Modeling Uncertainty of Predictive Inputs in Anticipatory Dynamic Configuration

Vahe Poladian
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
+1-412-268-5941
Vahe.Poladian@cs.cmu.edu

Mary Shaw
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
+1-412-268-2589
Mary.Shaw@cs.cmu.edu

David Garlan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
+412-268-5056
David.Garlan@cs.cmu.edu

ABSTRACT
Dynamic adaptive systems based on multiple concurrent applications typically employ optimization models to decide how to allocate scarce resources among the applications and how to tune their runtime settings for optimal quality-of-service according to the preferences of an end user.

Traditionally, such systems have avoided dealing with *uncertainty* by assuming that current snapshots of the relevant inputs are precise and by solving for an optimal system point. To achieve dynamic behavior, a system performs an optimization loop upon discovering changes in the input variables (e.g. changes in the available level of resources) and adapts the applications according to the new optimal solution. Unfortunately, when certain adaptation actions incur costs, such *reactive* adaptation strategies suffer from a significant shortcoming: several locally optimal decisions over time may often be less than optimal globally.

By using predictive information about the future values of the problem inputs, we can model and implement an *anticipatory* adaptation strategy that helps improve the global behavior of the system in many situations. However, modeling predictions requires representing and dealing with uncertainty from different sources. In this paper, we describe our proposed approach to represent multiple sources of uncertainty and outline algorithms for solving the anticipatory configuration problem with predictive inputs.

Keywords
Adaptive, dynamic, optimization, prediction, uncertainty.

1. INTRODUCTION
Several dynamically adaptive systems (see: [8] [9][10] [12]) have used optimization models to improve the quality of service delivered to end users when resource are scarce, i.e. when the maximum resource demand by all the applications in a user's task exceeds the available level of resources. The optimization models typically require several inputs: (1) requirements for the task and preferences for various dimensions of task quality from the user, (2) application resource demand for each adaptive setting, and (3) the available level of the resources. The adaptive system uses an algorithm to determine the best suite of applications for a task, allocates resources among these applications, and dictates to each application the optimal level of quality so that the utility of the user is maximized according to his preferences. We call this *dynamic configuration*. Once the configuration is determined, the states of the applications are changed using adaptive mechanisms.

Typically, existing optimization models use only the current snapshots of the inputs to make a decision. When the value of an input changes, the optimization algorithm is re-run with the updated values of the inputs, resulting in a new optimal point. The system is then adjusted to the new optimal point using adaptive mechanisms. For example, when the available level of a resource drops, the previous optimal resource allocation is no longer feasible, and the system needs to re-configure. Some adaptive actions are costly because they are disruptive to the user or require additional resources. These costs are modeled either as penalties in the preference functions or as a resource expenditures. We call such a configuration strategy *reactive*, and note that several locally optimal reactive decisions might often be less than optimal over time.

We have proposed an alternative, anticipatory, strategy of configuration ([13]) that explicitly models predictions of input values into future. This strategy allows making configuration decisions that are optimal over time. To model a configuration strategy with predictions, we need a way to explicitly represent several sources of uncertainty, potentially complicating the decision making of the system.

In this paper, we address the challenges of representing and dealing with the uncertainty in the predictions of the inputs in an optimization model of a dynamically adaptive system in a pervasive computing domain. We also propose how existing algorithms can be modified to solve the dynamic optimization problem under uncertainty of the predictors.

The paper is organized as follows. Section 2 summarizes related work. Section 3 introduces the problem in an existing dynamic adaptive system. In Section 4, we introduce the various sources of uncertainty and describe our proposed representations. Section 5 describes how the optimization problem with predictive inputs can be solved. Section 6 concludes the paper.

ASE'07, November, 2007, Atlanta, GA, USA.

2. RELATED WORK

In his 2001 paper on the challenges of pervasive computing [16], Satyanarayanan identified prediction as an important desired feature for systems that aim to provide an improved and seamless user experience.

There is extensive research in resource demand and availability prediction. NWS [18] and RPS [1] are tools for gathering and analyzing resource demand and supply. Dinda [3] presents a comprehensive overview of prediction using linear time series models. A number of studies (Qiao [14], Sang [15], Wolski [19]) have demonstrated that: (1) resources have good predictability and (2) when resources are predictable, inexpensive models with autoregressive (AR) components work just as well as more complex schemes. Qiao et al [14] conjecture that resource prediction can be done online, using software running on routers or compute servers. Gurun et al designed a lightweight version of NWS called NWSLite suitable for mobile devices [6].

