Robust Resource Allocation Using Edge Computing for Smart Oil Fields

Razin Farhan Hussain, Mohsen Amini Salehi, Anna Kovalenko
High Performance Cloud Computing (HPCC) Laboratory,
School of Computing and Informatics
University of Louisiana at Lafayette, LA, USA
{razinfarhan.hussain1, amini, aok8889}@louisiana.edu

Saeed Salehi
Petroleum & Geological Engineering Department
University of Oklahoma, OK, USA
salehi@ou.edu

Omid Semiari
Department of Electrical and Computer Engineering
Georgia Southern University, GA, USA
osemiari@georgiasouthern.edu

***Abstract*—Efficient and safe petroleum extraction in remote and offshore oil fields is a challenging and hazardous operation. To address this challenge smart oil fields have been proposed and deployed for remote offshore oil fields. An smart oil field includes a wide range of sensors (*e.g.,* for gas density, pipeline pressure, and temperature sensors) that collectively generate one to two terabytes of data per day. Most of the data needs to be analyzed in real-time for safety and decision-making purposes. Existing smart oil field solutions operate based on satellites communication and distant (onshore) cloud servers which are not feasible due to a significant transfer time delay. In this paper, we investigate the use of edge computing to obviate the challenges of such remote oil fields. We propose a robust resource allocation model which is aware of the connectivity, limited computational capacity, and resource intensiveness of applications in the oil fields. To achieve robustness, we develop a resource allocation model that efficiently allocates tasks to appropriate edge or cloud resource to satisfy the real-time constraint of applications. Evaluation results show that proposed model can significantly improve the performance of the system in compare to conventional cloud-based architectures. Hence, making oil and gas industry safer for workers and the environment.**

***Index Terms*—Cloud computing, Edge computing, Smart oil field, Robustness.**

I. Introduction

For nearly two centuries, petroleum has been a main natural resource used to produce many industrial products such as gasoline, diesel, oil, gas, asphalt, and plastic. Over the years, high demand for petroleum products led to a scarcity of natural resources in easy-access areas and forced companies to reallocate their oil and gas (O&G) extraction sites to remote offshore (*e.g.,* sea) reservoirs [23]. Meanwhile, operations at remote sites are costly, hazardous, and constrained with limited crew and equipment resources. Moreover, petroleum extraction is a fault-intolerant process that requires ultra-high reliability, specifically in the face of a disaster (*e.g.,* oil spill [5]). Therefore, achieving efficient and safe petroleum extraction —especially when coupled with the location constraints of remote offshore reservoirs is a challenging task for oil and gas (O&G) companies.

In order to address these challenges, smart oil fields have been proposed and deployed in over the past decade. Smart oil fields include a diverse set of sensors (*e.g.,* Gas density, Pipeline pressure, Temperature sensors, Fire / Gas / H2S alarms, Flow monitoring & Tank levels) and computational facilities. Real-time monitoring of the site, including rigs' structure, wells, and distribution lines is performed to avoid oil and gas leakage, identify corrosion level of the infrastructure, and predict potential future incidents to maximize production efficiency and minimize negative environmental impacts. Therefore, a large amount of raw sensor data (between one to two terabytes) is generated in a single day of operation an smart oil field [19]. With an absence of a full management crew at the site, real-time decision making depends on the ability to quickly process and analyze large amounts of data. The situation can become extremely hazardous due to uncertainties in O&G extraction process (stemming from stochastic gas pressure in the reservoir and leakage of hazardous gases such as H_2S) especially in presence of on-site human workers. This imposes a major challenge —data management and analytics. According to the Cisco Public white paper on New Realities in Oil and Gas [1] 48% of all respondents involved in O&G industry admitted that proper data management and analysis is the major challenge for acquiring the best efficiency from the smart oil fields.

Most of the generated data, such as those pertaining to drilling-platform safety, are time-sensitive and must be processed in real-time to be effective for decision makers. For instance, data obtained from sensors to monitor the release of toxic gases (*e.g.,* Hydrogen Sulphide (H_2S) which is common in remote oil fields) need to be processed in less than five seconds to preserve workers' safety [14]. Processing such volume of data requires high-end communication and computation facilities that are not available in remote (offshore) oil fields. Satellite connection is the common vehicle of transmitting data from oil fields to onshore Data Centers. However, the bandwidth of such connections ranges from 64 Kbps to 2

Mbps, making it 12 days to transmit one day's worth of oilfield data to an onshore Data Center [19]. As such, the major challenges for the remote offshore oil fields can be specified as follows:

- Real-time processing of resource-intensive emergency applications.
- Constant decision-making during the extraction process.
- Real-time monitoring of the site.

