Application Performance in the Cloud

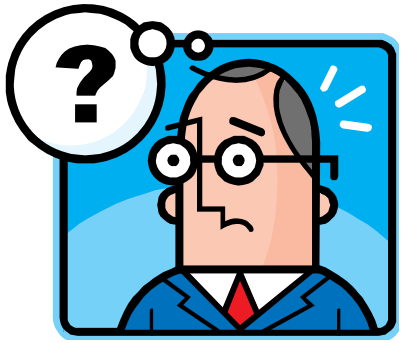
Understanding and ensuring application performance in highly elastic environments

Albert Mavashev, CTO
Nastel Technologies, Inc.
amavashev@nastel.com

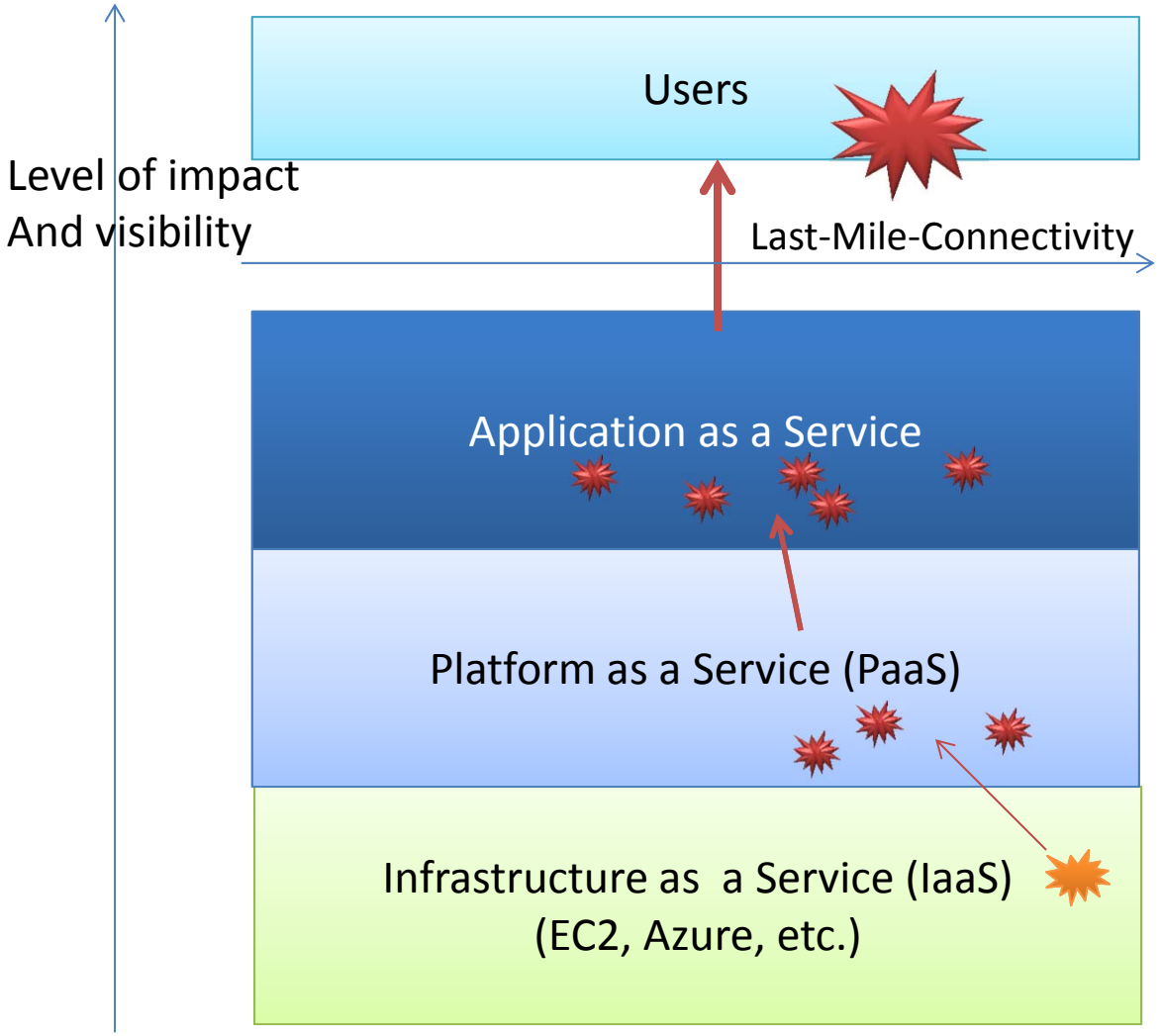
What is Cloud?

- Typically defined by it properties
  - Elastic – on-demand scalability
  - Agile – quick to deploy provision
  - Reliable – business continuity, ability to survive failures
  - Maintenance – single deployment, no need to deploy on each user
  - Independence -- device, hardware, location independence
  - Cost – typically pay-per-use
  - **Performance – could be very well defined for IaaS, less so for PaaS, and even less so for SaaS**
- Clouds can be public, private, hybrid, federated
  - Hybrid and federated are the most complex
  - Public and Private -- off premise vs. on-premise (similar but differ in delivery mechanism)

What does it mean Application Performance in the Cloud?



