

SGI UV (Predator)

User Guide

Air Force Research Laboratory (AFRL)

DoD Supercomputing Resource Center (DSRC)

## Contents

1	Introduction.....	4
1.1	Document Scope & Assumptions .....	4
1.2	Policies to Review.....	4
1.3	Obtaining Accounts.....	4
1.4	Requesting Assistance .....	4
2	System Configuration .....	5
2.1	Processors.....	6
2.2	Memory.....	6
2.3	Operating System .....	6
2.4	File Systems .....	7
3	Accessing the System .....	7
3.1	Kerberos .....	7
3.2	Logging In.....	7
3.3	File Transfers.....	7
4	User Environment.....	8
4.1	User Directories .....	8
4.1.1	Home Directory .....	8
4.1.2	Work Directory .....	8
4.2	Shells.....	9
4.3	Environment Variables.....	9
4.3.1	Login Environment Variables .....	9
4.3.2	Batch-Only Environment Variables.....	11
4.4	Modules .....	12
4.5	Archive Usage .....	12
4.5.1	Archival Command Synopsis .....	12
5	Program Development.....	13
5.1	Programming Models.....	13
5.1.1	Message Passing Interface (MPI) .....	13
5.1.2	Shared Memory.....	15
5.1.3	Open Multi-Processing (OpenMP) .....	16
5.1.4	Hybrid MPI/OpenMP .....	17
5.2	Available Compilers .....	17
5.2.1	GCC Compiler Programming Environment .....	17
5.2.2	Intel Compiler Programming Environment.....	18
5.3	Libraries.....	18
5.3.1	Intel Math Kernel Libraries (Intel MKL).....	18
5.3.2	Additional Math Libraries.....	18
5.4	Debuggers .....	19
5.4.1	gdb .....	19
5.4.2	TotalView .....	19

5.5	Code Profiling and Optimization .....	19
5.5.1	SGI perfcatch .....	19
5.5.2	Additional Profiling Tools.....	20
5.5.3	Performance Optimization Methods .....	20
6	Batch Scheduling.....	21
6.1	Scheduler.....	21
6.2	Queue Information .....	21
6.3	Interactive Logins.....	22
6.4	Interactive Batch Sessions.....	22
6.5	Batch Request Submission.....	23
6.6	Batch Resource Directives.....	23
6.7	Sample Scripts.....	24
6.8	PBS Commands .....	26
6.9	Advance Reservations .....	26
7	Software Resources .....	26
7.1	Application Software .....	26
7.2	Useful Utilities .....	27
7.3	Sample Code Repository .....	27
8	Links to Vendor Documentation.....	27
8.1	SGI Links .....	27
8.2	Red Hat Links .....	27
8.3	GNU, Intel Links.....	27

# 1 Introduction

---

## 1.1 Document Scope & Assumptions

This document provides an overview and introduction to the use of the SGI UV (Predator) located at the AFRL DSRC and a description of the specific computing environment on Predator. The intent of this guide is to provide information to enable the average user to perform computational tasks on the system. To receive the most benefit from the information provided here, you should be proficient in the following areas:

- Use of the LINUX operating system
- Use of an editor (e.g. vi or emacs)
- Remote usage of computer systems via network or modem access
- A selected programming language and its related tools and libraries

## 1.2 Policies to Review

All policies are discussed in the [Policies and Procedures](#) section on the AFRL DSRC User Documentation page. All users running at the AFRL DSRC are expected to know, understand, and follow the policies discussed. If you have any questions about AFRL DSRC's policies, please contact the CCAC.

## 1.3 Obtaining Accounts

Authorized DOD and Contractor personnel may request an account on Predator through their Service Agency Approval Authority (S/AAA) and Principle Investigator (PI). Your S/AAA and PI will assign you an HPMCP project that allows you to have allocations on the HPC systems at the DSRC. As you go through the process, you may want to refer to our [step-by-step guide](#). More information can be found through the HPCMP [Portal to Information Environment \(PIE\)](#) or through the [Consolidated Customer Assistance Center \(CCAC\)](#).

## 1.4 Requesting Assistance

The Consolidated Customer Assistance Center (CCAC) is available to help users with any problems, questions, or training requirements for our HPC systems. Analysts are on duty Monday - Friday, 8:00 a.m. to 11:00 p.m. Eastern Time.

