



# Performance Suite Migration and Merge Lessons

Preparing for System 9

Kenneth E. Kane, Jr.  
WHITTMANHART  
kkane@whittmanhart.com

September 22, 2006



# Agenda

- Background
- Problem Analysis and Definition
- Planning
- Performance Lessons
- Conclusion / Q&A





# WHITTMANHART: What We Do

**OUR MISSION:** Enable each client to turn their vision into reality

**WHITTMANHART** is a privately-held, premier consulting firm providing integrated solutions in digital communications, process improvement and enabling technologies.

Our service model focuses on delivering rapid business value through pragmatic execution, aligning our services with client-specific ROI objectives, providing well-defined, component-based solution offerings, and backing our work with a **100% unconditional guarantee**.

**WHITTMANHART** offers solutions from two distinct, but highly collaborative divisions:

## WHITTMANHART | interactive

Helping our clients create more compelling and relevant interactive experiences.

- eCommerce
- Online Marketing
- Website development

## WHITTMANHART | performance management

Helping our clients more effectively leverage enabling technologies to achieve operational excellence.

- Financial management
- Business Intelligence
- Data warehouse solutions



# We're Growing in Michigan !



- **WHITTMANHART** is currently doing work with these Michigan organizations:

- ✓ **Stryker Corporation** (\$4.8B)

- Hyperion Enterprise to FM implementation

stryker®

- ✓ **Dura Automotive** (\$2.3B)

- Hyperion Planning implementation

**DURA**  
Automotive Systems

- ✓ **Compuware Corporation** (\$1.2B)

- Hyperion Planning implementation

COMPUWARE®

- ✓ **Champion Enterprises** (\$1.2B)

- Hyperion Planning implementation

 **CHAMPION**  
World's Largest Homebuilder

- ✓ **Comerica Bank** (\$3.6B)

- Hyperion Business Intelligence project

Comerica Bank

- ✓ **Pulte Corporation** (\$14.6B)

- Oracle 11i E-Business implementation

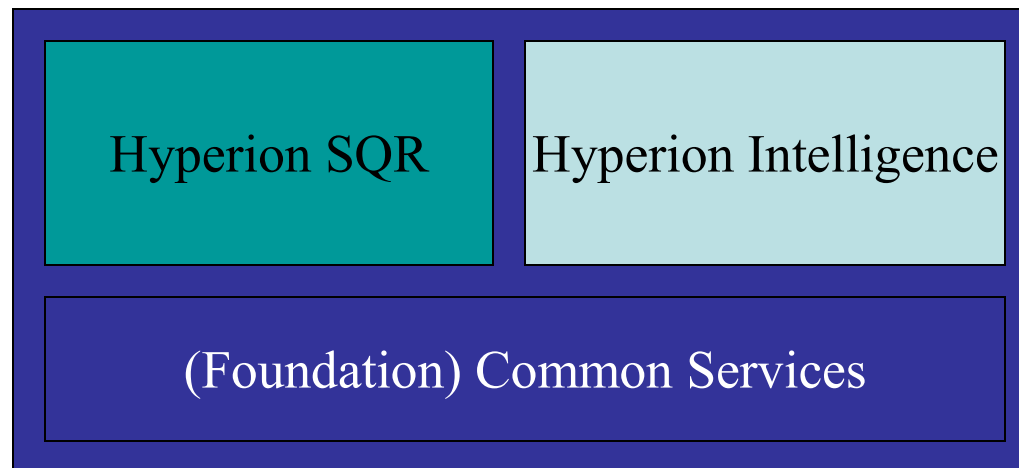
 **Pulte**  
Homes



## What is Hyperion Performance Suite?

- Integration of Hyperion SQR and Hyperion Intelligence into a **common framework**
- **Manages** creation and delivery of dashboards, queries, reports, analyses and other content

### Hyperion Performance Suite





# Issues facing business organizations

Control costs

Prepare for growth

Leverage existing IT investments



React quickly to change

Create efficient and agile IT infrastructure

Provide support for faster time-to-market products and services

Demonstrate support of business initiatives



# Hyperion Performance Suite

Hyperion Intelligence  
User-driven  
Query,  
Analysis &  
Dashboarding



Hyperion SQR  
Most Scalable  
Enterprise  
Reporting



Hyperion  
Performance  
Suite  
Easy to Use,  
Powerful  
Query, Reporting  
& Dashboards

Hyperion Intelligence  
Client tools: Designer,  
Dashboard Builder &  
Architect  
Server: Insight, iServer

