Position summary: Hinting for goodness' sake

David Petrou, Dushyanth Narayanan, Gregory R. Ganger, Garth A. Gibson, and Elizabeth Shriver[†]
Carnegie Mellon University and [†]Bell Labs, Lucent Technologies

Modern operating systems and adaptive applications offer an overwhelming number of parameters affecting application latency, throughput, image resolution, audio quality, and so on. We are designing a system to automatically tune resource allocation and application parameters at runtime, with the aim of maximizing user happiness or *goodness*.

Consider a 3-D graphics application that operates at variable resolution, trading output fidelity for processor time. Simultaneously, a data mining application adapts to network and processor load by migrating computation between the client and storage node. We must allocate resources between these applications and select their adaptive parameters to meet the user's overall goals. Since the user lacks the time and expertise to translate his preferences into parameter values, we would like the system to do this.

Existing systems lack the right abstractions for applications to expose information for automated parameter tuning. *Goodness hints* are the solution to this problem. Applications use these hints to tell the operating system how resource allocations will affect their goodness (utility). E.g., a video player might have no goodness below some allocation threshold and maximum goodness above another. Goodness hints are used by the operating system to make resource allocation decisions and by applications to tune their adaptive parameters. Our contribution is a decomposition of goodness hints into manageable and independent pieces and a methodology to automatically generate them.

One half of a goodness hint is a *quality-goodness mapping* which tells us how application qualities translate into user happiness. Qualities are measures of performance (latency, throughput) or of fidelity (resolution, accuracy). We hope to leverage user studies from the human-computer interaction community to generate these mappings. The system will also use user feedback to dynamically tailor the mappings to specific users.

A *resource-quality mapping* forms the other half of a goodness hint; our current research focuses on this half. This mapping describes the relationship between an application's resource allocation and its qualities. To do this, we first map adaptive parameters to resource usage by monitoring the application, logging its resource usage for various parameter values, and using machine learning to find the relationship between parameter values and resource usage. We create this mapping offline with controlled experiments to explore the parameter space, and update it online based on dynamic behavior.

Given the resource usage and allocation of an application, we predict its performance using simple models. E.g., a processor-bound computation requiring 1×10^6 cycles and allocated 2×10^6 cycles will have a latency of 0.5 sec. More complex applications will use multiple resources, perhaps concurrently. We will use machine learning techniques to specialize our models to particular applications.

Finally, given some resource allocation, an application must pick adaptive parameter values that maximize its goodness. An *optimizer* searches the parameter space to find the optimal values. By embedding the optimizer in the goodness hint, the operating system is also made aware of what the application will choose. The operating system itself uses a similar optimizer to find the resource allocation that will maximize goodness across applications.

We are building a prototype to validate these concepts. Currently, the prototype supports two resources: processor and network. To map adaptive parameters to resource usage we use linear least squares regression. To search through the space of application parameters and resource allocations, we use a stochastic version of Powell's conjugate direction-set method. We have two very different applications: a 3-D graphics radiosity application [Narayanan, et al., WMCSA 2000], and an Abacus data mining application [Amiri, et al., USENIX 2000].

Our initial results are encouraging. Our system generates accurate resource-quality mappings for both applications. (The quality-goodness half was constructed by hand.) In simulation, our resource allocator is always able to maximize overall goodness, which is a weighted sum of application goodnesses. However, the overhead of the search algorithm is prohibitive, and we are investigating alternatives.

This work raises several research questions: How can we talk about resource usage and allocation in a platform independent way? What is the best way to combine individual application goodnesses into user happiness? What kind of online feedback can we expect from a typical user, and how can we use it to dynamically refine goodness hints?