

# NIRVANA

---

## 1. Scenario Overview

---

### 1.1 Background and Purpose

NIRVANA is a grid-based computercode for the solution of the equations of magnetohydrodynamics and for the Poisson equation. It uses adaptive mesh refinement techniques to describe systems with large spatial scale variations.

Numerical methods: Godunov-type, semi-discrete, central-upwind solver for the equations of magnetohydrodynamics; multi-grid Vcycle/SOR to solve the Poisson equation on an adaptive/uniform grid.

### 1.2 More information

<http://nirvana-code.aip.de>

---

## 2. Current Scenario description

---

### 2.1 Environment

#### 2.1.1 Hardware

##### - Processing

single PC  
PC-cluster (MPI or multiple jobs)

##### - Storage

up to 2TB (total) disk space, problem-dependent;  
i.e 2 TB/number\_of\_nodes disk space per node in MPI applications

##### - Network

high bandwidth;  
latency relative unimportant

##### - special hardware

does not apply.

#### 2.1.2 Software

C/C++  
MPI library  
HDF5 library  
for data visualization: IDL, OpenDx  
for compiling: make

the program is deployed as C source code

## 2.2 User Interaction

### 2.2.1 Initiation

compilation with make;  
simulation parameters specified by the user via modification of a template ASCII file;

on PC-cluster: local directories created on each node, parameter file (and restart (binary) files for continued runs) copied in local directories

- Where is the program executed?

single shared memory system (single CPU or more)

single PC, PC-cluster

- How is the program initiated?

command line

### 2.2.2 Monitoring/Steering/Visualization during the run-time of the program

Information about the status of a simulation is written to an ASCII log-file.

- What methods/tools exists for accessing data produced by the program during run-time?

SSH for shell access;  
at present, monitoring is done by hand using vi on the produced ASCII log-file

- Does your application support any standard for monitoring/steering?

No.

- Describe any security measures related to program access for monitoring/steering/visualization.

None.

- Who can access the running program OR run-time produced monitoring data?

Owner and system administrators.

- How is the program termination detected?

Regular stop of program (exit).

- How much monitoring data and how often is monitoring data transferred during a program run (min/max/avg)?

monitoring via shell

- Does your program generate metadata and stores this externally (e.g. in

a catalog)?

No.

- Who accesses this metadata? From where? Does your program access metadata generated by other programs?

Does not apply.

- How many executions/jobs must be monitored/steered in parallel? By how many users?

1 job with multiple parallel threads monitored by 1 user

## 2.3 Input

### 2.3.1 Parameters

Describe the program parameters in detail.

approx. 60 program parameters collected in an ASCII file

### 2.3.2 Input data

ASCII parameter file + binary files in case of restarts

- Where is the input data stored? Describe all central and distributed locations.

local filesystem

- Are file-names known in advance (before the program is started)?

yes

- Are data locations (directory, server, ...) known in advance?

yes

- Describe the different ways data is accessed.

read by the program

- Non-file based data access (XML, database, ...)

does not apply.

- How much data is accessed at each run?

up to 50GB, problem-dependent

- Is it possible that a data set/file is accessed multiple times over a short period of time? For example by different "threads" of the program.

No.

- Elaborate on the use of metadata related to input data.

does not apply.

### 2.3.3 Additional Notes

none

### 2.4 Output

- Where is the output data stored?

local filesystem

- How is the output data structured?

single file or distributed file (MPI).

binary and ACSII format;

HDF5;

program-specific (encoded, platform-independent) format  
for visalization purposes

- Describe what happens when the program finishes?

produced data is copied to other location via script.

- Describe the different ways data is created/changed.

sequentially written by each thread of the program.

- Non-file based data access

does not apply

- How much data is written by the program at each run?

up to 2 TB - problem-dependent.

- Elaborate on the use of metadata related to output data.

does not apply.

### 2.4.2 Additional notes

none

### 2.5 Information resources

does not apply.

### 2.6 Data Stream Management

- Are data exchanged between distributed parts of the application?

yes via MPI function calls

- Are operations compute intensive?

yes

## 2.7 Resource Security and Access Restriction

no particular requirements

## 2.8 Additional Information

- How are workflow/pipeline steps interrelated to eachother?

Workflow/pipelines are not used.

- Is the application executed in several phases where each phase may have different resource requirements or may be executed at a different resource?

no

- How long (avg) does the scenario execute (minutes, hours, days)?

days - weeks, problem-dependent

- How often will the scenario be executed?

frequently.

- Are the executions time-critical?

No.

## 3. Future Scenario and AstroGrid-D Usage

### 3.1 General goals

use more compute resources

### 3.2 Environment

- Are there any constraints due to your participation in other projects or international collaborations?

