SAND2010-4676P

Hey! You Got Your DWA in My HPC

Experiences Integrating Netezza and Cray XT3

Cray Hybrid Solutions Summit

Ron Oldfield, Andy Wilson, George Davidson, Craig Ulmer

Sandia National Laboratories

Supported by NNSA/ASC Computer Science Research Foundation

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Why HPC and DWA

Increasing desire to use HPC for informatics

- Process massive amounts of data
  - Current approaches cull data before processing
  - HPC could identify global relationships in data
  - Time-series analysis to identify patterns (requires large time windows)
- Some problems have strong compute requirements
  - Eigensolves, LSA, LMSA (lots of matrix multiplies)
  - Graph algorithms
- National security interest

© Netezza Corporation

© Sandia Corporation

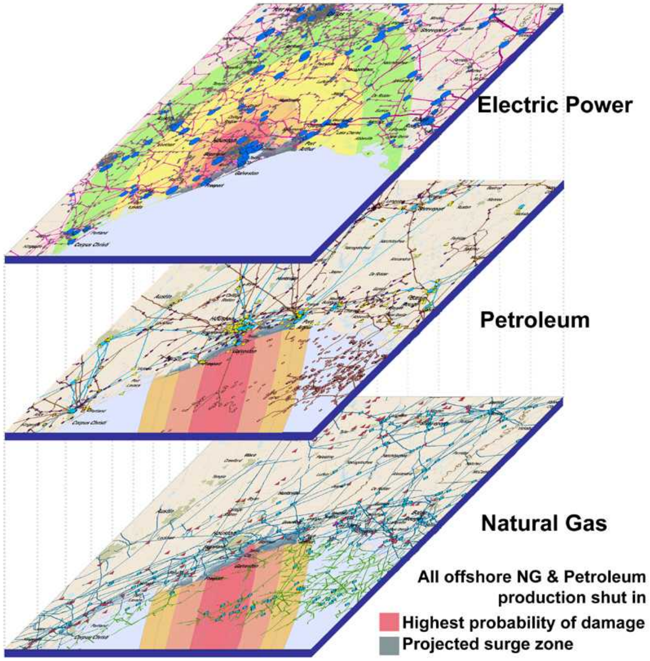
Increasing desire to use DWA in HPC-app workflow

- Post-processing sim data (e.g., economic modeling)
- I/O system metadata (fast indexing, searching)
- Feature selection/detection for “data triage” in DWA

Motivating Example: NISAC/N-ABLE

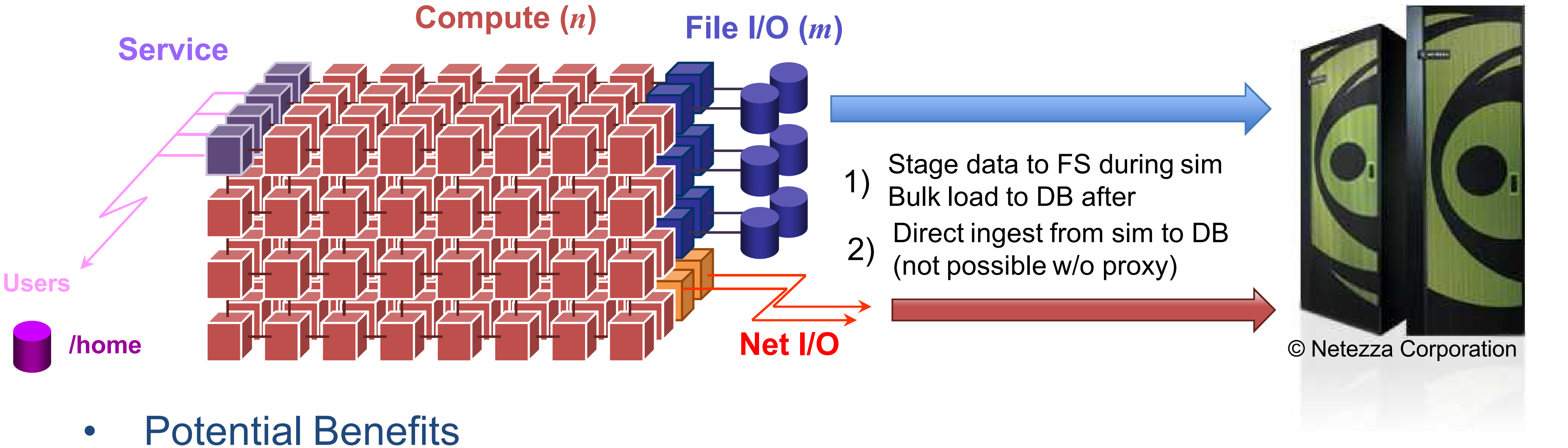
- Model economic impact of disruptions in infrastructure
  - Changes in U.S. Border Security technologies
  - Terrorist acts on commodity futures markets
  - Transportation disruptions on regional agriculture and food supply chains
  - Optimized military supply chains
  - Electric power and rail transportation disruptions on chemical supply chains

