Performance Optimization of Java Enterprise Systems

Omar Portillo. Supervised by: Prof. John Murphy

1 Motivation

Enterprise-level applications tend to be large and complex with heterogeneous environments and highly distributed architectures. They also have very tight performance requirements to achieve fast response time and high throughput.

There are different challenges in these type of applications:

Testing and performance-related activities are very challenging

Performance issues might cause outages on production or even cancellation of projects

Need to have smarter techniques to optimize the performance of Java enterprise systems

Performance diagnosis and analysis tools are limited and require expert knowledge.

Java is a predominant technology at enterprise level

Complexity of Performance Optimization in Java Enterprise Applications

2 Proposed Approach

Exploit memory information for performing MaGC forecast

Major Garbage Collection (MaGC) is a common cause of performance degradation in Java applications, usually affecting their throughput and response time. We aim to use memory information to predict when MaGC events will occur so that systems can benefit from this knowledge and take decisions that mitigate these performance costs.

Automation of expert tools in perf. testing

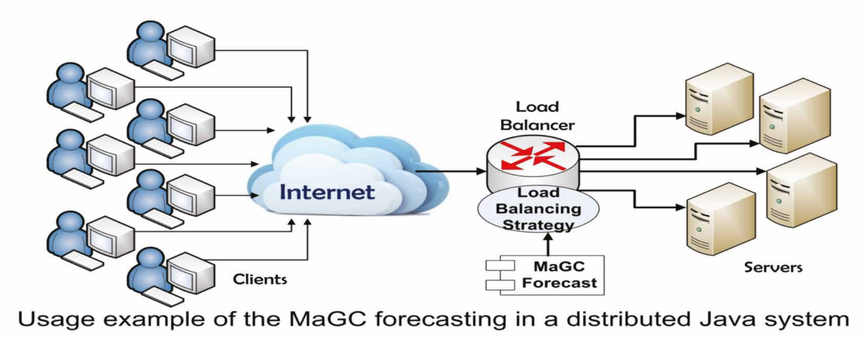
The automation of expert systems aims to address a common adoption barrier shared by many diagnosis tools, which are very time-consuming to use, and depend on the accuracy of their configuration to produce effective results.

Performance Optimization of Java Enterprise Applications

3 Methodology

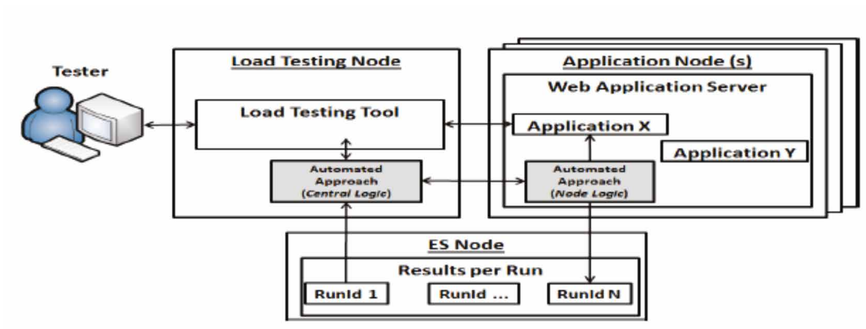
The following components are planned to be developed

GC-aware load balancing strategy: To avoid impacting the cluster's performance due to MaGC events, this strategy selects the nodes which are not expected to suffer a MaGC event in the immediate future, as optimal nodes for the incoming workloads. Internally, it leverages on MaGC forecasts to decide on the best way to balance the workload, and program families to adapt to the GC characteristics of the application.



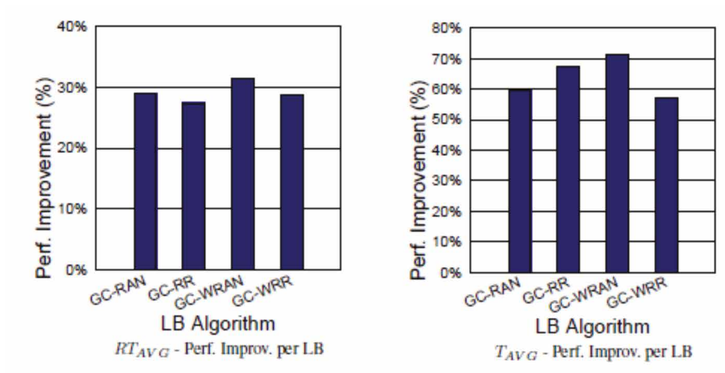
Usage example of the MaGC forecasting in a distributed Java system

Automation framework for expert tools: It shields testers from the complexities of properly configuring and using expert systems, so that they only need to interact with the load testing tool. Internally, it leverages on policies to control the different involved processes (e.g., data gathering, data processing, etc.).

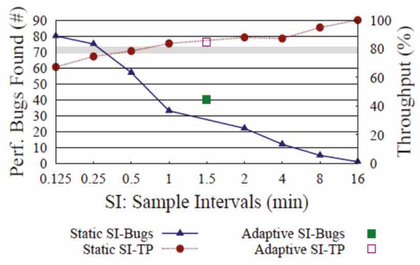


4 Results to date and Future work

» We have developed an algorithm to accurately forecast the MaGC events, a GC-aware load balancing strategy which exploits the forecasts, and a prototype to test them.



» We have developed a prototype of the automation framework. We also applied it to one expert tool to automate its usage and test the behavior of the framework and its policies.



Manual vs. Automated tool - Time/Effort Comparisons

Metric	M-WAIT (hr)	A-WAIT (hr)	M-WAIT vs. A-WAIT (%)
a. Duration of Perf. testing activity	32.8	24.1	-27%
b. Duration of Perf. Testing	24.0	24.0	0%
c. Effort of Perf. Analysis (d+e)	8.8	4.2	-52%
d. Effort of Bug Identification	6.8	2.2	-68%
e. Effort of Root Cause Analysis	2.0	2.0	0%

Results to date and Future work

» To apply the automation framework to other expert tools and extend its set of policies.

» To apply the load balancing strategy to other types of performance issues and extend its set of program families.

Ireland's European Structural and Investment Funds Programmes 2014-2020
Co-funded by the Irish Government and the European Union

European Union
European Regional Development Fund

SOUTHERN Regional Assembly
Promoting our Region