Foglight®

A New View on Application Management

www.quest.com/newview

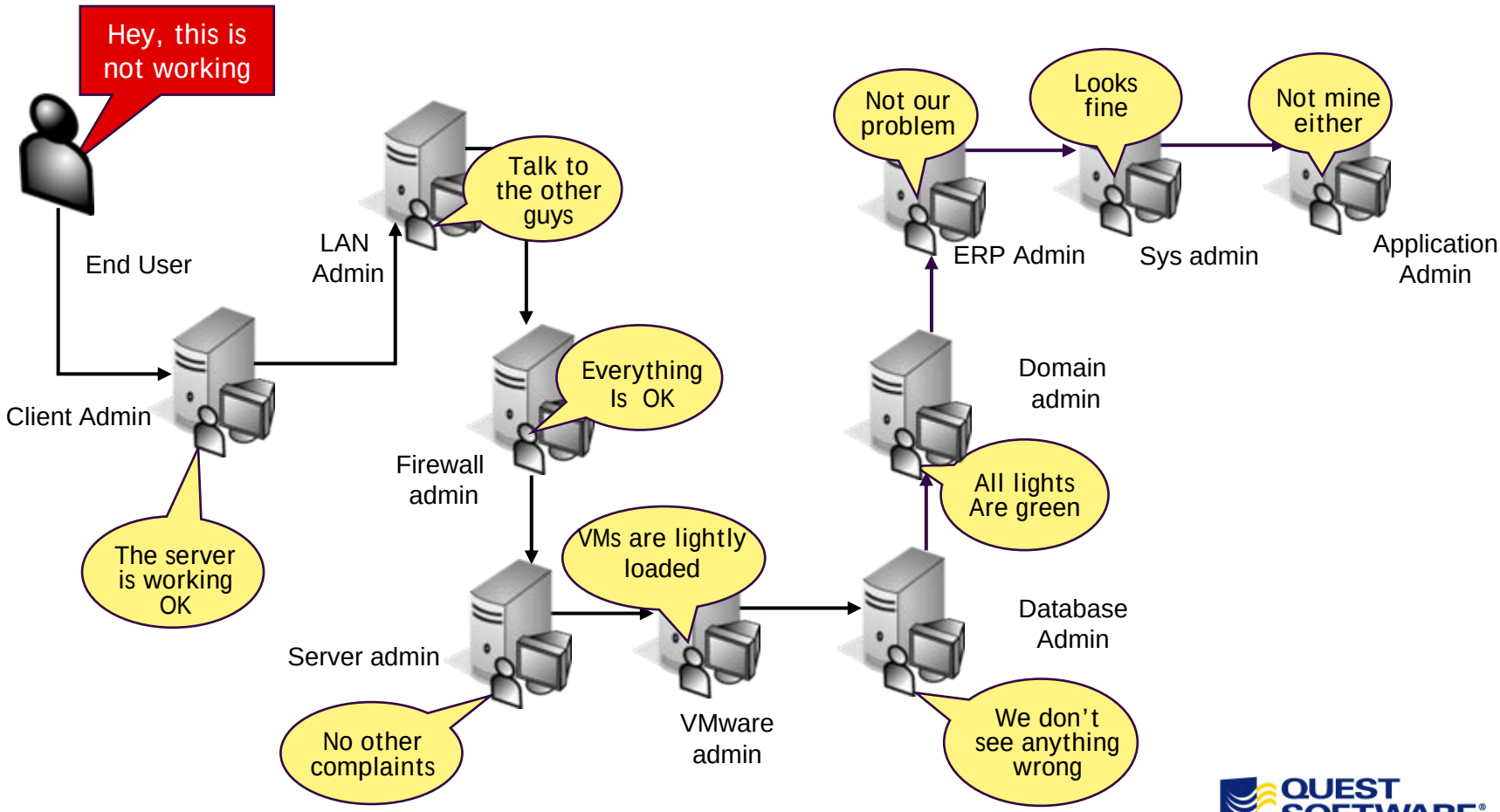
Monitoring applications in multitier environment

