

Bibliotheken

Hochleistungs-Ein-/Ausgabe

Michael Kuhn

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

2016-05-27

- 1 Bibliotheken
 - Orientierung
 - Einführung
 - SIONlib
 - NetCDF
 - HDF
 - ADIOS
 - Leistungsbetrachtung
 - Zusammenfassung

- 2 Quellen

E/A-Schichten

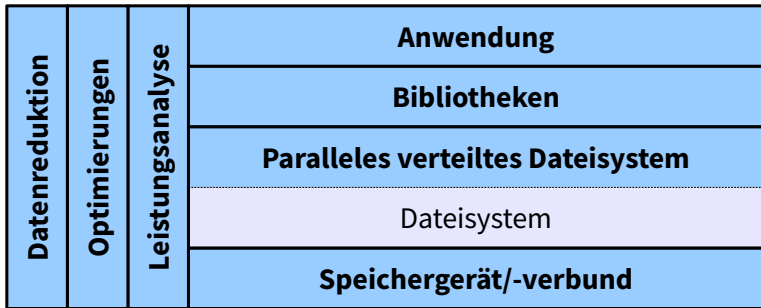


Abbildung: E/A-Schichten

Überblick...

- Bibliotheken stellen zusätzliche Funktionalität bereit
 - Selbstbeschreibende Daten
 - Interne Strukturierung
 - Abstrakte E/A-Definition
- Umgehung von Leistungsproblemen
 - Verursacht beispielsweise durch zu strikte Semantik

Überblick...

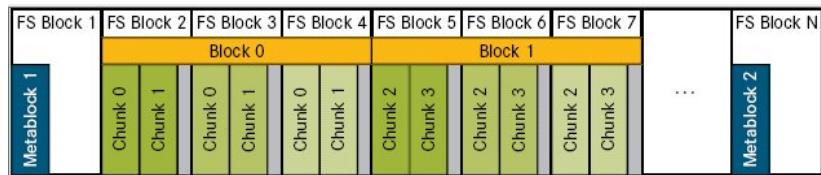


Abbildung: SIONlib-Dateiformat [2]

- Metablock 1: Statische Metadaten (beim Öffnen geschrieben)
- Metablock 2: Dynamische Metadaten (beim Schließen geschrieben)
- Jeder Block enthält Daten eines Prozesses (unterteilt in Chunks)

- `numfiles` gibt die Anzahl der zu verwendenden Dateien an (-1 für automatische Bestimmung)
- `chunksize` gibt die maximale Größe eines Schreibaufrufs an
- `fsblocksiz` gibt die Größe eines Dateisystemblocks/-streifens an (-1 für automatische Bestimmung)

Funktionalität

- Mehrere Zugriffsmodi
 - Kollektives Öffnen mit `sion_paropen_mpi`
 - Nicht-kollektives Öffnen mit `sion_open_rank`
 - Serieller Zugriff via `sion_open` und `sion_close`
- Intelligente Abbildung und Ausrichtung
 - Zusammenhängende Blöcke pro Prozess
 - Abbildung auf internes Dateilayout

Zusammenfassung

- SIONlib existiert hauptsächlich um Unzulänglichkeiten existierender Dateisysteme zu umgehen
- Einerseits Probleme mit vielen Dateien
 - Geringe Metadatenleistung
 - Dafür hohe Datenleistung
- Andererseits gemeinsame Dateien häufig langsam
 - POSIX macht Sperren notwendig
 - Korrektes Zugriffsmuster sehr wichtig

Datenformate

- Es existieren drei Formate
 - 1 Klassisches Format (CDF-1)
 - 2 Klassisches Format mit 64-Bit-Offsets (CDF-2)
 - 3 NetCDF-4-Format
- Datenformate sind offene Standards
- Klassisches und 64-Bit-Format sind internationale Standards des Open Geospatial Consortiums

Datenformate...

