HP PERFORMS LARGE SCALE DEPLOYMENT OF POSTGRES WITH ITS REMOTE SUPPORT SOFTWARE

John M. Dixon
Systems/Software Engineer
September 15, 2011
INSIGHT REMOTE SUPPORT

Postgres at Work
INSIGHT REMOTE SUPPORT

[Diagram showing the components of Insight Remote Support, including Management Console, Secure Internet Connection, Software Updates, Contract & Warranty Database, Support Center and Proactive Services, Streaming Media, Self-Help Support, Service Portals, Customer, HP/Channel Partner, Deployment and Entitlement, Contract/Warranty Reporting, Remote Monitoring, Problem Analysis, Remote Configuration Collection.]

©2011 Hewlett-Packard Company
INSIGHT REMOTE SUPPORT

– Delivers secure remote support for HP servers, storage, network, SAN environments and selected multivendor devices, 24 X 7.

– Daemon-like software that runs on your server

– Monitors the health of your servers and its connected devices
  • Configuration
  • Status
  • Serviceable events

– Notifies you and your support center
  • On-demand access to user interface via LAN
  • Alerts via email and web service protocols

– Synchronously and asynchronously collects, stores, reduces and forwards configuration and status information

– Asynchronously intercepts device events, triggering autonomous diagnostic analysis

– Analyzes events based on rules (“knowledge-based expert system”)

©2011 Hewlett-Packard Company
LARGE SCALE DEPLOYMENT

– Hosted on customer servers that are
  • In-warranty
  • Under service contract

– Estimated over 10,000 installed instances today (V5)

– Every installation contains PostgreSQL

– Insight Remote Support V7
  • Hope to expand deployment 10X over previous versions
  • I-RS V7
    – Windows
    – Single node

• I-RS V7+
  – Windows
  – Linux
  – Multi-node cluster
INSIGHT REMOTE SUPPORT

Objects of Service (Monitored Devices)

- Events
- Periodic Utility Collections
- Power on Announcements

Customer's Server

- Scheduled Collections
- Analysis Configuration Info
- Reconfiguration
- Indictment
- Connectivity Checks

Insight Remote Support

- Event Submission
- Data Collection Submission
- Entitlement requests
- Channel Partner Info

Partner

- Notification

OOS

PostgreSQL

HP

Customer

- Credentials
- Contact, Site Info
- Entitlement
CLIENT/SERVER VS. EMBEDDED

CLIENT

Insight Remote Support

SERVER

Insight Remote Support

vs.

PostgreSQL

HSQldb, Derby, SQLite, etc.
WHY NOT USE AN EMBEDDED DBMS?

Answers:
1. I-RS is multi-process
2. Development, Test, Replication
3. Need DBMS to run in separate processes for scalability

“The Other Clients”

CLIENT

Insight Remote Support

SERVER

PostgreSQL

Developer: pgAdmin etc.

Test Software

PostgreSQL Standby Servers
HOW IT’S DEPLOYED

– PostgreSQL executables image included as part of Insight RS release kit (for Windows, .msi)

– Installer drops image into Insight RS executables installation target location

– Insight RS runs as a Windows Service

– During its first-time startup, Insight RS Launches initdb with installation-specific configuration parameters
  • Non-default port
  • Non-default data folder

– After initdb completes, before Postgres startup, Insight RS programmatically edits default configuration files for security, logging preferences, etc.
  • postgresql.conf
  • pg_hba.conf
HOW IT RUNS

– Insight RS launches Postgres via pg_ctl
– Only Insight RS processes have access to login credentials
– Connections restricted
  • Localhost (127.0.0.1), or
  • For multi-node configuration, only the enumerated nodes on the LAN
– On shutdown, Insight RS stops Postgres as soon as all client processes have closed connections
– On boot, Insight RS is included in startup sequence
  • Insight RS starts up Postgres first
  • Then starts up its other components
– There is no DBA
UNDER THE HOOD

Postgres at Work
DEVELOPMENT LIFECYCLE

specs

kit test

SIT

Model

Integ. Test

UAT

Code Gen

Unit Test

Release
CLIENT APPLICATION DEVELOPMENT

CONFIGURATION

• **Currently in use:**
  - Java 1.6 SE
  - Eclipse 3.6 (Helios)
  - Postgres 9.0

• **Next up:**
  - Java 1.7 SE
  - Eclipse 3.7 (Indigo)
  - Postgres 9.1
SCHEMA DEVELOPMENT

- Product architects and designers write data specifications
  - Entities/Tables
  - Attributes/Columns
  - Data types
  - Constraints

- Specifications are captured in ERwin data modeler
- ERwin generates “Ingres-compatible” DDL (SQL create table statements, etc.) which requires very minor editing for Postgres
- ERwin exports XML for input to our Java code generator