Hyperion SQR  
Client tools: Developer  
Server: SQR, Viewer,  
Activator

Foundation  
Configurable  
Secure  
Open

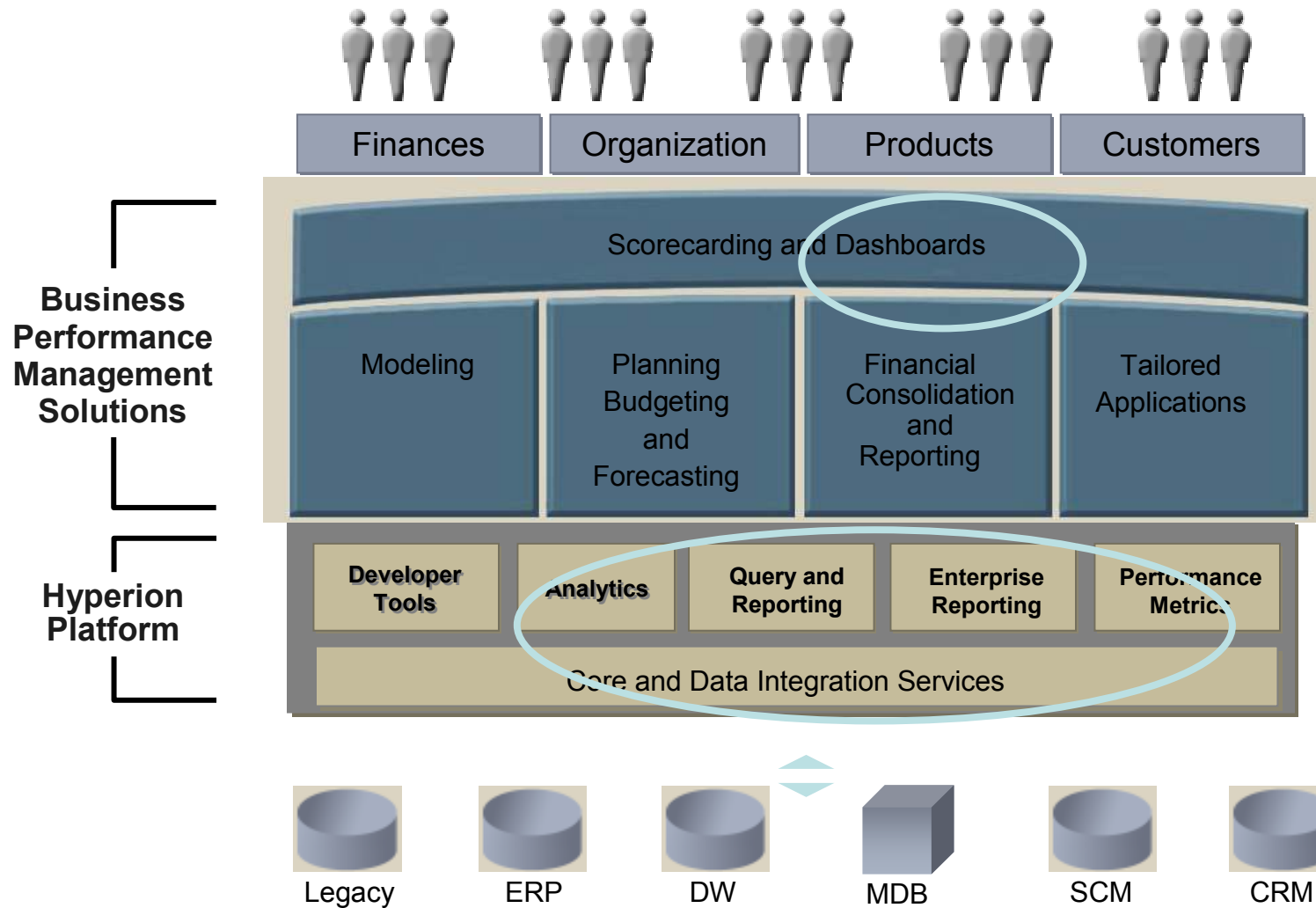


# Performance Suite History

- Originally, released as ReportMart in 1996 (SQRIBE)
- Integration with Brio Software in 1999
- Major release of version 8 in 2001
- Brio Software acquired by Hyperion Solutions in 2002
- Stabilization of Version 8 ongoing from 2002 forward
- QIQ acquired in 2004
- General availability of System 9 in 2005



# Hyperion Performance Suite





# Hyperion Performance Suite - Highlights

- **Dashboards for user-directed analysis**
  - Self-service access to graphical views of key performance indicators
  - Central distribution of dashboards for any group, role, or individual
- **Full range of Query, Reporting and Analysis Capabilities**
  - Management Reporting , End-User Reporting, Production Reporting
  - Report Delivery & Alerting
- **Zero footprint, browser-based query, reporting and analysis**
  - Distribution of dashboards, reports and analytics to a broader user base
  - Extending visibility to customers and partners outside the firewall



# Hyperion Performance Suite – Highlights (cont.)

- **Web-based administration, publishing, scheduling, and delivery**
  - Single management environment reduces administration time
  - Logging mechanism provides continuous system performance tuning
  - Enables Intelligence and SQR to benefit from common administration
- **Role-based access control**
  - Administration can be delegated so user communities can be self sufficient
  - Leverage existing investments in external authentication system (such as Active Directory Services)
- **Row level security (optional)**
  - Row level security can be enabled to provide data security
- **Unified, scalable architecture with single repository**
  - Services run as distributed, multi-threaded processes on back-end servers
  - High performance and dependability via fail-over and load balancing



# Performance Suite Components & Services

## Suite Components

*The components that make up the Performance Suite are:*

- Core Services tuned for each application
- Management Services two separate administrators
- Functional Services different tools being leveraged
- Client Components different tools being leveraged
- Web Services different versions of WAS being used
- Software Development Kit applications using Jython and Java