- Compute and data challenges
  - Models economy to the level of the individual firm
  - Model transactions from 10s of millions of companies
  - Simulation data ingested into DB for analysis
  - DB ingest is bottleneck (10x time to simulate data)
  - Time to solution is critical… want answers in hours


Electric Power
Petroleum
Natural Gas
All offshore NG & Petroleum production shut in
🟥 Highest probability of damage
▨ Projected surge zone

NISAC identifies potential consequences of disruptions to infrastructures and analyzes cascading impacts due to interdependencies

Integration Challenges for N-ABLE



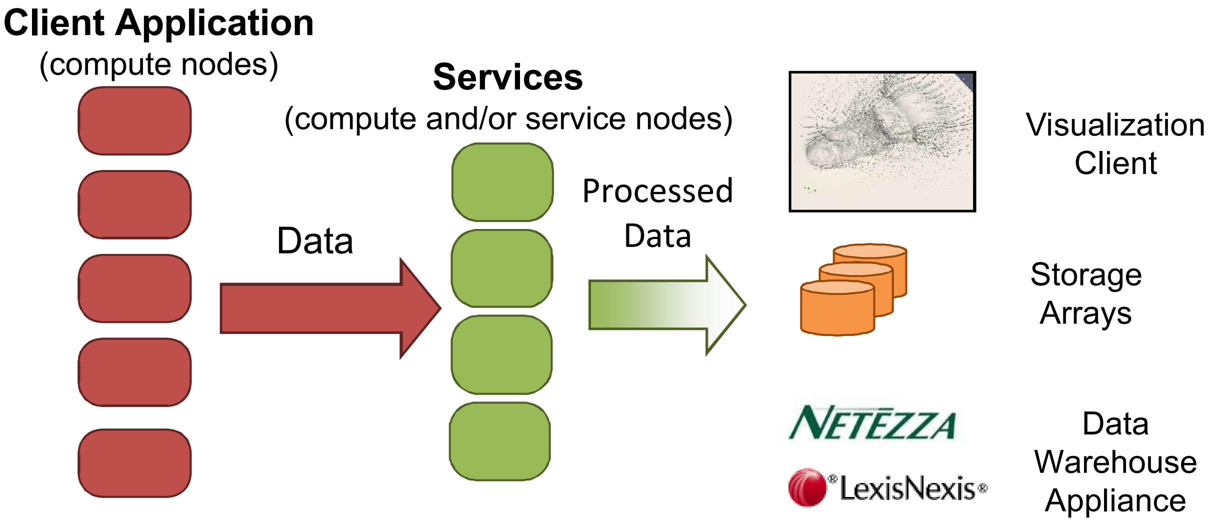
1) Stage data to FS during sim
Bulk load to DB after

2) Direct ingest from sim to DB
(not possible w/o proxy)

- Potential Benefits
  - Cray XT provides memory and compute resources for large-scale simulations
  - Netezza provides fast queries for post-analysis of data
- Software and Hardware Incompatibility
  - Specialized internal network APIs (Portals) for Cray
    - No support for standard DB interfaces (e.g., ODBC)
  - Fast network internally (2 GB/s/link), slow externally (1 Gb/s)
    - Networked integration of systems leads to I/O bottleneck

