

**AZTEC**SOFT™

[www.aztecsoft.com](http://www.aztecsoft.com)

Performance  
Engineering -  
Common Issues

# Abstract

“Effective software performance engineering requires that the various stakeholders of the project – performance testers, engineers, architects understand the fundamentals of performance testing, analysis and tuning.

In order to carry out effective performance tests it is very important to understand the objectives – is it to figure out the break points of the system? Is it for benchmarking? Is it for capacity planning etc? In order to design effectively for performance software engineers must understand the general performance software patterns and antipatterns.

The performance engineering group at Aztecsoft has worked with various performance engineering projects and this talk describes our experience with various stakeholders of the system and highlights the common mistakes that affect such projects. The speaker will share common performance testing blunders and common performance antipatterns that affects the system”

# Performance Testing - Common Mistakes

- Clear objectives
  - Is it for product benchmarking?
  - Is it for Capacity planning?
  - Is it for Performance analysis?
  - Is it for understanding the general performance behavior of your application or product?
  - Is it for beating competitors benchmarks?
- Understand the application
  - RIA?
  - Client/Server?
  - Web application?
  - SOA?
  - SaaS?

# Performance Testing - Common Mistakes

- Understand the workload characteristics
  - User load
  - Transaction mix
  - Usage Pattern
- Performance Test
  - No steady state load test
  - When to stop load testing
- Validate Performance Test Results
  - Validate Load – Little's law
  - Validate Error codes
  - Validate data dependent load
- Test Environment
  - Validate performance infrastructure usage
    - Isolated to performance testing
    - No scheduled jobs
    - No virus scans ☺

# Performance Analysis - Common Mistakes

- User Load
  - Over prediction.
  - Concurrency
  - Not understanding the request arrival rate.
  - Threshold limits
- Performance Analysis
  - Not able to identify the single lane pattern. (Antipattern)
  - Synchronous/Asynchronous behavior of the application
- Performance Monitoring
  - Keep it simple
  - Complement system monitoring with application monitoring wherever possible
  - Service demand

# Performance Analysis -Web Application bottlenecks

- Pagination
  - How much data?
  - How many pages?
  - Do you load all data from the database?
- Inappropriate usage of application cache
  - Too much data is cached
  - Too less data is cached
- Batch services implemented as real time services
  - Reporting
- Not leveraging database capabilities
  - In built sorting/searching algorithms
  - In built joins too!

# Performance Analysis - Web Application bottlenecks

- Too many hits to the database (Antipattern)
  - Within a given transaction scope
  - Within a given request scope
- Too many objects (Antipattern)
  - Heavily pattern oriented
- HTTP Session
  - HTTP Session size
  - Poor HTTP session management
  - Unreleased HTTP session objects.

# Performance Tuning - Common Issues

- Fixing symptoms rather than the source causing them
  - GC Tuning
  - JVM Tuning
- Database is always a bottleneck
- Hardware sizing
  - RAM size
  - CPU power
- Over tuning the code
- Re-active approach

Discuss a case study (If time permits)

## And on top of all ...

Performance Engineering must be a collaborative approach.

Must be Proactive than Reactive