

Barcelona 2016

Fast Track to Approval: Speed and Efficiency

9th-12th October Princesa Sofia Gran Hotel, Barcelona

Reducing Build Time of Integrated Clinical Databases with SAS[®] Grid and SAS/OR[®]

Eric C. Brinsfield

Meridian Analytics, Virginia Beach, VA USA

d-Wise Technologies, Morrisville, NC USA

Meridian Analytics

d-wise

Agenda

- Introduction
 - Problem Definition
 - Specific Use Case
 - Solution Value and Applicability
- Approach
- Methods
 - Analyzing SAS Logs and Performance Measures
 - Optimal Scheduling with CPM Procedure
 - Visualizing Results with Gantt Procedure
 - Submitting Programs to the SAS Grid
- Conclusions

Problem Definition

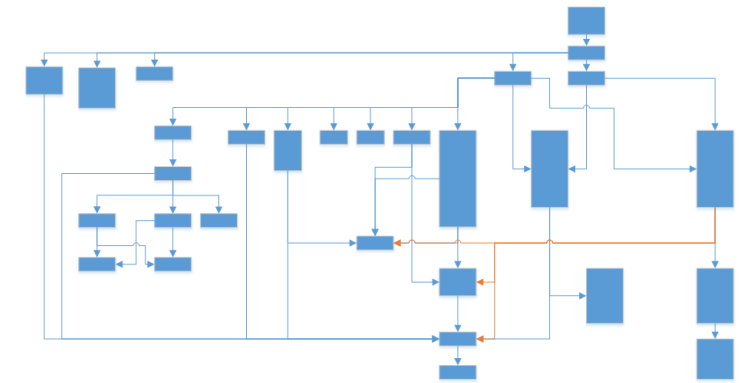
You must run a collection of programs that take several hours when running sequentially. You are sure that running some programs in parallel (asynchronously) would save overall execution time.

- How do I know if parallel processing is worth the effort to setup?
- How much time could I save?
- How do I determine which programs depend on other programs?
- How many parallel grid sessions do I need?
- How do I assign programs to separate nodes in an optimal allocation?
- How to I implement this with SAS Grid?

Specific Use Case: Building Integrated Databases

- Integrated SDTM or ADaM databases
 - Can require significant rearrangement of data when combining
 - Can require many programs some of which:
 - Run for a long time (for example labs)
 - Depend on data sets created by other programs (for example ADSL)
 - May need to be run often when under development or for other reasons
 - May be submitted:
 - As individual batch jobs
 - As %INCLUDES from a single SAS program
 - From Display Manager, Enterprise Guide, or SAS Studio

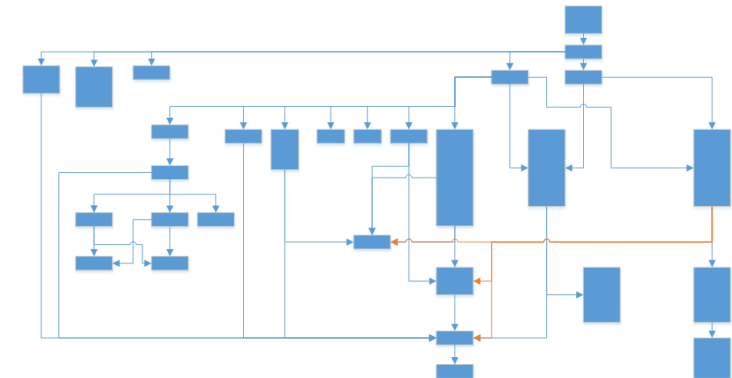
Sample Program Dependency Network Diagram



Value and Applicability

- A solution is valuable to any situation where:
 - Many programs take too much time to execute when run sequentially
 - All programs must be complete by a specific time (i.e. nightly jobs)
- “Too much time” is a relative measure and will depend on:
 - Frequency of execution
 - Total execution time
 - Deadlines and priority
 - Maturity of the programs
 - Effort required to run them in parallel

Sample Program Dependency Network Diagram



Approach

- Analyze SAS logs to collect:
 - Execution times, CPU times, and other statistics of interest
 - Input data sets required and output data sets created by each program
- Analyze program dependencies
 - Establish which programs depend on other programs based on the data sets that they create.
- Predict the shortest job execution time if running programs in parallel
- Evaluate return on investment of parallelization, then
 - Optimize the assignment of programs to SAS Grid sessions
 - Create new driver or auto-generate new driver program

