# `pvm_xstar`: Concurrent Execution of XSTAR in a Parallel Virtual Machine
# Reference Manual, Version 0.3.4

Michael S. Noble, `mnoble@space.mit.edu`

Aug 19, 2009

# Preface

pvm_xstar is a software tool whicn fosters concurrent execution
of the XSTAR command line application on independent sets of
parameters, within a Paralle Virtual Machine.  XSTAR is a computer
program for calculating the physical conditions and emission
spectra of photoionized gases (Kallman & Bautista 2001).


Author: Michael S. Noble <mnoble@space.mit.edu>

This software was developed at the MIT Kavli Institute for
Astrophysics; it was funded in part by NASA and the Smithsonian
Institution, through the AISRP grant NNG05GC23G and Smithsonian
Astrophysical Observatory contract SV3-73016.

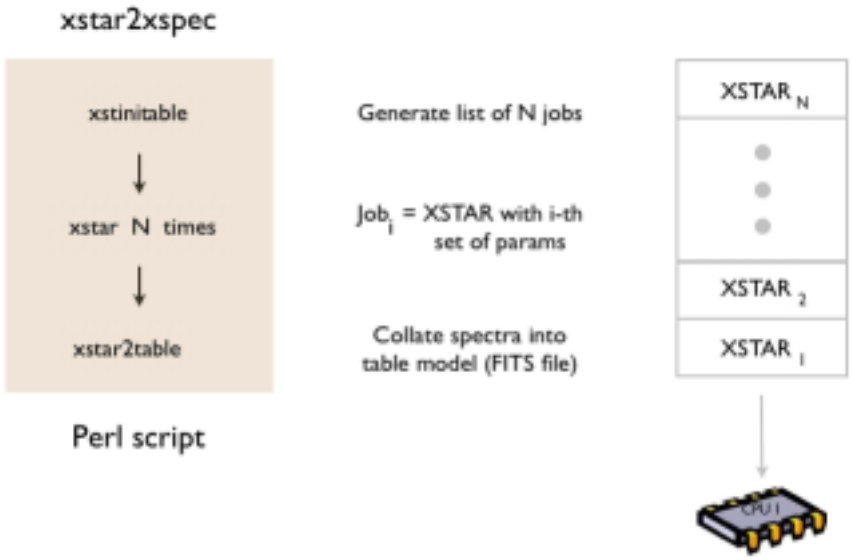
# Contents

# Chapter 1

# Introduction

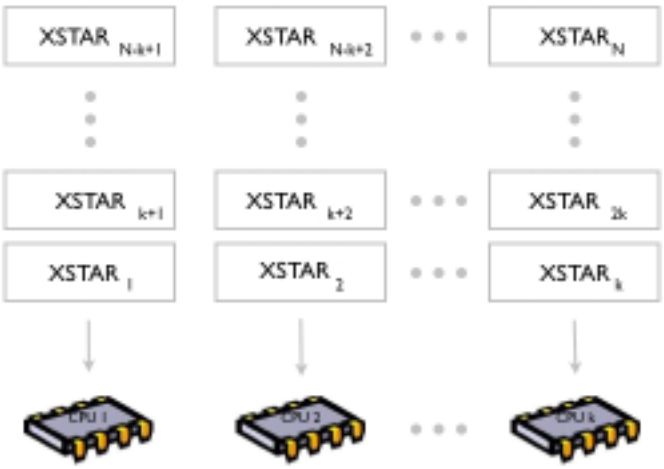
The `pvm_xstar` software tool fosters concurrent execution of the XSTAR command line application on independent sets of parameters. XSTAR is a computer code for calculating the physical conditions and emission spectra of photoionized gases (Kallman & Bautista 2001); the science it facilitates may be described most concisely by paraphrasing its documentation: a spherical gas shell surrounding a central source of ionizing radiation absorbs some of this radiation and reradiates it in other portions of the spectrum. XSTAR computes the effects on the gas of absorbing this energy, and the spectrum of reradiated light, while allowing for consideration of other sources (or sinks) of heat, such as mechanical compression & expansion, or cosmic ray scattering.