- Das klassische und 64-Bit-Format sind eigenständig
 - NetCDF-4 nutzt HDF5
- Mehrere Optionen für parallele E/A
 - NetCDF-4 unterstützt parallele E/A für NetCDF-4-Dateien
 - Parallele E/A für das klassische und 64-Bit-Format mit aktuellen NetCDF-Versionen (4.1.1+) oder Parallel-NetCDF
- Parallel-NetCDF hat eine inkompatible Schnittstelle

Funktionalität

- Schnittstellen für viele Sprachen
 - C, Fortran, C++, Java, R, Perl, Python, Ruby etc.
- Datenformat ist architekturunabhängig
 - Endianness-Konvertierung
- NetCDF unterstützt Gruppen und Variablen
 - Gruppen enthalten Variablen
 - Variablen enthalten Daten
- Zusätzliche Attribute für Variablen

Funktionalität...

- Unterstützung für mehrdimensionale Arrays
 - char, byte, short, int, float, double
 - NetCDF-4: ubyte, ushort, uint, int64, uint64, string
- Größe der Dimensionen ist beliebig
 - Bei klassischem und 64-Bit-Format nur eine unbeschränkt
 - Beispiel: Matrix kann nur in der Zeitdimension wachsen
 - Bei NetCDF-4 beliebig viele unbeschränkt

Funktionalität...

- Zusätzliche Funktionen
 - Transparente Kompression
- Diverse Werkzeuge verfügbar
 - Beispielsweise ncdump um Daten auszugeben
 - NetCDF Operators (NCO)



Beispiel

```
1 netcdf ... {
2 dimensions:
3     time = UNLIMITED ; // (8760 currently)
4 variables:
5     double time(time) ;
6         string time:units = "days" ;
7         string time:long_name = "Julian_date" ;
8
9 // global attributes:
10     string :Conventions = "None" ;
11     string :creation_date = "Wed Jul 16 12:52:44
12         ↪ CEST 2014" ;
13 }
```

Listing 2: ncdump-Ausgabe

Funktionsweise

- 1 Datei anlegen mit `nc_create`
 - Paralleler Zugriff mit `nc_create_par`
- 2 Dimensionen definieren mit `nc_def_dim`
- 3 Gruppen definieren mit `nc_def_grp`
- 4 Variablen definieren mit `nc_def_var`
- 5 Attribute schreiben mit `nc_put_att_*`
- 6 Variablen schreiben mit `nc_put_var_*`
- 7 Datei schließen mit `nc_close`

Funktionsweise...

- Lesen unterscheidet zwei Fälle
 - 1 Dateistruktur ist bekannt
 - 2 Dateistruktur ist unbekannt
- 1 Öffnen der Datei mit `nc_open`
 - Paralleler Zugriff mit `nc_open_par`
- 2 Gruppen-IDs auslesen mit `nc_inq_ncid`
- 3 Variablen-IDs auslesen mit `nc_inq_varid`
- 4 Variablen auslesen mit `nc_get_var`
- 5 Datei schließen mit `nc_close`

Funktionsweise...

- Unterschiedliche Modi
 - Nach dem Anlegen im Define Mode
 - Nach dem Öffnen im Data Mode
- Bei NetCDF-4 automatischer Moduswechsel
 - Ansonsten `nc_redef` bzw. `nc_enddef` notwendig
- Data Mode erlaubt das Speichern von Daten

Funktionsweise...

- Define Mode erlaubt das Ändern der Dateistruktur
 - Hinzufügen von Dimensionen, Attributen und Variablen
- Einige Einstellungen nur direkt nach Definition änderbar
 - Unter anderem Kompression, Byte-Reihenfolge, Fehlerkorrektur und Füllwert

Parallel-NetCDF

- Alternativer Ansatz für parallele E/A
 - Unterstützt das klassische und 64-Bit-Formate
- Entwickelt durch Northwestern University und Argonne National Laboratory
 - Teilweise dieselben Entwickler wie MPI-IO und OrangeFS
- Schnittstelle ist inkompatibel
 - NetCDF-4 kann aber Parallel-NetCDF nutzen

Überblick

- Besteht aus Dateiformaten und Bibliotheken
 - Erlaubt Verwaltung selbstbeschreibender Daten
- Aktuelle Version ist HDF5
 - HDF4 wird immer noch aktiv unterstützt
- Probleme mit Vorversionen
 - Komplizierte API
 - Einschränkungen wie z.B. 32-Bit-Adressierung

Überblick

- Unterstützt Gruppen und Datensätze
 - Datensätze speichern Daten
 - Gruppen strukturieren den Namensraum
 - Analog zu Dateien und Verzeichnissen
- Gruppen können Datensätze und Gruppen enthalten
 - Hierarchischer Namensraum
- Attribute für Datensätze und Gruppen
 - Beispielsweise Minimum und Maximum

Überblick...

