AASPI SOFTWARE PARALLELIZATION

Introduction

Generation of multitrace and multispectral seismic attributes can be computationally intensive. For example, each input seismic trace may generate 50 or more complex spectral components. Computing long wavelength curvature at a given voxel may include a window of 21 traces by 21 trace by 81 vertical samples. Our assumption is that our sponsors have access to commercial software that provides fast, but perhaps less accurate algorithms. For example, one of the larger but less expensive commercial packages computes coherence along time slices, rather than along structural dip. Another, high end more expensive system allows such computation along structure, but not as the default. In our implementation, we assume the user has access to multiple processors, whether it be on his or her desktop, on a cluster down the hall, or on a remote supercomputer. For this reason, almost all of our algorithms have been parallelized. Exceptions include high input/output to computation ratio programs such as data slicing and plotting.

The AASPI software is built on the Message Passing Interface (OpenMPI). The executables do not require any cost except installation of the proper run time libraries, which we include in the distribution. If you wish to recompile, OpenMPI is free on most Linux clusters. Windows has discontinued support of OpenMPI; in this case we advise purchasing the Intel f90 compiler which includes INTELMPI

Parallelization Flow Chart

We use three different parallelization structures – those for multitrace attributes computed using a computational stencil, single trace attributes such as spectral decomposition, prestack migration/demigration.

Parallelization of multitrace attributes

All of the volumetric attributes (e.g. dip3d, filter_single_attribute, similarity3d, sof3d, curvature3d, glcm3d, disorder, and all of the poststack image processing algorithms are computed on a rectilinear grid, requiring padding of the data when converting from SEGY to AASPI format. For these algorithms, each live seismic sample requires the same amount of work. In the figure below, the seismic data are sent by the master to \( n \)
Overview: AASPI Software Parallelization

slaves. Upon completion, the slaves send the data back to the master, which writes the results out to disk. The processors are synchronized after the completion of each seismic line.

Parallelization of single trace and single gather attributes

You will note that if there are large "no permit" zones in your padded data volume, some of the processors will remain idle. For this reason, we use a different strategy when computing single trace attributes, such as spectral decomposition programs spec_cmp, spec_cwt, spec_clssa, and spectral_probe. Note that matching pursuit spec_cmp and mp_nmo are a trace- or gather-independent nonlinear process whose computation
time depends on the seismic waveforms. For trace- and gather-parallelized algorithms, the parallelization is synchronized after a fixed number of traces per block (default = 10,000 traces).
Overview: AASPI Software Parallelization

Parallelization of migration and demigration algorithms

Currently (Fall, 2015) we parallelize over output lines and input blocks of seismic traces. Unfortunately, this process becomes inefficient when the seismic lines are so long that a single line of CRP gathers no longer fits in memory. For this reason, Yuji Kim is reworking the parallelization strategy to be limited by output blocks of CRPs as well as output lines. The flow chart will follow shortly.

The Parallelization Tab common to AASPI GUIs

Parallel computation on Linux and Windows standalone machines and clusters

All AASPI GUIs use the same parallelization tab. As an example, we display the one shown in the aaspi_dip3d GUI:

The (1) number of processors per node is read from the aaspi_default_parameters file, ideally set up to read reasonable values on your system (see that part of the documentation). If you do not know the number of processors on your local host, simply (2) click Determine Maximum Processors on localhost. I am running on a machine called tripolite. (The default local host name is localhost, which is also a valid entry on line 3. When I click this button I find that there are 24 processors.

1
2
3
Since I am running on a shared system, I don’t wish to use all 24 processors, or my colleagues will be angry with me. Instead, below, I (4) submit the job over three computers (or nodes). Specifically, I will use 8 processors on jade and 16 processors on hematite, and 12 processors on kwiatkowski. They syntax reads “jade:8 hematite:16 kwiatkowski:12” where the number of processors per node is separated by the node name with a colon, and the node names are separated by spaces.

When the job is submitted, it will run over 8+16+12=36 processors, and (if the network is sufficiently fast) complete approximately 36 times faster than on a single node system. Such parallelization is easy to set up on Linux. For Windows, it will require a local area network and a reasonably fast communication backbone. In both cases, your
Overview: AASPI Software Parallelization

Information Technology staff will need to initiate passwordless access for your user login ID to the target machine IP addresses or machine names (a fairly simple process). Default for both machines is to require a password each time someone accesses machine X from machine Y to avoid unwarranted access.

