

MAXIMIZE THE
BUSINESS VALUE
OF SOFTWARE

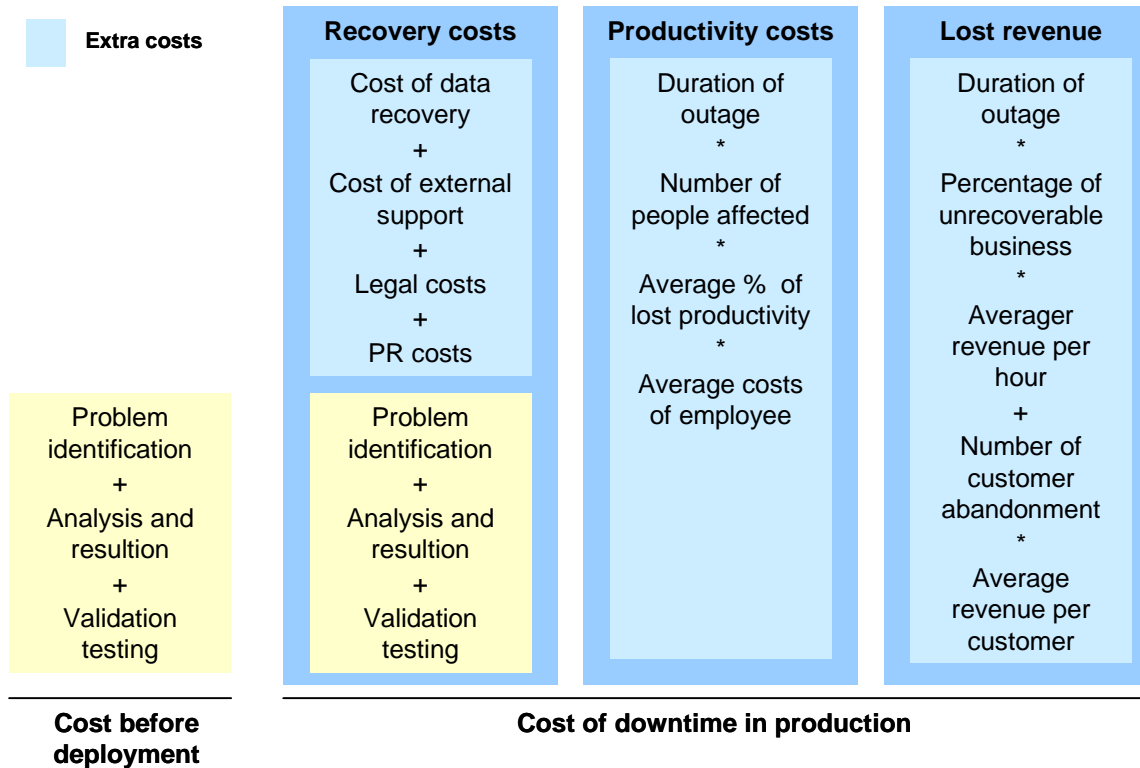
Choosing a Load Testing Strategy
P. R. Sriram
Solution Architect, Asia Pacific

What Is Load Testing?

- ❖ Systematic exposure of an application to real world, expected usage conditions – before deployment
- ❖ Analyzes an application's quality of service
 - Performance (response times)
 - Scalability (throughput)
 - Reliability (availability and functional integrity)
- ❖ **Goals**
 - Predict application/system behavior
 - Pinpoint/diagnose problems in an application and its underlying infrastructure
 - Understand resource requirements for capacity planning

Why Load Test?

