An Autonomic Resource Provisioning Framework for Mobile Computing Grids

Hariharasudhan Viswanathan, Eun Kyung Lee, Ivan Rodero, and Dario Pompili
NSF Cloud and Autonomic Computing Center
Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ
{hari_viswanathan, eunkyung_lee, irodero, pompili}@cac.rutgers.edu

ABSTRACT

Enabling data- and compute-intensive applications that require real-time in-the-field data collection and processing using mobile platforms is still a significant challenge due to i) the insufficient computing capabilities and unavailability of complete data on individual mobile devices and ii) the prohibitive communication cost and response time involved in offloading data to remote computing resources such as clouds for centralized computation. A novel resource provisioning framework is proposed for organizing the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in the vicinity in order to form an elastic resource pool (a heterogeneous mobile computing grid) that can be harnessed to collectively process massive amounts of locally generated data in parallel. The proposed framework is imparted with autonomic capabilities, namely, self-optimization and self-organization, in order to be energy and uncertainty aware, respectively, in the dynamic mobile environment.

Categories and Subject Descriptors

C.2.4 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed Systems

Keywords

Mobile grids, autonomic management, uncertainty

1. INTRODUCTION

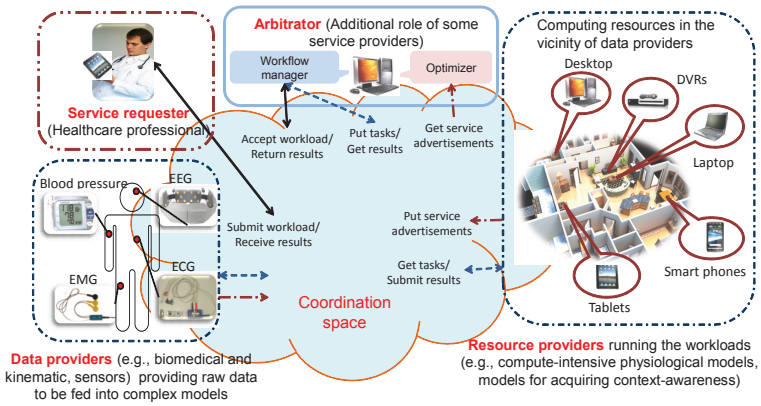
The computation and communication capabilities of mobile handheld devices such as smart phones, tablets, netbooks, and laptops have improved tremendously due to the advances in microprocessor, storage, and wireless technologies. As more and more of these mobile devices are coupled with in-built as well as external sensors capable of monitoring ambient conditions, acceleration, orientation, gravity, biomedical data (e.g., electrocardiogram, galvanic skin response, oxygen saturation) etc., and Global Positioning System (GPS) receivers, they can provide spatially distributed measurements regarding the environment in their proximity.

ICAC'12, September 18–20, 2012, San Jose, California, USA.

In this paper, we present a resource provisioning framework that organizes the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in the vicinity in order to form an *elastic resource pool* – a heterogeneous mobile computing grid. This local computing grid can be harnessed to enable innovative *data-* and *compute-intensive mobile applications* such as content-based distributed multimedia search and sharing [1], distributed object recognition and tracking, and ubiquitous context-aware health monitoring [7]. The response time, quality, and relevance of such mobile applications, which rely on *real-time in-the-field processing of locally generated data,* can be drastically improved using our envisioned framework. Presently, the primary impediments to real-time in-the-field data processing are, 1) insufficient sensing and computing capabilities on individual mobile devices, which prevents them from producing meaningful results within realistic time bounds *in isolation,* and 2) the prohibitive communication cost and response time involved in enabling such applications using the *wired-grid-computing* and/or *cloud-computing* approaches alone [15] – in which computation and storage are offloaded to remote computing resources on the Internet.

In order to address the research challenges associated with reliable mobile grid coordination and application performance under uncertainty (in terms of device availability due to node mobility and susceptibility to failures), we impart our proposed resource provisioning framework with autonomic capabilities, namely, *self-optimization* and *self-organization*. Applications are made up of one or more workloads, which are usually composed of multiple tasks whose order of execution is specified by a workflow. Workload here refers to compute-intensive mathematical models with different computational, storage, and deadline requirements.