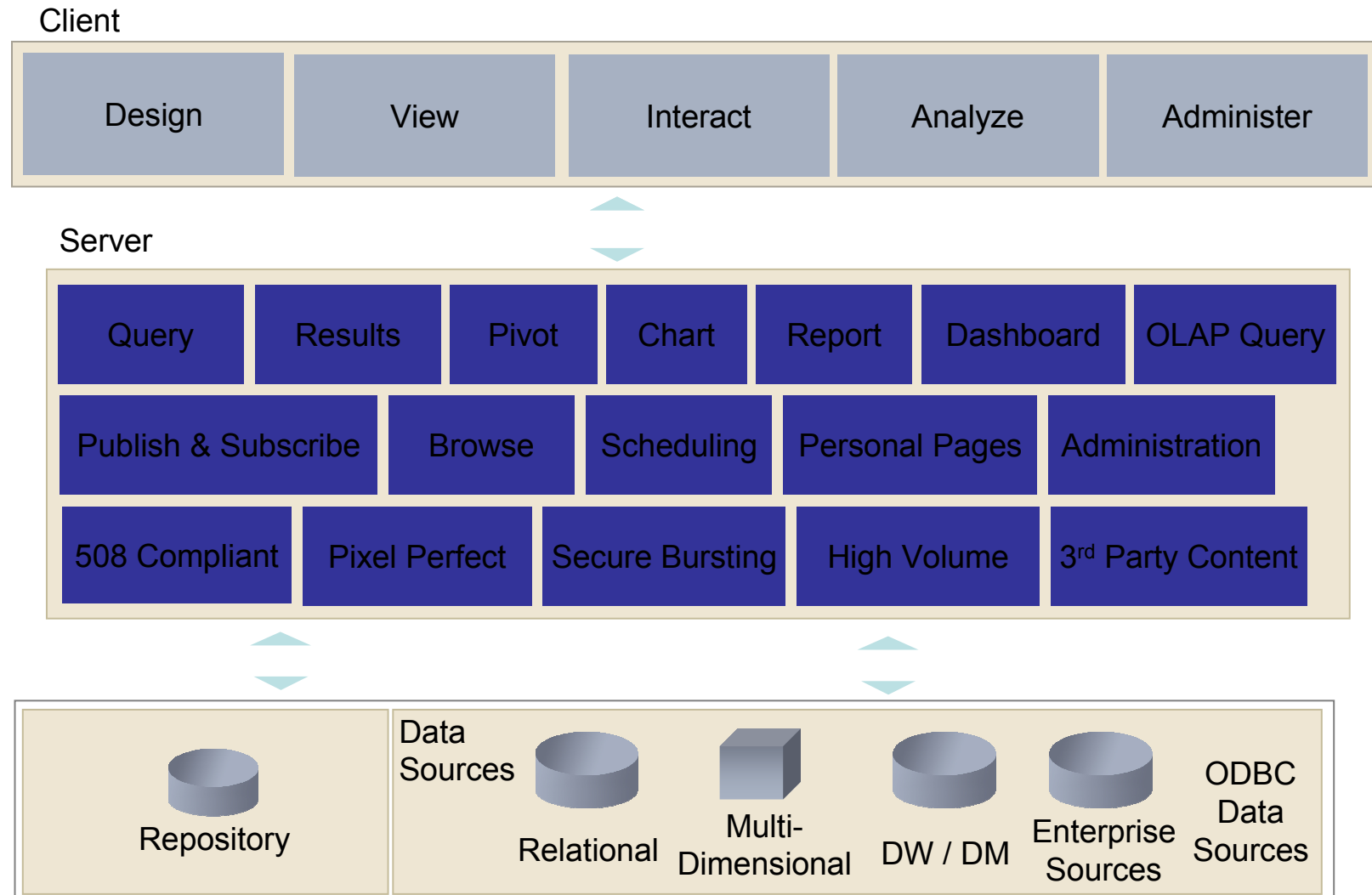
## Performance Suite Core Services

*The core services are:*

- Repository Service two separate repositories
- Publisher Service publication processes different
- Global Service Manager and Local Service Manager
- Authentication Service both applications use native authentication
- Authorization Service different security models
- Session Manager
- Service Broker one application using replication
- Name Service
- Logging Service (log4j) one application leveraging log4j

