SEED: Confidential Big Data Workflow Scheduling with Intel SGX Under Deadline Constraints

Ishtiaq Ahmed[†], Saeid Mofrad[†], Shiyong Lu[†], Changxin Bai[†], Fengwei Zhang[★], Dunren Che[✩]

Wayne State University, USA[†]

Southern University of Science and Technology, China [★]

Southern Illinois University, USA[✩]

WAYNE STATE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

World-Class Education in the Real World™

arriors

Overview

- Introduction
- The Problem
- Existing Approaches and Limitations
- Our Proposed Approach
- Experimental Results
- Conclusions and future work

Introduction

- Big data workflow management systems [2] (BDWFMSs) have recently emerged as popular data analytics platforms to perform large-scale data analytics in the cloud.
- The protection of data confidentiality and secure execution of workflow applications remains an important and challenging problem.
- Monetary cost optimization of executing workflows in the cloud while satisfying a user-defined deadline is also challenging.

The problem of big data workflow scheduling in the cloud

- The Protection of the privacy of the confidential workflow’s tasks in the cloud, whose proprietary algorithm implementations are intellectual properties of the respective stakeholders.

- The monetary cost optimization of executing workflows in the cloud while satisfying a user-defined deadline.

Limitation of previous approaches

- Require many VM instances.
- Operational cost is expensive.
- Scheduling algorithm that can handle confidential tasks is scarce.

What if I need low operational cost with confidentiality and integrity environment?

Challenges for scheduling algorithms

- Deadline-centric: When the deadline is fixed, how we schedule a workflow with minimal cost?

- Budget-centric: When the budget is fixed how we schedule a workflow with minimal makespan?

Contributions

- We propose a novel deadline-constrained and SGX-aware big data workflow scheduling algorithm, SEED (SGX, Efficient, Effective, Deadline Constrained) that leverage hardware-based security features such as Intel Software Guard eXtensions (SGX) [1] in the cloud hardware to protect the execution of security-sensitive workflow's tasks in the cloud environments.

- SEED features several heuristics, including exploiting the longest critical paths and reuse of extra times in existing VMs.

- The proposed SEED, as well as IC-PCP, have been tested in a modified version of DATAVIEW [2], which is one of the most usable big data workflow management systems in the community, to support confidential tasks in a workflow DAG with Intel SGX.

Planner level: Scheduling workflow

Algorithm 1: ScheduleWorkflow(G)

$startNodes \leftarrow nodesWithoutParent$ in G;
$endNodes \leftarrow nodesWithoutChildren$ in G;
connect t_{start} with $startNodes$;
connect t_{end} with $endNodes$;
determine the available virtual machines with cost and billing unit cycle;
$EST(t_{start}) \leftarrow 0$;
$LFT(t_{end}) \leftarrow$ Deadline δ;
calculate EST, EFT from t_{start} to t_{end} recursively;
calculate LFT from t_{end} to t_{start} recursively;
$G \leftarrow$ remove t_{start}, t_{end}, and edges in between from G;
$AssignChildren(G)$;

Planner level: Assigning Children

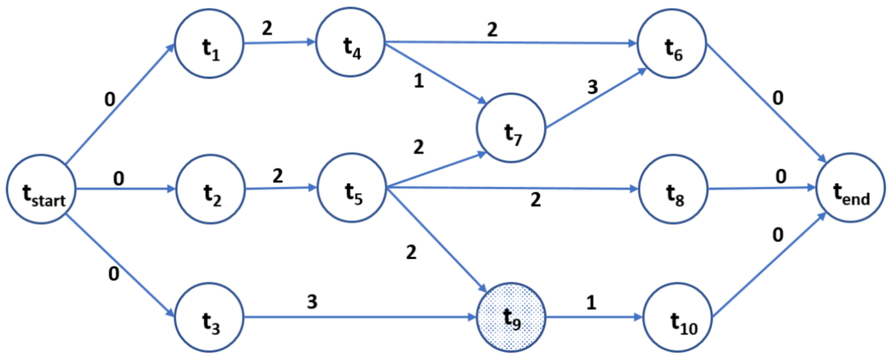
Algorithm 2: AssignChildren(G)

$list \leftarrow$ sort tasks in G in topological order;
while $list \neq \emptyset$ **do**
    $criticalPath \leftarrow \emptyset$;
    $t_i \leftarrow list[1]$ ▷ Selecting first task
    add t_i in $criticalPath$;
    while t_i *has children* **do**
        $t_i \leftarrow criticalChild(t_i)$;
        add t_i in $criticalPath$;
    end
    $paths \leftarrow$ split apart $criticalPath$ by confidential tasks as separators;
    foreach $path \in paths$ **do**
        $AllocateVirtualMachine(path)$;
        foreach $t_i \in path$ **do**
            recalculate LFT of the ancestors $\{t_k : t_k \notin path\}$ of t_i;
            recalculate EST, EFT of the descendants $\{t_k : t_k \notin path\}$ of t_i;
        end
        $G \leftarrow$ remove all tasks and edges involving tasks in $path$ from G;
        $list \leftarrow$ remove all tasks involving in $path$ from $list$;
    end
end

