Dynamic Collaboration Between Networked Robots and Clouds in Resource-Constrained Environments

Parul Pandey, *Student Member, IEEE*, Dario Pompili, *Senior Member, IEEE*, and Jingang Yi, *Senior Member, IEEE*

***Abstract*—Underwater mobile sensor networks such as Autonomous Underwater Vehicles (AUVs) or robots are envisioned to enable applications for oceanographic data collection, environmental and pollution monitoring, offshore exploration, and distributed tactical surveillance. These applications require running compute- and data-intensive algorithms that go beyond the capabilities of the individual AUVs that are involved in a mission. To execute these task-parallel algorithms in resource- and time-constrained environments, dynamic and reliable collaboration between local networked robots (e.g., AUVs) and remote public Clouds is needed. To this end, the heterogeneous sensing, computing, communication, and storage capabilities of local and remote resources are exploited to form a "loosely coupled" mobile Cloud, and a novel resource provisioning engine that dynamically takes decisions on "what" and "where" the tasks should be executed in the mobile Cloud is introduced. Comparison of benefits of collaboration between local and Cloud resources with purely local and centralized approaches are presented through exhaustive computer simulations.**

***Note to Practitioners*—The mission length and operations of any underwater application such as data collection, environmental monitoring, and undersea exploration are severely limited by battery capacity of AUVs. During the course of a mission, many computation-intensive tasks have to be executed, along with establishing communication with other vehicles in the team, which leads to further consumption of battery capacity. In this paper, a communication framework is introduced that expands the resources (computation and data resources) available to the team of AUVs by including public Clouds. A public Cloud consists in a set of networked computers that provide a range of computation and storage resources on demand and at a nominal price. A resource provisioning engine is designed to share the workload between local and Cloud resources based on communication cost, computation cost, and battery capacity. Such a framework enables increasing the lifetime of vehicles, execution of tasks with higher accuracy, and exploitation of any external information in the Cloud that is not available to a team of AUVs in the field.**

***Index Terms*—Amazon EC2, autonomous underwater vehicles, cloud computing, robot coordination.**

Manuscript received December 08, 2014; accepted January 28, 2015. Date of publication March 06, 2015; date of current version April 03, 2015. This paper was recommended for publication by Associate Editor M. Dotoli and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation (NSF) CAREER Award OCI-1054234.

P. Pandey and D. Pompili are with the Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ 08901 USA (e-mail: parul_pandey@cac.rutgers.edu; pompili@cac.rutgers.edu).

Dr. J. Yi is with the Department of Mechanical Engineering, Rutgers University, New Brunswick, NJ 08901 USA (e-mail: jgyi@rutgers.edu).

Digital Object Identifier 10.1109/TASE.2015.2406115

I. Introduction

UNDERWATER mobile sensor networks have been envisioned to enable a variety of applications such as environmental and pollution monitoring, disaster prevention, and assisted navigation. These mobile networks are formed by variety of sensors attached to Autonomous Underwater Vehicles (AUVs), low-power propeller-less (buoyancy-driven) gliders, or unpowered drifters. The information extracted from the raw data (such as dissolved oxygen, pH, turbidity, and algae concentration) collected by these mobile sensors provide us with greater knowledge of physical, chemical, geological, and biological variables of an aquatic environment. This information is vital for early detection and forecasting of environmental changes and of their effects on biodiversity, coastal ecosystems, and climate. For example, detecting in real time the changes in water-quality of rivers, lakes, and water reservoirs from raw data is critical to prevent contaminated water from reaching the civilian population by deploying appropriate treatment solutions or at least by issuing early warnings. Water-contamination incidents such as the one occurred in West Virginia on January 9, 2014 caused by a chemical spill [1] may lead to the non-availability of drinking water for thousands of residents, livestock and irrigation, and may also result in adverse effects on aquatic life. Enabling *near-real-time* data acquisition and processing of high-resolution, high-quality, heterogeneous data from water bodies would allow the immediate withdrawal of water to a plant in the case of an emergency and save damage to human and aquatic life.

Deploying a static sensor network, with a predefined configuration of sensing nodes, or a mobile network with fixed/random trajectory of vehicles is not among the most effective ways to gather data as the area of scientific interest may occur *sporadically* and propagate *dynamically* through the water bodies. AUVs–with their ability to drift, drive, or glide through vast water bodies without real-time control by human operators [2]–have presented themselves as a new tool for a multitude of applications ranging from environmental monitoring to surveillance. However, light AUVs for low-depth water bodies such as lakes and rivers are battery-operated vehicles and, hence, mission duration and operations are severely limited by the vehicles' battery capacity. Moreover, any mission requires execution of multiple computationally demanding tasks on on-board processors, whose complexity increases as the size of the 3D region of interest being sampled increases. Examples of such tasks include optimal vehicle path planning, estimation of environmental phenomena from the raw data collected during a mission, sample-based statistical modeling, information

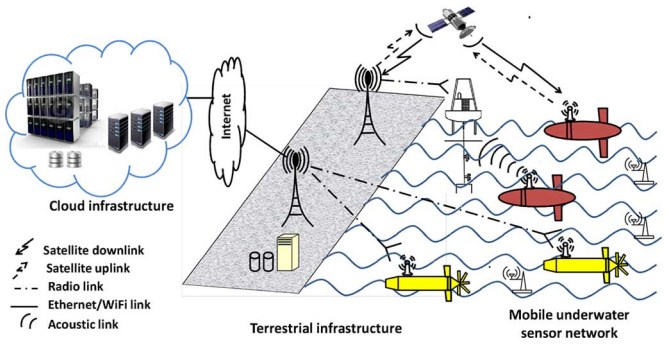

Fig. 1. Heterogeneous computing pool constituting *local resources* [static sensors/teams of Autonomous Underwater Vehicles (AUVs)] and *remote resources* (Public Clouds, on-shore station, satellites) to enable near-real-time computation-intensive underwater applications.

fusion, etc. Furthermore, these vehicles communicate among themselves, with the on-shore station, and satellite during the course of a mission, which leads to further consumption of battery capacity [3].

Public Clouds have presented themselves as a new opportunity for the scientific community to enable computationally intensive applications. A public Cloud consists in a set of networked computers that provide a range of computation as well as storage resources and that give the appearance of *infinite* computing capabilities available on demand and at a nominal price [4]. In this work, we present a novel collaborative framework (as shown in Fig. 1) that combines the benefits of both the mobile sensor network of AUVs (for *in-situ* measurements) and the Clouds (endowed with high-computing capabilities). Using Cloud resources allows energy-hungry AUVs to outsource their computation to the Cloud and, hence, to extend their lifetime and consequently the mission length. We envision a resource provisioning framework that enables collaboration between local and Cloud resources, and makes dynamic decisions on "what" and "where" the tasks in an application should be executed based on network latency, deadline to finish the application, network conditions, and vehicle battery capacity. We present a representative application, water-quality monitoring of river, lakes, and reservoirs to show the benefits of our proposed framework.