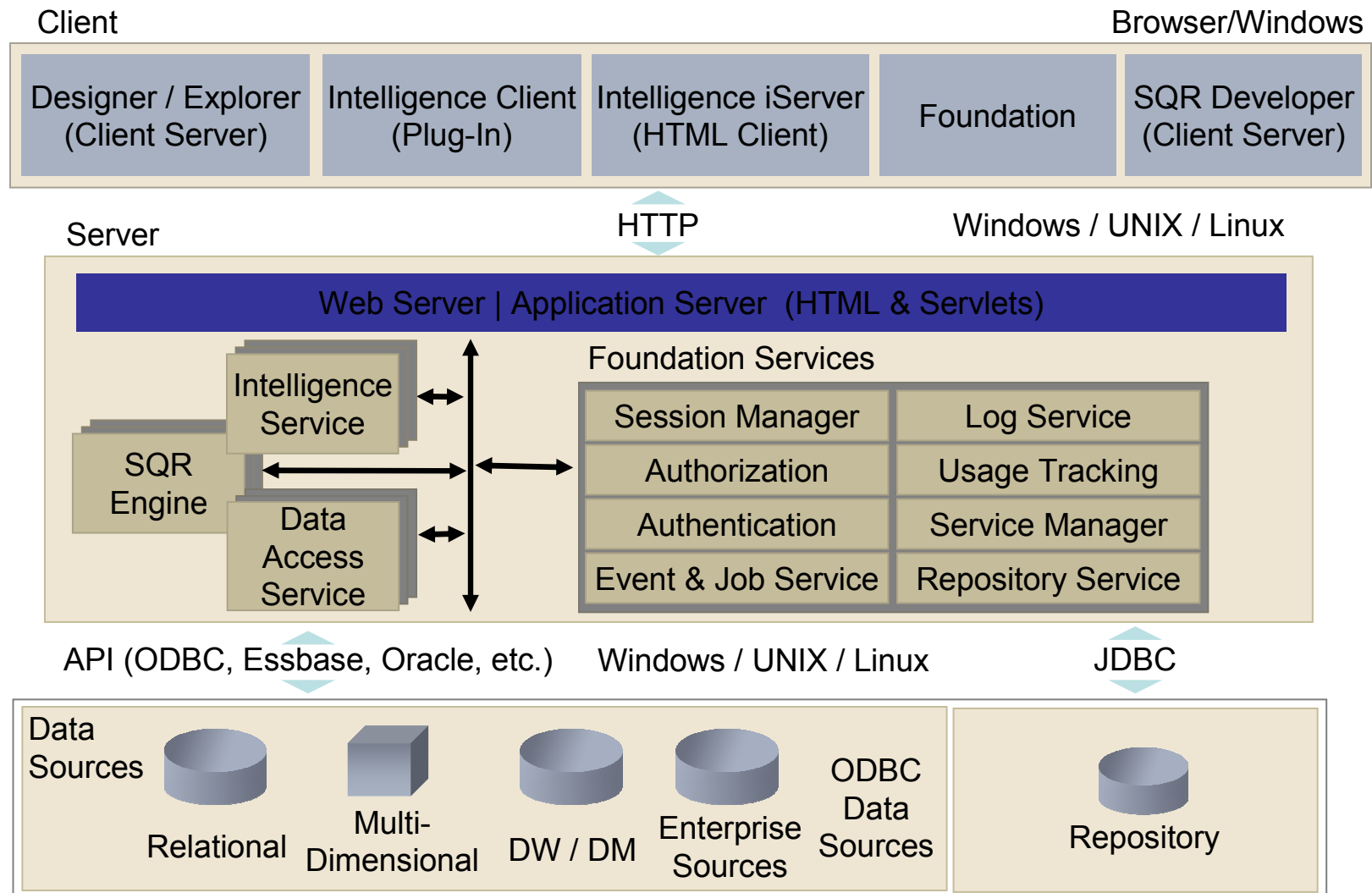


# Hyperion Performance Suite Functional Architecture





# Hyperion Performance Suite Architecture





## The Challenge and Goal

To bring together two similar business intelligence applications under one Performance Suite instance





# Business Drivers

- Hardware Consolidation
- Software Consolidation
- Future Platform Opportunities
- Improved Security Options
- Internal support benefits
- Financial Considerations

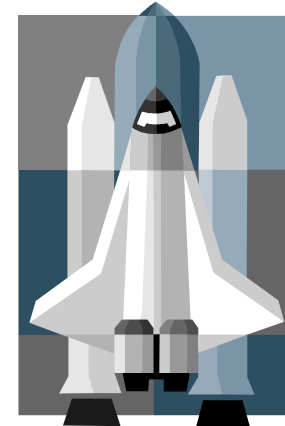


# Problem Analysis and Definition

- Double expense to maintain two separate instances
- Inconsistent versions to maintain
- One application is about to go off maintenance
- Analysis of the App 1 system showed low server utilization
- Analysis of the templates showed high degree of consistency
- User overlap of approximately 80% or more
- Common requirement to upgrade to 8.3.2 SP4 for both applications
- Both applications usage was being expanded
- And...preparing for System 9



## Getting on the launchpad...



- **App 1: Migrated to 8.2.1 SP4 from Brio ONE 7.0.3**
- **in 2003**
- **App 1: Stable and mature application to support reporting for about 2000 users**
- **App 1: Dynamic security model (changes each month)**
- **App 1: Heavily customized application (Java API) along with template and personal page changes**
- **App 2: Was using Portal 7.0.3, Brio Enterprise 6.6 and SQR 6.2 (Brio ONE)**
- **App 2: Mature application that was not being highly used due to performance issues**
- **App 2: Highly customized parameter collection and scheduling processes using Javascript, Jython and HTML template changes**
- **BOTH apps need OS, WebServer and Database Upgrades!**

