



VINCI SAS/Grid Optimization

Mark Ezzo

VINCI SAS Administrator

April 11, 2012

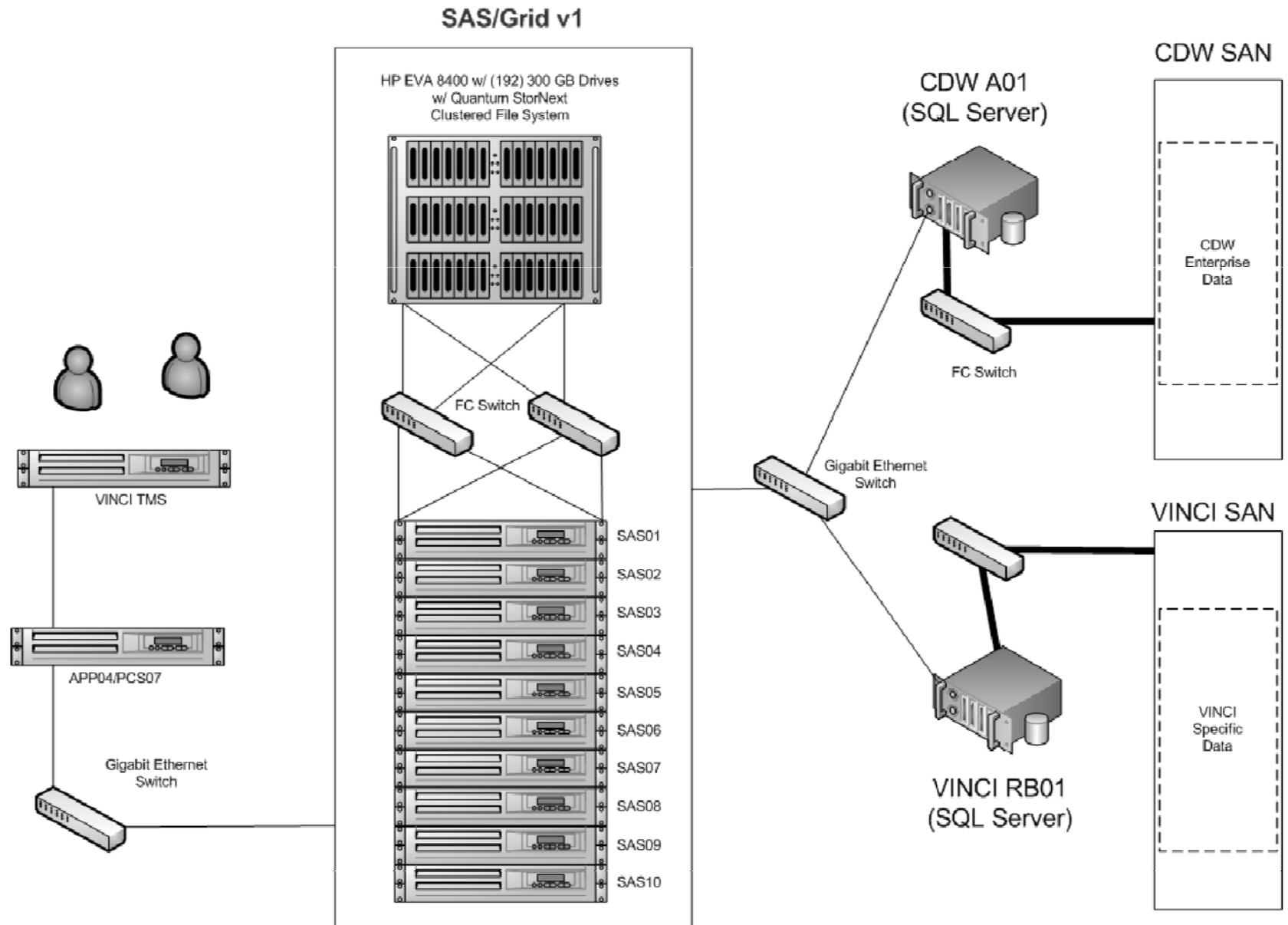
VINCI SAS/Grid Agenda

- **SAS Grid Introduction**
 - Overview
 - User Interfaces
 - Grid Enabling Existing SAS
- **Examples**
 - Base SAS
 - Enterprise Guide
- **Summary**

VA Informatics and Computing Infrastructure (VINCI) Foundations

- **VINCI Research Infrastructure**
 - A Leading – Edge Research Venue utilizing SAS and SAS/Grid
- **New Data Privacy & Security Capabilities**
 - New and improved methods to protect the privacy and security of data used in the research environment.
- **Remote Analysis Capability**
 - New methods for researchers to perform research remotely without removing data from the VINCI system.
- **Prepare for Future Data Capabilities**
 - Develop and implement methods to extract and make useable unstructured data such as text, genomic data, and images.