`pvm_xstar` can be used to perform large-scale XSTAR simulations, consisting of thousands or more individual XSTAR jobs, in the time formerly required to compute only a handful of such tasks. In the past this batch execution of XSTAR has been managed by `xstar2xspec`, as follows:

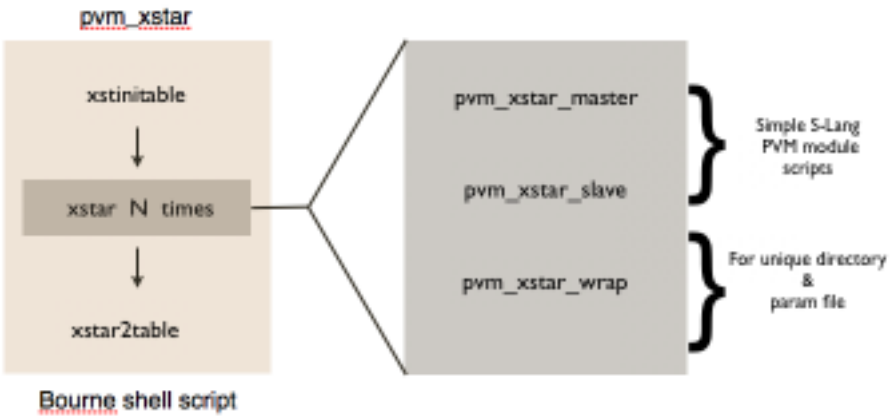



The XSTAR application is repeatedly invoked over sets of unique input parameter tuples; one spectrum is generated per XSTAR run and saved as a FITS file, and these are collated into a single FITS table model that can be incorporated into an analytic model for fitting. However, since each

XSTAR job executes completely independent of any other the entire set of jobs may be executed on multiple processors as follows:



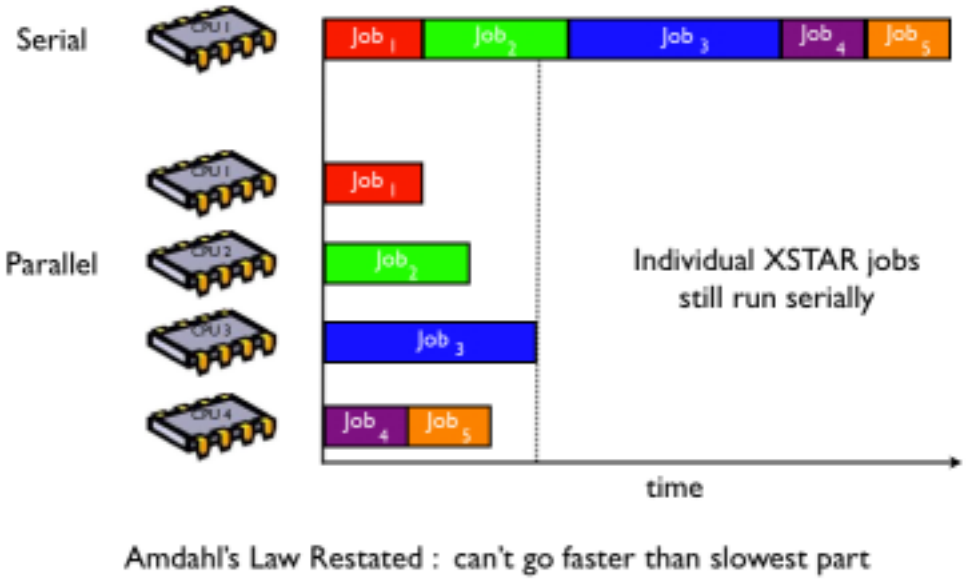

This is exactly what `pvm_xstar` does, using the (P)arallel (V)irtual (M)achine toolkit to distribute tasks across multiple workstations on a network, or multiple CPUs within a Beowulf cluster, or even multiple cores within a single desktop machine. It is effectively a parallel replacement for the serial `xstar2xspec` script, and in fact `pvm_xstar` has exactly the same flow:




For example, consider a representative simulation of 600 XSTAR jobs, generating power law spectra of Hercules X1, which consumed 26.4 hours of wallclock time on a single 2.6Ghz AMD Opteron processor with 2GB RAM; a linear scaling of this workload to 4200 jobs would consume 7.5 days on the same machine. In contrast, a similar physical simulation of 4200 XSTAR jobs completed in 110 minutes when executed via `pvm_xstar` on our Beowulf cluster of 52 2.4Ghz Opteron (4GB RAM) processors. This represents a substantial enlargement of the problem space to which XSTAR may be applied within practical timeframes.