Web: <http://centers.hpc.mil/>

Email: [help@ccac.hpc.mil](mailto:help@ccac.hpc.mil)

Phone: 1-877-CCAC-039 (1-877-222-2039) or 937-255-0679

Fax: 937-656-9538

## 2 System Configuration

Predator is an SGI UV System. The login and compute nodes populated with 2.7-GHz Intel E5 Sandy Bridge 64-bit, 8-core processors. Predator uses a dedicated Numalink v6 communications network for MPI messages and IO traffic, a direct-attached XFS file system, and has 1004 compute cores that can share memory with every other core in a cpuset.

Each compute node has two 8-core processors (16 cores) with its own Redhat Enterprise Linux (RHEL) operating system and 32 GBytes of DDR3 memory, with no user-accessible swap space. Predator has 2.7 PBytes (formatted) of disk storage.

Predator's intended use is as a batch-scheduled HPC system and, as such, its login nodes are not to be used for large computational (e.g. memory, IO, long executions) work. All executions that require large amounts of system resources must be sent to the compute nodes by batch job submission.

Node Configuration		
	Login Nodes	Compute Nodes
<b>Total Nodes</b>	1	1
<b>Operating System</b>	Redhat Enterprise Linux (RHEL)	Redhat Enterprise Linux (RHEL)
<b>Cores/Node</b>	8	1016
<b>Core Type</b>	Intel E5 Sandy Bridge	Intel E5 Sandy Bridge
<b>Core Speed</b>	2.7 GHz	2.7 GHz
<b>Memory/core</b>	4 GBytes	4 GBytes
<b>User Accessible Memory/core</b>	3.5 GBytes	3.5 GBytes
<b>Memory Model</b>	Shared	Shared
<b>Interconnect Type</b>	Numalink v6	Numalink v6
File Systems		
File System	File System Type	Formatted Capacity
<b>/workspace</b>	XFS	88 TBytes

(\$WORKDIR file system)		
/home (\$HOME file system)	XFS	88 TBytes

Figure 1: Configuration and File System Types

## 2.1 Processors

Predators uses the same Intel Sandy Bridge E5-4650 64-bit processors on its login and compute nodes and are clocked at 2.7 GHz and has a total of 1004 compute cores available for batch processing.

Each processor has 8x32 KBytes of L1 instruction cache, 8x32 KBytes of L1 data cache, 8x256 KBytes of L2 cache, and access to a 20 MByte L3 cache that is shared among all 8 cores of the processor.

## 2.2 Memory

Predator uses a shared memory model, with memory being shared among all the cores across the cluster.

The login node contains 128 GBytes of main memory. All memory and cores on the node are shared among all users who are logged in. Therefore, users should not use more than 8 GBytes of memory at any one time.

Each compute core contains 3.5 GBytes of user accessible shared memory for a cluster total of 3.5 TBytes of shared memory.

## 2.3 Operating System

Predator's Operating system is SGI's Performance Suite, which is a combination of then Red Hat Enterprise Linux (RHEL) Operating System and SGI-specific optimizations, utilities, and tools. More information on SGI Performance Suite can be found at the following URL:

- <http://www.sgi.com/products/software/sps.html>

The compute nodes can provide access to dynamically shared objects, most typical Linux commands and basic functionality, either by compiling your application (in the case of shared objects) or by including the command in the PBS batch submission script.

## 2.4 File Systems

Predator has the following file systems available for user storage, however, please note that these file systems are part of the same direct-attached XFS file system. As such, they share the same available space of 88 TBytes.

### **\$HOME**

On the compute nodes, `$HOME` is a locally mounted XFS file system with a formatted capacity of 88 TBytes. This file system is mounted to the interactive node via NFS. All users have a home directory located on this file system which can be referenced by the environment variable `$HOME`.


### **\$WORKDIR**

On the compute nodes, `$WORKDIR` is a locally mounted XFS file system with a formatted capacity of 88 TBytes. This file system is mounted to the interactive node via NFS. All users have a home directory located on this file system which can be referenced by the environment variable `$WORKDIR`.

## 3 Accessing the System

---

### 3.1 Kerberos

A Kerberos client kit must be installed on your desktop to enable Kerberos authentication. Kerberos is a network authentication tool that provides secure communication by using secret cryptographic keys. Only users with a valid HPCMP Kerberos authentication can gain access to Predator. More information about installing Kerberos clients on your desktop can be found at the [CCAC Support page](#) .

### 3.2 Logging In

The system host name for the Predator cluster is `predator.afrl.hpc.mil`. The IP address to this node is available upon request from CCAC.

The preferred login to Predator is ssh via the following command:

```
% ssh predator.afrl.hpc.mil
```

Kerberized `telnet` and `rlogin` are also allowed.

