Beating Gridlock: Parallel Programming with SAS® Grid Computing and SAS/CONNECT®

Thursday, February 20, 2014

Jack Fuller
Experis Business Analytics
Introduction – The Problem

• A SAS program is taking a very long time to execute

• The program is constructed of multiple, independent subtasks

• Simulation programs frequently exhibit this behavior
Introduction – The Solution

- Split the program into a driver program and a set of subprograms which run in parallel

Elapsed Real Time

Total CPU Time
Overview

• **When to use parallel processing**

• **How to use parallel processing**

• **Points to consider**
When to Use Parallel Processing
Independent Subtasks

Initialization

Independent Subtask 1

Independent Subtask 2

Independent Subtask 3

Finalization
Independent Subtasks

- Initialization
- Independent Subtask 1
- Independent Subtask 2
- Independent Subtask 3
- Finalization
Independent Subtasks

Initialization

Independent Subtask 1

Independent Subtask 2

Independent Subtask 3

Finalization
Independent Subtasks

- Initialization
- Independent Subtask 1
- Independent Subtask 2
- Independent Subtask 3
- Finalization
Independent Subtasks

Initialization

Independent Subtask 1

Independent Subtask 2

Independent Subtask 3

Finalization
Independent Subtasks

Initialization

Independent Subtask 1

Independent Subtask 2

Independent Subtask 3

Finalization
Independent Subtasks

SAS Connect 9.3 User’s Guide

Initialization

Server

Independent Subtask 1

Client

Independent Subtask 2

Client

Independent Subtask 3

Client

Finalization

Server
Independent Subtasks

Grid Computing in SAS 9.3

Initialization

Task

Subtask

Finalization

Task

Subtask

Subtask

Subtask
Types of Concurrency

• **Data Parallelism - subtasks work on different pieces of data**
  – Subtask 1 loads observations 1-10,000
  – Subtask 2 loads observations 10,001-20,000
  – ...

• **Task Parallelism - subtasks perform different functions**
  – Subtask 1 processes demographics data
  – Subtask 2 processes vital signs data
  – ...

Amdahl’s Law

• $P$ is the number of processors
• is the fraction of work that is not parallelized

Describes the optimal increase in speed gained from converting a serial process to a parallel process
Amdahl’s Law as

\[ \text{Speed} = \frac{1}{\frac{1}{P} + \alpha} \]

- \( P \) is the number of processors
- \( \alpha \) is the fraction of work that is not parallelized

As the number of processors (\( P \)) increases, the increase in speed is constrained by the fraction of work that is not parallelized (\( \alpha \)).
Graphical Representation of Amdahl's Law
How to Use Parallel Processing
Our Problem – Creating Test Data

do pt=1 to 10000;
  if uniform(0) > .40 then do;
    gender = 'M';
    /* Generate test data */
  end;
else do;
  gender = 'F';
  /* Generate test data */
end;
output;
end;

How do we want to define our concurrency?
Our Problem – Creating Test Data

```plaintext
do pt=1 to 10000;
    if uniform(0) > .40 then do;
        gender = 'M';
        /* Generate test data */
        end;
    else do;
        gender = 'F';
        /* Generate test data */
        end;
    output;
end;
```
Our Problem – Creating Test Data

do pt=1 to 10000;
    if uniform(0) > .40 then do;
        gender = 'M';
        /* Generate test data */
        end;
    else do;
        gender = 'F';
        /* Generate test data */
        end;
    output;
end;
Our Problem – Creating Test Data

do pt=1 to 10000;
    if uniform(0) > .40 then do;
        gender = 'M';
        /* Generate test data */
        end;
    else do;
        gender = 'F';
        /* Generate test data */
        end;
    output;
end;

Let’s choose task parallelism by SDTM domain
Example – Initialization

```sas
%let seed = 0;
%let numPt = 1000;

%let rc=%sysfunc(grdsvc_enable(_all_,
    resource=SASApp));

options autosignon;

libname shared "%sysfunc(pathname(work))";
```

Passed to each subtask
Example – Initialization

%let seed = 0;
%let numPt = 1000;

%let rc=%sysfunc(grdsvc_enable(_all_,
    resource=SASApp));

options autosignon;

libname shared "%sysfunc(pathname(work))";
Example – Initialization

%let seed = 0;
%let numPt = 1000;

%let rc=%sysfunc(grdsvc_enable(_all_,
    resource=SASApp));

options autosignon;

libname shared "%sysfunc(pathname(work))";
Example – Initialization

%let seed = 0;
%let numPt = 1000;

%let rc=%sysfunc(grdsvc_enable(_all_,
resource=SASApp));

options autosignon;

libname shared "%sysfunc(pathname(work))";
Example – Create the Initial Data

```
data PT GENDER
  1     ... 
  2     ...
  3     ...
...         , 'F') ;
run;
```

