2017 IEEE 5th International Conference on Future Internet of Things and Cloud

Edge-supported Approximate Analysis for Long Running Computations

Ali Reza Zamani, Javier Diaz-Montes, Manish Parashar
Rutgers Discovery Informatics Institute
Rutgers University, USA
alireza.zamani@rutgers.edu

Ioan Petri
School of Engineering
Cardiff University, UK
petrii@cardiff.ac.uk

Omer Rana
School of Computer Science & Informatics
Cardiff University, UK
ranaof@cardiff.ac.uk

***Abstract*—With the increasing availability of Internet of Things (IoT) devices, and potential applications that make use of data from such devices, there is a need to better identify appropriate data processing techniques that can be applied to this data. The computational complexity of these applications, and the complexity of the requirements on the data processing techniques, often derives from the capabilities of current IoT devices and the need to integrate data streams across multiple IoT devices, which result in larger data sizes and loads on the computing infrastructure. Furthermore, due to the dynamics and uncertainties of edge environments, it is essential that these techniques are capable of adapting across a range of computational and data transfer requirements (such as execution performance) and infrastructure scales (processing nodes, storage needs, network requirements) to carry out a particular analysis task, in response to changing requirements and constraints. Approximate computing offers techniques that can simplify the overall analysis workflow, trading off loss in quality and optimality of the solution with time to reach a particular outcome. These techniques have two main advantages: (i) reduced time to execute a particular data analysis; (ii) reduced requirements on the computational infrastructure (i.e., lower energy, computational resource needs, etc) to carry out such analysis. With data processing capabilities available IoT devices and associated gateway nodes, such approximate computing can be achieved at or close to the network edge. In this paper, we propose in-transit and edge-supported approximation techniques, which can undertake partial/approximate data processing at the data generation/capture or aggregation site, prior to delivery to a cloud data center. We also demonstrate how such an approach can be used in practice by applying it to support energy optimization in built environments (utilizing a combination of sensors and cloud-based data analysis). Several approximation techniques that are relevant in this context are presented, and their relevance explored and evaluated in the context of an energy simulation application scenario.**

***Keywords*-Approximation Techniques; Internet of Things; Smart Built Environments; Distributed Cloud Computing**

I. Introduction

With the increasing availability of IoT and pervasive devices, and the need to connect these devices to cloud-based systems in order to support data analysis, there is a realization that a computational infrastructure closer to these devices is required (in addition to a data center). A number of requirements have been identified that support this need: (i) the network latency to transfer data from these devices to a cloud data centre can be excessive, especially if the IoT device is connected over a network with a varying availability and bandwidth profile; (ii) data generated from the IoT device may have security/privacy constraints, which limits how much of this data can be transferred to a cloud data center; (iii) such devices may be constantly generating data, however only a small subset of the data may be *useful* or require further analysis. Transferring all of this data to a cloud data center, where, for example, a significant part of it is subsequently deleted, is inefficient and does not make effective use of network and storage resources. Understanding when and how much data should be analyzed close to where it is captured, remains an important research challenge – an aspect addressed in this paper. Specifically, this paper focusses on how "approximation" techniques can effectively used to process data close to the data sources with reduced computational resource requirements.

There is a significant class of data intensive applications (especially those involving data generated from IoT devices) that have associated analysis deadlines with direct relationships to a *quality of results* metric. The incoming workloads for these applications can include a number of sources, such as various types of sensors (i.e., potentially inexact inputs), and the associated data analysis algorithms are often stochastic in nature (e.g., iterative algorithms). For example, in the building energy optimization scenario being considered in this work, simulations are iterative and have different numerical configuration parameters that directly influence the duration of simulation, as well as the quality of results. Therefore, it becomes important to develop a suitable computational infrastructure wherein both hardware and software are continually optimized, in-line with application specific objectives.

Approximate computing provides a useful mechanism for supporting the requirements identified above, and has attracted significant interest from both academia and industry. It enables techniques for creating robust and resilient applications by proposing the introduction of intervals of acceptable errors into the execution to address both computing and application uncertainties. Approximate computing views regions in the application's execution in terms of the degree of error tolerance, and uses this tolerance to trade off between storage, result accuracy, and efficient usage of computing resources (i.e., energy, storage size, etc.). Thus, approximate computing provides a balance between the level of accuracy required by the user and that provided by the computing system, to achieve a spectrum of optimizations. Such techniques are applicable to a wide range of applications/frameworks, for example, data analytics, scientific computing, multimedia and signal processing, machine learning and MapReduce [6, 12].