Scalable I/O Services

A Software Solution to Remote Access



Client Application
(compute nodes)

Data

Services
(compute and/or service nodes)

Processed
Data

Visualization
Client

Storage
Arrays

NETEZZA
LexisNexis

Data
Warehouse
Appliance

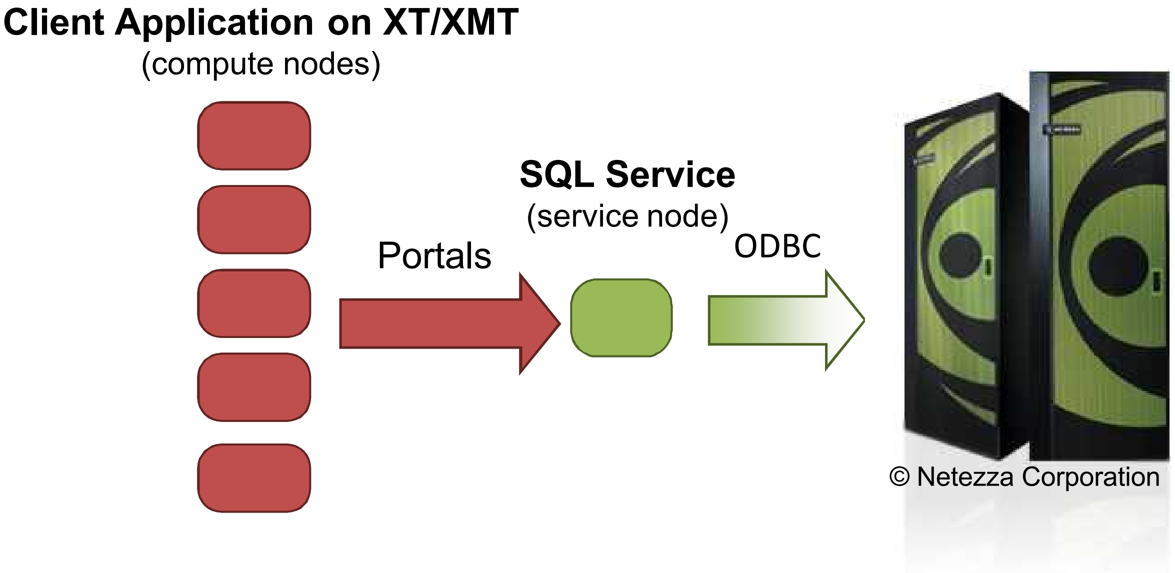
Network Scalable Service Interface (Nessie)

- Developed for the Lightweight File Systems Project
- Framework for HPC client/server development
- Designed for scalable data movement
- RPC-like API (client and server stubs)
- Implementations for Portals, InfiniBand, LUC (in development)

NEtwork-Scalable Service InterfacE

SQL Service

A Network Proxy Between Cray and Netezza



SQL Service Features

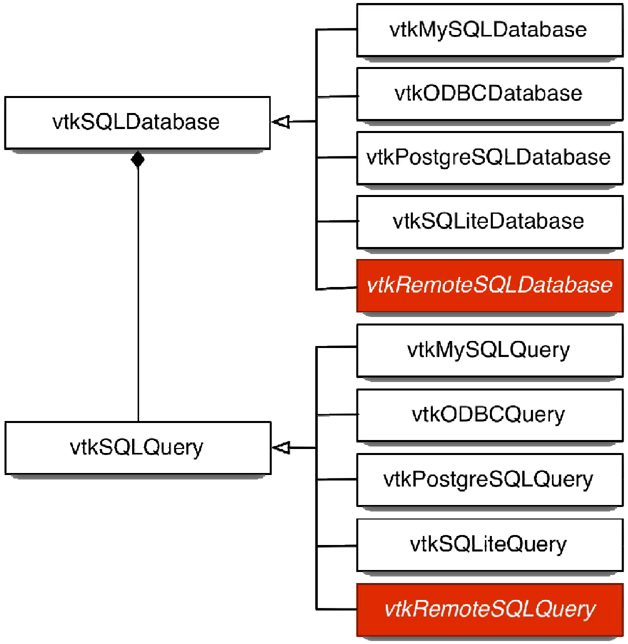
- Provides “bridge” between parallel apps and external DWA
- Runs on Red Storm/XMT network nodes
- Titan apps communicate with SQL service using Nessie (over Portals)
- Service accesses Netezza through standard interface (e.g., ODBC)

