



Peaceful Distributed Microservice Architecture





Hello!

Vanjikumaran Sivajothy

Committer and PMC - Apache Synapse

Senior Lead Solution Engineer - WSO2

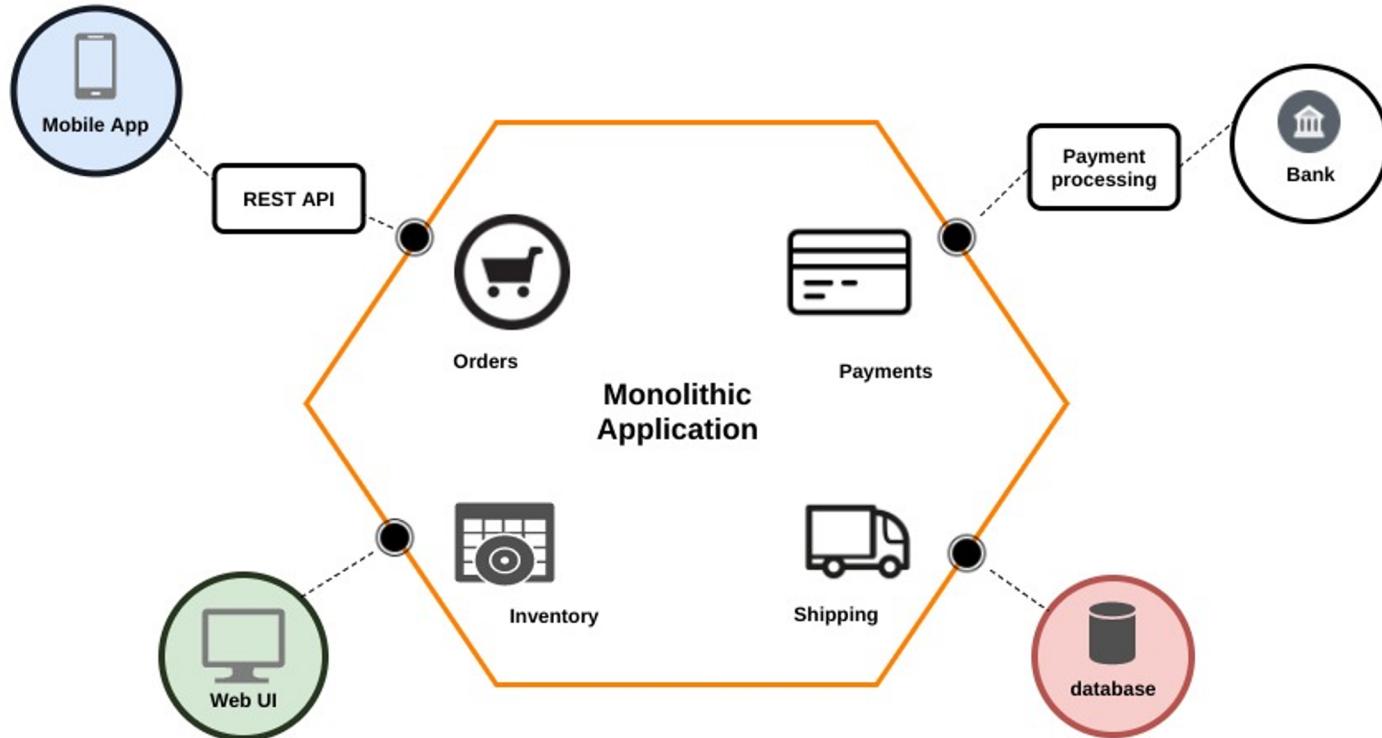


Vanjikumaran



Monolithic

Overview of Monolithic Applications



Monolithic Application (continued)

- 1 Regardless of logical modular, application is packaged as a single monolith.
- 2 Packaging depends on the languagewar, .jar or directory structure
- 3 Simple to test and deploy
- 4 If simple, what is the issue?
Simple and easy only at the beginning

Problems with Monolithic applications.