The proposed framework is the first work to enable interaction between Cloud and mobile underwater vehicles for near-real-time applications for aquatic environment. Our novel resource allocation engine enables real-time water-quality monitoring by sharing computation tasks between local and remote resources in terms of computation by dividing the tasks between the local and Cloud resources. We support collaboration based on two different objectives, namely: i) minimizing energy and ii) minimizing cost of using Cloud resources. We further propose two polynomial-time heuristics on how to share tasks between different Cloud and local resources based on these two objectives. Also, the proposed collaborative framework allows seamless task migration from local network to the Cloud.

To summarize, our main contributions in this paper are the following:

- We present a framework—composed of resources in local network (AUVs and static sensors) and remote network (Public Clouds)—that forms a "loosely coupled" mobile/fixed Cloud, where the resources collaborate to enable high-accuracy, high-resolution applications.
- We present two polynomial-time heuristics to solve the resource-allocation problem depending on the objective of the mission, i.e., saving energy of local resources or cost of using Cloud resources.
- We demonstrate the benefits of the exploitation of Cloud resources, and compare the performance of our solutions with a purely local or a purely centralized approach.

The rest of the paper is organized as follows. In Section II, we discuss the related work in the area of Cloud robotics. In Section III, we present the different entities of our proposed communication framework. In Section IV, we introduce our novel resource provisioning framework to allocate tasks between Cloud and local resources. In Section V, we describe our experimental methodology and results. Finally, in Section VI, we provide the conclusive summary.

II. Related Work

Cloud robotics offers a new paradigm by providing computational capabilities and storage space much beyond that supported by current robotic infrastructures. This opens up new applications and widens the scope of current ones. We briefly review the work done so far in the area of Cloud robotics and automation, and discuss how it differs from our contributions.

Current Cloud robotic projects provide access to global libraries of images, maps, and object data annotated with geometry and mechanical properties. Projects such as RoboEarth [5] have developed a giant database where robots share information about objects, environments, and tasks. This database stores knowledge generated by humans and robots in a machine-readable format. The data set includes software components, maps for navigation, task knowledge, and object-recognition models. In our work, instead, we use Clouds as an on-demand computational resource for execution of tasks rather than simply as a database.

Current solutions provide massively parallel computation on demand for computationally intensive tasks like optimal motion planning and sample-based statistical modeling. In [6], the author proposes Cloud-enabled robots that can offload CPU-heavy tasks to remote servers, relying on smaller and less power-hungry on-board computers. Similar to this work, few researchers at Singapore's A-Star Social Robotics Laboratory (ASORO) have built a Cloud-computing infrastructure that allows robots to generate 3D maps of their environments [7]. Some authors have provided a software framework where many of the robotic computationally intensive algorithms are parallelized as Map/Reduce tasks in Hadoop [8]. This framework is presented as Software-as-a-Service (SaaS). We, on the other hand, use the Cloud as Infrastructure-as-a-Service (IaaS) to enable cooperation between Cloud and local resources and execute parallel tasks. Also, our resource provisioning framework, while allocating tasks to Cloud resources, considers the application deadline and quality of wireless network connectivity, which the current solutions are agnostic about.

Current robotic cloud solutions enable sharing of open-source code, data, and designs for programming, experimentation,

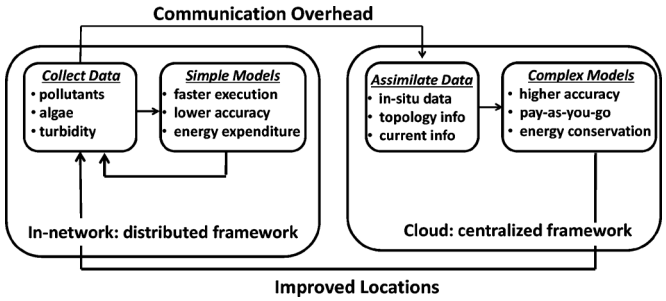

Fig. 2. Interaction between local and Cloud resources, which contribute *uniquely* to the provisioning framework, i.e., the former offer *in-situ* data while the latter offer vast computing capabilities.

and hardware construction. In [9], the authors present an open source Platform-as-a-Service (PaaS) framework called Rapyuta, which is designed specifically for robotics applications such as RoboEarth. In our work, we use the Cloud as an IaaS and, depending on deadline and monetary constraints, we select the required hardware/software resources in the Cloud.

The current robotic cloud solutions also enable sharing of outcomes, trajectories, and dynamic control policies. In [10], the authors present a framework where a distant group of robots can share and exchange learned skills through Cloud services. These services have been represented as a storage repository in an assisted-living scenario in order to utilize information obtained from spatially separated robots and reproduce a user's situation. This is different from our work where we conduct on-demand execution of tasks in the Cloud rather than using them as a storage space.

III. Resource Provisioning Framework

We first introduce our envisioned heterogeneous computing framework, which is composed of resources from both the local and the Cloud resources. We explore the communication and computing capabilities of different entities in the framework, and describe the logical roles of each of the resources to enable different applications in underwater mobile sensor networks. We utilize the water-quality monitoring application as an example to show how similar applications can benefit from collaboration between local and remote resources. We also present collaboration of resources in the field (rivers, lakes, or water reservoirs) to enable distributed coordination control and collision avoidance for a team of vehicles.

A. Entities of Our Framework

We now explore the physical architecture of our framework, which constitutes the computing and communication capabilities of different resources. Next, we describe the logical roles of each of the resources used to enable different underwater applications in resource-constrained environments. Fig. 2 illustrates how these service providers interact with each other and the benefits of using each one of them.

Physical Architecture: The resources in the field (e.g., AUVs, drifters, static sensors, and buoys) are part of the *local resources*. Each AUV in a team involved in a mission is equipped with satellite phones as well as with radio modems. The vehicles

TABLE I
Specification of Different Lightweight AUVs

	Folaga	**REMUS 100**	**Bluefin-12**	**BlueStar**
Manufacturer	Graal Tech	WHOI	Bluefin Robotics	National Univ. of Singapore
Communication	Acoustic, Iridium, RF, WiFi, GPRS	Acoustic, Iridium, WiFi	Acoustic, Iridium, RF	Acoustic, WiFi, GSM
Battery [Wh]	540	22	4.5	110
Max. Depth [m]	100	100	200	100
Speed Range [m/s]	$0-1$	$0-0.5$	$0-5$	$0-1.54$

TABLE II
Performance of Different Amazon EC2 Instances

Instance	**ECUs** [Cores]	**RAM** [GB]	**Archi** [bit]	**I/0** [Perf.]	**Disk** [GB]	**Cost** [$/h]
m1.small	1	1.7	32	Med	160	0.1
m1.large	4	7.5	64	High	850	0.4
m1.xlarge	8	15	64	High	1690	0.8
c1.medium	5	1.7	32	Med	350	0.2
c1.large	20	7.0	64	High	1690	0.8

communicate with the surface buoy via acoustic communication and the buoy communicates with the surface station via RF. Table I lists the specifications of a few different lightweight AUVs on the market in terms of communication capabilities, total energy (battery capacity), maximum depth, and speed. The on-shore base station could range from a simple laptop to an enclosure of servers endowed with an array of RF modems. On-shore station, satellite, and the Cloud form the *remote resources*. The on-shore station communicates with the surface buoys via an on-board RF modem and with the remote Cloud resources via an Internet (e.g., Ethernet or WiFi) connection. Allocating resources ahead of time (e.g., in a private datacenter) may lead to *over- or under-provisioning* of resources, leading to inefficiencies or to the infringement of the application service level agreement, respectively. Conversely, public Clouds provide resources on demand, thereby eliminating the need for users to plan far ahead for Cloud service provisioning. Also, the users pay only for the computing resources they use and free the machines as well as storage space when they are no longer needed. As a result, only a basic infrastructure is needed at the on-shore station (to communicate between local resources and Clouds), which saves time and cost of maintenance of a private datacenter. Table II presents the various computing resources in terms of Virtual Machines (VMs) and storage available in a public Cloud Amazon EC2.