SQL Service Implementation

Client

- Compute-Node Client
  - Extensions of vtkSQL{Database,Query) classes
  - Marshal args (id, string) for remote func.

```
bool vtkRemoteSQLQuery::Execute()
{
    // handle to the RPC request
    nssi_request req;

    // XDR data structure for args and results (these get serialized)
    vtk_sql_query_execute_args args;
    vtk_sql_query_execute_res res;

    // Set the arguments for the remote Execute function.
    args.qid = this->GetRemoteQueryID();
    args.qstr = this->GetQuery();

    // Marshal and send the request to the SQL Service
    nssi_call_rpc(this->GetRemoteService(),
        VTK_SQL_QUERY_EXECUTE_OP, &args, NULL, 0, &res, &req);

    // Wait for async request to complete (no timeout used)
    nssi_wait(&req,NSSI_INFINITY);

    return res.status;
}
```



SQL Service Implementation

Server

- Server (on network node)
  - De-serialize request
  - Execute query on behalf

```
int vtk_sql_query_execute_stub(
        const nssi_remote_pid *caller,
        const vtk_sql_query_execute_args *args,
        const nssi_rma *data_addr,    // not used
        const nssi_rma *res_addr)
{
    // A data structure for the result
    vtk_sql_query_execute_res res;

    // Lookup the partner query object (stored in an STL map)
    query = query_map[args->qid];

    // Execute the query
    if (query) {
        query->SetQuery(args->qstr);
        status = query->Execute();

        res.status = status;
    }

    // Send the result of the Execute back to the client
    return nssi_send_result(VTK_SQL_QUERY_EXECUTE_OP, rc, &res, res_addr);
}
```