### 3.3 File Transfers

File transfers to DSRC systems must be performed using Kerberized versions of the following tools: `scp`, `ftp`, `sftp`, and `mpscp`, except file transfers to the local archive system.

## 4 User Environment

---

### 4.1 User Directories

#### 4.1.1 Home Directory

Each user is allocated a home directory (the current working directory immediately after login) of permanent storage that is backed up. The home directory can be referenced locally with the `$HOME` environment variable from all nodes in the system.

Due to the configuration of the direct-attached XFS file system that `$HOME` and `$WORKDIR` share, there is currently no technical way to enforce a quota system on the `$HOME` file system. Since `$HOME` and `$WORKDIR` share the same file system, space consumed by one directory structure will affect the other.

#### 4.1.2 Work Directory

Predator has one large file system (`$WORKDIR`) for the temporary storage of data files needed for executing programs. You may access your personal working directory by using the `$WORKDIR` environment variable, which is set for you upon login. Your `$WORKDIR` directory has no disk quotas, and files stored there do not affect your permanent file quota usage. Because of high usage, the `$WORKDIR` file system tends to fill up frequently. Please review the [Workspace Policy](#) and be mindful of your disk usage.

**REMEMBER:** `$WORKDIR` is a "scratch" file system and is not backed up. You are responsible for managing files in your `$WORKDIR` by backing up files to the MSAS and deleting unneeded files when your jobs end.

All of your jobs should execute from your `$WORKDIR` directory, not `$HOME`. While not technically forbidden, jobs that are run from `$HOME` are subject to disk space quotas and have a much greater chance of failing if problems occur with that resource. Jobs that are run entirely from your `$WORKDIR` directory are more likely to complete, even if all other resources are temporarily unavailable.

If you use `$WORKDIR` in your batch scripts, you must be careful to avoid having one job accidentally contaminate the files of another job. One way to avoid this is to use the `$JOBDIR` (or `$WORK_DIR`) directory which is unique to each job on the system. The `$JOBDIR` directory is not subject to the [Workspace Policy](#) until the job exits the workload management system.

**REMEMBER:** Since `$HOME` and `$WORKDIR` share the same file system, space consumed by one directory structure will affect the other.



## 4.2 Shells

The following shells are available on Predator: `cs`, `bash`, `ksh`, `tcsh`, and `sh`. To change your default shell, please modify your preferred shell entry in pIE (<https://ieapp.erd.c.hpc.mil>). Once changed in pIE, your preferred shell will become your default shell on the spirit cluster within 48 hours.

## 4.3 Environment Variables

A number of environment variables are provided by default on all HPCMP high performance computing (HPC) systems. We encourage you to use these variables in your scripts where possible. Doing so will help to simplify your scripts and reduce portability issues if you ever need to run those scripts on other systems.

### 4.3.1 Login Environment Variables

The following environment variables are common to both the login and batch environments:

Common Environment Variables	
Option	Purpose
<code>\$ARCHIVE_HOME</code>	Your directory on the archive server.
<code>\$ARCHIVE_HOST</code>	The host name of the archive server.
<code>\$BC_HOST</code>	The generic (not node specific) name of the system.
<code>\$CC</code>	The currently selected C compiler. This variable is automatically updated when a new compiler environment is loaded.
<code>\$CENTER</code>	Your directory on the Center-Wide File System (CWFS).
<code>\$CSI_HOME</code>	The path to the directory for the following list of heavily used application packages: ABAQUS, Accelrys, ANSYS, CFD++, Cobalt, EnSight, Fluent, GASP, Gaussian, LS-DYNA, MATLAB, and TotalView, formerly known as the Consolidated Software Initiative (CSI) list. Other application software may also be installed here by our staff.
<code>\$CXX</code>	The currently selected C++ compiler. This

	variable is automatically updated when a new compiler environment is loaded.
\$DAAC_HOME	The path to the directory containing the ezVIZ visualization software.
\$F77	The currently selected Fortran F77 compiler. This variable is automatically updated when a new compiler environment is loaded.
\$F90	The currently selected Fortran 90 compiler. This variable is automatically updated when a new compiler environment is loaded.
\$HOME	Your home directory on the system.
\$JAVA_HOME	The path to the directory containing the default installation of JAVA
\$KRB5_HOME	The directory containing the Kerberos utilities.
\$PET_HOME	The path to the directory containing the tools installed by the PET CE staff. The supported software includes a variety of open-source math libraries (see <a href="#">BC policy Policy FY06-01</a> ) and open-source performance and profiling tools (see <a href="#">BC policy Policy FY07-02</a> ).
\$PROJECTS_HOME	A common directory where group-owned and supported applications and codes may be maintained for use by members of a group. Any project may request a group directory under \$PROJECTS_HOME.
\$SAMPLES_HOME	The path to the <a href="#">Sample Code Repository</a> . This is a collection of sample scripts and codes is provided and maintained by our staff to help users learn to write their own scripts. There are a number of ready-to-use scripts for a variety of applications.
\$WORKDIR	Your work directory on the local temporary file system (i.e., local high-speed disk).

