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

Thursday, February 20, 2014

Jack Fuller
Experis Business Analytics

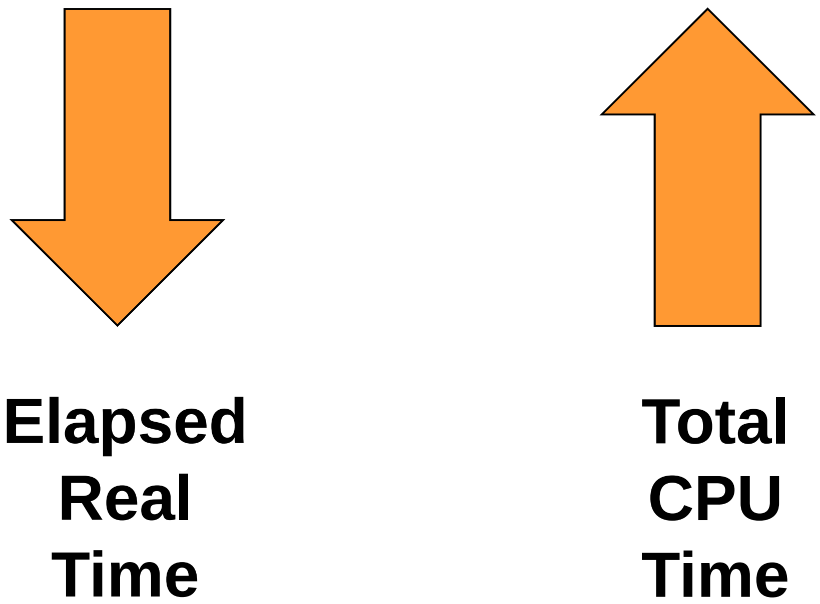
Experis™
ManpowerGroup

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**

