Three tools for high-throughput computing with GAMESS

Riccardo Murri
<riccardo.murri@uzh.ch>

Grid Computing Competence Centre, University of Zurich
http://www.gc3.uzh.ch/

May 5, 2011

University of Zurich

Three tools for high-throughput GAMESS

GRunDB	Compute energy of a set of molecules.
GGamess	Run an arbitrary set of GAMESS `.INP` files in parallel.
gc3libs.template	Generate a set of files from a given template.

The purpose of this talk is to determine whether these can be of any use to you or GC3 should stop supporting them.

What is GRunDB then?

GRunDB is a tool to automate:

- running GAMESS on a data-bank (GMTKN24) of molecule geometries
- comparing computed stoichiometry results with known-good ones

Functional high-level view

GRunDB is implemented as a Linux command-line tool on top of the GC3Libs/GC3Utils toolkit.

GRunDB takes as input:

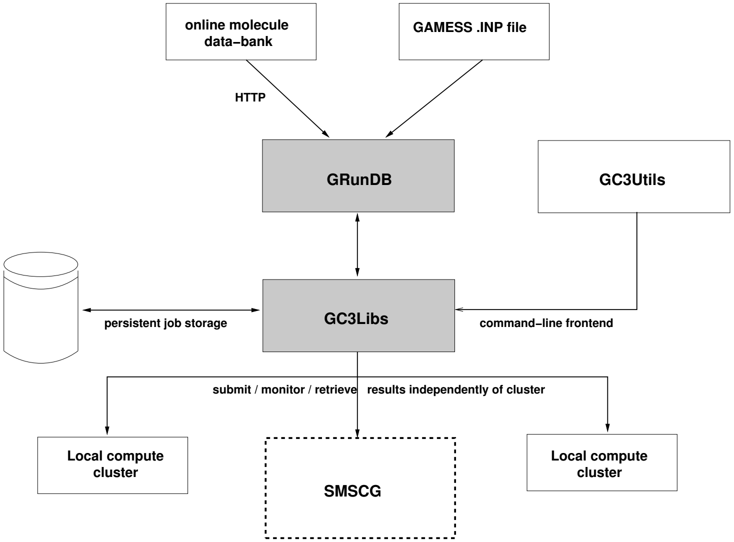
- a set of molecules (a subset of an online data-bank: either the original GMTKN24 or its local copy at UZH)
- a GAMESS input file (except the `$DATA` section)

It creates a GAMESS job for each molecule in the subset, plugging its geometry in the template input file.

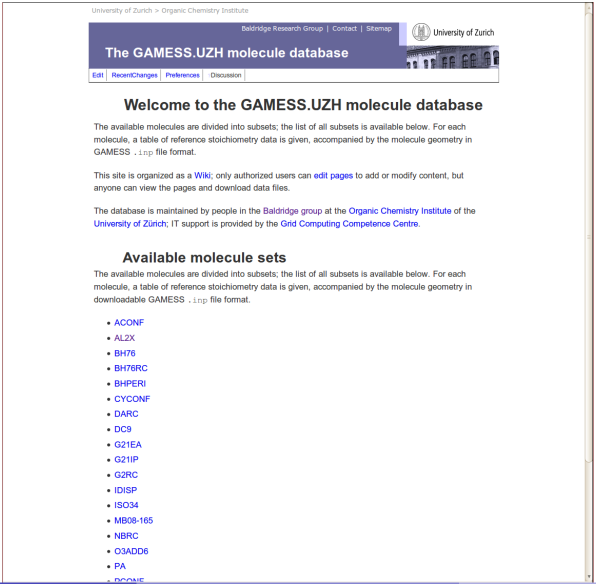
When all jobs are done:

- results are extracted from the output files
- a summary table compares computed energy with reference data

Architecture



The GAMESS.UZH molecule database: Subset index


University of Zurich > Organic Chemistry Institute
Baldridge Research Group | Contact | Sitemap
University of Zurich
The GAMESS.UZH molecule database
Edit | RecentChanges | Preferences | Discussion

Welcome to the GAMESS.UZH molecule database

The available molecules are divided into subsets; the list of all subsets is available below. For each molecule, a table of reference stoichiometry data is given, accompanied by the molecule geometry in GAMESS `.inp` file format.

This site is organized as a Wiki; only authorized users can edit pages to add or modify content, but anyone can view the pages and download data files.