SAS/Grid 9.2 Windows:

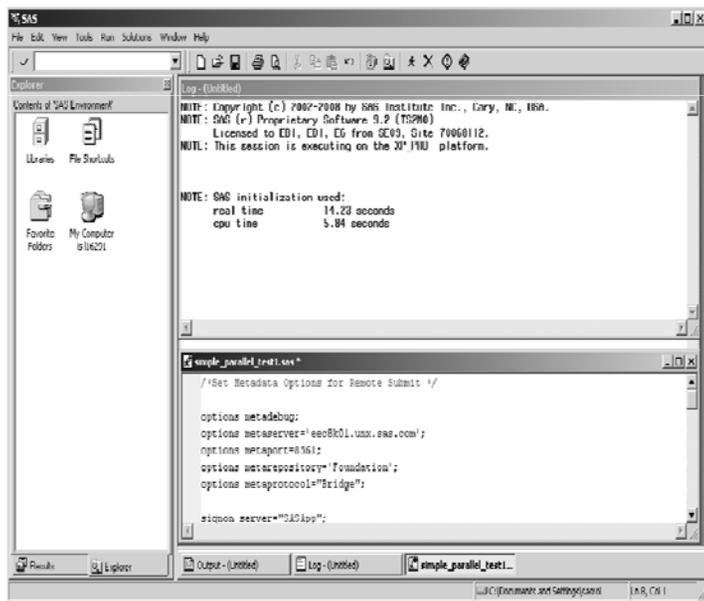


SAS/Grid Road Map

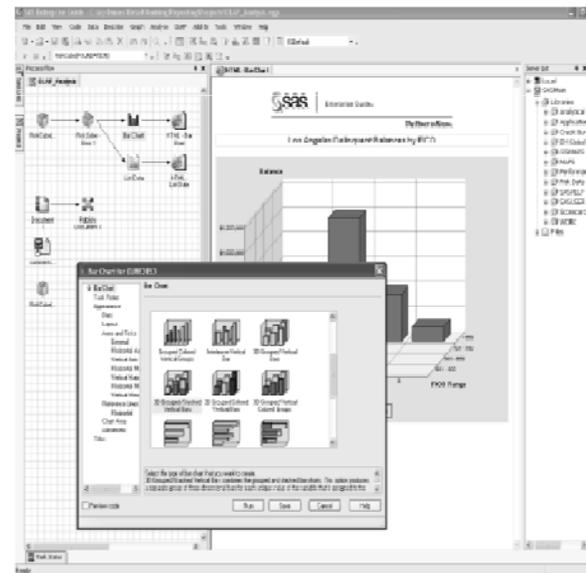
- **SAS/Grid v1.0** Now: (10) Node Windows 2008 R2 Cluster
- **SAS/Grid v2.0** Plus (6) Node RHEL Linux Cluster w/ modified clustered file system architecture

Run Windows and Linux in parallel for a period of time then decide on one platform going forward (Testing now and would like participants!)

SAS User Interfaces



Base SAS®



SAS® Enterprise Guide

New Additions

- **The SAS Add-In for Microsoft Office is a Component Object Model (COM) add-in that extends Microsoft Office by enabling you to use the power of SAS analytics and access data directly from Microsoft Excel, Word, PowerPoint, and Outlook.**
- **SAS 9.3 Stored Processes introduces stored process reports and the STP procedure. A stored process report is a new object type that contains stored process output that is cached. The output can be viewed without re-executing the stored process. PROC STP enables users to execute a stored process from a SAS program. PROC STP can be executed in an interactive, batch, or server SAS session and can even be executed by another stored process. Essentially, anyone with a Web Viewer can execute and view the results, without using SAS itself.**

VINCI SAS/Grid Summary

- **Load Balancing & Parallel Processing**
 - Support for large SAS Community
- **Reuse of existing SAS Programs**
 - Small learning curve for immediate value or submit legacy code immediately to the grid in SAS 9.3 via the GSUB Server
- **High Availability/Automatic Failover**
 - Less downtime = more research
- **Easy Administration**
 - Quickly provide help and support to users

Grid Enabling

- **SAS Display Manager and the SAS Grid Parallel Execution (Examples)**

Submit to Grid from SAS Display Manager (The “Five Lines of Code”)

- `options metaserver='dnnnn'; options
metaport=8561; %let
rc=%sysfunc(grdsvc_enable(grid,
server=SASApp)); signon grid; rsubmit
wait=no persist=no;`
- `/* insert SAS program here */`
- Most simple method, does not optimize the program for the Grid!

