Dynamo

1. Scenario Overview

Solution of the induction equation with turbulent electromotive
force (alpha tensor) for modelling of turbulent dynamos in planets
stars and galaxies. The program serves as an example for
similar applications.

Multiple runs should be performed for different input sets.
Depending on the architecture, the program has to be compiled
on different execution hosts separatly.
For each run the input and output data are stored in a special
directory. The source code with architecture dependent makefiles
resides in a separate directory. A submission of the runs via a script
would be favored. The data transport to the excution hosts should be
automatic in both directions (input and results).
An updated ascii file produced by the program should be readable during
execution. For long time simulations a visualization of snapshots
is desired during runtime.
A functionality to enable automatic restarts on other hosts should be provided.

2. Current Scenario description

2.1 Environment

2.1.1 Hardware

- Processing

  fast single or multiple CPUs, about 500 MB memory

- Storage

  up to 10GB disk space

- Network

Because of Taskfarming only for jobsubmission and results

2.1.2 Software

Fortran 90
IDL for online data visualization (not necessary job execution)
compile with make

2.2 User Interaction

2.2.1 Initiation

(i) The user defines the configuration parameters for the program
(ii) compilation with make

- Where is the program executed?

  single shared memory system (single CPU or more)

- How is the program initiated?

  commandline
  the input data should be in the current directory
  and may be also a restart file

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

   Information about the progress of the program is written to stdout
   and an ASCII file.

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

   SSH for shell access

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

   No. Monitoring is done via stdout and analyzing one ASCII file.

- 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, group and system administrators

- How is the program termination detected?

  Normal stop of program (exit).
  Errors are handled by the program and written to stdout/err.

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

   problem dependent: 1-10MB
                      0.1-10GB for retrieving the results

- 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?

  problem dependent (1-100 jobs)
  1 user (may be more in the future)

2.3 Input

2.3.1 Parameters

Describe the program parameters in detail.

2.3.2 Input data

  a) parameter file for compilation
  b) input file: problem dependent (2KB)
  c) input data in case of restart (0.1-1GB)

- 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?

  directory

- Describe the different ways data is accessed.

  read by the program (binary data)
  read by idl for visualization

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

  does not apply

- How much data is accessed at each run?

  for a new start several KB
  for a restart 10MB up to 1GB

- 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.

  only for restart or visualization but not over a short period

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

  no metadata

2.3.3 Additional Notes

none

2.4 Output

- Where is the output data stored?

local filesystem

- How is the output data structured?

different binary files for different timesteps
one file ascii for time evolution of specified and problem dependent parameters

- Describe the different ways data is created/changed.

the program writes cyclic binary-dumps
and appends cyclic some control parameter to an ASCII-file

- Non-file based data access

does not apply

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

0.1-10GB with variable number of files, depends on time resolution

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

no metadata

2.5 Information resources

No additional resources, only plain input files.

2.6 Data Stream Management

- Can single operations be performed on any compute node or do they need special hardware or software?

Single operations could be performed on almost every node (memory & cpu performance)

- Are data exchanged between distributed parts of the application?

no

- Are operations compute intensive?

yes

2.7 Resource Security and Access Restriction

None.

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)?

hours - days

- 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?

Not yet!

- hardware/network

High Flop rate
network for final data retrieval

3.3 User Interaction

- Which parts should be automated?

resource selection
data transfer before initiation and after termination
distribution of multiple jobs

- Which user interface are you planning to use?

may be script based, portal

- 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, ...?

Neither a client nor a server ...

- Aspects of a Portal / WWW based interface:
. Which portal features are mandatory/optional (e.g. credential management, job management, job monitoring/steering, data transfer, ...)?

stdout/err and one ASCII output file accessible through the portal or via a script

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

No own user management.

. Which authentication/authorisation methods are needed ?

General Grid methods.

. 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 ?

Generic interfaces 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.) ?

Does not apply.

. What programming languages must be supported ? Which platforms ?

(Fortran 90, idl if possible)

. 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

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

Automated management would be helpful.

3.5 Output

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

Automated management would be helpful.

3.6 Additional Information

Dynamic usage of more resources is a plan for the future, i.e. distributed computing.

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

hours to days

- How often will the scenario be executed?

Frequently.

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

More memory for the single job.

4. Bigger Picture for the far future

Any plans and development should be mentioned here.

Running parallel MPI-Versions

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, ...

multiple runs already in the near future
each run has all data (input and output) in a specified directory

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.

Metadata generation and storage (data production parameters like execution host, directory/job name, execution files, input parameters)

4.3 Constructing More Complex Runs

For example, combine existing single programs.

add post-processing of results

automatic checkpoint and restart organisation may be on changing execution hosts

---

Use Case written for the 1st Call (15.11.05)

AIP: Use Cases

==========

delstner@aip.de

==========

Programmname: Dynamo
Anwendungsbereich: Parameterstudien zu 3D-Dynamomodellen
Code-Art: Finite Differenzen für MHD-Gleichungen, festes Grid,
verschiedene Geometrien (kart., zylindr., sphärisch)

Code: Fortran90
Libs: --
Parallellisierbar: Shared memory
Memory Requirements: 0.02 -1.0 GB
OS: keine speeziellen Anforderungen
Local diskspace: 0.1 -10 GB

Einsatz/Workflow:
Input file: editieren oder via Script Parameter-files erzeugen
Run:
Output: Ascii-file: Log von speziellen Kenngrößen während des runs.
Zeit-Folge von Simulationsschritten (=restart-files)
Bei restart muss Input-file entsprechend angepasst werden
(binary, spezial,IO-Routinen liegen in code-form vor)
Verarbeitung: IDL-Routinen zur Visualisierung des log
IDL-Routinen für Simulationsdaten

Anforderungen: Interaktiver Zugriff auf Log-Daten (Visualisierung)
Längerfristig: remote Zugriff /Visualiserung von binary Daten (mit IDL)
Längerfristig: Automatische Parameter-Erzeugung auf Basis vorangegangener und laufender runs (Job-Steering)

Einsatzbreich: Taskfarming
keine Kommunikation zwischen den Runs
Einsatz Grid: bessere Resource-Ausnutzung
Notwendige Grid-Anpassungen: Wrapper für Automatisierung des Restart

Interface: Sollte Start von Scripten zum Jobmonitoring gestatten, (Portal?)