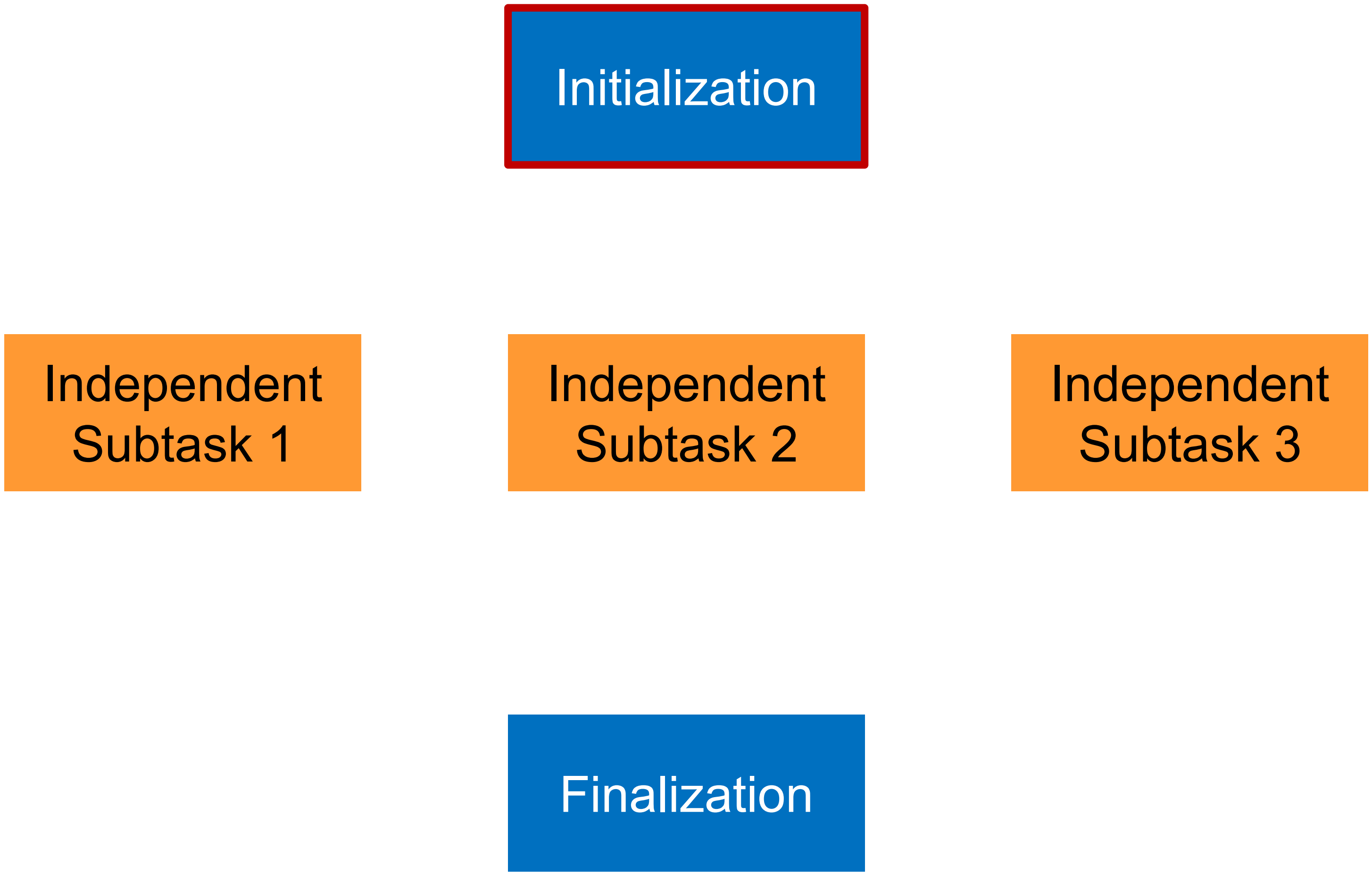


Overview

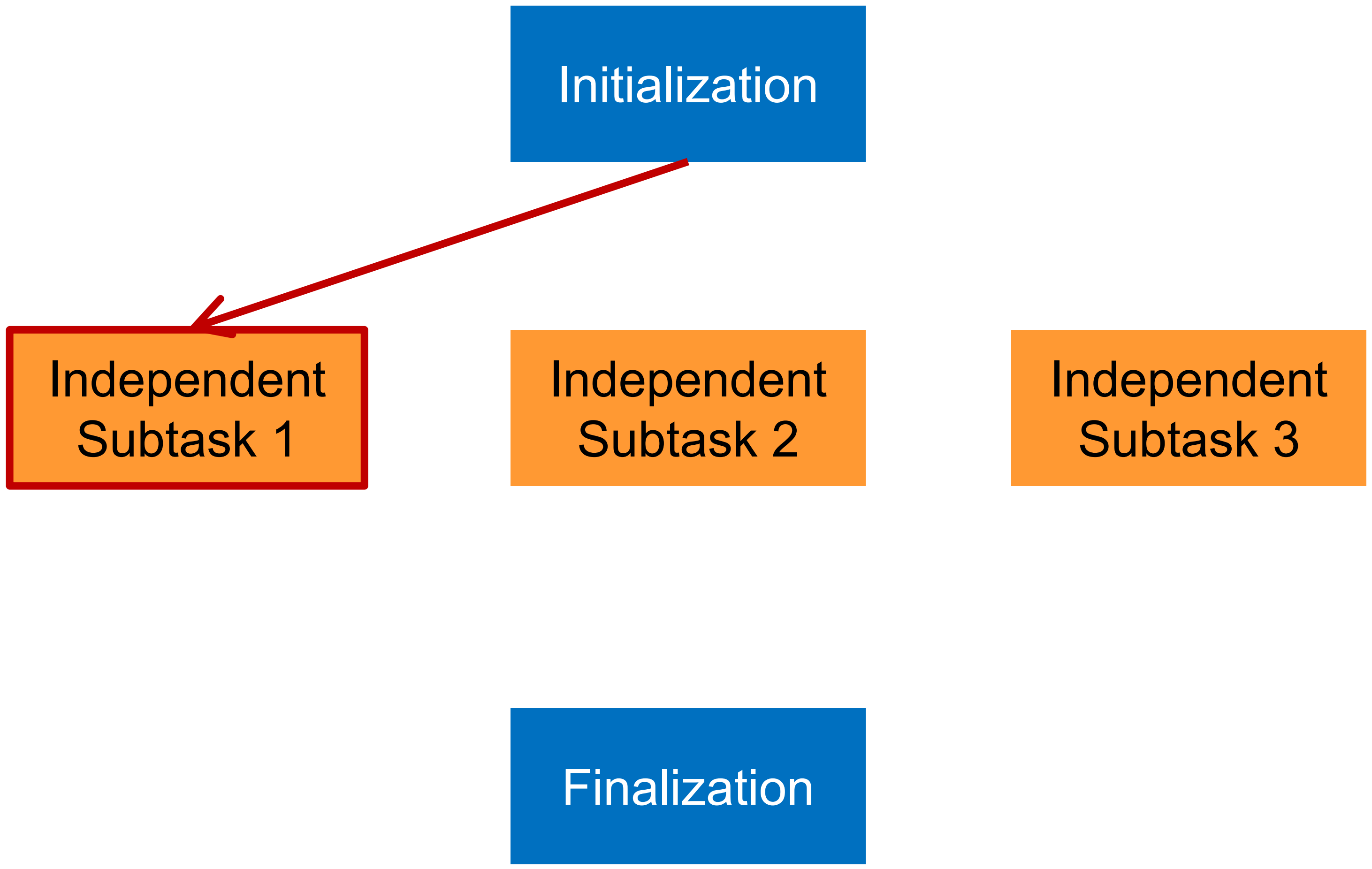
- **When to use parallel processing**
- **How to use parallel processing**
- **Points to consider**