It should kept in mind that `pvm_xstar` is merely a dispatch program: it launches instances of XSTAR & manages their execution, but does not parallelize the internals of XSTAR itself. This means that any single XSTAR task that takes, for example, 3 hours on a single CPU will still take 3 hours with `pvm_xstar`. The implication is that `pvm_xstar` cannot reduce the runtime of your entire suite of XSTAR tasks below the time required to compute the longest-running individual task. This is

commonly referred to as Amdahl's Law, and can be shown pictorially as




## 1.1 References

Kallman, T. & Bautista, M. 2001, Photoionization and High-Density Gas, Astrophysical Journal Supplements, 133, 221-253

Noble, M.S., Young, A., Ji, L., Lee, J. 2009, Parallelizing the XSTAR Photoionization Code, ASP. Conf. Ser. Proc. of Astronomical Data Analysis Software and Systems XVIII (In press. arXiv e-print at http://arxiv.org/abs/0901.1582)

# Chapter 2

# Installing pvm_xstar

After the dependencies described below are met, installing `pvm_xstar` should be easy because it does not involve any software compilation. The package consists of 4 scripts:

```
pvm_xstar            Bourne shell script
pvm_xstar_wrap       Bourne shell script
pvm_xstar_master     S-Lang script
pvm_xstar_slave      S-Lang script
```

although end-users generally need to concern themselves only with the first. Simply use the standard GNU `configure` and `make` commands within a UNIX-like (e.g. Linux, Mac OS/X terminal, or Cygwin) environment:

```
./configure [options]
make install
```

You may specify the `--prefix` option to `configure` to steer the installation to a non-default location, or use `--help` to get more information.

It is *strongly* encouraged that you test your PVM setup and `pvm_xstar` *both* before and after installation. These can be done from the top of your `pvm_xstar` distribution with:

- `make test_local`: a smoketest-style of testing which operates directly from your unpacked distribution, and can be done prior to installation
- `make test`: operational test, to be done after installation, to ensure that your PATH is set properly, your PVM hosts file enables PVM to find `pvm_xstar` and its components at runtime, and so forth.

These tests can also generate sample PVM hosts files, which can serve as templates for your own virtual machines. Refer to the `tests` subdirectory for more details.

## 2.1 Dependencies

`pvm_xstar` requires that the following software is installed on your system:

- XSTAR ( http://heasarc.nasa.gov/lheasoft/xstar/xstar.html/ )
- The PVM toolkit version 3.4.5 or later ( http://www.csm.ornl.gov/pvm/ )
- S-Lang version v2.1 or greater ( http://www.s-lang.org )
- S-Lang PVM module version v1.5 or greater ( http://space.mit.edu/cxc/modules/pvm/ )

Installing S-Lang and PVM should be relatively straightforward, because on most modern systems (e.g. Redhat, Debian or Ubuntu Linux, or Mac OS/X) they can be done with binary install tools such as `Yum, apt-get` or `MacPorts` (which for Mac OS/X we recommend over `Fink`). The S-Lang PVM module is very small, and should also take little time to download and install. Although XSTAR may be the largest download of any of these dependencies, much of that is due to the data required for its operation; as a standard Fortran 77 code that has been tested for many years, XSTAR builds cleanly on most Unix-like systems.

## 2.2 Configuring PVM

While PVM may be installed easily on modern systems via binary package management, additional user-specific configuration is usually required before it can be employed. Common PVM configuration problems include

- Forgetting to enable remote-login utilities like SSH to be used non-interactively, so that typing passwords is not required to launch slave daemons on remote hosts.
- Having a mis-configured $PATH, which can lead to messages like

```
Unable to run pvm_xstar_slave on XXX.YYY.ZZZ: not found
```

- Defining a PVM_TMP path that is greater than 64 characters long; this is the default length hard-coded into the PVM source and is unfortunately reflected verbatim in some pre-compiled PVM binary distributions.

Some potential solutions are outlined below.

### 2.2.1 Passwordless SSH

If you are using PVM on a single machine, such as a multicore workstation or massively-parallel-processor (MPP), then you will not need to perform remote logins and can skip this section.

A common way of automating remote SSH logins involves generating keys on the master,

```
master%  ssh-keygen -t dsa
...
```

copying the public keyfile that's generated to each slave host,

```
master%  scp ~/.ssh/id_dsa.pub slave-host-1:master.key
```

adding it to the authorized keys file for that host,

```
slave-host-1%  cd ~/.ssh
slave-host-1%  cat ~/master.key >> authorized_keys
```

and ensuring that authorized_keys has the required permissions:

```
slave-host-1%  chmod 600 authorized_keys
```

Another option is to use `ssh-agent`.