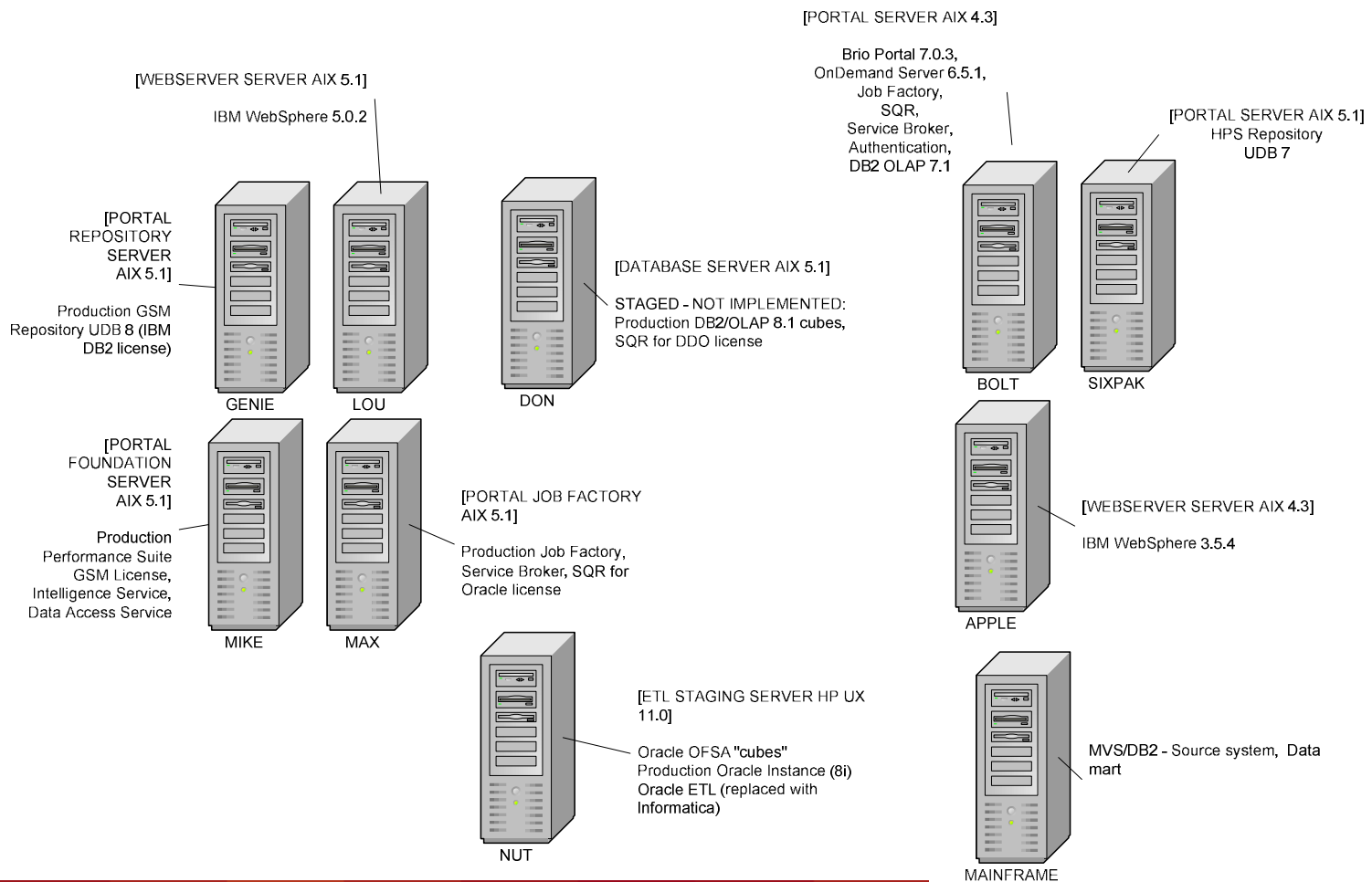


# Technical Architectures

## Production Server Architecture

### App 1 Production Environment HPS 8.2.1 SP4

### App 2 Production Environment Brio Portal 7.0.3



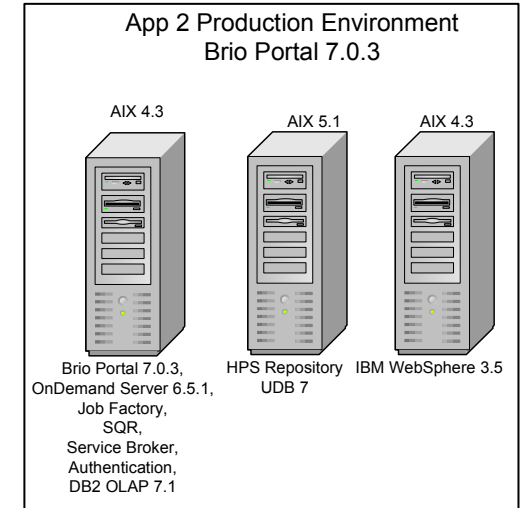
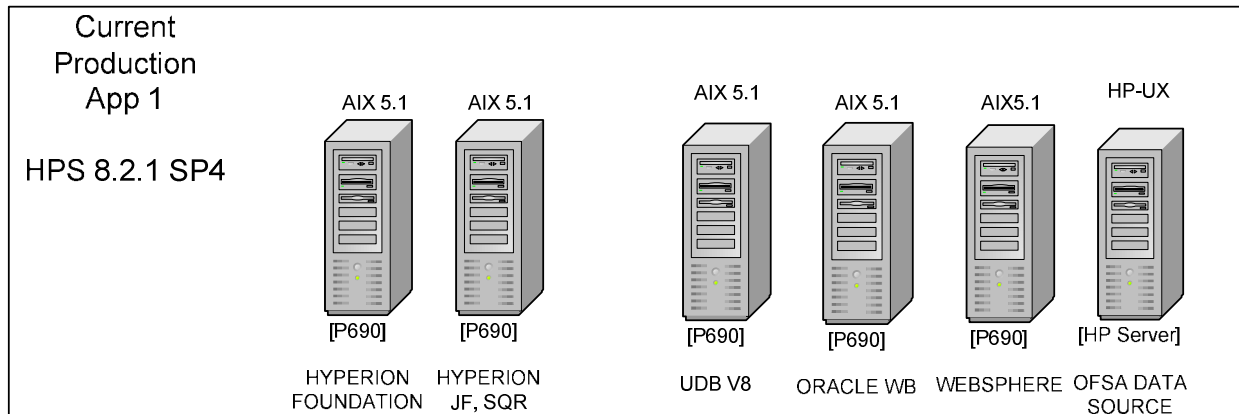


# Overview and History

- App 2 project initiated in 2000 – chronic performance problems from outset
- App 2 requirements to migrate environment to Hyperion Performance Suite 8.3.2 to alleviate performance issues and add functionality
- App 1 project defined technical architecture during sessions with Brio Software and IBM in 2002 – technical architecture built with lessons learned from App 2
- App 1 requirements to upgrade environment to 8.3.2 to leverage Hyperion Reports and Essbase functionality
- Commonality of 8.3.2 drive discussions forward for migration and merge
- Feasibility discussions to combine environments in 2004
- Proof-of-concept in 2005



