 Die Dateiformate NetCDF und CDL

Das Network Common Data Format (NetCDF) (3) ist ein binäres Dateiformat für den Austausch wissenschaftlicher Daten. Auch wenn es nicht fachspezifisch konzipiert wurde, wird es besonders häufig von der Erdsystemforschung genutzt, z.B. auch im C3Grid (1). Das Dateiformat ist selbstbeschreibend. Es gibt einen Header, in dem neben Metadaten (in Form von geordneten Paaren aus Schlüsseln und Attributen) auch die Struktur des Datenbereichs beschrieben ist. Die Daten selbst sind als (ein- oder mehrdimensionale) Arrays abgelegt. Dies ermöglicht einen schnellen Zugriff.

NetCDF ist jedoch nicht nur ein Datenformat, sondern auch die dazugehörige Standard-Software (3) zur Konvertierung aus und in CDL (Common Data for Language) (3).



CDL ist die ASCII-Entsprechung von NetCDF. CDL-Dateien sind wie NetCDF-Dateien in einen globalen Header und einen Datenbereich gegliedert. Im CDL-Datenbereich ist jede Variable namentlich genannt. Die zu einer Variablen gehörigen Daten sind als Block im CSV-Format (Comma Separated Value) abgelegt.

```
netcdf test {
dimensions:
    lon = 7 ;
    lat = 4 ;
    lev = 2 ;
    time = UNLIMITED ; // (1 currently)
variables:
    double lon(lon) ;
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
    double lat(lat) ;
        lat:long_name = "latitude" ;
        lat:units = "degrees_north" ;
    double lev(lev) ;
        lev:long_name = "pressure" ;
        lev:units = "Pa" ;
        lev:axis = "z" ;
    double time(time) ;
        time:units = "days since 2050-01-31 18:00:00" ;
        time:calendar = "proleptic_gregorian" ;
    float temp2(time, lat, lon) ;
```

```
        temp2:long_name = "2m temperature" ;
        temp2:units = "K" ;

// global attributes:
:Conventions = "CF-1.0" ;
data:

 lon = 5, 7, 9, 11, 13, 15, 16.875 ;

 lat = 55, 53, 51, 49 ;

 lev = 100000, 50000 ;

 time = 0 ;

 temp2 =
  280, 279, 277, 277, 276, 277, 277,
  279, 278, 276, 276, 276, 276, 275,
  278, 277, 276, 275, 274, 274, 273,
  276, 275, 273, 273, 272, 273, 273 ;
}
```

Die Konvertierung NetCDF → CDL gelingt mit dem NetCDF-Standardwerkzeug `ncdump`, die Umkehrung mit dem Standardwerkzeug `ncgen`.

Koordinatenvariablen stellen einen Satz von Grundkoordinaten zur Verfügung. In der Erdsystemforschung sind dies häufig geografische Länge und Breite, Höhe (z.B. als Luftdruck-Level oder Meerestiefe) und Zeit. Koordinatenvariablen lassen sich von anderen Variablentypen dadurch unterscheiden, dass sie mit Hilfe einer Indexmenge gleichen Namens definiert sind, z.B. `lon(lon)`. Die Indexmenge `lon` ist zusammen mit den Indexmengen der anderen Koordinatenvariablen im Dimensionsteil des Headers definiert.

Datenvariablen werden mit Hilfe von Koordinatenvariablen definiert. Eine Datenvariable besitzt dort einen Wert, wo ein Gitterpunkt liegt. Die Zahlenwerte sind im Datenbereich des NetCDF- bzw. CDL-Files zu finden.

Die Indexmenge der Datenvariable ist das kartesische Produkt der Indexmengen der Koordinatenvariablen. Welche Koordinatenvariablen dies sind, ist in der Definition jeder Variable individuell festgelegt. Im Beispiel ist `temp2`, die Temperatur in zwei Metern Höhe über der Oberfläche, als dreidimensionales Array in Abhängigkeit der Koordinatenvariablen `time`, `lat` und `lon` definiert. Die Koordinatenvariable `lev` findet hingegen keine Verwendung. Im NetCDF- bzw. CDL-Format lassen sich so sehr leicht den Bedürfnissen der Erdsystemforschung angepasste, mehrdimensionale Felder definieren. Neben Koordinaten- und Datenvariablen gibt es noch Hilfsvariablen.