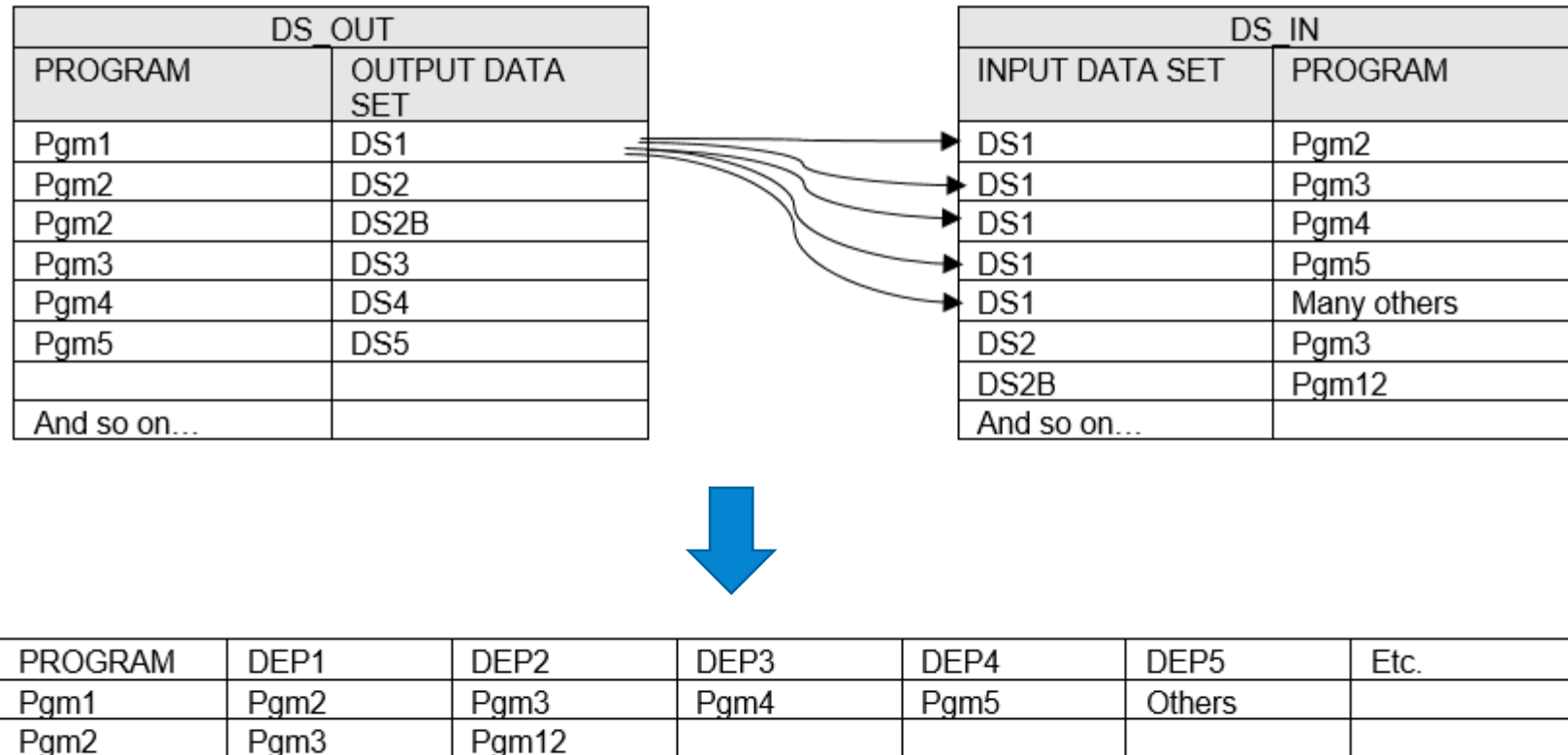
Analyzing SAS Log

- To parse the SAS logs, we used the macro provided by SAS called LOGPARSE, which can be downloaded from SAS at the following sample note:

Sample 34301: Parse SAS® Logs to Extract Performance and Timing Information.