Planner level: Allocating virtual machines

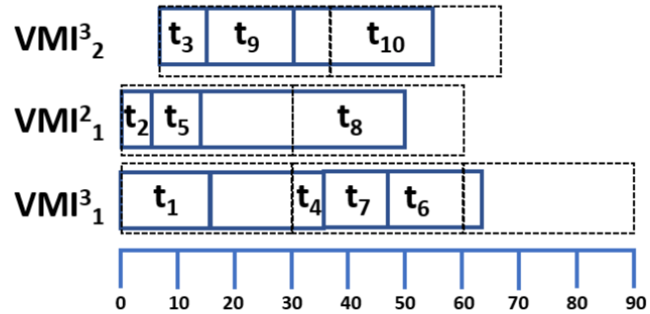
Algorithm 3: AllocateVirtualMachine(path)

$VMI^c \leftarrow$ existing-VM-instances;
if secure environment is required for $path$ **then**
  launch a new instance vmi^{SGX} for $path$;
  $VMI^c \leftarrow vmi^{SGX}$;
  return;
foreach existing VM instance $vmi \in VMI^c$ in ascending order of cost **do**
  oldAssignment $\leftarrow$ assignment(vmi);
  $Q \leftarrow assignment(vmi)$; ▷ queued tasks at vmi
  ▷ Case 1: Insert $path$ into middle of Q
  if there exists a task $t_k \in Q$ such that $t_k \in path[last].children$ **then**
    insert $path$ into Q before t_k and recalculate EST, EFT, LFT of affected tasks in Q;
    if assignment(vmi) is valid **then**
      return; ▷ keep current $assignment(vmi)$
    else
      assignment(vmi) $\leftarrow$ oldAssignment;
    end
  end
  ▷ Case 2: Insert $path$ at head of Q
  insert $path$ at head Q and recalculate EST, EFT, LFT of affected tasks in Q;
  if assignment(vmi) is valid **then**
    return; ▷ keep current $assignment(vmi)$
  else
    assignment(vmi) $\leftarrow$ oldAssignment;
  end
  ▷ Case 3: Insert $path$ at end of Q
  insert $path$ at end of Q and recalculate EST, EFT, LFT of affected tasks in Q;
  if assignment(vmi) is valid **then**
    return; ▷ keep current $assignment(vmi)$
  else
    assignment(vmi) $\leftarrow$ oldAssignment;
  end
end
▷ No existing VM instance can accommodate $path$
launch a new cheapest but capable VM instance vmi^c for executing the tasks in $path$ before their LFT;
$VMI^c \leftarrow vmi^c$;
return

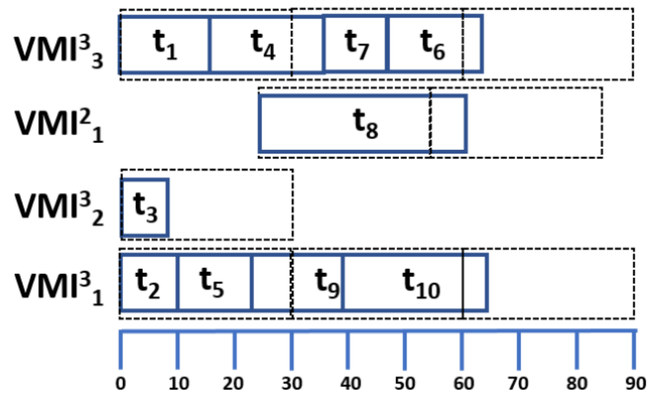


Type	Start	End	Duration	Interval cycle	Cost	Selected task
VMI_1^3	0	66	66	3	15	$\{t_1, t_4, t_7, t_6\}$
VMI_1^2	0	51	51	2	16	$\{t_2, t_5, t_8\}$
VMI_2^3	8	58	50	2	10	$\{t_3, t_9, t_{10}\}$

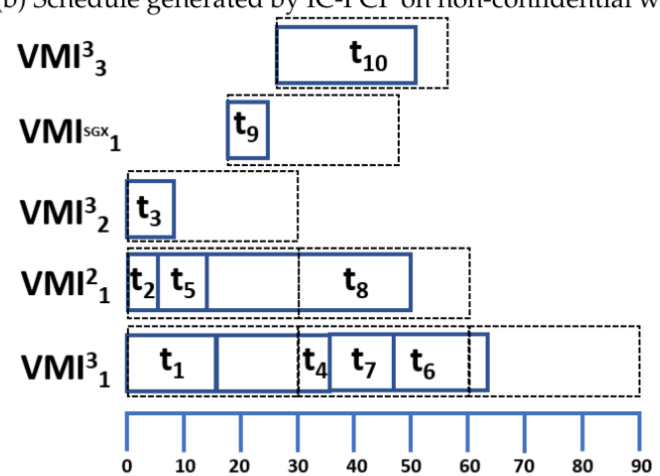
Type	Start	End	Duration	Interval cycle	Cost	Selected task
VMI_1^3	0	66	66	3	15	$\{t_1, t_4, t_7, t_6\}$
VMI_1^2	0	51	51	2	16	$\{t_2, t_5, t_8\}$
VMI_2^3	0	9	9	1	5	$\{t_3\}$
VMI_1^{SGX}	17	24	7	1	25	$\{t_9\}$
VMI_3^3	25	50	25	1	5	$\{t_{10}\}$



(a) Schedule generated by SGX-E2C2D on non-confidential workflow.

