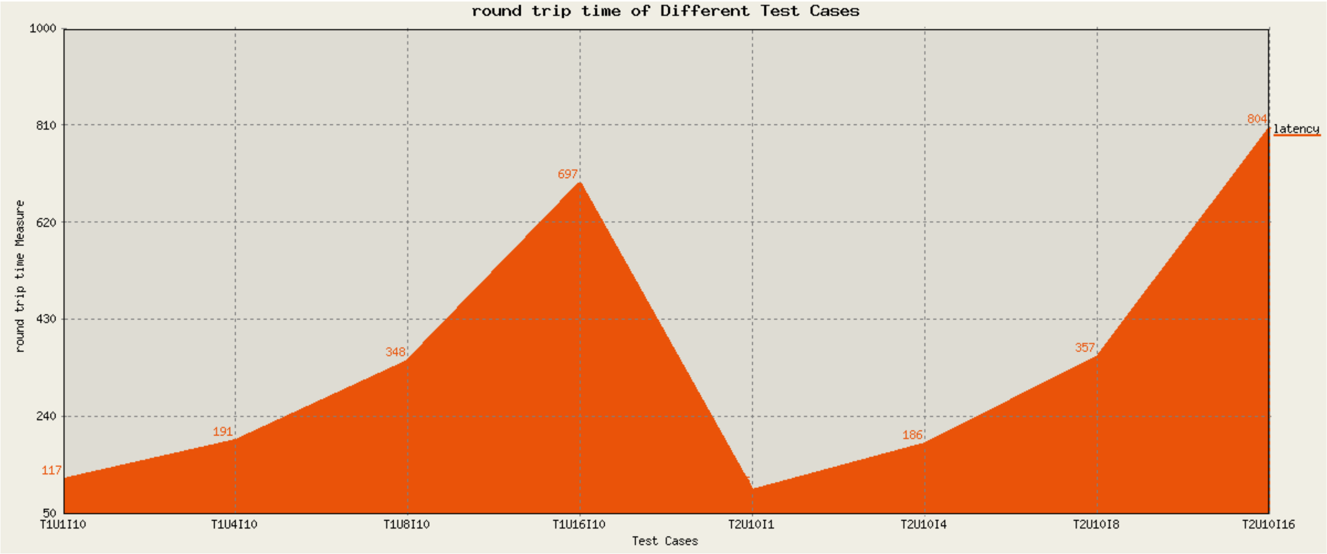
Performance and Automature

Automature provides a robust, unique, and extendable architecture for studying application and system performance, leveraging the concept of atoms and molecules, that is core to the design of its automation product suite. What this means is that the user gets a core set of performance tools, that provide charts and metrics out-of-the box with very little customization required. It also means that all the charts and metrics can be expanded easily by the user, and user-defined metrics can be charted just as easily by providing simple to write “sensor-atoms”, that know how to access the specific metrics the user is interested in.

What is Performance Characterization?

It is important for us, at this point, to share how we view application performance characterization, and establish some ground rules about Automature's offerings in this space.

At its core, studying performance is equivalent to studying the behavior of certain metrics, when some variables change. One example of a metric could be the time it takes an application to respond to a specific type of user request. An example of a variable could be the number of users concurrently sending that request. In the chart below, the response time is plotted for two different types of requests, for an increasing number of simultaneous users.

