

GADGET

----- 1. Scenario Overview -----

1.1 Background and Purpose

Citing from the website:

"GADGET is a freely available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. GADGET uses an explicit communication model that is implemented with the standardized MPI communication interface.

The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

GADGET computes gravitational forces with a hierarchical tree algorithm (optionally in combination with a particle-mesh scheme for long-range gravitational forces) and represents fluids by means of smoothed particle hydrodynamics (SPH). The code can be

used for studies of isolated systems, or for simulations that include the cosmological expansion of space, both with or without periodic boundary conditions. In all these types of simulations, GADGET follows the evolution of a self-gravitating collisionless

N-body system, and allows gas dynamics to be optionally included. Both the force computation and the time stepping of GADGET are fully adaptive, with a dynamic range

which is, in principle, unlimited.

GADGET can therefore be used to address a wide array of astrophysically interesting problems,

ranging from colliding and merging galaxies, to the formation of large-scale structure in

the Universe. With the inclusion of additional physical processes such as radiative cooling

and heating, GADGET can also be used to study the dynamics of the gaseous intergalactic medium,

or to address star formation and its regulation by feedback processes. "

1.2 More information

<http://www.mpa-garching.mpg.de/gadget>

Articles: see the website for an overview

manual: <http://www.mpa-garching.mpg.de/gadget/users-guide.pdf>

----- 2. Current Scenario description -----

Describe the current scenario including as much details as possible.

no uniform scenario available.

I pick one here for illustration, describing a cosmological simulation

Gadget is using a special format for I/O, which is architectur-dependent (bigendian/littleendian)

a) Preparation of initial conditions:

```

# done by astrophysicist      (out of scope here)
files (init.dat )
b) Compiling Code for target system with appropriate compile-time settings in make-
file
#options for in/excluding cosmological physics, eg. starformation rate,
metallicity, cooling, magnetic fields etc
#options for in/excluding algorithms (FoF, PeanoHilbert)
#options for boundary-conditions, time-stepping
done by done by astrophysicist (according to initial conditions)
#compiler+library-path+output format, log-settings
specific to target-system
c) Runtime-settings:
done in a start-file
# location/name of initial conditions -file(s), several physics runtime-
parameters
# outputdir
# cpu and memory consumption, time stepping,
# timesteps for snapshots, restart-files
# logging options
a medium sized init.dat (2x70^3 particles) takes 21MB,
a bigger (2x167^3 particles) takes 114MB

```

2.1 Environment

2.1.1 Hardware

Describe the hardware resources that are currently used.

- Processing

```

single PC: serial version of GADGET 1/2.x
cluster, High Performance Computing-system (Cray etc)
parallel version of GADGET

```

Parallel Version: always requires 2n CPU

$N = N_{\text{particles}} / N_{\text{processors}}$

$N_{\text{Sph}} = \text{Number of SPH-particles}$

Numbers: $\text{TreeAllocFactor} = 0.7$, $\text{ParticleAllocFactor} > 1.2$

Memory Consumption:

$M1 = \text{PartAllocFactor} * N * (68 + \text{TreeAllocFactor} * 64)$ (=>
110byte/Particle

$M2 = \text{PartAllocFactor} * N_{\text{Sph}} * 84$

(for 1:1 mixture => 152byte/particle)

$M3 = \text{BufferSize} * 1024^2$ byte (Communications-Buffer)

$M4 = (\text{PMGRID}^3 * 16)$ (TreePM ,simple periodic box)

(for 2x70^3 particles, the code allocates eg 20MB per processor)

- Storage

no database required

input and output on local disk or nfs-mounted storage

each snapshot is the size of the initial conditions file

snapshots are retained for tracing the development of the simulated system

after being written, snapshot can be moved to a different storage

snapshots can be written in parallel (i.e. groups of processors write their

state to

files)

- Network

Gadget uses MPI, so low latency is desired
from tests, a rough estimate of 70:30 relation of compute-intensive and
network-intensive

activity can be made, i.e. the impact of lower bandwidth in WAN is less
critical than for other mpi-codes

initial files have to be staged to compute-resources

if connecting clusters, to make huge particle numbers feasible,
dedicated bandwidth is required

(this has to be investigated further with mpichG2-runs)

- Describe special hardware or other hardware resources that are relevant
for the scenario.

older Gadget-codes support GRAPE3 or GRAPE6 hardware, but Gadget2 does not

2.1.2 Software

- Describe used software such as operating system, software libraries,
e.g. HDF5-plugin for GridFTP, ...

. list non-standard software with min. version

MPI < 1.0 (Message Passing Interface)

GSL (Gnu Scientific Library)

FFTW FastFourierTransform (FFTW V2.1.5)

HDF5 (if this output format is chosen)

for grid-run: MPICHG2 (ie. mpich 1.2.7 compiled against Globus >= 2.4)

- What programming language is used and what compiler/linker version
is required?