No

- hardware/network

no

### 3.3 User Interaction

- Which parts should be automated?

resource selection

data transfer before initiation and after termination  
monitoring of program status via API or web portal

- Which user interface are you planning to use?

portal for logfile access

- Are you planning to use any standard for application monitoring/steering?

yes

What ports are used, does the program contact a server or does the program host the server, how is it contacted, ...?

- Aspects of a Portal / WWW based interface:

. Which portal features are mandatory/optional (e.g. credential management, job management, job monitoring/steering, data transfer, ...)?

access/visualization of ASCII log-file data

. How are user managed? Where is information about users defined / stored?

Does not apply.

. Which authentication/authorisation methods are needed ?

Does not apply.

. Do you want to access specific data services (web services, databases, etc.) via a portal?

No.

. Are there any existing programs, on which the user interface should be based OR which should be replaced by the portal?

No.

. Should there be a central AstroGrid portal OR do you want to set up a portal server for each scenario/application ?

central portal preferred

. Does the scenario require any special interfaces OR is it sufficient to use

generic interfaces ?

A generic interface should be sufficient.

- Aspects of a generic Grid Application Programming API (GAT)

. Which GAT functionality would you like to make use of (eg. job submission, file handling, resource brokering, etc.) ?

None.

. What programming languages must be supported ? Which platforms ?

C/C++

. Which Grid Middleware should be supported (Globus, Unicore, gLite, etc.) ?

Does not apply.

. For specific GAT functionality, which protocols/packages/tools should be supported ?

eg. for job management: clusters with PBS, SGE, Condor

Does not apply.

### 3.4 Input

automated management of data transfer

### 3.5 Output

- Do you handle output data manually or do you need an automated management of data?

automated management would be fine.

### 3.6 Additional Information

- How long (avg) does the scenario execute (minutes, hours, days)? Do you aim at a specific speedup?

days - weeks

- How often will the scenario be executed?

frequently

- Which restrictions of the current approach (as described in section 2) do you want to overcome?

Manual monitoring  
Manual data transfer

## 4. Bigger Picture for the far future

Any plans and development should be mentioned here.

MPI-simulations

### 4.1 Organization of Multiple Runs

Maintain a list of all simulations, to repeat simulations with a different binary, with different input data, to check if a program was already executed with a certain set of parameters/input data, ... .

### 4.2 Handling relationships between data products

For example, store metadata on how a data product was generated (from which input data, by which program, with which parameters) and how it can be used by others.

no

#### 4.3 Constructing More Complex Runs

not yet, maybe at a later time

-----  
Use Case written for the 1st Call (15.11.05)  
-----

AIP: Use Cases

=====

uziegler@aip.de, Udo Ziegler, 0331-7499392

=====

Programmname: NIRVANA

Anwendungsbereich: zeitabhaengige Gravito-Magnetohydrodynamik

Code-Art: Finite-Volumen-Methode (Godunov-central scheme),

Mehrgittermethode fuer Poissongleichung,

adaptive Gitterverfeinerung, Kartesische Koordinaten

Code: C (Ansi-Standard)

Libs: --

Parallelisierbar: Shared memory, MPI

Memory Requirements: bis 256 GB

OS: keine speeziellen Anforderungen

Local diskspace: bis 2 TB

Einsatz/Workflow:

Input file: editieren von ACSII-file

Run:

Output: 1. Kenngroessen als ASCII-file (log-file)

2. Simulationsdaten als binary (Plattformabhaengig, verteilt auf Knoten bei MPI-Rechnungen)

3. encodierte, komprimierte Daten zu

Visualisierungszwecken (Plattformunabhaengig, verteilt auf Knoten bei MPI-Rechnungen)

Verarbeitung: Eigenes tool zur Decodierung und Aufbereitung von

Visualisierungsdaten;

IDL und DX zur Visualisierung;

user-abhaengige Bearbeitung

Problem-spezifischer Daten

Anforderungen: 1. evtl. GUI fuer Eingabe von Simulationsparametern

2. Interaktiver Zugriff auf Log-Daten mit graphischer Darstellung

3. automatische Zwischenspeicherung von

Simulationsdaten auf Datenspeicher

mit remote-Zugriff bzw. Endspeicherung

der Simulationsdaten auf System mit

Visualisierungsmoeglichkeit grosser Datenmengen

auch mit Block-Struktur.