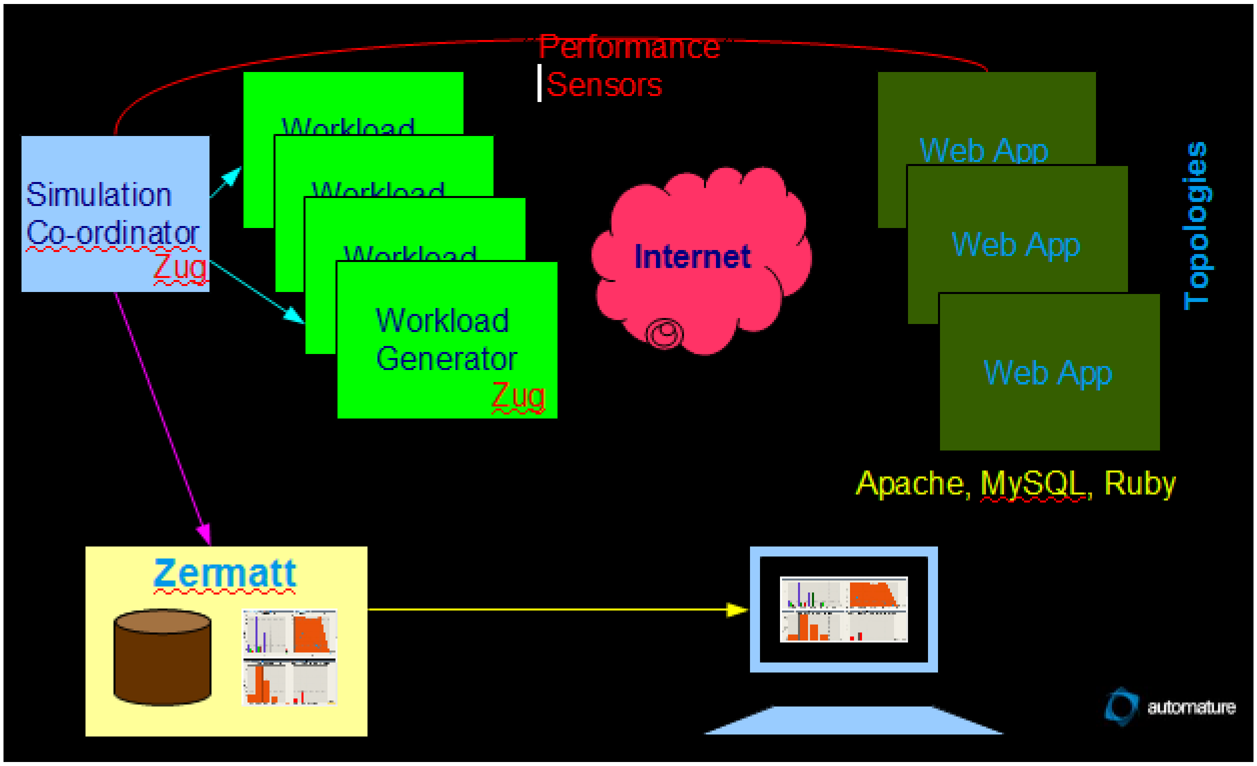


The goal of a performance study is to find which variables have the most influence on the behavior of the chosen metrics, with the intent to forecast how the system would behave in a production environment, so that the application, or the system can be tuned in advance to meet the demands placed on it, under specified loads.

To state it another way, the goal is really to stress test the application, to verify that it will meet its performance targets, but done in a way, that exposes the exact nature of its failings in a manner, that can be acted upon.

Design goals for this architecture

By clearly defining the responsibility of each component participating in a study, this architecture has been specifically designed to maximize the flexibility by allowing users to enhance the performance characterization in many ways, without having to re-engineer major parts of the system. The load simulation (i.e. the specific mix of the transactions) is completely decoupled from the specific metrics that need to be measured, as well as the variables that need to be modified in each experiment. The user can add to one, without having to worry about the other aspects of the experiment.



How do we measure?

In order to do measurements, we need to simulate a load, consisting of a typical mix of the types of user requests, that we expect the system to be able handle.

The HW setup consists of the following components, viz.

- One Load Coordinator
- One or more workload Simulators
- A server, or server farm where the application under study executes.

The assumption here is that, typically, the requests can be made through web services, or simulating browser based interactions, though other application types can also be supported.

The Load Coordinator

This is the machine, designated to act as a master to the workload simulators. Its role is to orchestrate the load, by distributing the responsibilities to a group of workload simulators, waiting until the job is complete, and then probing the machines, whose behavior was the target for the study (i.e. the machine where the application is deployed) to recover the metrics.

It needs to be configured with the following settings, viz.

- The number of workload simulators
- The number of concurrent users each workload simulator will emulate
- The location of the server system, where the application under study is running
- The processes on the server system, deemed to be significant to the study
- The metrics of those processes, we need to monitor
- The specific types of request that will need to be simulated

Workload Simulators

These are the machines, which are responsible for generating the requests, which the application will need process and to respond to. The coordinator tells each machine how many concurrent users should be simulated, and how long the simulation needs to run. The workload simulators do not need to be aware of the existence of their peers. At the end of the simulation, each machine reports its findings back to the coordinator.