Application in the Cloud



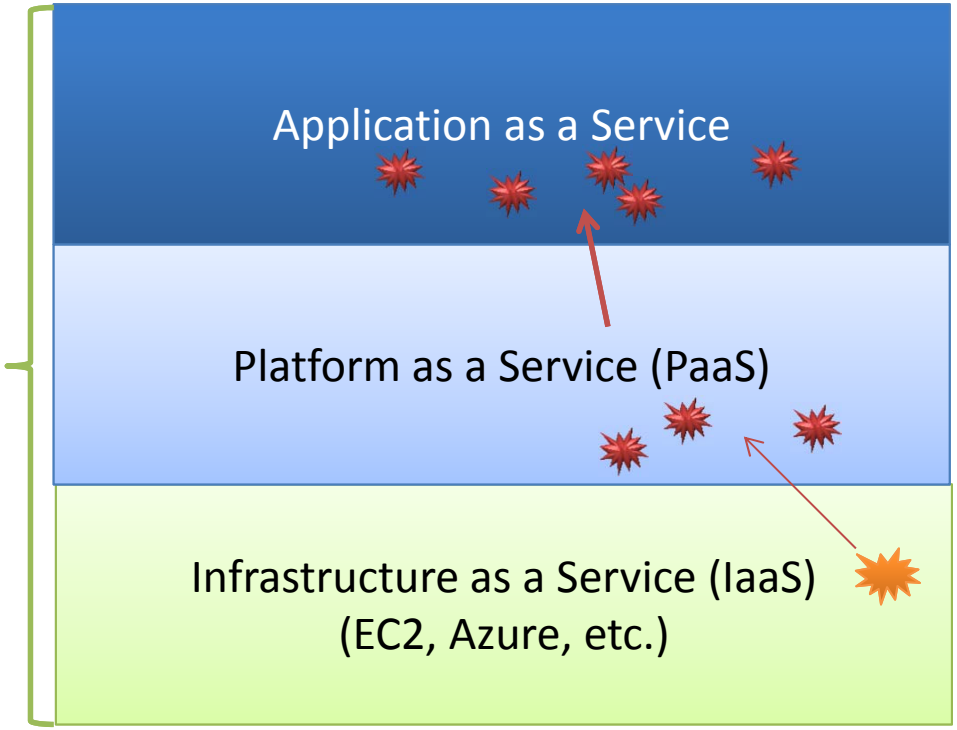
Demanding
Opinionated
Socially connected
Facebook, twitter, etc.

User-experience
Multi-tenancy
Security
Mobility

Clustering
Workload management
Inter-connectivity
Big Data (structured storage)

Virtualization
On-demand provisioning
Resource auto-scaling

Application Performance: Take 1
Monitoring Performance of each stack



Measure user experience
Orders filled, missed
Users serviced, revenue booked
Web page response times

JVM, CLR performance, Clustering
Servlet, web service response times
GC behavior, resource pool utilization
Replication, data availability, data delivery

CPU, Memory, I/O
Storage capacity, utilization
Network bandwidth

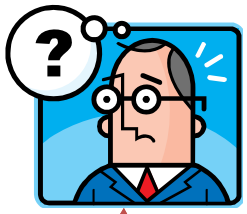
Application Performance: Take 2

What happens when problems are between stacks?

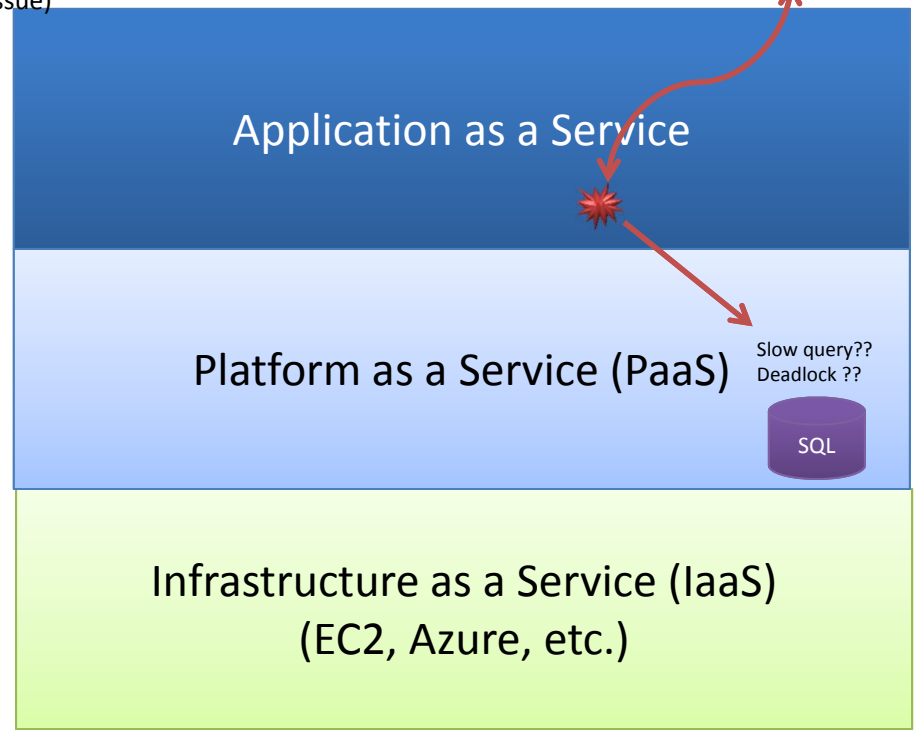
What if performance impact is data dependent?

What about Last-Mile-Connectivity?

(May be not so much in US, but other countries LMC is a big issue)



