CWS4DB: A Customizable Web Service for Efficient Access to Distributed Nuclear Physics Relational Databases

FY 2008 SBIR Phase II Proposal Award Number: DE-FG02-07ER84757

Mark L. Green, PI
Tech-X Corporation, Buffalo Office
Systems Integration Group
CWS4DB Project

A customizable Web Service for Efficient Access to Distributed Nuclear Physics Relational Databases
DOE NP Phase I and II – Manouchehr Farkhondeh

Tech-X: Mark L. Green (PI), Catherine L. Ruby, Krishna Kantam, Srilakshmi Ramireddy

Need: As the size of NP data grows and the collaborative nature of HENP experiments increases, the ability to access differently organized relational databases remotely, efficiently, and yet in a user-friendly and interoperable manner is becoming very important.

Partners: Jerome Lauret (STAR project at BNL), Kate Keahey (Nimbus project at ANL), Doug Olson (Open Science Grid), Alexandre Vaniachine (ATLAS project ANL/CERN)

DOE Beneficiaries: Nuclear and high energy physics communities, national laboratories, and collaborative projects

Commercial Beneficiaries: Companies requiring efficient web service access to distributed relational databases with high-level database and user APIs
Problem Identification

• The importance of this project comes from the fact that a large fraction of the ever-growing data generated by Nuclear Physics (NP) experiments is stored in relational databases. For example:
  – The BNL Relativistic Heavy Ion Collider (RHIC) supports STAR (Solenoidal Tracker at the RHIC) which composed of 52 institutions from 12 countries, with a total of 529 collaborators;
  – relational databases (such as Condition databases, Calibration databases, and Geometry databases) are heavily used in the STAR experiment;
  – while accessing data in such databases is convenient and available for local users who are familiar with a particular database, the situation becomes more complicated when the databases are distributed and heterogeneous.

• Tech-X therefore proposes a system to overcome the outlined challenges by bridging relational databases with high-level APIs through Web services.
  – In particular, the distributed and heterogeneous nature of the databases will be addressed by creating Web services in the Orbiter Federation Service Oriented Architecture (SOA), which provides mechanisms coordinating access to diversified data resources through ReST (Representational State Transfer) services, caching, authentication, and authorization.
CWS4DB Technical Objectives

• Tech-X proposes to develop a customizable Web service for efficient access to distributed NP databases. The proposed system will consist of:
  – a generic Web service for accessing arbitrary distributed relational databases,
  – a reference client implemented at the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory (BNL), for the Solenoidal Tracker at the RHIC (STAR) experiment, and
  – a tool for creation of the high-level and domain-specific clients required by particular applications.

• The Phase II objectives include:
  – Take into account what was learned from the research in Phase I and extend the CWS4DB prototype into a production-quality, load-balanced, auto-caching, grid-enabled, fault-tolerant, and on-demand system.

  Use a flexible work plan involving a separate piece of technical functionality that can be implemented in a way that can be exercised in the STAR computing environment, yet developed in a general way for application’s from other NP projects.

  The ultimate goal is to produce a set of software tools and services that can be easily adapted by the NP application developer.
CWS4DB Tasks

- **Task 1: Determine CWS4DB System and Load Balancing Additional Requirements and Properties (Tech-X & BNL)**
  - Extend the Phase I developed requirements and properties and continue prototype work with our partners.

- **Task 2: Design and Implement Tiered Deployment Capabilities (Tech-X)**
  - Develop a tiered deployment based protocol for the CWS4DB system.

- **Task 3: Design and Implement Auto-Caching Infrastructure (Tech-X & BNL)**
  - Provide a sophisticated auto-caching mechanism in order to increase the effective system performance based on work with our partners.

- **Task 4: Enable Multi-Virtual Organization Role-Based Capabilities (Tech-X)**
  - Develop the CWS4DB infrastructure required for user-friendly management and caching capabilities.

- **Task 5: Develop Dynamic On-Demand Data Resource Access (Tech-X)**
  - This on-demand service will provide a STAR MySQL database instance using the Virtual Workspaces infrastructure, Virtual Machine Computing resources, and investigate Grid deployments.
CWS4DB Tasks Continued

• **Task 6: Develop Fault Resilient Data Resource Pathways (Tech-X)**
  – Investigate eliminating a single point of failure for the STAR C++ API bound codes database query requests.

