“If You Graph It, They Will See It”
Identifying Root Issues From Product Testing to Production Crisis.
Splunk@Murex
For Test and Development

Robert Lynch | Global Splunk Manager & Performance Manager
September 2017 | Washington, DC
Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are updatesmarks and registered updatesmarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or updatesmarks belong to their respective owners. © 2017 Splunk Inc. All rights reserved.
1. Who is Robert Lynch?
2. A little about Murex
3. Murex in-house solution had limitations!
4. How Murex uses Splunk for test and development
5. Some Use Cases:
   • The trillion dollar problem
   • Millisecond analysis
   • So, where is my test?
Who Am I?

- **Name:**
  - Robert Lynch

- **Current Position:**
  - Global Splunk Manager @ Murex
  - Non-Functional Test Manager @ Murex

- **History:**
  - 13 years working at Murex in Non-Functional Testing
  - Masters “High Performance Computing”
  - Introduced Splunk to Murex 3 Years ago
Murex: Leading Vendor in Financial Software

- **Murex** is a leading vendor in the provision of integrated solutions for front office, back office and risk management for the capital markets industry
  - 30 years dedicated to our domain and clients
  - Around 45,000 users and 250+ clients in 65 countries
  - 3 Regional Hubs, 17 offices covering all time zones
  - Staff of 2,000, over 60 nationalities
  - Revenue of USD 0.5bn in 2016
  - Over USD 1 billion invested in R&D over the last 10 years
  - #1 Overall Vendor: Risk Magazine Technology Awards 2016
  - #7 in Chartis RiskTech100 for 2017 (highest ranked non-US headquartered company)
A Worldwide Presence

3 REGIONAL HUBS, 17 OFFICES COVERING ALL TIME ZONES

45,000+

COUNTRIES

USERS
Splunk Deployment At Murex

LIVE Environments

*20 Second to add new Environment

Log Drops From Clients

Internal Testing

Splunk

Any Murex Employee
What is TPS tracing?

- TPS is light performance tracing across any JAVA service
- Code level timing on the Class.Method
  - Class(X).Method(Y) does action
  - It calls Class(A).Method(B) 6000 times

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th>Average</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class(X).Method(Y)</td>
<td>1</td>
<td>5 Minutes</td>
<td>5 Minutes</td>
</tr>
<tr>
<td>Class(A).Method(B)</td>
<td>6000</td>
<td>50 ms</td>
<td>5 Minutes</td>
</tr>
</tbody>
</table>

Time (Measured)

- 5 Minutes
- ~50ms

TPS event

22:05 to 22:10
Murex in-house solution had limitations!

- TPS Report was a PDF = static report
- The existing PDF report was slow to graph millions of lines
- We could not zoom into areas of investigation
- Experts were always needed for analysis
- Introducing the new TPS_VIEW in Splunk
“If You Graph It, They Will See It”

- When you graph an issue, it becomes more obvious
- Sometimes “Average” and “Mediums” and “Maximums” are not enough to see issues…
- In the below graph an obvious increase in the “class.method” over time can be seen
- This might not have been obvious without a visualization
TPS Real-Time Activity: Class = * -- GANTT Option will appear for 5 Minutes or Less

Display By: Duration (ms)[Default]

Span (how much time per bucket) Auto

Log vs linear - Y Axis linear

Real Time vs Index INDEXED

Time Selection Time

No Of Buckets 20 Default
What is a **Splunk_ID**?

- Splunk_ID is a reference to your data in Splunk
- A numeric code that will reference your “Test Data” @ “Time” = Splunk_ID

![Standard Time/Data Graph](image)

- This means we can create URL’s to a specific “Time” in a set of “Test Data”
- This is used by developers/testers to quickly share data with one another
Use Case 1: “The One Trillion Dollar Problem”

▶ What is a “Central Counterparty Clearing House” (CCP)?
  • “An organisation that becomes the counterparty to the buyer and the seller of a trade and guarantees the terms of that trade even if one party defaults on the agreement”
  • After 2007-2009 Financial Crash clearing became a regulatory priority

▶ Murex has a large CCP client in Europe were it needed to perform 60 position updates per second as part of its clearing and risk process
  • 17 Milliseconds per update (1000 Milliseconds = 1 Second)

▶ This client can clear up to One Trillion Dollars Notional Daily!
Use Case 1: Overlapping Different Data Sets

- By using **Splunk** we were able to overlap different data sets
  - TPS Tracing with Garbage Collection (GC) [GC is native to Java to clean memory, however if not handled correctly it can slow down performance]

- Below we can see the GC Overlapped with TPS
  - The **green line** is updates Insertion (COMMIT), a lot of times in parallel we can see purple, yellow or red lines, this is “Java GC” slowing down the COMMIT!
Use Case 1: Before optimizations

▶ In Splunk we could add up all the GC time divided by the TPS
  • Par_new_%_of_TPS = \( \frac{\sum \text{ParNew(GC)}}{\sum \text{TPS}} \)
  • Par_new_%_of_total_time = \( \frac{\sum \text{ParNew(GC)}}{\text{Duration}} \)

▶ Test 1: Before any optimizations
  • The average was 96 milliseconds [That is 10 updates per second, we needed 60!]
  • Par_new_%_of_TPS = 37% [Major bottle neck identified]
  • Par_new_%_of_total_time = 20% [Major bottle neck identified]
Use Case 1: After optimizations