Logical Roles: In our framework, we assign logical roles to each of the resources based on the tasks performed by them. This is an extension of our previous middleware work where we only considered *in-situ* resources [11]. Our vision is to organize the heterogeneous sensing, computing, and communication capabilities of fixed devices (in remote resources) and mobile devices (AUVs) to form an *elastic resource pool*. The entities of this resource pool play one or more of the following logical roles: (i) *requester*, which places requests for application workloads that require additional data and/or computing resources

from other devices, i.e., AUVs; (ii) *service provider*, which can be a data provider (a sensing device like AUV or static sensors), a resource provider (a computing device) or both, i.e., Cloud instances and AUVs; and (iii) *arbitrator*, which is in charge of handling requests and orchestrating the execution of applications in the heterogeneous computing environment, i.e., on-shore station. The on-shore station communicates its decision to the buoy, which in turn communicates with the AUVs. These vehicles may or may not rely on underwater acoustic communications to make resource-allocation decisions depending on the achievable bandwidth of such communication interface.

B. Two Representative Applications

We present here two representative applications that can be enabled by our framework: the first is water-quality monitoring, which illustrates a possible collaboration between Clouds and local resources to help AUVs save energy and increase their mission length; the second allows AUVs to serve as a resource pool to enable distributed coordination control and collision avoidance for a team of vehicles while they are at the field collecting raw data for the mission.

Water-Quality Monitoring via Adaptive Sampling: Every year hundreds of billions of gallons of untreated sewage flow into our rivers, lakes, and coastal waters. Untreated sewage contains a wide array of pathogens, chemicals, and nutrients, many of which pose a serious threat to human health. Impact to humans from these harmful algae include severe illness and potential death following consumption, or indirect exposure to toxins in algae. Moreover, these chemicals cannot be treated by basic water-treatment methods and require additional biological treatments. To enable near-real-time detection and tracking of these algae blooms in rivers, we leverage the work done by Smith *et al.* in [12]. Fig. 3 shows the steps undertaken to detect the presence of algae bloom in a water body. Based on raw data from a preliminary scanning of a river, we can detect the presence of algae bloom. In the next step, we generate locations (waypoints) for the AUVs to detect the algae concentration, which requires tracking the centroid and boundary of the algae concentration. As this concentration moves with the water body, the Regional Ocean Modeling System (ROMS) is used to predict the hourly forecast of location of algae bloom for a certain duration, which will help detect the movement of algae over a period of time. While a team of AUVs is executing the mission in the field, e.g., by performing adaptive sampling, the Cloud continues to track the phenomenon at a higher resolution for both spatial and temporal scales so to predict with higher accuracy the movement of algae bloom. If the phenomenon does not follow the pre-determined trend, the team of AUVs communicate with the satellite, receive a new/updated trajectory from the Cloud based on the execution of high-resolution prediction models, and retrigger local decision making.

Coordination Control for a Group of AUVs: To prevent collisions among AUVs in a team and help the vehicles follow a specified trajectory, we consider velocity-consensus control. We have previously proposed algorithms in the area of formation and steering for a team of AUVs [13]. These algorithms rely on underwater acoustic communications and were shown

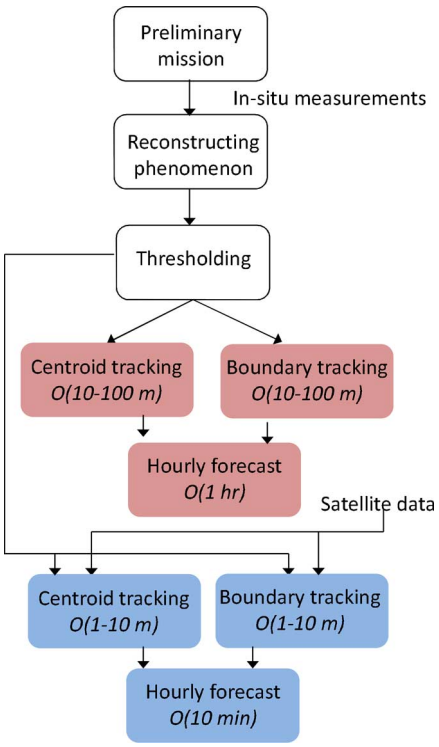

Fig. 3. Water-quality-monitoring workflow for detection and tracking of algae bloom in rivers. The application involves tasks at different spatial and temporal scales, and gives output at different resolutions. Blue-colored boxes (at the bottom) indicate tasks that are executed at a higher spatial and temporal resolution than those in red.

to be robust against ocean currents and acoustic channel impairments (e.g., high propagation and transmission delay, and low communication reliability). In this work, we employ a Local Minimum Spanning Tree (LMST)-based consensus control of multi-AUVs [14] for coordination control of team of AUVs. A *safety-region* concept is considered for each AUV with physical kinematics and dynamics constraints. Each AUV achieves the same velocity by communicating and exchanging its velocity and position information only with its neighboring AUVs. One of the advantages of the LMST-based adaptive sampling formation control of AUVs is that not only the connectivity of the AUV network is preserved but also the energy consumption and network communication quality are improved. Our proposed resource provisioning framework allows the team of AUVs to serve as a local resource pool so to enable distributed coordination control and collision avoidance. Underwater communication also requires estimating the position of AUVs in a team before communicating. In [15], we have presented a statistical approach to estimate the position uncertainty of AUVs in a team based on which a favorable network topology is predicted with relatively short links and transmission is postponed in favor of a lower transmission energy and a higher data rate in the future.

We now present our coordination control algorithm for a team of vehicles while they are at the field collecting samples. An N-AUV system is assumed to lay in a 2D planar space as we consider these vehicles to float in water at a particular depth (hence a 2D dimension for the AUVs is considered). We denote the AUV system as $\mathcal{R} := \{R_1, \ldots, R_N\}$, where the ith AUV

is denoted as R_i with a circular shape of radius s_i. For R_i, the (sensing) neighbors is a subset $\mathbf{SN}_i \subset \mathcal{R}$ that satisfies $\mathbf{SN}_i = \{R_j \in \mathcal{R} \,|\, \|\boldsymbol{r}_j - \boldsymbol{r}_i\| \le l_i^s\}$, where $l_i^s > 0$ is the sensing range of AUV R_i. We also define the *(physical) neighboring robots* for R_i as $\mathbf{N}_i = \{R_j \in \mathcal{R} \,|\, \|\boldsymbol{r}_j - \boldsymbol{r}_i\| \le l_i^R\}$, where $l_i^R > 0$ is the maximum communication range of R_i.