In our solution, the entities of the hybrid grid may at any time play one or more of the following three *logical roles* as shown in Fig. 1(a): i) *service requester,* which places requests for workloads that require additional data and/or computing resources from other devices, ii) *service provider,* which can be a *data provider, resource provider,* or both, and iii) *arbitrator* (also typically known as broker), which processes the requests from the requesters, determines the set of service providers that will provide or process data, and distributes the workload tasks among them. Data providers provide scalar or multimedia data while resource providers lend their computational, storage, and communication resources for processing data. The arbitrator – an additional role played by some of the service providers – is aided by a novel *energy-aware resource allocation engine,* which will distribute the workload tasks optimally among the service providers. This way, we ensure that the data providers do not drain valuable energy. Figure 1(a) depicts the envisioned framework enabling an ubiquitous healthcare application that relies on processing collected biomedical data in-the-field for



(a)

Focus	Chu et al. [2]	Giurgiu et al. [8]	Chun et al. [3]	Darby et al. [5]	Costa et al. [4]	Huang et al. [9]	Lima et al. [6]	Jiang et al. [10]	Liao et al. [14]	OURS
Infrastructure										
Mobile					✓		✓	✓	✓	
Static						✓				
Hybrid	✓	✓	✓	✓						✓
Concern										
Energy			✓			✓				✓
Connectivity				✓	✓					✓
Uncertainty				✓						✓
Contribution										
Protocol				✓			✓		✓	
Middleware	✓	✓	✓		✓	✓		✓		✓

(b)

Figure 1: (a) Autonomic resource provisioning framework enabling ubiquitous healthcare; (b) Summary of related work.

real-time physiological monitoring. Workloads in this example scenario include mathematical models for acquiring context awareness that operate on biomedical and kinematic sensor data.

Prior research efforts, summarized in Fig. 1(b), have aimed at integrating mobile devices into the wired-grid and cloud-computing infrastructure mainly as service requesters. In contrast, we exploit mobile devices as service providers and address uncertainty-aware autonomic resource management for ensuring application Quality of Service (QoS) even in highly dynamic and unpredictable environments. Our resource allocation engine relies on long-term statistics regarding the dynamics of the underlying resource pool to protect application performance from the undesired effects of uncertainties. The major contributions of this paper include, 1) a role-based architectural framework for handling service discovery and service request arrivals as well as for task distribution and management, 2) a novel energy-aware resource allocation engine for imparting the self-optimization capability, i.e., for allocating the workload tasks optimally among the computing devices, 3) an innovative mechanisms to estimate the uncertainty (in terms of dynamics and size) in the resource pool for imparting uncertainty-aware self-organization, and 4) a detailed performance analysis of our proposed autonomic resource provisioning framework through experiments on a prototype testbed as well as simulations.

The rest of the paper is organized as follows. In Sect. 2, we present our autonomic resource provisioning framework for mobile grids. In Sect. 3, we describe our experimental methodology and results. Finally, in Sect. 4, we present our conclusions and plans for future work.

2. PROPOSED SOLUTION

Our *role-based architectural framework* facilitates coordination and seamless switching among the three logical roles. The *energy-aware resource allocation engine* and *mechanisms for uncertainty awareness* impart the self-optimization and self-organization capabilities, respectively.

2.1 Role-based Architectural Framework

Service discovery: Service discovery at the arbitrators is achieved through voluntary *service advertisements* from the service providers. Service advertisement from a service provider n includes information about the current position, amount of computing (γ_n^{cpu}, in terms of normalized CPU cycles), memory (γ_n^{mem} [Bytes]), and communication (γ_n^{net} [bps]) resources, the start (t_n^{in}) and end (t_n^{out}) times of the availability of those resources, and the available battery capacity (e_n^{adv} [Wh]). The arbitrator is aware of the power drawn by the workload tasks of a specific application when running on a specific class of CPU and memory (together given by c_n^{comp} [W]) as well as network (c_n^{net} [W]) resources at each service provider as the information about the different types of devices is known in advance. The arbitrators use the information from service advertisements of the N computing devices to derive the following: $\mathbf{S} = \{s_n\}_{1\times N}$, where $s_n \in \{1, 0\}$, which conveys whether n is a resource provider or not, and $\mathbf{D} = \{d_n\}_{1\times N}$, where $d_n \in \{1, 0\}$, which conveys whether n is a data provider or not.

We advocate the use of a distributed arbitrator self-election mechanism similar to the one in [12]. Our self-election mechanism works as follows: each service provider will determine the potential size of its resource pool based on the number of advertisements it has received. Then, all the service providers advertise this number and determine their *rank* in their neighborhood in terms of the potential size of their resource pool. The service providers use a pre-determined rank threshold (which varies depending on the network size and density) to elect themselves as arbitrators.

