JAGGER: INDUSTRIAL-STRENGTH PERFORMANCE TESTING AND TRACKING

PERFORMANCE TRACKING IN LARGE PROJECTS

Deployment of a new system or module to production, releasing of a new version is always a bunch of risks. Among them, risks related to software performance and stability are ones of the most wild, unpredictable, and dangerous since they directly threaten system availability. Despite of variety of tools for performance testing there is a number of topics that remain challenging in many projects:

- How to estimate required capacity of the production hardware?
- How to extrapolate results measured on performance testing environments to production environment?
- How to make sure that functional behavior of the system will be correct under heavy workload and with large data volumes?
- How to prepare the system to the huge diversity of real-life failures and maintenance operations?

Moreover, classical approach for performance testing leaves many other open questions when we include into the conversation not only production system stability but delivery risks and development costs:

- How to make sure that system performance is continuously tracked and delivery won't be threaten by issues revealed right before release?
- How to detect a moment when system performance degraded due to some inaccurate change?
- How to minimize efforts on performance results analysis, issues investigation and fixing?
- How developers can obtain undistorted insights into an application (CPU and memory profiles) which are close to what will be in production?
- How to minimize risk of testing incorrectly deployed of poorly configured system and check that obtained results are valid?

GridDynamics, using its vast consulting experience in area of high-performance applications and delivery processes, addresses all these questions in one system for industrial-strength performance testing and tracking – Jagger.

JAGGER IN A NUTSHELL

Jagger is an application suite that was designed for continuous performance testing, monitoring and passive performance measurements, results warehousing and analysis. The logical schema of Jagger is shown in the figure below. As most of performance testing systems, Jagger provides a comprehensive set of features for workload generation, statistical analysis of the collected metrics and so on. But, of course, this does not address the issues that were described in the previous section. The following items provide an overview of features that help Jagger to achieve its goals and rarely found in the competing solutions:

Continuous testing. All processes in Jagger are completely automated, performance testing sessions are typically executed by a schedule, on daily or weekly basis. Testing results are automatically compared with baselines and thresholds using pluggable strategies, alerts are rises in case of inappropriate deviations. This helps to reveal performance issues as soon as possible and leaves time for investigation and fixing.

Embedded monitoring and profiling. Jagger is bundled with subsystems for monitoring and profiling. Collected information like CPU utilization, JVM heap usage, hottest methods in the application under test is an integral part of the reports. These capabilities dramatically facilitate investigation and fixing of performance issues and memory leaks detection since developers clearly see root cause of the problem, not only fact of performance degradation. It is worth noting that profiling information collected directly on performance environment, not local developers' environments, hence profile is much less distorted and more close to the real production situation. Another important notion is that availability of monitoring information inside a performance test allows advanced workload management and capacity testing, for example test scenario can be specified like *"find maximal workload that can be sustained having load average of the system under test less then 2"*

Results warehousing. As apposed to many other solutions, Jagger is designed not like a single-user tool, but like a server that processes all submitted jobs in a centralized fashion. All results of the performance testing, both raw, like a duration of each transaction, and aggregated, are stored inside Jagger and external database. All results can be browsed via web interface, reports can be generated after each test session. This guarantee completeness and traceability of test results even in large multi-team projects, availability of data for all participants of testing and development.

Production performance tracking and passive measurements. Jagger offers both active performance testing, i.e. workload generation, and passive measurements, i.e. registration and statistical processing of events that come from a process that is already workloaded – like a production system. This allows to combine under one roof all performance statistics and profiling information, both from test environments and production deployments. This facilitates mapping of performance testing results to production system, increases accuracy of hardware capacity planning, and mitigates risk of insufficient performance in production.

Simulation of failures. Maintenance operations (restarts, backups etc.) and hardware problems like network delays often cause production failures because testing of system robustness against such issues is challenging. Jagger is bundled with tools that allows to simulate network packet losses, restart nodes, call custom maintenance operations like data reload as a part of performance test scenario. As a result, many real-life situations can be covered in automatic tests, mitigating risks of production failures.

Horizontal scalability and integration with Hadoop. Jagger is designed as a distributed shared-nothing platform to provide excellent horizontal scalability both in terms of generated workload and volume of collected and stored data. Jagger leverages Apache Hadoop technologies to organize distributed data storage and provide ability to analyze huge amounts of the collected data.

Functional validation. In many projects, functional testing and performance testing is done by separate teams. This leads to a gap where functional defects that become visible only under workload sit, like non thread safe operations. Jagger provides ability to validate responses returned under workload using both custom checkers and automatic capturing of the expected results before workloaded test. This allows to reveal a whole class of issues that are often discovered neither by functional testing team nor by performance team.

Integration with cloud environment managers. Although deployment of the system under test to the test environment is not a part of Jagger, Jagger, due to high level of automation, can be combined with deployment and environment management solutions to form an end-to-end performance testing pipeline, as it shown it the picture below. Such continuous and entirely automatic process minimizes risks of human mistakes and allows organization of testing in really large projects.