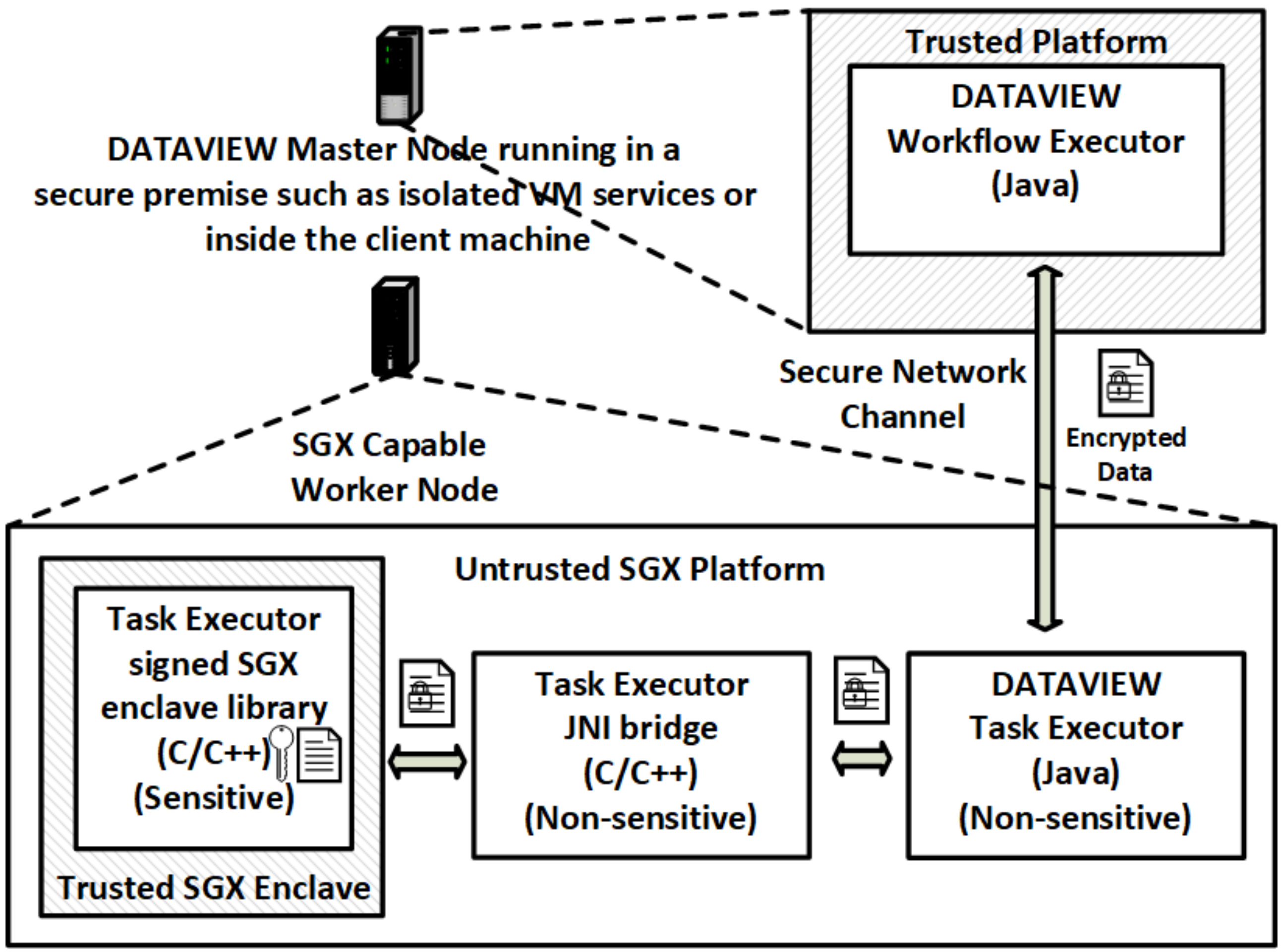


(b) Schedule generated by IC-PCP on non-confidential workflow.



(c) Schedule generated by SGX-E2C2D on confidential workflow.

