User Guide

Overview

The ACES II program system has an extensive set of capabilities. All capabilities have been implemented to run on a single processor. Some capabilities can now be executed on multiple processors. These are the parallel direct SCF energy and gradient and parallel direct MBPT(2) energy and gradient and parallel CCSD energy and gradient capabilities. The SCF capability is implemented using standard MPI technology and only requires setting INTEGRALS=GAMESS, DIRECT=ON, and FOCK=AO in the ACES II ZMAT file to run.

This *User Guide* focuses on the parallel CCSD capabilities which use a different design. The CCSD capability is implemented in two components: The data for CCSD computations is organized not as arrays of numbers but as arrays of blocks, each block containing typically 10,000 floating point numbers. The first component in the implementation takes care of the communication and processing of individual blocks and is called the *Super Instruction Processor* or SIP. The second component contains the actual CCSD algorithm written in a high level symbolic language suited to perform tensor contractions called *Super Instruction Assembly Language* or SIAL (pronounced "sail"). The SIP reads the SIAL instructions from a .sial file, or its compiled version from a .sio file, and performs the CCSD energy and gradient calculations. The SIP can run on multiple processors and uses MPI for communicating between those tasks. One task, designated the master, reads the ACES II ZMAT file and reads and writes other files of the ACES II program system, so that the SIP can run as an integrated component of ACES II.

More details about the design of the SIP can be found in the *SIP Design* document.

The SIP is designed to run SIAL programs on a wide variety of hardware platforms, under different versions of UNIX. For the purposes of explaining how to run the program, we assume the following:

- The user is attempting to run a job on a system on which ACES II is supported.
- Some systems have batch queues with job classes specially set up for running software under MPI. The user must be familiar with the job queue requirements of the system, and be able to submit jobs into the necessary queues.
- It is assumed that each processor has a file system containing disk space that is local to its own processor, as well as a common file system that may be accessed by all of the processors.
- When a batch job is submitted, the user has some control over the directory in which the job begins execution. This will be referred to as the run directory.
- The SIP must be run after some preparatory components of the serial ACES II have completed, as it uses data from the ACES II SCF step to perform integral transformations and energy calculations.

When a batch job begins execution, the SIP program may be invoked from a command line in an execution script as

xaces3

The SIP executable expects the following files to be present in the run directory at job startup time:

1. JOBARC This is the JOBARC from a prior ACES II run.
2. JAINDX This is the JAINDX file created by the same ACES II run.
3. MOL. This file is created by the preparatory ACES II run
4. SIAL This file contains the SIAL instructions (.sial or .sio file)

These files may reside in some other directory as explained below.

The program sets up one task as a master task. The master reads basis function data from the MOL file, contraction coefficients, SCF energy, initial gradient values, and eigenvalues from the JOBARC/JAINDX files, and determines whether the calculation is UHF or RHF based on information in the JOBARC file. Program configuration data is read from the ZMAT file.

After all startup data has been read from these files, the data is distributed to each task in the SIP, the program is configured, and execution of the SIAL code begins. SIAL program execution is explained in the *SIAL Programmer Guide*. Processing may continue through several steps, with each step consisting of the execution of a different SIAL program that performs a different type of computation.

When each SIAL program has completed, all predefined output values are written back to the JOBARC file and necessary ACES II LISTS files. Currently, the only such values are TOTENERG, which is the CCSD total energy computed by the SIAL program and GRADIENT, which consists of the 3 gradient component values for each atom. Then the master task collects timing and memory statistics from each task, prints a summary of this information, and shuts down all tasks.

SIP Program Configuration

The SIP program is configured by the parameters in the *SIP record in the ZMAT file read from *stdin*. Each parameter is specified on one line in the *SIP record in the general form

keyword = value

Each line is free form, there are no requirements that certain data be in a certain column. Both the keyword and value are case-sensitive. In general, only the first

instance of a particular keyword is recognized. The COMPANY keyword is an exception and multiple instances are recognized.

A description of each parameter is given in the following table.

Parameter	Data type	Description
MASTER_IS_WORKER	Integer	Must be 0 or 1. If 1, the master task also functions as one of the SIAL workers in a company. Otherwise, the master only functions as the master task. Default value is 1.
MAXMEM	Integer	Amount of memory (RAM) in Megabytes to be allocated by each task for the entire run.
SIP_MX_SEGSIZE	Integer	Maximum segment size used in the SIAL program. Required parameter. This parameter determines the default segment size for all SIAL AO, occupied, and virtual index segments.
SIP_MX_OCC_SEGSIZE	Integer	Maximum segment size to use for all SIAL occupied index segments. If specified, this value overrides the value of SIP_MX_SEGSIZE for occupied index types only in the SIAL program.
SIP_MX_VIRT_SEGSIZE	Integer	Maximum segment size to use for all SIAL virtual index segments. If specified, this value overrides the value of SIP_MX_SEGSIZE for virtual index types only in the SIAL program.
SIP_MX_AO_SEGSIZE	Integer	Maximum segment size to use for all SIAL AO index segments. If specified, this value overrides the value of SIP_MX_SEGSIZE for AO index types only in the SIAL program.
NWORKTHREAD	Integer	Number of simultaneous MPI messages processed by the server thread on each processor. Default = 5.

