

Clusterfinder

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

1.1 Background and Purpose

Clusters of galaxies are tracers of the large-scale distribution of matter in the Universe. The purpose of clusterfinder is improving the source-identification of X-Ray galaxy clusters by correlating data of X-Ray photon maps (ROSAT All Sky Survey - RASS) and optical galaxy maps (Sloan Digital Sky Survey - SDSS). The combination of the two databases increases the signal-to-noise for the detection of even very weak clusters of galaxies. Using the clusterfinder helps to plan observation programmes with satellite missions and telescopes.

The clusterfinder uses well known astronomical methods to generate likelihood-maps from the information available from the catalogues. The calculated likelihood "l" is represented as a number in (ra,dec,z)-space. Evaluation of (l,z)-values for a given point provides information for differentiating between point sources and clusters.

1.2 More information

Article:

Astronomy & Astrophysics 420, 61 (2004), available at
<http://xxx.lanl.gov/abs/astro-ph/0403116>

Website:

<http://www.g-vo.org/portal/tile/products/services/clusterfinder/index.jsp>

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

In a manual data preparation step, the catalog data is retrieved through a Web interface and stored in the format required by the I/O-routines of the program. Since the results for each patch of sky are independent of results from neighbor patches, retrieval of the data is a separable step of the workflow.

The core of the application is a fortran program, whose first action is to read in a parameter file containing

- * tuning parameters (search area, stepping, limits for background, redshift, richness, energies),
- * filenames for resources (mask maps, pointspread function),
- * ra/dec-ordered values of catalog data, and
- * I/O files.

The map-files, pointspread function file and the catalog data file are read by the program before the calculation starts. The calculated likelihoods are stored in an output file.

For postprocessing there are different methods. One is to generate contour-maps (with redshift z as parameter) and plot this against catalog-data for the ra/dec area. Since the results for each patch of sky are independent of results from neighbor patches, the postprocessing is also a separable step of the workflow.

Based on Globus 2.4.x, the following program execution within the GAVO-grid was

implemented:

1. Get data from a catalog (RASS photons, SDSS) via a VO-query (HTTP-GET)
2. Store data in a file in the format required by program I/O-routines
3. Manually edit the parameter file
- 4.a Stage data, program and other resource-files to execution machine
- 4.b Generate a globus-rsl for execution
5. Call globus-run-job with RSL-script
6. Retrieve results to initiating host

2.1 Environment

2.1.1 Hardware

- Processing

two single-CPU workstations, 100MB RAM

- Storage

local disk: 20MB

archives: access to (either)

* databases (RASS, SDSS) (possible in Garching)

* VO-interface (Conesearch) for Archives, possibly other archives

- Network

* connection to internet (http-GET or SOAP ports not blocked)

alternatively: connection to database

* no special bandwidth-requirements or latency

* no limitation on traffic-volumes

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

None

2.1.2 Software

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

linux / unix

(windows possible as well)

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

fortran (g77 minimum)

compiler (gcc or Intel)

perl, python

Java

- How is the program deployed?

. source code, pre-compiled binaries, ...

both ways possible

precompiled binaries preferred

- How is the program compiled?

make

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

no license

2.2 User Interaction

2.2.1 Initiation

- Describe how the program is started and any steps needed before the actual initiation.

simple execution:

\$>clusterfinder: executable without any commandline switches

expects:

- * parameterfile (name set at compile-time)
- * datafile with (ra,dec,photoncount|r*-magnitudes) columns
(parameterfile indicates which columns)
- has to be prepared by astronomer
- * mask (file with a mask for blotting background effects)
- * pointspread-function (given as number cols in a file)
- * exposure-map(for RASS-photons), other maps (dust etc. for optical catalog)

output: likelihood-file

grid-execution:

perlscript: RunJob.pl

- * reads args from a config file,
- * splits target area in chunks
- * stages (using globus-url-copy)
parameterfiles, resourcefiles,
executables(clusterfinder+GetData*) to
machines (from a machine-file in standard
globus-mpi-format)
- * creates rsl-files for each Job
with appropriate params (path to rundir,
params for GetData* (ra,dec,sr)
and GLOBUS_TCPRANGE, log-files settings
- * calls globus-run-job with rsl-file [globus with
fork-job-manager]

perlscript: GetDataLWP_(RASS|SDSS).pl -r (ra) -d (dec) -s (searchradius)