Workload management: Each arbitrator is composed of two components, namely, *workload manager* and *scheduler/optimizer,* as shown at the top of Fig. 1(a). The workload manager (also called *master*) tracks workload requests, allocates workload tasks among service providers, and aggregates results. The optimizer identifies the number of service providers (also called *workers*) available for the requested duration and determines the optimal distribution of workload tasks among them. The optimizer shares the workload submitted by the data providers among the available service providers based on one of several possible policies. One policy may aim at minimizing the battery drain while another policy may just place emphasis on response time without considering battery drain. Our framework applies to applications exhibiting *data parallelism* (in which data is distributed across different parallel computing nodes that perform the same task) as well as to applications exhibiting *task parallelism* (in which parallel computing nodes may perform different tasks on the same or different data).

2.2 Energy-aware Resource Allocation Engine

Here, we explain our energy-aware resource allocation engine (an optimization problem corresponding to one of the aforementioned policies) for hybrid grids in detail. In the following, we explain the sequence of events happening at one of the arbitrators while similar events happen simultaneously at the other ar-

bitrators in the computing grid. When a service requester needs additional data or computing resources, it submits a service request to the nearest arbitrator and also specifies δ^{max} [h], the maximum duration for which it is ready to wait for a service response. The arbitrator extracts the following information based on the service advertisements: the devices' (service providers') capability, $\mathbf{\Gamma}^x = \{\gamma_n^x\}_{1\times N}$, where $x = cpu, mem, net$; the associated costs, $\mathbf{C}^{comp} = \{c_n^{comp}\}_{1\times N}$ and $\mathbf{C}^{net} = \{c_n^{net}\}_{1\times N}$; the devices' availability, $\mathbf{T}^{in} = \{t_n^{in}\}_{1\times N}$ and $\mathbf{T}^{out} = \{t_n^{out}\}_{1\times N}$; and their battery status $\mathbf{E}^{adv} = \{e_n^{adv}\}_{1\times N}$.

The variables that the optimization problem has to find are, $\mathbf{A}$, $\mathbf{U}$, $\mathbf{\Delta}^d$, and $\mathbf{\Delta}^s$. Matrix $\mathbf{A} = \{a_{ij}\}_{N\times N}$ conveys the associativity of data provider i with service provider j, $\mathbf{U} = \{u_n\}_{1\times N}$ with $u_n \in \{1, 0\}$ conveys whether a resource provider n is used for computing or not, $\mathbf{\Delta}^d = \{\delta_n^d\}_{1\times N}$ [h] conveys the duration for which the services of each service provider will be used for data collection, and $\mathbf{\Delta}^s = \{\delta_n^s\}_{1\times N}$ [h] conveys the duration for which the resources of each service provider will be used for computation (cpu, mem, and net) and/or for multi-hop communication (net) as a relay node. In this formulation, the *objective* of the optimization problem, given by (1) and (2), is *maximization of minimal residual battery capacity* at all the service providers, $\max \min_n e_n^{res}$ [Wh], while ensuring that the service response is delivered within δ^{max}. This objective maximizes the lifetime of every single service provider and, thus, maintains the heterogeneity of the resource pool for longer periods. The set of service providers and the duration for which each of their capabilities are availed will be determined by considering the trade-offs among the cost e_n^{data} [Wh] (3) for transferring the data locally from data providers to the resource providers, the computational cost e_n^{comp} [Wh] (3) for availing the computational capabilities of the resource providers for servicing the request and for aggregating and generating the final response.

$$\textbf{Maximize}: \quad \min_n e_n^{res}, \tag{1}$$

$$where, \quad e_n^{res} = e_n^{adv} - (e_n^{data} + e_n^{comp}); \tag{2}$$

$$e_n^{data} = \delta_n^d \cdot c_n^{net}; \quad e_n^{comp} = u_n \cdot \delta_n^s \cdot c_n^{comp}. \tag{3}$$