In previous works [14, 20], we developed an analytical model to improve the use of computational resources at the network edge and within network data centers to support data transformation and analysis from source to destination. The objective of those works was to combine efficient use of resources within a cloud data center and those at the network edge, while providing an extra source of revenue for those who operate and manage such resources. In this paper, we extend this approach by investigating how network edge resources can be used to support a data analysis workflow deployed leveraging approximation techniques. We seek to find ways to optimize workflow execution over resources that are located at the network edge and to determine which approximation techniques are most beneficial given resource constraints at the edge. A key assumption in this work is that the core data center are far away from data source and have a greater capability (capacity, function) compared to those at the network edge.

In this paper, we investigate two uses of approximate computing:(i) The ability to reduce computation using approximate computing techniques. (ii) The ability to execute parts of the workload or increase the accuracy of approximation techniques using resources at in-transit nodes or at the edge of the network.

Our application workflow performs real-time energy optimization within a building, and makes use of multiple approximation techniques as part of the simulation(s). In this work, we determine which of these techniques can be executed in-transit or at the edge of the network in the presence of multiple constraints such as execution time and quality of results. The rest of the paper is organized as follows. In Section II we present related work in approximate computing. In Section III we explain the application use-case as identified in building optimization, followed by the associated approximation techniques presented in Section IV. The methodology is presented in Section V. We present our evaluation in Section VI and conclude in Section VII.

II. Aproximation techniques

In this section, we summarize several approximate computing techniques that can be applied at the network edge.

Memory access skipping, task dropping/skipping, Memoization: Samadi et al. [7] propose a pattern-based approximation technique to reduce number of memory accesses by skipping tasks in a loop, apply memoization to optimize map and scatter/gather patterns and cache result of computationally expensive function calls, to reduce computational overhead. Samadi et al. [8] propose SAGE, a self-tuning approximation for graphics engines which uses lossy compression (data packing) to reduce access to memory. Goiri et al. [4] present an ApproxHadoop module which can apply approximations to MapReduce frameworks using data sampling and task dropping.

Using multiple inexact program versions/lossy compression: Vassiliadis et al. [10] propose a programming model and runtime system to improve energy efficiency of the programs. The programming model enables developers to identify the impact of different sections of their program on the final output.

Neural network-based accelerators: McAffee and Olukotun [5] developed EMEURO which is a neural-network emulation and acceleration platform. With small approximation error rate, EMEURO can achieve considerable speedup in various applications within the image processing domain. Amant et al. [9] developed a general-purpose code acceleration and end-to-end solution to utilize analog circuits in order to accelerate approximate applications and neural network training phase.

Approximating neural networks: Venkataramani et al. [11] have explored the use of approximate computing to design a new energy efficient hardware implementation for large scale neuromorphic systems(AxNN). Zhang et al. [18] propose an approximate computing framework for ANN. It is based on approximating neurons which are less critical.

Precision Scaling: Anam et al. [1] explore the trade-off in precision for energy and throughput in a generic matrix multiplication and one dimensional convolution. Yeh et al. [13] apply dynamic precision tuning in floating point computation. This techniques can increase the performance and decrease energy consumption in physics-based animation.

Loop Perforation: Baek and Chilimbi [2] develop a framework for supporting energy-conscious programming using controlled approximation formed of 2 phases – "calibration" phase, which involved building a model for Quality of Service (QoS) loss, and an "operational" phase which involves directly applying the approximation decisions.

III. Energyplus Use Case

An instrumented built environment, which can consist of single/multiple buildings (homes, office buildings, sports facilities, etc), provides a useful scenario to validate the use of edge-supported approximation. Depending on the number of sensors within a single building, the frequency at which data is captured from such sensors and the particular data analysis objective (e.g. reduce energy consumption, improve efficiency of HVAC (heating, ventilation and air condition) function, improve comfort levels based on occupancy, etc), the computational capability requirements can vary significantly. In some instances such data is often analyzed offline (in batch mode) to enable improvements in building design or to support long term facilities management. In other instances (evidenced by recent use of such instrumented environments), real time analysis needs to be carried out (over intervals of 15 to 30 minutes generally) to enable better energy efficiency and use of such infrastructure. When multiple such buildings are considered (e.g. within a business park, University campus or a housing association), the overall computational requirement can increase considerably.