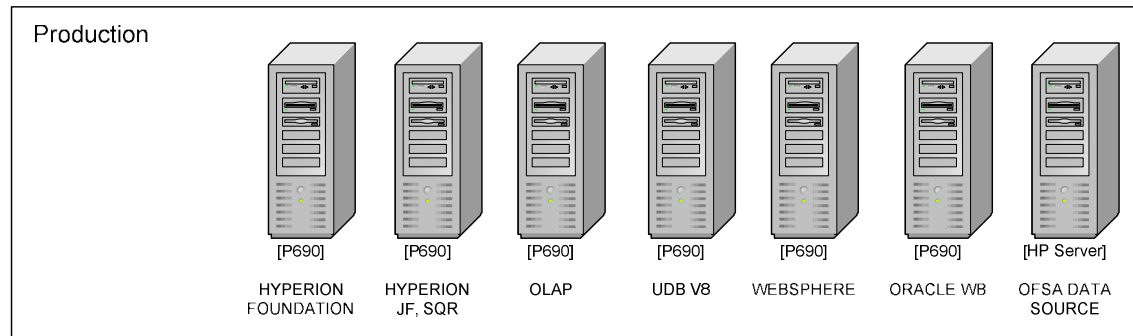
# Hardware – “Current” State



- Current State – separate environments for the two applications means separate support, license and maintenance
- Pros – each environment is autonomous
- Cons – not cost-effective, less opportunity to develop center of excellence, double-license and maintenance costs, double infrastructure support costs
- Risks – currently each client is supported separately creating support and maintenance risks, out of sync conditions with versions of the product, vendor communication risks



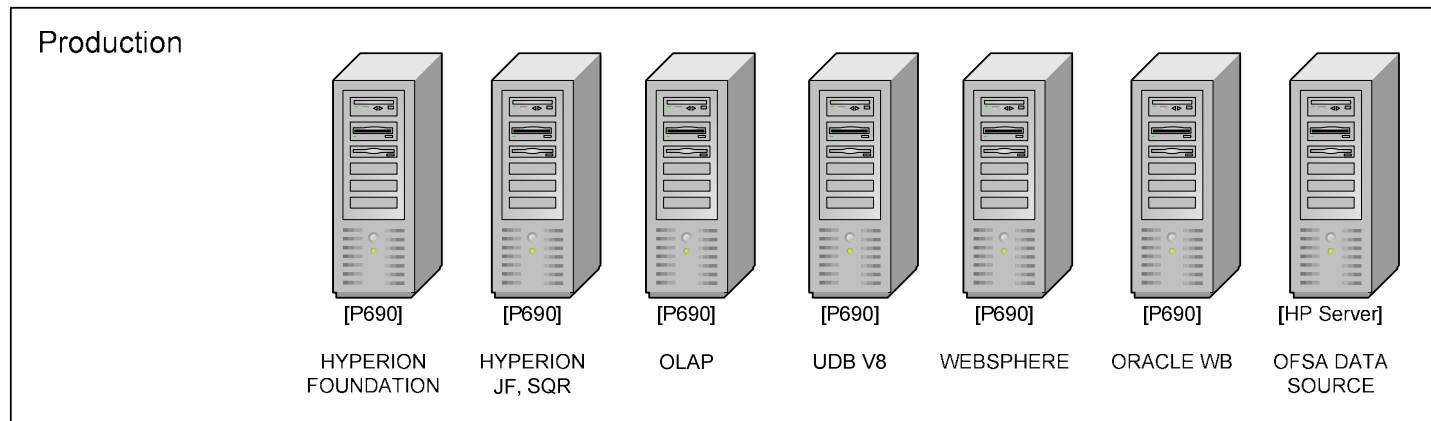
# Hardware – Coexistent Option



- Proposed state – create an environment that provides a single Performance Suite instance and allows the coexistence of the two applications
- Pros – a single maintainable environment, ability to support more departments in the future, reduced cost of licenses for Hyperion software and other supporting software, better utilization of existing hardware resources, ability to performance tune and troubleshoot because of single application, the customer can provide single interface to the vendor
- Cons – coordination between clients and IT resources must be managed
- Risks – potential for one application to cause the other to fail, potential performance risks if machine resources are constrained (future state), communication risks between support personnel



## Hardware – Shared Space Option



- Proposed state – shared space where the applications share the hardware, but are installed and maintained separately
- Pros – sharing of hardware resources improves utilization, reduction of supporting software costs, separate application reduces risk of one application affecting the other
- Cons – no cost savings on Hyperion licenses, two environments to maintain (higher maintenance costs), not building environment for future applications,
- Risks – potential version differences in each application, shared space could mean source code violations (between applications), machine contention still exists (as it does in co-existent model) but more difficult to tune because of separate applications



# Software Consolidation

- Current licensing – duplication of all products
- Proposed license – single, expanded set of CPU licenses
- Benefits
  - Cost savings
  - Reduced footprint
- Issues
  - Who manages the vendor relationship?
  - Will we lose negotiation leverage on future software sales?
  - How will maintenance costs be distributed?