### 4.3.2 Batch-Only Environment Variables

In addition to the variables listed above, the following variables are automatically set only in your batch environment. That is, your batch scripts will be able to see them when they run. These variables are supplied for your convenience and are intended for use inside your batch scripts.

Batch-Only Environment Variables	
Option	Purpose
<code>\$BC_CORES_PER_NODE</code>	The number of cores per node for the compute node on which a job is running.
<code>\$BC_MEM_PER_NODE</code>	The approximate maximum user-accessible memory per node (in integer MBytes) for the compute node on which a job is running.
<code>\$BC_MPI_TASKS_ALLOC</code>	The number of MPI tasks allocated for a job.
<code>\$BC_NODE_ALLOC</code>	The number of nodes allocated for a job.

## 4.4 Modules

Software modules are a very convenient way to set needed environment variables and include necessary directories in your path so that commands for particular applications can be found. Predator uses "modules" to initialize your environment with system commands and libraries and compiler suites,

A number of modules are loaded automatically as soon as you log in. To see the modules which are currently loaded, run "module list". To see the entire list of available modules, run "module avail". You can modify the configuration of your environment by loading and unloading modules.

## 4.5 Archive Usage

All of our HPC systems share an on-line Mass Storage Archival system (MSAS) that currently has more than 43 TBytes of Tier 1 archival storage (disk cache) and 6 PBytes of Tier 2 high speed archival storage utilizing a robotic tape library. Every user is given an account on the MSAS.

Kerberized login and ftp are allowed into the MSAS system. Locally developed utilities may be used to transfer files to and from the MSAS as well as to create and delete directories, rename files, and list directory contents. For convenience, the environment variable `$ARCHIVE_HOME` can be used to reference your MSAS archive directory when using archive commands. The command `getarchome` can be used to display the value of `$ARCHIVE_HOME` for any user.

### 4.5.1 Archival Command Synopsis

A synopsis of the main archival utilities is listed below. For information on additional capabilities, see the [Archive User's Guide](#) or read the on-line man pages that are available on each system. These commands are non-Kerberized and can be used in batch submission scripts if desired.

Copy one or more files from the MSAS:

```
archive get [-C path] [-s] file1 [file2...]
```

List files and directory contents on the MSAS:

```
archive ls [lsopts] [file/dir ...]
```

Create directories on the MSAS:

```
archive mkdir [-C path] [-m mode] [-p] [-s] dir1  
[dir2 ...]
```

Copy one or more files to the MSAS:

```
archive put [-C path] [-D] [-s] file1 [file ...]
```

## 5 Program Development

---

### 5.1 Programming Models

Predator supports three base programming models: Message Passing Interface (MPI), Shared-Memory (SMP, SHMEM), and Open Multi-Processing (OpenMP). A Hybrid MPI/OpenMP programming model is also supported. MPI and SHMEM are examples of the message- or data-passing models, while OpenMP only uses shared memory on a node by spawning threads.

#### 5.1.1 Message Passing Interface (MPI)

Predator utilized SGI's Message Passing Toolkit (MPT) by default, but also has Intel's Message Passing Interface (IntelMPI) available for those that require it.

Both MPI implementations support the MPI 2.2 standard, as documented by the MPI Forum.

MPI establishes a practical, portable, efficient, and flexible standard for message passing that makes use of the most attractive features of a number of existing message-passing systems, rather than selecting one of them and adopting it as the standard. See "man mpi" for additional information.

A copy of the MPI 2.2 Standard, in PDF format, can be found at the following URL:

- <http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf>

When creating an MPI program on Predator, ensure that the following actions are taken:

- Make sure the Message Passing Toolkit (`module mpt` or `module intelmpi`) is loaded. To check this, run the "module list" command. If the `mpt` or `intelmpi` module is not listed, use the command, "module load mpt" or "module load intelmpi".

