Existing Perl/Oracle Pipeline

Daniel Flath (SLAC)

SAS J2EE Review

Nov 23, 2004

Requirements

- Handle MC, Data and be configurable to run arbitrary linked tasks
- Envisaged as the heart of the ISOC (Instrument Science Operations Center) triggering all its automated work
  - Will be in use for 10+ years
- Talks to central databases, batch system and file servers in SCS
- Must run different tasks (eg flight data; MC; re-Recon) in parallel and not choke with hundreds to thousands of queued/running jobs
- Portability would be nice – for potential use at other GLAST sites and as backup at the GSSC (Science Support Center at Goddard)

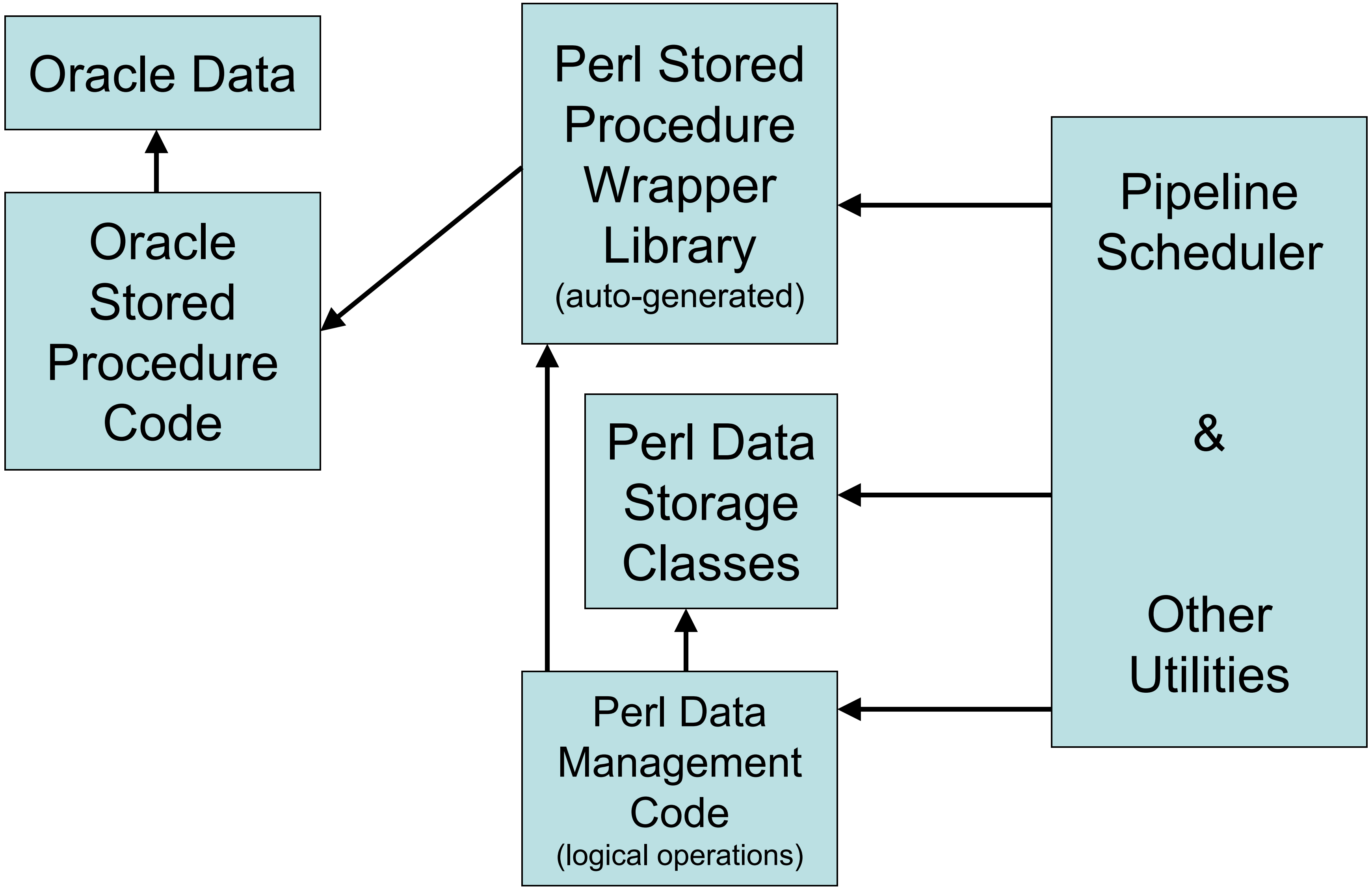
Required Functionality

- automatically process Level 0 data through reconstruction (Level 1)
- provide near real-time feedback to IOC
- facilitate the verification and generation of new calibration constants
- re-process existing data
- produce bulk Monte Carlo simulations
- backup all data that passes through