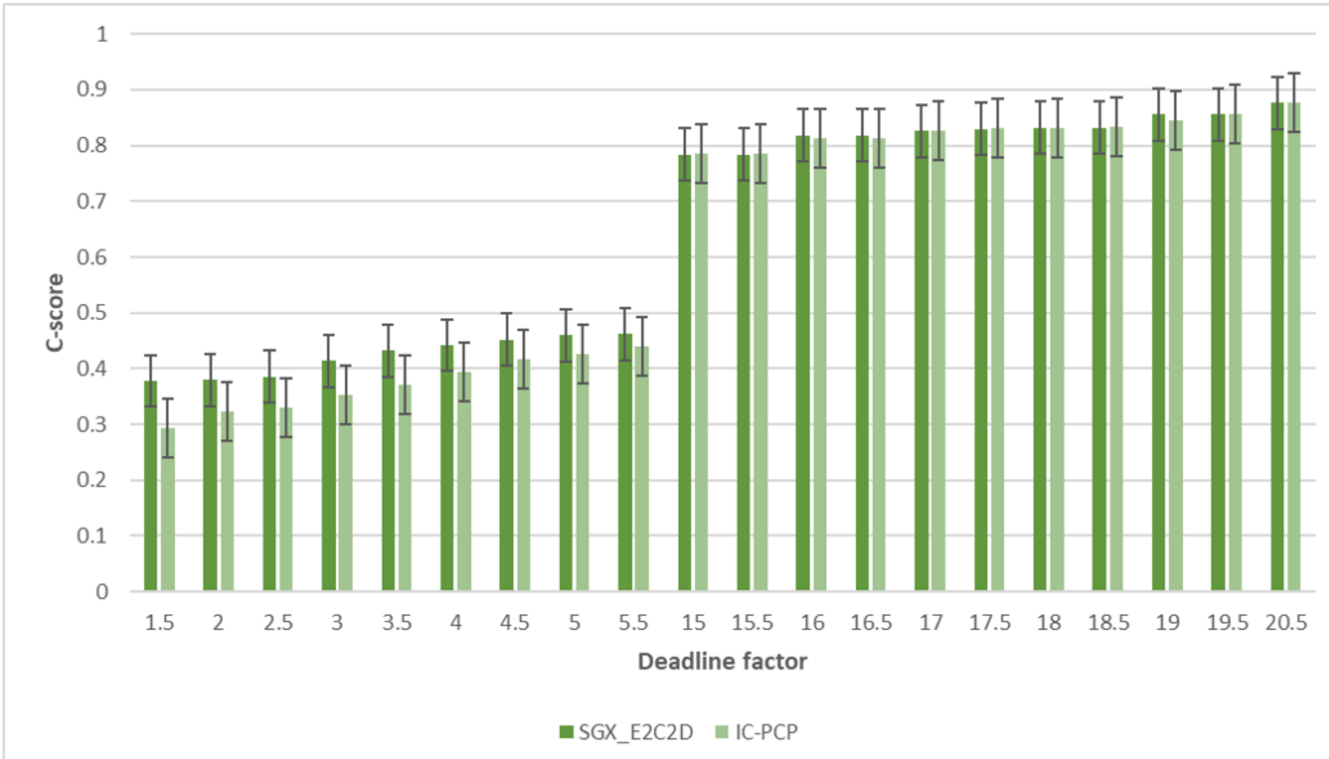
The architecture of DATAVIEW Task Executor with SGX support



Experimental settings

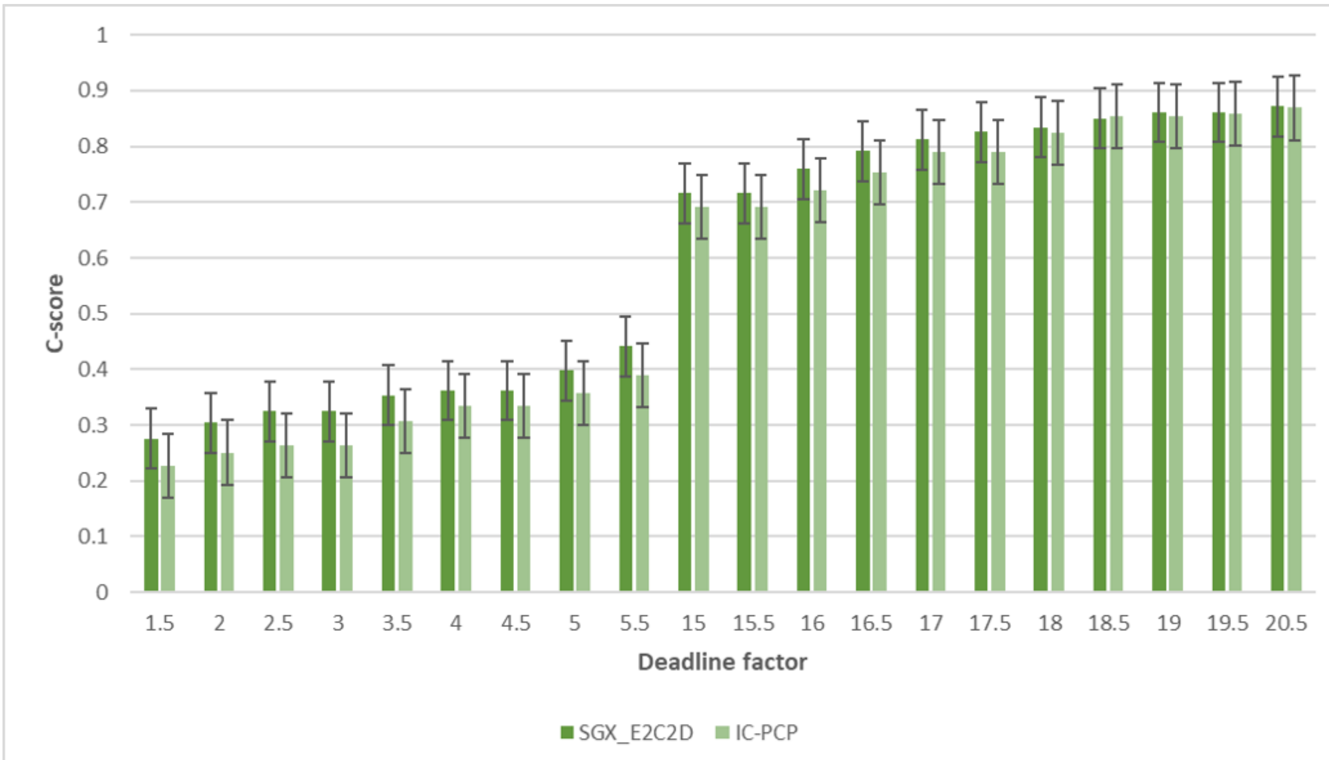
Testbed Machine	Intel SGX	AWS
CPU Model	Xeon E3-1275 V6	Intel(R) Xeon(R) CPU E5-2676 v3
CPU Physical Core Number	4	8
CPU Logical Thread Number	8	16
CPU Base Clock	3.8GHz	2.4GHz
CPU Boost Clock	4.2GHz	3.2GHz
Cache Type	Smart Cache	Smart Cache
Cache Size	8MB	20MB
Motherboard	Supermicro X11SSW-F	AWS
Memory	32GB DDR4 ECC	1GB DDR4 ECC
Storage	HDD	EBS
Operating System	CentOS 7.0	Linux 16.04 LTS
Kernel Version	3.10.0-862.9.1.el7.x86_64 x86_64	4.4.0-1060-aws
SGX SDK Version	SGX SDK Ver 2.2	SGX SDK Ver 2.0
AMI family	N/A	General purpose
AMI type	N/A	t2.micro
AMI Assigned CPU Core	N/A	1

Performance analysis Montage Workflow (10sec)

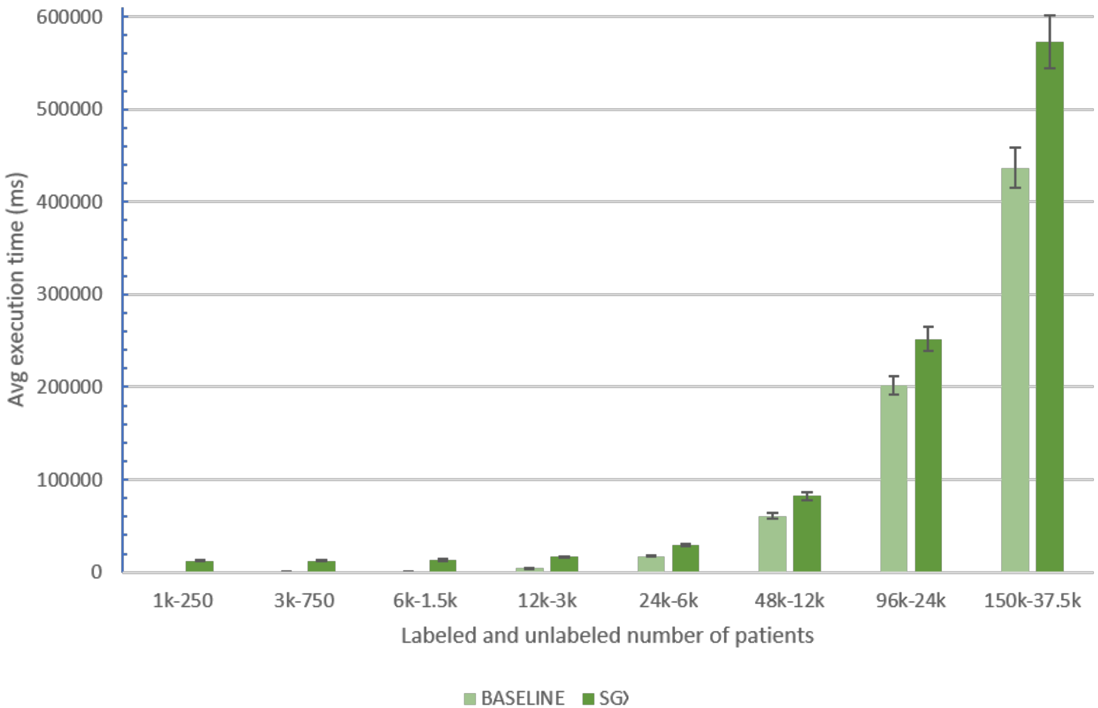


$$C = \begin{cases} 0.5 + 0.5 * \frac{(maxcost - cost)}{maxcost}, & if\ makespan \leq \delta \\ 0.5 - 0.5 * \frac{(makespan - \delta)}{(maxmakespan - \delta)}, & otherwise \end{cases}$$

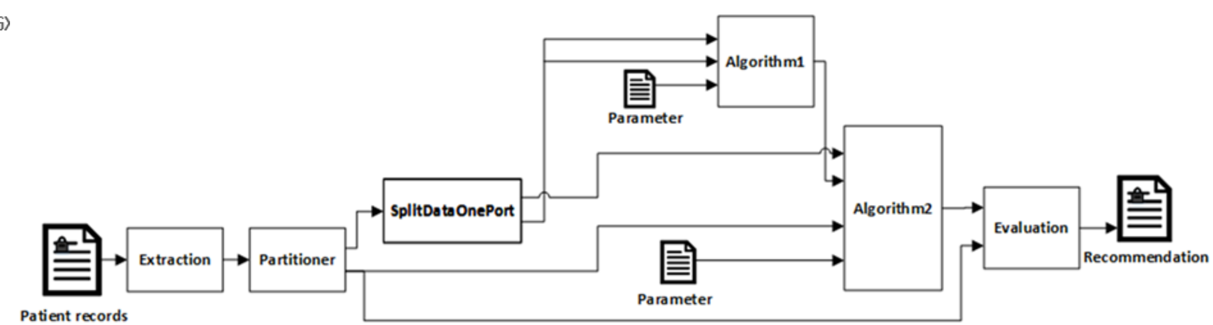
Performance analysis Montage Workflow (60s)