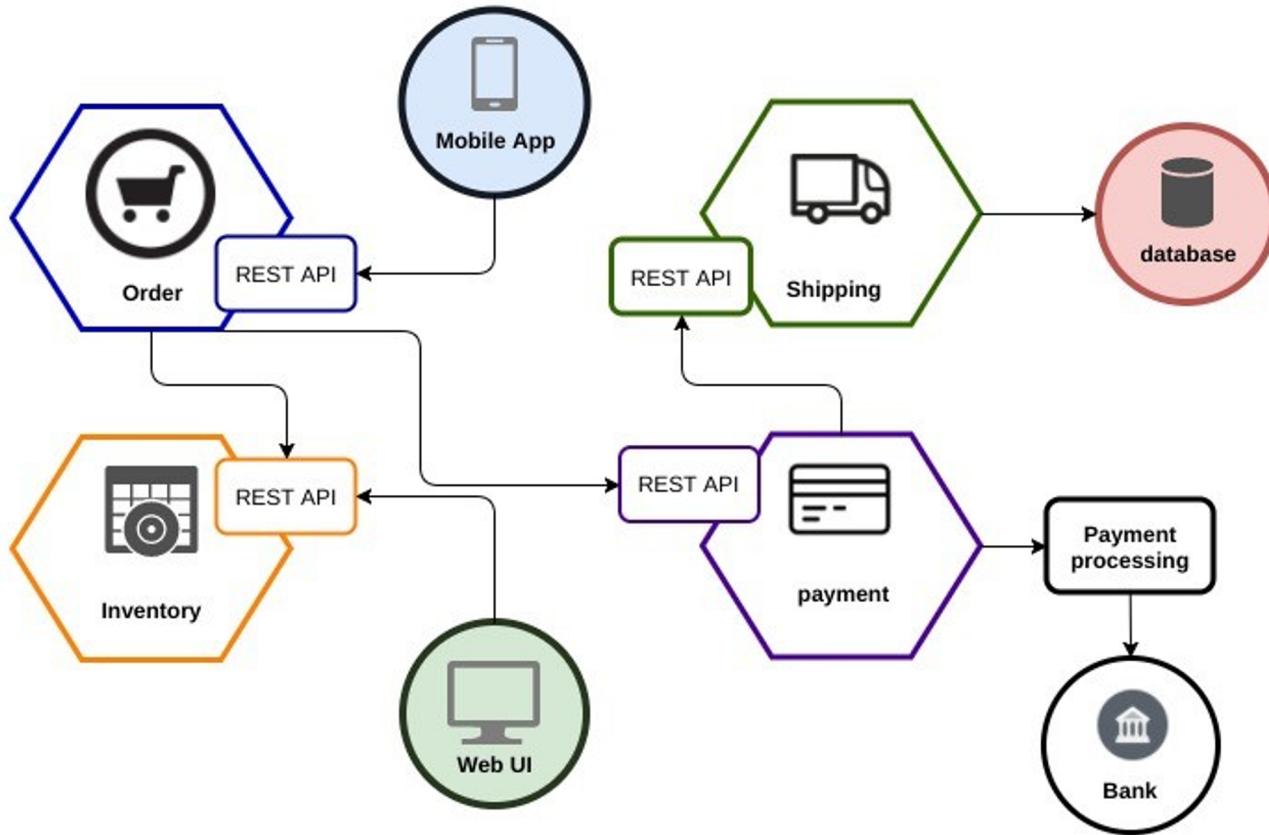
Many Pain Points!!!!

- Increasingly difficult to make code changes
- Disrupts agile development
- Overtime, no single developer will understand the entire code. Changes will be error prone
- CI/CD would become painful
- Scaling would be difficult
- An issue in one component could potentially bring down the entire application
- Stuck with a single language



Microservices

Overview of Microservices



Microservices Architecture pattern

- 1 An application written as small interconnected services, each implementing distinct functionality
- 2 Self contained, maintains its own datastores
- 3 Each service may expose a REST API and most services use other services
- 4 Services may also use other Inter-process-communication methods to interact, such as queue etc.