- After multiple binary fixes and garbage collection tuning we hit the 17ms mark
- Some GC peaks are still present, however massive improvement was seen
  - We can see from the graph below that the GC peaks in yellow have reduced and we have also reduced the COMMIT peaks
  - Par_new_\%_of_TPS = 37% Before, 7.8% After
  - Par_new_\%_of_total_time = 20% Before, 0.89% After
Use Case 2 – Millisecond analysis

- Viewing the code line in chronological order, helped to improve visualizations
- We can zoom into the Millisecond to see the sequence of method calls
Use Case 2: Sequential vs. parallel testing

During product development we have to stress code to make sure it can take Parallel load

Sequential Import:

- `#addpoint` 734ms
- `#storeExisting` 73ms
Use Case 2: Sequential vs. Parallel Testing

- **Parallel Import: 15 Imports in Parallel**
  - `#addpoint` Increased from 734ms to 2,063ms => 3 time slower – Not Bad!
  - `#storeExisting` Increased from 73ms to **1,236ms** => 17 Times slower – Possible Issue Found!

### TPS Real-Time Stats TPS.Class = * [For the time-period selected in the chart above]

<table>
<thead>
<tr>
<th>FullyQualifiedMethod</th>
<th>average</th>
<th>count</th>
<th>stddev</th>
<th>median</th>
<th>perc95</th>
</tr>
</thead>
<tbody>
<tr>
<td>murex.risk.tsr.repository.dao.AbstractRepositoryService�行ExecuteSeriesQuery</td>
<td>4.90</td>
<td>15</td>
<td>1.8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>murex.risk.tsr.repository.dao.AbstractRepositoryService执行storeExisting</td>
<td>1,236.80</td>
<td>6</td>
<td>19.5</td>
<td>1243</td>
<td>1251</td>
</tr>
<tr>
<td>murex.risk.tsr.repository.dao.AbstractRepositoryService执行storeInstant</td>
<td>1,035.70</td>
<td>15</td>
<td>234.6</td>
<td>1216</td>
<td>1269</td>
</tr>
<tr>
<td>murex.risk.tsr.repository.dao.AbstractRepositoryService执行storeNew</td>
<td>797.30</td>
<td>7</td>
<td>76.9</td>
<td>821</td>
<td>853</td>
</tr>
<tr>
<td>murex.risk.tsr/riskfactors/rest/application point.EquityVolPointRecordService执行#addPoints</td>
<td>2,063.60</td>
<td>9</td>
<td>19.2</td>
<td>2051</td>
<td>2090</td>
</tr>
</tbody>
</table>
Use Case 3: So Where Is My Test?

▷ Over a large timeline we need to be able to identify different tests quickly
▷ Overnight a user might run 10 different tests
▷ By developing an overlap it becomes obvious where each test is (Test 1 or Test 2)
Conclusion: The **Splunk “TPS View”**

- **Problem:**
  - We needed to view millions upon millions of TPS tracing quickly

- **Problem:**
  - We needed to be able to get data into Splunk quickly

- **Problem:**
  - We needed to overlap different data sets to find bottlenecks

- **Problem:**
  - How can I show my colleagues quickly what I am looking at?

- **Solution:**
  - How about….100 million lines of data (sliced and diced) in 10 seconds!!
  - New TPS VIEW is hyper accelerated to handle heavy volumes

- **Solution:**
  - **LIVE Monitoring:** 20 seconds to attach any environment to Splunk (one line command)
  - **Log DROP:** Upload logs from anywhere (one line command)

- **Solution:**
  - By overlapping TPS tracing with GC, CPU etc.. new bottle necks became obvious:

- **Solution:**
  - **Splunk ID:** Unique ID to allow you to save a point of investigation and pass it on to you colleagues
Thanks

Don't forget to rate this session in the .conf2017 mobile app
Appendix - Simple TPS Example

- Annotation for basic classes and methods for TPS tracing to produce traces into a file.

```java
class Example{
    @Traced
    public void exampleMethod() {
        doWork();
    }
}
```

Output: in “json”

```
"endTime":1473177098093,
"startTime":1473177098031,
"operationIdentity":"exampleMethod",
"name":"Murex.Example"
```
Appendix – SPLUNK ID

- A SPLUNK ID is stored in a Lookup Table, there are 4 main parts
  1. ID = Unique ID numeric value
  2. Host = What data set are you looking at
  3. Start_Time = Start time of the time slice
  4. Stop_Time = Stop time of the time slice
- Additional Information can be stored and displayed if needed
Appendix - Charts

Using the TimeLine app it was possible to get the below visualizations
Disclaimer

Copyright © 2017 Murex S.A.S. All rights reserved. All intellectual property rights and other proprietary rights in and associated with the whole and every part of this presentation (including all text, logos, graphics and images) shall at all times remain vested in Murex S.A.S. or its affiliates and shall not be used without Murex’s express authorization. You shall do all that is necessary to protect Murex’s rights, including but not limited to, taking all measures necessary to keep the content of this presentation confidential and not, directly or indirectly, using or divulging, or allowing to be used or divulged such presentation to or by any third party. In addition, you shall not reproduce, copy, distribute, republish, download, display, post or transmit this presentation or any part thereof in any form or by any means whatsoever. The information in this presentation is provided by Murex “as is” and without any representation or warranty, express or implied, including with respect to accuracy or completeness. Any unauthorised use of any content contained in this presentation may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes. If you are aware of any unauthorised use affecting our rights and interests in and associated with this document, you will immediately notify Murex S.A.S..