- The program is documented.
- Post-processing is required, but not discussed today.
- You need:
 - Total execution time (elapsed time) of each program
 - List of input data sets and output data sets for each program
 - Subset to permanent library of interest

Determine Dependencies



Predict Shortest Time and Best Schedule

- Use SAS/OR's CPM procedure for Critical Path Method
 - Other procedures can also solve the problem, but CPM fits.
- Used to schedule resources, where:
 - Resources in this case are grid nodes or sessions
 - Tasks to complete are the programs or their execution
 - Costs are the execution times
- Determines:
 - Shortest possible job execution time based on:
 - Dependencies, grid nodes available, and program execution times
 - Feasible assignment of programs to grid nodes for shortest time
 - Number of nodes sufficient

Predict Shortest Time and Best Schedule

```
proc cpm  data=pgmsuccessors out=firstrun  
          date='00:00:00't interval=second  
          resin=nodes_avail  
          rsched=node_schedule resout=nodes_used;  
activity pgmname;  
duration dur_seconds;  
id seq_order tot_realtime;  
successor dep1-dep&dim;  
resource grid1-grid&nodes_available /  
          obstype=obstype period=period resid=gridname  
          esprofile rcprofile routinterval=minute  t_float;
```

ResourceIn Data Set: Nodes_Avail

Defines how resources (grid sessions) are used, evaluated, and assigned.

OBSTYPE	GRIDNAME	PERIOD	GRID1	GRID2	GRID3	GRID4	GRID5	GRID6
restype			1	1	1	1	1	1
resrcdur			0	0	0	0	0	0
altrate	grid1		1	1	1	1	1	1
altprty	grid1		1	2	3	4	5	6
reslevel		0:00:00	1	1	1	1	1	1

- restype = 1 resources (grid nodes) are replenishable (reusable).
- resrcdur = 0 indicates that the resource uses a fixed duration.
- altrate = 1 indicates that if grid1 is not available, the Grid Manager select only 1 alternate
- altprty = the priority of alternate selection.
- reslevel = the number of resources available per grid node starting at time 0:00:00.

Out= Data Set

Contains:

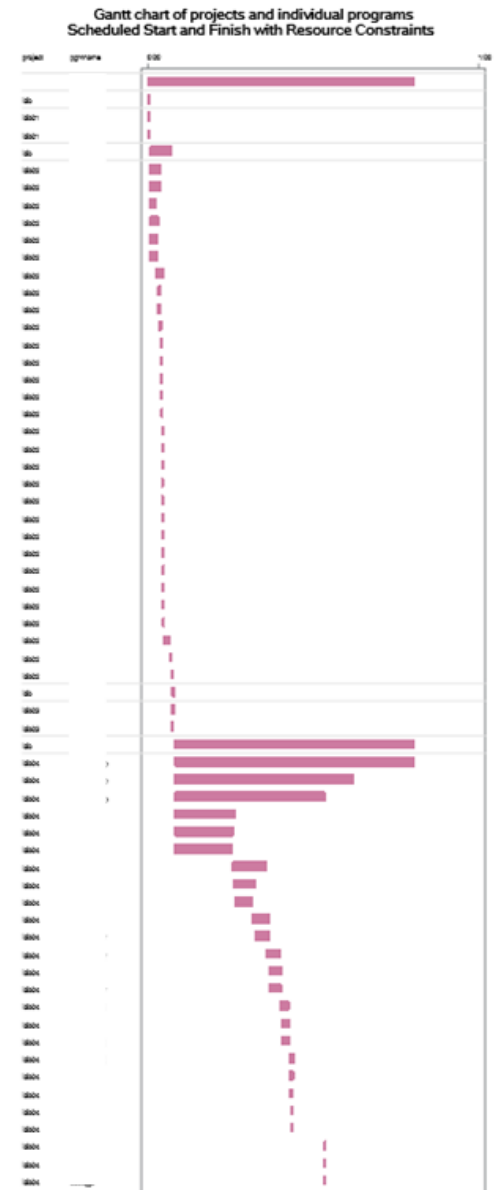
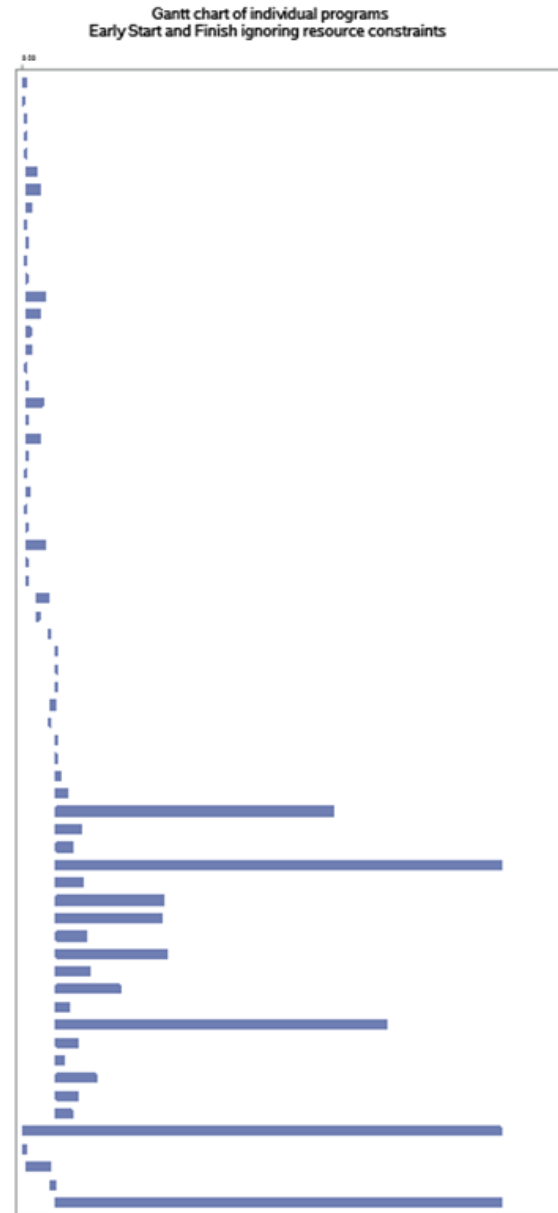
- Program names
- Grid session numbers
- Project IDs
- Durations (from input)
- Scheduled start and finish times
- Float time