User experiences a timeout



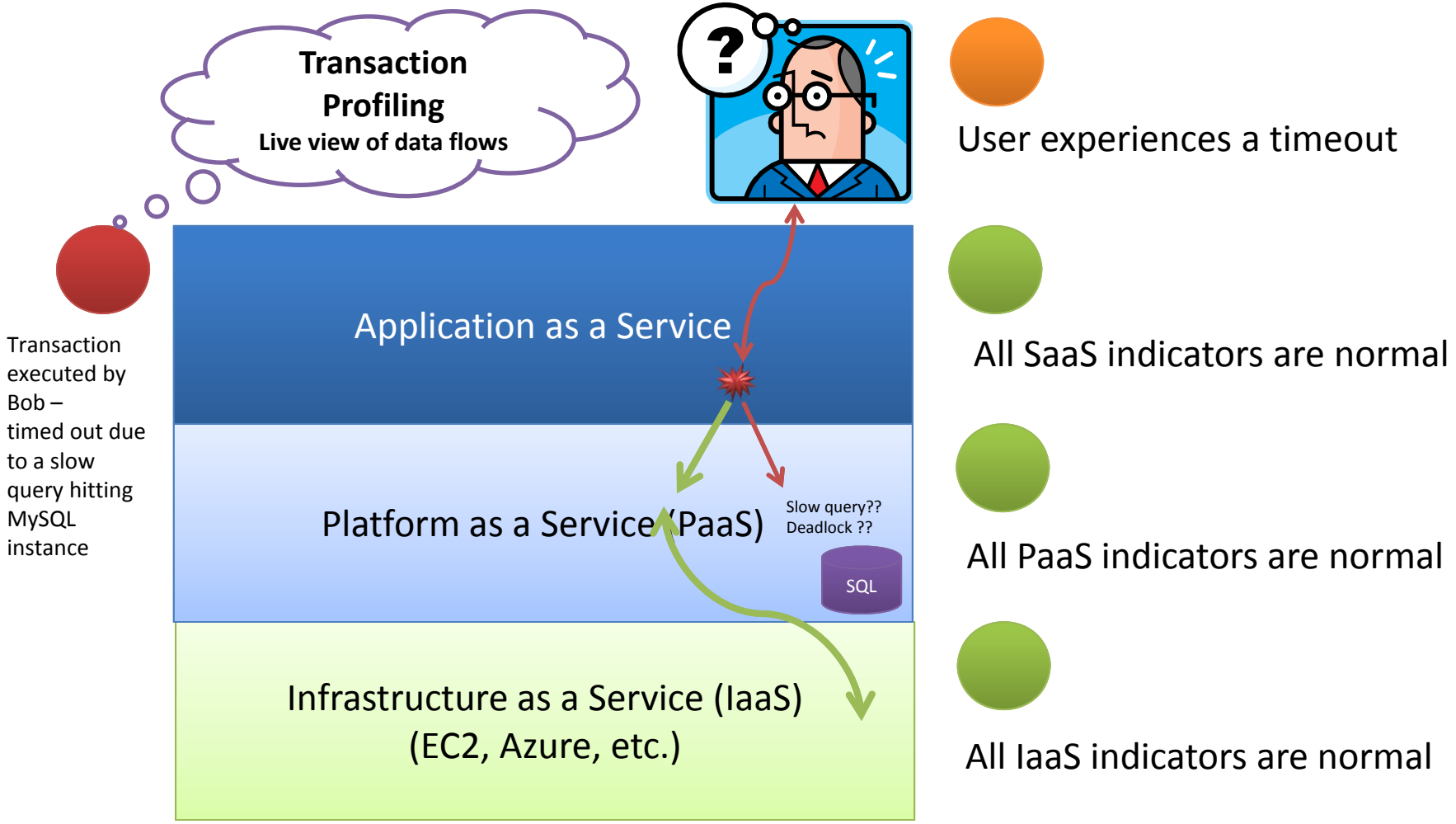
All SaaS indicators are normal

All PaaS indicators are normal

All IaaS indicators are normal

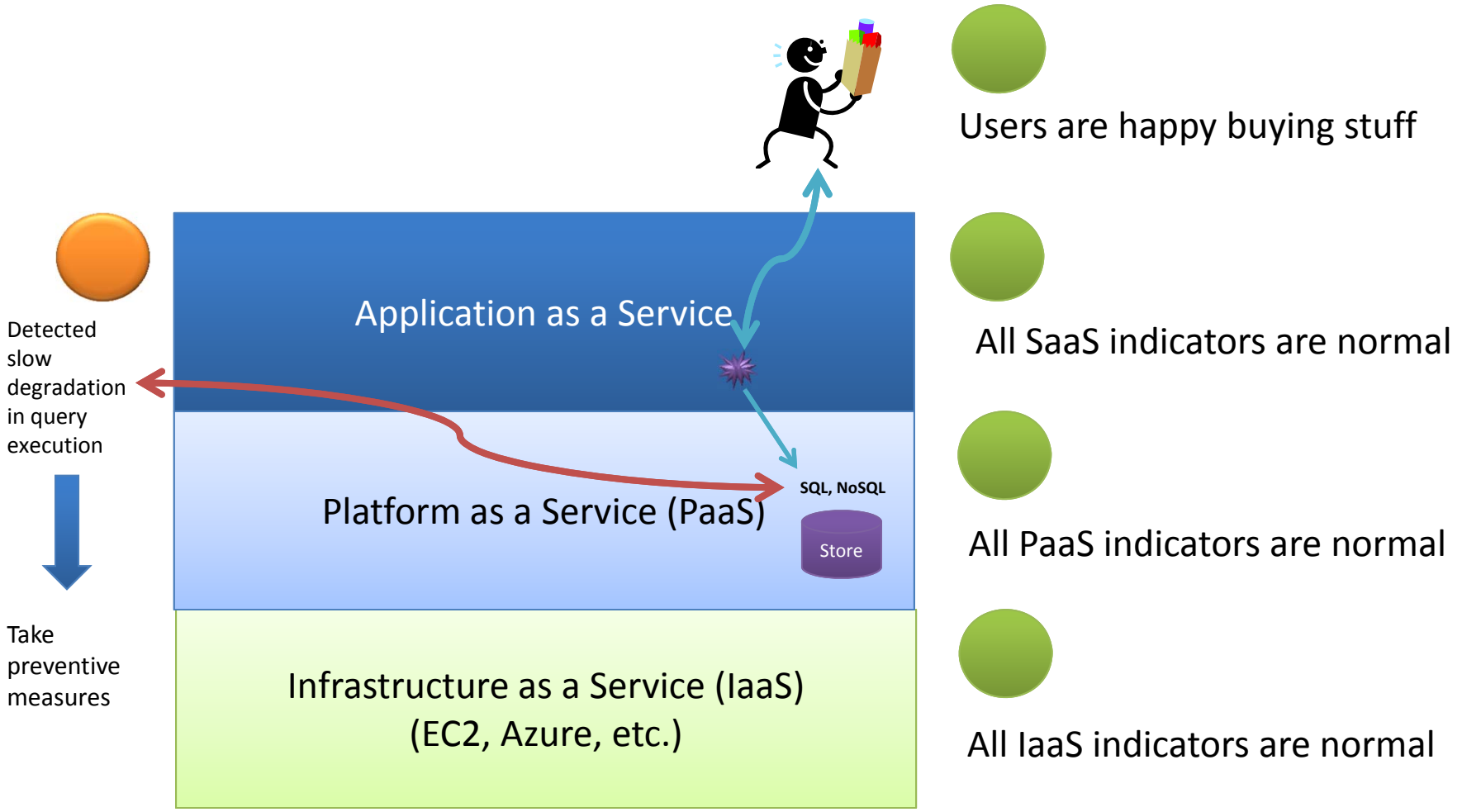
Application Performance: Take 3

Monitoring each stack as well as transactions that flow between stacks?

