RFC: Modifications to remove the nagg tool’s limit on the maximum number of input granules

Larry Knox

The NPP aggregating and packaging tool prototype included a fixed limit on the number of input granules so that information for all input granules could safely be stored in memory. This document summarizes proposed modifications to remove the limit on the number of input granules when aggregating and packaging NPP data products with nagg. The risk of excessive memory usage will be eliminated by sequentially reading, processing and writing subsets of the input granules to output files.

# Introduction

# Nagg (<http://www.hdfgroup.org/projects/npoess/nagg_index.html>) is a tool for aggregating NPP data granules from existing files into new files with a different number of granules or different combinations of compatible products than in the original files.

The procedure used by nagg version 1.6.1 and earlier to aggregate and package granules in a list of input files is described below:

1. Read all input granules, storing information including granule attributes and the input file name needed for copying the granule to an output file containing the specified aggregation.
2. Sort the granules by granule ID followed by product ID to provide time ordering. This facilitates packaging aggregations of related products into output files with matching data time periods.
3. Calculate the ending time of the aggregation bucket defined by DDS Aggregation Methodology1 in which the first granule belongs. Write all granules belonging to this bucket to an output file (packaged) or set of output files (unpackaged).
4. Repeat step 3 until all granules have been copied to output files according to the nagg command parameters.

# In order to avoid exhausting the machine memory resources, nagg currently has a limit of 10,000 granules in the input files. Nagg will terminate without producing any aggregations if it is given a set of input files containing more than 10,000 granules.

If nagg is modified to sort input files by the Data Start Date and Data Start Time fields of the input file names, a subset of the granules can be read from files in the input file list that match the Data Start fields of the first file. When these granules have been processed, granules from files in the input file list matching the Data Start fields of the next file in the list can be read and processed. This process can be repeated until all granules are aggregated.

Calculation of buckets will not be affected by the change. The only change for writing granules is that nagg will need to determine when creating output files whether a bucket includes the last input granule. The number of granules in each aggregation is determined by the command parameter for the –n, --number option, except for the first aggregation, which may contain a partial aggregation if the first granule is not at the beginning of a bucket, and the last aggregation, which may contain a partial aggregation if its last granule is not at the end of a bucket. For all other aggregations, fill granules will be added until the aggregation is complete.

There is one scenario in which sorting the input files by their Data Start fields may not be effective: when a file has been renamed with a name that does not conform to the NPP/NPOESS Data Products File-Naming Convention. In this case it is possible to sort the input files by reading the Granule IDs in either the user block or an aggregate dataset attribute of the input files. In order to minimize the overhead for input file sets with compliant names, this could be added as an additional command option, e.g. --noncompliant-names.

# Proposed nagg modifications

There will be two main areas of modification. Nagg will need to sort files in order of granule time before reading granules. It currently reads granules from all files into an array of granule information structures, then sorts the granules for writing. Instead, files with granules from a specific time interval should be opened to read only those granules needed for the next aggregation to be written. Once written the granule information can be discarded.

## Sort input files prior to reading granules

With a table of input file names, start times, and end times, nagg can sort the input files by the starting time of the aggregation in the file. The granules in the files can then be read in smaller groups with matching times and processed group by group.

* 1. By Data Start fields in the input file names.

The Data Start and Data Stop Time fields will be extracted for sorting the input file list and matching the granules contained.

* 1. (Option for files with non-compliant names) by BeginningGranuleID.

BeginningGranuleID and EndingGranuleID can be found in either the user block or an attribute of the Aggregation dataset for each product in the file. Reading from the user block may save a couple of seconds for getting the BeginningGranuleIDs from 1850 input files. (We probably need a table of BeginningGranuleIDs, EndingGranuleIDs, and filenames.)

## Read and process granules grouped by time

1. Open files with Data Start times matching the first file in the sorted input list, and determine the total number of granules contained in those files. If the number of granules does not exceed 10,000 read them into the granule information table and sort it to eliminate duplicates. If the number exceeds 10,000, determine a number of granules per file in the group that can be read without exceeding a total of 10,000 in the granule table, read that many granules from each file, and sort the granule table to eliminate duplicates.
2. (First group of granules only) If any product to be aggregated does not have at least one “real” granule in this beginning collection, nagg will need to search the remaining files for a “real” granule of that product to be used as a granule pattern to create missing granules for the product. If no “real” granule is found, nagg will terminate with an error message, and the user must either provide a file with the granule or remove the product from the nagg command parameters.
3. Write granules to output files, which are created and closed according to the nagg command parameters and the number of granules written.
4. Repeat reading in step 1 and writing in step 3 as necessary to read all granules in files opened; repeat steps 1 and 3 for remaining files in the input file list until all granules are processed.

# Revision History

|  |  |
| --- | --- |
| *October 30, 2014:*  *November 12, 2014:*  *January 27, 2015:* | Version 1 circulated for comment within The HDF Group.  Version 2 circulated for comment within The HDF Group.  Version 3 circulated for comment to JPSS and nagg users. |