To address these challenges, there is a need for a system to collect data from oil fields, process them instantly, and make necessary decisions for a seamless and reliable O&G site. Despite recent technological advancements, to date, no comprehensive solution exists that can support bandwidth and computationally intensive operations expected in smart oil fields. Provided the difficulties in achieving a real-time response required for time-sensitive applications in remote smart oil fields, enabling smartness for remote oil fields remains as an open challenge. Edge computing systems, if deployed cleverly, has the potential to obviate these difficulties in remote oil fields and enable them to take advantage of services offered by smart oil fields.

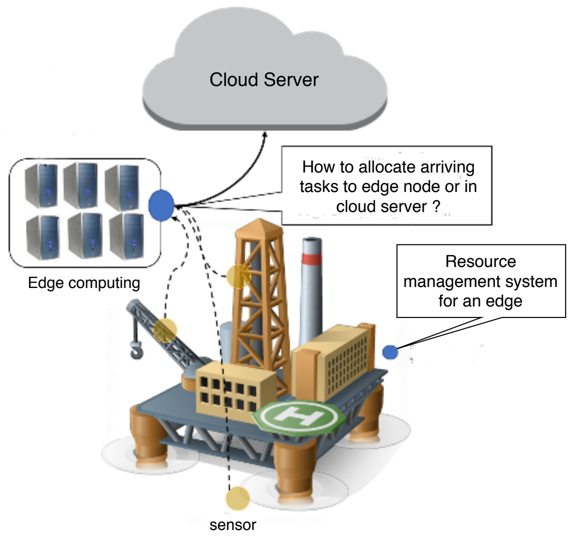
Cloud Server
How to allocate arriving tasks to edge node or in cloud server ?
Edge computing
Resource management system for an edge
sensor

Fig. 1. A smart oil field equipped with sensors and the edge node .

As depicted in Figure 1, our approach to the problems of smart oil fields in remote areas is to use an Edge Computing system that is aware of the Quality of Service (QoS) demands of different applications types. It accounts for performance characteristics of the computational resource and their limited availability in the edge environment. It also considers low and unreliable connectivity to the onshore Data Centers. The question is how to efficiently process resource-intensive and time-sensitive applications in the presence of a weak and unreliable connection to onshore Data Centers? Efficiency here refers to solutions that are aware of the connectivity, limited computational capacity, and resource intensiveness of emergency management applications in remote oil fields. Therefore, the specific problem definition in this research can be stated as *how to allocate arriving tasks to an Edge resources or in cloud server in a robust manner (i.e., in a manner that the number of tasks missing their individual deadline is minimized)?* The contributions of this paper are as follows:

- Proposing a resource allocation model to efficiently use limited computational resources of Edge resources and minimizes reliance on onshore resources.
- Developing a coordinator heuristic that provides robust task processing for real-time monitoring and decision making.
- We analyze the performance of our proposed heuristic under various workload conditions.

The rest of the paper is organized as follows. Section 2 introduces the system model with formulation, assumption and system model scenario. Section 3 discusses allocation approach. Sections 4 and 5 present heuristics and performance evaluation where experimental setup and experiments with the results are particularly described. Section 6 presents the related work. Finally, section 7 concludes the paper.

II. System Model

*1) **Formulation**:* In system model we propose, a set of tasks is generated from the sensor raw data and sent to a Processing Unit (Edge or Cloud). Every task has its own deadline within which it has to be completed. Allocation algorithm aims to maximize the number of tasks meeting their deadline. According to the problem definition, the set of arriving tasks can be defined as "T", where T = $\{t_1, t_2, t_3, t_4 \ldots, t_n\}$ and the set of Processing Units (PUs) "S", where S = $\{s_1, s_2, s_3, s_4 \ldots, s_m\}$. The set of tasks that meet their deadline can be denoted as T_s, which is the subset of T ($T_s \subseteq$ T). It is assumed that a task t_i is allocated to the PU s_j when the task t_i can meet its deadline δ_i in that specific PU. The problem can be formulated as:

$$\begin{aligned} \min_{t} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij}, where\ t_{ij} \in \ \mathrm{T}_s \\ \text{subject to} \quad & t_i \leq \delta_i \ , where\ t_{ij} \in \ \{0,1\} \end{aligned} \tag{1}$$

In the above equation t_{ij} is an indicator random variable that shows whether or not a task is allocated to a PU. Successful allocation is represented by 1 and by 0 otherwise. Therefore the above problem is Binary Integer Linear Program (BILP).

