



Faster Results By Multi-threading Data Steps

Paper 109-2010 - SAS Global Forum 2010

Ian J. Ghent
Technical Consultant
SAS Canada - Ottawa

Overview of scalability

- There are two ways to make a process scale out to tackle complex problems quickly and efficiently
 - Vertical scaling, adding more processors, I/O channels, and memory to the server/workstation
 - Horizontal scaling (grid), uses lots of smaller servers/workstations in parallel to complete a task

How SAS handles scaling

- SAS has developed solutions to deal with both vertical and horizontal scaling
 - Thread enabled procedures, take advantage of multiple processors and I/O channels (using SPDE/SPDS) to complete tasks quicker on one machine
 - SAS Grid computing provides a solution to enable complex tasks to be divided and shared amongst a whole group of machines, with sophisticated management capabilities
 - Automated code analysis for identifying independent tasks to divide into separate machines

Vertical scalings' missing piece

- Many SAS procedures happily make use of multiple processors and I/O channels to improve performance, but what about data steps
- Because of the flexible nature of a data steps, it is very difficult to come up with a generalized way to share the load amongst all the processors in a machine for a single data step
- But, by carefully considering how to divide up your data, you can get many data steps to scale to numerous processors

A Multi-threaded Data step

- Take advantage of multiple cpu's in datasteps by dividing (partitioning) the dataset into smaller chunks that can be procesesed separately
- The type data steps being converted dictates the technique in which the dataset can be partitioned
- Inter-row functions/statements/operators, such as "retain", "lag" and "first."/"last." can sometimes be overcome by careful partitioning
- Example macros are in the paper to show the various concepts and how they can be applied

Who should be interested

- Have a long running or repetitive intensive data steps
- Access to a multi-processor server / workstation
- Willing to take some time to develop and test your parallel datastep
- A data step that has a high level of CPU utilization

OR

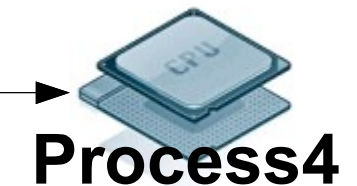
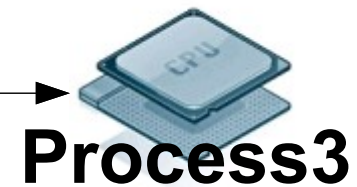
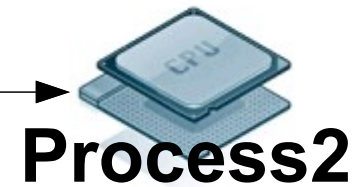
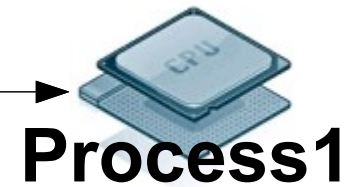
- An SPDE/SPDS setup with multiple I/O channels

Some words on partitioning

- This paper discusses three types of partitioned datasets
 - Horizontal partitioning
 - Thread 1 processes the first n rows in the dataset
 - Simple, and can solve some LAG function problems
 - Interleaved partitioning
 - Thread 1 processes rows 1,5,9, etc.
 - Ensures evenness of thread processing
 - Indexed (clustered) partitioning
 - Thread 1 processes all rows with variable values a,b,c
 - Can overcome I/O constraints using SPDS/SPDE, and some retain/“first.”/“last.” statement problems

Partitioning Compared Horizontal Partitioning

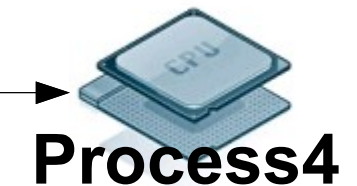
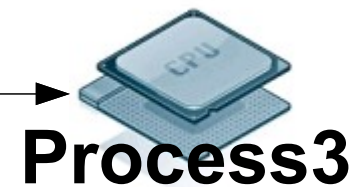
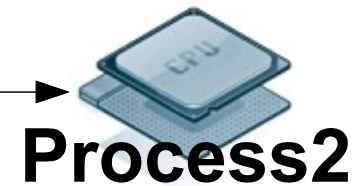
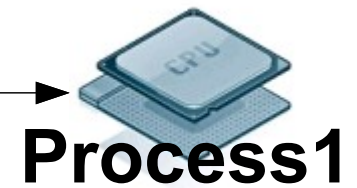
| <u>Obs#</u> | <u>Cluster Variable</u> | <u>Var1</u> |
|-------------|-------------------------|-------------|
| 1 | CAT | 0.245 |
| 2 | CAT | 2.563 |
| 3 | MOUSE | 5.276 |
| 4 | DOG | 7.524 |
| 5 | SQUIREL | 1.853 |
| 6 | CAT | 6.216 |
| 7 | CAT | . |
| 8 | MOUSE | . |
| | | |



