

# SGE Scripting Baseline

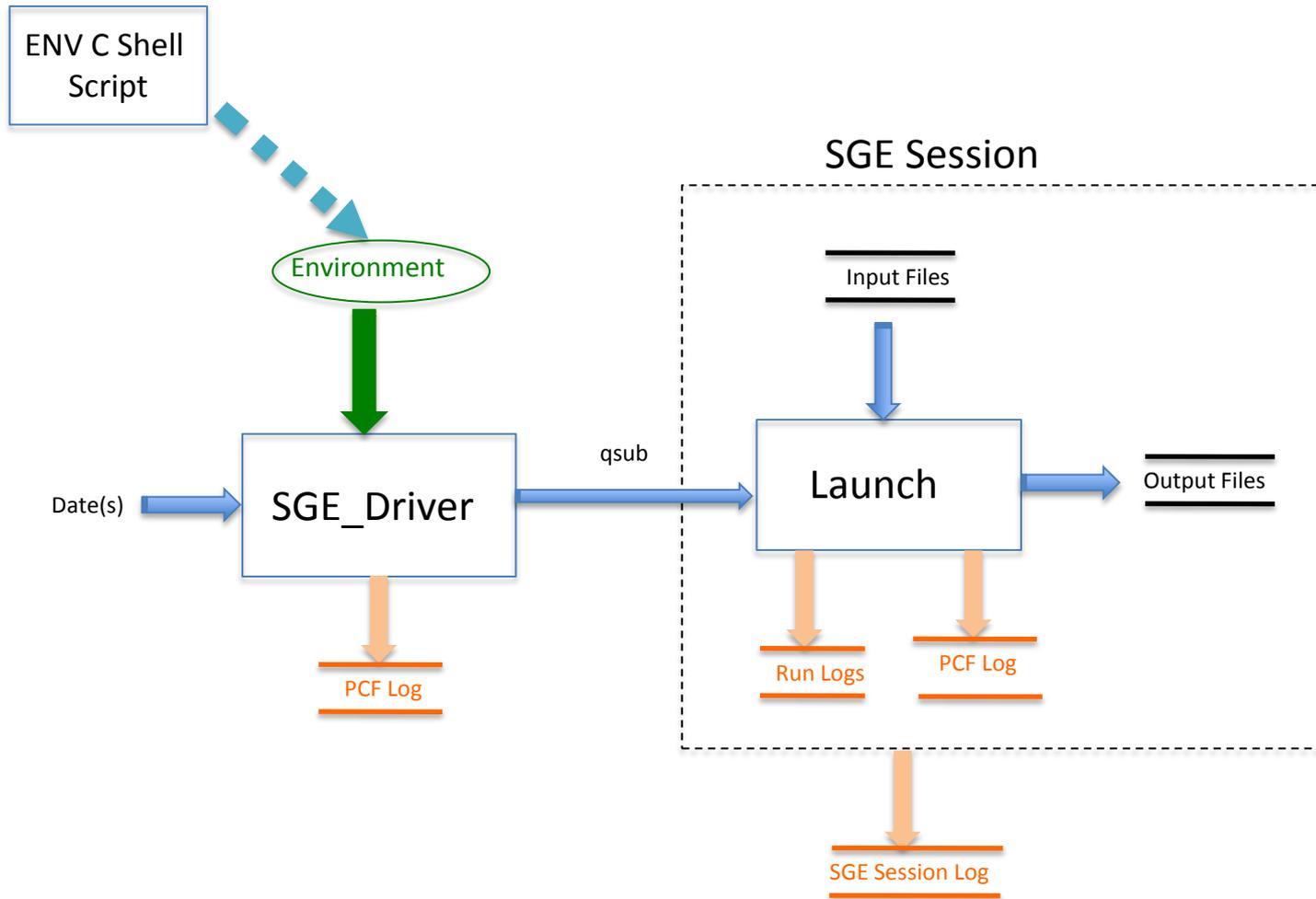
Brian Magill

7/27/2010

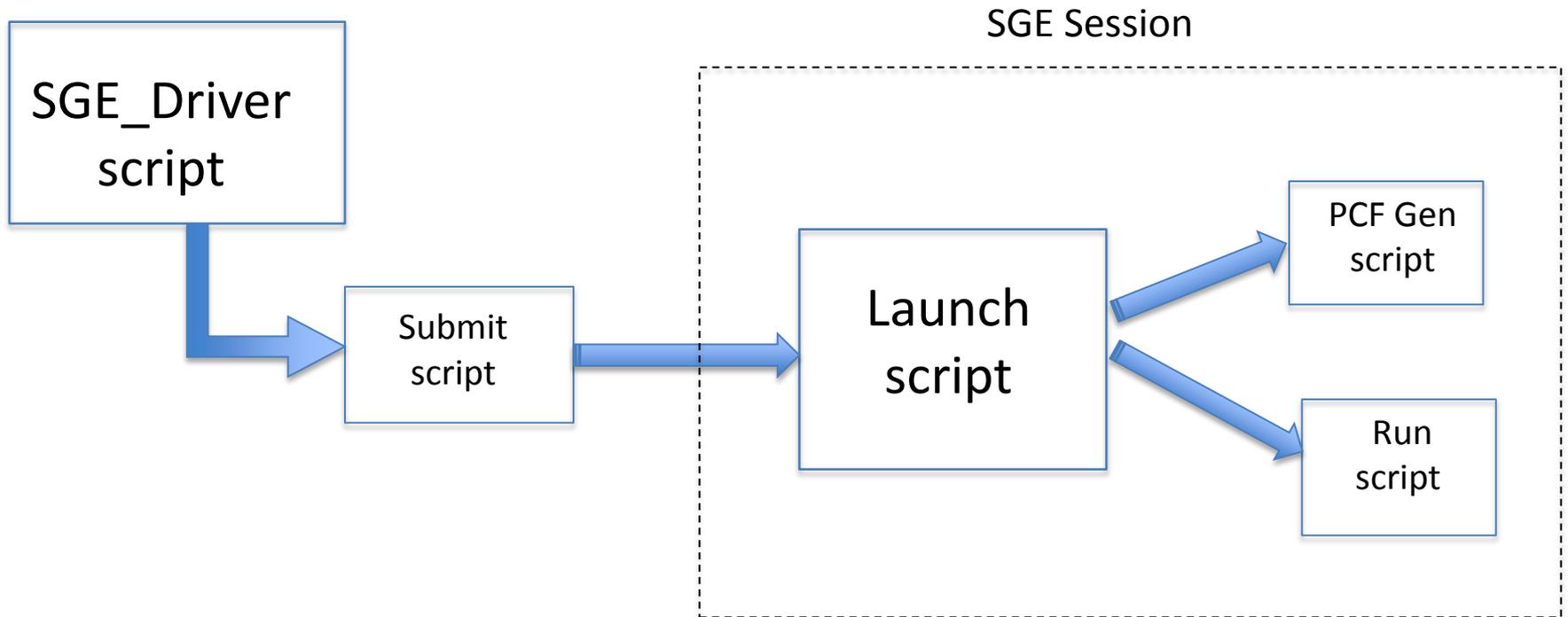
# Processing Steps

- User sources environmental script for PGE
- User enters either a single date or a range of dates
- SGE Driver script
  - Checks the input's validity
  - If a range of dates is given, verifies that this is what the user wants
  - Checks input files existence and that old output files have been deleted
  - Submits job with Launch script to SGE
- Launch script controls all remaining processing
  - Generates PCF and PCF log files
  - Creates output data files
  - Writes run logs
  - All messages that used to be printed on the screen are written to the SGE log.