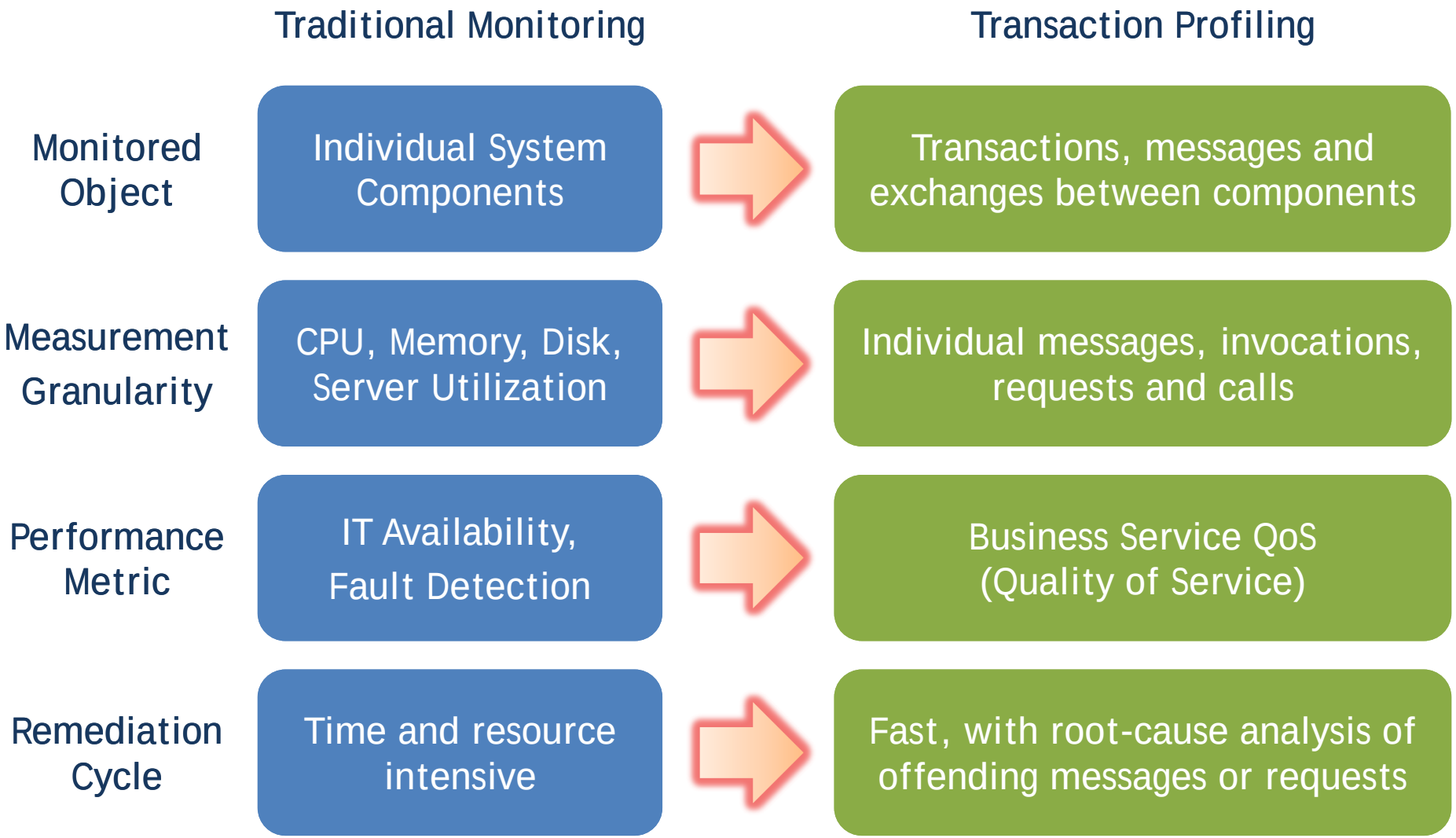


Application Performance: Take 4

What about proactively monitoring trends before service degrades?