For motion planning and adaptive sampling formation, each AUV's motion is considered as discrete-time particle dynamics in 2D space as

$$\begin{cases} \boldsymbol{r}_i(h+1) = \boldsymbol{r}_i(h) + \boldsymbol{v}_i(h)\Delta T_c \\ \boldsymbol{v}_i(h+1) = \boldsymbol{v}_i(h) + \boldsymbol{u}_i(h)\Delta T_c \end{cases} \tag{1}$$

where $\boldsymbol{u}_i(h)$ is the controlled acceleration (at the hth step) and ΔT_c is the control updating time period. For AUVs R_i and R_j, let $\boldsymbol{r}_i, \boldsymbol{v}_i$, and $\boldsymbol{r}_j, \boldsymbol{v}_j$ denote their position and velocity vectors (in the navigation frame), respectively. The relative position vector is $\boldsymbol{r}_{ij} := \boldsymbol{r}_j - \boldsymbol{r}_i$ and the directional vector is $\boldsymbol{n}_{ij} := (\boldsymbol{r}_{ij})/(\|\boldsymbol{r}_{ij}\|)$. It is straightforward to calculate the relative distance $\Delta r_{ij} = \|\boldsymbol{r}_{ij}\|$ and the relative velocity magnitudes $\Delta \dot{r}_{ij} = (\boldsymbol{v}_j - \boldsymbol{v}_i) \cdot \boldsymbol{n}_{ij}$, where $\boldsymbol{v}_{ij} = \boldsymbol{v}_j - \boldsymbol{v}_i = -\boldsymbol{v}_{ji}$. We also define θ_{ij} as the angle between $\boldsymbol{v}_{ij}$ and $\boldsymbol{n}_{ij}$.

A *safety region*, denoted as X_{ij}^S, of two AUVs R_i and R_j is defined as a set of all triples $(\Delta r_{ij}, \Delta \dot{r}_{ij}, \theta_{ij}) \subset \mathbb{R}^2 \times \mathbb{S}$ such that the $\Delta r_{ij} > l_j$ and $\Delta \dot{r}_{ij} > 0$, where $l_j := s_i + s_j$. From the results in [14], the safety region between AUVs R_i and R_j is calculated as

$$\begin{aligned} X_{ij}^S = \big\{ (\Delta r_{ij}, \Delta \dot{r}_{ij}, \theta_{ij}) \;\big|\; & \theta_{ij} \ge \theta_{ij}^c \\ & \text{or } \theta_{ij} < \theta_{ij}^c,\ \Delta r_{ij} \ge \Delta r_{ij}^S(\theta_{ij}) \\ & \text{and } \|\Delta \dot{r}_{ij}\| \le v_{ij}^{\mathrm{m}} \big\} \end{aligned} \tag{2}$$

where $v_{ij}^{\mathrm{m}} = v_{\max}^{R_i} + v_{\max}^{R_j}, \theta_{ij}^c = \sin^{-1}(l_j/\Delta r_{ij})$, and $\Delta r_{ij}^S(\theta_{ij}) = \sqrt{(r_{ij}^m)^2 \sin^2\theta_{ij} + l_j^2 + 2 l_j r_{ij}^m} - r_{ij}^m \sin\theta_{ij}$, with $r_{ij}^m = (\Delta \dot{r}_{ij}^2)/(a_{max}^{R_i} \cos^2\theta_{ij})$. With the above safety-region definition, we calculate the collision-free region for AUV R_i

$$X_i^S = \bigcap_{j \in \mathbf{SN}_i} X_{ij}^S. \tag{3}$$

It is proven that X_i^S in (3) is a *convex set*.

Through communication, the AUVs team forms an undirected simple graph $G = (V, E)$, where V is the set of AUVs, and E is the edge set defined by the (physical) neighbors $\mathbf{N}_i$ of R_i, namely, $E = \{(R_i, R_j) \,|\, R_j \in \mathbf{N}_i\}$. For each AUV R_i, we assign a unique id, for example, $\mathrm{id}(R_i) = i$, and we denote $G_i = (V_i, E_i)$ as the induced subgraph of G such that $V_i = \mathbf{N}_i$. We construct the LMST in several steps. First, each node periodically broadcasts a `hello` message using its maximal transmission power to obtain its physical neighbors $\mathbf{N}_i$. Based on $\mathbf{N}_i$, AUV R_i can construct a local minimum spanning tree $T_i = (V(T_i), E(T_i))$ of G_i that spans all nodes within its neighbors in $\mathbf{N}_i$. The construction of LMST can be obtained using existing algorithms, such as the Prim's algorithm. Here a unique weight function (as a triplet) has been defined on the edge (R_i, R_j) as $(\|\boldsymbol{r}_i - \boldsymbol{r}_j\|, \max(id(R_i), \mathrm{id}(R_j)), \min(\mathrm{id}(R_i), \mathrm{id}(R_j)))$ such that the constructed LMST is unique. With the LMST, R_j is a logical neighbor of R_i, denoted as $R_i \to R_j$, *if and only if* $(R_i, R_j) \in E(T_i)$. The (logical) neighbor set $\mathbf{LN}_i$ of R_i is defined as $\mathbf{LN}_i = \{R_j \in V(G_i) \,|\, R_i \to R_j\}$.

With the above LMST topology, we consider the AUV mobility's impact on the local topology G. We assume that all N AUVs are distributed in the region and let $n_i = |\mathbf{N}_i|$ denote the number of physical neighbors of R_i, where $|\cdot|$ denotes the cardinality of a set. The probability p_e that any other AUVs enter the covered communication zone D_i of R_i can be calculated and is obtained in [14]. We then obtain the expected (or estimated) number of AUVs that enter D_i within period ΔT_c as $\hat{n}_i^e = (N - n_i - 1)p_e$. The number of nodes that leave D_i within ΔT_C can also be estimated as $\hat{n}_i^l = |\mathbf{L}_i|$, where $\mathbf{L}_i = \{R_j \in \mathbf{N}_i \,|\, \|\hat{\boldsymbol{r}}_j(t + \Delta T_c) - \boldsymbol{r}_i(t)\| > l_i\}, \hat{\boldsymbol{r}}_j(t + \Delta T_c) = \boldsymbol{r}_j + \boldsymbol{v}_j \Delta T_c$ is the estimated position vector of R_j in $\mathbf{N}_i$. With the estimate of the number of AUVs that are entering and leaving D_i, we obtain the estimated net change of the node number $\Delta n_i = \hat{n}_i^e - \hat{n}_i^l$ in $\mathbf{N}_i$ of AUV R_i.