# Future Platform Considerations

- Maintenance of single platform with ability to provide improved performance and service
- Opportunity for additional departments to leverage the reporting architecture
- Reusability and cost savings can be realized
- Leverage with Hyperion Solutions due to:
  - Single support point for issues and bugs
  - Platform to move to System 9
  - Licensing opportunities to expand footprint
- Internal knowledge sharing
- Positioned for future competency center development
- Single environment to ensure SOX compliance and improved security



## Financial Gains

- Software consolidation will reduce overall licensing and maintenance costs for Performance Suite (savings of approximately \$1.2m over five (5) years)
- Reduced headcount to support the combined environment
- Cost avoidance of additional hardware purchases (servers, routers, etc)
- Cost reduction of supporting software licenses (WebServer, OS, database)
- Reduced cost to support technical architecture (DBA, System Admin, Webmaster, etc)
- Single, unified environment will reduce future maintenance complexities



# Financial Gains – Server Software

## Maintenance Costs for Supporting Software

Software	Optimized Current Cost	Combined	Difference
UDB	2 licenses	1 license	1 license
DB2/OLAP	2 licenses	1 license	1 license
AIX	7 server licenses	4 licenses	3 licenses
Performance Suite	2 licenses	1 license	1 license
WebSphere	2 licenses	1 license	1 license
<b>Totals</b>	<b>15 licenses</b>	<b>8 licenses</b>	<b>7 licenses</b>



# Project Planning

- Creation of project plan to move to 8.3.2 SP4
  - 8.3.2 was chosen due to documented performance improvements
  - 8.2.1 was stable
- Risk mitigation
  - Identified ownership risks
  - Managing peak period usage
  - Performance considerations of both apps running technical architecture
  - Managing integration of batch cycles
- Created plan to manual migrate custom application
- Created plan to merge applications
- Combined all plans to have a single master plan





# Project Objectives

- provide consistent support to existing clients while reducing overall cost of software footprint
- provide environment for all departments to support client reporting requirements
- create reporting architecture that will reduce overall time-to-delivery for future business intelligence projects
- define environment that takes advantage of existing hardware resources
- reduce overall headcount to support reporting environment
- create technical architecture that meets or exceeds performance system level agreement (SLA) for all projects
- provide an extensible and flexible reporting environment
- improve the technical development and quality assurance environments to ensure production deployed software meets quality standards



# Capacity Planning Considerations

- Number of expected users and active users
- Number of expected concurrent users
- Usage of static vs dynamic content
- Number of objects in the repository
- Number of expected online jobs
- Ratio of batch vs online job execution
- Usage of thin-client vs Plug-in
- Number of replicated services (hardware availability)
- System Level Agreement (SLA) expectations





# Performance & Tuning Considerations

- Replication of services (fault tolerance and performance)
- Tuning of Performance Suite CommonServices (startup parameters)
- Connection allocation by service
- DAS tuning parameters (and replication)
- Evergreen Services – automatic detection and restart of BI Service and DAS
- Database tuning for high-impact queries and reports
- Batch capabilities to leverage off hours CPU capacity





# Performance and Testing Findings

- **Job execution and job statuses can be tricky**
  - Execution of jobs by Performance Suite do not always reflect the correct execution status
  - Jobs will also disappear (possible cause is lost database connection along with a series of reportedly fixed bugs)
    - (1-183588405 - JobStatus api does not work, 1-185523180 - On startup, BI jobs attempt to run before BI Service has started.
    - 1-218429148 - Using FTP in a cycle as an externally triggered event does not perform consistently
    - 1-218429485 - On startup, BI jobs attempt to run before BI Service has started.
    - 1-220055201 - DAS process crashes - BQY with Auditing events defined.
    - 1-ZZ04D - An RTE that is triggered from an ETE does not run when the ETE is triggered from the database.
    - 4-BRIO100024326 - SQL update for External Trigger not initiating Recurring Event
    - 4-BRIO100027286 - Job notification job log attachment is NOT delivered to lotus notes email)
  - Apparent solution: tuning the execution to control the job flow
  - These parameters are required in the startCommonServices.sh



## Performance and Testing Findings, cont.

- GROUPS can cause tremendous bottlenecks
  - In some cases, the Performance Suite moves 30 megs of data to authenticate the user (using native authentication)

Possible solution: use ROLES instead of GROUPS



## Performance and Testing Findings, cont.

- SQRP functionality was failing
  - Jobs continued to fail when opening output
  - Appeared to be security related
  - Nothing documented regarding the change
  - Logging provided confusing messages

Solution: Hyperion Support and our team finally tracked down the problem through a comparison of an upgrade install and a fresh install (no SQRP was found on the fresh install and is an undocumented change)

- In order to address this problem, a job service needs to be configured to execute the SQRP



## Performance and Testing Findings, cont.

We found on large outputs, we could not load all of the report into the browser memory space

We could do this in 8.2.1, but not in 8.3.2

We also found that the back button was “grayed out” and we could not retrace backward

Solution: After considerable testing, this ended up being classified a Performance Suite bug and we were issued a patched JAR file



## Performance and Testing Findings, cont.

### Load testing revealed a weakness with WebSphere 5.0.2

When scaling users up to 200 concurrent users during load testing, we found that WAS would hang because it was not returning threads back to its pool for reuse. Possible memory leak.

Solution: we now bounce WebSphere with our weekly restart of Hyperion Performance Suite services.