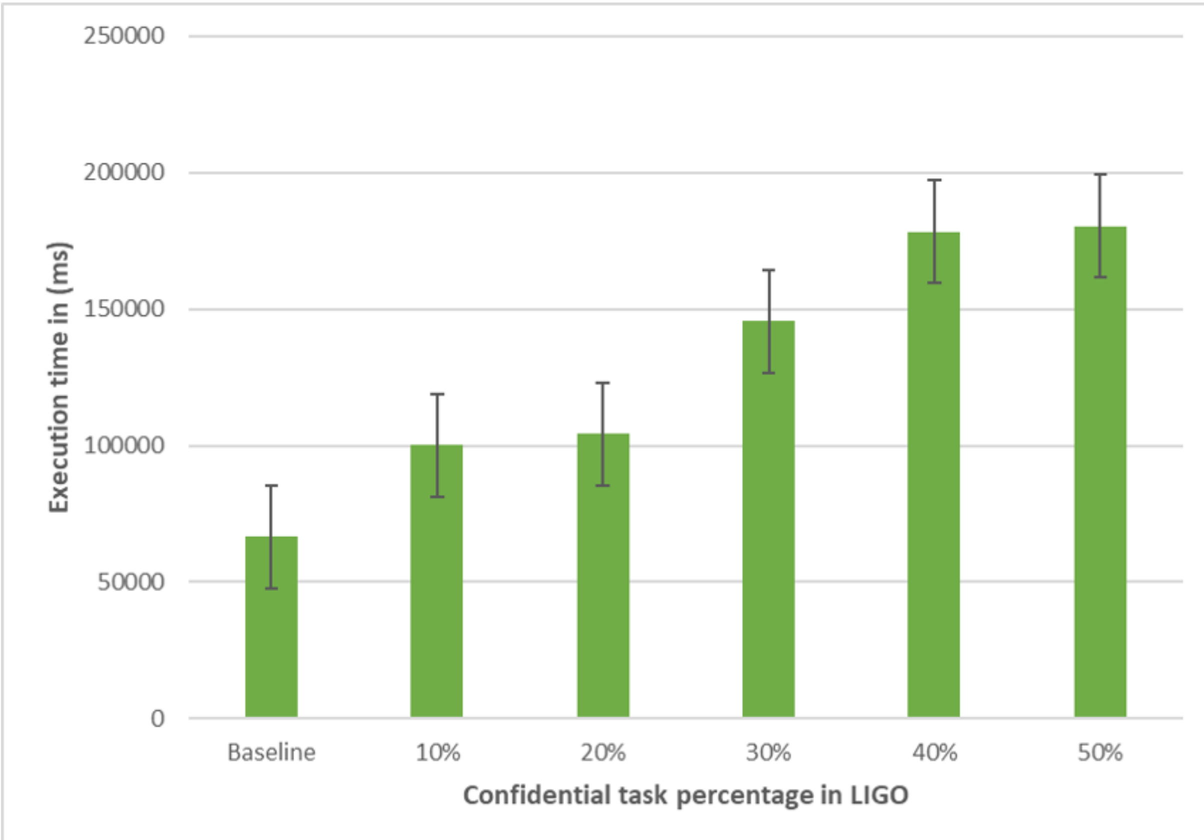
Performance analysis of algorithm1 in the diagnosis recommendation workflow[3]



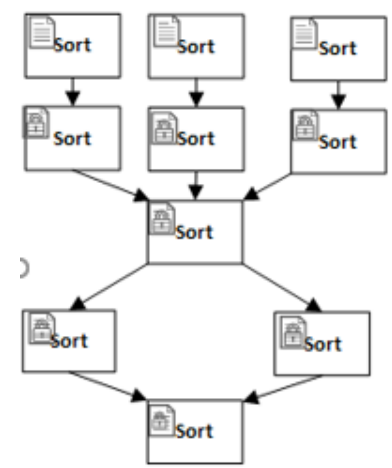
- SGX overhead reported a range of 1.24x to 1.31x overhead compared with the base line with 96k-150k labeled and 24k-37.5k unlabeled instances, respectively



