

# Running Cloudy in parallel

Peter van Hoof  
Royal Observatory of Belgium

Cloudy Workshop  
Tokyo, August 2024

# Running Cloudy in parallel (1)

There are two commands in Cloudy that are parallelized:

- The optimizer
- The grid command

Both commands generate lots of individual models that can be run independently and are therefore very easy to parallelize.

First I will briefly introduce both commands.

The optimizer is typically used to determine model parameters that best reproduce a set of observables (typically line ratios from an observed spectrum plus a few other observables). It works by minimizing a  $\chi^2$ .

Cloudy has a built-in routine called PHYMIR that can minimize the  $\chi^2$  function running  $2N$  models in parallel, where  $N$  is the number of model parameters you want to optimize.

The PHYMIR algorithm was specially designed for Cloudy. Also the  $\chi^2$  value has a heuristic definition that helps create meaningful fits.

# Running Cloudy in parallel (2)

Jul 24, 24 18:28	optimize.in	Page 1/2	Jul 24, 24 18:28	optimize.in	Page 2/2
	<pre> title NGC 6720 sphere constant density cosmic rays background table star rauch 5.130187 LOG vary optimize increment 0.1 LUMINOSITY 35.982113 LOG range -7.999566 6.866524 vary optimize increment 0.2 RADIUS= 17.157808 LOG vary optimize increment 0.2 optimize range 17 17.3 HDEN 2.610496 vary optimize increment 0.2 abundances planetary no grains ELEMENT HELI ABUNDANCE 10.979260 - 12 LOG vary optimize increment 0.2 ELEMENT CARB ABUNDANCE 8.331850 - 12 LOG vary optimize increment 0.2 ELEMENT NITR ABUNDANCE 7.980493 - 12 LOG vary optimize increment 0.2 ELEMENT OXYG ABUNDANCE 8.684902 - 12 LOG vary optimize increment 0.2 ELEMENT NEON ABUNDANCE 8.041930 - 12 LOG vary optimize increment 0.2 ELEMENT SULP ABUNDANCE 6.508011 - 12 LOG vary optimize increment 0.2 ELEMENT CHLO ABUNDANCE 5.029389 - 12 LOG vary optimize increment 0.2 ELEMENT ARGO ABUNDANCE 6.264313 - 12 LOG vary optimize increment 0.2 ELEMENT IRON ABUNDANCE 5.451066 - 12 LOG vary optimize increment 0.2 GRAIN ABUND=0.209935 LOG ISM silicate vary optimize increment 0.2 print line flux seen at earth distance 740 linear parsec atom H-like levels 20 element hydrogen atom H-like levels 15 element helium stop eden 0.1 linear stop zone 2000 normalize to "H 1" 4861 scale factor 100 iterate optimize phymir 14 optimize tolerance 0.01 optimize iterations 5000 </pre>			<pre> optimize lines C 2 1335 6.882 TOTL 1549 30.28 He 2 1640 138.2 TOTL 1665 12.32 TOTL 1750 10.39 TOTL 1909 155.6 TOTL 2326 45.39 Ne 4 2424 8.416 O II 2471 7.319 # uncertainties in optical lines assume a 1-sigma unc of 0.03 in F(lambda) He 1 3614 0.280 H 1 3683 0.658 H 1 3687 0.758 H 1 3692 0.890 H 1 3697 1.072 H 1 3704 1.229 H 1 3712 1.622 O II 3726 246.5 O II 3729 257.7 H 1 3734 2.193 H 1 3750 3.078 H 1 3771 3.880 H 1 3798 5.214 He 1 3820 1.251 H 1 3835 7.328 Ne 3 3869 136.8 He 2 3924 0.188 0.16 Ne 3 3968 41.98 H 1 3970 15.09 He 1 4026 2.607 S II 4070 3.753 S II 4078 1.151 H 1 4102 27.07 He 1 4144 0.377 0.08 He 1 4169 0.096 0.31 H 1 4340 48.56 TOTL 4363 8.027 He 1 4388 0.621 He 1 4438 0.083 0.36 He 1 4471 5.005 He 2 4542 0.704 # likely misidentified or blended... # Ar 5 4626 0.010 # ... lines omitted ... O 3 51.80m 2.63e-17*100./8.32e-18*1.14*0.631/0.999 0.05 N 3 57.21m 7.90e-18*100./8.32e-18*1.14*0.631/0.999 0.05 # the FIR [O I] lines may be formed predominantly in the cold condensations # O 1 63.17m 5.33e-18*100./8.32e-18*1.14*0.631/0.999 0.05 O 3 88.33m 1.82e-17*100./8.32e-18*1.14*0.631/0.999 0.05 N 2 121.7m 4.12e-19*100./8.32e-18*1.14*0.631 0.05 # O 1 145.5m 1.81e-19*100./8.32e-18*1.14*0.631 0.05 C 2 157.6m 7.01e-19*100./8.32e-18*1.14*0.631 0.05 end optimize continuum flux at 43. micron 1.14*4e-18 W/sqcm/micron 0.20 optimize continuum flux at 115. micron 1.14*5e-19 W/sqcm/micron 0.20 optimize radio continuum flux 4850 MHz 360 mJy 0.10 optimize radio continuum flux 1400 MHz 440 mJy 0.10 optimize angular diameter 76" 0.10 </pre>	

Note that this model will not run in the current version of Cloudy!