*2) **Assumptions**:* In our system model, the Edge Node is a stationary device with memory storage, limited computational capacity and wireless communication capability [6]. The Edge Node is located near the oil rig on the platform above the water surface. The Edge Node is capable of processing tasks in a real-time manner. Therefore, computationally intensive tasks are not suitable for the Edge Node. Nevertheless, Cloud Server has a massive processing power which is suitable for intensive computations. Different sensors (*e.g.,* pipeline pressure, cathodic protection, flow monitoring, air pollution, tank levels, gas density) produce different data types (*e.g.,* numbers, texts, images, videos) which are utilized by different applications (disaster management, remote monitoring system, etc.) [7]. According to the variety of applications, we categorize arriving tasks by a task type. A task type represents different computational requests that are submitted by different applications. For instance, pipeline pressure sensor data, small video clips or images for remote monitoring center, drilling strategy, routine report generation, video streaming, temperature sensors data. Depending on the nature of different task types, some tasks are urgent or delay sensitive, and some are not urgent or delay tolerant. For instance, drilling strategy, routine report generation, video streaming, are delay tolerant tasks whereas pipeline pressure monitoring, oil spill monitoring, temperature monitoring tasks are delay sensitive.

Upon the arrival of the task to a PU, it is assigned an individual deadline. An individual deadline includes the task arrival time and the end-to-end delay the task can tolerate. In a real-time scenario, a disaster management application can submit requests that require high data rate (*e.g.,* thermal-maps, live footage of pressure valve). On the other hand, the number of applications can be large and each application may request several tasks simultaneously which may strain the capacity of the wireless network. For above-mentioned reasons, communication delay (uplink and downlink delay) can have a significant impact on the overall delay. Thus, we also consider a communication delay for a deadline calculation. For arriving task t_i, deadline δ_i can be defined as: $\delta_\mathrm{i} = arr_\mathrm{i} + E_\mathrm{i} + \varepsilon + \beta$, where arr_i is the arrival time of the task, E_i is the average task completion time, ε is a constant value defined by the processing device (slack time) and β is the communication delay.

*3) **System Model Scenario:*** Upon the arrival to the *computing unit* the task gets into a coordinator which works in an immediate mode to allocate arriving tasks to the *computing units*. It allocates the task to the Edge Node or to the distant Cloud Server. Therefore, an arriving task gets immediately allocated. Then, it enters the batch queue of the *computing unit* for processing. The time passed from the arrival of the task until its processing completion can be defined as a "computational delay" (d_C). Therefore d_P can be defined as: $d_\mathrm{P} = d_C$, where d_C represents an average computational delay.

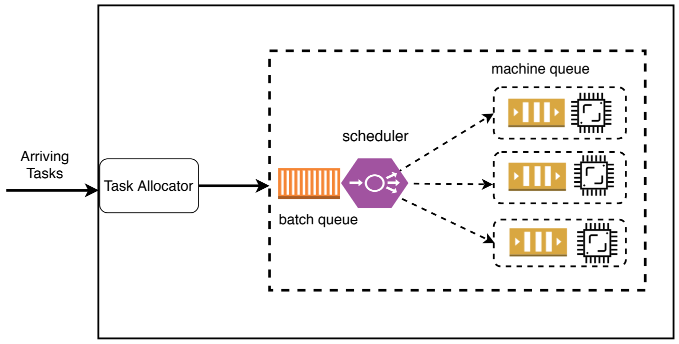

Fig. 2. Architecture of the Edge Computing unit in smart oil field.

III. Approach

According to the system model, tasks arrive at the PU randomly and the arrival rate is not known in advance. Receiving *computing unit* is considered to be oversubscribed, which means that it receives the number of tasks beyond its capacity to execute within a deadline. Therefore, some tasks are projected to miss their deadline. If such tasks are delay-sensitive, then they are dropped. Dropping tasks is a usual practice in oversubscribed real-time systems [9]–[11]. In such systems, the execution of a task that is going to miss its deadline has no value.

