

DH02: Easy Parallel Processing in SAS/BASE: a Case Study.

MSD 2021

Anna Socha

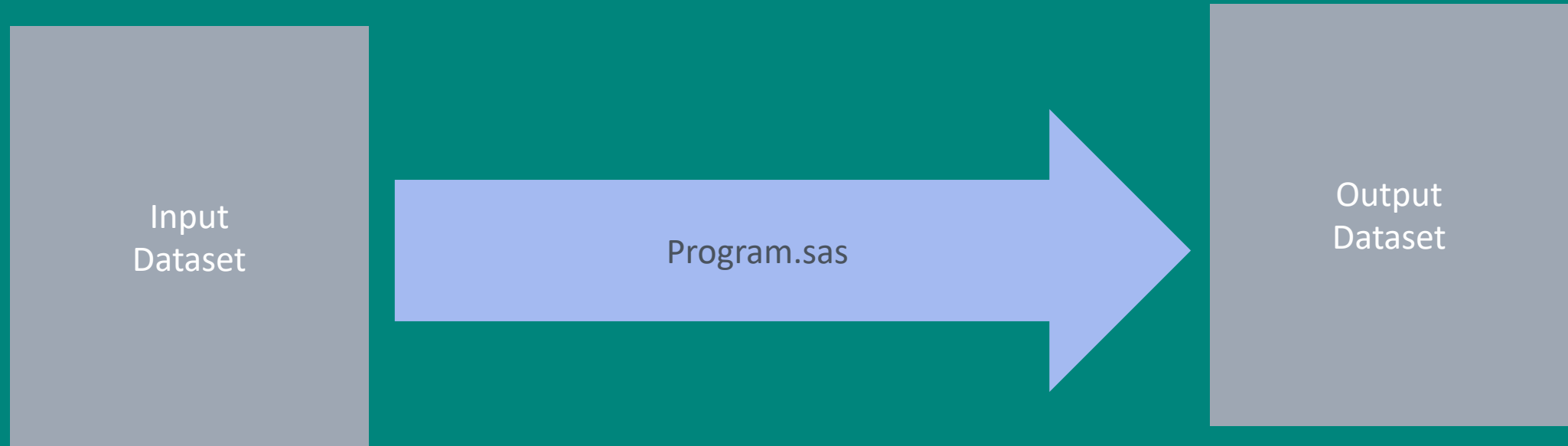
Carl Herremans

Justyna Wieczorek

Alexey Stalbovskiy

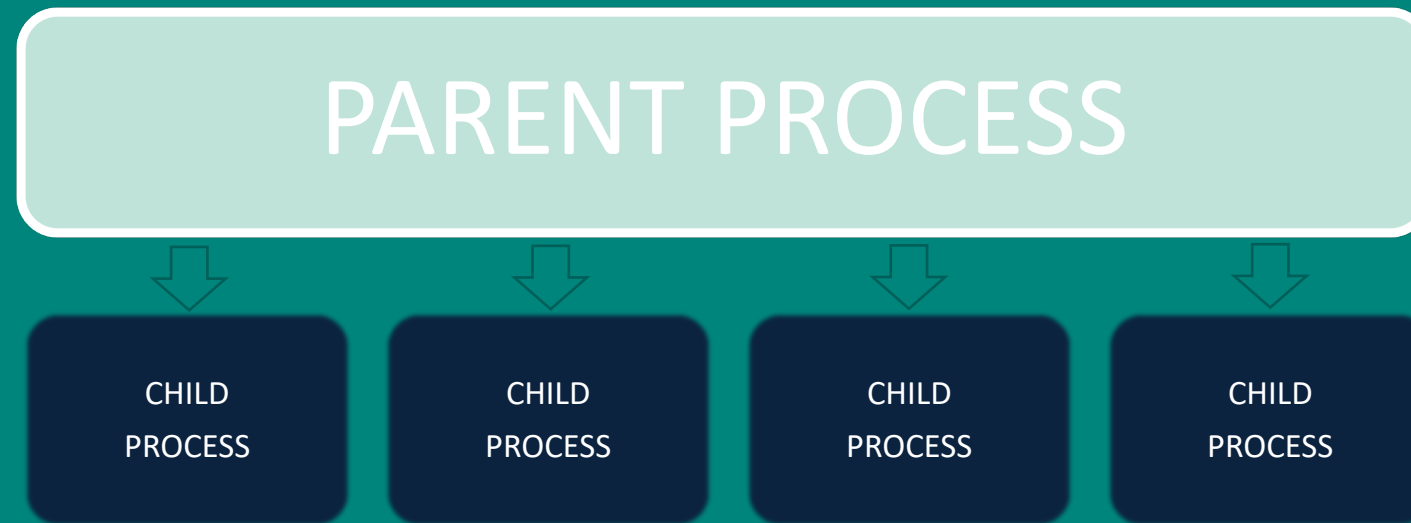
Paul Wicks

Classic Process: 1 CPU



Parallel Processing

Efficient and easy to implement way to speed up the processing of large datasets



Parallel Processing: parts required



Program.sas

