Aligning Confidential Computing with Cloud-native ML Platforms

Angelo Ruocco, Chris Porter, Claudio Carvalho, Daniele Buono,
Derren Dunn, Hubertus Franke, James Bottomley, Marcio Silva,
Mengmei Ye, Niteesh Dubey, and Tobin Feldman-Fitzthum
IBM Research

***Abstract*—Developers leverage machine learning (ML) platforms to handle a range of their ML tasks in the cloud, but these use cases have not been deeply considered in the context of confidential computing. Confidential computing's threat model treats the cloud provider as untrusted, so the user's data in use (and certainly at rest) must be encrypted and integrity-protected. This host-guest barrier presents new challenges and opportunities in the ML platform space. In particular, we take a glancing look at ML platforms' pipeline tools, how they currently align with the Confidential Containers project, and what may be needed to bridge several gaps.**

ML pipelines are a programming abstraction for executing multiple steps in a typical ML development flow. These "pipelines" are in fact directed acyclic graphs (DAGs), where each node is some computation, and the edges represent dependencies. For example, the nodes may represent Python scripts for data cleaning, training, hyperparameter tuning, evaluation, deployment, monitoring, etc.; and the edges signify their input/output dependencies. ML platforms may expose pipeline support to the programmer via Python libraries (e.g. in Azure ML Pipelines or AWS SageMaker Pipelines) or decorators (e.g. in Kubeflow or Metaflow). Regardless, the programmer is responsible for specifying the input/output edges of the DAG and the functions/computations of the nodes.

The Confidential Containers (CoCo) [1] project has shown how one can set up and run ML workloads in the cloud [2], but there is no consideration yet for ML pipelines. In fact, CoCo is yet to release a v1.0, but there are already several usage assumptions taking shape. In rough terms, the Kubernetes control plane (which CoCo leverages) is assumed to be untrusted and part of the cloud provider's software stack. The Kubernetes operator pattern, however, allows CoCo to deploy its custom container technology (based on Kata Containers [3]), which is ultimately responsible for jump-starting the secure pods within a trusted execution environment (TEE).

We outline several challenges and considerations at the confluence of CoCo and ML pipeline technologies:

1) CoCo requires hardware, firmware, and kernel support.
2) CoCo deployments run each pod inside of a Kata VM in a Kubernetes deployment. This (a) assumes Kubernetes is used with a custom operator and (b) poses challenges when the Kubernetes control plane itself is virtualized.
3) ML pipelines do not currently expose any way for users to indicate a confidential environment.
4) An ML pipeline's interface and features may require restrictions when confidentiality is a first-class concern.
5) Most importantly, ML pipelines are not designed to work with protected nodes and edges in their DAGs.

Any ML pipeline framework would of course need to deploy on CPUs with confidential computing features (e.g. Intel TDX or AMD SEV-SNP technology) and associated kernel support; for GPUs, such support is still novel, but it would be a smaller concern with wider-spread adoption (1). CoCo currently supports a "peer pods" feature to handle nested virtualization issues (2). Thus, launching an ML pipeline on virtualized Kubernetes would require compatibility with this kind of CoCo feature. Regarding (3), CoCo users specify a confidential environment as part of their deploymet YAML. ML pipelines need some minimal configuration support to match this. Any ML pipeline-specific features would require careful vetting (4). For example, AWS SageMaker allows users to cache pipeline steps, and Google Vertex AI Pipelines allows users to leverage "predefined components." Such features may need to be dropped or modified for confidential computing-enabled pipelines. Lastly, all ML pipelines would need to secure the computations and data (5). For example, specifying mounts in a pipeline step's source code must result in *encrypted* mounts (data edges); all computations (nodes) must be encrypted before they are uploaded to the cloud provider; and such (trusted, client-side) library support would need to be open to the user. Thus, in a "CoCo-enlightened ML pipeline," each compute node in the DAG could be a container image; each of these would interact with a relying party for attestation and decryption; and the I/O dependencies would be encrypted.

References

[1] Confidential containers. Accessed: 2024 Jan 8. [Online]. Available: https://confidentialcontainers.org/
[2] Confidential containers for enhancing ai workload security in the public cloud. Accessed: 2024 Jan 8. [Online]. Available: https://www.redhat.com/en/blog/enhancing-ai-workload-security-in-the-public-cloud
[3] Kata containers. Accessed: 2024 Jan 10. [Online]. Available: https://katacontainers.io/

Workshop on AI Systems with Confidential Computing (AISCC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-4-5
https://dx.doi.org/10.14722/aiscc.2024.23014
www.ndss-symposium.org