# Currently How It Works



# Top Command Flow



# Recently Added Features

- Renamed SGE Wrapper script as the Launch Script
- CheckFile log created by Driver renamed PCF log
- Toolkit (Vault) Hash initialized with a subroutine. Has optional arguments for toolkit version and revision.
- Subsystem Hash also initialized with a subroutine
- Routines to check for old output files
- Optional arguments passed from Driver on to PCF generator

# Proposed Change

- Turn Submit script into a function called directly by the Driver script
- Make this function generic
  - Need to divide Subsystem Hash into Cataloguing and Subsystem Hashes
  - SGE log name needs to be determined before the subroutine is called.
  - Only value needed from subsystem hash is SGE log's directory
- Benefit: Reuse across PGE's and subsystems

# Example: Current CER1.4P1-Submit.pl

```
my %Instr = CER1_4P1_env::getHash();
my $jobName = "CER1.4P1_${Instr{SS1}} \_${Instr{PS1_0}} \_${Instr{CC1_3}} \_${run_date}";
push(@Args, "-N $jobName");
push(@Args, "-j y -o ${Instr{SGELOGdir}}");
.
.
.
push(@Args, "-v SS1=${Instr{SS1}}");
push(@Args, "-v PS1_0=${Instr{PS1_0}}");
push(@Args, "-v CC1=${Instr{CC1}}");
push(@Args, "-v CC1_3=${Instr{CC1_3}}");
push(@Args, "-v SW1=${Instr{SW1}}");
push(@Args, "-v SW1_3=${Instr{SW1_3}}");
push(@Args, "-v DATA1=${Instr{DATA1}}");
push(@Args, "-v DATA1_3=${Instr{DATA1_3}}");
push(@Args, "-v SAT=${Instr{SAT}}");
push(@Args, "-v INST=${Instr{INST}}");
.
.
$script_to_run = "CER1.4P1-Launch.pl";
$sge_command = "qsub @Args $script_to_run $run_date @inputArgs\n";
#print "$sge_command\n";
system $sge_command;
```

# Example: Proposed Submit function

```
sub submit {  
  
    my %catalog = %{{shift}};  
    my %additionalArgs = %{{shift}};  
    my ($run_date, $jobName, $sgeLogDir) = @_;  
  
    push(@Args, "-N $jobName");  
    push(@Args, "-j y -o $sgeLogDir");  
    .  
    .  
    .  
    foreach my $key(keys %catalog) {  
  
        push(@Args, "-v $key=".$catalog{$key});  
    }  
  
    .  
    .  
    .  
    my $pgeName = $catalog{'PGE'};  
    $script_to_run = "$pgeName \-Launch.pl";  
    $sge_command = "qsub @Args $script_to_run $run_date @inputArgs\n";  
    #print "$sge_command\n";  
    system $sge_command;  
}
```

# Hash Table:

- Data structure containing name to value associations
- Also known as an associative array, dictionary, or map depending upon computer language.
- Like an array, but “index” is a string instead of an integer
- Stored unordered. Not guaranteed to be contiguous in memory

# Perl example:

```
my %hash;  
$hash{'PS1_1'} = 'Edition1-CV';  
print $ENV{'HOSTTYPE'}, "\n";
```

# Proposed Hash Tables common to scripts

## Vault Hash

CERESlib Directory

Toolkit Directories

Load Libraries

## Catalog Hash

PGE Name

Sampling Strategies

Production Strategies

Configuration Codes

## Subsystem Hash

Subsystem Directory  
Paths

# Conclusion

- Minor changes to Phase I since last May's presentation.
- No substantial future changes envisioned for Phase I.

# Looking Forward

- PR Data Base – design review tentatively planned for week of August 16
- After review, build PR Data Base and applications to enter, review, and modify PR information
- Prepare an end-to-end operational concept and explore possible alternatives to current approach for handling sourced environments, as requested by Pam Rinsland on May 13.
- Expand scripting team to include members from ASDC and more seasoned DMT developers
- Script modifications to enable interfacing with data base and alternative approaches to handling sourced environments will be evaluated by larger group of people than the expanded scripting team - SEC
- Team also to participate in development of a plan for an incremental implementation of script updates and actual deployment of interface with PR database
- Adopt a more iterative development --> SIT testing approach