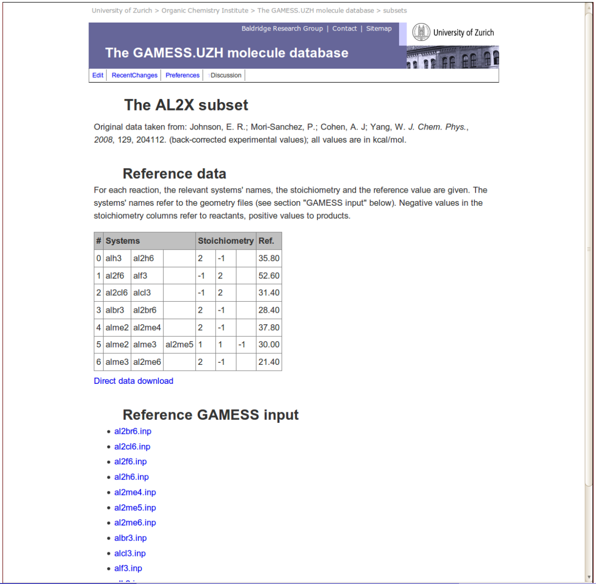
The database is maintained by people in the Baldridge group at the Organic Chemistry Institute of the University of Zürich; IT support is provided by the Grid Computing Competence Centre.

Available molecule sets

The available molecules are divided into subsets; the list of all subsets is available below. For each molecule, a table of reference stoichiometry data is given, accompanied by the molecule geometry in downloadable GAMESS `.inp` file format.

- ACONF
- AL2X
- BH76
- BH76RC
- BHPERI
- CYCONF
- DARC
- DC9
- G21EA
- G21IP
- G2RC
- IDISP
- ISO34
- MB08-165
- NBRC
- O3ADD6
- PA
- PCONF

The GAMESS.UZH molecule database: The AL2X subset



University of Zurich > Organic Chemistry Institute > The GAMESS.UZH molecule database > subsets

Baldridge Research Group | Contact | Sitemap
University of Zurich
The GAMESS.UZH molecule database
Edit
RecentChanges
Preferences
Discussion

The AL2X subset

Original data taken from: Johnson, E. R.; Mori-Sanchez, P.; Cohen, A. J; Yang, W. *J. Chem. Phys.*, *2008*, 129, 204112. (back-corrected experimental values); all values are in kcal/mol.

Reference data

For each reaction, the relevant systems' names, the stoichiometry and the reference value are given. The systems' names refer to the geometry files (see section "GAMESS input" below). Negative values in the stoichiometry columns refer to reactants, positive values to products.

#	Systems			Stoichiometry			Ref.
0	alh3	al2h6		2	-1		35.80
1	al2f6	alf3		-1	2		52.60
2	al2cl6	alcl3		-1	2		31.40
3	albr3	al2br6		2	-1		28.40
4	alme2	al2me4		2	-1		37.80
5	alme2	alme3	al2me5	1	1	-1	30.00
6	alme3	al2me6		2	-1		21.40

Direct data download

Reference GAMESS input

- al2br6.inp
- al2cl6.inp
- al2f6.inp
- al2h6.inp
- al2me4.inp
- al2me5.inp
- al2me6.inp
- albr3.inp
- alcl3.inp
- alf3.inp

Start analysis of a molecule set

GRunDB creates a job for each molecule in the specified subset. A *session* name must be given to record the current analysis.

Subsets can be specified by their name on the web page. More than one (or *ALL*) can be analyzed in a single GRunDB go.

```
$  ./grundb.py new AL2X session1
Input file name  State (JobID)       Info
=============================================================================
al2br6           NEW (job.680)       New at Mon Nov 29 14:06:46 2010
al2cl6           NEW (job.681)       New at Mon Nov 29 14:06:46 2010
al2f6            NEW (job.682)       New at Mon Nov 29 14:06:46 2010
al2h6            NEW (job.683)       New at Mon Nov 29 14:06:46 2010
 ...
alf3             NEW (job.689)       New at Mon Nov 29 14:06:47 2010
alh3             NEW (job.690)       New at Mon Nov 29 14:06:47 2010
alme2            NEW (job.691)       New at Mon Nov 29 14:06:47 2010
alme3            NEW (job.692)       New at Mon Nov 29 14:06:47 2010
```

Submit jobs to the Grid

Once a session has been created, a single command invocation is needed to submit jobs to SMSCG or University clusters.

```
$ ./grundb.py progress session1
Insert AAI/Switch password for user  m1058036 :
Queue selected: all.q@idgc3grid01.uzh.ch
File uploaded: /tmp/rmurri/rsl.q0p5wn
File uploaded: /home/rmurri/gc3/gc3utils/0.10/grundb/take1.inp.d/AL2X/al2f6
   ...
Input file name  State (JobID)       Info
===================================================================================
al2br6           SUBMITTED (job.680)  Submitted at Mon Nov 29 14:08:04 2010
al2cl6           SUBMITTED (job.681)  Submitted at Mon Nov 29 14:07:53 2010
al2f6            SUBMITTED (job.682)  Submitted at Mon Nov 29 14:07:29 2010
al2h6            SUBMITTED (job.683)  Submitted at Mon Nov 29 14:07:41 2010
   ...
alh3             SUBMITTED (job.690)  Submitted at Mon Nov 29 14:07:50 2010
alme2            SUBMITTED (job.691)  Submitted at Mon Nov 29 14:07:59 2010
alme3            SUBMITTED (job.692)  Submitted at Mon Nov 29 14:08:02 2010
```