Uroš Majcen
uros@quest-slo.com

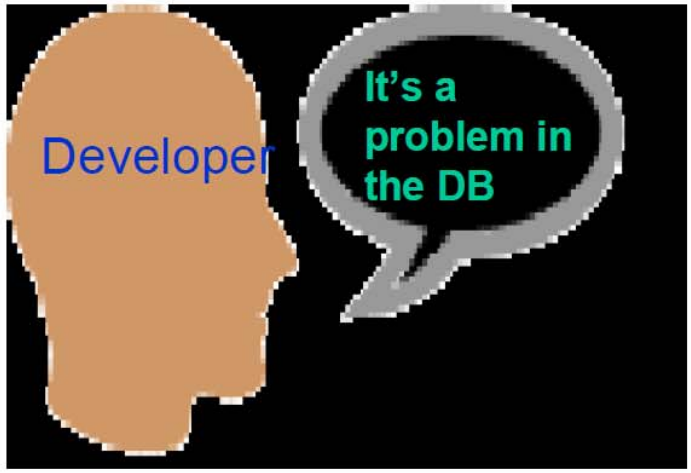
QUEST SOFTWARE®

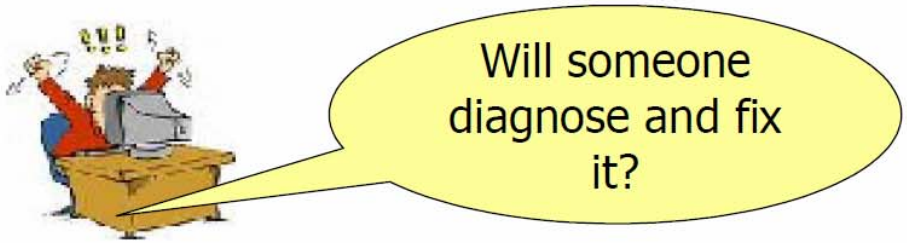
Management Challenges

Siloed organizations result in the “It’s not me!” syndrome



Who will own the issues?