- Objekte werden über POSIX-ähnliche Pfade zugegriffen
 - Beispielsweise `/path/to/dataset`
 - Pfad kann Informationen über Daten enthalten
- HDF-Dateien sind selbstbeschreibend
 - Können ohne vorheriges Wissen über Struktur und Inhalt geöffnet und interpretiert werden

Überblick...

- Datensätze können mehrdimensionale Arrays eines Basisdatentypen speichern
 - Integer, Float, Character, Bitfield, Opaque, Enumeration, Reference, Array, Variable-length, Compound
- Datensätze haben Eigenschaften
 - Größe, Genauigkeit, Byte-Reihenfolge etc.
- Beliebig viele unbeschränkte Dimensionen

Sprachspezifische Datenspeicherung

- Sprachspezifische Datenspeicherung
 - Daten werden nach C-Konvention zeilenweise gespeichert
 - Fortran-Daten werden automatisch umgewandelt

1	2	3
4	5	6
7	8	9

Abbildung: 3x3-Matrix

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

(a) C-Speicherlayout

1	4	7	2	5	8	3	6	9
---	---	---	---	---	---	---	---	---

(b) Fortran-Speicherlayout

Chunking

- Chunking erlaubt Daten in allen Dimensionen zu erweitern
 - Bei zusammenhängender Speicherung nicht möglich
- Mögliche Optimierungen
 - Anpassung an Streifenbreite
 - Effizienter spaltenweiser Zugriff
- Zusatzaufwand
 - Üblicherweise geringere Leistung

Chunking...

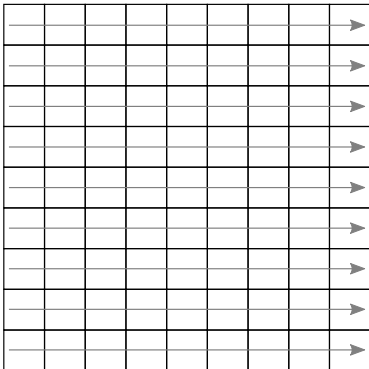


Abbildung: Zusammenhängender Datensatz

- Daten können nur in eine Dimension wachsen

Chunking...

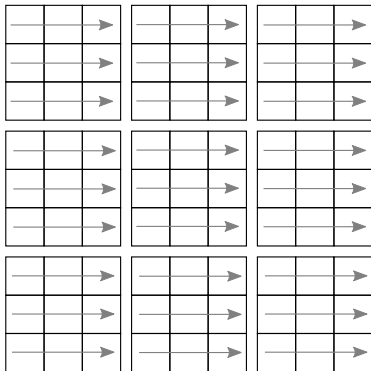


Abbildung: Datensatz mit Chunking

- Daten können in mehrere Dimensionen wachsen

Sonstiges

- Unterstützung für mehrere Backends
 - POSIX und MPI-IO
 - MPI-IO erlaubt parallelen Zugriff auf gemeinsame HDF-Dateien
- Diverse Werkzeuge verfügbar
 - Beispielsweise h5dump um Daten auszugeben
- Zusätzliche Funktionen
 - Transparente Kompression
 - Benutzerdefinierte Filter

Beispiel

```
1 HDF5 "... " {
2 GROUP "/" {
3     ATTRIBUTE "creation_date" {
4         DATATYPE H5T_STRING {
5             STRSIZE H5T_VARIABLE;
6             STRPAD H5T_STR_NULLTERM;
7             CSET H5T_CSET_ASCII;
8             CTYPE H5T_C_S1;
9         }
10        DATASPACE SIMPLE { ( 1 ) / ( 1 ) }
11    }
12 }
13 }
```

Listing 3: h5dump-Ausgabe

Überblick

- ADIOS ist stark abstrahiert
 - Kein byte- oder element-orientierter Zugriff
 - Direkte Unterstützung für Anwendungsdatenstrukturen
- Entworfen für hohe Leistung
 - Insbesondere von wissenschaftlichen Anwendungen
 - Caching, Zusammenfassen von Operationen etc.

Überblick...