INTEGRAL_PACKAGE	Character	Possible values are ERD or GAMESS. This parameter determines the type of integrals used in SIAL integral computation. Default is ERD.
MOLFILE	Character	This is the filename of the MOL file. It may be the Unix absolute pathname if desired. Default = 'MOL'.
LOCAL_PATH	Character	This is the name of the local file system described above. There is no default for this parameter.
COMPANY	Character	See description below.
IOCOMPANY	Character	See description below.
TRACE	Character	Type of program trace used in the SIAL program. The only possible values at this time are INSTRUCTIONS, PROCS, or CONTRACTIONS. Used only for debugging. This parameter could produce a very large amount of printout.

The directory where *xaces3* will look for SIAL programs can be configured by setting the environment variable **ACES_EXE_PATH**, for example in the Bourne shell

export ACES_EXE_PATH=/home/my/sial

The program looks for SIAL programs, .sio files, in the following order:

1. First it looks in the current directory where xaces3 is running;
2. Next, if **ACES_EXE_PATH** is set, it looks in **$ACES_EXE_PATH/sio**;
3. Finally it looks in the subdirectory **sio** of the directory where the ACES II binaries are installed. This is the default location.

SIP COMPANY configuration

SIAL programs are executed by each individual worker task of the parallel SIP program. The tasks are organized into *companies*. Each *company* may be further subdivided into *platoons*. Every task in a *company* runs the same SIAL program. All tasks in a *platoon* together hold their own copy of every *distributed* array. In other words, a *company* with 4 *platoons* has 4 identical copies of every *distributed* array declared in the SIAL program. The details of *distributed* arrays are explained in the *SIAL Programmer Guide.*

The configuration of *companies* and *platoons* is specified with the COMPANY parameter. This parameter may be specified multiple times. The COMPANY parameter is a character string containing integer and character subfields as follows:

1. Company ID.

2. Platoon ID.
3. Number of tasks in the platoon.
4. Amount of memory to allocate for SIAL program data on each platoon member (in megabytes); this number is now redundant and the MAXMEM parameter in the *SIP section is used instead.
5. SIAL program file name.

Example:

COMPANY= 1 1 3 500 mp2_n6_rhf.sio
COMPANY = 1 2 3 600 mp2_n6_rhf.sio
COMPANY=2 1 5 800 test.sial

This example specifies 2 companies. The first company, company 1, contains platoons 1 and 2. Each of these platoons contains 3 tasks. Platoon 1 of company 1 uses 500 megabytes of memory in each task, and platoon 2 of company 1 uses 600 megabytes for each of its tasks. Each member of company 1 runs the SIAL program mp2_n6_rhf.sio. The "sio" file extension indicates this is a pre-compiled SIAL program.

Company 2 contains only one platoon, with 5 tasks, each using 800 megabytes of memory. Each of these tasks runs the SIAL program contained in the file test.sial. The "sial" file extension indicates the file contains SIAL source code, which will be compiled on the fly by the SIP master before it is distributed to each task.

The filename for the SIAL file may be a complete pathname. The example given here requires that the SIAL files are present in the run directory at job startup.

SIP IOCOMPANY configuration

The SIAL program may use *served* arrays, which require the presence of a separate *company* of I/O manager tasks. This *company* is defined by the IOCOMPANY parameter, which is similar to the COMPANY parameter. There can only be one *platoon* in the IOCOMPANY. The IOCOMPANY parameter contains the following subfields:

1. Company ID. This must be unique and different from the COMPANY parameter company IDs.
2. Platoon ID. Must be 1.
3. Number of tasks in the I/O company.
4. Amount of memory to allocate for each IOCOMPANY task (in megabytes); this number is now redundant and the MAXMEM parameter in the *SIP section is used instead.

Each I/O company task uses its designated memory to hold blocks of data which are sent to it by the tasks in the SIAL worker companies. When the memory is exhausted, the data overflows to a scratch disk file in the local or global file system, specified by the LOCAL_PATH parameter.

All of the SIAL worker companies have access to the data maintained by the IOCOMPANY tasks.

Example:

IOCOMPANY=3 1 5 550

This defines the IOCOMPANY as company 3, with a single platoon with ID 1. There are 5 tasks in the company, each of which is allocated 550 megabytes of memory.