- Code generator
  - Generates XML with embedded SQL code for iBatis (MyBatis) Object-relational Mapping
  - Generates Postgres-specific physical data access code in Java
  - Generates logical data service layer API used by hand-crafted business process code
  - Generates unit tests
DATABASE-APP ARCHITECTURE

API

Business Process
- Process Logic
- Business Objects
- Transactions

Data Service
- Security Context
- Predefined Queries

Data Access
- Java -> SQL
- JDBC Interface
- Postgres-specific

Hand Written
Generated

©2011 Hewlett-Packard Company
INsight RS Data Model

By the Numbers

- CA ERwin 7.2.8
- 3 Layers (Business Object, Logical, Physical)
- 109 Tables
- 2.7 MB of XML
- 4700 Lines of DDL
OBJECT RELATIONAL MAPPING

- Java: Object Oriented
- Postgres DBMS: Relational
- Mapping: iBatis (MyBatis)
- Automation: Code Generator

  • In-house program itself written in Java
  • Transforms xml-form Data Model into
    - Java-bean-like classes = tables
    - Accessor methods
    - XML support for iBatis
    - SQL for common SELECT, INSERT, UPDATE, DELETE statements
    - Unit Tests (over 3000)
CODE GENERATOR

– Enforces compliance with corporate standards
– Present a clear API for access to persistent data
– Maintains distinction between Logical (“Data Service”) and Physical (“Data Access”) implementation levels
– Logical level is DBMS-vendor agnostic
– Physical level supports Oracle and PostgreSQL (so far)
– Code generator uses the XML-form data model exported from ERwin as bindings for free variables in Freemarker templates to emit standardized Java and XML/SQL code
– Freemarker
UNIT TESTS

- The API seen by the Java application programmers is generated
- Every method in the API has at least one unit test
- **The unit tests also are generated, not hand-written**
  - Generated unit tests expose discrepancies between the API and the implemented schema in Postgres
  - Generated unit tests include mock forms that run at very high speed without a real database
  - Generated unit tests include standard forms that run against a lab Postgres server

```java
/**
 * Insert duplicate identity fails
 */
@Test(expected = DataIntegrityViolationException.class)
public void insertWithDuplicateIdentityFails() {
    Cntct cntct = createCntct(1);
    Cntct inserted = accessor.insert(cntct);
    accessor.insert(inserted);
}
```
ENTERPRISE SERVER TRENDS

Postgres Beyond the Cloud
ENTERPRISE SERVER TRENDS

– Key industry trends will transform the way Postgres and other database systems will be deployed and used in the enterprise of the future

• Exponential growth:
  − More nodes, more processors
  − More storage, more bandwidth
  − Capacity to process data is accelerating above 30-50% per year, driven by Moore’s Law*

• Steady growth:
  − More users, more uses, more usage
  − Consumption of information is growing steadily at only 5.4% year, limited by demographics*

DATA-INFORMATION GAP

CHALLENGE

- It’s not really information until it’s consumed
- Data volume grows exponentially
- Information consumption capacity grows linearly
  - ZB: $10^{21}$ bytes

KEY TRENDS

– Structured database systems deployment rates are now flat-to-declining

– Unstructured (not-only-SQL) systems deployment rates are growing at 50%/year
  • But structured and unstructured databases must co-exist far into the future

– Analysis and reduction of data to consumable information is becoming increasingly important to close the gap
  • Capacity to process data is growing at over 30% per year, driven by Moore’s Law*
  • Consumption of information is growing at only 5.4% year, driven by demographics*

– HP’s Insight Remote Support software is designed to anticipate and exploit these trends by moving intelligence to the edge of the network
DATA GOES UP

- Raw Data
- KB Input
- Telemetry

Edge Devices
- Workstations
- Desktops
- Laptops
- Tablets
- Smartphones
- Sensors

Low-end, Local Servers
- Consumer PC-as-Server
- Microserver
- Tower Server
- 2/3 of all processing here

Mid-/Hi-end Servers/Clouds
- Rack servers
- Blade servers
- Storage systems
- Data centers

- Reduced Data
- Queries

- Transactions
- App & Web Services
INFORMATION COMES DOWN

Mid-/Hi-end Servers/Clouds
- Centralized Database Servers
- Critical App Servers
- Internet-facing Web Servers

Low-end Servers
- Local Database Servers
- Local App Servers
- Local Web Servers

Edge Devices
- User display updates
- Device control actions
NETWORK LAGS CUT PERFORMANCE

CONVERGENCE

Enterprise Private Cloud

Site A

Site Z

Dept. 1

Dept. 2

Staff 1

Staff 500

DIFFUSION
CONVERGENCE AND DIFFUSION

- **Convergence**: Trend toward consolidation of computing resources into private, public and hybrid clouds
  - Clouds are implemented as highly integrated data centers with automated, on-demand provisioning of processing power, storage space, application services, etc., to users that may be local or geographically remote
  - Cloud infrastructures employ virtualization technology to provide encapsulation and isolation of resources allocated to each user