❖ Save costs by reducing the number of application failures



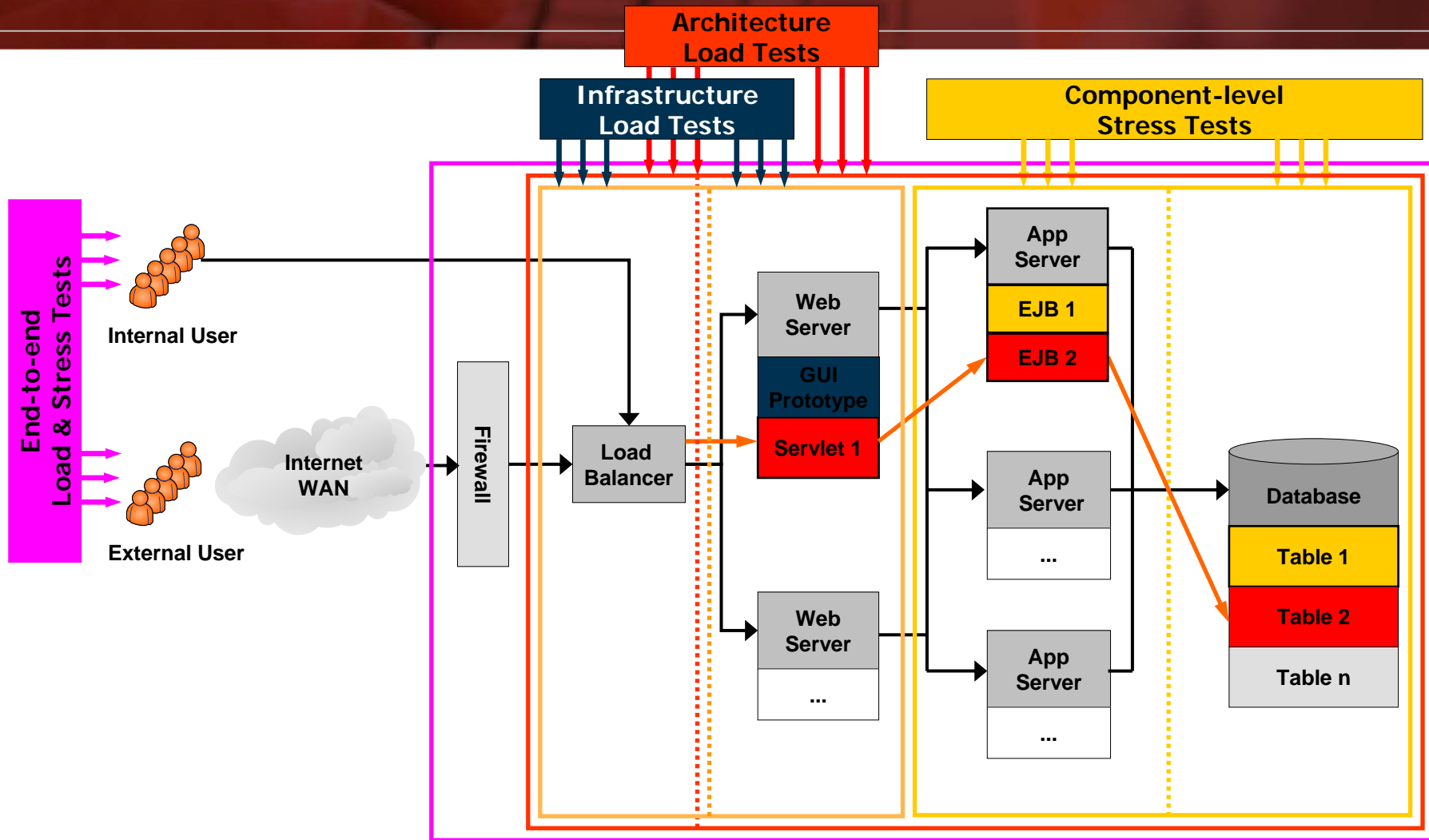
\$100k per hour*

❖ Lower costs of the application infrastructure

- Inefficient application/system performance
- Unused system capacity

* Avg cost of downtime for mission critical applications;
Source: Gartner

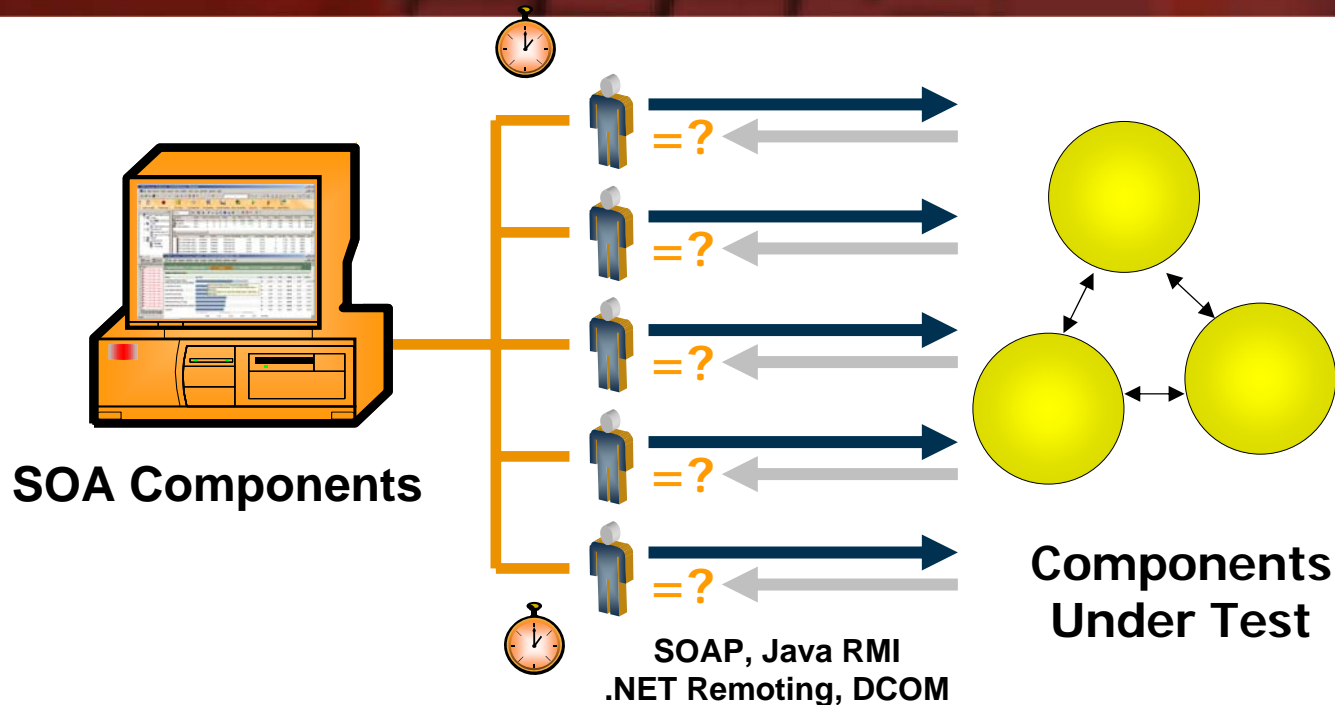
When To Load Test?



Component Stress Testing

- ❖ Issues in middleware components are expensive to fix
- ❖ Interoperability between all the different components becomes more and more an issue
- ❖ Classic Unit Testing Is Not Enough
- ❖ Ordinary unit testing does not find defects that occur due to concurrent access and high load

Find Multi-User Access Problems Through Emulating Realistic Server Conditions



- ❖ Does each component work to its specification – also when accessed concurrently?
- ❖ Do all middleware components meet their performance criteria?
- ❖ Does performance degrade under concurrent access

Infrastructure Load Test

- ❖ Hardware Infrastructure
- ❖ Software Installations
- ❖ Network Infrastructure
 - Routers / Switches
 - Bandwidth Utilization / Availability
 - Data Transactions – Data Sent & Data Received
- ❖ Data Preparedness
- ❖ External Production User Behavior

- ❖ Based on a Business Scenario

End-to-End Load Test

- ❖ **Benchmarking**
 - Test QoS of various architectural options
- ❖ **Regression Load testing**
 - Verify QoS from build-to-build
- ❖ **Application tuning**
 - Optimize you application's QoS
- ❖ **Capacity planning**
 - Accurately predict the required system and network capacity
- ❖ **Reliability Testing**
- ❖ **Scalability Testing**
- ❖ **Meeting Customer's Requirements**

Strategies For Load Testing

- ❖ Manual load testing
- ❖ Home-grown load testing tools
- ❖ Open-source load testing tools
- ❖ Testing frameworks integrated within IDEs
- ❖ Web-only load testing tools
- ❖ Enterprise-class load testing solutions