Advantages with MSA

Faster development

Focused development

Easy Deployment

Scalable

Cloud Native

Less Downtime



Drawback of Microservices Architecture

Many Pain Points!!!!

- What the size limit of microservice?
- Inherent complexity of distribution of systems Handling transactions (partial failures)
- Multiple databases
- Need for advanced technology (service mesh, service discovery, circuit breaker, containers, orchestration etc)



MTTR vs MTTD

MTTR - Mean time to Recover?

Client waiting you to fix bugs
in production

by /u/erickvasilev
Programmer Humor / 2020-09-14 13:20



How a Amature vs Pro Diagnose issues?



<https://picture-funny.blogspot.com/2006/09/mechanical-genius.html>



master diagnostic technician Kurt Juergens, of Foxborough



Optimize MTTD - Mean Time to Detect Observability vs Surveillance



Monitoring

Log
aggregation

Alerting

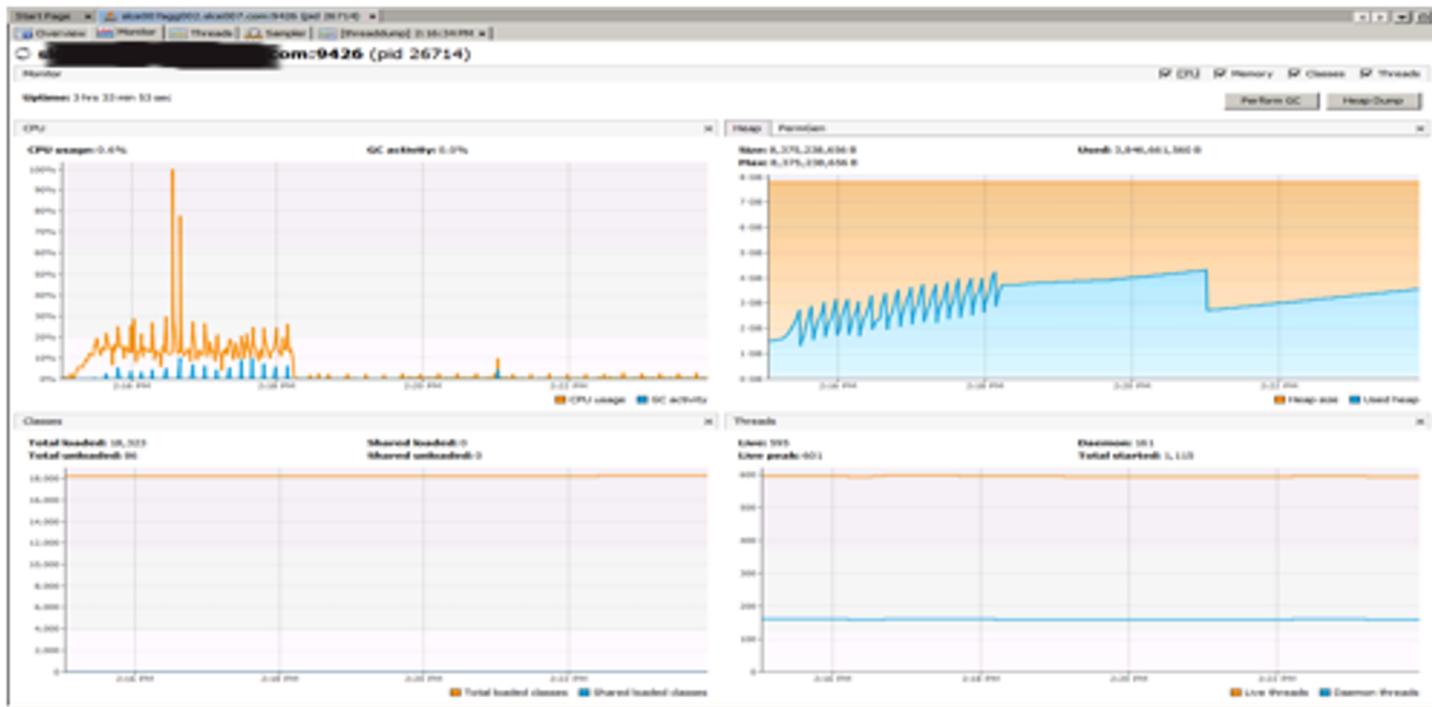
Observability

Visualization

Tracing

People

Monitoring



Monitoring



Monitoring

Application Dashboard

Dashboard Events Top Business Transactions Transaction Snapshots Transaction Score Actions ▾

Business Transactions Service Endpoints Tiers & Nodes Servers Database Calls Remote Services Troubleshoot More Alert & Respond Metric Browser Configuration

Application Flow Map

The Application Flow Map displays the flow of requests between several components:

- 2 JDBC backends → JDBC gateway (401 calls/min, 0.5 ms JDBC)
- JDBC gateway → Web Service (400 calls/min, 6 ms JDBC)
- Web Service → Web Service (140 calls/min, 8 ms JDBC)
- Web Service → APFOYNAMECIMPL...@ubuntu010.04.1 (140 calls/min, 8 ms JDBC)
- APFOYNAMECIMPL...@ubuntu010.04.1 → 2 JDBC backends
- Web Service → wsse2keymanager (60 calls/min, 24 ms Web Service)
- wsse2keymanager → 2 JDBC backends (10 calls/min, 79 ms JDBC)
- wsse2keymanager → JDBC gateway (499 calls/min, 4.5 ms JDBC)

Legend: Load (200), Response Time (ms) (500ms), Errors (1).

Events: Application Changes (0), Business Transaction Health (0 critical, 0 warning, 0 normal), Node Health (0 critical, 0 warning, 4 normal), Server Health (0 critical, 0 warning, 0 normal).

Transaction Scorecard: Normal (100.0 %), Slow (0.0 %), Very Slow (0.0 %), Stale (0.0 %), Errors (0.0 %).

Exceptions: Exceptions (- total), HTTP Error Codes (- total), Error Page Redirects (- total).

Not comparing against Baseline data.

Load: 140 calls, 35 calls/min. Response Time (ms): 349 ms average. Errors: 1 error.

Time: 9:26 AM, 9:28 AM, 9:30 AM, 9:32 AM, 9:34 AM, 9:36 AM, 9:38 AM.

Tracing



Log aggregation

Search Datasets Reports Alerts Dashboards Search & Reporting Save As ▾ Close All time ▾ Job ▾ Verbose Mode ▾

New Search index=apigateway host=*lvsp3!* "already in the process of writing a response while another..."