Our input data lives in the task’s work directory
Example – Demographics Subtask

```
%syslput seed=&seed / remote=task1;
rsubmit task1 wait=no inheritlib=(shared);
data shared.task1(keep=pt ht wt);
  set shared.pt;
  if gender = 'M' then do;
    /* Compute statistics */
    end;
  else do;
    /* Compute statistics */
    end;
run;
endrsubmit;
```

Create a macro variable in the subtask and initialize it to &seed
Example – Demographics Subtask

%syslput seed=&seed / remote=task1;
rssubmit task1 wait=no inheritlib=(shared);
data shared.task1(keep=pt ht wt);
   set shared.pt;
   if gender = ‘M’ then do;
      /* Compute statistics */
      end;
   else do;
      /* Compute statistics */
      end;
run;
endrssubmit;

Delineate the code which will run in the subtask
Example – Demographics Subtask

%syslput seed=&seed / remote=task1;
rsreply task1 wait=no inheritlib=(shared);
  data shared.task1(keep=pt ht wt);
    set shared.pt;
    if gender = ’M’ then do;
      /* Compute statistics */
      end;
    else do;
      /* Compute statistics */
      end;
    run;
endrsreply;
Example – Demographics Subtask

%syslput seed=&seed / remote=task1;
rsSubmit task1 wait=no inheritlib=(shared);
   data shared.task1(keep=pt ht wt);
      set shared.pt;
      if gender = 'M' then do;
         /* Compute statistics */
         end;
      else do;
         /* Compute statistics */
         end;
   run;
endrsSubmit;

Specify asynchronous execution
Example – Demographics Subtask

%syslput seed=&seed / remote=task1;
rssubmit task1 wait=no inheritlib=(shared);
  data shared.task1(keep=pt ht wt);
    set shared.pt;
    if gender = 'M' then do;
      /* Compute statistics */
      end;
    else do;
      /* Compute statistics */
      end;
  run;
endrsubmit;
Example – Demographics Subtask

```sas
%syslput seed=&seed / remote=task1;
rsubmit task1 wait=no inheritlib=(shared);
data shared.task1(keep=pt ht wt);
   if gender = 'M' then do;
      /* Compute statistics */
   end;
   else do;
      /* Compute statistics */
   end;
run;
endrsubmit;
```

<table>
<thead>
<tr>
<th></th>
<th>PT</th>
<th>HT</th>
<th>WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example – Labs Subtask

```plaintext
%syslput seed=&seed / remote=task2;
RSubmit task2 wait=no inheritlib=(shared);
data shared.task2(keep=pt wbc rbc);
set shared.pt; /* Compute statistics */
run;
endrsubmit;
```

<table>
<thead>
<tr>
<th></th>
<th>PT</th>
<th>WBC</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example – Vital Signs Subtask

```sas
%syslput seed=&seed / remote=task3;
rssubmit task3 wait=no inheritlib=(shared);
data shared.task3(keep=pt syst_bp diast_bp);
  if gender = 'M' then do;
    /* Compute statistics */
    end;
  else do;
    /* Compute statistics */
    end;
run;
endrssubmit;
```

<table>
<thead>
<tr>
<th>PT</th>
<th>SYST_BP</th>
<th>DIAST_BP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example – Finalization

```plaintext
waitfor _all_ task1 task2 task3;

data final;
   merge pt task:;
       by pt;
run;

signoff _all_;
```

Wait for subtasks
Example – Finalization

<table>
<thead>
<tr>
<th>PT</th>
<th>GENDER</th>
<th>HT</th>
<th>WT</th>
<th>WBC</th>
<th>RBC</th>
<th>SYST_BP</th>
<th>DIAST_BP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

```sql
waitfor _all_ task1 task2 task3;
data final;
merge pt task:;
by pt;
run;
signoff _all_
```

Notice the use of the task work library
Example – Finalization

waitfor _all_ task1 task2 task3;

data final;
    merge pt task:;
        by pt;
run;

signoff _all_;
Points to Consider
How many subtasks should there be?

• **This will depend on a number of factors**
  – The fraction of work that is not parallelized
  – The number of available grid nodes

• **My rule of thumb**
  – I like to benchmark my applications at 5, 10 and 15 subtasks and then adjust accordingly
How much work should occur in each subtask?

• **There has to be enough work in each subtask**
  – SAS/CONNECT has a login process which I have benchmarked as taking between 5 and 15 seconds
  – The goal is to have the friction involved with creating/managing the subtasks become negligible

• **My rule of thumb**
  – I like to have each subtask take at around 10 minutes
Should libraries be shared between subtasks?

• **Arguments for using INHERITLIB**
  – One thing is in one place

• **Arguments against using INHERITLIB**
  – Subtasks can bottleneck as they queue up awaiting access

• **My rule of thumb**
  – I like to use INHERITLIB unless I have a reason not to
Conclusion
Conclusion

• *Put as much work as possible in the subtasks*

• *Get something up and running, benchmark it and then adjust things*

• *Good luck!*
Questions
Acknowledgement

• **SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.**

• **Other brand and product names are registered trademarks or trademarks of their respective companies.**
Thank you…

Jack Fuller
Senior Application Developer
Jack.fuller@experis.com