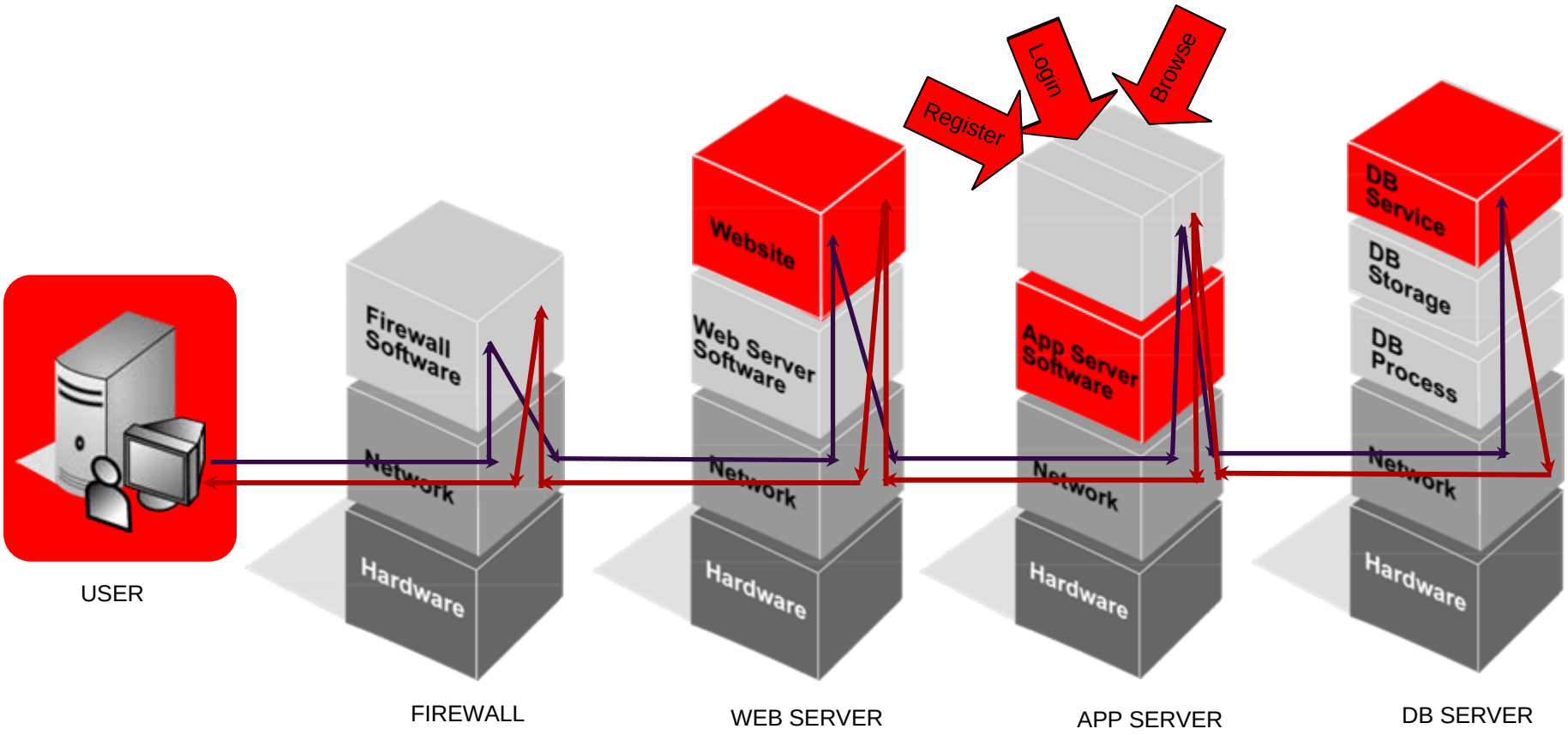




Administrator

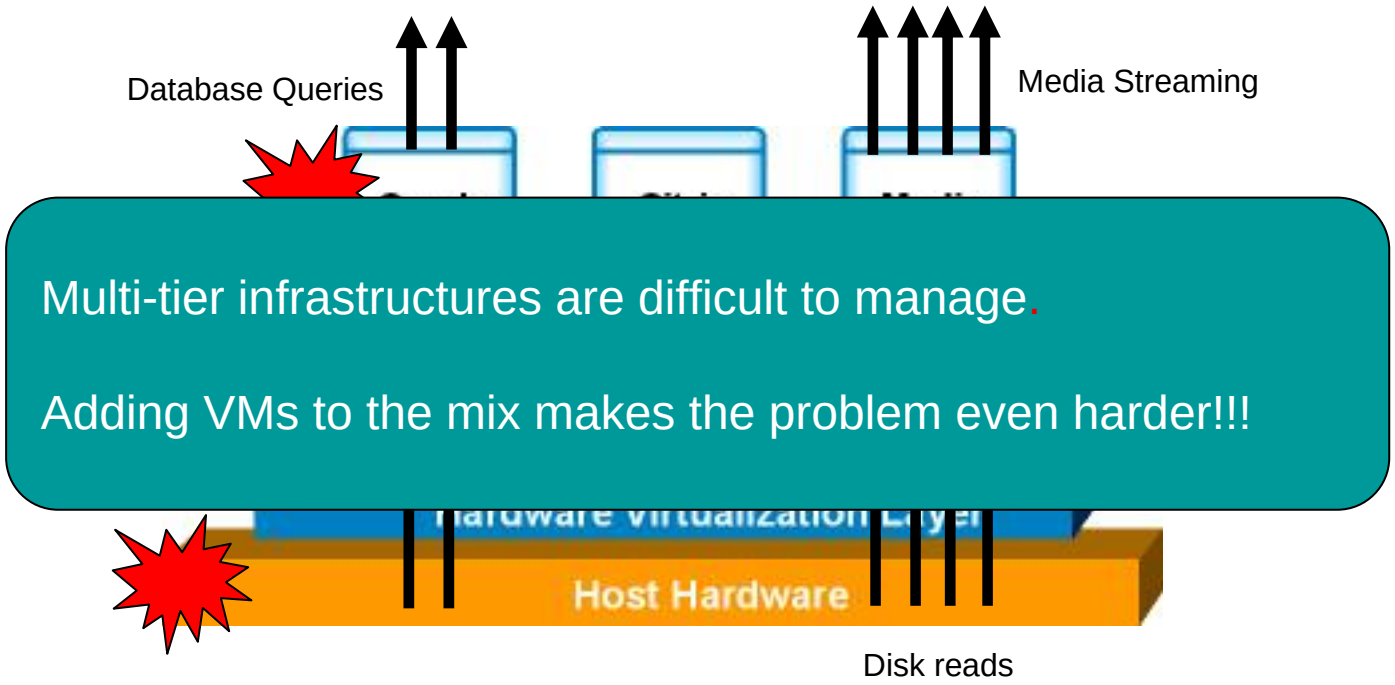
Monitoring Silos Doesn't Work

Suppose the database server is 50% slower than normal