• **Task 7: Develop a Prototype On-Demand Data Resource Node (Tech-X & BNL) ★
  – Investigate and prototype the deployment of a on-demand data resource node to meet the dynamic data demands of the STAR collaboration.**

• **Task 8: Prototype Pre-Cache Capabilities for Production Job Workflow (Tech-X & BNL) ★
  – We will provide a pathway for an authenticated and authorized user upon configuration of the CWS4DB system to execute the customizable site specific test suite for pre-caching production job queries.**

• **Task 9: Develop a Customizable Site Specific Test Suite (Tech-X) ★
  – In order to deliver a high quality of service infrastructure a customizable and site specific test suite is required to validate and verify the performance and data delivery capabilities of the CWS4DB system.**
Project Management

- **Subversion Repositories**
  - Multiple readers and committers

- **Redmine, Trac, and Wiki Sites**
  - Integrates ticketing system, repositories, milestones, and roadmap

- **Eclipse Integrated Development Environment**
  - Tracks code modifications based on Redmine and Trac tickets

- **Zend Studio, Development Server, and Server**
  - Commercial PHP development and enterprise level server

- **Content Management System (Drupal)**
  - Offsite collaborator access to project information

- **Knowledgebase Manager**
  - Coding best practices, design patterns, systems and integration information

- **MacA&D Developer**
  - Analysis and Design (A&D) with requirements management and use case development

- **dotProject**
  - Open source PHP based project management software
## Project Status

### Systems Integration Group

#### Task Log

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Task Creator</th>
<th>Assigned Users</th>
<th>~Start Date</th>
<th>Duration</th>
<th>Finish Date</th>
<th>Last Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1: Determine CNS4DB System and Load Balancing Additional Requirements and Properties</td>
<td>sirmireddiy</td>
<td>mleone (100%)</td>
<td>08/15/2011 09:00 am</td>
<td>312 hours</td>
<td>09/10/2010</td>
<td>12:35 pm</td>
</tr>
<tr>
<td>Task 1a: Write Progress and Final Reports</td>
<td>sirmireddiy</td>
<td>mleone (100%)</td>
<td>08/15/2011 09:00 am</td>
<td>0 hours</td>
<td>08/14/2011</td>
<td>05:00 am</td>
</tr>
<tr>
<td>Task 1b: Write Progress and Final Reports</td>
<td>sirmireddiy</td>
<td>mleone (100%)</td>
<td>08/15/2011 09:00 am</td>
<td>0 hours</td>
<td>08/14/2011</td>
<td>05:00 am</td>
</tr>
<tr>
<td>Task 1c: Write Progress and Final Reports</td>
<td>sirmireddiy</td>
<td>mleone (100%)</td>
<td>08/15/2011 09:00 am</td>
<td>0 hours</td>
<td>08/14/2011</td>
<td>05:00 am</td>
</tr>
<tr>
<td>Task 2: Design and Implement Tiered Deployment Capabilities</td>
<td>sirmireddiy</td>
<td>mleone (100%)</td>
<td>08/15/2011 09:00 am</td>
<td>0 hours</td>
<td>08/14/2011</td>
<td>05:00 am</td>
</tr>
</tbody>
</table>

**Key:**
- `Future Task` - Started and on time
- `Should have started` - Overdue
- `Done` - Open / Close All Tasks
CWS4DB Database Query Caching and Optimization

- Network bandwidth is important and depends on the last mile normally
- Database server load is minimal
- Investigate the database service payload size
- Wrote a custom ReSTful PHP database service with a JSON (JavaScript Object Notation) payload to compare with the XML payload
CWS4DB Database Query Caching and Optimization

- Log performance data for each SQL operation
- Calculate and log JSON and XML payload size

On average over a dataset the equivalent JSON payload is 8.8 – 10.1 times smaller

In general an order of magnitude lower bandwidth loading is required with the JSON PHP service
CWS4DB Load Balancing Design
CWS4DB Proxy Implementation

Single-Cluster scenario

Cluster A

Database Node

Distributed Data Provider

Database Server

Worker Node

Execution Environment

STAR Job

Multi-Cluster scenario

Cluster A

Database Node

Distributed Data Provider

Database Server

Gatekeeper Node A

Distributed Data Provider

Gatekeeper Node B

Distributed Data Provider

Cluster B

Database Node

Distributed Data Provider

Database Server
CWS4DB Cloud On-Demand Resources

Tech-X has installed Nimbus and utilized the Nimbus client with the available science clouds in support of the STAR on-demand database service.