For AUV R_i with dynamics specified in (1), we consider the following coordination control law:

$$\boldsymbol{u}_i(h) = \sum_{j \in \mathbf{LN}_i} a_{ij}(h)(\boldsymbol{v}_j(h) - \boldsymbol{v}_i(h)) \tag{4}$$

where $a_{ij}(h) > 0$ are the weighting factors. If we define $a_{ii}(h) = 0, i = 1, \ldots, N$, then the matrix $A(h) = [a_{ij}(h)]$ can be considered as a weighted adjacency matrix of LMST G. Note that, due to the AUV mobility, matrix $A(h)$ is time varying. We also consider the constraint $\sum_{j=1}^{|\mathbf{LN}_i|} a_{ij} = 1$, $a_{ij} > 0$ for a scaled velocity distribution among AUVs. To satisfy the requirements in collision avoidance, preserving connectivity among the AUV networks, and physical dynamic constraints, we consider to optimize the weighting factors $a_{ij}(h)$ over the safety region X_i^S and with the consideration of topology change among $\mathbf{N}_i$, namely

$$\begin{aligned} a_{ij}^*(h) = & \arg\max_{a_{ij}(h)} \Delta n_i(h) \\ & \text{s.t. } \boldsymbol{v}_i(h) \in X_i^S(h),\ |u_i(h)| \le a_{\max}^{R_i} \\ & \sum_j a_{ij}(h) = 1, a_{ij}(h) > 0. \end{aligned} \tag{5}$$

We have proven that, if there exists a set of a_{ij} by (5) such that the number of neighboring nodes is kept non-decreasing (i.e., $\Delta n_i(h) \ge 0$), then under the consensus control law (4), the AUV team $\mathcal{R}$ converges to the same velocity ($\boldsymbol{v}_i \to \boldsymbol{v}^*$ as $t \to \infty, \forall i$, where $\boldsymbol{v}^*$ is determined by the commanded initial configuration of the networks and AUVs.

IV. Proposed Solution

Our resource provisioning framework can be harnessed to enable innovative mobile applications that rely on *real-time* processing of massive amounts of sensor data generated in the field. Our idea is to offload certain tasks to the remote Cloud resources so that we can save the battery expenditure incurred in executing these tasks locally on AUVs. We represent our application workflow by a Directed Acyclic Graph (DAG), with tasks denoted by nodes in the graph. A directed edges indicate the information flow from one task to another. Nodes on the same level (i.e., stage) of the workflow are executed in parallel,

whereas nodes in consecutive levels are executed sequentially. Each stage is to be considered as *blocking*, i.e., the execution of tasks in successive levels (stages) can occur *only* when the tasks in the previous stage are completed (in general, in fact, the inputs of tasks at stage k are the outputs of the tasks at stage $k-1$). The DAG is defined by a tuple $D = (F, G)$, where F is the set of task nodes, $F = \{v_j, j = 1 : f\}$, and $f = |F|$ is the number of nodes, G is the set of communication edges, where $G = \{g_{i,j} = \langle v_i, v_j \rangle\}$ is the communication cost from node v_i to v_j, and $g = |G|$.

We assume that there are N AUVs, M types of Cloud instances, and K (equal to $|F|$) number of tasks in a workflow. Let t_i denote a task i in the workflow, with $t_i = \langle v_i, d_i \rangle$, where the task execution time depends on the computation of v_ith task with input data d_i. Also, let $L_n(t_i)$ denote the execution time of the task t_i at a local resource, i.e., an AUV in our case, and $R_m(t_i)$ denote the remote execution of task t_i at the m resource type. Furthermore, let O_{t_i} be the communication time for sending data from the local resources to the on-shore resources. Execution time for task t_i comprises of computation time (E) of the task and initialization time (IN), which includes the time required to acquire the resources and the communication time to receive the input data from the local resources. Let C_m denote the communication time required to send input data from the on-shore station to the m type Cloud instance, and C_n denote the communication time required to send input data between AUVs. S_m is denoted as the time required to acquire the mth Cloud instances and is zero for AUVs. Let P_m denote the price/hr of using the mth Cloud instance. Last, $\mathbf{I_l}$ and $\mathbf{I_m}$ denote the indicator functions.

To share the computation cost between local and Cloud resources we propose two policies: in the first, MinEnergy, the goal is to minimize the time to execute an application and, consequently, the energy expenditure of the mobile devices; while in the second, MinBudget, the goal is to minimize the price budget of using Cloud resources.

A. Minimization of Residual Energy of the AUVs (MinEnergy)

The objective of the first policy, MinEnergy, is to minimize the execution time of an application. This policy does not take into account the monetary expenditure of running the application in the Cloud. The parameters involved in the policy are residual battery capacity of AUVs, data rate from buoys to on-shore station and from on-shore station to Cloud resources, and computational capability of different VMs in the Cloud

$$\begin{aligned}
&\mathbf{Find}: \mathbf{I_l} = I_l(i,n); \mathbf{I_m} = I_m(i,m)\\
&\mathbf{Min}: P_E = I_l(i,n)\sum_{i=1}^{K} L_n(t_i)\\
&\qquad + I_m(i,m)\sum_{i=1}^{K}[O(t_i) + R_m(t_i)]\\
&\text{where}\ \ L_n(t_i) = E_n(t_i) + \mathrm{IN}_n(t_i);\\
&\qquad R_m(t_i) = E_m(t_i) + \mathrm{IN}_m(t_i);\\
&\qquad \mathrm{IN}_m(t_i) = C_m(t_i) + S_m(t_i);\\
&\qquad \mathrm{IN}_n(t_i) = C_n(t_i);\\
&\mathbf{S.t.}: \forall n = 1 \ldots \mathrm{N}, \forall m = 1 \ldots \mathrm{M}\\
&\qquad \sum_{n=1}^{N} I_l(i,n) + \sum_{m=1}^{M} I_m(i,n) = 1 \qquad (6)
\end{aligned}$$

where $L_n(t_i)$ denote the execution time of the task t_i at a local resource, and $R_m(t_i)$ denote the remote execution of task t_i at the mth resource type. Initialization time in the remote resource pool (IN_m) includes the time required to acquire Cloud resources from the remote server (S_m) and the communication time to receive the input data from the local resources (C_m). Initialization time in the local resource pool (IN_n) only includes the time required to receive the input data from the neighboring local resources (C_n). Equation (6) imposes that a task can be allocated to only one resource.

B. Minimization of Budget for Using Clouds (MinBudget)

The objective of the second policy, MinBudget, is to minimize the price budget of using Cloud resources. The resource provisioning engine involves remote resources in the execution of application only when it is within the budget

$$\begin{aligned}
&\mathbf{Find}: \mathbf{I_l} = I_l(i,n); \mathbf{I_m} = I_m(i,m)\\
&\mathbf{Min}: I_m(i,m)\sum_{i=1}^{K}(R_m(t_i)\cdot P_m)\\
&\mathbf{S.t.}: \sum_{n=1}^{N} I_l(i,n) + \sum_{m=1}^{M} I_m(i,m) = 1; \qquad (7a)\\
&\qquad P_B = I_l(i,n)\sum_{i=1}^{K} L_n(t_i) + I_m(i,m)\sum_{i=1}^{K}[O(t_i)\\
&\qquad\qquad + R_m(t_i)]; \qquad (7b)\\
&\qquad \frac{P_E}{P_B} = \delta \in \{0,1\} \qquad (7c)
\end{aligned}$$

where (7a) imposes that a task can be allocated to only one resource, (7b) represents the time taken for executing the application, and (7c) constraints the time taken to execute using policy MinEnergy, P_E, to be δ times of policy P_B, where δ, decided by the arbitrator, is a proper fraction (i.e., less than 1). This is done to meet the limited budget requirements. By making δ less than 1, the resource allocation framework prefers assigning tasks to local resources (no cost is associated to execute applications on them) and/or VMs with low computation capabilities (hence cheaper than VMs with high computational capabilities). As a result, the application is able to meet budget requirements at the cost of increase in execution time.