- Make sure the source code includes one of the following lines :

```
INCLUDE "mpif.h"    //for Fortran, or
#include <mpi.h>    //for C/C++
```

To compile an MPI program, use the following examples:

```
mpif90 -o mpi_program mpi_program.f    //for Fortran
mpicc -o mpi_program mpi_program.c    //for C
mpicxx -o mpi_program mpi_program.c    //for C++
```

To run an MPI program within a batch script, use the following command:

```
mpiexec_mpt -np ${BC_NODE_ALLOC} ./mpi_program
[user_arguments]
```

The `mpiexec_mpt` command launches executables across a set of compute cores allocated to your job. When each member of the parallel application has exited, `mpiexec_mpt` exits.

A common concern for MPI users is the need for more memory for each process. By default, one MPI process is started on each core of your cpu allocation. This means that on Predator, the available memory on the node is split between the total number of cores you request. To allow an individual process to use more of the node's memory, you need to start fewer processes than your total core allocation. To accomplish this, the user must request more cores from PBS, but only run on a certain number of them. For example, the following select statement requests 16 cores, but only uses 8 of those cores for MPI processes:

```
#PBS -l select=1:ncpus=16:mpiprocs=8
```

To tell the system to only start your job across those 8 cores, the following `mpiexec_mpt` command can be used:

```
mpiexec_mpt -np ${BC_NODE_ALLOC} ./mpi_program [user_arguments]
```

For more information about `mpiexec_mpt`, type "man `mpiexec_mpt`".

### 5.1.2 Shared Memory

These logically shared, distributed-memory access routines provide high- performance, high- bandwidth communication for use in highly parallelized scalable programs. The SHARED MEMORY data- passing library routines are similar to the MPI library routines: they pass data between cooperating parallel processes. The SHMEM data- passing routines can be used in programs that perform computations in separate address spaces and that explicitly pass data to and from different processes in the program.

The SHMEM routines minimize the overhead associated with data- passing requests, maximize bandwidth, and minimize data latency. Data latency is the length of time between a process initiating a transfer of data and that data becoming available for use at its destination.

SHMEM routines support remote data transfer through put operations that transfer data to a different process and get operations that transfer data from a different process. Other supported operations are work- shared broadcast and reduction, barrier synchronization, and atomic memory updates. An atomic memory operation is an atomic read and update operation, such as a fetch and increment, on a remote or local data object. The value read is guaranteed to be the value of the data object just prior to the update. See "man intro\_shmem" for details on the SHMEM library.

When creating a SHMEM program on Predator, ensure that the following actions are taken:

- Make sure the Message Passing Toolkit (module `mpt`) is loaded. To check this, run the "module list" command. If the `mpt` module is not listed, use the command, "module load mpt" to load it.
- The source code includes one of the following lines:

```
INCLUDE 'mpp/shmem.fh'    //for Fortran, or
#include <mpp/shmem.h>    //for C
```
- The compile command includes an option to reference the SHMEM library.

To compile a SHMEM program, use the following examples:

```
ifort -o shmem_program shmem_program.f90 -lsma -lmpi    //for Fortran
icc -o shmem_program shmem_program.c -lsma -lmpi        //for C
ipcp -o shmem_program shmem_program.c -lsma -lmpi++    //for C++
```

The program can then be launched using the `mpiexec_mpt` command as follows:

```
mpiexec_mpt -np N ./shmem_program [user_arguments]
```

where  $N$  is the number of processes being started, with each process utilizing one core. The `mpiexec_mpt` command launches executables across a set of CNL compute nodes. When each member of the parallel application has exited, `mpiexec_mpt` exits. For more information about `mpiexec_mpt`, type “`man mpiexec_mpt`”.

### 5.1.3 Open Multi-Processing (OpenMP)

OpenMP is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. OpenMP is a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications.

When creating an OpenMP program on Predator, take the following actions:

- If using OpenMP functions (for example, `omp_get_wtime`), ensure that the source code includes the line, “`USE omp_lib`”

Or, includes one of the following:

```
INCLUDE 'omp.h'      //for Fortran, or
#include <omp.h>     //for C
```

- The compile command includes an option to reference the OpenMP library.

To compile an OpenMP program, use the following examples:

For C codes:

```
icc -openmp -o OpenMP_program OpenMP_program.c //Intel
cc -fopenmp -o OpenMP_program OpenMP_program.c //GNU
```

For Fortran codes:

```
ifort -openmp -o OpenMP_program OpenMP_program.f //Intel
gfortran -fopenmp -o OpenMP_program -fopenmp OpenMP_program.f //GNU
```



To run an OpenMP program within a batch script, you also need to set the `$OMP_NUM_THREADS` environment variable to the number of threads in the team. For example:

```
setenv OMP_NUM_THREADS 16
omplace ./OpenMP_program [user_arguments]
```

In the example above, the application starts `OpenMP_program` on 16 threads; one thread per core.