Performance overhead of LIGO workflow

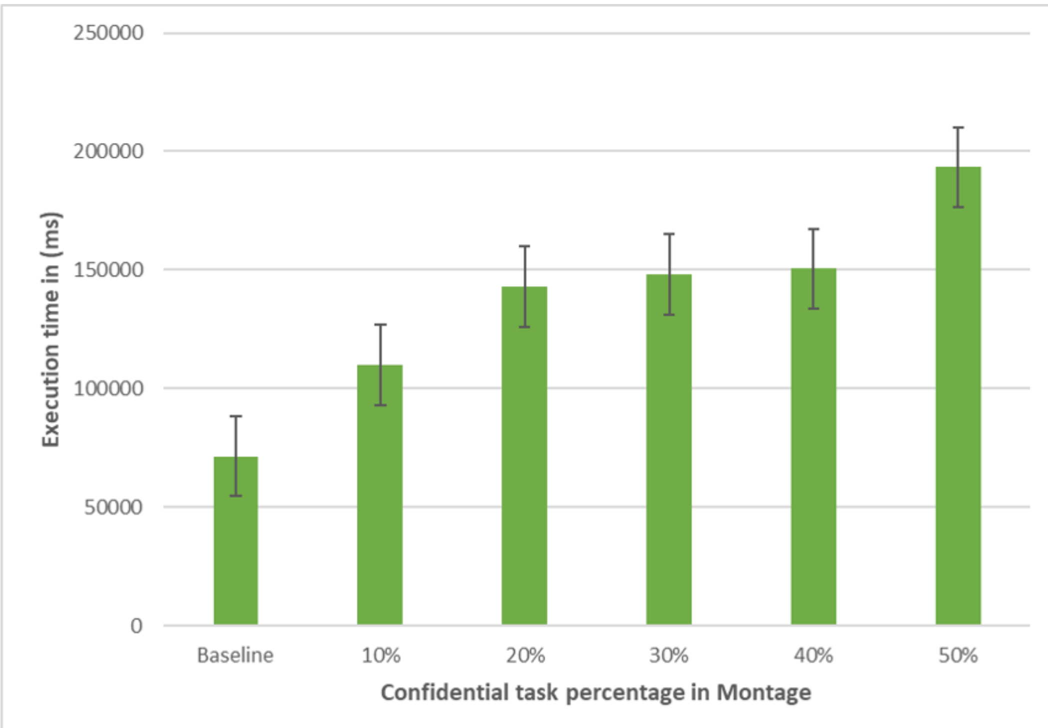


- The average performance overheads for the LIGO workflow shows a range between 1.81x in 10% and increases to 2.50x in 50% confidential task assignments.

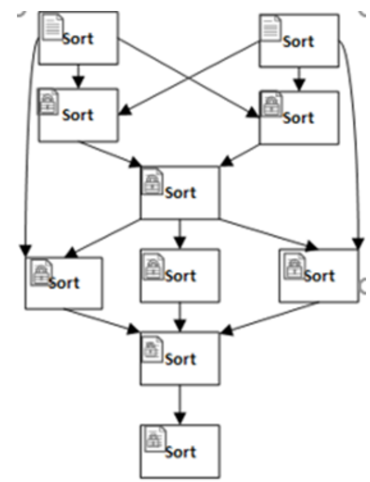


LIGO Workflows

Performance overhead of Montage workflow



- The average performance overheads for the Montage workflow shows a range between 1.34x in 10% and increases to 2.50x in 50% confidential task assignments.