C. Heuristic for Policy—MinEnergy

Allocating tasks in a DAG to different resources in the framework so to minimize the execution time is similar to the problem of partitioning a DAG and allocating each partition to a unique resource such that the objective of minimizing execution time is achieved. However, graph partitioning is an NP-hard problem [16]; hence, we present a polynomial-time heuristic to solve this problem. For the policy MinEnergy there are no restrictions on the monetary requirement, hence, we can use any number of VMs in the Cloud without considering the price that we have to pay.

Algorithm 1: Heuristic: MinEnergy

Input: A-Application, K-No. of tasks, M-Type of VMs, N-No. of AUVs, IN- Initial list of nodes, FL-Free List, PFL-Partially Free List
Output: CLUST-Clusters of tasks
Communication to Computation Ratio:
$\widetilde{CCR(t_i)} = Md\left(\frac{E_{res}(t_i)}{IN_{res}(t_i)}\right) \; \forall res \in \{N, M\}$;
$\overline{CCR} = \frac{1}{K}\sum_{i=1}^{K} \widetilde{CCR(t_i)}$;
if $\overline{CCR} > 1$ **then**
  Construct DAG with edge ($\hat{e}_i$) and task weights ($\hat{v}_i$) of resources with the lowest communication cost
else
  Construct DAG with edge($\hat{e}_i$) and task weights ($\hat{v}_i$) of resources with the lowest computation cost
end
Task clustering:
Find Dominant Sequence (DS);
for $i \in IN \vee i \notin PFL$ **do**
  $tempFL = poll(Child(\hat{v}_i))$;
  if $CLUST(\hat{v}_i) \cup E(tempFL) <= DS$ **then**
    $CLUST(\hat{v}_i) \leftarrow CLUST(\hat{v}_i) \cup tempFL$
  end
  $IN = IN \setminus tempFL$;
  for $c \in$ Child($tempFL$) **do**
    if *All* Parent(c) $\in FL$ **then**
      $FL = FL \cup c$;
    else
      $EFL = EFL \cup c$;
    end
  end
end

Algorithm 2: Heuristic: MinBudget

Input: A-Application, K-No. of tasks, M-Type of VMs, N-No. of AUVs, Q_m-Queue length of m type of resource, LT-Lease Time as given in SLA
Output: SCHLD
Find task with maximum resource requirements
for $\forall i \in V$ **do**
  $task\_max\_resr = find \max_{v_i}(E(t_i) + IN(t_i))$;
  $V \leftarrow V \; task\_max\_resr$;
  Find computation resource with earliest finishing time
  $res = find \min(Queue\_len(res) + E(task\_max\_resr) + IN(task\_max\_resr))$;
  if $res \in M$ **then**
    Check residual time of instance
    if $Queue\_len(res) + (E(t_i) + IN(t_i)) < LT$ **then**
      $Queue(res) = Queue(res) \cup task\_max\_resr$;
    else
      Create new instance
      $Queue(new\_instance\_res) = task\_max\_resr$;
    end
  else
    $Queue(n) = Queue(n) \cup task\_max\_resr$;
  end
end
$SCHLD(task\_max\_resr) = res$;

A lot of work has been done in scheduling tasks in a DAG for homogeneous processor. However, our problem consists of allocating tasks to heterogeneous resources as each local and Cloud resources have different computational capability. We improve upon the solution in [17], which presents scheduling of tasks in a DAG to homogeneous processors.

The solution in [17] involves two lists: Free List (FL), which contains the nodes that have been allocated to a resource and will not be considered in the clustering algorithm, and Partially Free List (PFL), which contains the nodes whose at least one of the parent node is not in FL. Initially, FL is an empty set. In the beginning of the clustering algorithm, each task is given to a unique processor, which executes the task in the lowest possible time. The algorithm then finds the Dominant Sequence (DS), which is defined as the length of the longest path in the DAG. If on merging two sequential nodes, i.e., executing the task on the same VM and making the edge cost zero, the DS reduces, then these nodes (tasks) are executed on the same processor and are executed based on their order in DAG. The set of tasks executed on the same processor are said to form a cluster. This method continues until DS cannot be reduced further. Data-structure CLUST stores different clusters, where each cluster is a set of tasks executed at a unique processor. The advantage of this method is that, when the tasks are merged, the communication time between nodes becomes zero and the overall time taken by the DS decreases. This algorithm employs linear clustering, which clusters parallel tasks (tasks on the same level/stage) to different processors.

Algorithm 1 shows our proposed heuristic. To cluster tasks for heterogeneous processors, we recreate a new DAG. We first find the *Communication to Computation Ratio (CCR)*; this can be achieved by finding the median of the time taken by a task at different resources ($\widetilde{CCR(t_i)}$) and then taking an average over all these median times ($\overline{CCR}$). If in *high CCR domain* ($\overline{CCR} \gg 1$), then we create a task DAG by including the edge weight that gives the lowest communication cost of sending data to a particular resource and node weight of executing task in that particular resource. Conversely, if in *low CCR domain* ($\overline{CCR} \ll 1$), we create a new task DAG by including edge weight that gives the lowest computation cost of executing data at a particular resource and edge weight of sending data to that particular resource.

D. Heuristic for Policy—MinBudget

For the policy MinBudget, a bounded number of VMs is considered since we want to minimize the cost of using Cloud resources. In Amazon EC2, each VM instance is allocated for an hour and, if a task in an application completes execution in less than an hour, then another task can be reallocated to that VM. We present a greedy algorithm to allocate task to these resources. Our idea is to allocate the task that requires the maximum computation resource first and then add it to the resource that is δ times the best resource to execute the task. Our proposed heuristic is given in Algorithm 2. This heuristic task allocation is done by allocating tasks to resources that give the earliest finishing time. Each Cloud resource has a finite queue length (Queue_len), which is equal to the Lease Time (LT) given in the Service-Level Agreement (SLA). We order the tasks on the basis of consumption of resources (for computation as well as communication). We pick the task from this list that consumes the maximal resources. If there are more than one, we pick a task randomly from the set. The resource that gives the earliest finishing time for this task is selected and the task is added to its queue ($\text{Queue}_{\text{res}}$). If the addition of an incoming task makes the queue length become greater than the leased time, then a new instance of the resource is created and the task is given to the new instance. A schedule (SCHLD) is finally created in which tasks will be assigned to different resource providers.

