FAQ: MPI Problems

My MPI program fails to compile
If for your C++ MPI compilation you receive error messages like "SEEK_SET is #defined but must not be for the C++ binding of MPI.", sometimes also "Include mpi.h before stdio.h", then please consider reworking the header ordering in your source code. As a workaround, it is also possible to set the macro -DMPICH_IGNORE_CXX_SEEK.

sgi MPT consumes to much memory
see: specific page about SGI's MPI implementation and how to control it.

sgi MPT does not allow static linking
The message passing toolkit from sgi is delivered with shared libraries only. Hence it is not possible to perform static linking (-static or -fast switch) of MPI programs in the mpi.mpt environment.

My MPI program crashes. What do I do?
The symptom will look somewhat like this (sgi MPT):

MPI: MPI_COMM_WORLD rank 0 has terminated without calling MPI_Finalize()
MPI: aborting job
MPI: Received signal x (x may e.g. be 11)

Even if your program appeared to run correctly on another machine/with a different number of CPUs, there still may be bugs in the program. There also may be bugs in the MPI implementation, but that is less probable. To find out where bad things are happening, please perform a traceback procedure as described below.

MPI crash due to incorrect header information (any MPI)
If debugging shows that MPI calls very obviously deliver incorrect results (especially administrative calls), please check whether you've got a file called mpi.h, mpi.f or mpi.mod somewhere in your private include path which interferes with the corresponding files in the system include path. This may lead to errors since different MPI implementations are not binary or even source compatible. Please either remove the spurious files or change your include path so these files are not referenced.

MPI crash due to exceeding internal limits (sgi MPT)
Note that if the crash is initiated by a message similar to

*** MPI has run out of unexpected request entries.
*** The current allocation level is:
*** MPI_REQUEST_MAX = 16384

this is typically due to exceeding an MPT internal limit. In this case, you simply need to set the referenced environment variable (in the above case, MPI_REQUEST_MAX) to a value sufficient to cover your application's needs. Some experimenting may be necessary, also consult the mpi (1) manual page for the functionality and possible side effects of the referenced variable.

Traceback for parallel codes
The following recipe works for SGI's MPI implementation (MPT).

First, build your application as described in the section about obtaining an error traceback (in the serial case), except that you should use mpif90, mpicc etc. Then, perform the following command sequence inside a SLURM batch script or inside a salloc shell:

\$ mpiexec -n 32 ./myparprog.exe

to trace back to the point in the code where the crash happens.

Master-Slave Codes
Some user applications run in master-slave mode. This may be a configuration where e.g. the process with MPI rank 0 does not actually do any computational work but is only responsible for administrative stuff. Here are some hints on how to deal with this situation.

Master consumes CPU Resources
On some MPI variants, the Master will consume CPU resources even though it is actually only waiting e.g., for an incoming message (spinlock). Depending on your code, this may lead to a performance and/or scalability problem if you configure for resource sharing as described in the previous subsection. Here are some suggestions on how to deal with this situation:

- SGI MPT: A setting of e.g., export MPI_NAP=100 will put idling processes to sleep after 100 milliseconds.
- Self-regulatory: Teach the master to do a renice on itself. But note that this is not reversible.
If nothing helps, you will need to return to allotting the master its own CPU. In any case, please check your performance with suitable test scenarios before burning lots of low-quality cycles.

**-fast compiler switch prevents linking on UV systems**

You cannot use the `-fast` compiler switch at the linking stage with SGI MPT. See the [section below](#) for an explanation.