- * creates parameterfile from a template,
- * fetches data from catalog
- * stores them into a plain ASCII-file with required cols
- * executes clusterfinder

perlscript: Retrieve.pl

- * reads from a list generated from RunJob.pl
- * retrieves a tarball of rundir (results, catalog-data.
logfiles), (using globus-job-run and globus-url-copy).
Has to be called separately

- compilation (Cf. Section 2.1.2)

- Where is the program executed?

single CPU, no MPI,
grid-environment

- How is the program initiated?

command line

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

- What type of data is produced by the program during run-time used for monitoring/steering/visualization?

log-files, stdout/err, intermediate results, ...
logging info: coarse grained location (ra,dec),
results are written to file for each finished step
[row by row (ra,dec,likelihood+ some more cols)]

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

No tools yet (beyond a simple text editor). Basically something like tail -2 (result-file) would be sufficient to monitor progress.

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

no

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

normal unix-security

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

user/owner, group
for grid-execution only a gridrunner-account

- From where can run-time produced monitoring data be accessed?

from initiating host
from grid-host

- How is the program termination detected?

process terminated
from initiating host

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

on demand, 1kB

- Does your program generate metadata and stores this externally (e.g. in a catalog)?

not at the moment

- Who accesses this metadata? From where? Does your program access metadata

generated by other programs?

N/A

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

depends on available machines, no parallel monitoring required

2.3 Input

2.3.1 Parameters

Describe the program parameters in detail.

- * area (ra|dec)-min/max (square degrees of sky) and stepping
- * min/max of catalog-data values (energy-levels, richness, redshift etc)
- * filenames for I/O

2.3.2 Input data

- How is the input data prepared?

- * preparation through queries to a database or catalog
- * pointspread-function calculation
- * map-selection
- * catalog-data are required in chunks (rectangular sky-areas)

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

- * local file system,
- * remote (or local) database server
- * remote archive server

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

- * paramfile name is set at compile time
- * all other params (dirs, inputdata etc are set via paramfile)

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

- * for catalog-data: server could be mirror (on the fly)
- * pointspread-function can be changed before paramfile is read from clusterfinder
- * other files could change as well before the execution

- Describe the different ways data is accessed.

- * POSIX read
- * http-get (by wrapper)
- * SOAP (by wrapper)

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

database access possible (with a wrapper):
for RASS-data: postgresql through commandline or perl-dbi, jdbc
for SDSS-data: mssql through commandline or perl-dbi, jdbc

access pattern: bursty

possibility to replicate the data through some mechanism:
whole archives are too big, replication of chunks possible

- How much data is accessed at each run?

(500k-50MB, depends on parameter settings and data from archive,
no uniform distribution)

. number of files/data sets: min/avg/max,

5/5/5

. total-size: min/avg/max,

10MB/30MB/50MB

. retrieved-size: min/avg/max

10MB/30MB/100MB

- 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. Then, replicating and/or caching might be interesting.

yes

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

not at the moment

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

. amount,
. how it is accessed,
. security restrictions,
. metadata format [key/value ?],
. life-cycle: creation, usage time, used by multiple program runs, deletion, ...

TBD, only conceptually

2.3.3 Additional Notes

N/A

2.4 Output

2.4.1 Output data

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

local file system

- How is the output data structured?

single file, plaintext

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

- * manually moved
- * combined into larger areas
- * used as input for graphics
 - (IDL, gnuplot, etc., user interaction or scripts)
 - * contourplots of likelihood values against catalog maps of astronomical objects in the area
 - * plots of redshift/likelihood at fixed ra/dec -values

TBD: archiving of likelihood for parameter-settings

- 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? (min/avg/max)

size: 1/20/50MB
number of files: 1/1/1

- Describe the parameters which influence the amount of data and number of files/data sets generated.

ra, dec, stepping, available catalog data

- 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, ...

TBD, nothing fixed but column-names

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

2.4.2 Additional Notes

N/A

2.5 Information resources

Give a summary of each information resource that is accessed by the program. Include information about data input/output, locations, access methods (XQuery, SQL, ...), security related restrictions, search of metadata (exact key search i.e. "ABC", range queries i.e. "AB*", ...)

```
RASS-photons: ra,dec (min/max or searchradius) :query params
              output: ra,dec,photon energy, exposure time
SDSS ?
```

2.6 Data Stream Management

There is currently no provision made for data streams. It would be conceivable to let the main process begin with the data available while waiting for additional data from the remote data bases. Feeding the output to a visualization program while the main code is still executing is also possible (but would be considered monitoring).