Here, we define the task robustness as the certainty of the task to meet its deadline in a particular *computing unit* (Edge or Cloud). For the arriving task of type i in a *computing unit* j, the robustness can be defined as:

$$C_j(t_i) = \delta_i - t_i^c \tag{2}$$

where δ_i is the deadline of the task type and t_i^c is the completion time of the task type i. Therefore, if the arriving task has a greater certainty to meet its deadline in the *computing unit* (Edge Node) A ($C_A(t_i)$) than in a *computing unit* (Cloud Server) B, then the Edge Node A provides a greater robustness for this task.

*1) **Delay Estimation:*** The calculation of the end-to-end delay of the task during the processing in a smart oil field can be defined as:

$$D_s = d_C + d_P \tag{3}$$

where d_C is a communication delay and d_P is a processing delay. For the Edge Node a communication delay can be further broken down into the average uplink (d_U) delay and the average downlink (d_D) delay.

For the task t_i requested by an application i from the Edge Node m, the uplink delay is

$$d_\mathrm{U} = \frac{L_\mathrm{i}}{R_i^m} \tag{4}$$

and the downlink delay is

$$d_\mathrm{D} = \frac{L_\mathrm{i}}{R_m^i} \tag{5}$$

where L_i is the task packet size, R_i^m and R_m^i is the transmission data rate for the link from i to m (uplink bandwidth) and from m to i (downlink bandwidth) respectively.

For the Cloud Server, we need to consider a propagation delay (d_R) when calculating a communication time. Satellite is used for transferring the task to the Cloud Server. Therefore, in this scenario, a propagation delay can be defined as:

$$d_R = \frac{Distance}{Speed} \cdot 2 \tag{6}$$

In the above equation, we consider a round trip time which is twice the delay. Therefore, communication delay for the Cloud Server consists of a transmission delay along with a propagation delay. As such, we define communication delay for the Cloud Server as:

$$d_C = \frac{L_i}{R_i^m} + \frac{Distance}{Speed} \cdot 2 + \frac{L_i}{R_m^i} \tag{7}$$

Different applications request different jobs through the uplink channel from the *computing unit* . Upon request, the task first goes through the coordinator. The coordinator is the component that allocates arriving tasks to the appropriate *computing unit*s. It makes the decision of task allocation, considering that its robustness must be maximized. As a reflection of this decision, the task is then transferred to the *computing unit* that offers the highest robustness. The coordinator uses the average task completion time extracted from the historical data to calculate robustness of the task in the Edge Node. In case of the Cloud Server, the coordinator also incorporates the task transfer time(transmission delay & propagation delay) for determining the robustness of task. This process also takes historical data into account for calculating the transfer time.

When coordinator gets the task t_i of task type i, it calculates the deadline (δ_i^j) of this task for the *computing unit*s (Edge Node & Cloud). Then it calculates the certainty ($C(t_i)$) of the task by deducting the completion time of that particular task from its deadline. The acquired difference indicates the time remaining before the deadline occurs. The higher the difference, the more chances the task has to meet its deadline. Therefore, coordinator checks the certainty of an arriving task both for the Edge Node and the Cloud Server. After all, the task is assigned to the *computing unit* (Edge or Cloud) that provides the highest certainty.

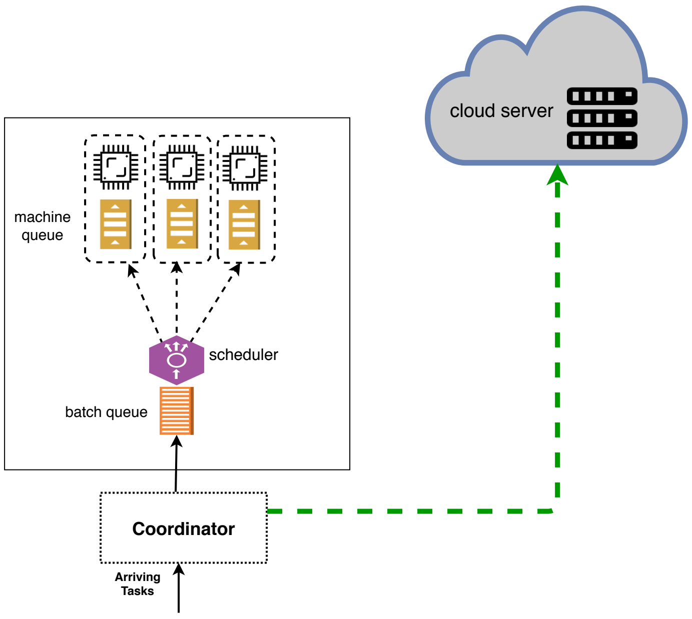

Fig. 3. A proposed model where the coordinator efficiently allocates arriving tasks