- **Diffusion**: Trend toward decentralization and increasingly granulated and optimized, context-aware distribution of *information* to the edge devices in the user’s hands or embedded within computational appliances such as local virtual servers, storage subsystems, etc.
  - Enterprise applications, like consumer applications, are being refactored and deployed onto tree-like, multi-tier networks with *intermittently connected hand-held devices* (e.g. laptops, tablets and smartphones as opposed to desktops and servers) increasingly dominating the outer edges
  - Edge devices require significant *local* storage, processing and communication capacity to be useful notwithstanding their ultimate dependence on the cloud back-end
OPPORTUNITIES AT THE EDGE

Postgres Beyond the Cloud
OPPORTUNITIES AT THE EDGE

– Note the trend toward **increasing numbers** and **increasing power** of edge devices

– **80% of information stored on these devices is likely to be unstructured**, but this does not mean it’s disorganized. Relational databases have an important and growing role here to control access to unstructured data

– Edge devices’ data-network communication is primarily hierarchical: converging up the chain to a central server; then diffusing back down again to edge devices

– Edge devices for the most part do not communicate with each other peer-to-peer, but the potential is there for them to do so

– **The key is to invert the client/server relationship**, making edge devices not just clients of centralized systems but also servers to their peers, enabling mutual access to information caches

– As a client/server database solution, **Postgres has the potential to support high-availability and high-scalability solutions at the leading edge** – the rapidly expanding domain of edge devices.
OPPORTUNITIES AT THE EDGE

- Processing power
- Bandwidth
- Storage capacity
- *How to exploit?*
EXPLOIT EDGE PROCESSING POWER

– To reduce unstructured data to
  • Metadata for queries
  • Structured data for relational manipulation

– To validate and compress data
  • Reject invalid data
  • Discard irrelevant data
  • Expunge expired data
  • De-dup redundant data
  • Compress the remainder for storage and transmission
    – Use metadata for queries on compressed data

– To transform data into information
  • Analyze raw data to extract relevant facts and near-facts (propositions)
  • Move analytical intelligence as close as possible to data source so that data can be reduced to essential, useful information as soon as possible in the data flow
  • Use intelligent data reduction to reduce brittleness
EXPLOIT EDGE BANDWIDTH

- **Key idea:** Whenever possible, migrate the most valuable information predictively and pro-actively to edges where it most likely will be consumed

- **What information has the highest value?**
  - Value = Utility X Usage
  - Improve value by
    - Constructing information so that it is as useful as possible. Omit irrelevant stuff.
    - Increase number of users who are able to take advantage of information
  - Utility = Usefulness X Accessibility / Latency
  - Improve Utility by
    - Increasing accessibility (for example, improving user interface)
    - Reducing latency (for example, minimizing transaction time)

- **Who will use the information?**
  - Make them aware of availability
  - Use broadcast techniques to transmit to multiple concurrent users

- **Where is the information most likely to be used?**
  - Transmit information to sites where likely to used *in advance of need*

- **When is the information most likely to be needed?**
  - Schedule transmission ahead of use.
  - Expunge information from use sites after need has passed
EXPLOIT EDGE STORAGE CAPACITY

— “Cache” information locally
  • For example, a Postgres “database cluster” on the local node could be a “cache” containing parts of a globally distributed enterprise database
  • Local cache hits avoid network access
  • Local cache misses invoke queries to remote peers for cache updates
  • Writes (inserts, updates, deletes) to local cache trigger
    - Low-bandwidth broadcast-transmission of cache-invalidation messages to remote peers
    - Remote caches may not really need to be updated until next query-cache-miss event
    - Remote caches could be updated proactively to reduce latency of later queries
    - Conflict resolution protocols must be carefully designed and tested
    - Otherwise synchronous commits may be used selectively where conflicts cannot be tolerated (at expense of performance)

— Pre-render and store information for quick access
  • Anticipate queries
  • Execute and collect query results in advance of need
  • Format results for display or input to next stage of processing
  • Store for quick recall
TAKEAWAY: POSTGRES CAN BE DEPLOYED WITH CLIENT APPLICATIONS

- If client applications can benefit from the presence of a “local” (client-side) database server, PostgreSQL can be deployed as an application “component” as we do with Insight Remote Support
  - Binaries can take up less than 30 MB
  - Unneeded utilities, docs, and other files can be omitted from distribution
- The local server can function as a persistent cache, improving performance on the client and reducing network load
- The local server can permit operations to continue while network connection to central server is down
- The local server can replicate to peers or to a centralized primary server for redundancy
  - Not everything has to be replicated (separate databases)
  - Critical things can be replicated synchronously (9.1)
FOR MORE INFORMATION

– HP Insight Remote Support Software

– HP Global IT Services

– HP Products and Services

THANK YOU

johndixon@hp.com