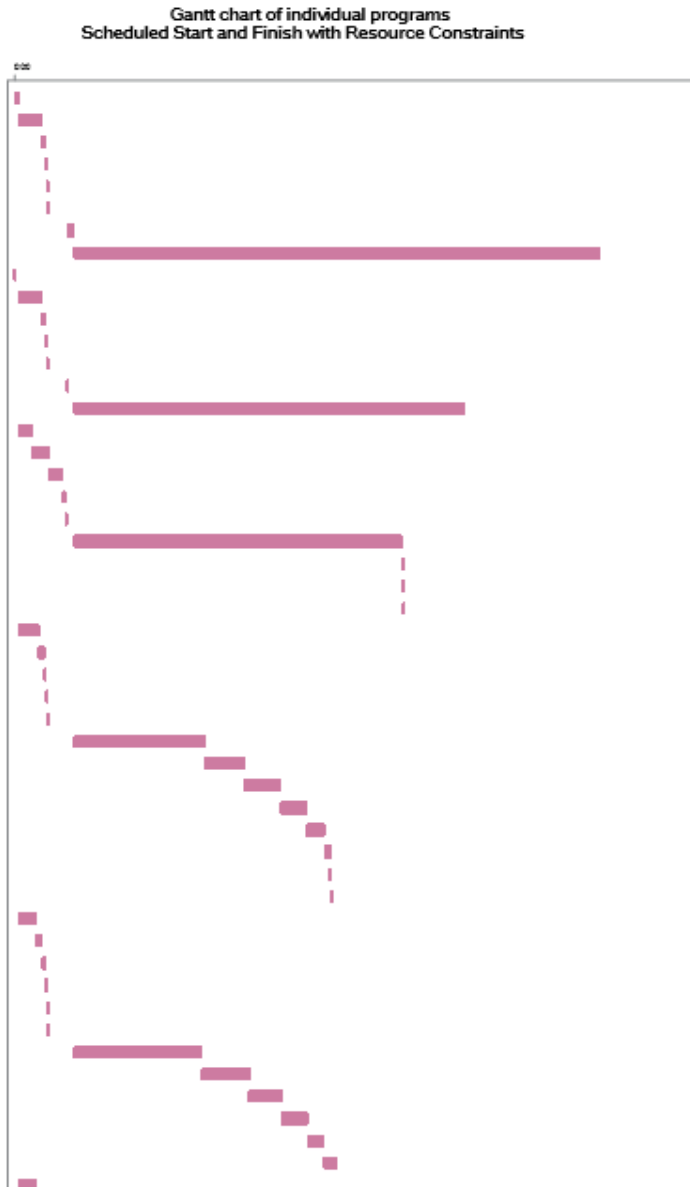
CPM results showing programs, elapsed time, and start options
Based on project assignments and some targeted grid nodes