Monitor job progress and execution

The same command is used to monitor job execution.

```
$ ./grundb.py progress session1
Input file name  State (JobID)       Info
===============================================================
al2br6           SUBMITTED (job.680)  Submitted at Mon Nov 29 14:08:04 201
al2cl6           RUNNING (job.681)   Running at Mon Nov 29 14:08:40 2010
al2f6            RUNNING (job.682)   Running at Mon Nov 29 14:08:40 2010
al2h6            RUNNING (job.683)   Running at Mon Nov 29 14:08:40 2010
   ...
alh3             RUNNING (job.690)   Running at Mon Nov 29 14:08:40 2010
alme2            SUBMITTED (job.691)  Submitted at Mon Nov 29 14:07:59 201
alme3            SUBMITTED (job.692)  Submitted at Mon Nov 29 14:08:02 201
```

Output retrieval and post-processing

Again, `grundb progress` will automatically retrieve and post-process results of jobs that have finished execution, extracting the energy values needed to compute stoichiometry results.

```
$ ./grundb.py progress session1
File downloaded: gsiftp://idgc3grid01.uzh.ch:2811/jobs/17057129103608390550
File downloaded: gsiftp://idgc3grid01.uzh.ch:2811/jobs/17057129103608390550
   ...
Input file name  State (JobID)       Info
=============================================================================
al2br6           RUNNING (job.680)   Running at Mon Nov 29 14:09:00 2010
al2cl6           RUNNING (job.681)   Running at Mon Nov 29 14:08:40 2010
al2f6            DONE (job.682)      Final-b2plyp energy= -1084.1929109480
al2h6            DONE (job.683)      Final-b2plyp energy= -488.2225951188
   ...
alh3             DONE (job.690)      Final-b2plyp energy= -244.0835095562
alme2            DONE (job.691)      Final-b2plyp energy= -322.6947308201
alme3            DONE (job.692)      Final-b2plyp energy= -361.9996423212
```

Finally...

When all jobs are done, GRunDB computes stoichiometry data and compares it to the reference data.

```
$ ./grundb.py progress session1
Input file name  State (JobID)       Info
================================================================================
al2br6           DONE (job.682)      Final r-m06 energy is -15929.9575541891
   ...                                                   after 19 iterations
alme3            DONE (job.694)      Final r-m06 energy is -362.1226427375
                                                         after 18 iterations
STOICHIOMETRY DATA
Reaction                            Comp. energy  (Ref. data; deviation)
================================================================================
                                       AL2X
--------------------------------------------------------------------------------
2*alh3 + -1*al2h6                         +35.70  (+35.80; -0.10)
-1*al2f6 + 2*alf3                         +48.46  (+52.60; -4.14)
-1*al2cl6 + 2*alcl3                       +28.15  (+31.40; -3.25)
2*albr3 + -1*al2br6                       +25.41  (+28.40; -2.99)
2*alme2 + -1*al2me4                       +34.52  (+37.80; -3.28)
1*alme2 + 1*alme3 + -1*al2me5             +28.55  (+30.00; -1.45)
2*alme3 + -1*al2me6                       +21.72  (+21.40; +0.32)
```

What is GGamess then?

GGamess is a command-line tool to submit a set of GAMESS `.INP` files in parallel.

- For each job, manage the entire lifecycle: submit, monitor, retrieve output when done.
- Each job is independent of others.
- You can stop and restart it at a later time, processing continues from where it was interrupted.
- Finally exits when all jobs are done.

Example: running the GAMESS tests / 1

Running ggamess once submits the the jobs.

```
$ ls tests
exam01.inp  exam02.inp  exam03.inp  exam04.inp  exam05.inp
[...]
exam41.inp  exam42.inp  exam43.inp  exam44.inp

$ ./ggamess.py -r ocikbpra tests/
Status of jobs in the 'ggamess' session: (at 11:14:23, 05/05/11)
        NEW   0/44     (0.0%)
    STOPPED   0/44     (0.0%)
  SUBMITTED   44/44   (100.0%)
 TERMINATED   0/44     (0.0%)
TERMINATING   0/44     (0.0%)
      total   44/44   (100.0%)
```

Example: running the GAMESS tests / 2

Running it again updates status and fetches results of finished jobs.