C

C-compiler

std-libs

GSL

FFTW (Version 2.1.5) because today, only this version supports

-mpi/-threads switches needed by gadget2

- How is the program deployed?

Source code under GPL available,

likely modifications by users

compiling necessary

(on grid, either binaries or source+ suitable makefile have to be staged)

- How is the program compiled?

make (gnu make)

- State the program license and any commercial 3rd party licenses.

2.2 User Interaction

Describe the user interaction necessary for starting the program and additional interaction with a running program.

2.2.1 Initiation

- Describe how the program is started and any steps needed before the actual initiation.
 - initial conditions preparation
 - compiling of program (setting compile-options)
 - editing runtime-options file (parameter.txt)
 - preparing machine-file (list of machines to run on, no skew number of processors)
 - or queue-job-script
 - (including: copying init.data to suitable locations, if not on a clusterwide nfs-mounted disk)
- compilation (cf. Section 2.1.2),
 - setting system/architecture dependend options and library paths
- Where is the program executed?
 - cluster (queue)
 - supercomputers
 - grid
- How is the program initiated?
 - commandline call: mpirun -np <NP> -machinefile <machines> ./gadget2 <parameterfile>
 - batchscript for queue-system (example see below)
 - runscript for grid-jobsubmission (example see below)

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

Definitions:

Monitoring - Information retrieval regarding the state of the program. For example, "Program is running" or more application specific information such as "Current simulation iteration is 42".

Steering - Remote alteration of the programs state. For example, program stop or application specific information like "at iteration 42, set parameter x = 78".

Visualization - Remote access of the application data needed for visualization of, for example, a simulation.

- What type of data is produced by the program during run-time used for monitoring/steering/visualization?
 - log files:
 - cpu.txt: giving several numbers for evaluating code-performance

info.txt: time-steps
timings.txt: work-load and particle load balance
energy-txt: controlling the energy-balance
stdout-log

snapshots (state of simulation)
restart-files

steering: only stopping current run after completion of a time-step
create file "stop" in output-dir, causing a restart file to be
written

- What methods/tools exists for accessing data produced by the program during run-time?
normally: commandline (tail -f <logfile> etc.)
controlling enery-values

snapshots can be transferred and view with other tools (IDL, other graphics programs e.g PMViewer)

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

only default logging process with job-queues

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

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

user/owner
sysadmin

- From where can run-time produced monitoring data be accessed?
controlling node of run,

- How is the program termination detected?
-finish of queue-run
-log-file

- How much monitoring data and how often is monitoring data transferred during a program run (min/max/avg)?
at (internal) time-steps, configurable by parameter file

- 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?

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

2.3 Input

2.3.1 Parameters

Describe the program parameters in detail.

2.3.2 Input data

- How is the input data prepared?

pre-execution data generation,

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

local file system of nodes
or nfs file server

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

given by selection in parameter-file
eg snapshot_XXX(.N) , restart.N
since selectable by parameter-file, it's only possible to use metadata-
catalog, if the catalog
builds on parsing the parameter-file

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

yes, from mpi-machinefile or pbs-node-file and settings in parameter-file

- Describe the different ways data is accessed.

POSIX read (special file-format)

- Non-file based data access (XML, database, ...) should include description of

- How much data is accessed at each run?

. number of files/data sets: min/avg/max, 1-2
. total-size: min/avg/max,
 depending on particle-size: see intro (50MB-20GB)
. retrieved-size: min/avg/max
 number of snapshots x (40MB-20GB)

- Is it possible that a data set/file is accessed multiple times over a short period of time?

not likely

- How many users are using the same data simultaneously?
Are these users geographically distributed?

normally 1 user, or 3-5 in a collaboration(then geogr. distrib. likely)

- Elaborate on the use of metadata related to input data.
no metadata in use
metadata could be: input-file, parameter-file, energy-file
extracted data from snapshots (number of halos etc - see MilleniumSimulation)

2.3.3 Additional Notes

Describe any additional information regarding the input data which has not yet been covered.

2.4 Output

This covers what data products are generated (INTERMEDIATE and FINAL results), where they are generated and how they are handled after the program finished (transferring data or removing it, ...).

2.4.1 Output data

- Where is the output data stored? Describe all centralized or distributed locations.

- local file system (node)
 - cluster nfs file server
 - external storage media

- How is the output data structured?

- sequence of snapshots
 - if parallel option used, snapshots splitted in X files

- dataformats: special for gadget, hdf5

- Describe what happens when the program finishes? How are the results used?

- program leaves datafiles at output nodes, have to be collected
 - normally moved to a storage, away from local nodes for
 - postprocessing by various means (IDL, PMViewer & many more)

- Describe the different ways data is created/changed.