To provide practical real time decision making in building energy management based on real time monitored data, it is necessary to develop a 'behaviour' of a building energy system by using various simulation tools. During the process, domain experts are often involved in order to identify the main use cases and scenarios with associated input parameters and feasible outputs. In the modelling process, a number of components have to be assessed and calibrated iteratively, and the developed building energy simulation model is then executed (as the calculation engine) within a generic optimization program. In this work we seek to identify approximation techniques that can complement or replace the execution of multiple EnergyPlus[1] instances, a software that requires significant computational resources to run, with different input parameter ranges.

Various types of sensors are used to monitor energy efficiency levels within a building, such as: (i) solid-state meters for accurate usage levels, (ii) environmental sensors for measuring temperature, relative humidity (RH), carbon monoxide (CO), and carbon dioxide (CO_2) levels, (iii) temperature measurements using both mechanical (e.g., thermally expanding metallic coils) and electrical means (e.g., thermistors, metallic resistance temperature detectors (RTD), thermocouples, digital P-n junctions, infrared thermocouples) to provide sufficient accuracy. When dealing with large buildings such as sports facilities, the accuracy of these sensors is often questionable, largely because of the significant drift that occurs after initial calibration. In some buildings, there are specific requirements for sensors when monitoring CO_2 concentration, air flow, humidity, etc and these sensors are more expensive to use and deploy. We use sensor data from the $SportE^2$ project pilot called FIDIA, a public sports building facility, located in Rome, Italy. $SportE^2$ is a research project co-financed by the European Commission FP7 programme under the domain of Information Communication Technologies and Energy Efficient Buildings. This project focuses on developing energy efficient products and services dedicated to needs and unique characteristics of sporting facilities.

[1]http://apps1.eere.energy.gov/buildings/energyplus/

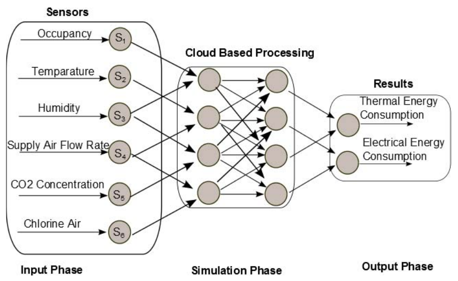

Figure 1: Energy optimization scenario. ©2016 IEEE. Reprinted with permission, Zou et al. [20].

IV. EnergyPlus approximation techniques

We identify the following approximation techniques applicable to EnergyPlus:

EnergyPlus loop reduction: this technique involves reducing the number of EnergyPlus instances by reducing the number loop iterations used within this simulation. As EnergyPlus execution needs to be carried out over a particular time frame, we can reduce the number of iterations/loop counter used, leading to a reduction in time over which EnergyPlus executions are carried out. This reduces the overall execution time while keeping the quality of results within a pre-defined error interval. The error rate is: $err = \frac{1}{times}$, where $times \geq 1$ is the number of times/loops to repeat the simulation.

Use of Artificial Neural Network: such a method involves the use of a learning algorithm for replacing the EnergyPlus simulation altogether. A neural network is trained based on historical (input/output) data obtained from previous executions of EnergyPlus simulations. This data is then used to train a neural network as a function approximator for the behaviour of an EnergyPlus simulation. The corresponding error rate is based on the size of historical data and on the efficiency of the neural learning algorithm being used. We set the error rate to 0.05, so that additional EnergyPlus simulations can be triggered if the error exceeds this threshold.

Parameter value skipping: based on a set of parameters that the simulation requires, this methods reduces the number of the parameter values which are used as input to the EnergyPlus simulation. The corresponding error rate of this method is based on the skipping interval. The associated error rate is $\frac{k}{100}$, where k is the number of parameter values skipped.

- Execution time without approximation techniques: $total.time = n * m * time$, where $time$ is the time of one EnergyPlus simulation
- Execution time with approximation techniques $total.approx.time = (n * m * total.time) - (k * total.time)$, n represents total number of parameters values, m is the number of parameters, and k represents the number of parameter values skipped;
- error rate $= \frac{k}{total.param}$, where $total.param$ is the total number of parameter values.

Parameter interval reduction: from the interval associated with a parameter we reduce the interval limits so the simulation would use only values from a predefined average value/centrality associated with a parameter interval. The error of this method depends on the remaining number of parameter values to use as input in the simulation. The error rate is: $=\frac{n+k}{100}$, where n is the number of the total parameters and k is the number of intervals being used.