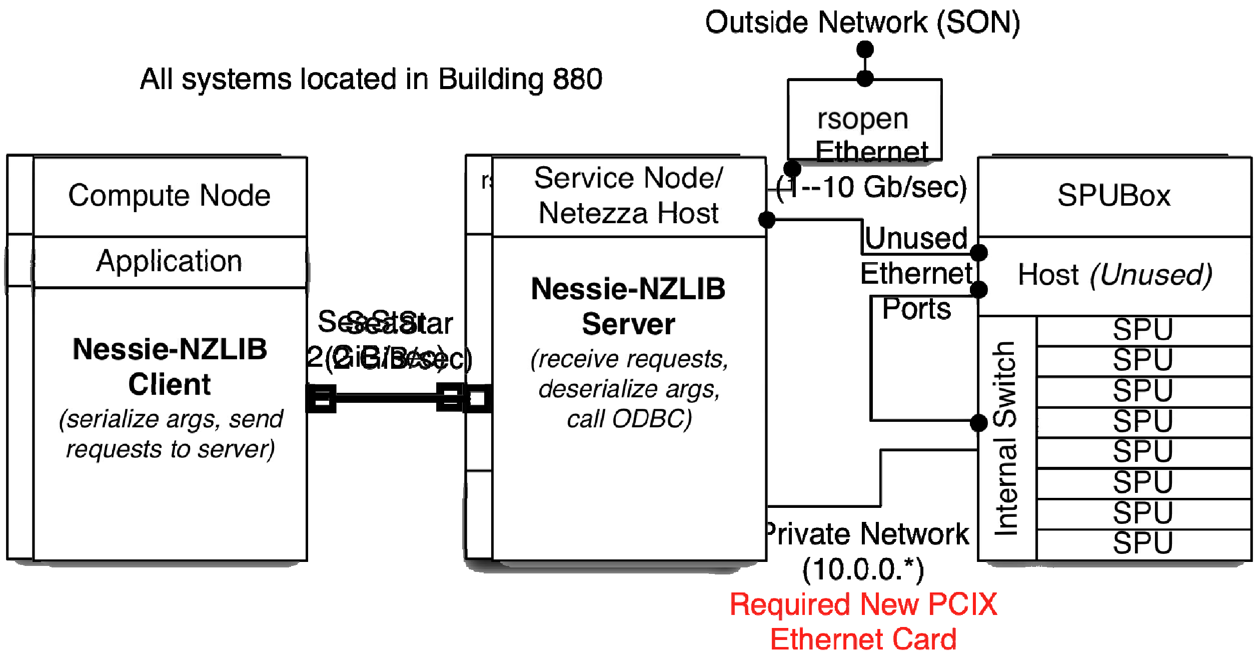
Parallel Statistics Demonstration

- Implemented Parallel Statistics Code as Demo
  - Pull one or more data sets from Netezza using SQL Service
  - Use MPI to distribute rows of query results evenly to compute nodes
  - Compute mean, variance, skewness, kurtosis, covariance, Cholesky decompositon
  - Insert results in a new table on remote Netezza using SQL Service
- Demonstration of functionality, not performance
  - Implemented minimal set of methods to demonstrate functionality
- Performance issues
  - Limitations of API (small requests)
  - ODBC implementation
  - Netezza limited to one head node (1 GigE/s max)

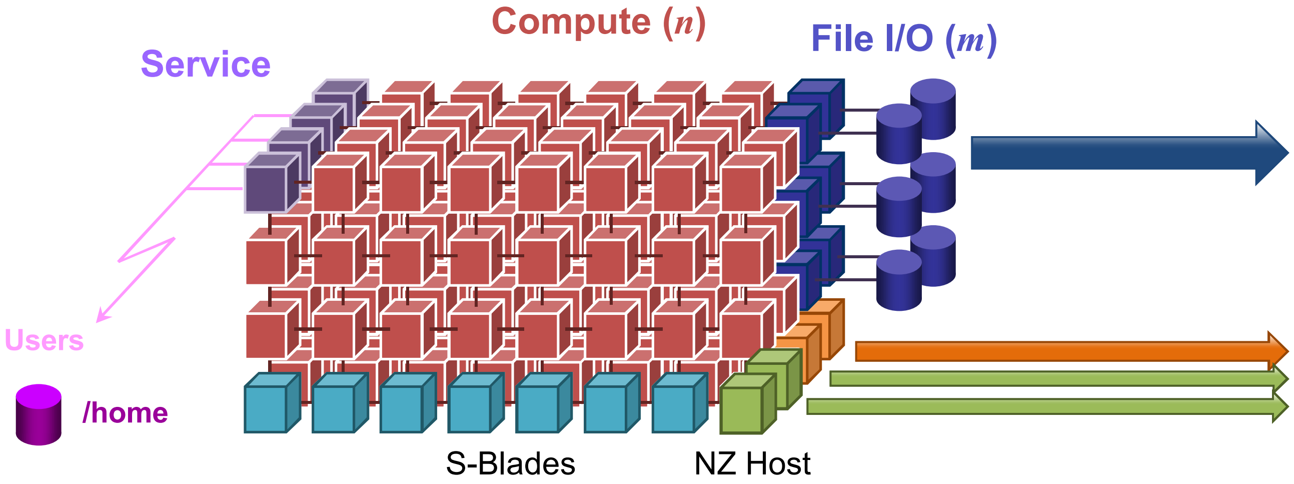
Tight Coupling of Cray and Netezza

Original Network Implementation

Move Host to Service Node



Hybrid Architecture Evolution



© Netezza Corporation

Research Questions (yet to be answered)

- What ingest rates will keep up with scientific workloads?
- Where are bottlenecks? Between host and S-BLADE?
- What software/networking infrastructure will resolve the bottlenecks?

An evolving architecture to support rapid ingest for HPC workloads

1) Stage data to FS during sim, bulk load to DB after. (post-processing)
2) SQL Server sends ODBC requests to remote Netezza (slow network to host)
3) SQL Server becomes host (fast access to host, slow to S-BLADEs)
4) Multiple service-node hosts (parallel access to back-end S-BLADEs)
5) Really wacky! Hosts and S-BLADEs on fast network (fully integrated)