- The Nimbus infrastructure provided limited upload/download bandwidth consistently.
- The required STAR image is relatively large due to the size of the MySQL database.
- We investigated several ways of populating the STAR database and tested query performance with our ReSTful PHP JSON database service successfully.
- The Open Grid Services Architecture - Database Access and Integration (OGSA-DAI) XML database services could not be loaded on the Nimbus science cloud due to memory constraints.
- We are still investigating utilizing Eucalyptus and the cloud enabled MySQL database Drizzle.

Nimbus components (Keahey, ANL)
CWS4DB Summary

File Name: star.pp500.full.sql

/cache/off/
/format/XML/
/host/local/
/file/tmp/testfiles/star.pp500.full.sql/
/address/http://64.240.154.24/orbiter/service/star/

Result:

Number of trials averaged: 1
Total number of queries: 6549
Total size of queries: 38,926,201 bytes
Total query time: 76.9 seconds
Total query rate: 85.1 query/second.
CWS4DB Summary

New class files and services developed to accomplish the above tasks:

-- orbiterAutoLoader.php (150)
-- OrbiterAttributeParser.class.php (147)
-- OrbiterCacheFileService.php (723)
-- OrbiterCacheManager.class.php (236)
-- OrbiterDatabaseConnection.class.php (212)
-- OrbiterErrorHandler.class.php (509)
-- OrbiterErrorHandlerMessageService.class.php (526)
-- OrbiterMailer.class.php (187)
-- OrbiterMasterSlaveDatabaseValidationService.class.php (439)
-- OrbiterQueryDbConnectionStringStarService.class.php (467)
-- OrbiterQueryDbLoadBalancerStarService.class.php (399)
-- OrbiterRestAuth.class.php (655)
-- OrbiterServiceAttributes.class.php (132)
-- OrbiterServiceLogger.class.php (234)
-- OrbiterStarQueryService.class.php (530)
-- OrbiterStarSimulatorService.php (489)

Services developed:
-- OrbiterCacheFileService.php
-- OrbiterQueryDbConnectionStringStarService.php
-- OrbiterQueryDbLoadBalancerStarService.php
-- OrbiterQueryDBService.php
-- OrbiterStarQueryService.php
-- OrbiterStarSimulatorService.php

Unit Test scripts developed:

-- OrbiterAttributeParserStubTest.php
-- OrbiterAttributeParserTest.php
-- OrbiterServiceTestSuite.php
-- OrbiterAutoLoaderTest.php
-- OrbiterCacheManagerFileTest.php
-- OrbiterCacheManagerTest.php
-- OrbiterDatabaseConnectionMasterTest.php
-- OrbiterDatabaseConnectionSlaveTest.php
-- OrbiterDatabaseConnectionTest.php
-- OrbiterDataProcessManagerTest.php
-- OrbiterErrorHandlerEmailTest.php
-- OrbiterErrorHandlerErrorDetailedTest.php
-- OrbiterErrorHandlerLogTest.php
-- OrbiterErrorHandlerNotifyTest.php
-- OrbiterErrorHandlerSampleTest.php
-- OrbiterErrorHandlerShowContentsOutputTest.php
-- OrbiterMailerTest.php
-- OrbiterQueryDbLoadBalancerStarServiceTest.php
-- OrbiterRestAuthTest.php
-- Services_JSONTest.php
CWS4DB Continuous Integration
CWS4DB API
Documentation

Star: OrbiterQueryDbLoadBalancerStarService

[Index] [Star classes] [Star elements] [All elements] [Errors]

Class OrbiterQueryDbLoadBalancerStarService

This Service is responsible for load balancing the query and other databases. Also used for updating the database rank and status.

The Service response will return connection strings of the type given by the user based on the rank and status.

Brief example of use:

```php
// Create an instance of OrbiterQueryDbLoadBalancerStarService
$service = new OrbiterQueryDbLoadBalancerStarService();

// Process the service request
$service->processRequest();
```

Author(s):
Mark L. Green <mrlgreen@sxcorp.com>

Version: Release @package_version@
Copyright: 2006-2010 Tech-X Corporation. All rights reserved.
Link: http://www.sxcorp.com/
License: BSD License

Method Summary

```php
public __construct()

OrbiterQueryDbLoadBalancerStarService Constructs a new OrbiterQueryDbLoadBalancerStarService class
```
STAR Commander Implementation

The STAR experiment
at the Relativistic Heavy Ion Collider, Brookhaven National Laboratory

Commander

Query

Select a workspace

Commander stores your information in a folder called a workspace. Choose a workspace folder to use for this session.