## Performance and Testing Findings, cont.

### DB2 database configuration considerations

During testing, we found specific parameters had profound effect on real-time processing

- Query Optimization level
- Consistent execution of RUNSTATS and REORGs
  - to ensure we're using appropriate objects to retrieve the data
  - Foundation will dynamically determine what access path to use based upon the statistics available
- Locksize parameter
- Ensure that fix pack levels are reviewed and leveraged

Logging (log4j) became our “best friend”



# Performance and Testing Findings, cont.

- ✓ **Leverage the Capacity and Performance Guide from Hyperion**
- ✓ **Tune each service accordingly (Remote and Local Service Configurator tool)**
- ✓ **Remember to use and then turn off “DEBUG” level logging**

## **Set startup Java parameters as needed (see Appendix A)**

-Dmax\_db\_pool\_size=<# of connections>  
-DPerformance.PreparedStatementPoolSize=<#of cursors per connection>  
-DPerformance.JDBCFetchBlockSize=<# of items to fetch at a time from the database,defaults to 50>  
-DPerformance.ContainerCache=3000  
-DPerformance.ContainerOutputCache=600  
-Xnoclassgc  
-Xincgc  
-Dbrio.content.path  
-DPerformance.PFCacheLifeTime=300000

## **Job Execution Parameters**

-Djob\_limit=500  
-DPerformance.SchedulerDelay=15  
-DPerformance.SchedulerBatchSize=16  
-DPerformance.MaxSTWorkers=8  
-Dsched\_retry\_processing=false

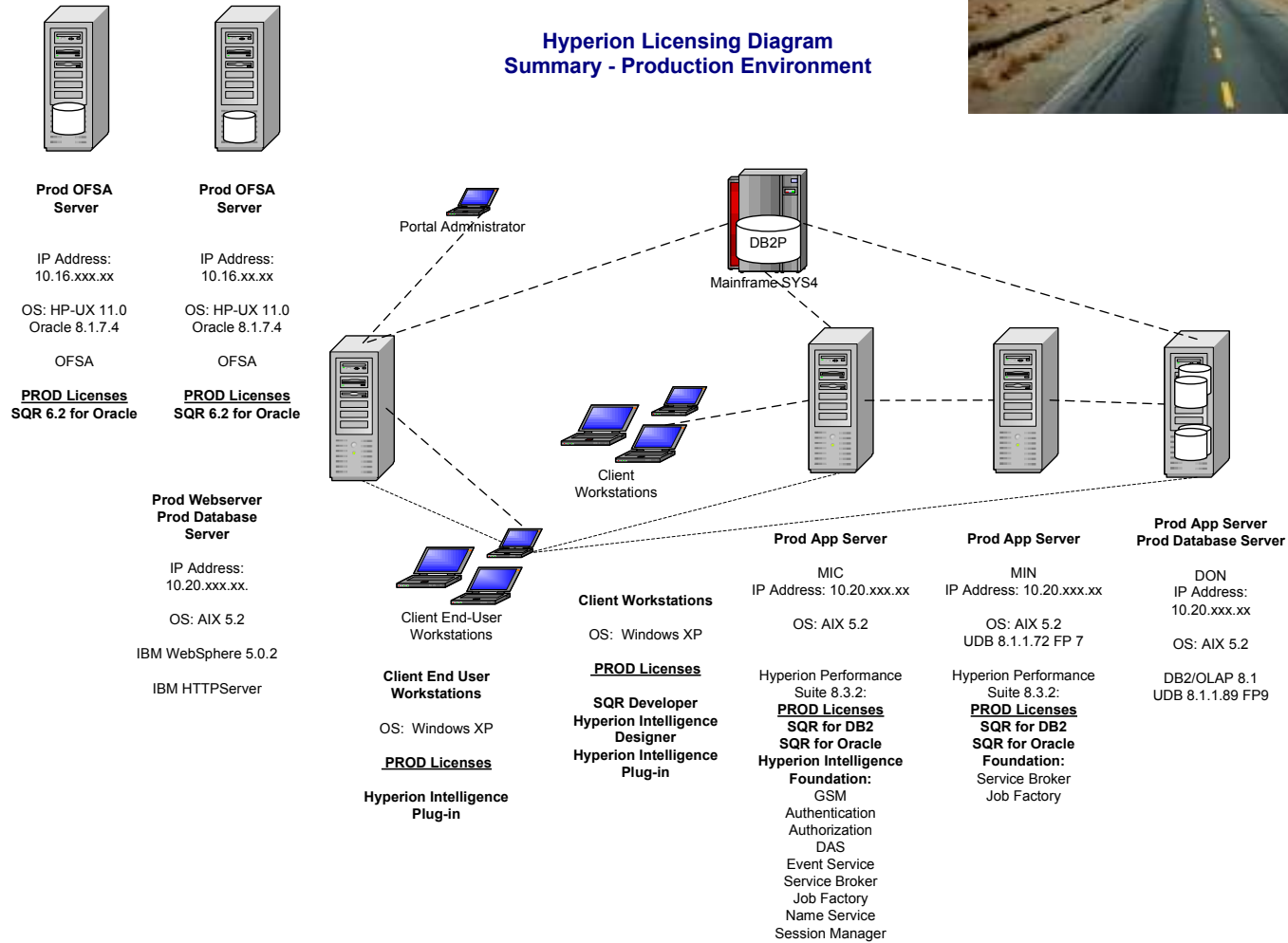
- ✓ **Tune DAS and BI Service appropriately (including replication and distribution)**



# Conclusion: Getting There



Hyperion Licensing Diagram  
Summary - Production Environment



# Conclusion: Getting There



- Planning sessions to identify and mitigate risks
- Resource planning
- Internal marketing and selling of the combined environment
- Feasibility prototyping
- Rigorous test plans and test execution
- Performance analysis and testing
- Technical architecture research
- Review of security concerns for combined environment
- Vendor awareness of our project goals
- Consideration for future migration plans to System 9