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

Neither catalog data fetching, the core calculation, nor visualization requires special hardware.

- Are data exchanged between distributed parts of the application? does this happen at the beginning, during run-time or at the end?

All data exchange between parts of the application is one-way.

- Are operations compute intensive?

Data fetching can be time-consuming and unpredictable.
Visualization is not necessarily compute intensive, but can be.

2.7 Resource Security and Access Restriction

Catalog, workstation, and file access are based on username/password.

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?

yes, could be feasible

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

for 1 sq-deg of sky on avg 30 min - 3 h
normally ~4 sq-deg for 1 run
initial paper reported on 150 sq-deg

- How often will the scenario be executed?

- Data Release 4 of SDSS covered 6670 sq-deg
- ultimate common footprint of RASS/SDSS: (estimated) 40000 sq-deg
- 40000 /4 = 10000 runs ~ 10000 h ~ 400 d
- possibly several times for different maps/pointspread-functions

- Are the executions time-critical?

yes, because running on multiple processors is the only way to avoid a run for a whole year (but not on a level of 1 single run)

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

The overriding goal is to speed up the processing. Using the current scenario, an analysis of the whole sky would take on the order of a year. If between improving the code and utilizing the grid the speed could be increased by a factor of 100, a whole sky analysis would take only a few days. This would open up the possibility of

- * searching for other types of galaxy clusters,
- * improving the resolution, and
- * doing detailed parameter studies to test the program and to characterize the systematics of the selected cluster sample.

Another major goal is to make the template functions and source catalogs generic, so that any user can customize the analysis to pursue his own scientific questions. An important part of this would be generalizing to more than two catalogs.

Additional features we hope to improve, partly using grid technology, are

- run-time monitoring
- archiving results and parameters to avoid reruns with same settings
- visualization (may be with interfacing e.g Aladin from VO)

With resource broker, scheduling available, and job-management, the old scenario could be reorganized into a different kind of workflow: retrieving data on a master, calculations on "slave"-nodes, perhaps using only temporarily available resources

3.2 Environment

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

no specific constraints

3.3 User Interaction

- Which parts should be automated?

(compute) resource selection,
data transfer before initiation and after termination,
jobs-status/monitoring

- Which user interface are you planning to use?

Portal / Webservice

- Are you planning to use any standard for application monitoring/steering?
. Do you want to use such standards in collaboration with the DGI
or the other communities OR will you develop your own methods?

if there is something available from DGI, it could be used

- Aspects of a Portal / WWW based interface:

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

credential manag., job-management job monitoring, data transfer

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

should be a VO

- . Which authentication/authorization methods are needed ?

TBD

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

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

see www.g-vo.org Clusterfinder-webapp

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

could be a portlet in AstroGrid portal

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

no special interfaces

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

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

Ultimately the application should utilize all of the generic utilities.

. What programming languages must be supported ? Which platforms ?

fortran, scripting, Java

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

Globus

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

e.g. for job management: clusters with PBS, SGE, Condor

3.4 Input

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

There are currently manual steps, but most of them could be automated.

3.5 Output

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

There are currently manual steps, but most of them could be automated.

3.6 Additional Information

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

One run takes about an hour but is not useful by itself. Full sky (or full catalog) currently would take about a year, which is not feasible. By optimizing the code and applying grid computing, we hope to obtain a speed up by a factor of 100, which would make the method a convenient and flexible tool.

- How often will the scenario be executed?

With 1 sq-deg runs, several tens of thousands of times.

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

Above all the limitation on speed.

4. Bigger Picture for the far future

4.1 Organization of Multiple Runs

The researcher should be able to specify in a high-level way the area of sky he is interested in and the order (priority) in which it is processed.

4.2 Handling relationships between data products

A researcher who comes to this tool should be able to first establish whether all or part of the calculation he needs has been made already. This requires a

catalog (metadata) of results with a useable interface.

4.3 Constructing More Complex Runs

Combining other catalogs, other sorts of astronomical data, and a larger number of data sets simultaneously.