Summary for allhands

Morten Sommervoll

June 1, 2005

1 QMP testing on QCDOC

QMP_declare_send_to: non-nearest neighbor communication is not allowed with this library

QMP_declare_send_to: sending to itself results in QMP printing:
  declareSend to self not allowed
but it does not return a null pointer to indicate error.
QMP_declare_receive_from: also does not return a null pointer when receiving from itself.

QMP_declare_multiple:
  com_arr[0] = QMP_declare_multiple (sendh, ndims);
  com_arr[1] = QMP_declare_multiple (recvh, ndims);
  com_all = QMP_declare_multiple (com_arr,2);
prints:
QMP_declare_multiple:: individual msg handle should be single (with or without stride) then hangs. So the issue here is trying to use msg handles containing multiple msg’s in QMP_declare_multiple. Works on other implementations.

On the call QMP_get_error_number(comm_all) where comm_all is a multiple msg handle, it prints: MultiOperation::getErrorNumber, does this mean that it does not have an total error for a multiple message handle?

QMP_layout_grid:
When using on a machine with 6 dimensions, it declares an error and prints:
  User is declaring for smaller dimension than QCDOC partition: Machine_dimensi on[4](2)>1
Works fine when running qpartition_remap first to make the machine <= 4 dims. My question here is should/can this function handle remapping?

Spec vs QCDOC implementation with const modifier
spec: QMP_ictype_t QMP_get_msg_passing_type (void);
QCDOC: const QMP_ictype_t QMP_get_msg_passing_type (void);

spec: int QMP_get_allocated_number_of_dimensions (void);
QCDOC: const int QMP_get_allocated_number_of_dimensions (void);

2 Q* packages overhead

The numbers used here are a single run for each datapoint, not an average over multiple runs, thus datapoints might be skewed from unknown interference. Here I compiled MILC, adding the SciDAC packages incrementally to investigate the overhead of adding the next package. The notation used in the graphs is for instance QMP/noQ means (runtime W QMP - runtime no SciDAC libs)/(runtime no SciDAC...
libs), the others are similar. The QIO numbers were only included here for completeness. Originally I checked them to have a complete picture when comparing QDP vs QMP.

The file all.xsl holds an excel spreadsheet with the percentage overhead for the different packages. The graph’s below shows a simple picture for each machine.

For the graphs I vary either ns or nt, giving the lattice size $ns^3 \times nt$. On a side note for why I have both graphs, keeping a fixed lattice size, varying ns and nt does not give the same runtime. Below are two graphs for each machine, keeping one parameter fixed at a time. Note that there is only 3 measured points for fixed nt, since smaller ns makes the runs very short, I judged them to be untrustworthy for comparisons.

In general QMP adds a bit of overhead. QDP decreases the overhead compared to QMP overhead. It also decrease the overhead compared to no Q* packages at times.

Dante P4, 6GB ram, Infiniband.
Dante 16 procs
Ram is a 256 processor SGI Altix, 1.5 GHz, 8 GB ram.
Ram 16 procs
Tungsten is a Xeon Linux Cluster. Dual Intel Xeon 3.2 GHz, 3GB ram, ethernet and myrinet interconnect.

Using the single proc per node numbers on tungsten. The 2 procs per node run was not much slower than the single proc per node, but I haven’t got a full set of numbers for that. Surprisingly many of the runs with two processes per node failed in communication, with a checksum error.

Tungsten 16 nodes 1 proc per node
Tungsten 32 nodes 1 proc per node
Dru was very helpful answering my questions so that I could run MILC on qdoc, was a few gotchas there. Unfortunately I haven’t got any numbers from qdoc at this point. the qdp executable segfaults now when running.