Manual Testing

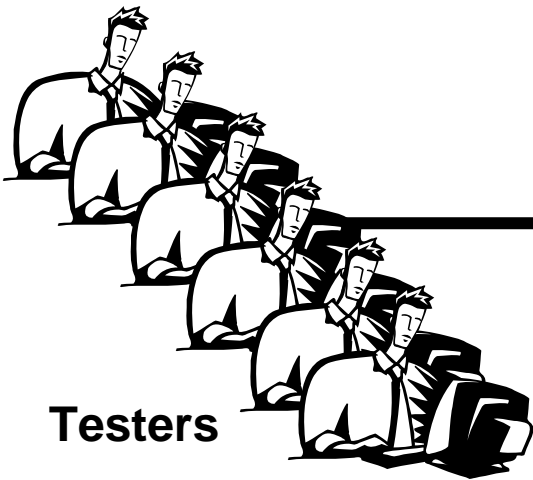
Resource intensive

- Testing staff
- Client computers



Coordinator

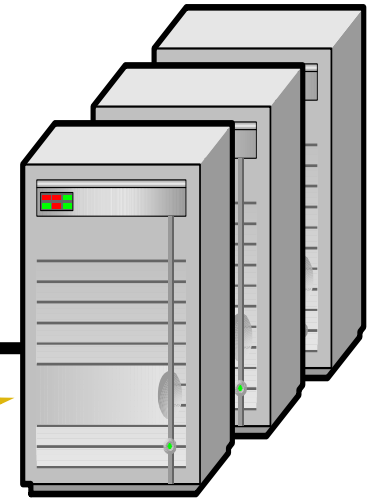
Repeatability?



Testers

Internet

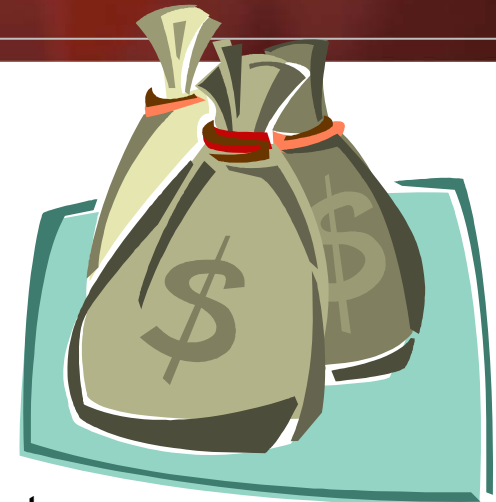
Results Analysis?



System Under Test

Home-Grown Load Testing Tools - Disadvantages

- ❖ Limited test coverage and accuracy
 - Only test very specific functions
- ❖ Lack of reusability
 - For other projects or applications
- ❖ Staffing specificity
 - Only the tool developer understands the test script
- ❖ Poor reporting
 - Lack of standardized reporting
 - Reporting not adjustable to user/management needs
- ❖ Lack of diagnostic tools for root-cause analysis
 - Manual analysis of log files is difficult



Open Source Load Testing Tools - Disadvantages

- ❖ May satisfy very basic needs for automated load testing
 - FREE*
 - Extensible
 - Every developer's dream come true – a tool with source available so it can be “customized”
- ❖ *eWEEK*: “The main weaknesses in these tools are their lack of advanced scripting options for testing complex applications and their often-limited reporting options.”
 - Missing high-level scripting APIs -> long scripts
 - Often based on Java technology -> performance overhead
 - Inaccurate emulation of a Web browser's user load
- ❖ **Point solutions**
 - Lack application support beyond Web/HTTP
 - Developer oriented with no recorder and Java scripting only
 - Do not integrate in an overall testing and performance lifecycle solution

Testing Frameworks Integrated Into IDEs - Disadvantages

- ❖ Very developer-centric view
 - Developer does coding, unit, load and production testing
 - View of life cycle maps only the perspective of a developer
- ❖ Only offer support for either .NET or J2EE
- ❖ Not for mission-critical, heterogenous applications
- ❖ Don't cover all aspects of testing and the SLC