The simulators get their marching orders from the coordinator. The simulators are identically configured, and work on a preloaded set of request types, sent to them by the coordinator. They are not aware that they are participating in a performance characterization study.

The metrics

- Response time
- CPU usage by system logic
- CPU usage by application logic
- Disk I/O rate by specific processes
- Network I/O rate by specific processes

The variables

- Request type
- Number of Users
- Type of System
- Application Version

The charts

Chart What	Compare What? (X-Axis)	Individual Metrics (Y-Axis)
For each specific application release and workload	All selected topologies (varying number of CPU cores)	Response Time, CPU usage, Disk IO, Network I/O
For each topology and application release	All different workloads	
For a given workload or scenario	Multiple application releases across each topology	
	Multiple application releases across all topologies (avg, min, max)	

The following are examples of some of the charts that are available for viewing after each simulation.

Chart to compare a metric for a specific workload type across multiple releases on individual topologies

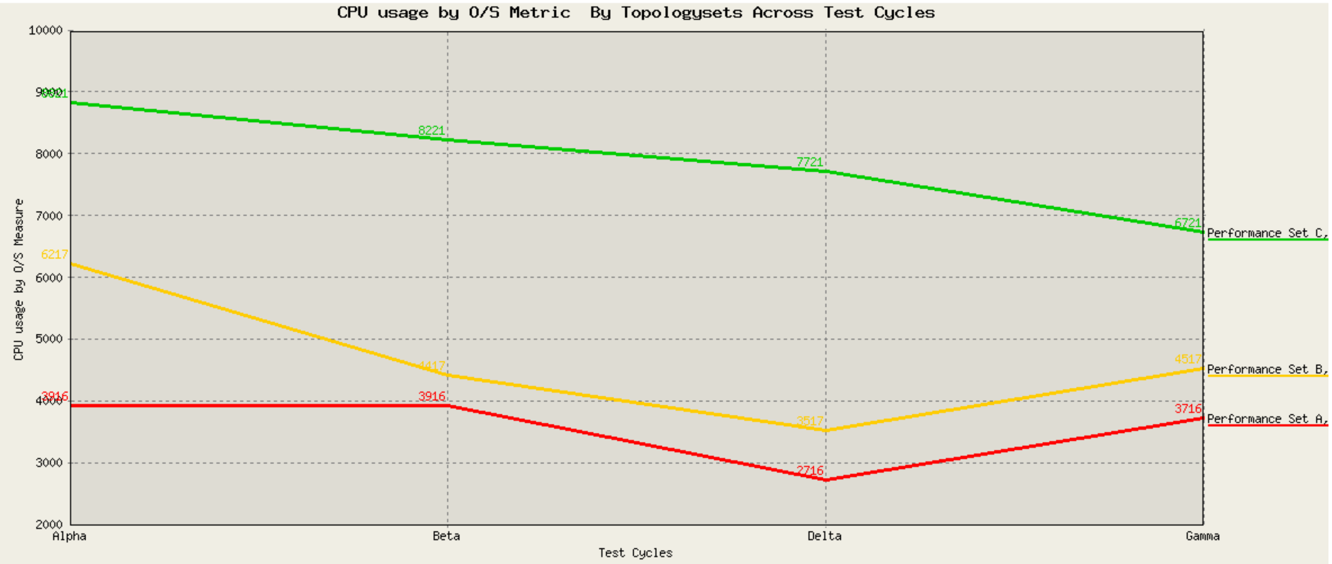


Chart to compare a metric for a specific workload type across multiple releases on an aggregate of topologies