#### 5.1.4 Hybrid MPI/OpenMP

An application built with the hybrid model of parallel programming can run on Predator using both OpenMP and Message Passing Interface (MPI).

When creating a hybrid (MPI/OpenMP) program on Predator, follow the instructions in the MPI and OpenMP sections above for creating your program. Then use the compilation instructions for OpenMP.

To run a hybrid program within a batch script, set `$OMP_NUM_THREADS` equal to the number of threads in the team. Then launch your program using `mpiexec_mpt` as follows:

```
setenv OMP_NUM_THREADS 16
mpiexec_mpt -np N omplace ./mpi_program [user_arguments]
```

Where  $N$  is the number of MPI tasks.

## 5.2 Available Compilers

The following compiler suites are available on Predator:

- GCC compiler programming environment
- Intel compiler programming environment

### 5.2.1 GCC Compiler Programming Environment

The GCC Compiler Programming Environment can be accessed by loading the module `*gnu-compilers/[version level]*`.

```
module load gnu-compilers/[version level]
```

## 5.2.2 Intel Compiler Programming Environment

The Intel Compiler Programming Environment can be accessed by loading the module `*intel-compilers/[version level]*`

```
module load intel-compilers/[version level]
```

## 5.3 Libraries

### 5.3.1 Intel Math Kernel Libraries (Intel MKL)

Predator provides the Intel Math Kernel Libraries (Intel MKL), a set of numerical routines tuned specifically for Intel platform processors. The routines, which are available via both FORTRAN and C interfaces, include:

- Basic Linear Algebra Subroutines (BLAS) - Levels 1, 2, and 3
- Linear Algebra Package (LAPACK)
- Fast Fourier Transform (FFT) routines for single-precision, double-precision, single-precision complex, and double-precision complex data types
- Random Number Generator
- Fast Math and Fast Vector Library

Linking to the Intel Math Kernel Libraries can be complex and is beyond the scope of this introductory guide. Documentation explaining the full feature set along with instructions for linking can be found at the following URLs:

- <http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation>

Intel also makes a link advisor available to assist users with selecting proper linker and compiler options:

- <http://software.intel.com/sites/products/mkl/>

### 5.3.2 Additional Math Libraries

There is also an extensive set of Math libraries available in the `$PET_HOME/MATH` directory on Predator. Information about these libraries may be found on the Baseline Configuration Web site at [BC policy FY06-01](#).

## 5.4 Debuggers

### 5.4.1 gdb

The GNU Project Debugger (gdb) is a source level debugger that can be invoked either with a program for execution or a running process id. To launch your

```
gdb a.out corefile
```

program under gdb for debugging, use:

To attach gdb to a program that is already executing on this node, use the following command:

```
gdb a.out pid
```

For more information, the GDB manual can be found at <http://www.gnu.org/software/gdb/> .

## 5.5 Code Profiling and Optimization

Profiling is the process of analyzing the execution flow and characteristics of your program to identify sections of code that are likely candidates for optimization, which increases the performance of a program by modifying certain aspects for increased efficiency.

### 5.5.1 SGI perfcatch

SGI perfcatch will run an MPI or SHMEM program with a wrapper profiling library that prints communication and synchronization call profiling information to a summary file upon program completion.

To use perfcatch, insert the `perfcatch` command in front of the executable name:

```
mpiexec_mpt -np N perfcatch ./mpi_program [user_arguments]
```

Where  $N$  is the number of MPI tasks.

Man pages are available for `perfcatch`. Additional information can be found in [Chapter 9 of SGI's Message Passing Toolkit User Guide](#).

Additional profiling options are available. See "man `perfcatch`" for additional instrumentation options.

## 5.5.2 Additional Profiling Tools

There is also a set of profiling tools available in the `$PET_HOME/pkg` directory on Predator. Information about these tools may be found on the Baseline Configuration Web site at [BC policy FY07-02](#).

## 5.5.3 Performance Optimization Methods

Optimization generally increases compilation time and executable size, and may make debugging difficult. However, it usually produces code that runs significantly faster. The optimizations that you can use will vary depending on your code and the system on which you are running.

Note: Before considering optimization, you should always ensure that your code runs correctly and produces valid output.