In (2), $e_n^{data} + e_n^{comp}$ is the amount of battery capacity drained at each service provider n. δ_n^d for a service provider n depends on the amount of data it has to transmit (ω [Bytes] as a data provider) or aggregate ($\omega \cdot \sum_{i=1}^{N} a_{in}$ [Bytes] as a resource provider), and the availed communication capability, given by $\delta_n^d = f(\omega, \gamma_n^{net})$ when $u_n = 0$ and $\delta_n^d = f(\omega \cdot \sum_{i=1}^{N} a_{in})$ when $u_n = 1$. Here, without any loss of generality, ω is considered to be the problem size of a trivial task and each data provider provides the same amount of data. Function $f()$ monotonically increases as the amount of data to be transmitted or received increases. δ_n^s for a service provider n depends on the amount of data it has to process and the availed computing capabilities specified by γ_n^{cpu} and γ_n^{mem}, given by $\delta_n^s = g(\gamma_n^{cpu}, \gamma_n^{mem}, \omega \cdot \sum_{i=1}^{N} a_{in})$. Function $g()$ monotonically increases with the amount of data to be processed. The *constraints* to the optimization problem are, $\forall n = 1 \ldots N$,

$$s_n \geq u_n; \quad 0 \leq \delta_n^d, \delta_n^s; \tag{4}$$

$$\delta_n^s \leq \min\{t_n^{out}, t^{now} + \delta^{max}\} - \max\{t^{now} + \delta_n^d, t_n^{in}\}; \tag{5}$$

$$\delta_n^d \cdot c_n^{net} + u_n \cdot \delta_n^s \cdot c_n^{comp} \leq e_n^{adv}. \tag{6}$$

Constraint (4) ensures that only a resource provider is chosen to perform the computing. Constraints (5) and (6) ensure that every service provider's advertised availability (duration) and battery limit are not exceeded while satisfying the consumer's deadline for service response.

2.3 Uncertainty Awareness

Inaccurate estimation of the availability (duration) of service provider is a major source of uncertainty that results in a large number of incomplete workload task migrations. The duration of availability specified in the service advertisements is based on the battery drain estimates and may not accurately reflect the duration for which the service provider will be associated with the arbitrator. One or more of the service providers may lose network connectivity to the arbitrator or go offline, i.e., run out of energy due to an unexpected increase in battery drain because of other concurrent compute-intensive critical operations. As the arbitrator is also one of the service providers, this problem holds for arbitrators too.

We advocate the use of multiple arbitrators to avoid a single point of failure. In order to ensure that the unavailability of an arbitrator does not lead to the failure of the entire system, each arbitrator shares with all of its active data and service providers a list of alternate arbitrators – referred to as *proxies* – ranked according to their proximity (primary key) and physical addresses (secondary key). In case of an arbitrator failure, the service providers collaborate with the pre-specified proxy until all active workload tasks end. The arbitrators also share their current state information with their proxies to handle any unexpected failures.

In order to impart the uncertainty-aware self-organization capability to the proposed resource-allocation framework, we designed a mechanism that helps the arbitrator extract the following long-term statistics from the underlying resource pool: the average arrival (joining) rate of service providers ($\widetilde{W}$), the average service provider availability duration ($\widetilde{T}$), and the average number of service providers associated with the arbitrator at any point in time ($\widetilde{N}$). The relationship among these three long-term statistics is given by Little's theorem, $\widetilde{N} = \widetilde{W} \cdot \widetilde{T}$. The arbitrators update continuously these statistics and share at least two of the three aforementioned averages with its proxies. Knowledge of these average statistics helps the arbitrators assess the *churn rate* of service providers. Churn rate is a measure of the number of service providers moving into or out of an arbitrator's resource pool over a specific period of time. Note that the arbitrators need not extract or be aware of the underlying probability distribution of service provider arrivals or of availability durations.

Churn rate of service providers will be different in different geographic location. For example, the churn rate of service providers at a shopping mall is far greater than the one at a coffee shop. Also, at a particular location, the churn rate can vary over time (say, depending on the time of the day). When the churn rate of service providers is high, i.e., the average duration of service providers availability is low, the percentage of migrated workload tasks will be high if the resource-allocation engine does not possess uncertainty awareness. A mismatch between the ground reality and the optimization at the arbitrator occurs when the long-term average of availability duration is not taken into account at the arbitrator and when the durations advertised by the service providers are used as constraints in the optimization problem (presented in the previous subsection). However, our framework with uncertainty awareness achieves a smooth degradation (if any) in QoS (because of the small number of task migrations) when churn rate increases as it effectively exploits the knowledge gathered over time and/or acquired from its predecessors.

3. PERFORMANCE EVALUATION

We have implemented a small-scale prototype of the proposed framework and performed an experimental evaluation. We have also used simulations to show the scalability of the framework be-

Table 1: Heterogeneity of computing devices in the testbed.