Major Components

- relational database (management system)
- database access layer
- user interface
- scheduler
- execution layer

Components


Oracle Data
Oracle Stored Procedure Code
Perl Stored Procedure Wrapper Library (auto-generated)
Pipeline Scheduler & Other Utilities
Perl Data Storage Classes
Perl Data Management Code (logical operations)

Database Overview

- Currently Oracle, using Stored Procedures for all query access
- Perl Module provided as entrypoint to stored procedures (API)
- Primarily two sets of tables:
  - Pipeline Management tables allow users to configure processing flow of a pipeline
  - Processing History tables record status of processing and data

Task Configuration Tables

- Task Table
  - Linked to by TaskProcess and Dataset Tables
- A task is comprised of 1 record in the task table and 0 or more records in the TaskProcess and Dataset tables
- The latter are related by a linked list of Read and Write flags that determine processing flow

TaskProcess and Dataset Tables

- A TaskProcess record contains all information necessary to run a job (script version, location, etc.)
- It’s links to Dataset records allow the pipeline to determine where to find input and where to write output datasets (files) for the job
- These filenames are provided to the users’ wrapper scripts at job execution

Processing History Tables

- Records in the Run table represent instances of processing for a Task
- TPInstance, and DSInstance records are in the same way analogous to TaskProcess and Dataset records

TPInstance and DSInstance Tables

- TPInstance tracks the state of processing for a single job (Execution status, CPU time, memory used, etc.)
- DSInstance records act as file descriptors for datasets used in the processing chain that is a run (File size, location on disk, file format, data type, archive location, etc.)

Major Components

- relational database (management system)
- database access layer
- user interface
- scheduler
- execution layer

Database Overview

- Currently Oracle, using Stored Procedures for all query access
- Perl Module provided as entrypoint to stored procedures (API)
- Primarily two sets of tables:
  - Pipeline Management tables allow users to configure processing flow of a pipeline
  - Processing History tables record status of processing and data

Task Configuration Tables

- Task Table
  - Linked to by TaskProcess and Dataset Tables
- A task is comprised of 1 record in the task table and 0 or more records in the TaskProcess and Dataset tables
- The latter are related by a linked list of Read and Write flags that determine processing flow

TaskProcess and Dataset Tables

- A TaskProcess record contains all information necessary to run a job (script version, location, etc.)
- It's links to Dataset records allow the pipeline to determine where to find input and where to write output datasets (files) for the job
- These filenames are provided to the users' wrapper scripts at job execution