When to Use Parallel Processing

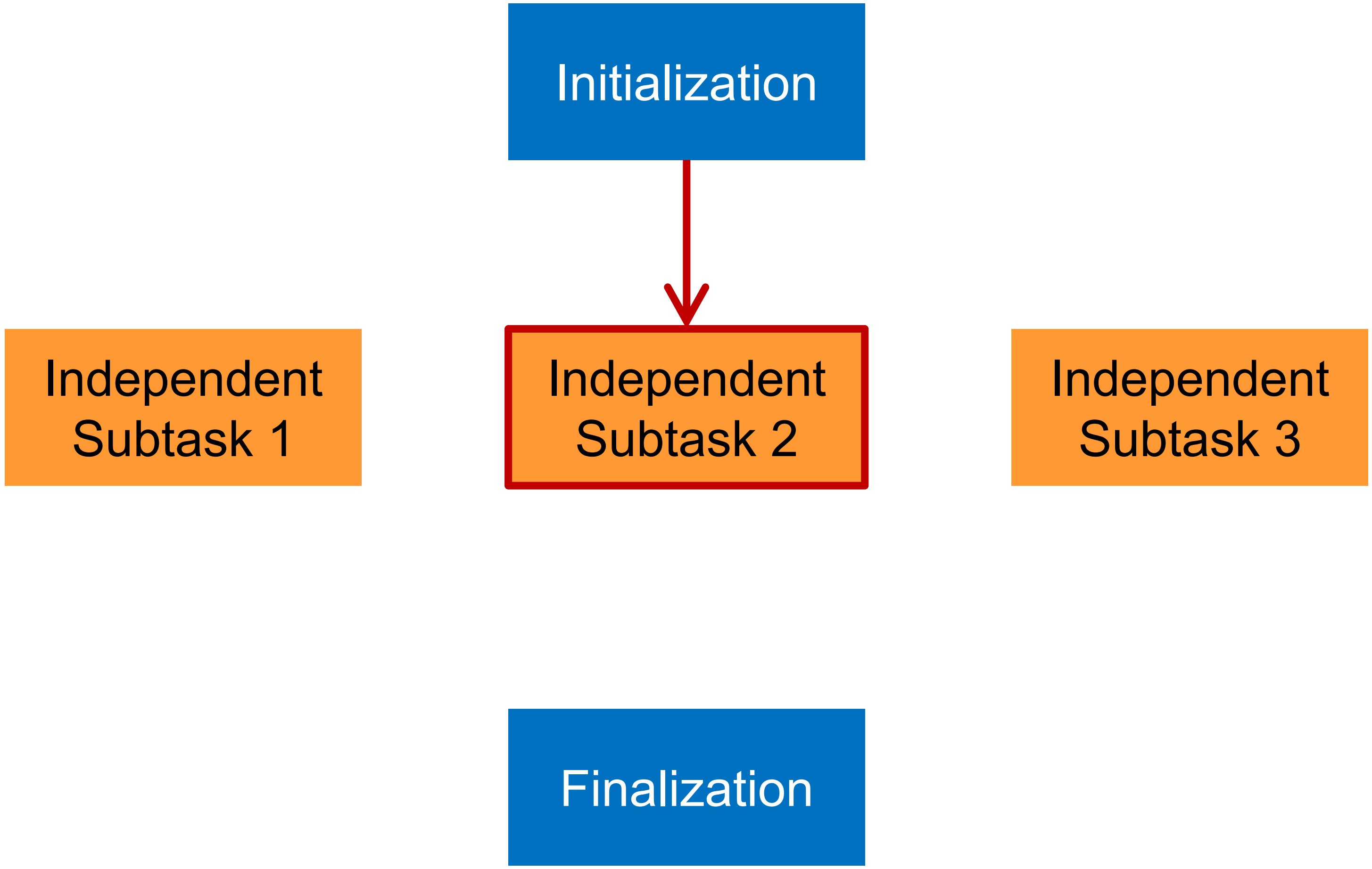
Independent Subtasks



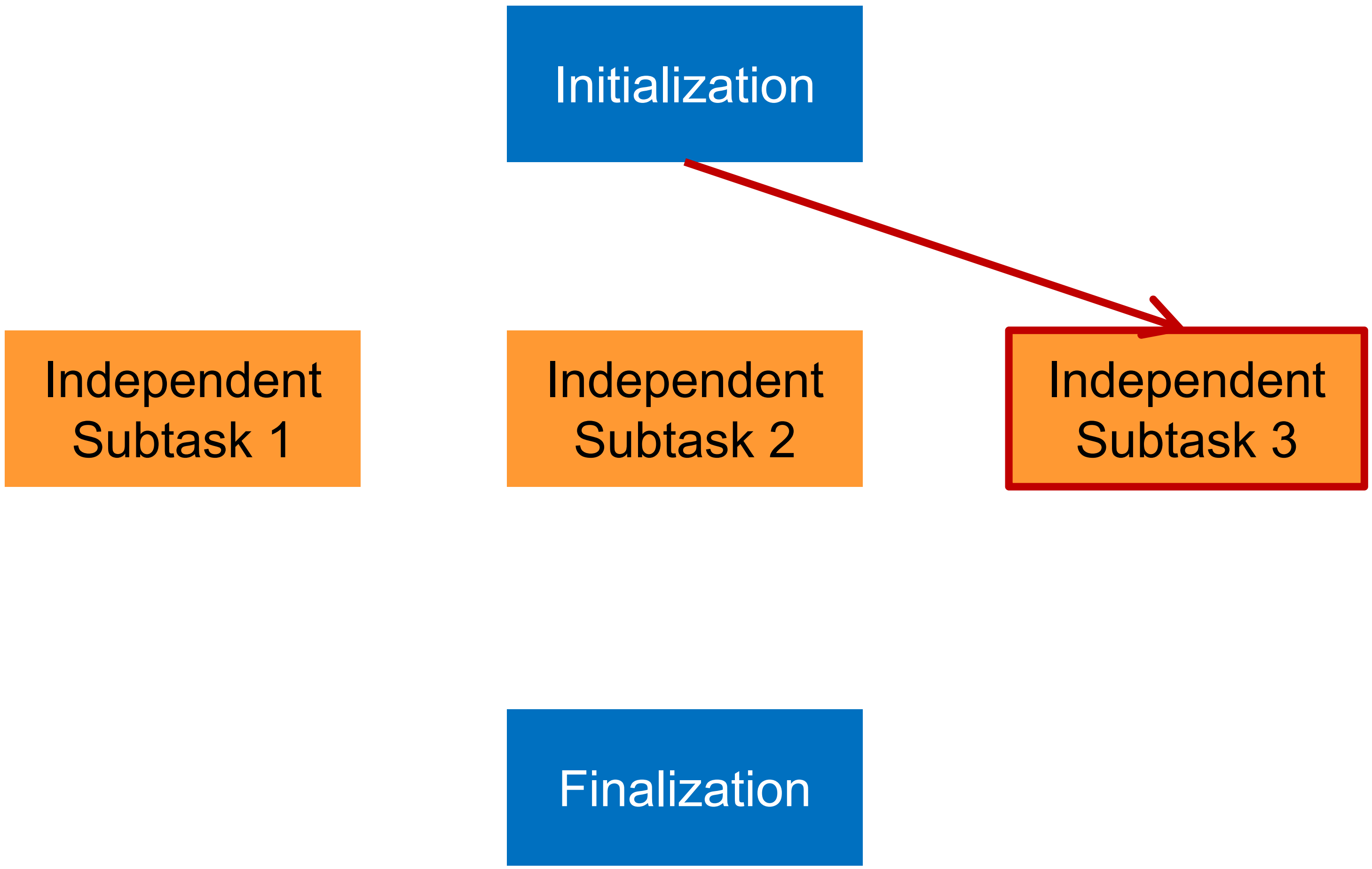
Independent Subtasks



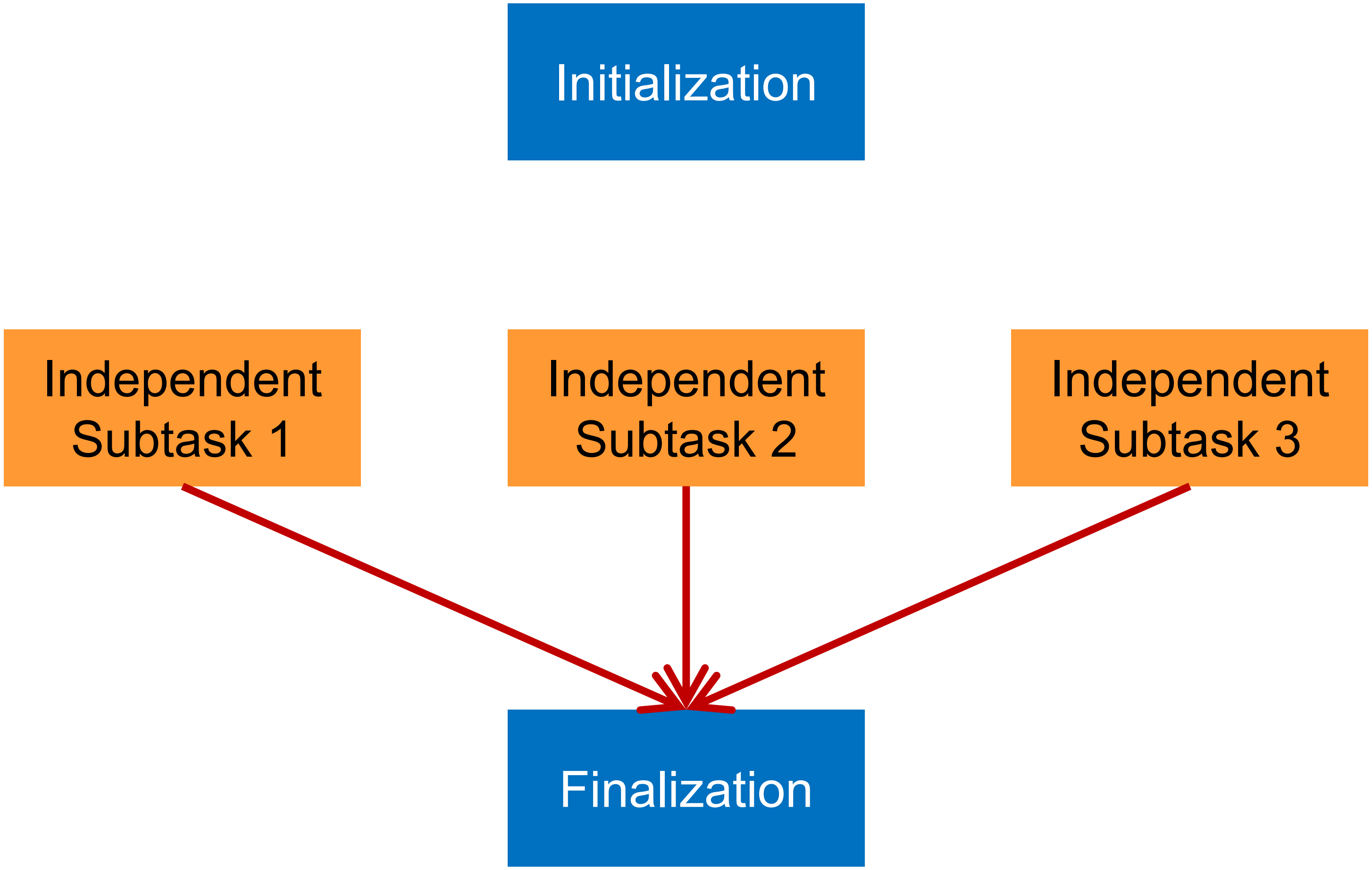
Independent Subtasks



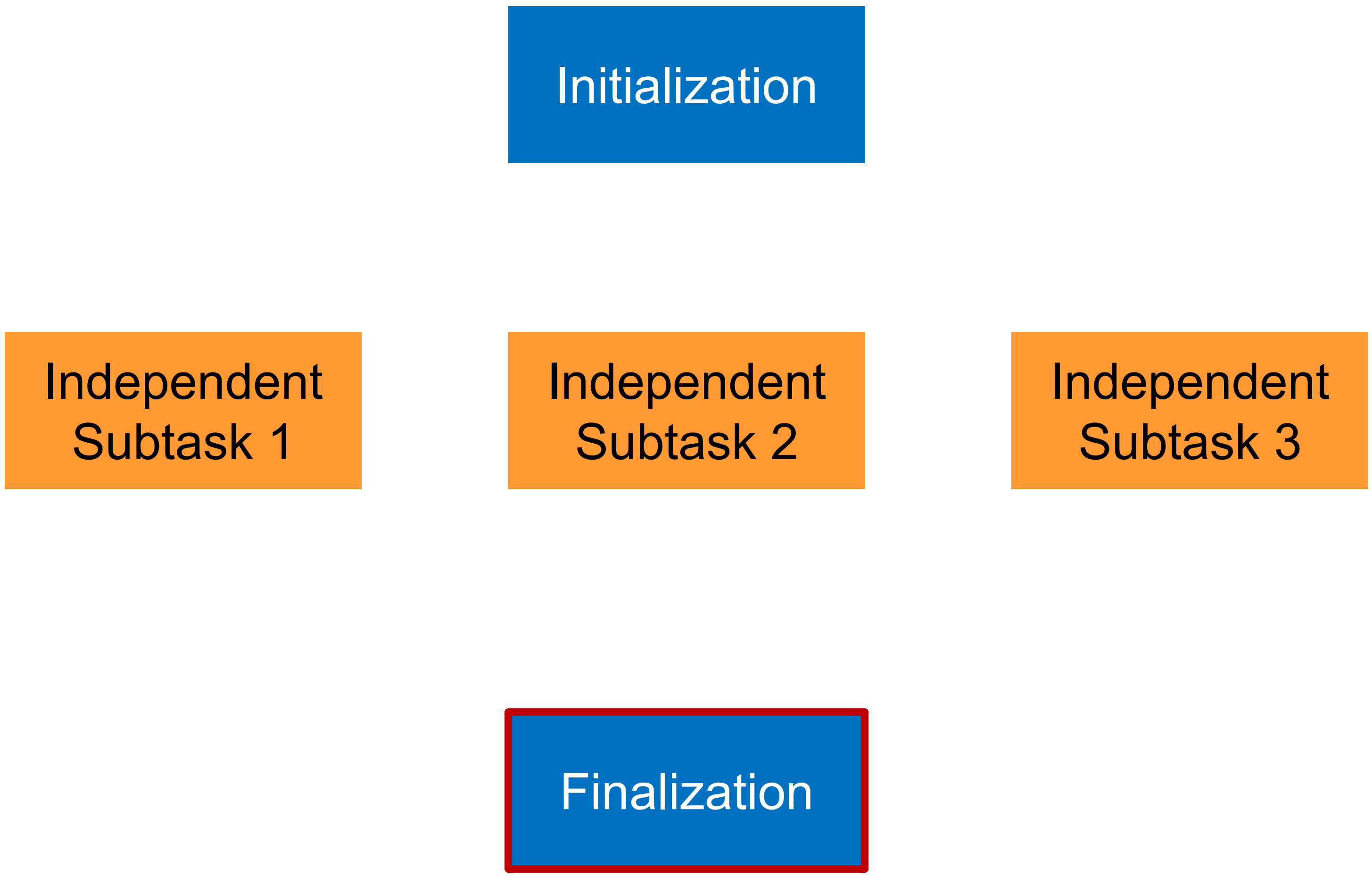
Independent Subtasks



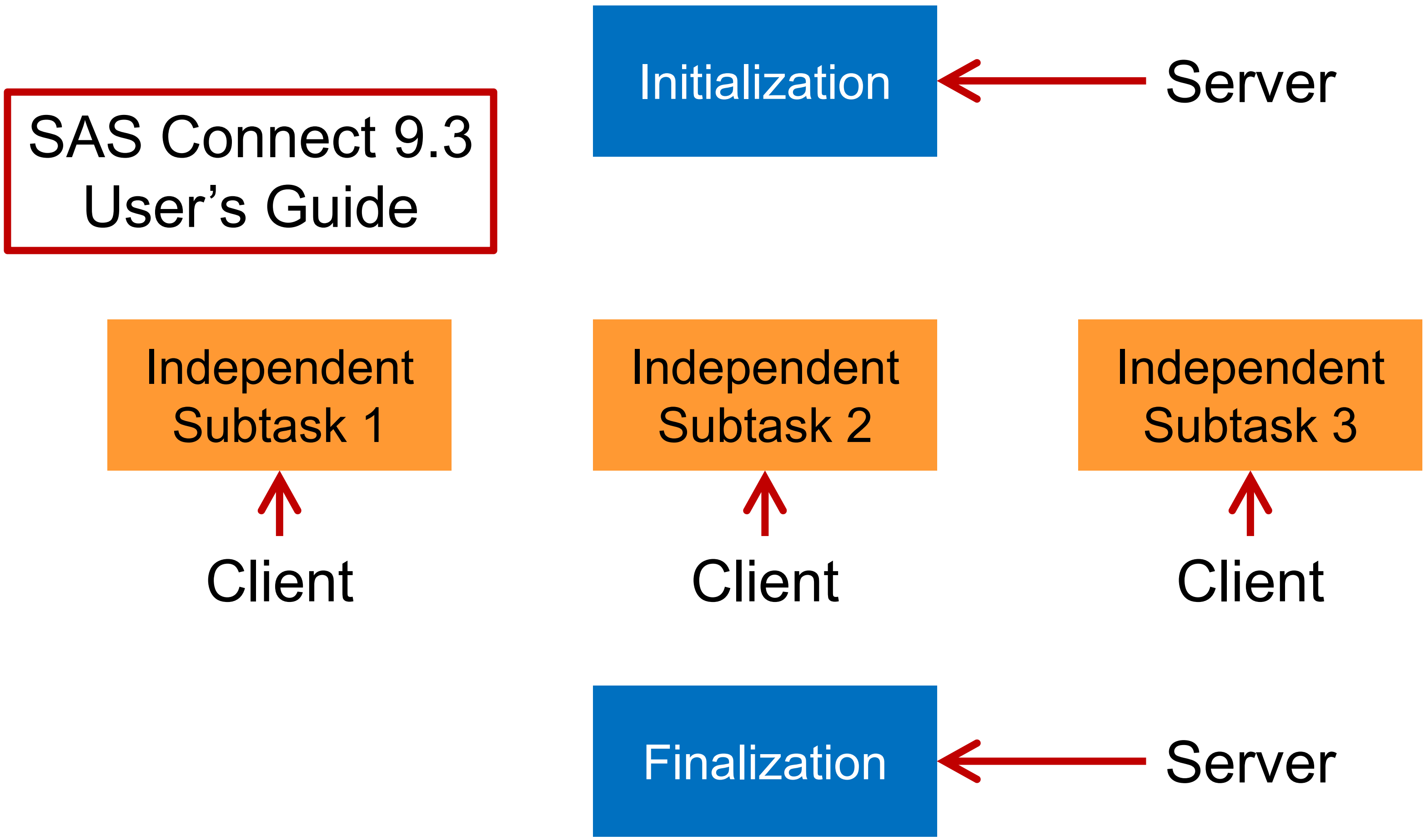
Independent Subtasks



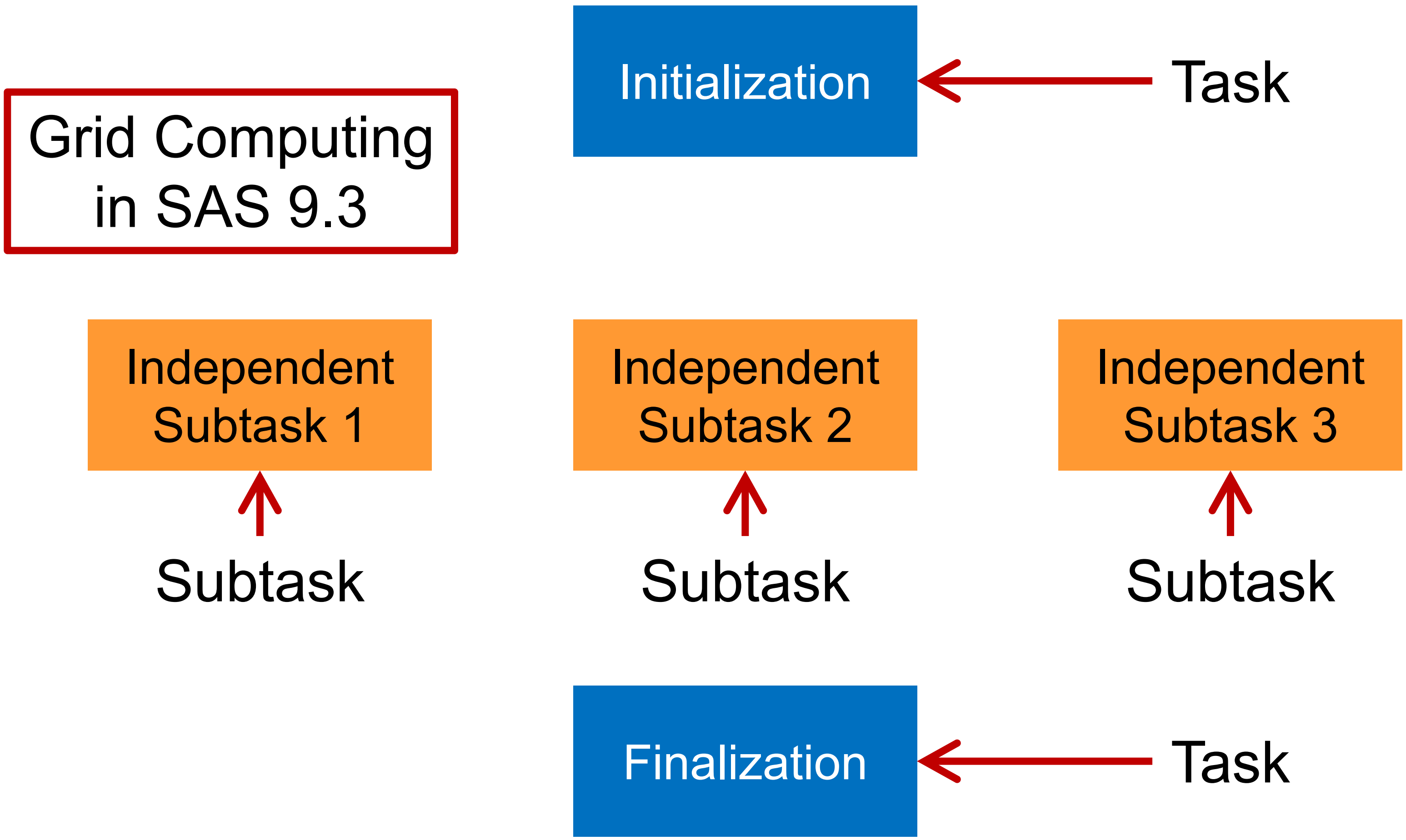
Independent Subtasks



Independent Subtasks



Independent Subtasks



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**

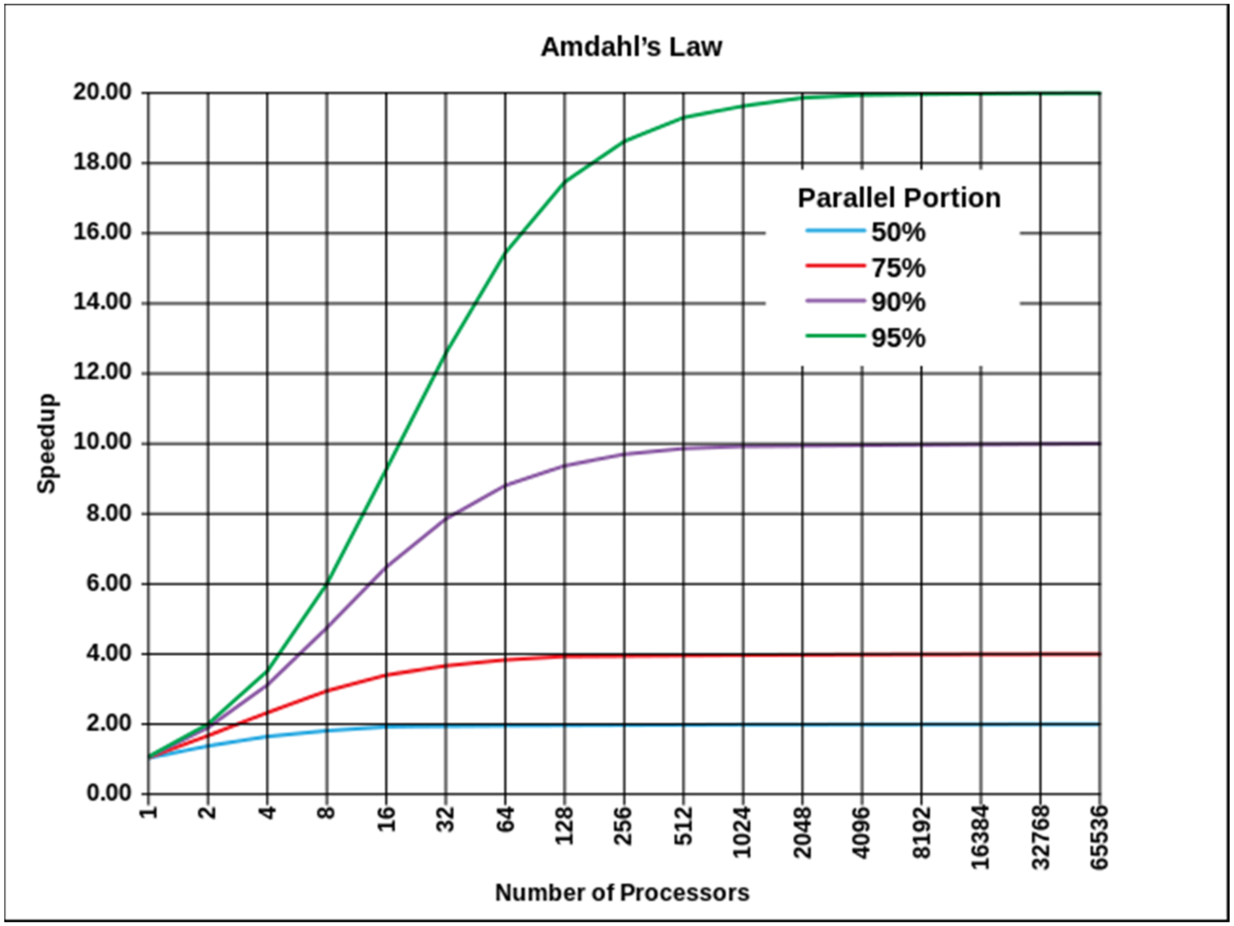