WBS_CODE	project	pgmname	gridnum	tot_realtime	dur_seconds	S_START	S_FINISH	T_FLOAT
0		idb	.	.	.	0:00:00.00	0:48:24.70	0.00
0.0	idb	idb01	.	.	.	0:00:00.00	0:00:19.16	0.00
0.0.0	idb01	---	1	0:00:19	19.17	0:00:00.00	0:00:19.16	0.00
0.0.1	idb01	---	2	0:00:05	4.53	0:00:00.00	0:00:04.52	14.64
0.1	idb		.	.	.	0:00:19.17	0:04:18.23	0.00
0.1.10	idb02		1	0:01:57	116.67	0:00:19.17	0:02:15.83	24.23
0.1.24	idb02		2	0:01:51	111.36	0:00:19.17	0:02:10.52	29.54
0.1.03	idb02		3	0:01:06	65.57	0:00:19.17	0:01:24.73	0.00
0.1.16	idb02		4	0:01:38	98.38	0:00:19.17	0:01:57.54	42.52
0.1.04	idb02		5	0:01:25	84.94	0:00:19.17	0:01:44.10	55.96
0.1.18	idb02		6	0:01:24	83.69	0:00:19.17	0:01:42.85	57.21
0.1.11	idb02		3	0:01:23	82.56	0:01:24.74	0:02:47.29	58.34
0.1.05	idb02		6	0:00:32	32.32	0:01:42.86	0:02:15.17	108.58
0.1.12	idb02		5	0:00:30	30.27	0:01:44.11	0:02:14.37	110.63
0.1.13	idb02		4	0:00:27	27.47	0:01:57.55	0:02:25.01	113.43
0.1.21	idb02		2	0:00:21	21.48	0:02:10.53	0:02:32.00	119.42
0.1.17	idb02		5	0:00:16	16.29	0:02:14.38	0:02:30.66	124.61
0.1.26	idb02		6	0:00:15	14.82	0:02:15.18	0:02:29.99	126.08
0.1.07	idb02		1	0:00:14	13.59	0:02:15.84	0:02:29.42	127.31

Gantt Charts

- Gantt 1
 - Shows simple schedule without constrained grid sessions
 - Projects are not used
- Gantt 2
 - Shows schedule with limited grid sessions
 - Project are used
- Execution times are the same





Gantt Charts

- Gantt charts per grid session
- You can see how programs are scheduled within grid sessions to minimize the number of grid sessions used.

Results

- For those jobs that were running for several hours when executing programs sequentially, we were able to:
 - Cut job execution times to about 25% - 33% of the sequential times through parallel processing.
- We create reusable methods, such as:
 - Automated analysis of logs to collect times and dependencies
 - Automated scheduling with the CPM procedure
 - Assignment of programs to grid sessions and in specific orders
 - Automated generation of driver programs
- We were able to:
 - Reduce the effort required to achieve this reduction in job execution time.
 - Predict if parallel processing was worthwhile.

Conclusions – Key Steps to Optimization

- Determine if shorter execution time is worth the effort
- Analyze logs to summarize execution times and identify dependencies
- Feed data into a scheduling tool such as PROC CPM
- Predict time savings from parallel process and decide whether to proceed
- Use the results of PROC CPM to schedule programs
 - Using both schedule tables and Gantt charts
- Adjust for “dependency groups” if necessary
- Generate driver programs manually or automatically

Conclusions

- Given a stepwise and measurable approach to optimizing build jobs for integrated databases, you can gain considerable efficiencies through:
 - Parallel processing with the SAS Grid platform
 - Analysis of the parallel distribution of programs with analytic techniques such as CPM.
- Overall, we were very pleased with the results of our experiment. We continue to refine the techniques and make the code more re-usable and generalized. We plan to apply the techniques to other optimization scenarios.
- Note that we have not discussed how to optimize within a given program, but that is also part of the process.

Questions?

Thank you.

Contact Information:

Eric Brinsfield

President, Meridian Analytics

Principal Consultant, d-Wise Technologies

Eric.Brinsfield@meridiananalytics.com

Eric.Brinsfield@d-wise.com

Meridian Analytics

d-wise