V. Performance Evaluation

The focus of this section is to study the benefits of collaboration between local and Cloud resources in the representative water-quality monitoring application. We first demonstrate how

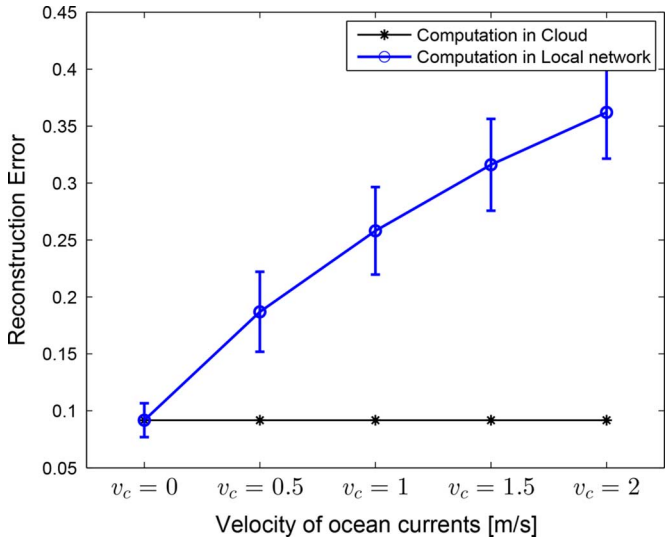

Fig. 4. Reconstruction error for temperature data as the speed of ocean currents is varied. We compare the scenario when computation is done locally with no knowledge of ocean currents versus when computation is done in Cloud with complete knowledge of currents.

additional knowledge about ocean currents increase the accuracy (in terms of reduction in reconstruction errors) of the reconstructed maps. Next, we quantify the benefits of collaboration between Cloud and local resources. We consider three cases: purely local computation of tasks, purely centralized computation of tasks in the Cloud, and collaboration between local and Cloud resources to execute a task. We performed our simulations in MATLAB using real-data traces.

Higher Accuracy: To estimate the accuracy of reconstruction algorithm in a purely local implementation, each AUV works with a subset of data to reconstruct a subset of a field, and then the field reconstructed by each AUV is consolidated to generate the map for the entire region. Local reconstruction has no knowledge of ocean currents. Although this reconstruction is faster and computationally less expensive, its accuracy is low. In a purely centralized reconstruction, the AUVs send all the data to the Cloud and the reconstruction is done at one of the Cloud instance. Here, the knowledge of ocean currents is available to the Clouds as they have the infrastructure required for estimation of ocean currents. In a collaborative effort, the local resources only receive data from the Clouds. The data here is ocean currents for the field being sampled. The local AUVs work on a subset of data because each mission results in the collection of a large amount of data that cannot be processed by on-board processors.

Simulation Setup: In our analysis, we use real-data traces (ocean surface temperature and salinity maps) from [18]. The data is from Southern California Bight region (the SCB is the oceanic region contained within 32°N to 34.5°N and 117°E to 121°E). The data is collected via satellite from a field of size 60 $\times$ 60 km^2 with 1 km of spatial resolution.

Observations: Fig. 4 shows the comparison of local and centralized reconstruction of algorithms when knowledge of ocean current is available to Clouds. We see that after including the knowledge of ocean currents the performance increases by 36% when the velocity of ocean currents is 1 m/s in the x direction and by 53% when the current velocity is 2 m/s in the x direction. As the current velocity increases, the performance of local reconstruction gets worse.

Execution Time: We compare the performance of our proposed collaboration framework to distribute tasks with respect to local and Cloud resources. Our goal is to understand the parameters that take a part when the tasks are shared between local and remote resources. Specifically, we focused on two parameters, namely, data rate of RF modems of AUVs and amount of data send from local network to Cloud resources.

Simulation Setup: We vary the data rate of the RF modem from 0.1 to 10 Kb/s and observed the execution time taken by the application. The AUVs communicate with each other by connecting to a buoy (local start topology). The buoys send data to the on-shore station via RF modem. We assume that the on-shore station is connected to the Internet and sends data to the Cloud over a high-speed connection.

Observations: In Fig. 5(a), we observe that, for a given amount of input data to be transferred from local to Cloud resources, the execution time depends heavily on the bandwidth of the medium. We see that the execution time for local nodes remains constant as they communicate via buoy. The communication time using our framework and from Cloud varies with the data rate and we see that, as the data rate increases for the RF modem, the execution time decreases. At the data rates 100 Kb/s and higher the collaborative framework performs as the Cloud framework because the data can be completely offloaded to the Cloud to get minimum execution time. We see that for 0.1 kb/s the execution time achieved by our framework is 38% less in comparison to execution only at Local resources, and 15% less than if the execution is carried out only at the Cloud resources. For 10 Kb/s, both Cloud and our framework perform 77% better than using only Local resources.

In Fig. 5(b), we observed that for a data rate of 0.1 Kb/s, the size of input data to tasks is varied. We observe how the variable input data changes the execution time for execution in local resources, Cloud resources, and collaborative effort. We observe that, as the data size increases, the execution time increases for both the framework and the Cloud. As the collaborative framework shares the resources between local and Cloud, it is able achieve the smallest execution time for $2 \cdot \text{Input data}$, where $Input\ data$ is used as the size of input (80 Kb) given to the application. However, for input data larger than that the execution time of local resources is the smallest.

Budget: We compare the performance of the two different policies considered above (i.e., MinEnergy and MinBudget) with a purely Cloud-based approach on the basis of execution time and budget.

Simulation Setup: For a fixed data rate of the RF modem from 0.1 Kb/s and fixed data input size, we observe the execution time taken by the application and budget in dollars for executing the application based on the two policies.

Observations: In Fig. 5(c), we see as expected that the execution is lowest for Policy–MinEnergy of our collaborative framework, as opposed to only Cloud resources, which has the higher execution time than Policy—MinEnergy. Policy–MinEnergy has lower cost and time as it shares tasks with local resources with no cost and lower communication

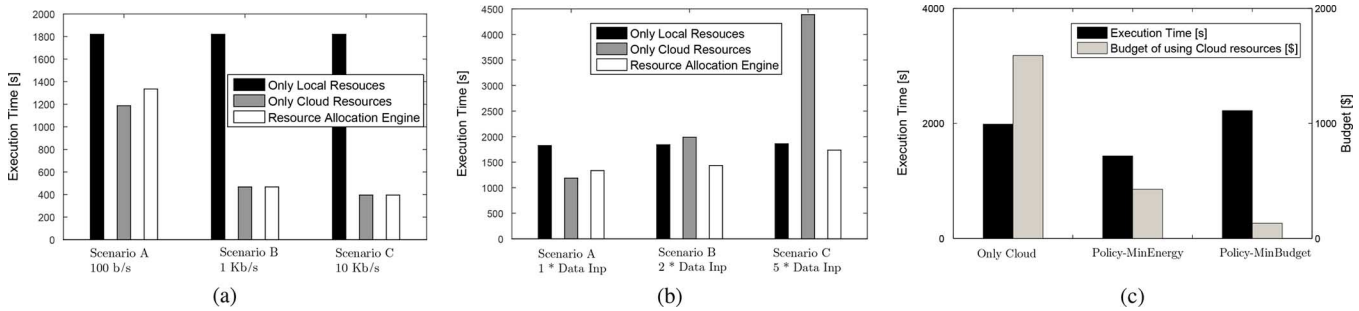

Fig. 5. (a) Comparison of execution time of water quality monitoring application when executed at only local resources, only Cloud resources, and shared between Cloud and local resources (Policy-MinEnergy). We vary the data rate of the RF modem to study the performance. (b) Comparison similar to (b) by varying the input data size of the application at a fixed data rate (1 Kb/s). (c) Variation of execution time and budget for only Cloud resources and our framework for different policies (MinEnergy and MinBudget). The data rate is fixed at (0.1 kb/s).

cost. However, if the bandwidth decreases further, the execution time will increase further, as shown in Fig. 5(a), and the framework might opt for doing tasks only in the local resources. However, as the data rate increases all the tasks will be send to the Cloud incurring lower execution time but higher cost. With Policy–MinBudget we get the lowest budget with highest execution times as it selects cheaper VMs with higher execution times. We see that Policy–MinEnergy consumes 47% more time than Policy–MinBudget; however, Policy–MinBudget gives 61% savings in budget for using Cloud resources. Both the techniques perform significantly better than completely offloading to the Cloud.

VI. Conclusion

We exploited the heterogeneous sensing, computing, communication, and storage capabilities of mobile and fixed devices in the field as well as in remote Clouds to form a loosely coupled mobile/fixed cloud. We presented a resource-provisioning engine that allocates computational tasks between the Cloud and the local resources based on the objective of minimization of execution time or the price of the executing the application. We presented via simulations the benefits of the collaboration between the Cloud and local resources for water-quality monitoring application using autonomous underwater vehicles. We demonstrated how the accuracy of a task can be improved by adding additional information in the Cloud, specifically how the accuracy of reconstruction of a phenomenon can be improved by also considering ocean currents. We demonstrated the benefits of our resource provisioning engine with respect to a purely local and centralized approach.

References

[1] "West Virginia Chemical Spill," [Online]. Available: http://www.bbc.com/news/blogsechochambers-25738286/
[2] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005.
[3] I. S. Kulkarni and D. Pompili, "Task allocation for networked underwater vehicles in critical missions," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 5, pp. 716–727, Aug. 2010.
[4] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[5] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, and A. Perzylo, "A world wide web for robots," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, 2011.
[6] N. Sadashiv and S. D. Kumar, "Cluster, grid and cloud computing: A detailed comparison," presented at the Proc. Int. Conf. Comput. Sci. Edu. (ICCSE), Singapore, Aug. 2011.
[7] ASTRO, [Online]. Available: http://www.asoro.astar.edu.sg
[8] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "DAvinCi: A cloud computing framework for service robots," presented at the Proc. Int. Conf. Robot. Autom. (ICRA), Anchorage, AK, USA, May 2010.
[9] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine," presented at the Proc. Int. Conf. Robot. Autom. (ICRA), Karlsruhe, Germany, May 2013.
[10] J. Quintas, P. Menezes, and J. Dias, "Cloud robotics: Towards context aware robotic networks," presented at the Proc. Int. Conf. Robot., Pittsburgh, PA, USA, Nov. 2011.
[11] H. Viswanathan, E. K. Lee, I. Rodero, and D. Pompili, "Uncertainty aware autonomic resource provisioning for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, to be published.
[12] R. Smith, A. Pereira, Y. Chao, P. P. Li, D. Caron, B. Jones, and G. Sukhatme, "Autonomous underwater vehicle trajectory design coupled with predictive ocean models: A case study," presented at the Proc. Int. Conf. Robot. Autom. (ICRA), Anchorage, AK, USA, May 2010.
[13] B. Chen and D. Pompili, "Team formation and steering algorithms for underwater gliders using acoustic communications," in *Computer Communication*. New York, NY, USA: Elsevier, 2012, vol. 35, pp. 1017–1028, no. 9.
[14] J. Yi, H. Wang, J. Liu, and D. Song, "LMST-based consensus control of multi-robot systems with kinodynamic constraints," presented at the Proc. ASME Dynamic Syst. Control Conf., Ann Arbor, MI, USA, Oct. 2008, in ..
[15] B. Chen and D. Pompili, "Modeling position uncertainty of networked autonomous underwater vehicles," in *Ad Hoc Networks*. New York, NY, USA: Elsevier, to appear in.
[16] A. E. Feldmann and L. Foschini, "Balanced partitions of trees and applications," in *Algorithmica*, 2012, pp. 1–23.
[17] T. Yang and A. Gerasoulis, "DSC: Sheduling parallel tasks on an unbounded number of processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 9, pp. 951–967, Sep. 1994.
[18] JPL, "OurOcean Portal, California Institute of Technology," 2012. [Online]. Available: http://ourocean.jpl.nasa.gov

Parul Pandey received the B.S. degree in electronics and communication engineering from the Indira Gandhi Institute of Technology, Delhi, India, and the M.S. degree in electrical and computer engineering from the Uuniversity of Utah, Salt Lake City, UT, USA, in 2008 and 2011, respectively. She is currently working towards the Ph.D. degree at the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, USA.

She is currently working on mobile and approximate computing, cloud-assisted robotics, and underwater acoustic communications under the guidance of Dr. Pompili as a member of the CPS-Laboratory.

Dario Pompili (S'04–SM'14) received the Laurea degree (integrated B.S. and M.S.) and the Doctorate degrees in telecommunications and system engineering from the University of Rome "La Sapienza," Roma, Italy, in 2001 and 2004, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 2007.

He is an Associate Professor with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, USA. He is the Director of the Cyber-Physical Systems Laboratory (CPS-Lab), which focuses on mobile computing, wireless communications and networking, acoustic communications, sensor networks, and datacenter management.

Prof. Pompilli is a Senior Member the Association for Computing Machinery (ACM). He is a recipient of the prestigious NSF CAREER'11 Award, the ONR Young Investigator Program'12 Award, and the DARPA Young Faculty'12 Award.

Jingang Yi received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1993, the M.Eng. degree in precision instruments from Tsinghua University, Beijing, China, in 1996, and the M.A. degree in mathematics and the Ph.D. degree in mechanical engineering from the University of California, Berkeley, CA, USA, in 2001 and 2002, respectively.

He is currently an Associate Professor of Mechanical Engineering, Rutgers University, Piscataway, NJ, USA. His research interests include autonomous robotic systems, dynamic systems and control, mechatronics, automation science and engineering, with applications to biomedical systems, civil infrastructure, and transportation systems.

Dr. Yi is a Member of the American Society of Mechanical Engineers (ASME). He is a recipient of the 2010 U.S. National Science Foundation (NSF) CAREER Award. He has coauthored papers that have been awarded several best papers at the IEEE/ASME AIM, ASME DSCC, IEEE ICRA, etc. He is currently an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and the *IFAC Journal Control Engineering Practice*, and has served on the IEEE Robotics and Automation Society Conference Editorial Board (since 2008). He was a Guest Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING in 2009 and an Associate Editor for the ASME Dynamic Systems and Control Division Conference Editorial Board from 2008 to 2010.