6.2 GNDMS

Für die Einbindung des Python-Skripts zur Formatkonvertierung NetCDF → ASCII in einen Grid-Job wurde auf bereits existierende Software zurückgegriffen, das Datenmanagementsystem GNDMS (7) (8).

GNDMS ist ein flexibles, verteiltes Grid-Datenmanagementsystem, über das im C3Grid alle Vorbereitungen, Kopiervorgänge, Zwischenspeicherungen und Weitergaben von Daten laufen. Zum Funktionsumfang gehört auch der Umgang mit logischen Pfadnamen, die Durchführung von Dateitransfers mit GridFTP, Workspace-Management und eine kontrollierte, automatische Speicherressourcenverwaltung.

GNDMS basiert auf Globus Toolkit 4.0. Die Kommunikationsstruktur der GNDMS-Dienste ist in Form von WSRF (Web Services Resource Framework) (12) definiert. WSRF ist eine Erweiterung der Web Services Description Language (WSDL) zur Beschreibung von Ressourcen. Die Erweiterung erleichtert die Definition neuer Webdienste. Dabei wird die Grundstruktur von WSDL 1.1 (13) verwendet, d.h. die Tags `definitions`, `message`, `portType`, `binding` und andere können auch in WSRF gefunden werden.

Außer im C3Grid wird GNDMS im Plasma-Technologie-Grid (PT-Grid) (14) genutzt und ist auch für den Einsatz in künftigen Grid-Projekten geeignet (7).

Die Abläufe bei der Anforderung von Daten über das C3Grid sollen nun beschrieben werden. Schnittstelle zwischen Grid und Nutzer ist das C3Grid-Portal (9). Der Nutzer meldet sich dort mit seinem Benutzernamen und Passwort an. Nach Auswahl eines Datensatzes (Abb. 2) legt der Nutzer fest, für welchen Koordinatenbereich, welches Zeitintervall und welche Variablen Daten bereitgestellt werden sollen. An dieser Stelle wird auch das Dateiformat gewählt, siehe Abb. 3.

Durch Mausklick auf „Get Data“ wird ein Grid-Job generiert und an das Scheduling geschickt. Das Scheduling leitet den Job weiter an das GNDMS (Schritt 3 in Abb. 4).

Für die Ausführung von Arbeitsaufträgen ist die GNDMS-Komponente GORFX (Generic Offer Request Factory) zuständig. Dazu gehört das Holen und Transferieren von Daten und auch die Formatkonvertierung. GORFX-Funktionen werden über ein Verhandlungsprotokoll (Offer-Request-Mechanismus), ein Kernbestandteil des GORFX-Dienstes, angeboten: Ein Client (nicht eingezeichnet) beauftragt den GORFX-Dienst mit der Erstellung eines Angebots für die Ausführung einer Datenmanagementaufgabe unter Berücksichtigung zeitlicher Rahmenbedingungen. Der GORFX-Dienst ermittelt das bestmögliche Angebot (Schritt 4) und legt es dem Client vor. Wenn dieser es innerhalb eines Zeitfensters akzeptiert, wird der Auftrag ausgeführt. Der Master wählt den Datenprovider so aus, dass Umfang und Dauer des Datentransports minimiert werden.