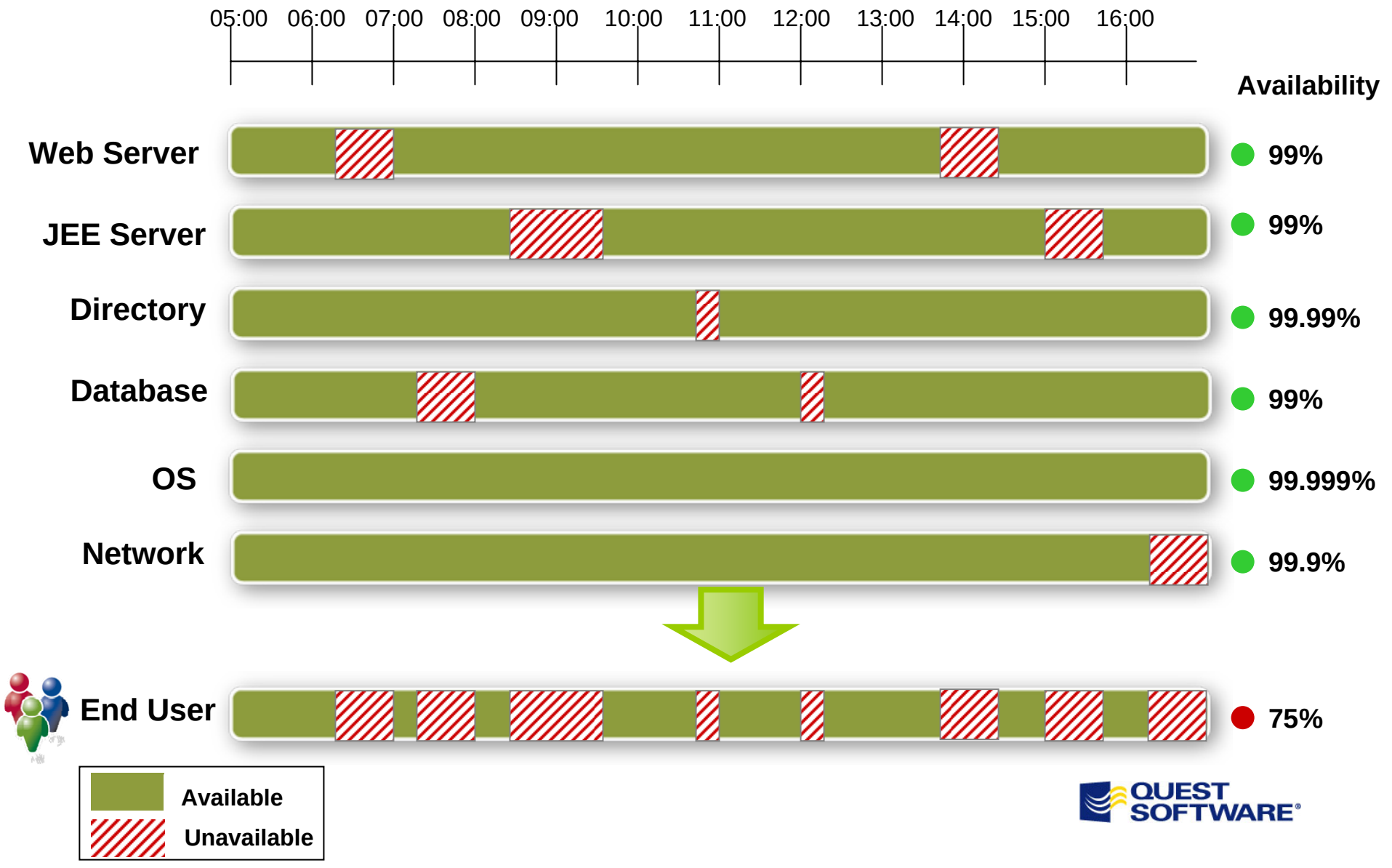
A problem in one application can affect all the other applications involved in the service delivery.