IV. Heuristics

A. *Coordinator Heuristics*

The coordinator is responsible for making an efficient task allocation decision so that the task can be processed within its individual deadline. The coordinator works in immediate mode. Therefore, upon the arrival, every task gets immediately allocated. A buffer is present in every coordinator in case many tasks arrive at the same time. Thus, the load balancer can store some tasks to serve them in the future. There is no re-allocation considered after the task has been allocated by the coordinator due to the overhead produced by additional task transfer.

1) ***Baseline:*** This is the baseline heuristic which means it considers there is no coordinator and no Edge Node. It allocates every task to the Cloud Server for processing in a conventional way (without considering any delay). Therefore, this heuristic allocates every arriving task to the Cloud for processing.

2) ***Maximum Certainty (MC):*** This heuristic aims to maximize the certainty of the task. When the arriving task enters the coordinator, the certainty of that task for the Edge and the Cloud is calculated. For the allocation, coordinator picks the *computing unit* (Edge or Cloud) that gives the highest task certainty.

3) ***Task Type (TT):*** This heuristic uses the task type. In our system model, we have 3 different task types. From the arrived tasks heuristic selects the tasks which have the shortest deadlines (task type 1). After, it allocates them to the Edge Node due to the task type's real-time nature. Other tasks that have a less strict deadline are sent to the Cloud for processing. Therefore, this heuristic leverages the short real-time tasks to be processed in the Edge Node.

B. *Scheduler Heuristics*

After the task allocation is performed by a coordinator, the scheduler of every Edge Node allocates the task to the VMs. In our research, we use conventional scheduling policies. These scheduling policies are used along with the coordinator heuristics and provided as follows:

1) First Come First Serve (FCFS): FCFS is one of the popular baselines for the task scheduling policy. According to this heuristic, the tasks that arrive earlier get scheduled first. In other words, the task that arrives first stays in the head of the queue and the tasks that arrive later stay in the tail. The scheduler allocates tasks from the head of the queue to VM's local queue for execution. Scheduling event occurs whenever a free spot appears in VM's local queue.

2) Shortest Job First (SJF): SJF leverages the tasks with the shortest execution time. At first, the heuristic arranges the tasks in the batch queue according to their execution time in ascending order. Therefore, the tasks with a shorter execution time stay in the head of the queue and are scheduled immediately whenever a free spot appears in VM's local queue.

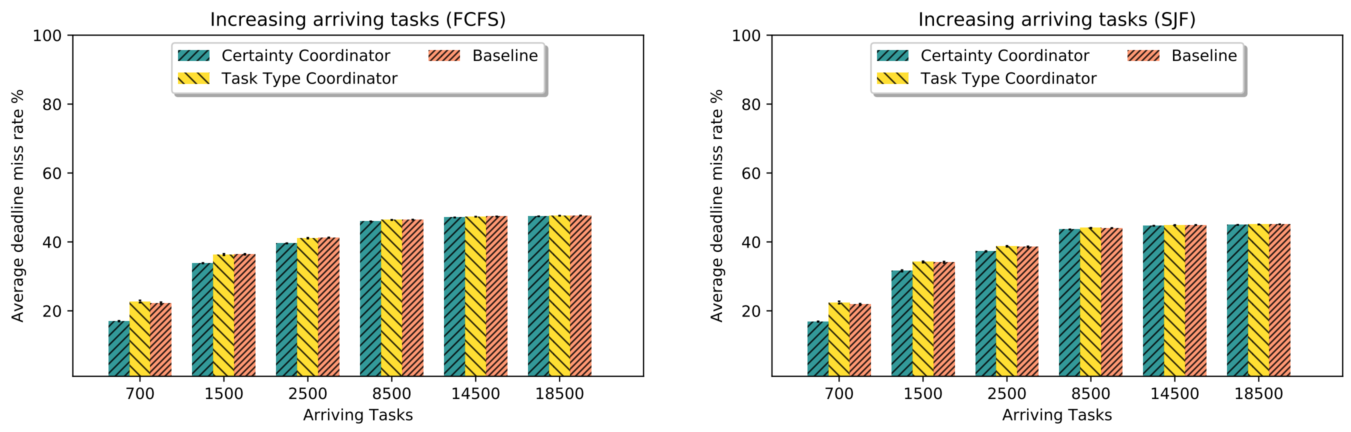

Fig. 4. Deadline missing rate is measured using 3 heuristics. Two different scheduling policies (FCFS & SJF) is used for evaluation.