### 2.2.2 Configuring $PATH

Enabling the PVM daemon to access your `pvm_xstar` scripts can be done in several ways, such as installing them into a system-wide location accessible to all users at login, but in our experience one of the most reliable methods has been to add a line like

```
* ep=$PATH
```

or

```
* ep=<path/to/your/pvm_xstar/scripts>
```

to your PVM hosts file, as described in the pvm_intro(1) and pvmd3 man page documentation. When the `* ep=` option appears at the top of your hosts file it applies to all hosts in the virtual machine, but PVM allows different values to be used simultaneously by defining `ep` on the hostname line, such as

```
mylinux.mysite  ep=/nfs/local/bin
mymac.mysite    ep=/usr/local/bin
```

# Chapter 3

# Using pvm_xstar

pvm_xstar has a simple text-mode interface, and will most often be executed directly from the interactive command line. A typical session might resemble:

```
linux%  pvm_xstar /home/mnoble/data  <RETURN>

pvm_xstar Version 0.3.0
Using PVM virtual machine id: pvm_xstar-17474

BEGIN: Fri Nov 21 16:33:13 EST 2008

Output files will be written to: /home/mnoble/data/pvm_xstar

xstinitable v1.0
Compiled: Mar 11 2008 17:09:25
spectrum type?[pow]                                  <RETURN>
radiation temperature or alpha soft maximum?[-1.]  <RETURN>
covering fraction soft maximum (0.:1.) [1.]        <RETURN>
density soft maximum (cm**-3) (0.:1.E18) [1.E11]   <RETURN>

...

Slave running pvm_xstar_slave spawned on rabble with task-id 262147

Slave running pvm_xstar_slave spawned on node12 with task-id 524289

Slave running pvm_xstar_slave spawned on node11 with task-id 786433

...
```

The working directory is a mandatory parameter which specifies where both intermediate results as well as the final output table model will be written. pvm_xstar requires that XSTAR be properly configured prior to its execution, and attempts to verify this by examining the $HEADAS environment variable.

After prompting for XSTAR physical parameters `pvm_xstar` attempts to launch a virtual machine and submit the XSTAR job list. By default `pvm_xstar` looks for a PVM hosts file in $HOME/.pvmhosts, but that can be customized with the `-h` option described below. To allow multiple PVM workloads to be processed simultaneously sans interference, `pvm_xstar` tags each virtual machine it launches with a unique identifier (PVM_VMID, as described in pvm_intro(1) man pages). By default this identifier will be `pvm_xstar-XXXXX` where XXXXX is a 5-digit integer generated pseudo-randomly each time `pvm_xstar` is invoked, however the user can specifying a custom id by using the `-v` option.

## 3.1 Command Line Options

Invoking `pvm_xstar` with `--help` or any unrecognized option will cause it to emit a help message:

```
pvm_xstar: manages parallel execution of multiple XSTAR jobs, with PVM
Version 0.3.2

Usage:  pvm_xstar [options] WorkDir [xstinitable parameters | joblist]

WorkDir must be writable and $HEADAS must be configured.

Supported options are:
  -h <file>  use this file to launch PVM, instead of $HOME/.pvmhosts
  -i <exe>   use exe to initialize work dir/env of each parallel job
  -l <log>   redirect console output to log file
  -nph <N>   set max number processes per host to N (default: 4)
  -p         ping: returns path through which pvm_xstar was invoked
  -v <vmid>  use this virtual machine id, instead of pvm_xstar-XXXXX
  -V         display the version number of pvm_xstar

Normally xstinitable will be launched to prompt for XSTAR physical
parameters and generate a list of XSTAR jobs to run in parallel.
This can be customized by supplying xstinitable parameters on the
command line (such as mode=h) OR by supplying the name of an
existing joblist file, in which case xstinitable will not be run
nor will the generated spectra be collated into a single table
model with xstar2table.
```

### 3.1.1 Customizing the Number of Processors Used

By default `pvm_xstar` will use at most 4 processors per host in the virtual machine. If you want to change that, say to use all 8 cores of a multicore workstation, specify the option `-nph 8`.

### 3.1.2 Capturing Console Output To a Log File

Both XSTAR and `pvm_xstar` can emit large volumes of output to the console. For this reason we advocate the use of a log file, such as

```
master%  pvm_xstar -l run.log ...
```

While this may seem superfluous in light of standard UNIX output redirection

```
master%  pvm_xstar ... > run-stdout.log
```

it isn't, because the latter would also capture the prompts for XSTAR input parameters, which is undesirable for interactive use. Progress can be monitored during long-running computations by viewing a snapshot of the log directly in an editor, or using tail to see incremental updates:

```
master%  tail -f run.log
```