Partitioning Compared

Interleaved partitioning

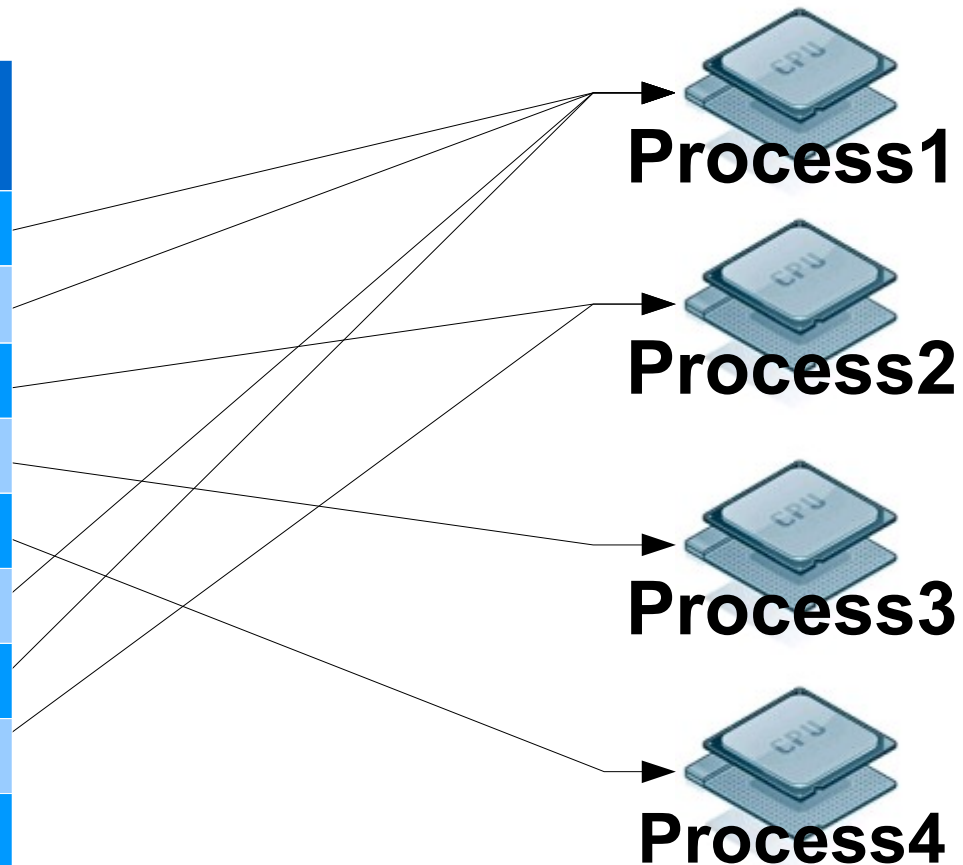
| <u>Obs#</u> | <u>Cluster Variable</u> | <u>Var1</u> |
|-------------|-------------------------|-------------|
| 1 | CAT | 0.245 |
| 2 | CAT | 2.563 |
| 3 | MOUSE | 5.276 |
| 4 | DOG | 7.524 |
| 5 | SQUIREL | 1.853 |
| 6 | CAT | 6.216 |
| 7 | CAT | . |
| 8 | MOUSE | . |
| | | |



Partitioning Compared

Indexed partitioning

| <u>Obs#</u> | <u>Cluster Variable</u> | <u>Var1</u> |
|-------------|-------------------------|-------------|
| 1 | CAT | 0.245 |
| 2 | CAT | 2.563 |
| 3 | MOUSE | 5.276 |
| 4 | DOG | 7.524 |
| 5 | SQUIREL | 1.853 |
| 6 | CAT | 6.216 |
| 7 | CAT | . |
| 8 | MOUSE | . |
| | | |



Example macros

- The example macros provided in the paper provide a basis to develop your own parallel data steps
- There are two files, a master thread file, and a mignon thread file
- The master thread code initiates multiple sas sessions of the mignon thread code and waits for them to complete
- Once completed, the master thread joins the data together to form a final output file

Master Thread

- Can be stand alone or part of other code
- Produces informational datasets so each mignon thread will know what to do when they start
 - Dataset contains: Total number of threads, Partitioning type, row range, where clauses, etc.
 - Renaming the datasets at each state is helpful for run-time tracking of mignon threads
- Sets up the mignon threads' environments and executes thier code via a “systask command”
- Waits for mignon threads to complete and joins the datasets
 - Systask command's "waitfor _all_" is awesome!

Mignon Thread

- Code should be in a separate file from the Master code to avoid recursive issues
- Reads a process id from the sysparm macro variable passed via command line parameter
- Has filtering logic, so only the records that are intended to be read are
- Should contain any variable processing statements normally in a datastep body
- Changes the state of the informational dataset to reflect that it's running and done

Extending the idea

- The example code is a simple data/set data step, but the idea can be extended to merge data steps as well, by mignon threads applying row filtering to only one dataset
- Any row-independent data and code blocks could be included in your mignon thread code, not just a single data step
- Approach can be easily adapted to SAS/CONNECT (MP Connect, piping) and to SAS Grid computing

Final Remarks

- There is a non-trivial effort involved in multi-threading many types of data steps, so it is only recommended with long running data steps
- CPU bottlenecks are much easier to solve than I/O bottlenecks (check the log to see the ratio of real to CPU time for your data step)
- Always test with a sample to ensure you get the exact same result as your original data step
- In-line partitioning can only be done when the data is stored with a random access supported engine (no flat files)

Additional Information

- Link to the paper (SGF 2010 proceedings):

<http://support.sas.com/resources/papers/proceedings10/109-2010.pdf>

Contact information

Ian J. Ghent
Technical Consultant
SAS Canada
1600 – 360 Albert St.
Ottawa, ON
ian.ghent@sas.com
613.755.2328

Questions?