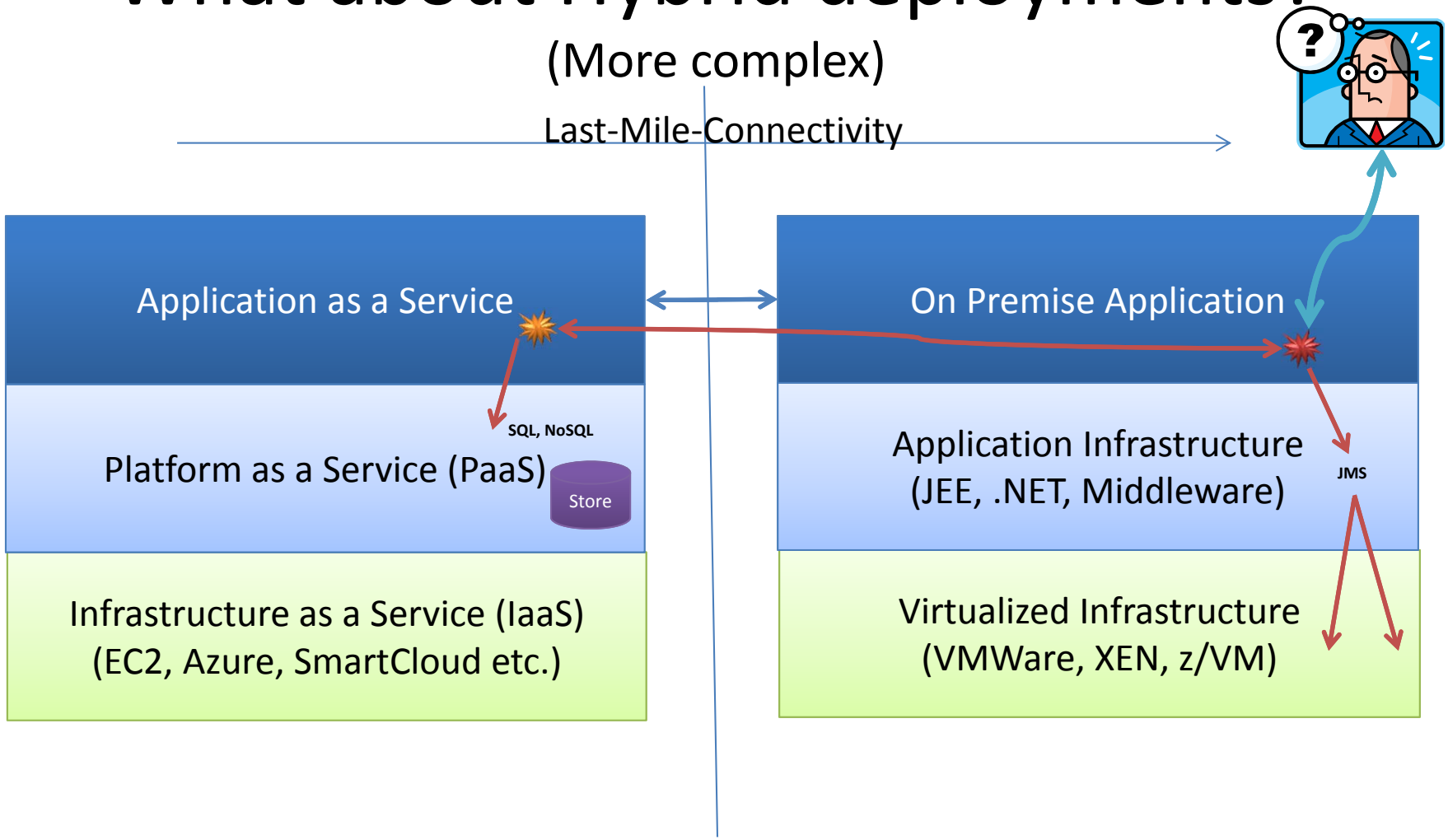


Traditional Monitoring vs. Transaction Profiling



What about Hybrid deployments?

(More complex)



On-premise and off-premise (Cloud Service) resources need to be monitored
Monitoring itself could be delivered as on-premise or off-premise
Monitoring instrumentation must be cloud/firewall ready/friendly (HTTP/HTTPS)
Inter-cloud interactions need to be monitored as well

USE CASE: TREND ANALYSIS

Ways to detect performance trends

- Measure relevant application performance indicators
  - Orders filled, failed, missed
  - JMV GC activity, memory, I/O
- Create a base line for each relevant indicator
  - 1-60 sampling for near real-time baseline
  - 1, 10, 15 min → daily, weakly, monthly for short, long term baseline
  - Samples can range anywhere from 1-60 seconds depending on level of required resolution
- Apply analytics to determine trends and behavior
  - Can vary from simple to complex
  - **Prefer KISS approach (Keep It Simple and Stupid)**

3 Simple methods to detect trends (No complex math required)

- **Bollinger Bands**
  - Determine high and low bands based on available baseline
  - Defines a normal channel which is typically within 2 standard deviations from the mean
  - Compute STDDEV, Mean, Current sample
- **% Change**
  - Sample to sample, day-to-day, week-to-week, etc.
- **Velocity**
  - Number of measured units per unit of time (example: response time drops from 10 to 20 seconds over 5 sec interval – means (20-10/5)=2 units/sec.

Typical Usage

- **High Band**
  - Given a set of metrics, alert when one or more are above High band for at least 2+ samples
  - Indication of abnormal activity over a period of time
  - Caution: abnormal can become the new normal
- **% Change**
  - Useful indicator for near real-time monitoring of resources (such as heap, memory, CPU, storage)
  - Useful indicator for long term trends (daily, weekly)
- **Velocity**
  - Very useful for monitoring metrics that measure usage of resource that have a finite upper bound (memory, storage, table space etc.)
  - Measuring velocity can help measure when upper limits can be reached