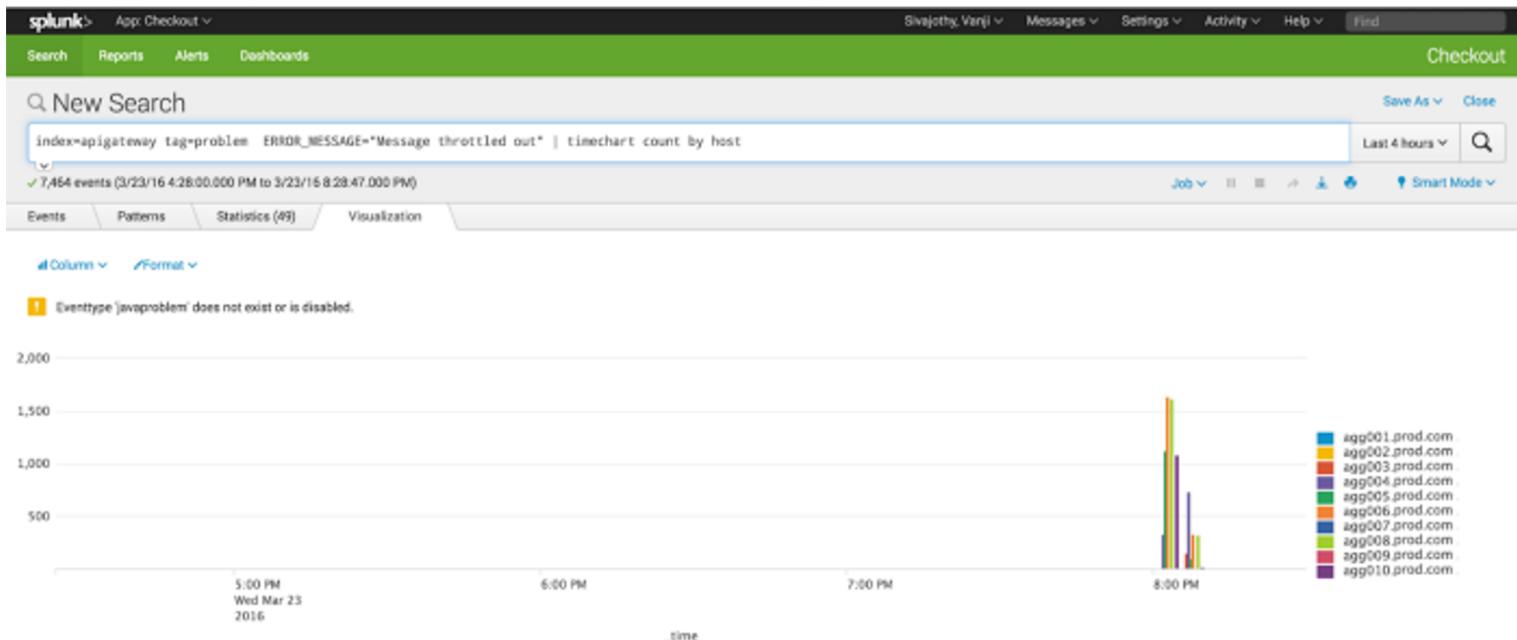
✓ 97 events (before 2/28/17 6:46:45.000 PM) No Event Sampling ▾

Events (97) Patterns Statistics Visualization Format Timeline ▾ 1 hour per column

List ▾ 50 Per Page ▾ 1

Hide Fields	All Fields	Time	Event
Selected Fields	# source 1	> 2/14/17 12:04:52.000 AM	TID: [-1234] [] [2017-02-14 00:04:49,142] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-269035 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log
Interesting Fields	# app_pool 1 # app_pool_index 1 # datacenter 1 # DC 1 # enviro 1 # environment 1 # eventtype 3 # host 2 # host_cage 1 # index 1 # linecount 1 # pod 1 # punct 1 # sourcetype 1 # splunk_server 25 # tag 1	> 2/14/17 12:04:43.000 AM	TID: [-1234] [] [2017-02-14 00:04:42,291] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-269155 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log
		> 2/13/17 11:37:03.000 PM	TID: [-1234] [] [2017-02-13 23:37:02,173] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-291812 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log
		> 2/13/17 11:35:02.000 PM	TID: [-1234] [] [2017-02-13 23:35:00,414] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-264094 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log
		> 2/13/17 11:30:24.000 PM	TID: [-1234] [] [2017-02-13 23:30:21,720] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-288772 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log
		> 2/13/17 11:30:21.000 PM	TID: [-1234] [] [2017-02-13 23:30:21,568] DEBUG {org.apache.synapse.transport.passthru.PassThroughHttpSender} - A Source Connection is closed, because source handler is already in the process of writing a response while another response is submitted: http-incoming-288701 (org.apache.synapse.transport.passthru.PassThroughHttpSender.submitResponse(PassThroughHttpSender.java:617)) source = /var/log/wso2am/wso2carbon.log