V. Performance Evaluation

A. Experimental Setup

To evaluate the performance of our system we use a CloudSim simulation [3]. Cloudsim is a discrete event simulator that provides Cloud and Edge Computing models. In our simulation Edge Nodes are Data Center like devices with limited computational capacity (8 cores). Each core is utilized by one VM. All VMs in the Edge Node are homogeneous, i.e. have the same computational power (MIPS). On the other hand, a cloud server is a simulated Data Center with larger computational power (16 cores). We also implement the bandwidth for the Cloud Server and simulate the propagation delay to calculate the communication latency in our scenario. All these parameters are scalable. In this work, we implement a small scale simulation which can be scaled up in future work.

In our simulation, we implement tasks as ”cloudlets” with additional parameters added to the CloudSim's default configuration. We create 1 edge node and a coordinator component which allocates the tasks according to our proposed heuristic.

B. **Workloads**

In our simulation, we randomly generate execution times using Gaussian distribution [12]. We use results of the Extreme Scale System Center (ESSC) at Oak Ridge National Laboratory (ORNL) [9], [11] to implement the arrival time for each task. Nevertheless, the arrival time from this workbench is very sparse which is not applicable for our work. For this reason we generate dense arrival time by increasing the number of tasks (24000) compare to the workbench. Initially, every *computing unit* (Edge and Cloud) is assigned an individual workload to create an oversubscription situation and to obtain historic estimated task completion times. Together with an individual workload, our receiving coordinator is assigned the major workload. This major workload is the one we consider for our results evaluation. The workloads are seeded and changed from trial to trial. By manipulating the number of tasks in the initial workloads, we are able to control the system's oversubscription level.

C. Experiments

To study the performance of our model thoroughly, we evaluate the system under various amounts of arriving tasks. We analyze the impact of the increasing workload on the oversubscribed system. Initially, each *computing unit* in the system has it's own workload and is proved to be oversubscribed (deadline miss rate of tasks is more than 20%). Additionally, we assign a testing workload to the Edge Node which uses a coordinator to allocate them. The individual workload for each *computing unit*is independent of the testing workload, as we assume the testing workload to be extra request bursts that can occur during the emergency situations. We consider the minimum testing workload to be a 700 tasks and the maximum to be a 18000 tasks. In each step, we increase the workload by a certain amount of tasks and account for the number of tasks that miss their deadline. For each step, we perform 10 trials and calculate the average number of tasks that miss the deadline. We compare the average number of tasks that miss their deadline while using certainty coordinator heuristic, task type heuristics and baseline heuristic.

Figure 4 demonstrates the results obtained from the described experiments. The first image of figure 4 represents the results obtained using the FCFS scheduling algorithm. As we can see, the coordinator performs better using the certainty heuristic rather than baseline and task type heuristics. As certainty heuristic accounts for the deadline and tends to support real-time tasks even a slight improvement in the miss rate is good considering the lack of computational power in the edge.

Only 17.01% of all tasks of the whole system workload miss their deadlines using the certainty heuristic, where more than 22.20% of the testing tasks miss their deadline with the baseline heuristic. Although the deadline miss rate grows with the increase of the workload for all three heuristics, the certainty heuristics performs better from the beginning to the midrange of the workload increase. From the midrange to the largest workload, certainty heuristic performs nearly the same as the other two heuristics.

We assume that the performance of the certainty heuristic depends on the level of the initial oversubscription of the system, but we leave finding the proof for our assumption for the future research. We can also notice that MR heuristic performs slightly better using the SJF scheduling rather than FCFS. The reason for that can be the efficient scheduling of short tasks first. Overall, we can observe that certainty heuristic performs better than task type and baseline heuristics, which can provide a better robustness for the whole system in the situations of oversubscription.

VI. Related work

Edge Computing concepts have been previously proposed in the literature for delay tolerant networks. Lorenzo *et al.,* in [13] proposed resource allocation methods for Edge Computing environments that consider unreliable network connectivity. However, in similar kind of research works [8], [24], [25] authors neither consider the case of emergency management applications nor the heterogeneity of the Edge resources, while performing resource allocation.

