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*
Overview

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

1. $\text{startNodes} \leftarrow \text{nodesWithoutParent} \text{ in } G$
2. $\text{endNodes} \leftarrow \text{nodesWithoutChildren} \text{ in } G$
3. connect $t_{\text{start}}$ with $\text{startNodes}$
4. connect $t_{\text{end}}$ with $\text{endNodes}$
5. determine the available virtual machines with cost and billing unit cycle
6. $\text{EST}(t_{\text{start}}) \leftarrow 0$
7. $\text{LFT}(t_{\text{end}}) \leftarrow \text{Deadline } \delta$
8. calculate $\text{EST}$, $\text{EFT}$ from $t_{\text{start}}$ to $t_{\text{end}}$ recursively
9. calculate $\text{LFT}$ from $t_{\text{end}}$ to $t_{\text{start}}$ recursively
10. $G \leftarrow \text{remove } t_{\text{start}}, t_{\text{end}}, \text{ and edges in between from } G$
11. $\text{AssignChildren}(G)$
Planner level: Assigning Children

**Algorithm 2:** AssignChildren(G)

1. `list ← sort tasks in G in topological order;`
2. **while** `list ≠ ∅` **do**
   3. `criticalPath ← ∅;`
   4. `t_i ← list[1]`  
      ▷ Selecting first task
   5. add `t_i` in `criticalPath;`
   6. **while** `t_i has children` **do**
      7. `t_i ← criticalChild(t_i);`
      8. add `t_i` in `criticalPath;`
   9. **end**
10. `paths ← split apart criticalPath by confidential tasks as separators;`
11. **foreach** `path ∈ paths` **do**
    12. `AllocateVirtualMachine(path);`
    13. **foreach** `t_i ∈ path` **do**
    14. recalculate LFT of the ancestors `{t_k : t_k ∉ path}` of `t_i;`
    15. recalculate EST, EFT of the descendants `{t_k : t_k ∉ path}` of `t_i;`
    16. **end**
17. `G ← remove all tasks and edges involving tasks in path from G;`
18. `list ← remove all tasks involving in path from list;`
19. **end**
20. **end**
Algorithm 3: AllocateVirtualMachine(path)

1. $VMIC \leftarrow$ existing-VM-instances;
2. if secure environment is required for path then
3.     launch a new instance $vmi^{SGX}$ for path;
4.     $VMIC \leftarrow vmi^{SGX}$;
5.     return;
6. foreach existing VM instance $vmi \in VMIC$ in ascending order of cost do
7.     oldAssignment $\leftarrow$ assignment($vmi$);
8.     $Q \leftarrow$ assignment($vmi$); ▷ queued tasks at $vmi$
9.     ▷ Case 1: Insert path into middle of $Q$
10. if there exists a task $t_k \in Q$ such that $t_k \in path[]$.children then
11.     insert path into $Q$ before $t_k$ and recalculate EST, EFT,
12.     LFT of affected tasks in $Q$;
13.     if assignment($vmi$) is valid then
14.         return; ▷ keep current assignment($vmi$)
15.     else
16.         assignment($vmi$) $\leftarrow$ oldAssignment;
17.     end
18. ▷ Case 2: Insert path at head of $Q$
19. insert path at head $Q$ and recalculate EST, EFT, LFT of
20. affected tasks in $Q$;
21. if assignment($vmi$) is valid then
22.     return; ▷ keep current assignment($vmi$)
23. else
24.     assignment($vmi$) $\leftarrow$ oldAssignment;
25. end
26. ▷ Case 3: Insert path at end of $Q$
27. insert path at end of $Q$ and recalculate EST, EFT, LFT of
28. affected tasks in $Q$;
29. if assignment($vmi$) is valid then
30.     return; ▷ keep current assignment($vmi$)
31. else
32.     assignment($vmi$) $\leftarrow$ oldAssignment;
33. end
34. ▷ No existing VM instance can accommodate path
35. launch a new cheapest but capable VM instance $vmi^{IC}$ for executing
36. the tasks in path before their LFT;
37. $VMIC \leftarrow vmi^{IC}$;
38. return

Planner level: Allocating virtual machines
(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

DATAVIEW Master Node running in a secure premise such as isolated VM services or inside the client machine.

SGX Capable Worker Node

Trusted Platform

DATAVIEW Workflow Executor (Java)

Secure Network Channel

Encrypted Data

Untrusted SGX Platform

Task Executor signed SGX enclave library (C/C++) (Sensitive)

Trusted SGX Enclave

Task Executor JNI bridge (C/C++) (Non-sensitive)

DATAVIEW Task Executor (Java) (Non-sensitive)
## Experimental settings

<table>
<thead>
<tr>
<th>Testbed Machine</th>
<th>Intel SGX</th>
<th>AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Model</td>
<td>Xeon E3-1275 V6</td>
<td>Intel(R) Xeon(R) CPU E5-2676 v3</td>
</tr>
<tr>
<td>CPU Physical Core Number</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>CPU Logical Thread Number</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>CPU Base Clock</td>
<td>3.8GHz</td>
<td>2.4GHz</td>
</tr>
<tr>
<td>CPU Boost Clock</td>
<td>4.2GHz</td>
<td>3.2GHz</td>
</tr>
<tr>
<td>Cache Type</td>
<td>Smart Cache</td>
<td>Smart Cache</td>
</tr>
<tr>
<td>Cache Size</td>
<td>8MB</td>
<td>20MB</td>
</tr>
<tr>
<td>Motherboard</td>
<td>Supermicro X11SSW-F</td>
<td>AWS</td>
</tr>
<tr>
<td>Memory</td>
<td>32GB DDR4 ECC</td>
<td>1GB DDR4 ECC</td>
</tr>
<tr>
<td>Storage</td>
<td>HDD</td>
<td>EBS</td>
</tr>
<tr>
<td>Operating System</td>
<td>CentOS 7.0</td>
<td>Linux 16.04 LTS</td>
</tr>
<tr>
<td>Kernel Version</td>
<td>3.10.0-862.9.1.el7.x86_64 x86_64</td>
<td>4.4.0-1060-aws</td>
</tr>
<tr>
<td>SGX SDK Version</td>
<td>SGX SDK Ver 2.2</td>
<td>SGX SDK Ver 2.0</td>
</tr>
<tr>
<td>AMI family</td>
<td>N/A</td>
<td>General purpose</td>
</tr>
<tr>
<td>AMI type</td>
<td>N/A</td>
<td>t2.micro</td>
</tr>
<tr>
<td>AMI Assigned CPU Core</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>
Performance analysis Montage Workflow (10sec)

\[ C = \begin{cases} 
0.5 + 0.5 \times \frac{(\text{maxcost} - \text{cost})}{\text{maxcost}}, & \text{if makespan} \leq \delta \\
0.5 - 0.5 \times \frac{(\text{makespan} - \delta)}{\text{maxmakespan} - \delta}, & \text{otherwise} 
\end{cases} \]
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.
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.
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.
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.
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-NMKFMIFbsKBYw
DATAVIEW TEAM
Thank You.

Questions?
References:


