Tony's starting questions

Initial thoughts/questions (and some answers) about PIO from Tony

NOTE: comments added since the 1/4/07 telecon are <i>Additional notes from 1/10/07 telecon (with Rob Latham) are red.
want memory and performance scaling I/O for cdf and binary
MPI-IO is the only safe way to read/write to/from a file from multiple pes. in addition, it has been optimized to run fairly well for parallel i/o (collective vs independent)
MPI-IO is part of the MPI2 standard and should be available everywhere
pnetcdf is a library written on top of MPI-IO to create netcdf files. it does NOT use the netcdf library and is completely independent from netcdf except it generates netcdf files.
pnetcdf is reportedly relatively robust. it may not be available everywhere.
is there a risk that pnetcdf and netcdf will diverge?
pnetcdf has done a good job tracking netcdf 3.x series. Tracking of 4.x series will depend on how much demand there is for its features. We expect 4.x will support reading 3.x
is there a risk about long term pnetcdf support?
Pnetcdf has 5 years of funding currently. Developers plan to support it indefinitely.
will ccsm include pnetcdf with ccsm? how hard is it to install/optimize pnetcdf on a platform? we need to handle situations where pnetcdf if not available. this could be done by reverting to standard serial netcdf reading/writing from the root or by implementing some slow, memory scalable, synchronous parallel i/o.
we have some flexibility here, probably we'd use the root rearrange capability in pio and serial netcdf calls which are triggered by a compile or run-time flag.
CCSM doesn't include netcdf, so won't include pnetcdf
can we mix netcdf and pnetcdf, say using netcdf in define mode and pnetcdf in data read/write mode? that would reduce the required code changes in ccsm and also allow the pio library to be alot smaller since all the define mode stuff would NOT have to be supported through pio.

not clear, but we probably don't want to do this. pio should support full set of i/o requirements through the interface and mixing pio and serial netcdf at user code level is discouraged for a single file. we may choose to mix netcdf and pnetcdf in pio, but that's a
separate implementation issue.

 should be possible to do this (from Pnetcdf) but we don't want to
we may want to do some rearranging for parallel i/o. say we're running on 1000 pes and want to write from 128 pes. two distinct issues need to be considered
- writing from a subset of pes
- writing memory continguous data which is generally required in the high level pnetcdf calls or serial netcdf.
this can be done with mct using gsmaps and rearrangers. or, we could have multiple write calls for each subset of contiguous data. alternatively, pnetcdf provides the ability to write non-contiguous data via MPI datatypes. i think i'm missing something here?
it looks like we want to have a pio level rearranger to address both bullets above. we expect the cost of rearrangement to be small compared to I/O. if that's the case, we are going to use rearrangement to simplify the decomp which should then simplify the calls to the i/o layer (pnetcdf). the initial strategy will be to rearrange data to a 1d decomp at the pio level and minimize reliance in the mpi-io layer for "aggregation" except to handle some i/o subsystem optimization. overall, this strategy will cost us in rearranging, but it should result in a more robust system and less risk because the usage of pnetcdf and MPI-IO should be quite straight-forward plus we'll have full control of the rearrange strategy and code.
some of pnetcdf optimization is based on MPI-IO hints. how hard are these to use?
not clear, but we plan to avoid this as much as possible.
MPI-IO hints might be ignored but will never hurt. They are a win when they do work. So we'll do both in pio and activate them in the build system
how hard are the intrinsic MPI/MPI-IO decomp info (start, count, stride, imap) to work with? are there other MPI tools that can help define decomps (MPI_CART? MPI_TYPE?) that we might use?
again, keep things pretty simple. we'll use the pnetcdf interfaces (start, count, stride) to write i/o. separately, we need to have a datatype that provides general decomp info. this could be a global seg map or something else. that's an implementation issue in pio.
does pnetcdf support writing from a subset of pes?
yes. but it's unclear yet whether all pes call pnetcdf or
just the subset of pes that are writing.
should we look at or use wrf's implementation of pnetcdf or i/o in general?
No. Not general enough for us
what kinds of issues do we have with pop chunked or clm "random" type decompositions wrt memory usage, decomp description, cost of rearranging, calls to 'read'/'write' due to non-contiguous memory, etc?
our "always rearrange" strategy takes care of most of the complexity here. we do need to address missing grid points, but that can be handled under the covers in pio. also, we need to look at memory usage, especially in mct if that's the tool we're going to use. we have a couple outstanding bugs that have been identified in mct, and
more discussion will take place off-line.
are there concerns about memory usage in MPI-IO or mct. for instance, running global 3600x2400 on 5000 pes with 2d decomp or even chunked decomp. gsmap memory usage could quite large. what else in mct will also have a large per pe memory footprint (rearrangers?, etc). can some of this mct memory be deallocated after initialization (ie gsmaps)?
discussion off-line to follow up on this issue. we've identified this as an important point.
MCT 2.3 release lowers memory usage in Router and Rearranger init
how difficult is pio going to be to maintain, port, optimize? what has to be supported? (real8, real4, integer?) (1d, 2d, 3d, 4d data?) (xy vs yz slabs?) (1d, 2d, random, chunked decomps?) (unlimited dimension?)
the maintenance question is a little unclear at this point. we've decided to support 1d, 2d, 3d, and other (hoome) grids, but all the data will be sent via a 1d array. in general then, we need to come up with a way to describe grids so pio can write the grid description in pnetcdf. at the same time, the data interface and rearrangement should be pretty straight forward since it's all 1d. we have decided, at least initially, that comonents take on the burden of reshaping data to the 1d format prior to calling the pio library. we will support random decomps via the "always rearrange" strategy and john feels pio can be easily extended to handle different slabbing (xy, yz, xz, etc) strategies. we will support writing only real8, real4, and integer data through the interface (probably overloaded). components will have to addresss issues related to other non-standard "kinds" used in their models.
can we mix parallel data writing with "analyzed" data writing from a single pe? if we want to write 3d u and v fields as well as analyzed meridional overturning. u and v are fully memory parallel. the meridional overturning analysis has been done such that the result is on a single pe. how will both fields be written to the same file through pio?
If output is binary, all output will be binary. If output is netcdf, use a separate file/communicator to write out the 1-proc fields.