The more specific problem of resource provisioning for real-time disaster management applications in Edge Computing with low-connectivity to the back-end Data Centers has not been explored in the context of remote smart oil fields and there is a limited research on these issues in other contexts. Efforts towards smart oil fields have been predominantly on analyzing the big data extracted from oil wells by Cameron *et al.,* in [4] or applying machine learning methods to reduce exploration or drilling costs by Parapuram *et al.,* in [15]. Warning systems for early prediction of disasters were analyzed by Xu *et al.,* in [26]. These solutions are all reliant on onshore Data Centers [7] which are not viable for remote and offshore oil fields.

In other domains, a common solution for low connectivity and allocation of resource-intensive applications is based on the Federation of Edge systems is proposed by Zhang *et al.,* in [27] which is not always an option in remote smart oil fields.

To date, limited work exists from academia and industry to design wireless communication networks for remote smart oil fields [17], [19]–[23]. Particularly, prior art cannot address the key challenges of remote operations, due to the following reasons. *First*, the works in [2] rely on satellite communications between the oil rigs and offshore management centers. However, satellite communication is not suitable for real-time decision making during oil extraction process, as the delay can be substantially large. *Second*, the works in [16] assume the existence of a macro cell BS at a nearby onshore location that provides wireless support for oil rigs. Nonetheless, remote reservoirs can be very far away from the shore. *Third*, most of existing networks [18] operate at sub-6 GHz frequency bands with limited capacity that cannot manage large data rates and URLLC requirements of smart oil fields. *Fourth*, ad-hoc communications protocol with random channel access cannot satisfy the ultra-reliability requirements in smart oil fields with a dense number of wireless devices. *Fifth*, existing works do not provide theoretical foundations for performance analysis and wireless resource management in smart oil rigs.

VII. Conclusions

In this paper, we presented a robust resource allocation model using Edge Computing that copes with the uncertainties exists both in communication and computation for offshore smart oil fields. We leveraged the task deadline and historical data for completion time to devise a coordinator that operates near the smart oil field (*i.e.,* at the Edge level) and distribute arriving tasks to the most certain *computing unit* (Edge or Cloud) in the face of oversubscription. The method considers the nature of the task type (*e.g.,* real-time and delay intolerant). Experimental results express that our proposed model can improve the robustness of the system compared to distant cloud data center working similarly in case of disaster or emergency-related tasks. From simulation results, it is observed that for various amounts of arriving task, our proposed heuristic outperforms the baseline and the Task Type heuristics. Approximately 17% of all tasks miss their deadline using the certainty-based coordinator, where more than 22.26% tasks miss their deadline with baseline and task type heuristics. In future, we plan to extend our proposed model to work more efficiently by considering heterogeneity within each *computing unit* (Edge and Cloud). We also plan to study the impact of approximate computing on the performance of the system.

Acknowledgments

We would like to thank the anonymous reviewers of the paper for their time and comments. This research is supported by the Louisiana Board of Regents under grant number LEQSF (2016-19)-RD-A-25.

References

[1] New realities in oil and gas: Data management and analytics. https://www.cisco.com/c/dam/en_us/solutions/industries/energy/docs/OilGasDigitalTransformationWhitePaper.pdf, accessed 2017.

[2] PM Bogaert, W Yang, HC Meijers, CM van Dongen, M Konopczynski, et al. Improving oil production using smart fields technology in the sf30 satellite oil development offshore malaysia. In *Proceedings of Conference on Offshore Technology*, May 2004.

[3] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience (SPE)*, 41(1):23–50, Aug. 2011.

[4] David Cameron et al. Big data in exploration and production: Silicon snake-oil, magic bullet, or useful tool? In *Proceedings of SPE Intelligent Energy Conference & Exhibition*, Apr 2014.

[5] Merv Fingas and Carl Brown. Review of oil spill remote sensing. *Marine Pollution Bulletin*, 83(1):9 – 23, 2014.

[6] Kai Liu and Victor CS Lee. Rsu-based real-time data access in dynamic vehicular networks. In *Proceedings of the 13th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1051–1056, Sept 2010.

[7] Wan Jun and Wang Hongyuan. Application of internet of things technology in digital oilfield. *Automation in Petro-Chemical Industry*, 1:001, 2015.

[8] Z. Hu, Y. Wei, X. Wang, and M. Song. Green relay station assisted cell zooming scheme for cellular networks. In *Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 2030–2035, Aug 2016.