In general, there are four main categories of optimization:

- Global Optimization
- Loop Optimization
- Inter-Procedural Analysis and Optimization(IPA)
- Function Inlining

### 5.5.3.1 Global Optimization

A technique that looks at the program as a whole and may perform any of the following actions:

- Performed on code over all its basic blocks
- Performs control-flow and data-flow analysis for an entire program
- Detects all loops, including those formed by IF and GOTOs statements and performs general optimization.
- Constant propagation
- Copy propagation
- Dead store elimination
- Global register allocation
- Invariant code motion
- Induction variable elimination

### 5.5.3.2 Loop Optimization

A technique that focuses on loops (for, while, etc.) in your code and looks for ways to reduce loop iterations or parallelize the loop operations. The following types of actions may be performed:

- Vectorization – rewrites loops to improve memory access performance. Some compilers may also support automatic loop vectorization by converting loops to utilize low-level hardware instructions and registers if they meet certain criteria.
- Loop unrolling – (also known as "unwinding") replicates the body of loops to reduce loop branching overhead and provide better opportunities for local optimization.
- Parallelization – divides loop operations over multiple processors where possible.

### 5.5.3.3 Inter-Procedural Analysis and Optimization (IPA)

A technique that allows the use of information across function call boundaries to perform optimizations that would otherwise be unavailable.

### 5.5.3.4 Function Inlining

Function Inlining is a technique that seeks to reduce function call and return overhead. It is:

- Used with functions that are called numerous times from relatively few locations.
- Allows a function call to be replaced by a copy of the body of that function.
- May create opportunities for other types of optimization
- May not be beneficial.

Improper use of this form of optimization may increase code size and actually result in less efficient code.

## 6 Batch Scheduling

---

### 6.1 Scheduler

The Portable Batch System (PBS) is currently running on Predator. It schedules jobs and manages resources and job queues, and can be accessed through the interactive batch environment or by submitting a batch request. PBS is able to manage both single and multiprocessor jobs.

### 6.2 Queue Information

The following table describes the PBS queues available on Predator:

Priority	Queue	Job	Max	Max	Comments
----------	-------	-----	-----	-----	----------

	Name	Class	Wall Clock Time	Cores Per Job	
Highest	urgent	Urgent	168 Hours	508	Jobs belonging to DoD HPCMP Urgent Projects.
	debug	Debug	1 Hour	16	User testing; In the debug queue, you may use 736 cores for 1 hour or 1472 cores for ½ hour.
			½ Hour	32	
	high	High	168 Hours	508	Jobs belonging to DoD HPCMP High Priority Projects.
	challenge	Challenge	168 Hours	508	Jobs belonging to DoD HPCMP Challenge Projects.
	standard	Standard	168 Hours	508	Standard jobs
	transfer	N/A	12 Hours	1	Data transfer for user jobs
Lowest	background	Background	120 Hours	32	Unrestricted Access – no allocation charge

Figure 9: Queue Information

### 6.3 Interactive Logins

When you log in to Predator, you will be running in an interactive shell on a login node. The login nodes provide login access for Predator and support such activities as compiling, editing, and general interactive use by all users. Please note the [AFRL DSRC Use Policy](#). The preferred method to run resource intensive executions is to use an interactive batch session.

### 6.4 Interactive Batch Sessions

To use the interactive batch environment, you must first acquire an interactive batch shell. This is done by executing a `qsub` command with the `-I` option from within the interactive environment. For example,

```
qsub -l ncpus=# -q queue_name -l walltime=HHH:MM:SS -I
```

Your batch shell request will be placed in the desired queue and scheduled for execution. This may take a few minutes because of the system load. Once your shell starts, you can run or debug interactive applications, post-process data, etc. This session will also be shared with other users.

At this point, you can launch parallel applications onto your assigned set of compute nodes by using the `m` command. You can also run interactive commands or scripts on this service node, but you should limit your memory and cpu usage. Use the Cluster Compatibility Mode for executing memory- and process-intensive commands such as `tar` and `gzip/gunzip` and certain serial applications directly on a dedicated compute node.

## 6.5 Batch Request Submission

PBS batch jobs are submitted via the `qsub` command. The format of this command is:

```
qsub [ options ] batch_script_file
```

`qsub` options may be specified on the command line or imbedded in the batch script file by lines beginning with "#PBS".

For a more thorough discussion of PBS Batch Submission, see the [PBS User Guide](#).

## 6.6 Batch Resource Directives

Batch resource directives allow you to specify to PBS how your batch jobs should be run, and what resources your job requires. Although PBS has many directives, you only need to know a few to run most jobs.