Monitoring Silos Doesn't Work

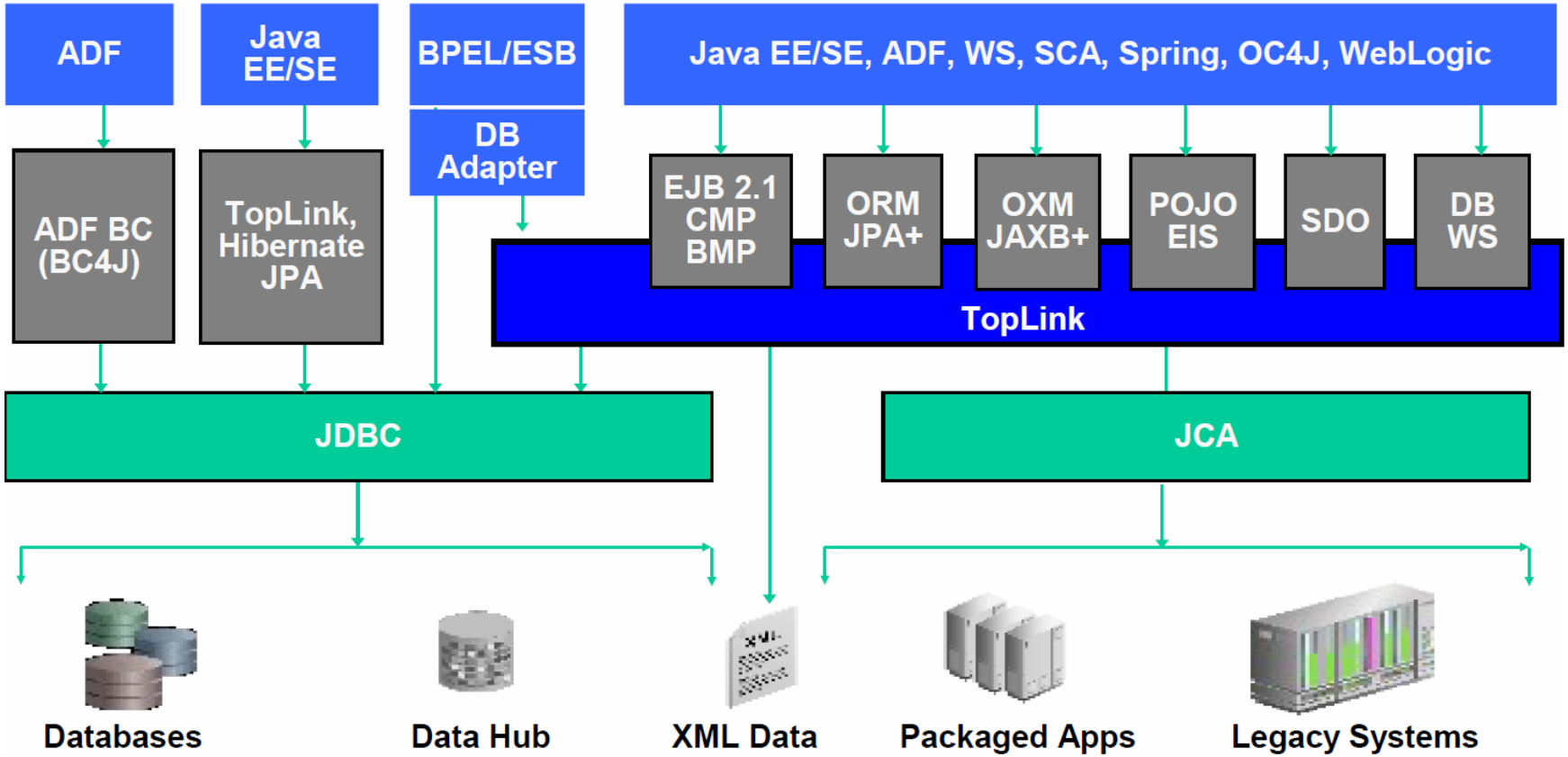


Excessive disk reads by the media server slow down Oracle database accesses

Holistic View of Performance



Applications using Java Technologies

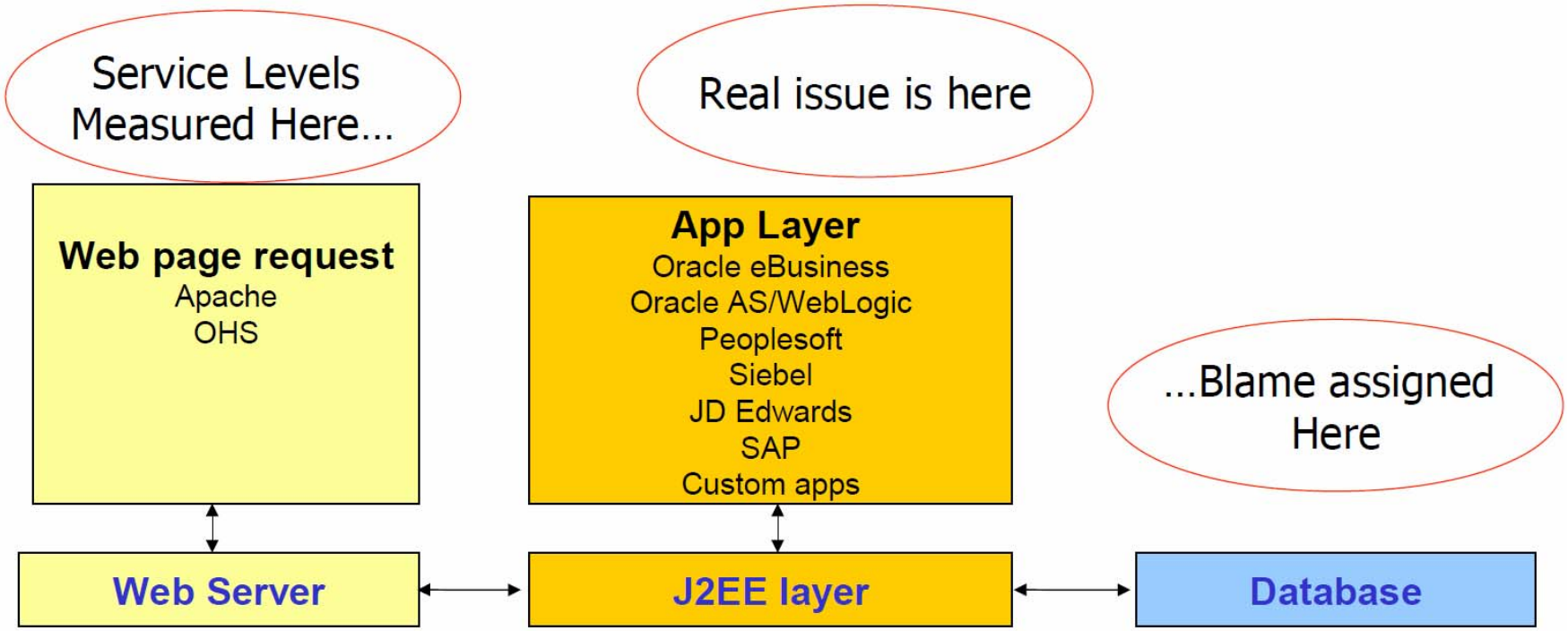


Application Performance Issues

- Slow and slower response times
- Slower under heavy load
- Sporadic Hangs and aberrant errors
- System Locks up
- Sudden Chaos/Unexpected errors