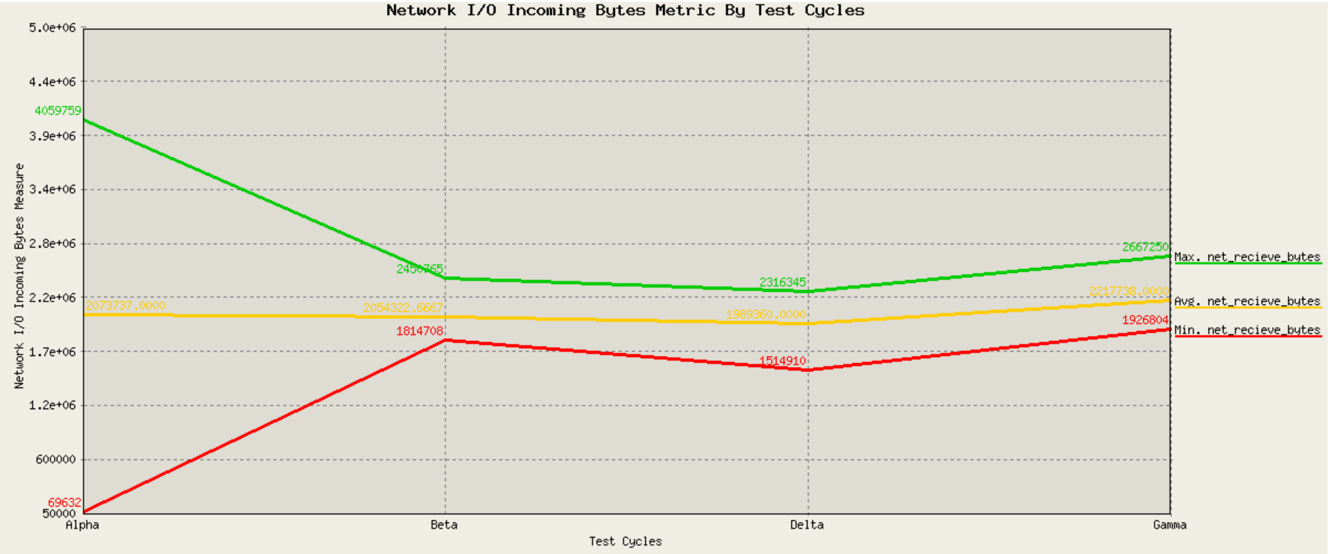
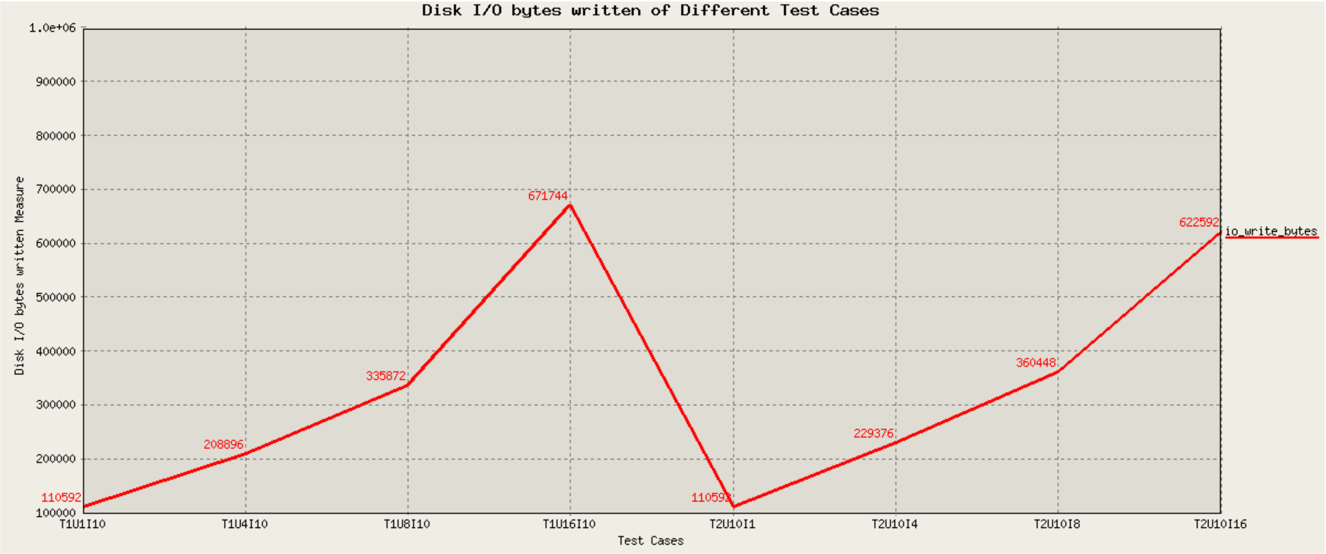


Chart to compare a metric across different workload types on a single testcycle on a specific topology


Disk I/O bytes written of Different Test Cases
Disk I/O bytes written Measure
1.0e+06
io_write_bytes
T1U1I10
T1U4I10
T1U8I10
T1U16I10
T2U10I1
T2U10I4
T2U10I8
T2U10I16
Test Cases