You can also request a detailed listing of the jobs with the `-l` option:

```
$ ./ggamess.py -l -r ocikbpra tests/
 JobID     Job name     State                                       Info
==================================================================================
job.8945   exam42     SUBMITTED    Submitted to 'smscg' at Thu May  5 11:11:58 2011
job.8944   exam16     SUBMITTED    Submitted to 'smscg' at Thu May  5 11:12:02 2011
[...]
job.8937   exam20     TERMINATED   Execution of gamess terminated normally thu may
job.8936   exam29     TERMINATED   Execution of gamess terminated normally thu may
```

Each job has its own output directory.

```
$ ls exam29
exam29.dat  exam29.out
```

Example: running the GAMESS tests / 3

Perhaps the most intersting thing is that you can tell ggamess to keep running until all jobs are done and their output retrieved.

Example: keep running, and update job status every 20 seconds.

```
$ ./ggamess.py -r ocikbpra tests/ -C 20
Status of jobs in the 'ggamess' session: (at 11:27:48, 05/05/11)
        NEW   0/44     (0.0%)
    STOPPED   0/44     (0.0%)
  SUBMITTED   5/44    (11.4%)
 TERMINATED   39/44   (88.6%)
TERMINATING   0/44     (0.0%)
         ok   39/44   (88.6%)
      total   44/44   (100.0%)
...continues running
```

What is `gc3libs.template`?

`gc3libs.template` is a tool for generating a set of files from a template.

It's a *programming library*, not a command-line tool; you need to do some Python language programming to exploit it fully.

Example: Timm's "GAMESS benchmark" / 1

To use `gc3libs.template` you need:

1. A string with the template contents of the file;
2. actual parameters to be substituted in the template;
3. an optional "filter function" to determine which combination of parameters are acceptable.

```
GAMESS_INP = Template("""
 $$CONTRL RUNTYP=ENERGY MAXIT=1 UNITS=BOHR $$END
 $$CONTRL $SCF ISPHER=$ISPHER $$END
 $$ACCURACY ITOL=$ITOL ILOAD=$ILOAD $$END
 $$SYSTEM MWORDS=10 $$END
 $$BASIS $BASIS $$END
 $$GUESS GUESS=HUCKEL $$END

 $$DATA
$GEOMETRY
 $$END
""",
   match_ispher_with_basis,
```

Example: Timm's "GAMESS benchmark" / 2

Substitution parameters can be simple list of values (including multi-line strings).

```
GEOMETRY = [
        """Water
C1
O   8.0       0.0              0.0                     0.0
H   1.0       0.0              1.428036                1.0957706
H   1.0       0.0             -1.428036                1.0957706""",
        """Methane
Td

C     6.0   0.0            0.0            0.0
H     1.0   0.6252197764   0.6252197764   0.6252197764""",
        ],
```

Example: Timm's "GAMESS benchmark" / 3

But they can be other templates, which are recursively expanded.

```
BASIS = [Template("GBASIS=$SIMPLEBASIS",
                  SIMPLEBASIS = ["MINI", "MIDI", "DZV", "TZV",
                                 "CCD", "CCT", "CCQ", "CC5", "CC6"]),
         Template("GBASIS=$GBASIS NGAUSS=$NGAUSS $NPFUNC $NDFUNC $NFFUNC",
                  acceptable_gbasis_and_ngauss,
                  GBASIS = ["STO", "N21", "N31", "N311"],
                  NGAUSS = [2, 3, 4, 5, 6],
                  NPFUNC = ["", "NPFUNC=1"],
                  NDFUNC = ["", "NDFUNC=1"],
                  NFFUNC = ["", "NFFUNC=1"],
                  ),
         ],
```

Thank you!

(Any questions?)

Three tools for high-throughput GAMESS

GRunDB	Compute energy of a set of molecules.
GGamess	Run an arbitrary set of GAMESS `.INP` files in parallel.
gc3libs.template	Generate a set of files from a given template.

Well, and the errors...?

Errors happen, and it's not necessarily a users' fault. (e.g., disk full on a remote cluster)

How does GRunDB deal with this?

Well, and the errors...?

Errors happen, and it's not necessarily a users' fault. (e.g., disk full on a remote cluster)

How does GRunDB deal with this?

Use GC3Utils to get control over the *individual* job:

- Inspect job output and logs in the *session.out.d* directory (or use the `gtail` command)
- Take measures against the failure
- Re-submit the failed job with `gresub`
- GRunDB will notice and re-process the results when the new job is done.