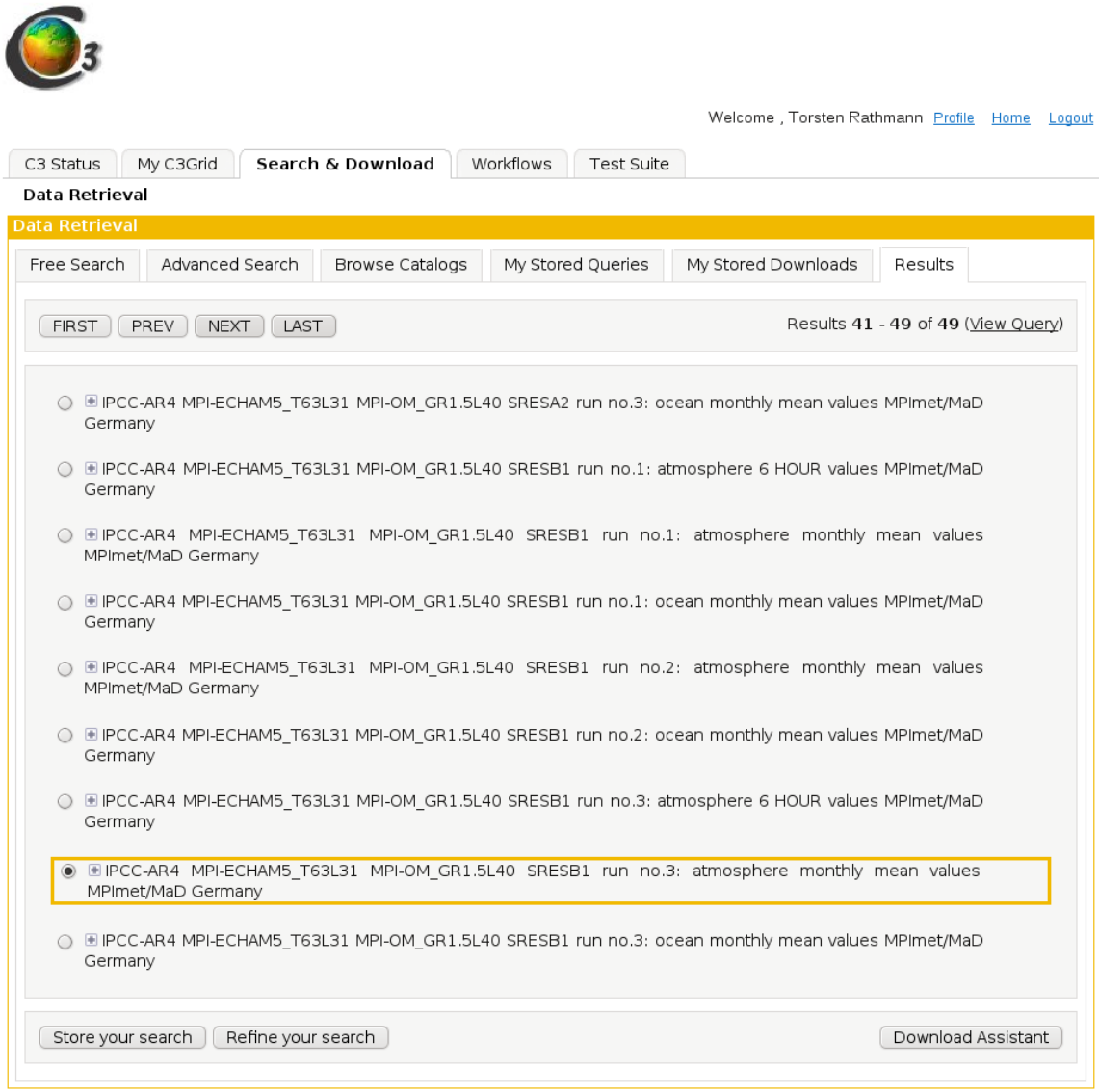


Abbildung 2: C3Grid-Portal, Auswahl eines Datensatzes