- POSIX write

- Non-file based data access (XML, database, ...) should include description of
 - . name of the database management system,
 - . how the database is accessed (ODBC, JDBC, WWW interface, command line, ...),
 - . typical create patterns (bursty, continuous, ...),
 - . physical location/distribution (local, externally, ...),
 - . possibility to replicate the data through some mechanism,
 - . any security related restrictions when data is written,
 -

- How much data is written by the program at each run?
 - . size: min/avg/max,
 - . number of files/data sets: min/avg/max
- Describe the parameters which influence the amount of data and number of files/data sets generated.
- Elaborate on the use of metadata related to output data.
 - . amount,
 - . how it is accessed,
 - . security restrictions,
 - . metadata format [key/value ?],
 - . life-cycle: creation, usage time, used by multiple program runs, deletion, ...

Note, the decision where results are stored is supported by information about the further use or free data storage.

2.4.2 Additional Notes

Describe any additional information regarding the output data which has not yet been covered.

2.5 Information resources

2.6 Data Stream Management

Definitions:

Data Stream - intermediate results can be processed by the subsequent processing module before the current module has processed the last element of the input.

- Can single operations be performed on any compute node or do they need special hardware or software?
- Are data exchanged between distributed parts of the application? does this happen at the beginning, during run-time or at the end?
- Are operations compute intensive?

MPI is used for communications during program execution

2.7 Resource Security and Access Restriction

Describe all security related information that considers access of resources. User based, Group based, by IP-address/netmask, certificates, nodes/resources within a private network, firewall restrictions, ...

cluster-execution: clusters have private networks,
access via cluster-headnode (ssh) node-access within cluster via rsh/ssh
(depends on cluster-policy)

for grid-runs:

firewall has to be open to GLOBUS_TCP_PORTRANGE

2.8 Additional Information

Give additional information not covered by the sections above.

- How are workflow/pipeline steps interrelated to eachother?

- Is the application executed in several phases where each phase may have different resource requirements or may be executed at a different resource?
application can be stopped and restarted by using snapshots
then, changing number of processors/nodes is possible
- How long (avg) does the scenario execute (minutes, hours, days)?
days or weeks, depending on various parameter-settings and size of the problem and involved no. of processors
- How often will the scenario be executed?
with varied parameter settings very often
- Are the executions time-critical?
sometimes

----- 3. Future Scenario and AstroGrid-D Usage -----

Describe the future scenario and envisioned usage of AstroGrid-D as detailed as possible. It is not assumed that the questions can be answered as detailed as in Section 2. Focus on what is expected by the Grid environment and how this new functionality can be used.

Note, there is no special section to describe workflows/pipelines or details about a phased execution of a program. If your scenario is a workflow/pipeline OR your application is executed in several phases, describe EACH step/part covering the sections 3.1 - 3.5. In addition you must describe how these steps/parts are interrelated to eachother (in Section 3.6).

3.0 General goals

Run with increased number of particles
(-> scaling : 16, 32, 64, 128,256,512,1024... processors)
using more compute resources
evaluate impact of network
do postprocessing (e.g. visualisation) in parallel

3.2 Environment

- Are there any constraints due to your participation in other projects or international collaborations?
. specific Grid middleware, hardware, standards, virtual organizations
no,
but need grid-infrastructure (for connecting clusters)
mpichGLobus

3.3 User Interaction

- Which parts should be automated?

data transfer before initiation and after termination,
log-file view (e.g cpu-performance values visualisation etc)

- Which user interface are you planning to use?
GridPortal for log-access

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

. Do you want to use such standards in collaboration with the DGI
or the other communities OR will you develop your own methods?

if they are suitable, yes

- Aspects of a Portal / WWW based interface:

. Which portal features are mandatory/optional
credential management,
job management,
job monitoring/steering,
data transfer
log-file view (may be visualisation of serveral logvalues)
resource view and selection

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

a central server

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

generic interfaces for logs are ok
visualisation of log values would be great

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

no GAT functionality required

3.4 Input

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

input data generation manually
staging to grid-resources required

3.5 Output

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

collecting and moving from grid-resources to other storage-location
required

3.6 Additional Information

- How long (avg) does the scenario execute (minutes, hours, days)? Do you aim at a specific speedup?
speedup (but to make increase of particle numbers feasible)
normally days to weeks
- How often will the scenario be executed?
often
- Which restrictions of the current approach (as described in section 2) do you want to overcome?

4. Bigger Picture for the far future

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,
catalogue the results of runs according to several criteria

4.2 Handling relationships between data products

4.3 Constructing More Complex Runs

do postprocessing in parallel to execution (eg sequencing a movie from snapshots)
steering: run at interesting locations (in simulation space) with a better resolution