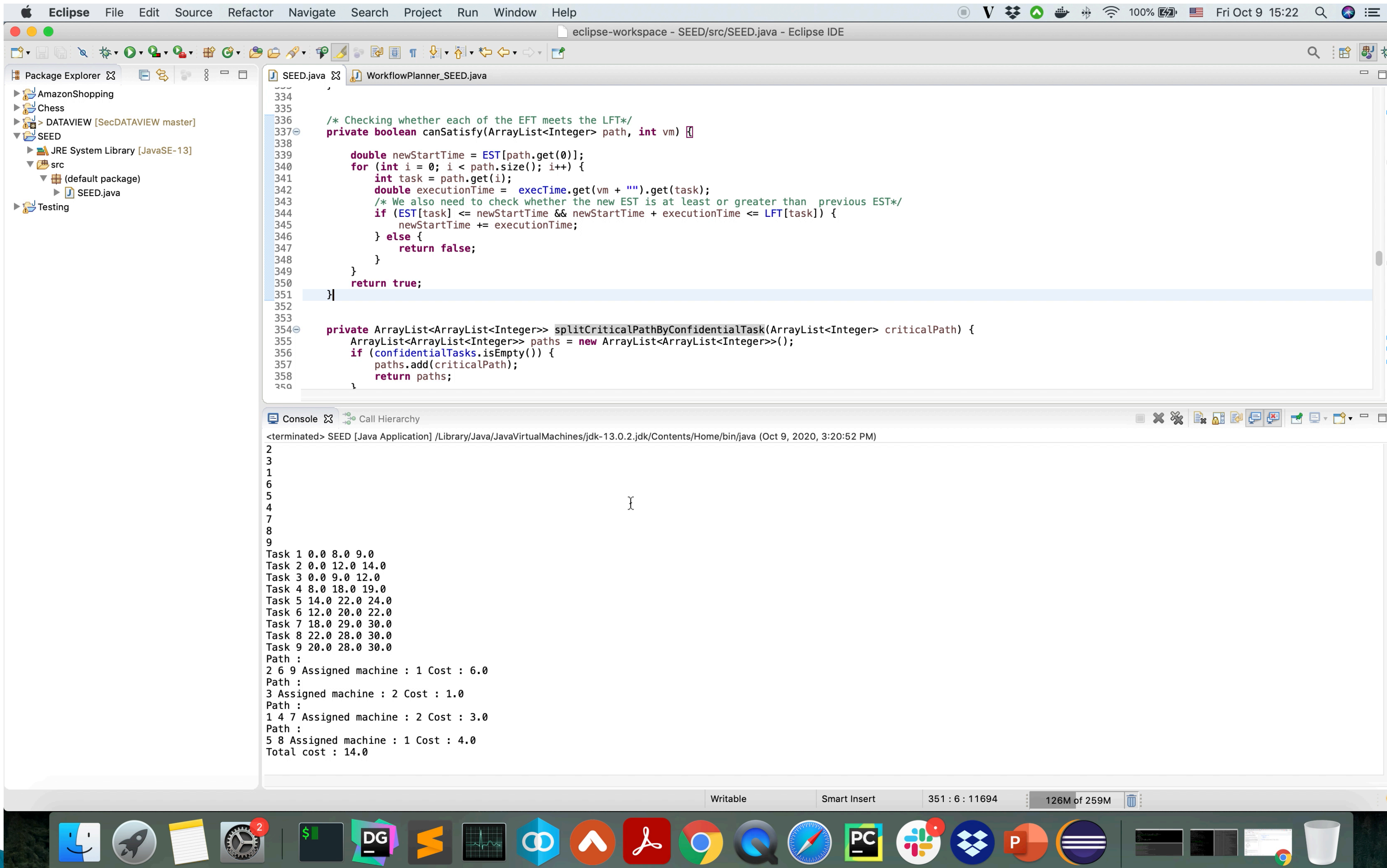


Montage Workflows

Conclusions and future work

- This paper presents an efficient, cost-effective deadline constrained algorithm SEED for scientific workflows in IaaS clouds.
- To the best of our knowledge, this is one of the first scheduling algorithms that can handle confidential workflow tasks.
- The primary intention is to schedule confidential tasks and achieve a low-cost scheduling algorithm within the given deadline while the complexity remains feasible for running large-scale workflows.
- For future works, we intend to protect the security of big data workflows in the heterogeneous cloud environments, including Intel SGX, AMD SEV [5], and GPU accelerators.

SEED Demo


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Fri Oct 9 15:22
eclipse-workspace - SEED/src/SEED.java - Eclipse IDE
Package Explorer
AmazonShopping
Chess
> DATAVIEW [SecDATAVIEW master]
SEED
JRE System Library [JavaSE-13]
src
(default package)
SEED.java
Testing
SEED.java
WorkflowPlanner_SEED.java
/* Checking whether each of the EFT meets the LFT*/
private boolean canSatisfy(ArrayList<Integer> path, int vm) {
double newStartTime = EST[path.get(0)];
for (int i = 0; i < path.size(); i++) {
int task = path.get(i);
double executionTime = execTime.get(vm + "").get(task);
/* We also need to check whether the new EST is at least or greater than previous EST*/
if (EST[task] <= newStartTime && newStartTime + executionTime <= LFT[task]) {
newStartTime += executionTime;
} else {
return false;
}
}
return true;
}
private ArrayList<ArrayList<Integer>> splitCriticalPathByConfidentialTask(ArrayList<Integer> criticalPath) {
ArrayList<ArrayList<Integer>> paths = new ArrayList<ArrayList<Integer>>();
if (confidentialTasks.isEmpty()) {
paths.add(criticalPath);
return paths;
Console
Call Hierarchy
<terminated> SEED [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.2.jdk/Contents/Home/bin/java (Oct 9, 2020, 3:20:52 PM)
2
3
1
6
5
4
7
8
9
Task 1 0.0 8.0 9.0
Task 2 0.0 12.0 14.0
Task 3 0.0 9.0 12.0
Task 4 8.0 18.0 19.0
Task 5 14.0 22.0 24.0
Task 6 12.0 20.0 22.0
Task 7 18.0 29.0 30.0
Task 8 22.0 28.0 30.0
Task 9 20.0 28.0 30.0
Path :
2 6 9 Assigned machine : 1 Cost : 6.0
Path :
3 Assigned machine : 2 Cost : 1.0
Path :
1 4 7 Assigned machine : 2 Cost : 3.0
Path :
5 8 Assigned machine : 1 Cost : 4.0
Total cost : 14.0
Writable
Smart Insert
351 : 6 : 11694
126M of 259M

SecDATAVIEW Introduction

- SecDATAVIEW [4] system is our advance design of DATAVIEW that protects the confidentiality and privacy of big data workflows in the heterogeneous cloud environments.

- SecDATAVIEW can leverage hardware-based security features such as Intel SGX [2] and AMD SEV [5] and guarantee the secure execution of confidential workflow in the untrusted cloud environments.

- Access DATAVIEW and SecDATAVIEW:
  - https://github.com/shiyonglu/DATAVIEW
  - https://github.com/shiyonglu/SecDATAVIEW
  - DATAVIEW YouTube Channel:
  - https://www.youtube.com/channel/UCrIEBUmju-NMKFMlFbsKBYw

DATAVIEW TEAM

Thank You.

Questions?

References:

- [1] McKeen *et al.*, "Innovative instructions and software model for isolated execution.," in HASP@ISCA, 2013, p. 10.
- [2] A. Kashlev, S. Lu, and A. Mohan, "Big Data Workflows: a Reference Architecture and the DATAVIEW System," *Serv. Trans. Big Data*, vol. 4, no. 1, pp. 1–19, 2017.
- [3] Ahmed et al., "Diagnosis Recommendation using Machine Learning Scientific Workflows," in *Big Data Congress, 2018 IEEE International Conference on*, 2018.
- [4] Mofrad, S., Ahmed, I., Lu, S., Yang, P., Cui, H., & Zhang, F. (2019, December). SecDATAVIEW: a secure big data workflow management system for heterogeneous computing environments. In *Proceedings of the 35th Annual Computer Security Applications Conference* (pp. 390–403).
- [5] Kaplan, D., Powell, J., & Woller, T. (2016). AMD memory encryption. *White paper*.