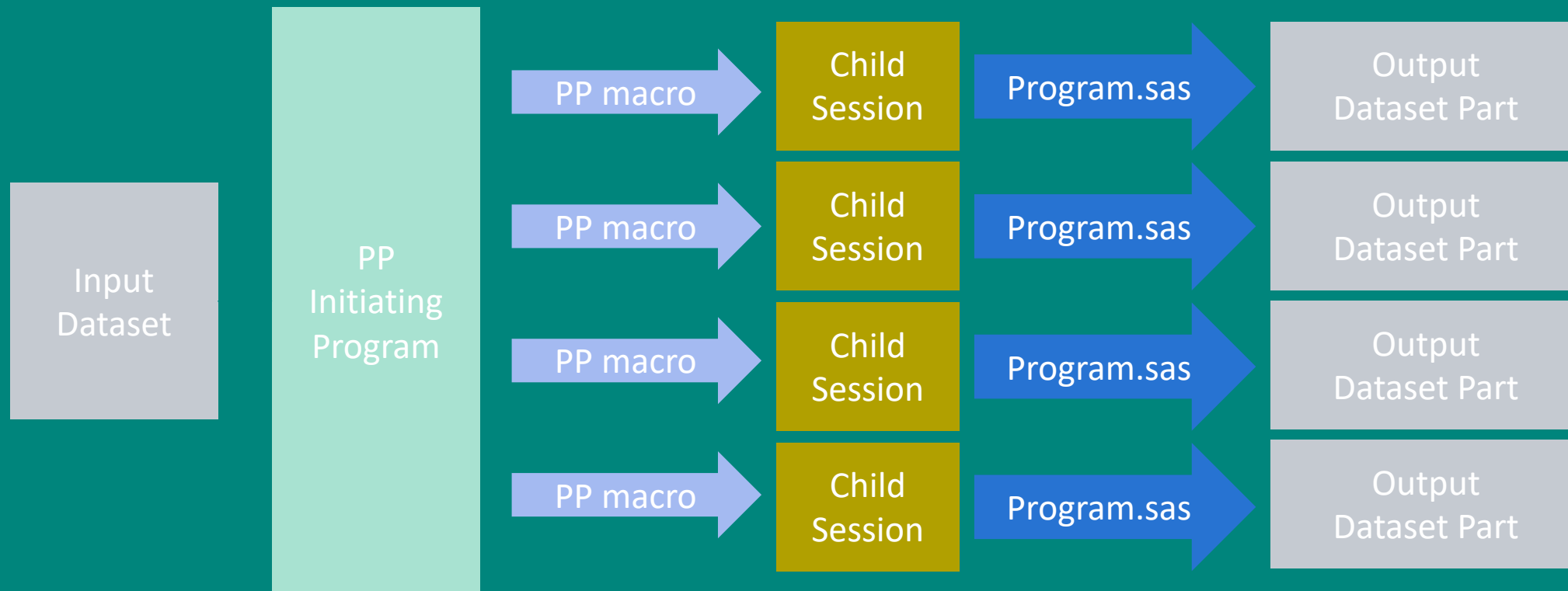


Parallel processing macro

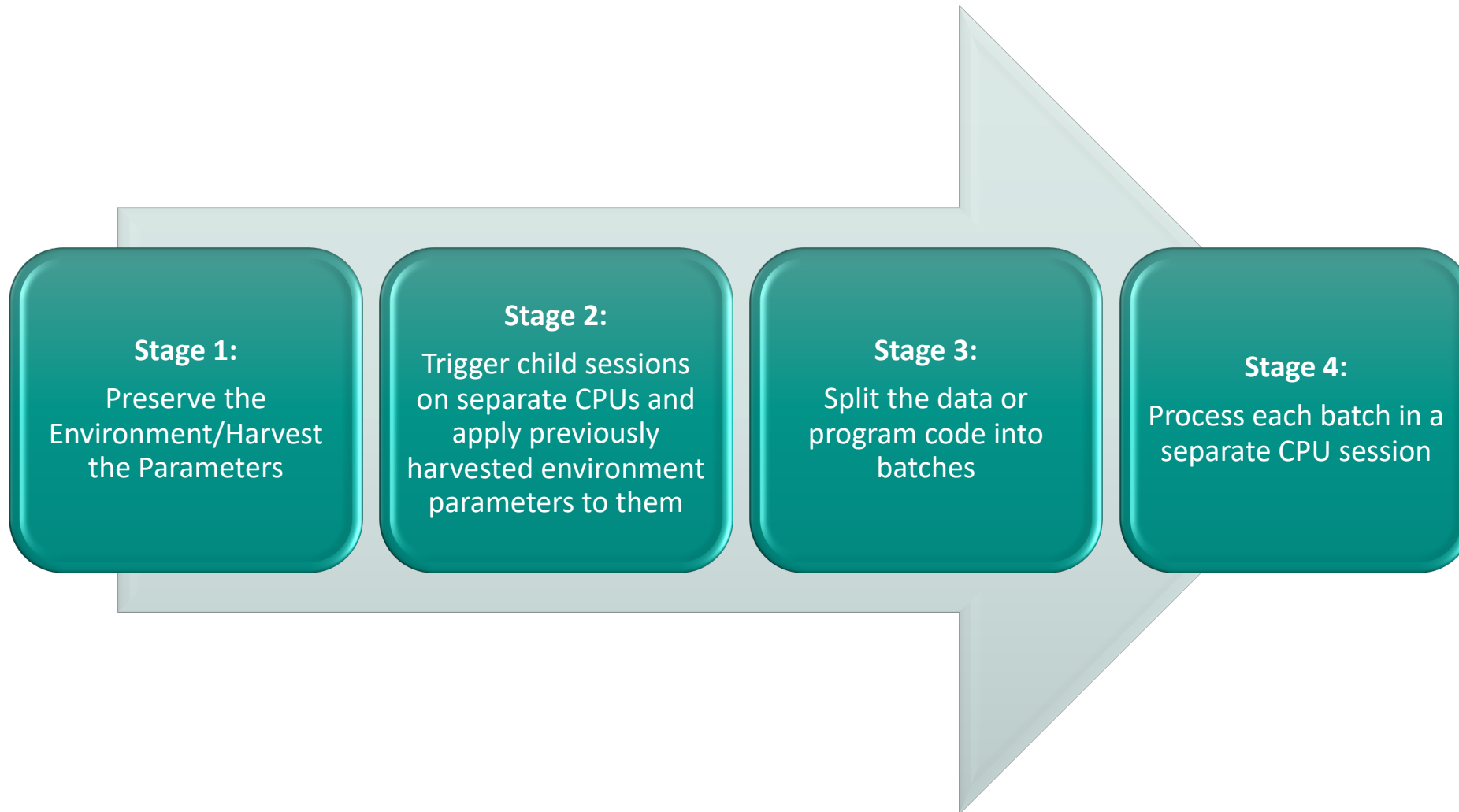


Program that splits the input dataset/proc into fragments and calls the parallel processing macro for each of them

Parallel Processing (PP)



Parallel Processing : Four Stage Process



Stage 1: The PP macro harvests the parameters of the parent session

Options

```
proc optsave  
out=options;  