	Samsung Galaxy Tab	Motorola Atrix 2	Samsung Galaxy S	LG Optimus	HTC Desire HD	Dell Netbook	Dell Laptop
CPU	1GHz Dual-core ARM	1GHz Dual-core ARM	1GHz ARM	600 MHz ARM	1GHz ARM	1GHz Atom	2GHz Dual-core Intel
Memory (RAM)	1GB	1GB	512MB	512MB	786MB	1GB	2GB
Network	802.11b/g/n, Bluetooth (BT)	2/3/4G, BT, 802.11b/g/n	2/3G, 802.11b/g/n, BT	3G, 802.11b/g, BT	2/3G, 802.11b/g/n, BT	802.11b/g/n, BT	802.11b/g, BT
Battery capacity	28Wh	6.612Wh	5.7Wh	5.7Wh	5.32Wh	49.95Wh	55.5Wh
Workload completion time	150s	300s	390s	590s	340s	100s	35s

yond ten nodes (the size of our testbed). For statistical relevance, we performed multiple trials until we achieved a very small relative confidence interval (less than 10%). First, we present details about our testbed and our experiment methodology. Then, we discuss specific experiment scenarios and the results that demonstrate the autonomic capabilities of our framework.

3.1 Testbed and Experiment Methodology

Heterogeneous devices: The testbed consists of Android- and Linux-based mobile devices with heterogeneous capabilities (summarized in Table 1). In our prototype, communications among the master and workers as well as among the optimizer and workers happen over *Comet Space* [13], a scalable peer-to-peer content-based coordination space developed at the NSF Cloud and Autonomic Computing Center, Rutgers University.

The workload: The mobile application that we used for our experiments is *distributed object recognition*. The service requester (which is also the data provider) submits an image of any object that needs to be recognized while also specifying a deadline. The predominant workload in this application is matrix multiplication and the most fundamental workload task is vector multiplication, which is assigned to the different service providers. Distributed object recognition is representative of a wide range of data-parallel applications that our framework can support. Table 1 shows the time taken by the different mobile devices to complete all the workload tasks of our application when operating in isolation. For near-real-time performance, the delay needs to be in the order of tens of seconds and the numbers clearly motivate the need to divide the tasks among service providers in the vicinity for speed up.

Application profiling: As the objective of the optimization problem is maximization of minimal residual battery capacity, the amount of battery drain in service providers as a result of running workload tasks needs to be estimated and used in decision making. However, the usage of actual Watt-hour (Wh) is unfair to devices with a higher battery capacity. Hence, in order to exploit heterogeneity and to ensure fairness, our prototype uses the residual battery capacity percentage instead of actual Wh values.

In order to ascertain battery drain while running a workload task, first, we ran all the workload tasks of our object recognition application on the individual mobile devices and measured the current drawn in mA (as the voltage drop remains constant) and the total time taken for the workload completion. Then, we determined the average time taken to complete one task. Such a straightforward estimation is possible as object recognition is a data-parallel application whose task (vector multiplication) completion time is not affected by the type of input. Information about task completion time along with the current drawn by the workload tasks and the number of tasks allocated to a service provider helps us calculate the resulting battery drain in Wh. Information about battery current consumption is readily available in most Android-based devices.

3.2 Self-optimization

Competing approaches: To assess the self-optimization capability of our framework, we compare it against two competing approaches: i) *Round-robin,* in which the workload tasks are divided equally among all the available service providers, and ii) *CometCloud* [11], a pull-based task-scheduling mechanism in which the service providers voluntarily pull tasks from the arbitrator, work on them, report the result, and pull the next task to work on. Round-robin is chosen for comparison to show the gains (in terms of application response time and battery drain) that can be achieved by exploiting the heterogeneity in computing capabilities of service providers. CometCloud inherently exploits the heterogeneity in computing capabilities as it schedules tasks on a First-Come-First-Served (FCFS) basis resulting in progressively faster devices completing a correspondingly higher number of tasks over time. It is also robust to service provider failures or loss in connectivity as it is purely pull-based. However, due to lack of self-optimization, there is usually unfair battery drain at the service providers.