Required instrumentation

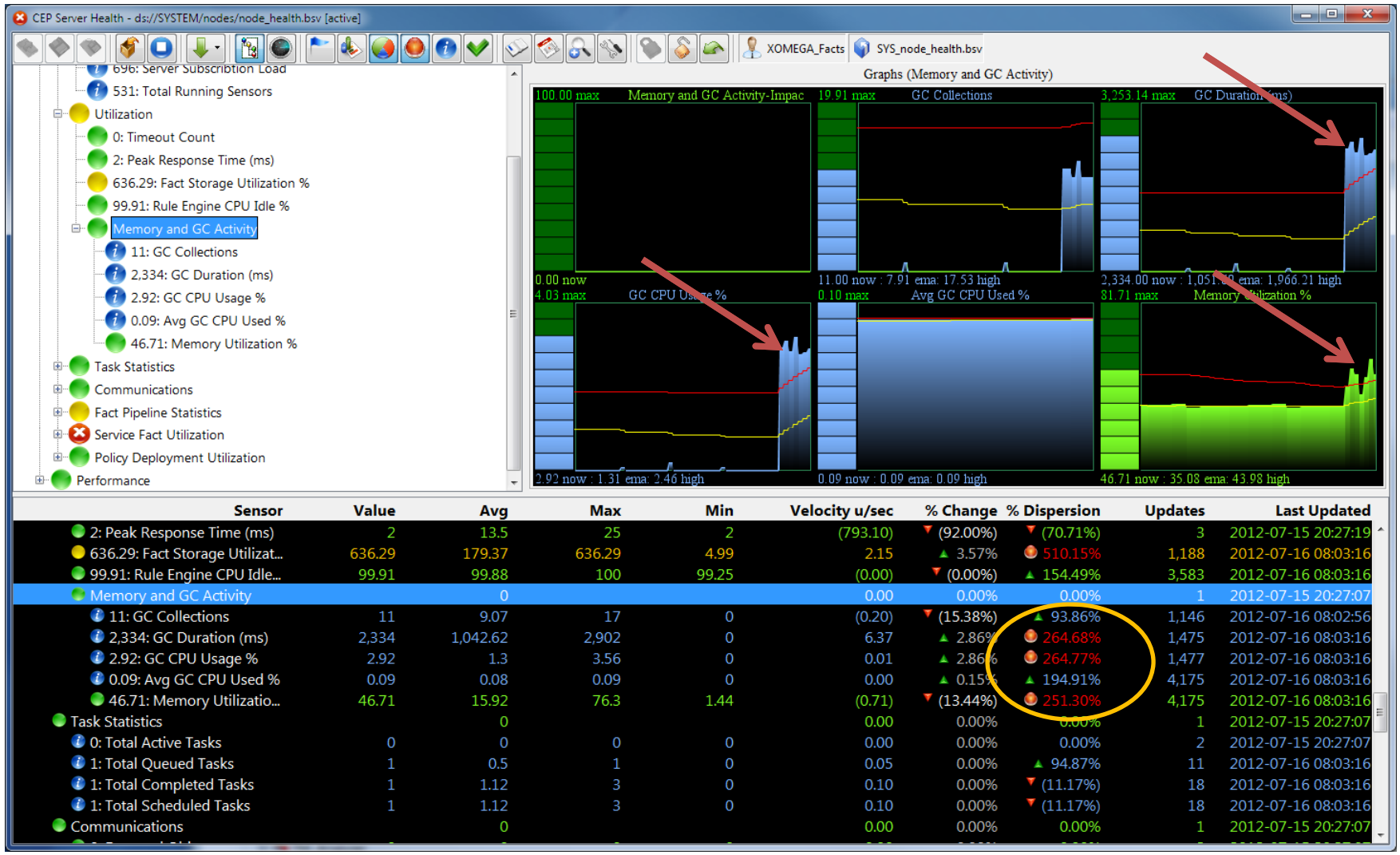
- **Data collectors**
  - Attempt to collect all relevant indicators within the same time tick
    - Response time, GC activity, memory usage, CPU usage
- **Build a history for each collected metric**
  - Either in memory for near real-time analysis
  - Storage for short, long term (min, hours, days)
- **Pattern matching, analytics**
  - Need to scan and pattern match application metrics (such as find all applications whose GC is above High Bollinger Band for 2+ samples)
  - Run as a continuous query, which is executed as metrics are collected and updated
- **Actionable Outcome**
  - Alerts, notifications, actions
  - Visualization, dashboards

Example: Monitoring Java Application by examining GC Activity

- Java Application running in a standalone JVM container
- Monitoring JVM GC (Garbage Collection) as a byproduct of application activity
  - Sample GC every 10 seconds
  - # GC Samples
  - GC Duration (ms.)
  - GC CPU Usage %
  - Avg. GC CPU Usage (since JVM startup)
  - JVM Heap Utilization %

Example 1: Java Application, Sudden Spike in Activity

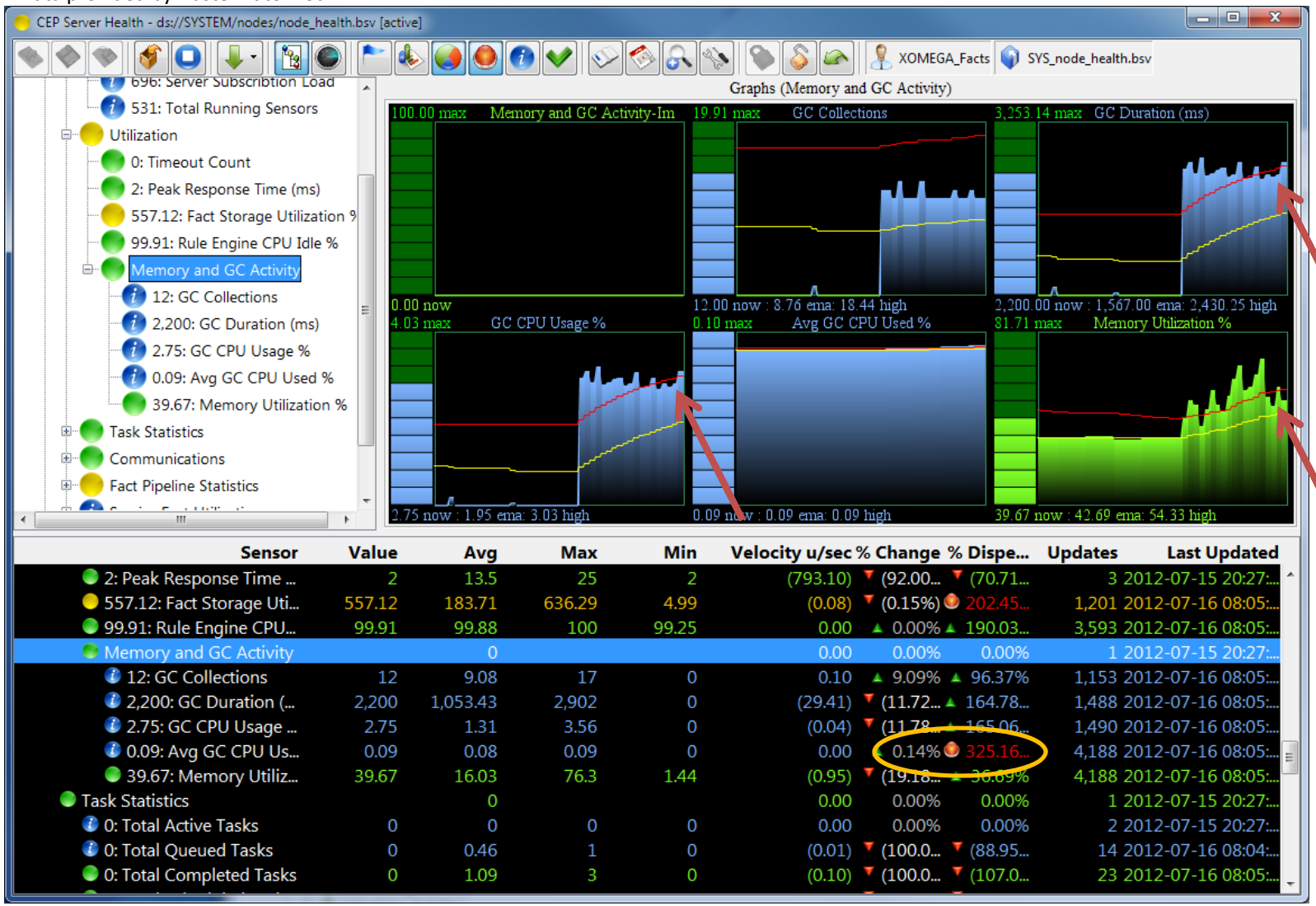
Data provided by Nastel AutoPilot™



Sensor	Value	Avg	Max	Min	Velocity u/sec	% Change	% Dispersion	Updates	Last Updated
2: Peak Response Time (ms)	2	13.5	25	2	(793.10)	(92.00%)	(70.71%)	3	2012-07-15 20:27:19
636.29: Fact Storage Utilizat...	636.29	179.37	636.29	4.99	2.15	3.57%	510.15%	1,188	2012-07-16 08:03:16
99.91: Rule Engine CPU Idle...	99.91	99.88	100	99.25	(0.00)	(0.00%)	154.49%	3,583	2012-07-16 08:03:16
Memory and GC Activity		0			0.00	0.00%	0.00%	1	2012-07-15 20:27:07
11: GC Collections	11	9.07	17	0	(0.20)	(15.38%)	93.86%	1,146	2012-07-16 08:02:56
2,334: GC Duration (ms)	2,334	1,042.62	2,902	0	6.37	2.86%	264.68%	1,475	2012-07-16 08:03:16
2.92: GC CPU Usage %	2.92	1.3	3.56	0	0.01	2.86%	264.77%	1,477	2012-07-16 08:03:16
0.09: Avg GC CPU Used %	0.09	0.08	0.09	0	0.00	0.15%	194.91%	4,175	2012-07-16 08:03:16
46.71: Memory Utilizatio...	46.71	15.92	76.3	1.44	(0.71)	(13.44%)	251.30%	4,175	2012-07-16 08:03:16
Task Statistics		0			0.00	0.00%	0.00%	1	2012-07-15 20:27:07
0: Total Active Tasks	0	0	0	0	0.00	0.00%	0.00%	2	2012-07-15 20:27:07
1: Total Queued Tasks	1	0.5	1	0	0.05	0.00%	94.87%	11	2012-07-16 08:03:16
1: Total Completed Tasks	1	1.12	3	0	0.10	0.00%	(11.17%)	18	2012-07-16 08:03:16
1: Total Scheduled Tasks	1	1.12	3	0	0.10	0.00%	(11.17%)	18	2012-07-16 08:03:16
Communications		0			0.00	0.00%	0.00%	1	2012-07-15 20:27:07

Example 2: Java Application, Adjustment to new workload – The New Normal

Data provided by Nastel AutoPilot™



Resource Leak Detection

Detecting Leaks using Trend Analysis

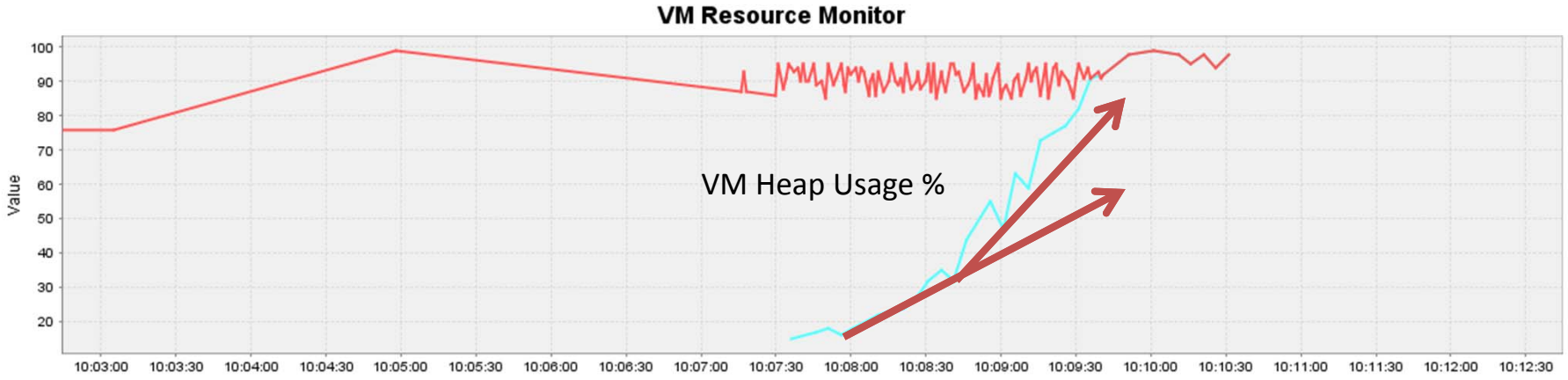
(Java Example)