run;
```

Libraries:

```
proc sql;  
  create table libname as  
  select * from sashelp.vslib  
  where libname not in ('MAPS'  
    'MAPSGFK' 'MAPSSAS' 'SASHELP'  
    'SASUSER' 'WORK' 'SASADMIN');  
quit;
```

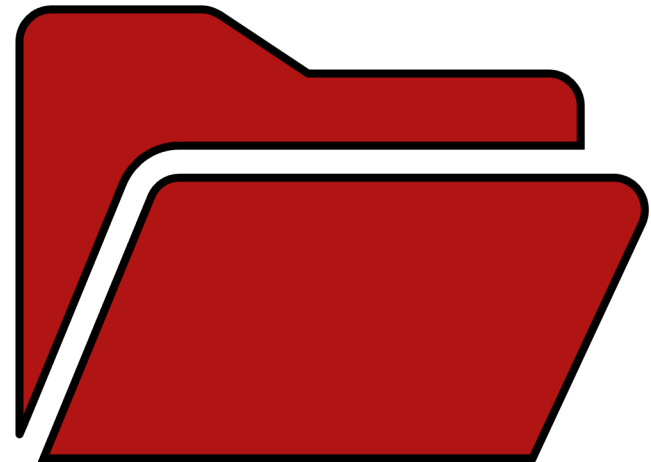
Filenames:

```
proc sql;  
  create table filename as  
  select *  
  from dictionary.extfiles  
  where lowercase(TEMPORARY) ='no';  
quit;
```