Task-oriented computing was proposed by Sousa [17] who defined an architecture for pervasive computing and a light-weight language for describing everyday user tasks. The task request primitives in 4.1.1 are from that work. The probability models for describing the activation times and durations of tasks are our own proposal. To our knowledge, there are no statistical studies confirming or denying those models.

Analytically, anticipatory configuration is similar to online stochastic combinatorial optimization (OSCU) problems such as packet routing and vehicle dispatch ([2][7]). While the problem domains are different, the configuration strategies have equivalent concepts and algorithms in that domain. Reactive strategy is equivalent to the Local algorithm in OSCU. When resource predictors are perfect, anticipatory strategy is equivalent to the Offline algorithm. And lastly, anticipatory strategy with noisy predictors is equivalent to various Expectation algorithms.

3. MODEL FORMULATION

In this section we describe the dynamic configuration problem in a pervasive computing system. We describe the reactive configuration model and strategy, and use this setup to introduce the sources of uncertainty in the problem later in section 4. Inputs of the model are described in section 3.1, decision variables are described in section 3.2. The complete mathematical notation of the reactive model is found in [12] and in the appendix of [13].

3.1 Model Inputs

3.1.1 User task and preference

The basis of configuration is one or more user tasks. A task is a set of services. Each service is a description of capabilities of similar applications comprised of a *service type* (e.g., "play video") and zero or more quality of service dimensions (QoS). For example, the QoS dimensions for a "play video" service are "frame rate" and "frame size".

Preferences are mathematical functions with weights that are specific to each task. There are three types of preferences: QoS preferences, application affinity preferences, and switch penalties. There is a QoS preference function per quality dimension and an application affinity preference for each service as a whole. A preference function for a QoS dimension captures how the user values improvements in that dimension. Associated with each service is a switching penalty that reflects the inconvenience to the user should the system decide to replace a running application with an alternative as part of an optimization decision. Penalties are designed to discourage or, in some cases, to disallow switching of running applications.

3.1.2 Application Profiles

Each installed application instance has a profile. A profile specifies the service types that an application can provide and an enumeration of quality and resource vector tuples: $<q, r>$. The application promises to provide a level of QoS q, if corresponding resource level r is allocated to the application. This is the estimated resource cost for each level of quality the application supports.

3.1.3 Resource Availability

Resource availability is a vector showing the current snapshot of available resources. Resource availability changes over time, but in the reactive model only the current level of the resource availability is considered.

3.2 Decision Variables

The decision variable in the reactive model is a *configuration* with three components: (1) a suite of applications, one per active service, (2) quality set-points, one per application selected, and (3) resource allocation among the applications.

3.3 Optimization Problem

The objective of the system is to optimize a user's instantaneous utility (IU) for the task. An IU of a configuration is computed as a weighted sum of the preference functions over that configuration. Candidate configurations can be compared using their IU by applying the preference functions.

4. MODELING UNCERTAINTY FOR ANTICIPATORY CONFIGURATION

In the anticipatory model of configuration, we need to reason about the cumulative effect of several configuration decisions over time. To do so, we use a discrete time model. Let t denote the current time and T denote the duration of a task.

The objective of dynamic configuration in the anticipatory model is to maximize the expected value of the *accrued utility* (AU) over time. Accrued utility of a task from time t to time $t+T$ is the integral of instantaneous utility over that period of time. In a discrete time model, that integral is the sum of instantaneous utility values over the period from t to $t+T$.

The anticipatory model of configuration requires predictions of the input variables over a foreseeable period of time. These predictions come from different sources and contain uncertainties of different types that are unique to the source. In the next three sections we describe the representations of the predictions and uncertainty from each source.

4.1 Predictive Input Sources and Uncertainty

4.1.1 User task

A user makes requests to the system using a small vocabulary:

{*add, replace, disband,*}

The "*add*" operation requests a service to be activated. The "*replace*" operation shows user's displeasure with a running application, and explicitly asks for that application to be replaced by another. The "*disband*" operation tells the infrastructure to save the state of the application and close it. The "*add*" and "*replace*" operations can be accompanied with a set of preferences.

Predicting a user's task requests amounts to predicting when in the future the user will make a service request operation with respect to any of the services currently under consideration as well as the request type. The set of services of interest is the union of the services in all currently defined tasks.

A trace of *past* user requests can be captured as follows:

s_1: (op_{11},t_{11}), (op_{12}, t_{12}),…,

s_2: (op_{21}, t_{21}), (op_{22}, t_{22}),…,

…

where op_{ij} denotes a service request operation and t_{ij} denotes the time. We can leverage this concise notation for predictions.