Web-only Load Testing Tools

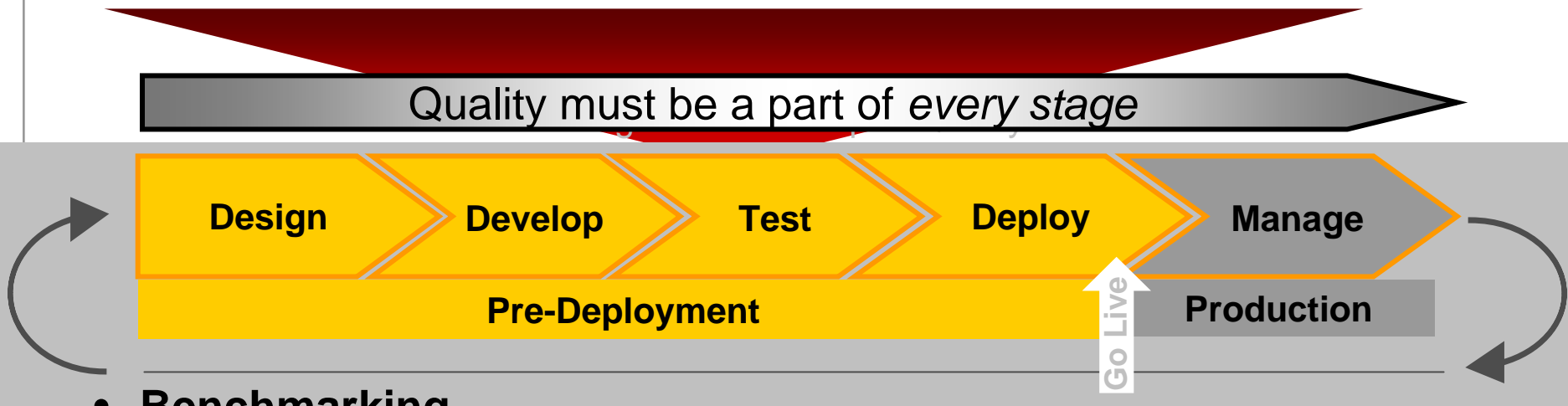
- ❖ Not future proof - new tool for testing non-Web applications required
 - Additional license, training and implementation costs
 - Existing knowledge cannot be reused
- ❖ Typical limitations for testing Web applications
 - Scalability and accuracy
 - Browser compatibility
 - Reusability
 - No component testing

Enterprise-Class Load Testing Solutions

❖ Advantages

- More than just a capture/playback tool for Web apps
 - Support load testing heterogenous application environments
 - High scalability levels with reasonable costs
 - Rapid test design and problem diagnostics
- Integrated into a holistic software quality optimization solution
 - Complementary tools that cover the complete SLC
 - Test management
 - Developer oriented (unit) testing
 - Network performance/readiness testing
 - End-user experience monitoring
- Training and consulting services to ensure succesful implementation

Leverage All The Benefits Of Load Testing Through Viewing Quality Holistically – Across The Entire SLC



- **Benchmarking**

- **Component-stress testing**
- **Architecture tests**
 - **Performance regression testing**
 - **Application tuning**
 - **Service-level/acceptance testing**
 - **System tuning**
 - **Capacity tests/planning**

**End-user
experience
monitoring**

Which Strategy To Choose?

- ❖ Each strategy has its advantages and disadvantages
 - But any of them are better than manual testing!
- ❖ Non-critical applications that require only very little testing
 - Open source or home-grown tools
- ❖ Small projects where developers do the testing
 - Testing frameworks integrated into IDEs
- ❖ Mission-critical applications
 - Enterprise-class load testing with full life-cycle support

How To Justify What You Need

- ❖ **Track true costs and recognize the risk**
 - Manual efforts; missed performance issues and defects
- ❖ **Spread the costs across multiple projects**
 - Testing is important for all software projects
 - An enterprise solution can be used across many different technologies, software applications and environments
- ❖ **Start modestly and grow**
 - Even a small number of virtual users, used early in the development process, usually pays off
 - Entry-level solutions - look at the upgrade path
- ❖ **Choose a solution that provides measurable value**
 - Metrics that document quality levels on a regular basis
 - Load test management

MAXIMIZE THE
BUSINESS VALUE
OF SOFTWARE

Thank You
P. R. Sriram
pr.sriram@borland.com

Borland[®]