Stage 1:

The PP macro generates new startup files for each child session

```
data _null_;  
  file "childstartup.sas" ;  
  put "<Parent Start Up File Code Line 1>";  
  put "<Parent Start Up File Code Line 2>";  
  ...  
  put "<Parent Start Up File Code Line n>";  
run;
```



Stage 1:

Startup Will Assign Options

```
data _null_;  
    file "childstartup.sas" mod ;  
    put "proc optload data= options ;run;";  
run;
```

Stage 1:

Startup Will Assign Libraries

```
data _null_;  
  file "childstartup.sas" mod;  
  set libname;  
  
  code=libname (LIBNAME, PATH) ;  
  if code=-70008 then delete;  
  
  if index(path, '(') then  
    put 'LIBNAME ' LIBNAME PATH + (-1) ' ';  
  else  
    put 'LIBNAME ' LIBNAME '"' PATH + (-1) '";'  
run;
```

Stage 1:

Startup Will Assign Filenames

```
data _null_;  
    file "childstartup.sas" mod;  
    set filename;  
    put 'filename ' FILEREF '"' XPATH + (-1) '"';  
run;
```

Stage 1: Additional Considerations

1) It is possible to preserve parent work library

```
libname mwork "%sysfunc(getoption(work))";
```

2) Data driven parameters and macro parameters can be passed to the child program

```
data _null_;  
    file "childprogram.sas";  
    call symget('_var',_var);  
    put '<required statements> _var <required statements>';  
run;
```

Stage 2:

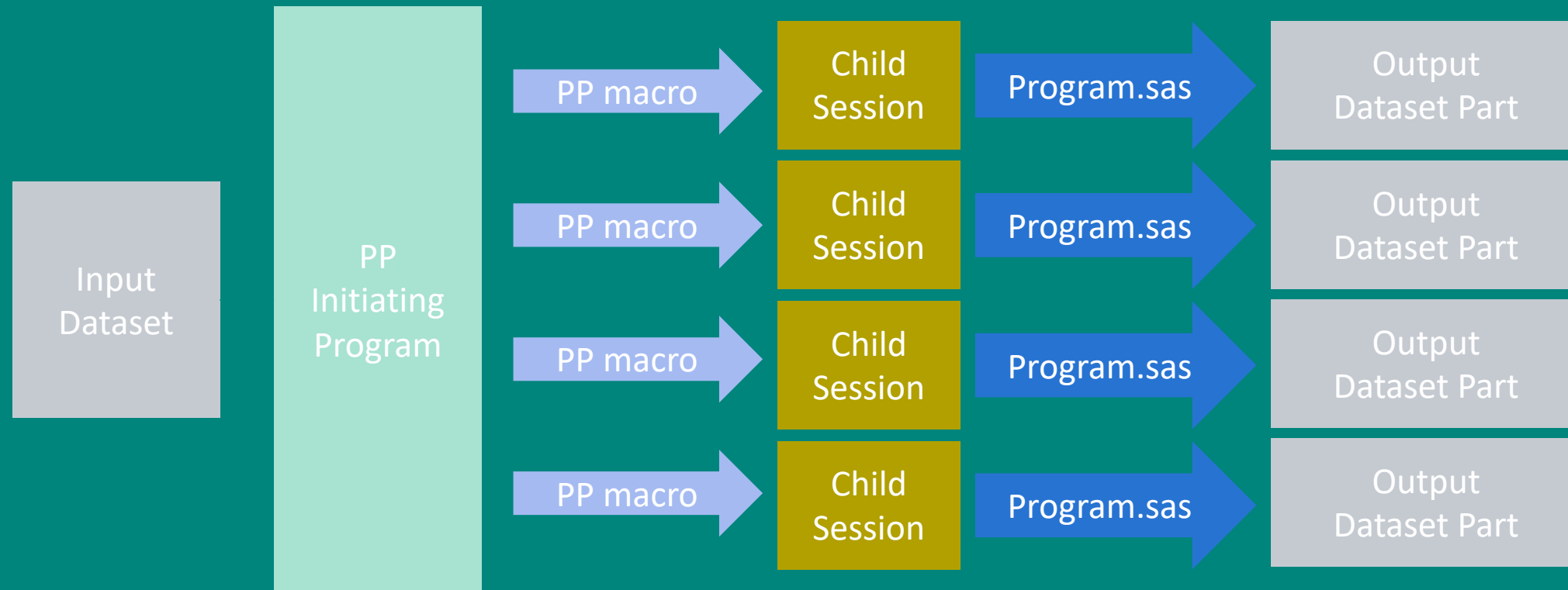
The PP macro initiates the child sessions