The primary objective of task operation sequence prediction is to predict future resource demand. The "*add*" operation will result in resource demand increase. The "*disband*" operation will result in resource demand decrease.

The "*replace*" operation is allows the user to over-ride a system decision and replace a running application that the user is unhappy with. In general, the effect of "*replace*" operations on resource demand is less clear.

Given the clear effect of "*add*" and "*disband*" operations on resource demand, we start with a simple prediction goal:

- predicting when a service (or a group of services) is added,
- predicting how long a service (or a group of services) is needed, i.e., when these services will be disbanded.

Using three representative examples we build our case for describing the predictions for "*add*" and "*disband*" requests.

In the first example, a professor has a task of giving a lecture. The lecture is a regularly scheduled activity occurring on specific days of the week at scheduled times. Although the lecture has scheduled start and end times, the actual start and end times might vary slightly from the schedule slightly.

In the second example, a movie critic is working on a review. The completion of the task is driven by quality and amount of work. We can elicit a completion time from the user and further enhance that estimate using past observations of estimated and actual duration pairs. Upon completion of the session, we might be able to elicit from the user the time of the next activation of the task.

In the third example, the user receives an e-mail from a friend and clicks on a link that opens to a video from a popular site such as youtube.com. If the video is not interesting, the user abandons watching it after a few seconds. On the other hand, if the video is interesting, there is a high chance that the user watches it to completion and might choose to watch similar videos.

These examples suggest that we need to allow different shapes to express the probability distribution of the activation time and the duration of a task depending on the type of the task:

- For tasks following a strict schedule, both the activation time of the task and the end of a task can be derived from a source such as an electronic calendar. We propose to use normal random variables with small variances to capture the times of both events,
- For tasks that are not subject to a strict deadline, the activation times and duration can be estimated by the user. Both estimates will have errors and we propose to represent these using normal random variables with variances determined by historical data,
- For ad-hoc tasks, we propose to represent their activation time using a random variable following an exponential distribution,
- Durations of some tasks that have multiple peaks can be captured using mixture of normal distributions,
- For deadline-driven tasks, we propose to capture the activation time of the task as a normal random variable around a known mean and the end-time as one-half of a normal random variable,

We propose a *hierarchical* representation for expressing predictions of future user tasks requests. This representation assigns a type to each task from a small vocabulary. Next, for each type of task, we use a probability distribution to describe the activation and ending times of the task. We assume that the type assigned to the tasks is precise, so uncertainty is in the activation and ending times of the task.

Because we are unable to fully enumerate all possible types of tasks and the probability distributions of their activation and end times, we propose to use a discrete representation for capturing distributions. In this manner, the notation will not require an agreement between the provider of information and the consumer for describing each distribution that might occur.

For example, a normal probability density function can be approximated using a binomial distribution. Desired precision can be achieved using any number of integer probability masses.

4.1.2 Application Profiles

The source of uncertainty in application profiles is the resource requirement for every adaptive quality level. In the reactive model, the resource-quality pairs are non-random. Research in adaptive systems (e.g., [9][11]) provides evidence that the variance of average resource usage per quality level is very small. We propose to ignore the uncertainty due to such small variance and continue treating the inputs in the application profiles as non-random, as was the case in the reactive model.

4.1.3 Resource Availability

We leverage and extend existing research in resource prediction to represent uncertainty in resource availability. First, we discuss the prediction of resources that are not directly under our user's control, e.g. network bandwidth and CPU of compute servers. These resources have many consumers, typically in the hundreds or thousands, and the predictability of such resources is a direct result of utilization by a large pool of users.

Modeling Network Bandwidth, CPU Prediction

The prediction for resource availability should describe the probabilities of future resource paths. So the variable being

predicted is a random process over time. Predictions of values further into the future have larger errors, making the graphical representation of a predictor look like a cone.

We propose combining three different types of predictors: (1) recent history linear predictor, (2) seasonal predictor, and (3) bounding predictor, to arrive at an aggregate resource predictor.

A recent history linear predictor is motivated by existing prediction research ([3][19]) and exploits serial correlation in the adjacent values of resource availability time series. Autoregressive (AR), moving-average (MA), and autoregressive and moving average (ARMA) Gaussian models of low order often predict resource time series very well if there is any predictability.