- Execution time without approximation techniques: $total.time = n * m * time$;
- Execution time with approximation techniques $total.approx.time = (n * m * total.time) - ((n - k) * total.time)$, n represents total number of parameters values, m is the number of parameters, and k represents the number of parameter intervals reduced;
- error rate $=\frac{n+k}{total.params+total.intervals}$, where $total.params$ is total number of parameters and $total.intervals$ is total number of intervals

V. Methodology and Approximate In-transit Computational Model

To use network infrastructure more effectively, we propose hosting a data processing service at the network edge (on a gateway node connected to IoT devices) or within a network (making use of programmable network-based approaches, e.g. OpenFlow and other Software Defined Networks (SDN) approaches) to offer idle/available computational capabilities at the edge/in-transit data centers. Furthermore, this service is able to provision computational resources and allocate workload into such resources. Since the computational capabilities of the network data centers and edge clouds are limited, the main purpose of the resources at the edge and in-transit is to carry out small parts of the workload and increase the accuracy of the approximation techniques at the edge of the network by utilizing unused capacity within network data centers.

In general, two types of resources are considered in this paper. The main source of computation is computational data centers(sites or resource providers), which collectively form the cloud federation established using CometCloud [3]. The secondary resources are network data centers, that are located at the edge of the network. Lets assume a client needs to compute a job J, composed of k tasks, which is generated at the client's location defined as source s. Whenever the client decides to outsource the job to be executed at a remote site defined as destination d, the data would be exposed to possible edge and in-transit resources. A set of q network data centers $R : \{r_1, ..., rq\}$ has been considered as potential resources at the edge of the network. Hence, it is essential to identify the workload placement, the best route from source to destination, and possible edge/in-transit resources that can contribute to the execution of the job. The service level agreement (SLA) of a job J includes: deadline ($Deadline(J)$) by which results have to be returned to the client and a budget ($Budget(J)$) that sets the maximum amount available to spend on computing job J.

To ensure the control over the network, all of the sites and network data centers are equipped with SDN enabled routers. We consider that there is some waiting time $W(J)$ before a job J can be executed at destination site d. During this time, the job is idle and it occupies storage space at the destination site. Hence, we would like to identify and configure a data path that leverages edge/in-transit computation to take advantage of $W(J)$ for a job. The following variables have been considered in our formulations:

- $P(r_i)$ is the average number of tasks that resource r_i completes per unit of time.
- $E(r_i)$ is the amount of time spent computing in resource r_i.
- $CE(r_i)$ is the cost per unit of time of using resource r_i for computation.
- $T(r_i, r_k)$ is the amount of time spent transferring data between resources r_i and r_k.
- $CT(r_i, r_k)$ is the cost of reserving a network channel per unit of time.
- $W(J)$ is the waiting time before job J can start its computation at destination resource.

The objective of our problem is to maximize the number of tasks completed at edge/in-transit resources:

$$\max \sum_i P(r_i) * E(r_i)$$

subject to being ready to be computed at destination resource d at the scheduled time (1) and making sure that the all of tasks within a job is executed completely(2), within the given deadline(3) and budget(4) :

$$\sum_i E(r_i) + Transfer(J) \le W(J), \tag{1}$$

$$\sum_i [P(r_i) * E(r_i)] + P(d) * E(d) = k, \tag{2}$$

$$\sum_i E(r_i) + Transfer(J) + E(d) \le Deadline, \tag{3}$$

$$Cost(J) \le Budget, \tag{4}$$

where $Transfer(J)$ is the overall transfer time of a job, defined as the sum of the time spent transferring data from source (s) to first network data center (r_i), the sum of the time spent transferring data between network data centers $\in R$, and the time spent transferring data between the last network data center (r_k) and destination (d):

$$Transfer(J) = T(s, r_i) + \sum_i^q \sum_{k \neq i,k}^q T(r_i, r_k) + T(r_k, d).$$

$Cost(J)$ is the overall cost of computing job J, defined as:

$$Cost(J) = CostExecMid + CostExecDest + CostNet,$$

where the cost of computing in-transit ($CostExecMid$) and computing at the destination resource d ($CostExecDest$) are defined as:

$$CostExecMid = \sum_i [CE(r_i) * E(r_i)],$$

$$CostExecDest = CE(d) * E(d),$$

and the cost of transferring data associated with a job ($CostNet$) is defined as:

$$CostNet = T(s, r_i) * CT(s, r_i) + \sum_i^q \sum_{k \neq i,k}^q [T(r_i, r_k) * CT(r_i, r_k)] + T(r_k, d) * CT(r_k, d)$$

subject to $E(r_k) \neq 0$. Note that the time and cost of returning results to the client is negligible as only a few parameters are sent.