Key Definition for Grid Submission

- Save following to external file (c:\gpre.sas for ex.)

```
options noconnectpersist;  
options noconnectwait;  
options metaserver='dnnnnn';  
options metaport=8561;  
%let rc=%sysfunc(grdsvc_enable(grid, server=SASApp));  
signon grid;  
rsubmit;
```
- Create new DMS key definition (F12 for ex.)

```
gsubmit "%include „c:\gpre.sas“;”; rsubmit;
```

Syntax to Enable Grid

SAS Grid Computing for 9.2 and 9.3 complete documentation.

syntax (available on the SAS Sharepoint site

<http://vaww.vinci.med.va.gov/vincicentral/projectsites/SASGrid/default.aspx>)

Tell SAS to Use the Grid

```
options metaserver='metadata.sas.com';  
options metaport=8561;  
%let rc = %sysfunc(grdsvc_enable (_all_,  
    server="explicit project server"));
```

We will create a custom project server and area for each group.

SAS Grid Parallel Execution

Parallelized Workload Balancing

```
%let rc=%sysfunc(grdsvc_enable(_all_, resource=SASMain));  
signon task1;  
rsubmit task1 wait=no;  
    /* code to be remote submitted */  
endrsubmit;  
signon task2;  
rsubmit task2 wait=no;  
    /* code to be remote submitted */  
endrsubmit;  
waitfor _all_ task1 task2;  
    /* continue local execution */
```

Grid Enabling Existing SAS Program – Original Serial Program

```
libname a "/u/users/sales";  
data a.one;  
    do x = 1 to 1000000;  
        output; end; run;
```

```
libname b "/u/users/sales";  
data b.two;  
    do y = 1 to 100000;  
        output; end; run;
```

Grid Enabling Existing SAS Program – Load Balance Parallel Work Units

```
options metaserver='xxx.yyy.zzz.com';
  options metaport=8561;
  %let rc=%sysfunc(grdsvc_enable(_all_, resource=SASMain));
signon host1;
rsubmit wait=no;
  libname a "/u/users/sales";
  data a.one;
    do x = 1 to 1000000;
      output; end; run;
endrsubmit;

signon host2;
rsubmit wait=no;
  libname b "/u/users/sales";
  data b.two;
    do y = 1 to 1000000;
      output; end; run;
endrsubmit;
signoff _all_;
```

The Problem

- Lots of existing SAS programs
 - long and complex
 - Written a long time ago
 - Original author no longer available
- Advances in computing architectures and SAS products
 - multi-core and distributed systems
 - SAS metadata, SAS Enterprise Miner/Guide
- Existing programs could be more efficient
 - manual improvements time consuming and costly

The Solution

- SAS code analyzer
 - SAS procedure new in 9.2
 - executes an existing SAS program
 - analyzes job steps, input/output data and dependencies
 - records information used to enhance efficiency and manageability of the program

A Sample Invocation

Live
Example!

A Testimony to the Result

I have reused the program at least 10 times, from cohort with unique patients as low as 400,000 to as high as 5 million. Each patient may have multiple entries, some about 50. So, the number of rows of data are millions.

I would not hesitate to vouch that the time taken is now 1/10th. A program would run for 16 hours before, now is all done in 2 hours.

Gowtham

Benefits

- Parallel execution results in accelerated run-times
- Easy to use tool
 - eliminates costly manual analysis of program flow
 - inserts necessary syntax to eliminate programming errors
 - accelerates learning curve for creating parallel programs

SAS Command-Line Grid Submission Utility

- Standalone utility that will allow user to
 - submit SAS program to grid for processing
 - display status of user's jobs on the grid
 - retrieve output from user's jobs to local directory
 - kill jobs

Advantages

- User can submit and forget (Batch Jobs)
 - no need to remain connected to process job
- User can view job output while job is running
- Allows for SAS checkpoint/restart capability
- Uses SAS Grid Manager metadata for centralized control

- NOTE - requires shared file system between client and grid

Submitting a Job

- `sasgsub -gridsubmitpgm <sas_pgm>`
 - other parameters stored in configuration file
 - `-GRIDWORK <shared_file_dir>`
 - `<metadata_connection_parameters>`
 - `-GRIDAPPSERVER <app_server_name>`
 - `[-GRIDLICENSEFILE <license_file_pathname>]`
 - `[-GRIDFILESIN <file_list>]`
 - `[-GRIDJOBNAME <job_name>]`
 - `[-GRIDJOBPTS <job_options>]`
 - `[-GRIDRESTARTOK]`
 - `[-GRIDSASOPTS <sas_options_for_job>]`
 - `[-GRIDWORKLOAD <workload_values>]`
 - `[-GRIDWORKREM <remote_shared_file_dir>]`