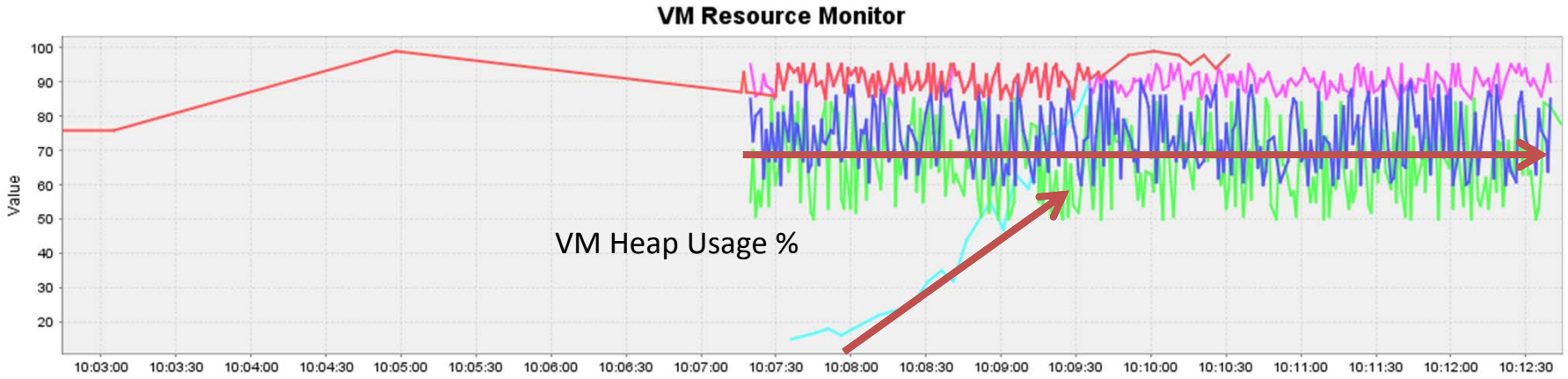
Typical causes of Java leaks

- Programming errors, bugs
  - Unchecked array, list, hash map growth
  - Not closing JDBC Prepared Statements
  - Not closing Sockets, File handles
  - Thread leaks, handle leaks
  - Class loader leaks
  - Resources allocated outside JVM

Leaking Chart Pattern – Detecting

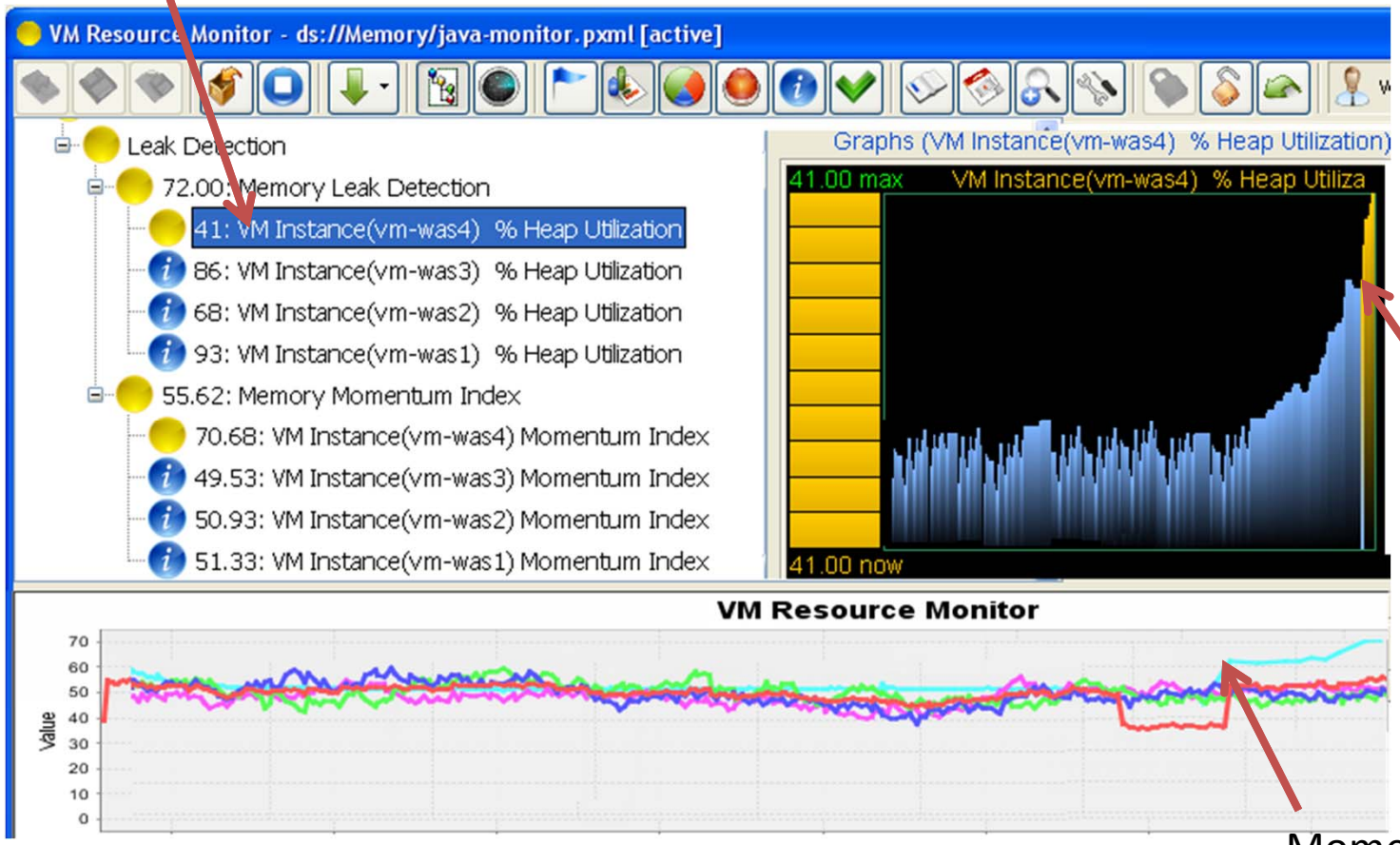
Resource Accumulation





Detecting Resource Leaks using Momentum Oscillator

Heap not yet exhausted



Leak pattern detected

Momentum Oscillator Trending higher

Momentum Oscillator: values between 0-100, difference between the sum of all recent gains and losses in the underlying metric. Value of 50 means that the net difference of gains and losses is zero – 0 net gain and loss.

Conclusion: Monitoring Elastic Environments

- **Elastic Applications can't be monitored using static models**
  - Static thresholds
  - Static data/transaction flow models
- **Complex systems layered on top of complex systems**
  - Too many constantly changing variables
  - Makes root cause analysis very difficult
  - Requires extensive cross technology expertize
- **Preferred approach – Holistic Application Monitoring**
  - Granular data collection:
    - Application and infrastructure metrics
  - Analytics, automated base lines
    - Real-time and historical
  - Resource monitoring coupled with Transaction Profiling
  - Visualization that connects different teams:
    - Application support, DevOps, IT Support

For more information

- Visit us at:
  - www.nastel.com
- Questions:
  - info@nastel.com
- Twitter:
  - twitter.com/nastel
- FaceBook:
  - facebook.com/NastelTechnologies
- LinkedIn:
  - linkedin.com/companies/nastel-technologies
- Phone:
  - +1.800.580.2344