Wikiprint Book

Title: Parallel I/O with SIONlib

Subject: DEEP - Public/User_Guide/SIONlib

Version: 15

Date: 20.04.2025 04:24:35

Table of Contents

Parallel I/O with SIONlib

Modules

Recent versions of SIONlib are available on the DEEP-EST SDV through the new software stack as modules:

```
$ module load Intel ParaStationMPI SIONlib
```

SIONlib documentation

Documentation for SIONlib can be found on the web at ?https://apps.fz-juelich.de/jsc/sionlib/docu/current/index.html

MSA aware collective I/O

Recent versions of SIONlib contain mechanisms to perform I/O operations collectively, i.e. all processes of a parallel computation partake in these operations. This enables an exchange of I/O data between the processes, allowing a subset of all processes, the *collector* processes, to perform the actual transfer of data to the storage on behalf of other processes. Collector processes should typically be those processes that are placed on parts of the MSA with a high bandwidth connection to the parallel file system. The mechanism has been extended with a new MSA-aware algorithm for the selection of collector processes. The algorithm is portable and relies on platform specific plug-ins to identify processes which run on parts of the system that are well suited for the role of I/O collector. So far, a specific plug-in for the DEEP-EST SDV has been implemented as well as a test plug-in. In the future, further plug-ins for new systems of the MSA type can be added.

In order to use the MSA aware collective I/O operations, a platform specific plug-in has to be selected when installing SIONlib during the configure step.

```
./configure --msa=deep-est-sdv # ... more configure arguments
```

When opening a SIONlib file for access from several MPI processes in parallel, the user has to enable the MSA aware collective I/O mode. This is done using the `file_mode` argument of the open function. `file_mode` contains a string that consists of a comma separated list of keys and key value pairs. The word `collmsa` must appear in that list to select MSA aware collective I/O.

```
sion_paropen_mpi("filename", "...,collmsa,...", ...);
```

Additionally, as is the case when using regular collective I/O, the size of collector groups has to be secified, either through the `file_mode` argument of the `sion_paropen_mpi` function or via the environment variable `SION_COLLSIZE`.

SIONlib CUDA aware interface

In order to match the programming interface offered by other libraries, such as ParaStation? MPI, more closely, functions of SIONlib have been made CUDA-aware. This means that applications are allowed to pass device pointers pointing to on device memory to the various read and write functions of SIONlib without needing to manually copy their contents to the host memory.

Like the MSA aware collective I/O operations, the CUDA aware interface has to be enabled when installing SIONlib. This is done by invoking the `configure` script with the argument `--enable-cuda` which optionally allows to specify the path to a CUDA installation.

```
./configure --enable-cuda=/path/to/cuda/installation # ... more configure arguments
```

When SIONlib has been installed with the CUDA aware interface enabled, the user may pass device pointers as the `data` argument to SIONlib's

- task-local
- key/value
- collective

read and write functions.

```
size_t sion_fwrite(const void *data, size_t size, size_t nitems, int sid);
size_t sion_fread(void *data, size_t size, size_t nitems, int sid);
size_t sion_fwrite_key(const void *data, uint64_t key, size_t size, size_t nitems, int sid);
size_t sion_fread_key(void *data, uint64_t key, size_t size, size_t nitems, int sid);
size_t sion_coll_fwrite(const void *data, size_t size, size_t nitems, int sid);
```

```
size_t sion_coll_fread(void *data, size_t size, size_t nitems, int sid);
```

SIONlib inspects the pointer and if it points to an on-device buffer performs a block-wise copy of the data into host memory before writing to disk or into device memory after reading from disk.