When resource providers cannot execute a job with the required SLA, the client can consider the use of approximation techniques. We consider that for an application *app*, we have a set of applicable approximation techniques $T = \{t_1, t_2, ..., t_m\}$, where each *app* has an associated required quality of solution and can be approximated with a subset of techniques $Q_k = \{t_1, t_2, ..., t_k\}$, $Q_k \subset T$, $m \ge k$, provided these techniques can satisfy a minimum accuracy threshold. Our two main objectives are to determine:

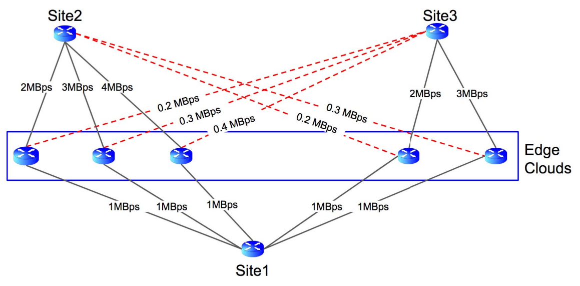

Figure 2: Infrastructure Setup. Solid and dashed lines indicate high and low bandwidth links, respectively.

1) set of approximation techniques T that are applicable to an application *app*
2) $t_j \in T$ that produces results within accuracy threshold, and techniques that can be executed on edge/in-transit resource(s)(i.e. create Q_K)

Given a set of suitable approximation techniques, $Q_k = \{t_1, t_2, ..., t_k\}$, each t_j has a corresponding error rate. Our strategy is to pick the approximation technique that has highest accuracy. If two approximation techniques have the same accuracy level, we pick the one with less computational resource requirement. If we pick an approximation technique, the edge/in-transit resource will use the waiting time ($W(J)$) to increase the accuracy of the chosen technique. For example, if a loop reduction approximation techniques is chosen, we try to increase the number of loops with resources at the edge of the network to increase the accuracy of the chosen approximation technique.

VI. Evaluation

In this section, we present the overall setup for our experimental infrastructure and several scenarios that we deployed to validate our hypothesis.

A. *Configuration of Testbed*

Our federation has been deployed on CloudLab [15] infrastructure platform. 8 VM instances are emulating geographically distributed environment to develop an evaluation testbed. Figure 2 shows the overview of the implemented infrastructure. 3VMs served as our resource providers' data centers: Site1, Site2 and Site3. Furthermore, we dedicated 5 more VMs as in-transit and edge resources(Edge Clouds) which are located at the edge of the network (i.e. between main sites). To emulate geographic distribution of the resources, we use Hierarchy Token Bucket (HTB) to configure various network bandwidth parameters. This network configuration is inspired in data obtained from previous experiments. Based on the computational capabilities at the edge/in-transit and resource provider sites, we considered three different infrastructure scenarios. The details of each

Table I: Infrastructure Scenarios.

Scenario	Edge Resources	Site Resources
Base	c4.2xlarge	c4.2xlarge
Higher	c4.2xlarge	c4.4xlarge
Highest	c4.2xlarge	c4.8xlarge

Table II: Resource Properties

Resource Type	vCPU	ECU	Memory	Price ($/Hour)
c4.2xlarge	8	31	15	0.464
c4.4xlarge	16	62	30	0.928
c4.8xlarge	36	132	60	1.856

scenario has been shown in table I. To model the mentioned scenarios, we used the characteristics of Amazon EC2 VM instances in our model. Summary of resource characteristics have been shown in table II.

To deploy the instances and create the network between them, we use Mininet [19]. In each VM, one Mininet host is connected to a virtual switch deployed by Open vSwicth [21]. These virtual switches in different instance are connected to each other using Generic Routing Encapsulation (GRE) tunneling [17]. Aside from the computational resources, a POX SDN controller [22] has been deployed to control the network infrastructure. In this work, the SDN controller has 2 main jobs: (i) installing the forwarding rules on switches. (ii) dedicating ports and connections to switches in order to receive necessary information from sites and optimizing the job execution. The controller establishes a dedicated connection between hosts and the controller. Here, we use UDP packets for communication between hosts and controller. Hence, the controller configures the switches to forward the UDP packet to the controller, unless those packets are generated from the controller. Specifically, when the source site, i.e. client, wants to talk to the controller, it should create a UDP message and send it to a specific address. The first switch receiving that message forwards the packet to the controller. Next, the controller sends UDP packets to all edge and in-transit sites asking for their status and computational capabilities. Since these UDP messages are from the controller, switches know where to send (i.e. in-transit resources) and deliver them flawlessly. Then, based on the feedback from the edge and in-transit resources, the controller selects the data transfer and execution pattern. This decision is sent to the client.

