CAREER: Transactional Machine Learning for Durability, Atomicity, Replicability, and Time Versioning

Yongjoo Park (yongjoo@g.illinois.edu), Computer Science, UIUC

From interactive AI to scientific computing, interpreted languages like Python are widely used for compute-/data-intensive applications. Although its syntax differs from specialized data languages like SQL, their fundamental nature remains the same: statements may alter the state of data managed by the application or the system. Unlike database systems (e.g., PostgreSQL), however, Python applications are fragile; they can unexpectedly terminate at any point due to human errors or engineering issues, losing all of its data. That is, ***Python applications lack durability, atomicity, replicability, and time versioning***. *No Durability:* If an app terminates, no state persists. *No Atomicity:* An interrupted function may result in partial updates. *No Replication:* A running application cannot be easily replicated to another machine and resumed there. *No Versioning:* We cannot retrieve a past state (e.g., models in a previous epoch). While orchestration services (e.g., Kubernetes) can enhance system robustness, they cannot restore data. Manual *point solutions* (e.g., ModelDB) may miss important information, are time-consuming, and incur unnecessary costs.

This project aims to bring ***durability, atomicity, replicability, and time versioning (DART) to Python applications*** with a focus on machine learning and data science. By reinterpreting *transactions* in a novel domain of Python, this project will allow applications to create *dynamic checkpoints* of their states efficiently in a platform-agnostic manner, enabling the following new opportunities. *Durability:* Application states will be automatically persisted to survive unexpected errors or failures. *Atomicity:* Only completed statements will result in valid states for persistence. *Replicability:* Running apps can be copied onto different machines and resumed. *Versioning:* Apps can resume from any past state. To accomplish DART, this project will investigate (1) new mathematical representations and techniques for efficiently expressing an application history (Thrust 1), and (2) correct and optimized mechanisms for saving the new states of running applications (Thrust 2). Our new framework will offer DART with no changes to existing code as well as the Python kernel (Fig 1) while ensuring low monitoring overhead. Ultimately, this project will contribute to reliable and elastic infrastructure for complex, data-intensive applications. Finally, the significance of this project will ever increase as machine learning involves larger models and computationally demanding operations.

Intellectual Merit

This project will bring a new perspective on managing machine learning applications through the lens of transactions. In Thrust 1, we will significantly extend existing work on data lineage, and also will develop mathematical representations for general-purpose applications. Thrust 2 will explore novel optimization problems and techniques for efficient replication and persistence.

Broader Impacts

The proposed education and outreach activities aim to maximize the project's societal benefits. ***Broadening Participation:*** Expanding on the *three-pronged* curriculum design — the PI's novel approach for emphasizing algorithm, system, and inclusive collaboration — the PI will offer data science courses at five levels of difficulties for a diverse range of learners, from middle school students to university graduate students. ***Infrastructure:*** Building data science frameworks in collaboration with people outside of computer science will contribute to secure and accessible infrastructure for data-intensive computing. ***US Leadership in Cloud Computing:*** The benefits of DART are maximized with reliable storage, fast networks, and elastic VMs, which can be best offered by large-scale infrastructure providers. By working closely with NCSA and U.S. companies, this project will solidify their positions in offering state-of-the-art computing services.

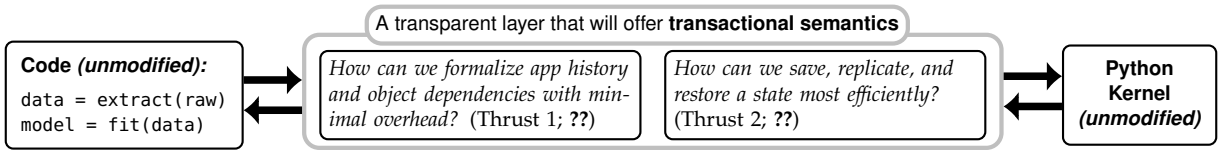

Figure 1: Our framework (middle box) aims to bring ***durability, atomicity, replicability, and time versioning (DART)*** with little to no modifications to the application code (left) as well as the Python kernel (right).