- E/A-Konfiguration wird in XML-Datei ausgelagert
 - Beschreibt relevante Datenstrukturen
 - Wird benutzt um automatisch Code zu erzeugen
- Entwickler spezifizieren E/A auf hoher Abstraktionsstufe
 - Kein Kontakt mit Middleware oder Dateisystem
 - Änderungen ohne Neukompilation möglich

Interaktion

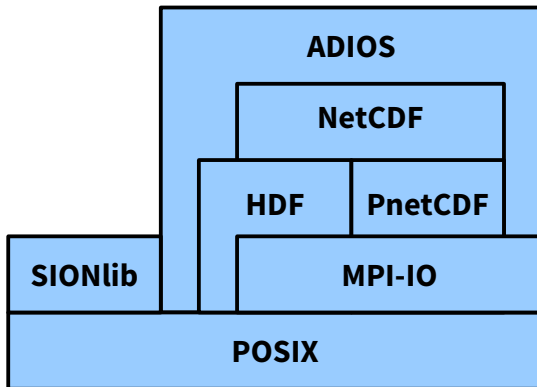


Abbildung: Interaktion zwischen Bibliotheken

Disjoint [1]

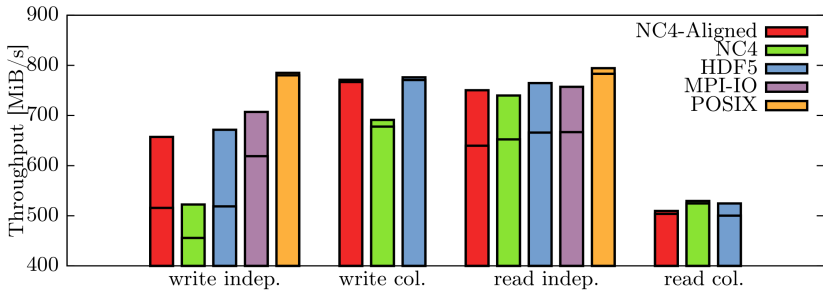


Fig. 5. Disjoint Pattern

- Clients für zusammenhängende Datenblöcke verantwortlich
- Keine Überlappungen
- Jeder Client kommuniziert potentiell mit allen Servern

1-OST [1]

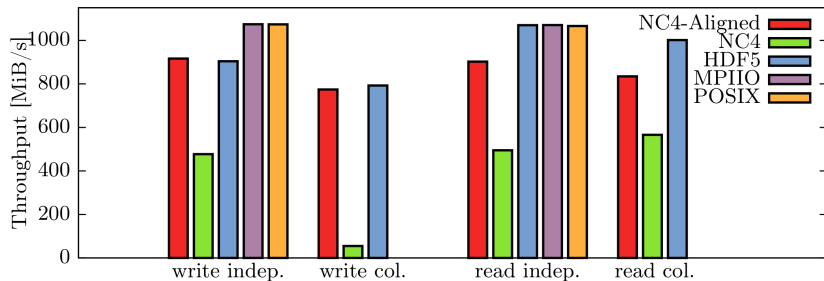


Fig. 6. 1-OST Pattern

- Jeder Client kommuniziert mit genau einem Server

Zusammenfassung

- E/A-Bibliotheken erlauben strukturierten Zugriff
 - Zusätzliche Annotationen und Metadaten erlauben einfachen Austausch
- Zoo von Bibliotheken für unterschiedliche Anwendungszwecke
 - Analyse von Fehlern und Leistungsproblemen schwierig
 - SIONlib umgeht Leistungsprobleme aktueller Dateisysteme
- NetCDF und HDF bieten ähnliche Funktionalität
 - Beide erlauben parallele E/A

- 1 Bibliotheken
 - Orientierung
 - Einführung
 - SIONlib
 - NetCDF
 - HDF
 - ADIOS
 - Leistungsbetrachtung
 - Zusammenfassung

- 2 Quellen

Quellen I

- [1] Christopher Bartz, Konstantinos Chasapis, Michael Kuhn, Petra Nerge, and Thomas Ludwig. A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre. In Julian Martin Kunkel and Thomas Ludwig, editors, *High Performance Computing*, number 9137 in Lecture Notes in Computer Science, pages 274–281, Switzerland, 06 2015. Springer International Publishing.
- [2] SIONlib. File Format. https://apps.fz-juelich.de/jsc/sionlib/docu/fileformat_page.html.