Diagnostic Challenges: Lack of Visibility is Source of Problem

Typical Multi-Tiered J2EE System Leaves the Administrators Blind



Challenges in Diagnosing

- Highly distributed systems
- Cross Tier - J2EE, DB, SOA
  - How does DB configuration affect application performance?
  - How to correlate SQLs to Java sessions?
- Was AS tuned properly?
- No visibility into runtime environment
  - What error customer saw on screen?
  - Which part of the code is taking time?
  - Which application server instance is causing a problem?
  - Was the problem in J2EE or DB Layer?

Application Diseases

- Java memory leak issues
  - Linear
  - exponential
- Bad Coding
  - Infinite loops
  - Exception handling
- Resource Leak
- Thread deadlock
- Incorrect Application Server Configuration
  - Pools, caches
  - memory
- External Issues
  - JDBC/DB Issues
  - Messaging Provider

Approaches to Application Diagnostics

- Monitoring Metrics
  - Application, J2EE Servers, DB and Machine resources
- Find top SQL from DB monitoring and find out
- application code responsible
- Rerun Use case in test environment
- Use JVM diagnostic tools in test environment
- Use logs

Challenges In these approaches

- Logs don’t have sufficient data
- Correlating logs across multiple application server instances and other tiers is very painful; impossible in some cases
- Test environment can’t reproduce the exact scenario - load, resources contention, etc
- The actual execution context is lost forever

Challenges In these approaches

- Challenge in finding offending SQL Code in modern app using JPA or O-R Framework (TopLink, Hibernate)
- No cross tier correlation of application code
- with DB tier
- How to correlate metrics from various sources

Monitor in production and diagnose in test/development environment

Application Problems and Possible Causes

Slower over time or under load	Memory leak
Application hangs in intermittently or times out	Application code or improper app server configurion
Slower response over time, intermittent hangs	Resource Leak
Slower consistently and under load	JDBC or DB problem
Sudden chaos	Threading problem
Hangs and Sudden chaos	Application Server or Back-end DB problem

Diagnose Application Server Issues

- Monitor Application Server resources and search for bottlenecks
  - Thread pools / Work managers
  - Resource usage (JMS, Data Sources)
  - Applications (EJB Pools, bad JSPs/Servlets)
- Look for possible errors in Logs
  - Time outs
  - Exceptions
  - Memory issues
- Use tools to proactively monitor using alert/event notifications

Application Code Issues: Using Diagnostics Tools

- JMX based
- Byte Code Instrumentation / AOP
- JMVTI

JMX Based Monitoring

- Monitoring tools JMX data exposed by the application server and/or applications
- Get some high level metrics such as Response Time, Load, Open Connections, EJB Count, etc
- Alerts on threshold can give some indicators
- Limitations
  - No insight into code
  - Can not correlate user requests with mid tier metrics

Byte Code Instrumentation

- Find out code traces
- Can correlate user requests to code stack
- Find out code bottlenecks
- Limitations
  - Need server restart or application redeployment
  - Either instrument every thing or know what to instrument
  - Over instrumentation will be have overheads
  - Functional instrumentation is difficult and is impossible for admins

BCI Tools

- Aggregate of traces
- Segregation by layers in J2EE
- Some provide remoting support
- Capture each trace
- Some products
  - ClearApp
  - Dynatrace
  - CA Wily
  - Some parts of Quest Foglight