The basic syntax of PBS directives is as follows:

```
#PBS option[ [=]value]
```

where some options may require values to be included. For example, to set the number of cores for your job, you might specify the following:

```
#PBS -l ncpus=8
```

The following directives are required for all jobs:

Option	Value	Description
-A	project_ID	Name of the project
-q	queue_name	Name of the queue
-l	ncpus=#	Number of cores
-l	walltime=HH:MM:SS	Maximum wall time

Figure 10: Required Directives

A more complete listing of batch Resource Directives is available in the [PBS User Guide](#).

## 6.7 Sample Scripts

While it is possible to include all PBS directives at the `qsub` command-line, the preferred method is to embed the PBS directives within the batch request script using `#PBS`. The following is a sample batch script:



```

#!/bin/csh

# Declare the project under which this job run will
be charged. (required)
# Users can find eligible projects by typing "show_usage"
on the command line.
#PBS -A project_ID

# Request 1 hour of wallclock time for execution
(required).
#PBS -l walltime=01:00:00

# Request 4 cores (required).
#PBS -l ncpus=4

# Submit job to debug queue (required).
#PBS -q debug

# Declare a jobname.
#PBS -N myjob

# Send standard output (stdout) and error (stderr) to the
same file.
#PBS -j oe

# Change to the $JOBDIR directory.
cd $JOBDIR

# Check MSAS availability. If not available, then wait.
archive stat -s

# Retrieve executable program from the MSAS.
archive get -C $ARCHIVE_HOME/project_name program.exe

# Retrieve input data file from the MSAS.
archive get -C $ARCHIVE_HOME/project_name/input data.in

# Execute a parallel program.
mpiexec_mpt -np 8 ./my_program < data.in > projA-7.out

# Check MSAS availability. If not available, then wait.
archive stat -s

# Create a new subdirectory on the MSAS.
archive mkdir -C $ARCHIVE_HOME/project_name output7

```

Additional examples are available in the [PBS User Guide](#) and in the Sample Code Repository (`$SAMPLES_HOME`) on Predator.

## 6.8 PBS Commands

The following commands provide the basic functionality for using the PBS batch system:

**qsub:** Used to submit jobs for batch processing.

```
qsub [...qsub options...] my_job_script
```

**qstat:** Used to check the status of submitted jobs.

```
qstat PBS_JOBID          #check one job
qstat -u my_user_name    #check all of user's jobs
```

**qdel:** Used to kill queued or running jobs.

```
qdel PBS_JOBID
```

A more complete list of PBS commands is available in the [PBS User Guide](#).

## 6.9 Advance Reservations

An Advance Reservation Service (ARS) is available on Predator for reserving cores for use, starting at a specific date/time, and lasting for a specific number of hours. The ARS is accessible via most modern web browsers at <https://reservation.hpc.mil/>. Authenticated access is required. An ARS User's Guide is available online once you have logged in.

# 7 Software Resources

---

## 7.1 Application Software

A complete listing with installed versions can be found on our [software page](#). The general rule for all COTS software packages is that the two latest versions will be maintained on our systems. For convenience, modules are also available for most COTS software packages.

## 7.2 Useful Utilities

The following utilities are available on Predator:

Utility	Description
<code>archive</code>	Perform basic file-handling operations on the MSAS
<code>mpscp</code>	High-performance remote file copy
<code>qpeek</code>	Display spooled <code>stdout</code> and <code>stderr</code> for an executing batch job.
<code>qview</code>	Display information about batch jobs and queues
<code>show_queues</code>	Report current batch queue status, usage, and limits
<code>show_storage</code>	Display MSAS allocation and usage by subproject
<code>show_usage</code>	Display CPU allocation and usage by subproject
<code>fromdos.sh</code>	Strip DOS end-of-record control characters from a text file

Figure 11: Local Utilities

## 7.3 Sample Code Repository

The Sample Code Repository is a directory that contains examples for COTS batch scripts, building and using serial and parallel programs, data management, and accessing and using serial and parallel math libraries. The `$SAMPLES_HOME` environment variable contains the path to this area, and is automatically defined in your login environment.

# 8 Links to Vendor Documentation

---


## 8.1 SGI Links

SGI Documentation Home: <http://techpubs.sgi.com/>   
[SGI Message Passing Toolkit User's Guide](#) 

## 8.2 Red Hat Links

Red Hat Home: <http://www.redhat.com/> 

## 8.3 GNU, Intel Links

GNU Home: <http://www.gnu.org>   
GNU Compiler: <http://gcc.gnu.org>   
Intel Compiler: <http://software.intel.com/en-us/intel-compilers> 