Setup: In order to show the superiority in performance of the proposed energy-aware resource allocation engine over the competing approaches under different operational scenarios (in terms of number and combination of service providers), we ascertain and compare the fairness in battery drain when each of the three task-scheduling mechanisms are employed. We use *Jain's fairness index* ranging in [0,1] (1 being the highest and 0 being the lowest) as measure of fairness. The four scenarios in Fig. 2 represent a progressive increase in the scale and the heterogeneity of the underlying service provider pool as well as the problem size (while keeping the deadline constant at 60s). The scaling up is achieved by increasing the resolution of the object's image, which is the input to the object recognition application. In order to determine the amount of battery drain while using the three task-scheduling mechanisms, we simulated 100 consecutive runs (for significant battery drain) of the workload. This procedure is referred to as one trial. We in turn performed multiple trials, each with a different initial condition in terms of available battery capacities in the service providers.

Observations: Figure 2 shows the average fairness in terms of residual battery capacity at the service providers after each trial. Our proposed solution achieves the best performance in terms of fairness in the residual battery capacity as it fully exploits the heterogeneity of the devices in the resource pool to achieve its objective while meeting the user-specified deadline.

3.3 Uncertainty-aware Self-organization

Setup: In order to show the uncertainty-aware self-organization capability of the proposed framework, we performed an experiment to ascertain the gain in terms of reduction in number of workload task migrations that can be achieved by using our framework. The evaluation was carried out under different operational scenarios with different service provider churn rates. The four scenarios

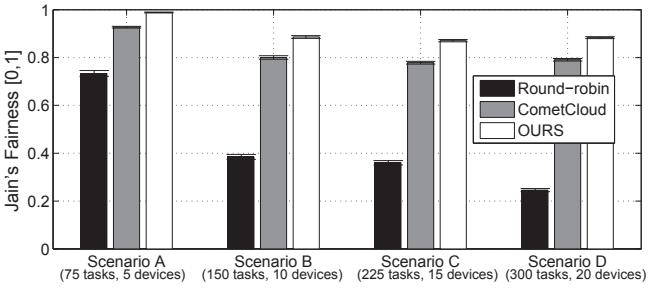


Figure 2: Performance of proposed framework (in terms of fairness) versus CometCloud and round-robin approaches.

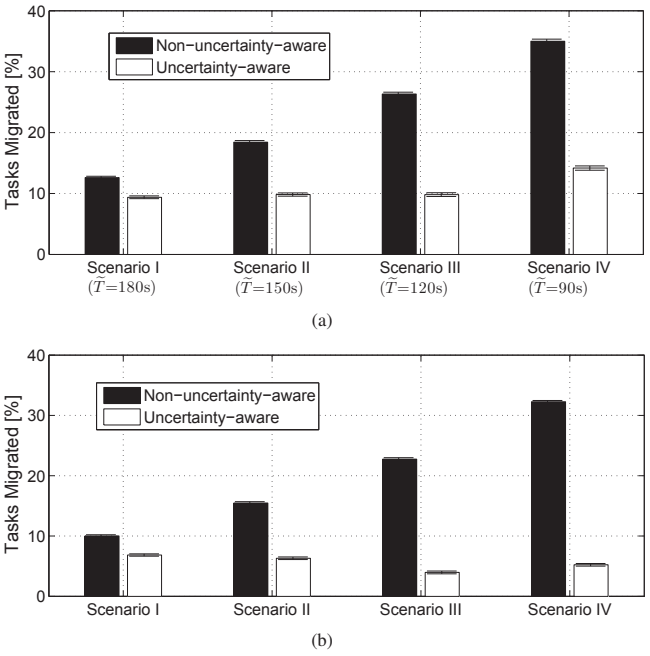


Figure 3: Effectiveness of uncertainty-awareness (in terms of % tasks migrations) when service providers' availability duration follows (a) Normal and (b) Weibull distributions.

(different from the previous ones) in Fig. 3 represent a progressive increase in the churn rate of the underlying resource pool (with a corresponding decrease in average duration of association with the arbitrator). The number (15 in total) and combination of service providers in the mobile grid, the number of workload tasks, and the deadlines remain the same for all the four scenarios.

We used *percentage of migrated workload tasks* to determine the effectiveness of uncertainty awareness. In order to ensure that the uncertainty awareness capability is not dictated by any particular distribution of service-provider-availability duration, it was picked at random based on i) Normal distribution (with mean $\mu = 180, 150, 120, 90$s; and standard deviation $\sigma = 60$s) and then on ii) Weibull distribution (with scale $\lambda = 200, 175, 150, 125$; and shape $k = 4$). Normal distribution is used for its generality while Weibull distribution is the most popular choice amongst statisticians performing reliability (or survivability) analysis.