B. Experiments

For the use case, we considered that there are multiple smart buildings requesting to evaluate and optimize their energy consumption. Considering the FIDIA pilot [16], three types of jobs have been considered in this work. Table III collects the characteristics of different job types. The budget has been chosen high enough for

Table III: Job Information.

JobType	Data Size(MB)	Budget	Deadline(s)	Tasks†
JobType1	10	20	120	10
JobType2	20	30	150	20
JobType3	30	40	180	30

† – A job is composed of a set of tasks

jobs to eliminate imposed cost limitation on the system. As a result, the deadline is the only limiting factor in our experiments. We have considered 4 strategies:

- **Traditional Approach(T)**: In this strategy, only resource providers' sites can perform computation.
- **Edge Approach(E)**: Using this strategy, the edge data center and in-transit resources can contribute to the execution of the EnergyPlus jobs.
- **Approximation Approach(A)**: For this strategy, we considered that only resource providers' sites can execute jobs. However, if the system cannot meet the SLA requirement, using different approximation techniques will be considered.
- **Edge plus Approximation Approach(E-A)**: In this approach, the edge and in-transit resources can contribute to the computations and also increase the accuracy of approximation strategy.

For all of the scenarios mentioned in table I, we conducted the experiments based on the different strategies mentioned above. In each experiment, 326 jobs were inserted from Site1 to a federated marketplace, generated using a Poisson distribution. Once a job was inserted in the federation, different sites (i.e. Site1, Site2, and Site3) offered their services using a blind auction mechanism. For all of the jobs, if approximation techniques were available(E and E-A approaches), we established the minimum accuracy requirement of 95%. Consequently, if there was not enough resources to execute the job within the deadline, the system executed the job using one of the available approximation techniques. The approximation techniques selection has been discussed in section V. Moreover, for the ANN approximation method, we considered a training period that prepares the ANN method for desired accuracy. We considered 100 complete EnergyPlus executions for ANN training phase. In other words, if we could accept 100 EnergyPlus jobs(excluding other approximation methods), ANN model would be ready and system could use ANN method afterwards.
Table IV shows the required completion time for all types of EnergyPlus jobs in different infrastructures. The execution time for the different approximation techniques has been explained in section IV and is based on the parameters selected for approximation techniques, e.g. number of loops, amount of parameter value skipped, etc.

In order to compare different scenarios and evaluate

Table IV: Time to completion of EnergyPlus job types.

JobType	c4.2xlarge	c4.4xlarge	c4.8xlarge
JobType1	80 s	40 s	20 s
JobType2	100 s	50 s	25 s
JobType3	120 s	60 s	30 s

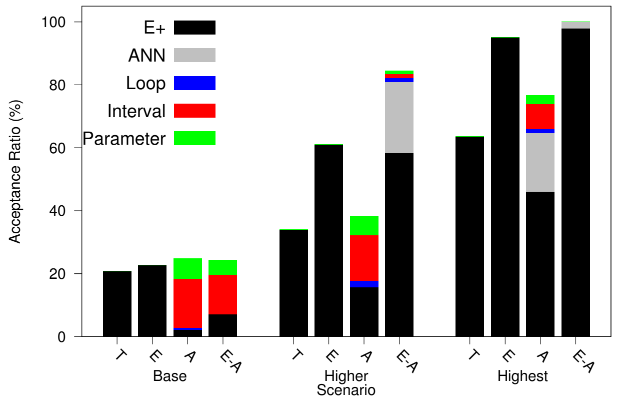

Figure 3: Job Acceptance Ratio. E+: EnergyPlus, ANN: Artificial Neural Network, Loop: Loop Reduction, Interval: Parameter Interval Reduction, Parameter: Parameter Value Skipping