JVM Native Diagnostics

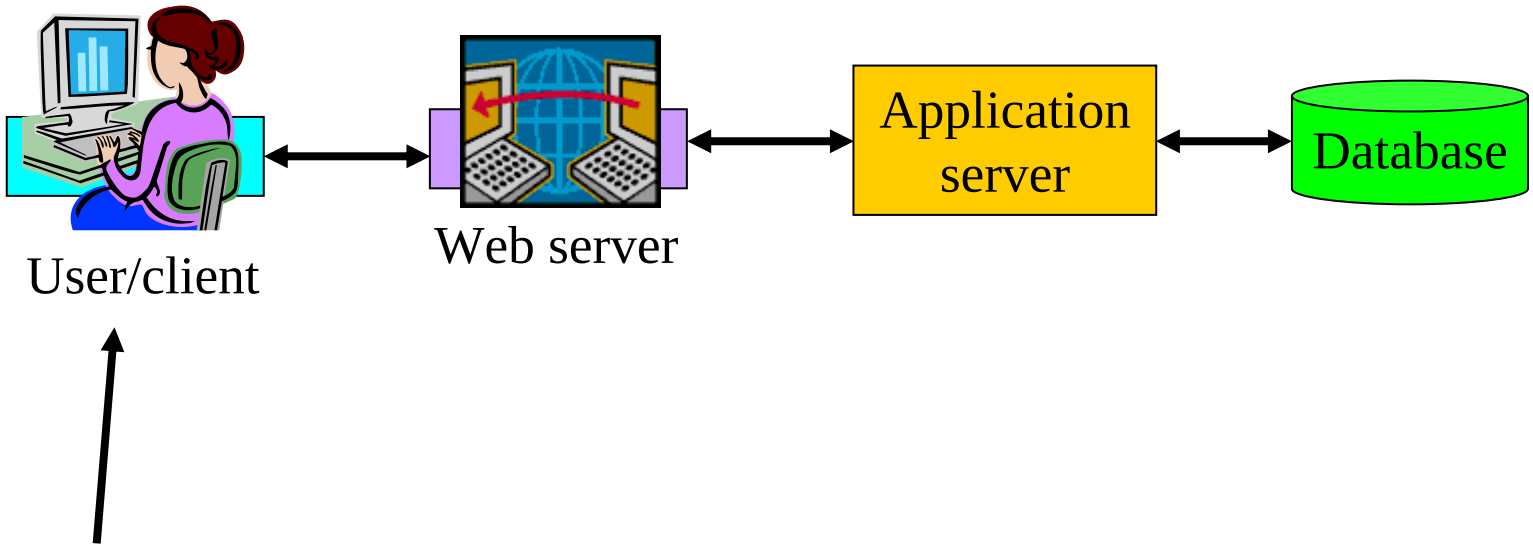
- JVM specific agent running within JVM
- Take snapshots of JVM threads and heap
- DB agent makes corellation with J2EE stack
- possible
- Advantages
  - Extremely low overheads (< 1%)
  - No byte code changes
  - No server restarts or application redeployments
  - Perfect for production scenarios

Also used in Quest Foglight

Additonal Approach or Add to BCI and JVMTI is End-User Performance Management (EUPM)

Definition:

The ability to proactively manage the performance and availability of enterprise applications including Web, Legacy, Client Server ,Citrix and Virtualization applications ***from the perspective of the end-user***.



- It is these folks who determine the application's value
- It is these folks who complain when something goes wrong
  - ➢*Even when there are no problems in the Application IT Infrastructure!!*

… so let's measure application success based on how the end-users are served

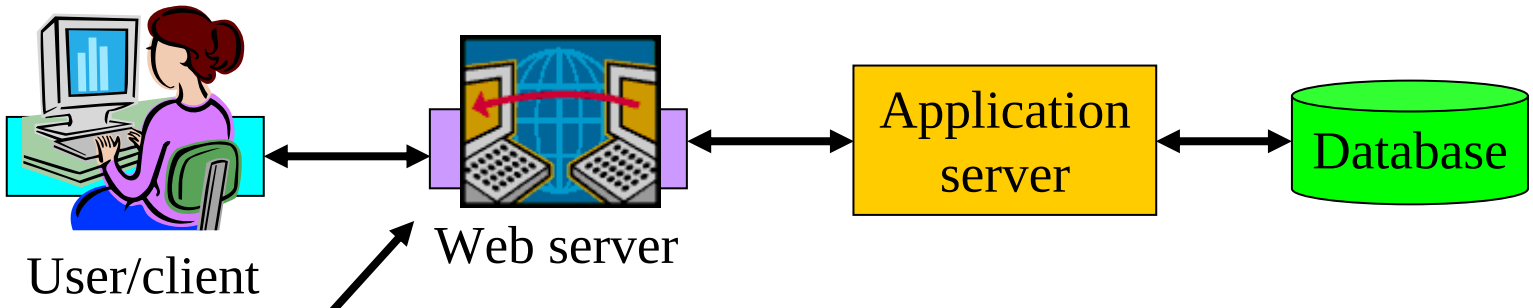
Complementary Approaches for End User Experience Monitoring (Active vs. Passive)

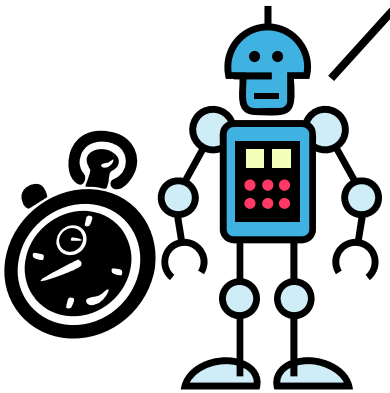
Synthetic Transactions

- *Record and playback same transactions at regular intervals and monitor the response times*
- Less variability is good for repetitive monitoring
  - Consistent locations
  - Consistent connectivity
  - Consistent browser type
  - Consistent paths
- Great for predictive/proactive monitoring (especially after a change)
- Great for availability monitoring and reporting