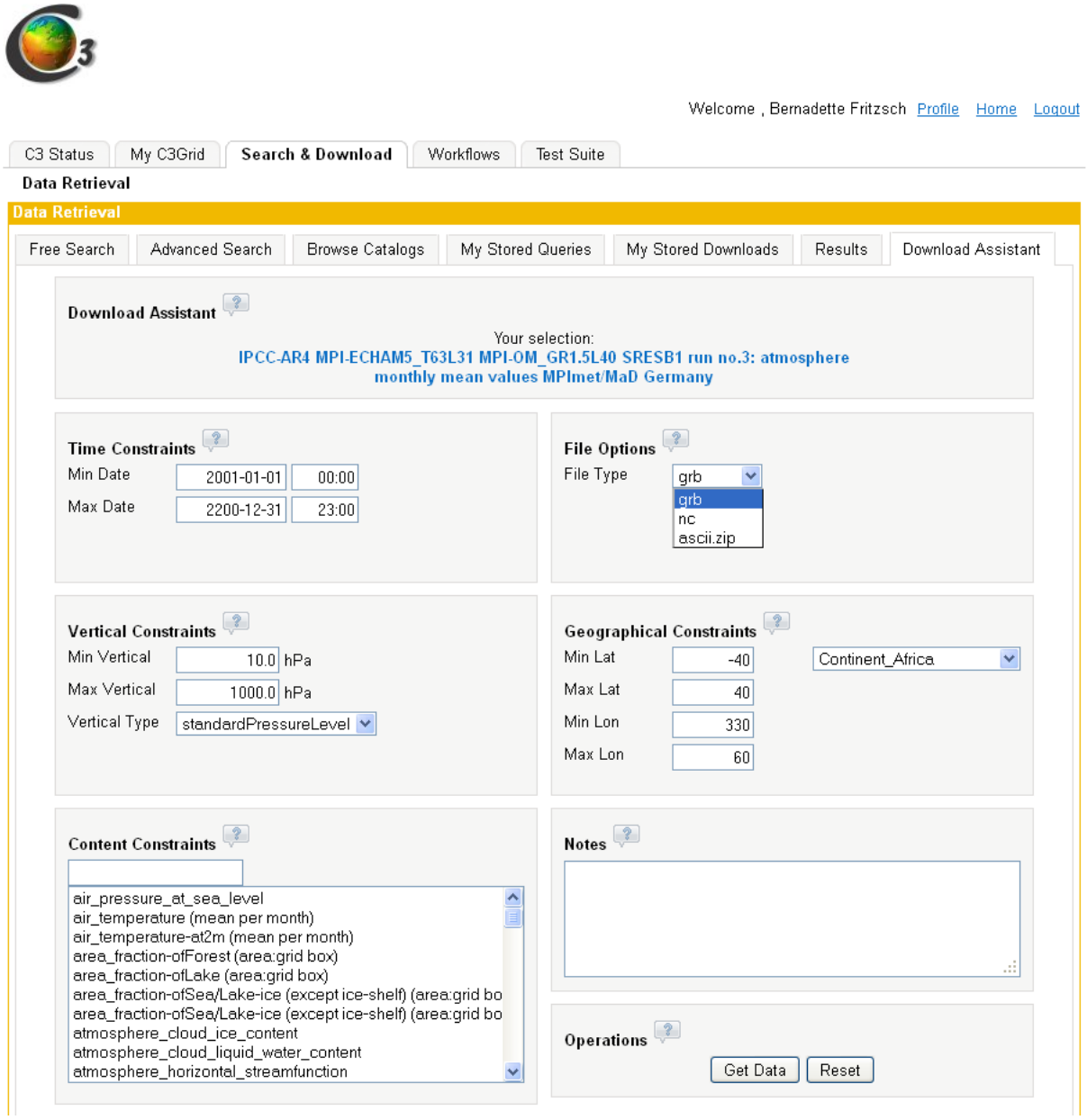


Abbildung 3: C3Grid-Portal, Vorgaben für das Ausschneiden von Daten, Wahl des Dateiformats