the impact of edge clouds and approximation techniques, job acceptance ratio is demonstrated in Figure 3. The acceptance ratio is the ability of the resources to execute jobs completely within the deadline. Various approximation techniques help to reduce the potential resource requirement and overcome the limitations imposed by lack of resources. Looking at Figure 3, addition of edge resources and approximation techniques increase the job acceptance ratio. In base scenario, ANN method has not been used due to the small number of accepted jobs. Therefore, ANN method training has not reached the desired accuracy threshold. Moreover, in the base scenario, since the computing power of the site resources is limited, addition of the edge and approximation techniques slightly increase the acceptance ratio(2% to 5%). In the higher scenario, the best result has been reached in E-A approach, where sufficient E+ (EnergyPlus) simulations are executed to pass the threshold of accuracy of ANN, so ANN has a great impact on the job acceptance ratio. The edge strategy(E) has 22% more acceptance ratio compared to approximation(A) strategy, which shows the importance of the edge clouds in the case where the site computing resources are limited. In the highest scenario, E-A strategy has reached 100% acceptance ratio from which ANN covers around 2% of the jobs. For the approximation(A) strategy, In all scenarios, if ANN is not available, parameter interval reduction has the most contribution among all approximation techniques followed by parameter value skipping and loop reduction techniques.

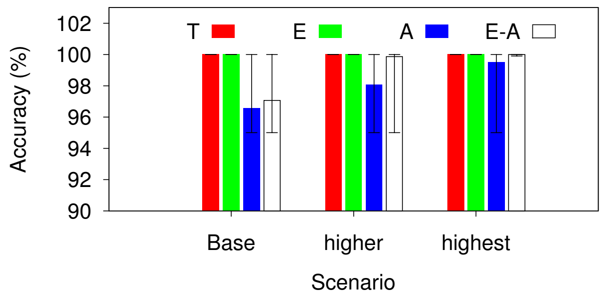

Figure 4: Average Accuracy

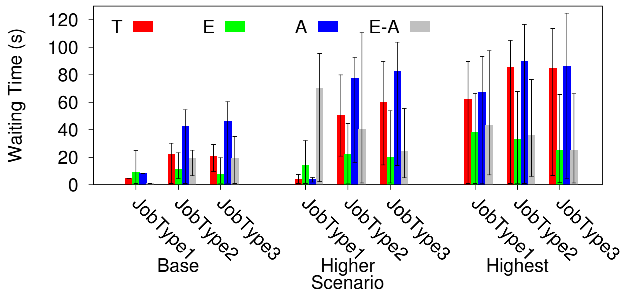

Figure 5: Idle time Overheads per job

Figure 4 compares the average accuracy achieved in different scenario and strategies. As we mentioned earlier, the minimum required accuracy in case of approximation is 95%. Since approximation is not available for traditional(T) and edge(E) strategies, the accuracy for those cases are 100%. However, for approximation(A) and edge-approximation techniques(E-A), the maximum accuracy among available techniques are selected. Moreover, addition of the edge to approximation strategy produces more accurate results. This behavior is due to the fact that the system uses edge clouds to execute part of the job or possibly increases the approximation accuracy. Considering both Figures 3 and 4, we conclude that approximation techniques have large impact on the acceptance ratio with slightly less accuracy. Addition of the edge resources results in more jobs to be executed with higher accuracy.

Figure 5 collects the information regarding the job waiting time (queue time). Having edge clouds can cause less waiting time because edge resources use waiting time to increase the accuracy of the approximation techniques or execute part of the jobs.

The average cost of the jobs has been shown in Figure 6. In general, approximation techniques(A and E-A strategies) result in cheaper job completion due to the reduction in job execution time. Specially, ANN techniques is significantly cost beneficial due to small execution time needed for ANN. However, we should note that approximation techniques are not as accurate as regular E+ execution.

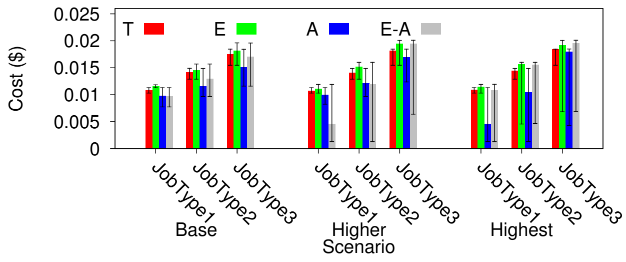

Figure 6: Average Cost

VII. Conclusions

In this paper, we explore the advantages of approximation techniques by presenting a real use-case scenario from the energy optimization domain. We demonstrate that long running computation jobs can be approximated using various techniques to reduce computation time without compromising the quality of results. This work identifies that approximation techniques, which have lower computational requirements, can be used directly closer to the data generation source to reduce latency of analysis, making more efficient use of available resources, improve acceptance ratio of tasks (i.e. enable a greater number of tasks to be completed within a given deadline), and provide a source of revenue for both owners of edge resources and network operators. Integrating approximation techniques with more conventional simulation can provide useful ways to improve utilization of our emerging computational infrastructure.

