Agenda

1) Application Description
2) Deployment Architecture
3) Understanding Performance
4) Performance Problems and Improvements
5) Summary
Application Description
Atlassian Tool Suite

- **Bitbucket**
  - Source Code
    - Products
    - Tools
    - Test Cases

- **Jira Software**
  - Issues
    - Backlogs
    - Defects
    - Workflows

- **Confluence**
  - Documentation
    - Specifications
    - Checklists
    - Processes

© 2023 NetApp, Inc. All rights reserved.
Users

- 3000 Developers
- 8 Development Sites
  - US & India
Application

- Bitbucket
  - Application
  - Database

- Jira Software
  - Application
  - Database

- Confluence
  - Application
  - Database
Production Deployment
Virtual Machine for Application and Database
Web Reverse Proxy for Network Facing Access
Virtual Machines Require Physical Servers
Virtual Machines Require Shared Storage
Understanding Performance
Is the Application Working?
Is the Application Working?

Predictable and acceptable level of response to user activity
183.983 Average Response Time (ms)

7,610,774 Total Requests

HTTP Response Time (ms)

@timestamp per 3 hours

© 2023 NetApp, Inc. All rights reserved.
Determining The Problem(s), Implementing the Solution(s)

- **Problem: Application is SLOW**
  - Why, when, for whom?
  - How to quantify a specific issue?
  - How to trace an issue to a cause?
  - How to gather data to support analysis?
  - System-wide issue or user-specific issue?

- **Solution**
  - Experience says there won’t be a silver bullet
  - Likely multiple underlying problems
  - One fix may not provide systemic improvement
  - Multiple bullets are going to be required
Performance Problems and Improvements
Key Operating System Measures

- **CPU**
  - Servers have one or more CPU
  - Applications execute across multiple threads which can distribute across CPU
  - CPU can be a limiting resource

- **Memory**
  - Servers have a certain amount of memory provisioned
  - Applications utilize memory to hold executable code and data
  - When OS and application memory requirements exceed RAM, disk level swapping can occur

- **IO**
  - OS and application have to write data to storage
  - IO is the slowest operation to execute
  - IO latency can create conditions where OS and application executions are blocked on IO operations (IOwait)
Infrastructure Based Issue

IOwait Spike

- The Symptom
  - Overnight a serious IOwait spike that hung multiple virtual machines in the deployment
  - Identified multiple VMs having large IOwait spikes during 3-4 AM window
  - No corresponding spike in users or requests

- The Investigation
  - Access with infrastructure virtualization and shared storage teams
  - Guess what? System backups run at 3 AM
  - Uncovered that IOwait issues in the virtual machines correlated with IO max limits being hit on underlying shared storage volume

- The Solution
  - Entire VM environment had been provisioned onto a single storage volume
  - During high IO load, max IOPS of a single volume being reached, causing IO waits
  - Separated DB and HA proxy VM each to separate storage volume (7 new storage volumes)
  - Separated Test environment VMs to separate storage volume (1 new storage volume)
Java Virtual Machine (JVM)

- Java programs compile to byte code language that is executable with a runtime environment called a JVM
- JVM manages program memory, loads byte code and performs dynamic code optimization (just in time (JIT) compilation), and interfaces to operating resources
- Internal aspects of JVM performance are measurable similar to OS level behavior
- JVM Memory has multiple segments
  - Heap Memory – used for program data
  - Reserve Memory – used for program code
  - Reserve Code Cache – used for optimized program code (JIT compiled)
Runtime Environment Based Issue

JVM Random Pauses

- **The Symptom**
  - Jira performs ok, then slows, then speeds back up, then slows again
  - The performance is cyclical but not on a consistent interval

- **The Investigation**
  - Atlassian Support was key here in digging in to JVM internals
  - Enabled debugging & instrumentation, additional garbage collection (GC) logging, performance profiling, analysis
  - Build visualizations of collected data with Elastic Stack (log collection and visualization tool)
  - First Time (less frequent): GC logs showed full garbage collections that lined up with slowdowns
  - Second Time (more frequent): Thread dumps indicated Code Cache too small for JIT Compiler; Jira bug

- **The Solution**
  - First Time: Increase JVM heap from 8gb to 16gb and restart
  - Second Time: Increase Reserved Code Cache from 512mb to 1gb and restart
JIRA Application Extensibility

- JIRA is a platform as well as an application
- Atlassian provides a marketplace for 3rd party vendors to create plugins to enhance JIRA behavior for different workflow needs
- Plugins are combinations of Java server code and java script browser level code
Application Designed Behavior

Slow Page Downloads

- **The Symptom**
  - Page loads are slow for some users; ok for others
  - Page size is huge! 30+MB, 120+ http requests

- **The Investigation**
  - HTTP compression?  
    - Maybe – slower UX for fast networks; haproxy bug with chunked encoding
  - Network speed?  
    - Contributing – slow UX for slow networks; problems both on and off VPN
  - Local caching servers?  
    - Maybe – complex to install for our network; maintenance burden
  - WAN accelerators?  
    - No – dynamic nature of the data limits effectiveness here
  - Plugin javascript bloat?  
    - Contributing – tedious to individually test 80+ plugins; which are needed?

- **The Solution**
  - Found handful of plugins that had bloated javascript requirements (unneeded dependencies, no minimization); worked with plugin vendors to patch plugins
  - Targeted HTTP compression; introduced WebBooster plugin to enable compression for selected offending plugins
  - Reduced issue view test page from ~30MB to ~7MB download (with empty cache)
JIRA Application Configurability

- JIRA does not impose a single type of issue and workflow configuration
- JIRA allows users to customize issues, fields, workflow statuses
- Configurability can quickly grow out of control and is difficult to managed across hundreds of projects within JIRA
Application Configuration Issues

Slow APIs

- The Symptom
  - Workloads involving certain API users show high HTTP response time
  - Workloads tend to impact system in general due request latency and frequency

- The Investigation
  - Determine specific API issue retrievals being done
  - Inspect the request query and response data of specific requests
  - Guess what? Response payload is a JSON payload per issue with 1000+ field names, mostly empty values

- The Solution
  - For legacy reasons, JIRA has in excess of 1000 custom fields; delete fields where possible
  - Unless specified otherwise, JIRA projects have all fields by default
  - Addition of a JIRA field configuration for specific projects for the ~40 fields actually used
  - Adjust API calls to specifically request needed fields instead of default full payload
  - API response times reduced to 25% of previous; payload size 30% of previous
Timeline Summary

- **Performance Problem Detected**
  - Cisco Switch Transmit Pause Bug
  - Bypass Misconfigured PGpool

- **Move Database VMs to Isolated Data Store**
- **Move Proxy VMs to Isolated Data Store**
- **Reduce Webhooks from Slack Integration**
  - Improve Efficiency for SolidFire CI/CD Process

- **Nov**
  - Failed Database Failover
  - Reduce Backup Frequency
  - Increase JVM Heap

- **Dec**
  - Add Field Configuration For SolidFire projects
  - Remove Defunct Bamboo Jobs (Jira Bug)

- **Jan**
  - Tune JIT Compiler

- **Feb**
  - Enable Targeted HTTP Compression

- **Mar**
  - Replace Failing Juniper Firewall HW

- **Apr**
  - Upgrade Juniper Firewall HW

- **May**

- **June**
Summary
Interesting Observations

- As predicted, no silver bullet
- Issues addressed at multiple levels from infrastructure to application configuration
- Data collection was critical
  - OS metrics from VMs
  - Java JVM metrics from internal
  - HTTP request data
- No issues found within internal database performance
- Improvements achieved without changing a single line of code
Thank you.