A systask COMMAND sets up a SAS session using options that are setup specific and runs the child startup code and the child program code

Use the WAIT and TASKNAME options so a waitfor statement can be used later to force SAS to wait until all parallel processes have finished before continuing

```
systask COMMAND "sas94 childstartup.sas childprogram.sas <other Linux SAS commands>"  
nowait taskname=&taskid.;
```

Stage 3: Split the data or program code into batches
Stage 4: Process each batch in a separate CPU session



Case study: Bootstrap Confidence Interval – Set-up

No Parallel Processing

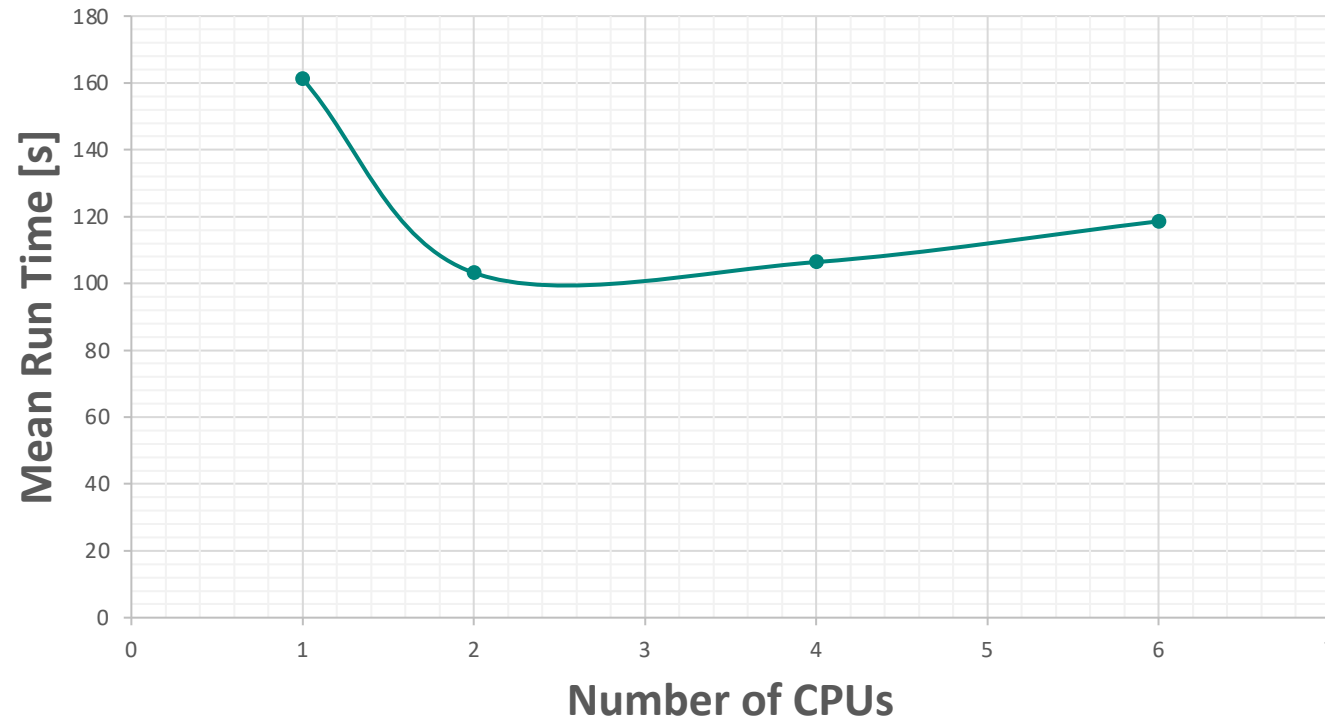
- Step 1a: Proc Means by sample number and treatment group on data set with 1000 samples. Output a data set of 1000 difference in means

Parallel Processing

- Step 1a: Divide 1000 samples into 4 data sets of 250
- Step 1b: Run 4 sessions in parallel, Running the Proc Means by sample number and treatment group on data sets with 250 samples. Run these AT THE SAME
- Step 1c: Wait for Processes to Finish
- Step 1d: Combine output data sets of each session to create an output data set of 1000 difference in means.

Case study: Bootstrap Confidence Interval - Results

Results of Parallel Processes on Run Time



Number of CPUs	n=1	n=2	n=4	n=6
Mean Run Time [s]	161.2	103.2	106.4	118.6
Standard Deviation	9.9	3.2	7.8	25.2

Thank you

anna.socha@msd.com