Visualization



Visualization

Search API Performance Application Performance API Profiling Exception Sniffer Quality Dashboards Mobile Performance Alerts Quality Monitoring

New Search

Last 3 hours

```
index=apigateway host=[REDACTED] "Error while trying to connect to the endpoint" | eval ERROR_TYPE="CONNECT-TO-ENDPOINT-ERROR" | stats count(start_date) by ERROR_TYPE, _time | append [search index=apigateway host=[REDACTED] "Error while trying to publish events to data receiver" | eval ERROR_TYPE="DATA-AGENT-ERROR" | stats count(start_date) by ERROR_TYPE, _time] | append [search index=apigateway host=[REDACTED] "Send timeout" | eval ERROR_TYPE="SYNAPSE-SEND-TIMEOUT" | stats count(start_date) by ERROR_TYPE, _time] | append [search index=apigateway "[REDACTED]cep" connection refused | eval ERROR_TYPE="CEP-CONNECT-REFUSED" | stats count(start_date) by ERROR_TYPE, _time] | append [search index=apigateway host="lvsp31" "Connection time out after request is read" | eval ERROR_TYPE="CONNECT-TIME-OUT-AFTER-READ" | stats count(start_date) by ERROR_TYPE, _time] | timechart count by ERROR_TYPE
```

✓ 0 events (2/7/17 8:00:00.000 PM to 2/7/17 11:55:54.000 PM) No Event Sampling

Events Patterns Statistics (48) Visualization

Job Smart Mode

Area Chart Format

200
150
100
50
0

8:00 PM Tue Feb 7 2017 9:00 PM 10:00 PM 11:00 PM

_time

CONNECT-TIME-OUT-AFTER-READ
SYNAPSE-SEND-TIMEOUT

Alerting

[REDACTED] Stall Policy

Summary of events occurring during the 1+ minute(s) prior to Wed Feb 08 01:44:36 UTC 2017:

Count	Event Type
1	Health Rule

! [New Critical Health Rule Violation](#)

Wed Feb 08 01:43:55 UTC 2017 | agg-31 | [REDACTED]

AppDynamics has detected a problem with Business Transaction All Other Traffic - [REDACTED]

[REDACTED] Stalls started violating and is now **critical**.

All of the following conditions were found to be violating

For Business Transaction All Other Traffic - [REDACTED]

1) condition 1

condition 1's value **396.0** was greater than the threshold **100.0** for the last 30 minutes



Surveillance

Surveillance

- Targeted Surveillance
- Predefined Rules



Image by [StockSnap](#) from Pixabay

Surveillance

Screenshot of a Java memory monitoring tool interface showing automatic leak detection results.

The top navigation bar includes: Dashboard, My Dashboards, Hardware, Memory (selected), JVM, JMX, Events, and a Help icon.

Sub-navigation tabs under Memory: Heap & Garbage Collection, Automatic Leak Detection (selected), Object Instance Tracking, and Custom Memory Structures.

Automatic Leak Detection status: On (blue button).

Table headers: Drill Down, More Actions ▾, Start On Demand Capture Session, and Showing 3 of 3 objects.

Table columns: Class, Collection Size, Potentially Leaking, Object Creation Time, JVM Start Time, Object Instance ID, Status, and Collection Size Trend.

Data rows:

Class	Collection Size	Potentially Leaking	Object Creation Time	JVM Start Time	Object Instance ID	Status	Collection Size Trend
java.util.concurrent.ConcurrentHashMap	44740	!	Yes	11/16/15 7:00:53	11/16/15 10:42:33 AM	1963828	
java.util.concurrent.ConcurrentHashMap	1137918	!	Yes	11/16/15 7:00:53	11/16/15 10:42:33 AM	1963827	
java.util.concurrent.ConcurrentHashMap	226890	!	Yes	11/16/15 7:00:53	11/16/15 10:42:33 AM	1963826	

Image by [StockSnap](#) from Pixabay

Conclusion

Question Time!