Einsatzbereich: MPI, Taskfarming  
Einsatz Grid: bessere Resource-Ausnutzung, Grand-Challenge-Probleme  
Notwendige Grid-Anpassungen:

Interface: siehe Anforderungen

-----  
Bemerkung Angelika: Die Eingabedaten sind minimal, die erzeugten Daten werden separat auf den einzelnen Knoten (derzeit 32) bestimmt und die Größe des lokalen Diskspaces bezieht sich auf alle Knoten. Während der Programmabarbeitung werden zwischen den Knoten nur Befehle ausgetauscht, keine Daten.  
-----

Antworten von Udo Ziegler auf Email von Thomas Röblitz

> - Ausgabedateien:  
> . Wo werden sie momentan gespeichert? In einer Partition auf dem  
> Master-Knoten der Anwendung (bei Benutzung von MPI o.Ä.)? In  
> einem NFS-Verzeichnis?

Bei MPI-runs sind die Daten lokal auf den Knoten verteilt.

> Bei NIRVANA sind 2 TB erforderlich. Bei AMIGA sind  $\geq 50$  GB  
> angegeben (wieviel größer als 50 GB?). Gilt das für eine Datei oder  
> für alle anfallenden Dateien?

Die 2TB sind als Obergrenze fuer alle Dateien anzusehen.

> . Wenn Dateien auf einer lokalen Partition (Master-Knoten) erzeugt  
> werden, was passiert mit ihnen nach Programmende? Wohin werden  
> sie kopiert?

Die lokalen Dateien eines runs werden per script auf andere festplatte kopiert.

> . Sind die Namen der Ausgabedateien vor der Ausführung bekannt?

Ja.

> - Programmausführung:  
> . Alle Programme haben keine speziellen Anforderungen an das  
> Betriebssystem. In welchen Umgebungen werden sie eingesetzt?  
> Windows/Linux/Solaris/..., Workstation/Cluster/...  
z.Z. Linux

> Ist es vorstellbar, dass die Programme erst zu Beginn der Ausführung  
> kompiliert werden?

Ja.

> Gibt es vorkompilierte Binaries für unterschiedliche Umgebungen?

Nein.

> . Ist bekannt wie lange eine Ausführung dauert? Sind die Anforderungen  
> an Speicher und Disk vor dem Programmstart bekannt (in Abhängigkeit  
> der Parameter)?

Die benoetigte rechenzeit fuer ein problem ist nicht notwendigerweise vorher bekannt. in der regel wird ein problem so angesetzt, dass der zu

verfuegung stehende speicher/plattenplatz ausreicht.

- > . Wie wird der interaktive Zugriff auf das/ein Logfile durchgeführt?
- > Wird der gesamte Inhalt/ein Teilinhalt einmal/mehrmals gelesen?

Der zugriff erfolgt per hand mit einem editor.

- > . Wie erfolgt der 'remote' (von wo aus?) Zugriff für die Visualisierung?
- > Wird auf temporäre Dateien oder über eine Schnittstelle des Programmes
- > auf die Daten zugegriffen? Werden alle Daten oder nur Teile gelesen?

NIRVANA hat ein eigenes plattformunabhaengiges datenformat zur visualisierungszwecken d.h. die visualisierung findet auf dem lokalen PC stattf, nachdem die daten vom cluster kopiert wurden.

- > . Welche Metadaten einer Programmausführung sollen gespeichert werden,
- > um sie für die Erzeugung neuer Parametersätze zu verwenden?

keine

- > - Wie werden die verteilten Ausgabedateien zusammengeführt, aufbewahrt
- > und im weiteren Verlauf genutzt?

die plattformunabhaengig generierten daten zur visualisierung werden mit einem zusatzprogramm zusammengefuehrt.  
bei MPI restarts erfolgt das lesen der daten parallel. eine zusammenfuehrung dieser daten ist nicht notwendig.

- > - Was ist unter der Block-Struktur beim Zugriff auf grosse Datenmengen
- > zu verstehen?

bei rechnungen mit adaptiver gitterverfeinerung sind die daten nicht in form einfacher arrays gespeichert, sondern haben eine komplexe blockstruktur, die auch informationen uber die gitterstruktur enthalten.

- > - Wie funktioniert der interaktive Zugriff auf die Log-Daten? Mit dem
- > erwähnten Tool, vi, ...?

per hand mit vi.

- > - Ist vor dem Programmstart bekannt, welche Daten zwischengespeichert
- > werden sollen?

Ja.

- > - Ist vor dem Programmstart bekannt, welche Ergebnisse auf welchen
- > Visualisierungssystemen gespeichert werden sollen?

Ja.

- > Könnten die Daten
- > auch an einem beliebigen Ort gespeichert und dann erst zum
- > Zeitpunkt der Visualisierung zugegriffen werden?

Ja.