Describes the **optimal** increase in speed gained from converting a serial process to a parallel process

Amdahl’s Law as

- **P is the number of processors**
- **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 (α)

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

```
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;
```

Data Parallelism

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;
```

Task Parallelism

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

```
%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))";
```

Enable the grid service

Example – Initialization

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

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

options autosignon;

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

Allow the program to sign on without user interaction

Example – Initialization

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

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

options autosignon;

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

Point a LIBREF to the task's work library

Example – Create the Initial Data

```
data
    d
                                              , 'F');

    e
run;
```

PT	GENDER
1	…
2	…
3	…
…	

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;
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;
```

Delineate the code which will run in the subtask

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;
```

Name of the subtask

NOTE: Subtask names are limited to 8 chars

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;
```

Specify asynchronous execution

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;
```

Give the subtask access to the specified library

Example – Demographics Subtask

PT	HT	WT
1	…	…
2	…	…
3	…	…
…	…	…

```
%sysl
rsubm
   da
          /* Compute statistics */
      end;
      else do;
          /* Compute statistics */
      end;
   run;
endrsubmit;
```

Example – Labs Subtask

```
%sysl
rsubm:
    dat
    run;
endrsubmit;
```

PT	WBC	RBC
1	…	…
2	…	…
3	…	…
…	…	…

Example – Vital Signs Subtask

```
%sysl
rsubm
    da                                                    );
```

PT	SYST_BP	DIAST_BP
1	…	…
2	…	…
3	…	…
…	…	…

```
            /* Compute statistics */
        end;
        else do;
            /* Compute statistics */
        end;
    run;
endrsubmit;
```

Example – Finalization

```
waitfor _all_ task1 task2 task3;

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

signoff _all_;
```

Wait for subtasks

Example – Finalization

PT	GENDER	HT	WT	WBC	RBC	SYST_BP	DIAST_BP
1	…	…	…	…	…	…	…
2	…	…	…	…	…	…	…
3	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

```
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_;
```

Sign off all subtasks

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