**Parallel computation on Linux Supercomputers running LSF**

Several of our larger sponsors have access to supercomputers purchased for reservoir simulation and prestack migration. Seismic attributes computed on very large 3D marine surveys can also take considerable time. As of November, 2015, sponsors have run our software using upwards of 1,500 processors. Supercomputers are not generally used for interactive jobs, but rather large, “batch” jobs. There are currently two popular batch job submission systems, LSF, PBS, and SLURM.

The Load Sharing Facility (LSF) is the batch system we use here on the OU supercomputer, called “OSCer”. In our case, AASPI has purchased a subset of dedicated “condominium” nodes, that forms part of the supercomputer system. When we submit a job, we go to the head of the line and other users page out. However, when we are not using the compute cycles, others in our system are free to use them (cycles cannot be warehoused!). Since the computer is about one mile from our building and is housed in the weather center, most of these computer cycles are used in severe weather modeling. I click the tab (5) *Build an LSF Script* and obtain the following GUI:

![Screen capture of LSF GUI](image)

Note that (5) the tab now reads Build an LSF Script while the previous options (4) are now disabled (grayed out). In my AASPI_default_parameters file I had previously set
the (10) maximum number of hours I will run, (6) the name of the batch queue, and (8) the maximum number of processors on that queue. Yes, the name of the queue was called kurt_q so that my students can say “kurt has crashed” or “kurt can’t get up” or “kurt is down again” and still sound professional. While this queue has only 144 processors, others have 1000. Let’s assume I request all 1000 processors. Examining the parallelization strategy for geometric attributes shown in the first flow chart above, you may notice that if I have only 550 cdps per line in a 3D survey, 450 of the processors would be reserved from other use, but remain idle. For this reason, the GUI has (8) the Determine Optimum Number of Batch Processors button, which shows (for the 144 processor example and the GSB_small survey where each seismic line has 1000 cdps):

Note that largest number of processors that can be used is 143. If I wish to allow others to use the system, another optimum value is 112 processors, running longer, but freeing up 144-112=32 processors for other folks to use. Chosing this value I obtain the following image:
Note that (11) the number 112 now appears after the *Number of LSF processors to use.*

When I click *Execute dip3d* on the main tab, the GUI generates a file called *dip3d_GSM_small.bsub* and then submits it. Let’s look at this file by typing *cat dip3d_GSM_small.bsub:*
This is the required batch job deck on our supercomputer system. We have not yet figured out how to string together multiple batch jobs to generate a workflow.

**Parallel computation on Linux Supercomputers running PBS**

The Portable Batch System (PBS) is used by at least three of our sponsors on their supercomputer system. Unlike LSF, we do not have this software here at OU. Also, unlike LSF, there will be one or two system specific components of the submission deck that needs to be tweaked by your IT folks. This will involve a one-time modification of a file used in building the job deck. The input parameters are identical to those used by LSF.
Parallel computation on Linux Supercomputers running SLURM

In 2016, OU upgraded our supercomputer, OSCeR, and supplanted LSF with SLURM, or Simple Linux Utility for Resource Management. Selecting the SLURM button generates a file call dip3d_GSB_small.sbatch which looks like this:

```bash
#!/bin/bash
#PBS -l select=33
#PBS -d dip3d_GSB_small.qsub
#PBS -q slurm1
#PBS -j oe
hello world!

mpirun -np $(nproc) --machinefile $(PBS_NODEFILE) dip3d_GSB_small.qsub

NPL_COMPILER= intel
export NPL_COMPILER
NPL_INTERCONNECT=ib
export NPL_INTERCONNECT
NPLVENOR=spirun
export NPLVENOR
cd /output5/mart2925/projects/GSB_small

mpiexec -n $(nproc) --machinefile $(PBS_NODEFILE) dip3d_GSB_small.qsub
```

The resulting dip3d_GSB_small.qsub file looks like this:
You may encounter installation-specific modifications of these files. Let us know and with your guidance, make appropriate modifications.