Passive User Monitoring

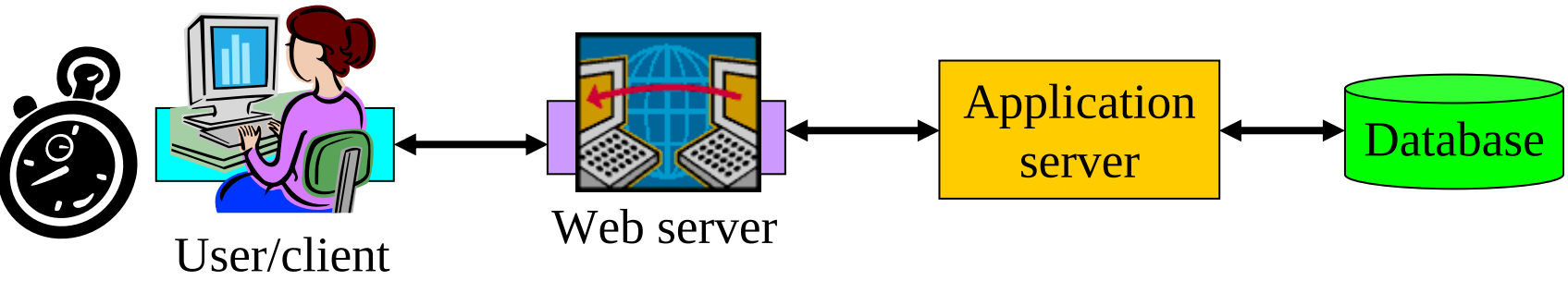
- *Monitor all the activity of the application users continuously*
- Covers all cases not covered by proactive monitoring
  - All users
  - All connectivity types
  - All browsers
  - All paths
- Great for service level monitoring
- Great for identifying slowest & most common user interactions
- Details traffic volume, network performance, server utilization, and backend time on user experience





Artificial user
Performing and timing
Standard artificial (“synthetic”) transactions

Focus: how long does the transaction take?



Focus: how are my actual users doing?

Normally concerned with diagnosing and resolving problems with individual transactions

Ideal solution should provide

- Provide a 24x7 Holistic View of the service delivery chain operation from the End Users perspective
- Quick identification of where the performance bottleneck lies within the service delivery chain
- Real-time alerts based on actual user activity
  - Ideally compare against historical baseline

Ideal solution (continued)

- Minimal impact on existing infrastructure
  - During peak periods, infrastructure is often pushed to its limits, additional management traffic burden can adversely impact end user experience
- Easy to Deploy, Easy to Use
  - Limited IT resources
  - Ideally drill-down based reporting interface, show high-level reports first
  - Web service delivery chain operation is complex, typically managed by different groups within the IT department, provide an interface where each group within the IT department can find relevant information from their vantage point

Several Approaches to this problem

Several products track "user experience" including:

- Web log analyzers
- Content tagging
- Synthetic transaction monitoring

	Implementation	Analysis	Limitations
Web Analytic Tools	▪Post-process of web logs ▪Scheduled log analysis ▪Spiders a site for content problems	▪Page design ▪Broken links ▪Usage/Navigation ▪Estimates end-user experience	▪**No** performance analysis ▪**No** historical baselines ▪**No** alerting
Content Tagging	▪Insert a "tag" on each page to be monitored ▪Delivers data at the end of the page during an image "get" ▪Processes the web logs to get mine the data	▪Captures real-user traffic (site traffic, usage, and performance) ▪Profiles users by browser type & connection speed ▪Performance of page and components ▪Still predominately web analytics data	▪**Cannot** track errors ▪**No** availability metrics ▪**Cannot** identify specific servers ▪**Cannot** distinguish between server and backend time
Synthetic Transactions	▪Scheduled playback of recorded transactions ▪Limited number of locations, connectivity options and paths ▪Must manage scripts	▪Transaction, page and component performance ▪Limited to external view of performance ▪Performance by location and ISP	▪**Limited** to recorded transactions only ▪**No** real user analysis ▪Alerting capabilities **limited** to trxn/page performance & errors

Best Practices

- Use single console to monitor all system components
- Choose right JVM diagnostic tool
  - Should not need restarts/redeployments
  - Monitor threads, heap real time
  - Near zero % overheads
  - Cross tier
- Capture the actual HTTP transactions seen by clients
- Capture traces in real time and segregate performance by various J2EE layers
- Correlate transactions across JVMs

- Dužina prezentacije je nestandarna za prikazivanje na web siteu.
- Za pogledati ostatak prezentacije koristite postkonferencijski DVD.