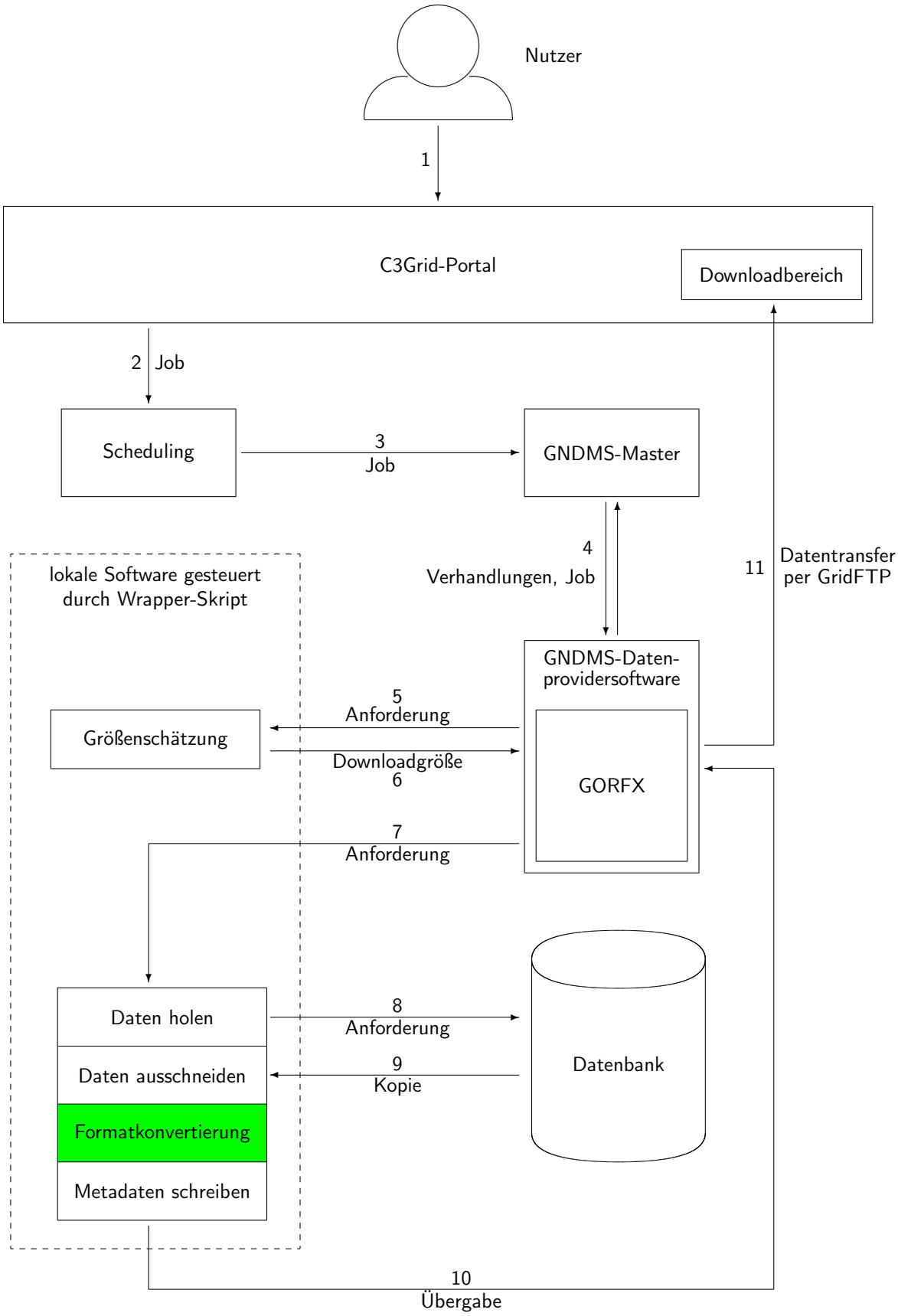


Abbildung 4: Workflow zur Formatkonvertierung, Realisierung im C3Grid

Der Offer-Request-Mechanismus findet sich als `CreateOfferRequest`-Operation in den WSRF-Dateien `GORFX.wsdl` und `GORFX_bindings.wsdl` wieder, die Bestandteil der Datenprovider-Software von GNDMS sind. Auszug aus `GORFX_bindings.wsdl`:

```
<wsdl:operation name="CreateOfferRequest">
  <soap:operation
   soapAction="http://GORFX.gndms.zib.de/GORFX/CreateOfferRequestRequest"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="UnsupportedOfferType">
    <soap:fault name="UnsupportedOfferType" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
```

Für das Binding wird — wie erkennbar ist — SOAP als Übertragungsprotokoll verwendet.