Formally, an autoregressive linear recent history predictor of order p for resource R is an equation of the form:

$$R_{t+1|t} = \varphi_1 r_t + \varphi_2 r_{t-1} + \ldots + \varphi_p r_{t-p+1} + Z_{t+1},$$

where r_i are the previous p observations of the resource (the small letters indicate that these numbers are not random), φ_i are parameters of the model and are known at prediction time, and Z_{t+1} is a normal random variable with mean 0 and variance σ, $Z_{t+1} \sim N(0, \sigma)$.

The above prediction is only one step-ahead, but using recursive substitution we can describe a multiple step-ahead, path-dependant prediction.

While a recent history predictor exploits serial correlation in the recent history of a time series, a seasonal predictor captures periodic patterns. In the time series classical decomposition, seasonal components are identified and removed first, before a time series is analyzed using recent history model. At prediction time, seasonal components are added back to compute the final prediction.

The prediction of a seasonal component is a time-series of non-random values. Because the predicted values of a seasonal component are not random, they don't change with passage of time.

And lastly, a bounding predictor specifies absolute minimum and maximum values for the series. This predictor is motivated by various sources of information that can guarantee or almost guarantee that the available level of the resource is within some minimum and maximum thresholds.

When one instance of a recent history and multiple instances of seasonal and bounding predictors are available, we propose the following algorithm to combine the predictors:

- combine all the seasonal predictors together using basic addition, resulting in a single seasonal predictor,
- combine all the bounding predictors together, using the maximum of all minimums and the minimum of all maximums, resulting in a single bounding predictor,
- add the aggregate seasonal predictor to the recent history predictor, resulting in a shift of the mean,
- use the aggregate bounding predictor to limit possible values of the resource.

In [13] we demonstrated how a data structure based on a trinomial tree can be used to approximate an aggregate resource prediction. The tree has a branching factor of 3 for nodes up to a specified depth; nodes beyond that depth have a branching factor of 1. Thus, we capture the mean and the variance of the prediction for the near future using three probability masses. For predictions farther in the future, we only capture the mean prediction. This structure was effective to capture predictions for anticipatory configuration. Similar trees with a branching factor equal to any odd number can be constructed.

Modeling Battery Prediction

Of the resources that are entirely under the user's control, battery energy is uniquely different. First, unlike bandwidth or CPU, this resource can be stored, depletes with usage, and can be charged. Second, currently available level of battery can be estimated very precisely. Third, the uncertainty of its availability in the future is dependent on user's own usage of the resource and the probability of being charged from a wall outlet. And fourth, there is a correlation between battery energy spent and available CPU as many modern hardware platforms offer power saving settings that can dynamically alter CPU voltage and clock rate.

We propose to model the aggregate drain rate of battery per unit as a decision variable. In order to limit the search space of possible drain rates over time, we propose to consider only those drain levels that have an impact on another resource's availability and to keep the drain rate unchanged if user's task has not changed.

And lastly, because remaining battery energy can be valuable for future tasks, we propose to account for that using an additional preference function elicited from the user. Using only two threshold values and one utility value, we can allow the user to express the following preference: remaining battery below the lower threshold level L provides a significant negative utility, between the lower threshold L and the upper threshold U utility increases monotonically from zero to some value U, and above the upper threshold value L utility remains at U.

4.2 Decision Variables

In the anticipatory model of configuration, the decision variable is a contingent time series of configurations. After making each configuration decision, the system commits to the first decision in the series, leaving the possibility of changing subsequent decisions. The system then observes changes in the input variables, and repeats the above optimization.

4.3 Optimization Problem

In the model of anticipatory configuration, we optimize the value of expected accrued utility (AU) subject to the quality-resource profiles of available applications and predictions of the resources.

5. CONFIGURATION ALGORITHM IDEAS

We propose to stage the problem of anticipatory configuration into increasingly difficult instances. We make assumptions about the predictions, and then gradually relax those.

These are the assumptions that allow the staging of the problem:

A1. The time frame of optimization is fixed from t to $t+T$, and the predictions of future user task requests are known exactly,

A2. Battery energy is excluded from consideration,

A3. The predictions of all the resources except battery are known exactly, i.e., they contain no uncertainty.

We propose to stage the problem as follows. Initially, we require all three assumptions to hold. In the second stage, we relax only A3, allowing for uncertainty in resource predictions but excluding battery from consideration. In the third stage, in addition to having dropped A3, we also drop A2, taking battery availability and drain into account. In the fourth stage, we drop A1 as well, allowing task activation times and durations to be random as described in section 4.1.1.