Workspace: /Users/cruby/Commander/OrbiterWorkspace

Use this as the default and do not ask again

Cancel OK

TECH-X CORPORATION
Future Directions

- Integrate On-Demand Application Resources (O-DAR) within the Open Science Grid.
- This is a new type of OSG virtual facility that can be used for cycle scavenging usage on hardware that is idle or migrated out of a production environment and might not even have OSG stack installed.
- It can represent a lightweight method of deploying OSG worker nodes and building more capacity for scientific application usage.
- Will support NP, HEP, Neutron Science, etc.
Orbiter Federation SOA via ReSTful Services

- Orbiter Infrastructure serves capabilities via ReSTful web services
- Services are standards-based and are scalable, reusable, and extensible
- Robust security standards using access keys and private-key authentication
- Reusable to ensure consistent and reliable Quality of Service
Orbiter Multitier Portal Architecture (MPA)

- Framework for delivering capabilities to thin- and thick-clients using the Orbiter Federation ReSTful SOA
- Flexible and re-usable architecture for developing capabilities for thin web clients and thick local clients
- Comprised of four tiers:
  - Orbiter Federation SOA
    - Thin-Client Applets
  - Orbiter Pilot
    - Thin-Client Portlets
  - Orbiter Commander
    - Thick-Client Applications
  - Orbiter Collective
    - Thick-Client Eclipse IDE
Orbiter Pilot – Thin Client

- Built on top of the Orbiter Federation SOA
- Tier II of the Orbiter Multitier Portal Architecture
- Accessible to users with accounts and internet access (via a web browser)
- Build upon the services provided by the Orbiter SOA infrastructure
- Capabilities are seamlessly integrated using these well-defined ReSTful web services
Orbiter Commander – Thick Client

- Built on top of the Orbiter Federation SOA
- Integrates Orbiter Pilot
- Tier III of the Orbiter Multitier Portal Architecture
- Run locally on user workstations or personal computers
- Uses Eclipse RCP (Rich Client Platform) to deliver a robust and powerful GUI to the end user, also allows Commander to integrate with other local resources like e-mail, the file system, and local applications.
- Build upon the services provided by the Orbiter Federation SOA infrastructure
- Allows users to run complex simulations or computationally-intensive tasks on their local machines, relieving Quality of Service concerns on web service providers
Orbiter Commander – Thick Client (continued)

- Atomic capabilities are provided as *modules* that can be installed as needed from a central module repository.
- The Orbiter Federation ReSTful SOA provides robust access to diverse capabilities, such as:
  - Multi-threaded streaming downloads of repository files
  - Live status monitoring of the beam
  - Slideshows of instrument application screenshots
  - Organization of modules into “Suites”
Orbiter Collective
(future capabilities)

- Modules will be continuously added to Commander to provide new capabilities, including:
  - A *collaboratory* providing live chat and data sharing capabilities
  - Opportunistic file slicing to support the retrieval and management of very large data sets
  - Real-time and offline scientific data visualization capabilities
  - Integration with other open-source tools such as data analysis and workflow management for computational, data movement, and visualization jobs
  - Support for 3\textsuperscript{rd} party module contributions as well as user integrated applications (MPA Tier IV Orbiter Collective)
#!/usr/bin/python
import os, sys, base64, hmac, commands, time
from hashlib import sha1 as sha
from urllib import urlencode
from urllib import urlopen
from urllib import quote_plus

myhome = os.environ.get('HOME')
os.environ['TZ']='GMT'
time.tzset()

idfile = open(myhome + "/.orbiter/my.id")
ACCESS_KEY = idfile.read().strip()
idfile.close()

keyfile = open(myhome + "/.orbiter/user.key")
PRIVATE_KEY = keyfile.read()
keyfile.close()

URI = sys.argv[1]
EXPIRES = str(int(time.mktime(time.localtime(time.time()+60))))
str = URI + '/OrbiterAccessKeyId/' + ACCESS_KEY + '/Expires/' + EXPIRES
SIGNATURE = base64.b64encode(hmac.new(PRIVATE_KEY, str, sha).digest()).strip()
print urlopen(str + '/Signature/' + SIGNATURE, params).read()}
Related Publications


Related Posters


Related Presentations


Sponsored Workshop


For More Information

Contact:

Mark L. Green, Vice President, Systems Integration Group

716-204-8690

mlgreen@txcorp.com

http://www.txcorp.com