Example Output

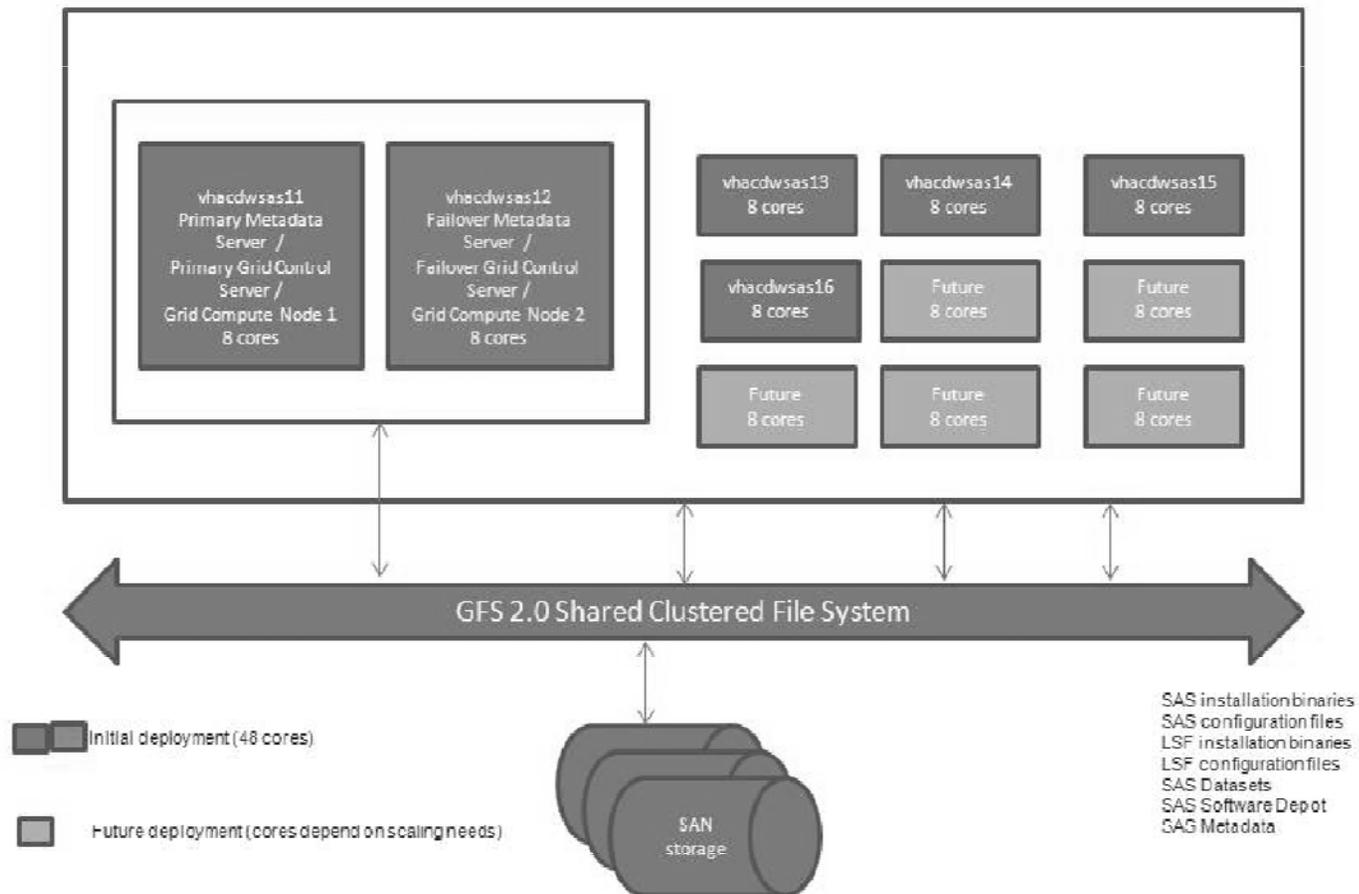
Job ID: 6772

Job directory: "/CNT/sasgsub/gridwork/sascnn1/SASGSUB-2009-03-17_14.09.52.847_testPgm"

Job log file: "/CNT/sasgsub/gridwork/sascnn1/SASGSUB-2009-03-17_14.09.52.847_testPgm/testPgm.log"

Parallel Linux 9.3 Grid

Scaling SAS 9.3 Grid Environment – Adding/Removing Grid Nodes



Good Gridding!

Thank you for attending.

**Please contact Mark Ezzo VINCI SAS
Administrator:**

Mark.Ezzo@va.gov

with any questions or comments.