Schritt 4 ist in Abb. 4 als Doppelpfeil in beide Richtungen eingezeichnet, weil Master und Datenprovider wechselseitig kommunizieren. Da im C3Grid jeder Datensatz nur einmal vorhanden ist, gibt es jedoch nur einen Anbieter. Entsprechend sind hier die Verhandlungen reine Formsache.

Die weitere Ausführung wird vom GORFX des Providers kontrolliert, den der Master ausgesucht hat. Im C3Grid ist bei jedem Datenprovider ein GORFX installiert.

Falls Daten geholt werden sollen, wird zunächst geschätzt, welchen Umfang die angeforderten Daten haben. GORFX ruft dafür lokale Software des Providers auf. Am DKRZ ist dies ein Wrapper-Skript, welches seinerseits die Größenschätzung startet (Schritt 5).

Anhand der Größenschätzung soll sichergestellt werden, dass die angeforderten Daten nicht zu voluminös für den späteren Download über das Internet sind. Ursprünglich war geplant, das geschätzte Datenvolumen auch in die oben erwähnten Verhandlungen einfließen zu lassen. Im Zusammenhang mit der Abarbeitung komplexer Postprocessing-Workflows sollte verhandelt werden, an welchen Provider die zugehörigen Compute-Aufgaben gehen. Zusätzlich sollte die voraussichtliche Rechenzeit geschätzt werden, ebenfalls für die automatischen Verhandlungen mit den Ressourcen-Providern. Diese Möglichkeit wird im C3Grid aber noch nicht genutzt. Für das Holen von Daten — allein oder verbunden mit einer Formatkonvertierung — ist das auch nicht nötig. Solche Aufgaben ohne komplexes Postprocessing benötigen nur relativ wenig Rechenzeit.

Wenn das geschätzte Datenvolumen unterhalb eines Schwellenwertes liegt, fordert GORFX die Daten an (Schritt 7). Dabei wird wieder das schon erwähnte Wrapper-Skript aufgerufen, das die Aufgabe an ein dafür geschaffenes Programm übergibt. Dieses Programm holt die Daten der vom Nutzer ausgewählten Klimamodellrechnung oder Messung aus einer Datenbank, schneidet die gewünschten Daten aus und schreibt passende

Metadaten in eine XML-Datei, die zusammen mit den Daten an den Nutzer ausgeliefert wird. An dieser Stelle wird gegebenenfalls auch die Konvertierung in das gewünschte Dateiformat vorgenommen.

Die so zur Auslieferung vorbereiteten Daten werden an die GNDMS-Datenprovider-Software übergeben, die sie per GridFTP an das C3Grid-Portal schickt. Der Nutzer wird per E-Mail darüber in Kenntnis gesetzt, dass die angeforderten Daten zum Download bereitstehen.

Literaturverzeichnis

[1] C3Grid, http://www.c3grid.de/

[2] Climate Service Center (CSC), http://www.climate-service-center.de/

[3] NetCDF, http://www.unidata.ucar.edu/software/netcdf/

[4] GRIB,
http://www.cpc.ncep.noaa.gov/products/wesley/reading_grib.html

[5] CDO, https://code.zmaw.de/projects/cdo/wiki/Cdo

[6] Info-Zip, http://www.info-zip.org/

[7] GNDMS, http://gndms.zib.de
Software-Download: https://github.com/zibhub/GNDMS/downloads

[8] WissGrid-Deliverable D2.1.2,
http://www.wissgrid.de/publikationen/deliverables/wp2/CG-Dienste_in_D-Grid.pdf

[9] C3Grid-Portal, http://www.c3grid.de/portal/

[10] Climate and Environment data Retrieval and Archiving system (CERA),
http://cera-www.dkrz.de/CERA/

[11] NetCDF Climate and Forecast (CF) Metadata Convention,
http://cf-pcmdi.llnl.gov/

[12] WSRF, http://docs.oasis-open.org/wsrf/

[13] WSDL 1.1, http://www.w3.org/TR/wsdl

[14] Plasma-Technologie-Grid, www.pt-grid.de/