

# QoS Arbitrator

Resource Allocation Framework for Adaptive Tunable Computations



The illustration shows two executions, of a tunable junction-detection program, which is a core component of several image-processing applications. The executions correspond to different resource allocations for the three steps of the computation. The first step samples a subset of the pixels in the image and performs a quick test to determine whether each pixel tested is of interest. The second step draws a region of interest around a cluster of interesting pixels, and the third step runs a longer junction-detection algorithm for every pixel in the interesting regions. In the left execution, during the first step,  $1/16$ th of the pixels were tested. In the right execution, during the first step,  $1/64$ th of the pixels were tested. The computation can compensate (with respect to result quality) for a coarser sampling in the first step by possibly drawing additional and/or larger regions of interest and consequently requiring a larger resource allocation in the third step. The QoS Arbitrator framework takes advantage of the computation's tunability for efficient resource allocation.

## Goals

QoS Arbitrator is a resource-allocation framework for executing adaptive tunable parallel computations. It is, at present, in the preliminary design and implementation stage. When completed, it will enable effective utilization of distributed platforms for applications with QoS specifications.

A computation is *adaptive* if it exhibits at least one of these two properties: (1) it can statically (at start time) and/or dynamically (during the execution) *ask* for resources satisfying certain characteristics and incorporate such resources when they are given to it; (2) it can continue executing if some resources are taken away from it.

A computation is *tunable* if it can provide, in a controlled manner, different levels of QoS as a function of the time profiles of the resources provided to it. It is able, in general, to trade off resource requirements over time, while

maintaining a desired level of QoS. A larger allocation of resources in one stage of the computation's lifetime may compensate for a smaller allocation in another stage, in a *parameterizable manner*.

The QoS Arbitrator framework takes advantage of both the adaptability and the tunability of computations to efficiently utilize resources in a distributed system. Computation *adaptability*—made possible by programming systems such as Calypso, Chime, and Charlotte—allows the resource scheduler to preemptively allocate, deallocate, and reallocate resources to running computations. Computation *tunability* guides the resource scheduler in making these scheduling decisions over the program's lifetime.

This novel integration of the resource scheduling with flexibility in both the computation's specification and time profile of its resource allocation, is the primary distinguishing feature of the QoS Arbitrator framework.

## Functionality

The QoS Arbitrator framework consists of three components: tunable programs, an application-level QoS agent, and a system-level QoS arbitrator.

A **tunable program** consists of several stages, each with a parameterizable QoS specification, perhaps statistical in nature. It maps the quality of the input, the resources available, and the execution deadline into the quality of the output. The execution of tunable programs produces tunable computations.

The abstraction of the computation for resource allocation purposes follows the approach taken by the EPIQ project. Computations are viewed as multiple, composite tasks, each a chain of simpler tasks. Simple tasks are described in terms of their resource requirements, an associated deadline, and the quality of the produced result. Multiple task chains express the tunable nature of the ap-

plication, by specifying alternative executions. The representation captures both discrete and continuous tunability.

The **application-level QoS agent** negotiates an appropriate level of resource allocation by transmitting the computation structure and QoS specifications to the system-level QoS broker and receiving resource allocation time-profiles in return.

The **system-level QoS arbitrator** allocates resources in the distributed system to competing computations based on their QoS requirements. Although the role played by the QoS arbitrator is the same as that of an online scheduler of real-time tasks, the QoS broker can take advantage of the additional flexibility inherent in tunable and adaptive computations. Together with the ResourceBroker, QoS Arbitrator will enable effective utilization of distributed platforms for the execution of applications with QoS requirements.

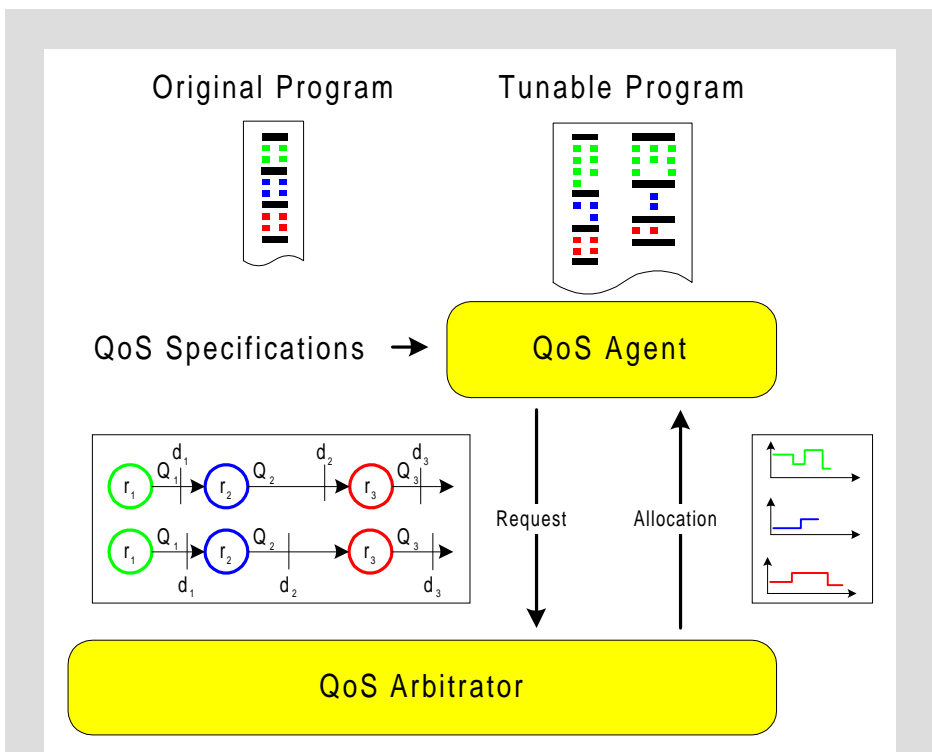
## Research with participation of

**Prof. Vijay Karamcheti**  
New York University  
+1 212 998 3496 (phone)  
+1 212 995 4123 (fax)  
vijayk@cs.nyu.edu

**MILAN is a joint project of**  
New York University  
Arizona State University

**Zvi M. Kedem**  
New York University  
+1 212 998 3101 (phone)  
+1 212 477 3265 (fax)  
kedem@cs.nyu.edu  
<http://www.cs.nyu.edu/milan>

**Partha Dasgupta**  
Arizona State University  
+1 602 965 5583 (phone)  
+1 602 965 2751 (fax)  
partha@asu.edu  
<http://milan.eas.asu.edu>



The illustration shows the three components of the QoS Arbitrator framework: tunable programs, an application-level QoS agent, and a system-level QoS arbitrator. Tunable programs are produced from the original program using knowledge of the application domain. The QoS agent transmits a flexible specification of the program to the QoS arbitrator, which leverages *both* program adaptability and tunability to efficiently allocate, deallocate, and re-allocate resources in a distributed system.