Processing History Tables

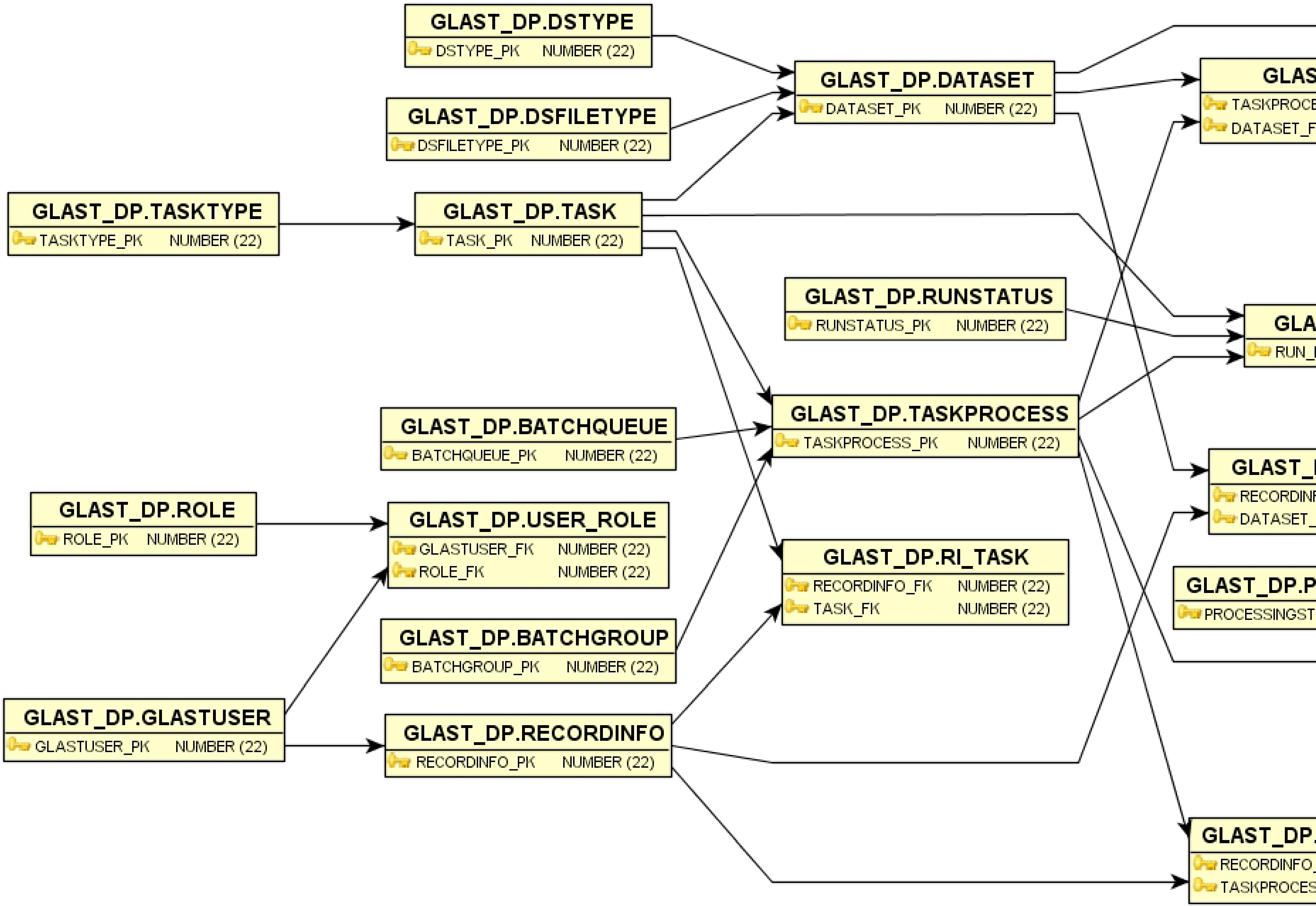
- Records in the Run table represent instances of processing for a Task
- TPInstance, and DSInstance records are in the same way analogous to TaskProcess and Dataset records

TPInstance and DSInstance Tables

- TPInstance tracks the state of processing for a single job (Execution status, CPU time, memory used, etc.)
- DSInstance records act as file descriptors for datasets used in the processing chain that is a run (File size, location on disk, file format, data type, archive location, etc.)

Current System, pros/cons

•Development is very fast •Interaction with O/S is effortless (strong support of process-mgmt, file-sys access, string manipulation.) •Strong, active user community – (free) Modules available to do *everything*. Feature additions are often simply a matter of finding appropriate libraries and gluing them together.	•Modification often more difficult than rewriting •Code often very slow running compared to non-scripted equivalent •Development tools (to my knowledge) lacking vs those available for Java (ie Debugging) •Database Stored Procedures (thousands of LOC) not portable to other RDMBSes (excepting, perhaps, PostgreSQL)


GLAST_DP.DSTYPE
DSTYPE_PK NUMBER (22)
GLAST_DP.DSFILETYPE
DSFILETYPE_PK NUMBER (22)
GLAST_DP.DATASET
DATASET_PK NUMBER (22)
GLAS
TASKPROCE
DATASET_F
GLAST_DP.TASKTYPE
TASKTYPE_PK NUMBER (22)
GLAST_DP.TASK
TASK_PK NUMBER (22)
GLAST_DP.RUNSTATUS
RUNSTATUS_PK NUMBER (22)
GLA
RUN_
GLAST_DP.BATCHQUEUE
BATCHQUEUE_PK NUMBER (22)
GLAST_DP.TASKPROCESS
TASKPROCESS_PK NUMBER (22)
GLAST_
RECORDIN
DATASET_
GLAST_DP.ROLE
ROLE_PK NUMBER (22)
GLAST_DP.USER_ROLE
GLASTUSER_FK NUMBER (22)
ROLE_FK NUMBER (22)
GLAST_DP.RI_TASK
RECORDINFO_FK NUMBER (22)
TASK_FK NUMBER (22)
GLAST_DP.P
PROCESSINGST
GLAST_DP.BATCHGROUP
BATCHGROUP_PK NUMBER (22)
GLAST_DP.GLASTUSER
GLASTUSER_PK NUMBER (22)
GLAST_DP.RECORDINFO
RECORDINFO_PK NUMBER (22)
GLAST_DP.
RECORDINFO_
TASKPROCES