# Running Cloudy in parallel (3)

The grid command is often used to create plots of a predicted quantity (e.g., a line ratio) as a function of one or more variables (typical examples would be the gas density and temperature).

Typical applications would be to create contour plots or BPT diagrams. The information you want to plot is often extracted from **save** commands.

All models in the grid are independent, so for large grids you can achieve high levels of parallelization.

```

Jul 20, 23 5:04          func_grid_line_ratios.in          Page 1/1
title test generating line ratios in a grid run
#
# commands controlling continuum =====
blackbody 4e4 K
ionization parameter -2
#
# commands for density & abundances =====
# these are to speed up the calculation, only do H, O, and Ne
init "honly.ini"
element oxygen on
element neon on
element sulphur on
element oxygen ionization 1 1 1
element neon ionization 1 1 1
element sulphur ionization 1 1 1
# vary the hydrogen density
hden 4 vary
grid 2 6.1 1 sequential
#
# other commands for details =====
# these are constant temperature models, vary T
constant temperature 4 vary
grid 4000 17000 3000 linear
stop zone 1
#
# commands controlling output =====
save overview "func_grid_line_ratios.ovr"
save monitors "func_grid_line_ratios.asr"
save performance "func_grid_line_ratios.per"
save line list "func_grid_line_ratios.pun" "func_grid_line_ratios.dat" ratio no hash
save grid "func_grid_line_ratios.grd"
#
# commands giving (lack of) assert =====
monitor nothing 0
#
# func_grid_line_ratios.in
# class function
# =====

This uses the grid command to compute line ratios for a wide range of
density and temperature. The ionization is set to a uniform value and
only a few elements are included. this makes the calculation faster
and prevents recombination [O III] 4363 from becoming important (there
is no O+3).

These are the line ratios mentioned as limits in the Johnstone et al.
Spitzer cooling flow filament paper (2007).

```

# Running Cloudy in parallel (4)

On the previous slides we have seen that both optimizer and grid runs can be parallelized. This can substantially reduce the run time at the expense of increased memory use.

There are two methods in use for parallelization:

- 1) Based on the `fork()` system call
- 2) Based on the Message Passing Interface (MPI)

Both methods have advantages and disadvantages which will be discussed on the next slide. It is also possible to run both commands sequentially on a single CPU, but this can take a long time!

All these methods should give essentially identical results. If they don't, you can report that as a bug.

# Running Cloudy in parallel (5)

How do you choose which method of parallelization to use?

- The `fork()` method works out of the box. It will be automatically compiled into the code on all systems that support it in a default compilation. This includes Linux and other UNIX systems, Mac OS X, and Cygwin. So this is a hassle-free solution.
- The downside is that you can only fork new processes on a single machine, i.e. you are limited to a single computer or node.

So using the default `fork()`-based method of parallelization is ideal for running small/medium-sized grids (or optimizations with few free parameters) on your laptop or desktop or even a single compute server.

- The big advantage of MPI is that it allows you to use cores on multiple nodes of an HPC machine. This allows you to use a virtually unlimited number of cores (well, as many as the admins allow you to use...)
- The downside of MPI is that it needs external support scripts and libraries. The compilation and startup of MPI codes is not standardized and can differ from one system to another. You may need to load specific modules, or set search paths to find the executables and libraries.

# Running Cloudy in parallel (6)

So for very CPU-intensive applications (large grids or optimizations with lots of free parameters) it can be advantageous to use MPI on a suitable HPC cluster.

In a default compilation, Cloudy will run the optimizer and grid commands in parallel on systems that support the `fork()` system call and where the number of cores can be determined. By default it will use all threads (except on Mac OS X, where it will ignore hyperthreads). The number of threads to use can be changed on the **optimize phymir** and **grid** commands.

How do you choose the number of cores?

- Optimizer runs can use no more than  $2 \times N$  cores simultaneously, where  $N$  is the number of parameters that are varied by the optimizer. So ideally you would use  $2 \times N$  cores, but you could also use  $N$  cores, or  $\text{ceil}((2 \times N)/3)$ , or some other small fraction of  $2 \times N$ . This choice assures that the threads are well-balanced.
- Grid runs will compute  $N_1 \times N_2 \times \dots$  independent models, where  $N_1$  is the number of grid points in the first variable,  $N_2$  is the same for the second variable, etc. For good load balancing you can choose an integer divisor of this total number of models, but this is not strictly necessary.

# Running Cloudy in parallel (7)

To compile Cloudy for an MPI run requires several steps.

- MPI typically uses a wrapper around the compiler, called something like mpiCC, but alternative names are mpicxx or mpic++. You may need to load a module to find this.
- You need to figure out which compiler the script wraps around. You can do this by typing “mpiCC --version”. Typically this will show that it is either g++ or icc (the Intel compiler).
- You need to compile Cloudy in one of the sys\_XXX subdirectories. If mpiCC wraps around g++, use the sys\_mpi\_gcc directory. If mpiCC wraps around icc, use the sys\_mpi\_icc directory. In these directories you can compile as you normally would.
- To run the MPI executable, you typically need to use the mpirun executable, i.e. “mpirun /path/to/cloudy.exe -r script”. A batch system will communicate to mpirun how many ranks to start on which nodes. Outside of a batch system, other methods are needed, like using the “-np” flag (for running on the local machine) or using a hostfile (for running on remote machines). See the man page for more info.