Observations: Figures 3(a) and (b) show how the arbitrator leverages its knowledge of the long-term average of service provider availability in order to reduce the number of workload task migrations. As the churn rate of service providers increases, i.e., the average duration of service providers availability decreases, the percentage of migrated workload tasks increases when we use our resource allocation engine "without" uncertainty awareness. When the advertised durations (from service providers) are used as constraints in the optimization problem it leads to a mismatch between the ground reality and the optimization at the arbitrator. However, our framework with uncertainty awareness achieves a smooth degradation (if any) in QoS (because of the small number of task migrations) when churn rate increases as it effectively exploits the knowledge gathered over time. Also, another advantage of uncertainty awareness is that it helps decrease churn rate, especially service provider departures caused by device users opting out of the application due to undesired battery drain.

4. CONCLUSIONS AND FUTURE WORK

We proposed a novel resource-provisioning framework for organizing the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in the vicinity in order to a form a mobile computing grid. We imparted the resource-provisioning framework with autonomic capabilities, namely, self-optimization and self-organization, in order to be energy and uncertainty aware in the dynamic mobile environment. We demonstrated the autonomic capabilities of the framework through experimental evaluation on a prototype testbed. Currently, we are investigating mechanisms for imparting the self-healing capability, i.e., for handling uncertainty arising out of inaccurate estimation of task completion times.

5. REFERENCES

[1] Hyrax: Cloud computing on mobile devices using mapreduce. http://www.dtic.mil/docs/citations/ADA512601.

[2] D. Chu and M. Humphrey. Mobile OGSI.NET: Grid Computing on Mobile Devices. In *Proc. of IEEE/ACM Intl. Workshop on Grid Computing*, Nov. 2004.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. CloneCloud: Elastic Execution between Mobile Device and Cloud. In *Proc. of The European Professional Society on Computer Systems (EuroSys)*, Apr. 2011.

[4] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco. TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks. In *Proc. of the Intl. Workshop on Middleware for Sensor Networks (MidSens)*, Nov. 2006.

[5] P. J. Darby and N. F. Tzeng. Peer-to-peer Checkpointing Arrangement for Mobile Grid Computing Systems. In *Proc. of Intl. Conf. on High-Performance Parallel and Distributed Computing (HPDC)*, June 2007.

[6] L. dos S. Lima, A. T. A. Gomes, A. Ziviani, M. Endler, L. F. G. Soares, and B. Schulze. Peer-to-peer Resource Discovery in Mobile Grids. In *Proc. of the Intl. Workshop on Middleware for Grid Computing (MGC)*, Nov. 2005.

[7] D. Estrin and I. Sim. Open mHealth Architecture: An Engine for Health Care Innovation. *Science*, 330(6005):759–760, Nov. 2010.

[8] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso. Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications. In *Proc. of the ACM/IFIP/USENIX Intl. Conf. on Middleware (Middleware)*, Nov. 2009.

[9] Y. Huang, S. Mohapatra, and N. Venkatasubramanian. An Energy-efficient Middleware for Supporting Multimedia Services in Mobile Grid Environments. In *Proc. of Information Technology: Coding and Computing (ITCC)*, Apr. 2005.

[10] N. Jiang, C. Schmidt, V. Matossian, and M. Parashar. Enabling Applications in Sensor-based Pervasive Environments. In *Proc. of Broadband Advanced Sensor Networks (BaseNets)*, Oct. 2004.

[11] H. Kim, Y. el Khamra, I. Rodero, S. Jha, and M. Parashar. Autonomic Management of Application Workflows on Hybrid Computing Infrastructure. *Telecommunication Systems*, 19(2-3):75–89, Feb. 2011.

[12] E. K. Lee, H. Viswanathan, and D. Pompili. SILENCE: Distributed Adaptive Sampling for Sensor-based Autonomic Systems. In *Proc. of the Intl. Conf. on Autonomic Computing (ICAC)*, June 2011.

[13] Z. Li and M. Parashar. Comet: A Scalable Coordination Space for Decentralized Distributed Environments. In *Proc. Intl. Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, July 2005.

[14] W.-H. Liao, J.-P. Sheu, and Y.-C. Tseng. GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks. *Telecommunication Systems*, 18(1-3):37–60, 2001.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), Oct.-Dec. 2009.