In [13], we demonstrated solutions to the first and second stage problems. The solution to the first stage is a dynamic programming algorithm that inducts backwards in time. The solution to the second stage problem is a modified version of that dynamic program that performs expectation maximization over the predicted resource paths.

We propose to solve the stage three version of the problem by modeling the drain rate of battery per unit time as a decision variable and including a term in accrued utility to account for the remaining battery energy. We have preliminary evidence that the algorithm from [13] can be modified to solve this problem.

At this time, we don't have a fully developed solution to the stage four version of the problem. Our proposal is to further modify the existing dynamic programming algorithm. However, with the activation times and durations of tasks allowed to be random, the definition of the accrued utility might need to be modified to allow for comparisons between utility from quality of service and utility from longer task duration.

6. CONCLUSIONS

In this paper we have described an improved optimization model for a dynamically adaptive system in the pervasive computing domain. This new anticipatory model uses predictions of future values of inputs and improves over an earlier reactive model in many situations. We have described representations for various sources of uncertainty in the predictions of the inputs. Further, we have presented ideas for staging and solving the resulting stochastic optimization problem by modifying an existing dynamic programming algorithm.

Ongoing work focuses on designing a runtime architecture that allows resource and user task request predictors. We are working on runtime representations of predictive uncertainty and implementations of the dynamic optimization algorithms.

7. ACKNOWLEDGMENTS

This work was funded in part by NSF grants CCR-0205266, CCF-0438929, CNS-0613823, and by DARPA grant N66001-99-2-8918. Authors thank professors Anthony Brockwell, Peter Steenkiste, and Mahadev Satyanarayanan (Carnegie Mellon); and Peter Dinda (Northwestern University) for their suggestions.

8. REFERENCES

[1] P. Dinda. Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction In Distributed Systems. *IEEE Transactions on Parallel and Dist Systems (TPDS),* 17:2, February 2006.

[2] R. Bent and P. Van Hentenryck. Regrets Only! Online Stochastic Optimization under Time Constraints. *Proc 19th AAAI*, 2004.

[3] P. Dinda, D. O'Hallaron. Host Load Prediction Using Linear Models. *Cluster Computing*, 3:4, 2000.

[4] D. Garlan, et al. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing,* 21:2, April-June, 2002.

[5] D. Garlan, et al. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *IEEE Computer*, 37:10, 2004.

[6] S. Gurun, et al. NWSLite: A Light-Weight Prediction Utility for Mobile Devices. *Proc. 2nd IEEE Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys), 2004.*

[7] P. Hentenryck, et al. Online Stochastic Optimization Under Time Constraints. Working paper, last accessed in February 2007 at http://www.cs.brown.edu/people/pvh/aor5.pdf.

[8] C. Lee. On Quality of Service Management. *PhD Thesis, Carnegie Mellon University Technical Report CMU-CS-99-165,* 1999.

[9] D. Narayanan, M. Satyanarayanan. Predictive Resource Management for Wearable Computing. *Proc. 1st IEEE Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys), 2003.*

[10] R. Neugebauer and D. McAuley. Congestion Prices as Feedback Signals: An Approach to QoS Management. *Proc. ACM SIGOPS European Workshop*, 2000.

[11] B. Noble, et al. Agile Application-Aware Adaptation for Mobility. *Proc. ACM Symp on Operating Systems Principles (SOSP)*, 1997.

[12] V. Poladian, et al. Dynamic Configuration of Resource-Aware Services. *Proc IEEE Intl Conf on Software Engineering (ICSE)*, 2004.

[13] V. Poladian, et al. Leveraging Resource Predictions in Anticipatory Dynamic Configuration. *Proc IEEE Intl Conf on Self-Adaptive and Self-Organizing Syst. (SASO)*, 2007.

[14] Y. Qiao, J. Skicewicz, P. Dinda. An Empirical Study of the Multiscale Predictability of Network Traffic. *Proc Intl Symposium on High Performance Distributed Computing (HPDC)*, 2004.

[15] A. Sang and S. Li. Predictability analysis of network traffic. *Proc. of INFOCOM*, 2000.

[16] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, August 2001.

[17] J.P. Sousa, D. Garlan. The Aura Software Architecture: an Infrastructure for Ubiquitous Computing. *Carnegie Mellon Technical Report, CMU-CS-03-183*, 2003.

[18] R. Wolski, et al. The network weather service: A distributed resource performance forecasting system. *J. of Future Generation Computing Systems*, 1999.

[19] R. Wolski, et al. Predicting the CPU availability of time-shared Unix systems. *Proc Intl Symp High Perf Dist Computing (HPDC)*, 1999.