[9] Bhavesh Khemka, Ryan Friese, Luis D Briceno, Howard Jay Siegel, Anthony A Maciejewski, Gregory A Koenig, Chris Groer, Gene Okonski, Marcia M Hilton, Rajendra Rambharos, et al. Utility functions and resource management in an oversubscribed heterogeneous computing environment. *Journal of Transactions on Computers*, 64(8):2394–2407, Aug 2015.

[10] Bhavesh Khemka, Ryan Friese, Sudeep Pasricha, Anthony A Maciejewski, Howard Jay Siegel, Gregory A Koenig, Sarah Powers, Marcia Hilton, Rajendra Rambharos, and Steve Poole. Utility driven dynamic resource management in an oversubscribed energy-constrained heterogeneous system. In *Proceedings of Parallel & Distributed Processing Symposium Workshops (IPDPSW)*, pages 58–67, May 2014.

[11] Bhavesh Khemka, Ryan Friese, Sudeep Pasricha, Anthony A. Maciejewski, Howard Jay Siegel, Gregory A. Koenig, Sarah Powers, Marcia Hilton, Rajendra Rambharos, and Steve Poole. Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system. *Journal of Sustainable Computing: Informatics and Systems*, 5:14 – 30, Mar 2015.

[12] Jun Li, Carlos Natalino, Dung Pham Van, Lena Wosinska, and Jiajia Chen. Resource management in fog-enhanced radio access network to support real-time vehicular services. In *Proceedings of 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 68–74, May 2017.

[13] Beatriz Lorenzo, Juan Garcia-Rois, Xuanheng Li, Javier Gonzalez-Castano, and Yuguang Fang. A robust dynamic edge network architecture for the internet-of-things. *arXiv preprint arXiv:1710.04861*, 2017.

[14] Sudhir Kumar Pandey, Ki-Hyun Kim, and Kea-Tiong Tang. A review of sensor-based methods for monitoring hydrogen sulfide. *TrAC Trends in Analytical Chemistry*, 32:87 – 99, 2012.

[15] George K Parapuram, Mehdi Mokhtari, Jalel Ben Hmida, et al. Prediction and analysis of geomechanical properties of the upper bakken shale utilizing artificial intelligence and data mining. In *Proceedings of Conference on SPE/AAPG/SEG Unconventional Resources Technology*, July 2017.

[16] Boselin Prabhu, E Gajendran, and N Balakumar. Smart oil field management using wireless communication techniques. 2017.

[17] S.R.B. Prabhu, E. Gajendran, and N. Balakumar. Smart oil field management using wireless communication techniques. *International Journal of Inventions in Engineering & Science Technology*, (2):2454–9584, Jan 2016.

[18] Mohammad reza Akhondi, Alex Talevski, Simon Carlsen, and Stig Petersen. Applications of wireless sensor networks in the oil, gas and resources industries. In *Proceedings of 24th International Conference on Advanced Information Networking and Applications (AINA)*, pages 941–948, June 2010.

[19] Cisco. A New Reality for Oil & Gas: Data Management and Analytics. White paper, April 2015.

[20] Gartner. Hype cycle for upstream oil and gas technologies. White paper, July 2014.

[21] Gartner. Hype cycle for upstream oil and gas technologies, 2014. White paper, July 2014.

[22] Gartner. Top 10 technology trends impacting the oil and gas industry in 2015. White paper, March 2015.

[23] WoodMackenzie. Why are some deepwater plays still attractive? White paper, September 2017.

[24] Y. Wang, X. Lin, and M. Pedram. A nested two stage game-based optimization framework in mobile cloud computing system. In *Proceedings of the 7th IEEE International Symposium on Service-Oriented System Engineering*, pages 494–502, March 2013.

[25] S. Wunderlich, J. A. Cabrera, F. H. P. Fitzek, and M. Reisslein. Network coding in heterogeneous multicore iot nodes with dag scheduling of parallel matrix block operations. *IEEE Internet of Things Journal*, 4(4):917–933, Aug 2017.

[26] B. Xu, W. Wang, Y. Wu, Y. Shi, and C. Lu. Internet of things and big data analytics for smart oil field malfunction diagnosis. In *Proceedings of the 2nd International Conference on Big Data Analysis*, pages 178–181, Mar. 2017.

[27] Zehua Zhang and Xuejie Zhang. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In *Proceedings of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA)*, volume 2, pages 240–243, May 2010.