We have determined four different approximation techniques for EnergyPlus and investigated how these approximation techniques can be deployed at the edge the network and assessing their associated impact. Our results show that some approximation techniques cannot reach a desired accuracy threshold although completion time is improved. However when site resources are limited, a combination of the edge and approximation techniques help to increase the acceptance ratio.

Acknowledgements

This work is supported in part by NSF via grants numbers ACI 1640834, IIS 1546145 and ACI 1441376, and by US Department of Energy through the SIRIUS grant number DE-SC0015160. The research at Rutgers was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2). We are also grateful for Prof. Yacine Rezgui and his team at Cardiff University for collaboration on the energy simulation scenario reported here, as part of the European SportE2 project (http://www.sporte2.eu/).

References

[1] M. A. Anam, P. N. Whatmough, and Y. Andreopoulos. Precision-energy-throughput scaling of generic matrix multiplication and discrete convolution kernels via linear projections. In *Embedded Systems for Real-time Multimedia (ESTIMedia), 2013 IEEE 11th Symposium on*, pages 21–30. IEEE, 2013.

[2] W. Baek and T. M. Chilimbi. Green: a framework for supporting energy-conscious programming using controlled approximation. In *ACM Sigplan Notices*, volume 45, pages 198–209. ACM, 2010.

[3] J. Diaz-Montes, M. AbdelBaky, M. Zou, and M. Parashar. Cometcloud: Enabling software-defined federations for end-to-end application workflows. *IEEE Internet Computing*, 19(1):69–73, 2015.

[4] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen. Approxhadoop: Bringing approximations to mapreduce frameworks. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 383–397. ACM, 2015.

[5] L. McAfee and K. Olukotun. Emeuro: A framework for generating multi-purpose accelerators via deep learning. In *Proceedings of the 13th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, pages 125–135. IEEE Computer Society, 2015.

[6] S. Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48(4):62, 2016.

[7] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke. Paraprox: Pattern-based approximation for data parallel applications. In *ACM SIGARCH Computer Architecture News*, volume 42, pages 35–50. ACM, 2014.

[8] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke. Sage: Self-tuning approximation for graphics engines. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 13–24. ACM, 2013.

[9] R. St Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger. General-purpose code acceleration with limited-precision analog computation. *ACM SIGARCH Computer Architecture News*, 42(3):505–516, 2014.

[10] V. Vassiliadis, K. Parasyris, C. Chalios, C. D. Antonopoulos, S. Lalis, N. Bellas, H. Vandierendonck, and D. S. Nikolopoulos. A programming model and runtime system for significance-aware energy-efficient computing. In *ACM SIGPLAN Notices*, volume 50, pages 275–276. ACM, 2015.

[11] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: energy-efficient neuromorphic systems using approximate computing. In *Proceedings of the 2014 international symposium on Low power electronics and design*, pages 27–32. ACM, 2014.

[12] Q. Xu, T. Mytkowicz, and N. S. Kim. Approximate computing: A survey. *IEEE Design & Test*, 33(1):8–22, 2016.

[13] T. Yeh, P. Faloutsos, M. Ercegovac, S. Patel, and G. Reinman. The art of deception: Adaptive precision reduction for area efficient physics acceleration. In *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pages 394–406. IEEE, 2007.

[14] A. R. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, A. Anjum, and M. Parashar. Deadline constrained video analysis via in-transit computational environments. *IEEE Transactions on Services Computing*, 2017.

[15] CLoudLab. https://www.cloudlab.us, Last accessed on May 2017.

[16] FIDIA project. http://www.asfidia.it, Last accessed on June 2015.

[17] GRE Tunneling. http://lartc.org/howto/lartc.tunnel.gre.html, Last accessed on May 2017.

[18] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu. Approxann: an approximate computing framework for artificial neural network. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 701–706. EDA Consortium, 2015.

[19] Mininet. http://mininet.org, Last accessed on May 2017.

[20] M. Zou, A. R. Zamani, J. Diaz-Montes, I. Petri, O. Rana, and M. Parashar. Leveraging in-transit computational capabilities in federated ecosystems. In *Service-Oriented System Engineering (SOSE), 2016 IEEE Symposium on*, pages 81–90. IEEE, 2016.

[21] OVS project. http://openvswitch.org, Last accessed on May 2017.

[22] POX controller. https://openflow.stanford.edu/display/ONL/POX+Wiki, Last accessed on May 2017.