Online Learning with SmartSim

An overview for the SDL21 workshop

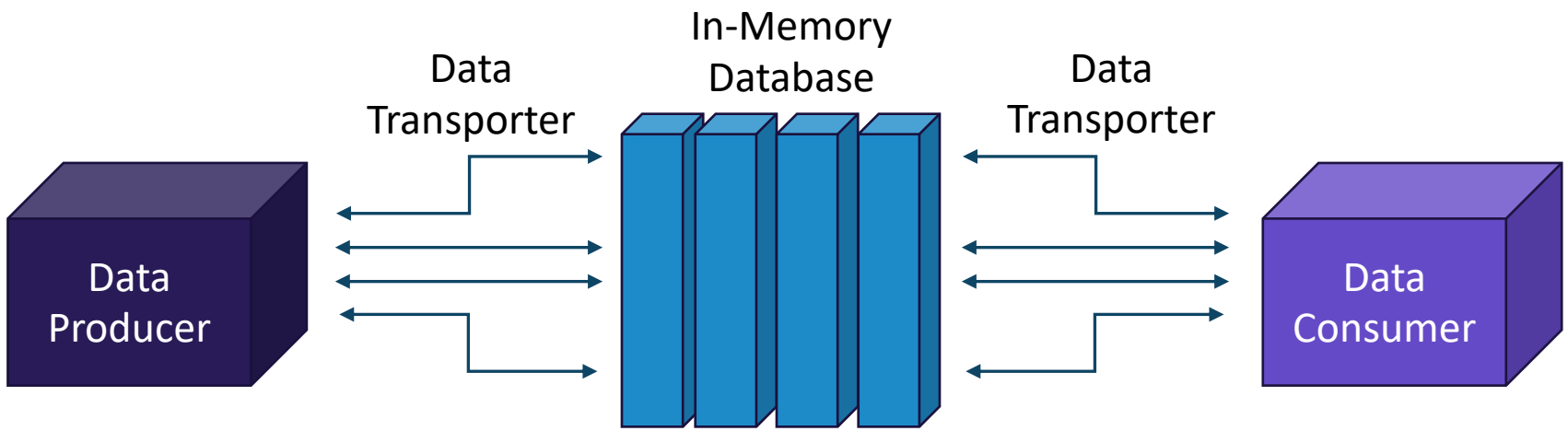
Riccardo Balin and Filippo Simini

Argonne National Laboratory, LCF

10/07/2021

Online Learning with SmartSim

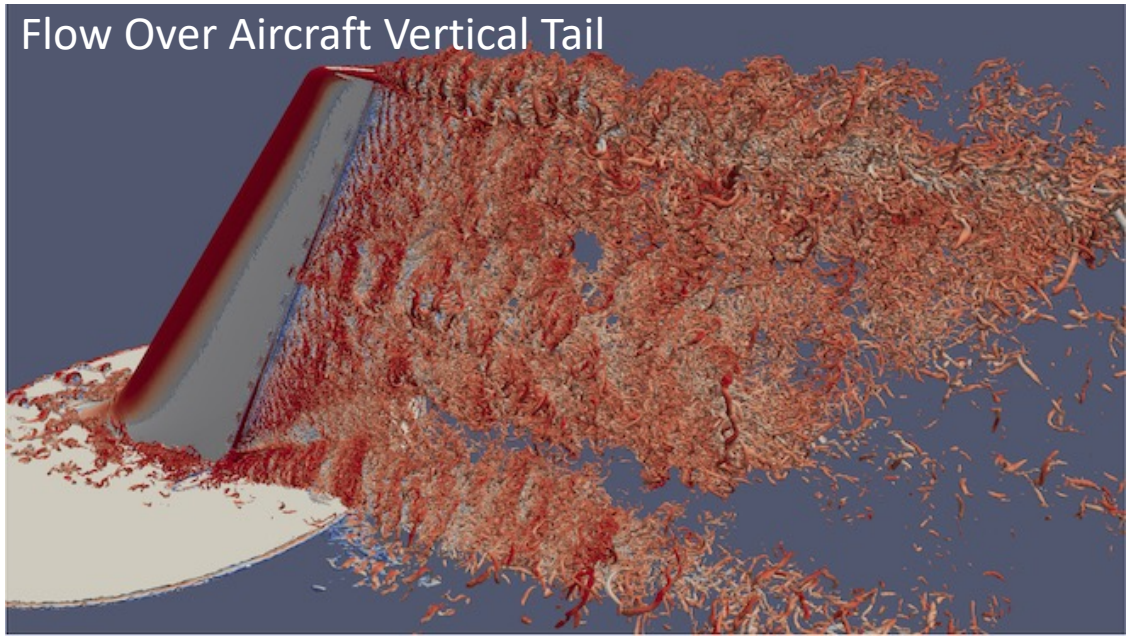
- Four components: data producer, data consumer, data transporter and in-memory database



Online Learning with SmartSim

Data Producer

- Any PDE solver that integrates a set of governing equations written in C, C++, Fortran, Python
- Successively generated solution states are used to produce instantaneous snapshots of training data
- The training data is sent to the database with desired frequency
- E.g., computational fluid dynamics (CFD) solver computing complex time/scale resolved turbulent flows



Online Learning with SmartSim

Data Consumer

- Any program consuming the solution data from the PDE solver
- Can be parallel or serial
- Can be written in C, C++, Fortran, Python
- Examples:
  - Data parallel ML algorithm
  - Compute intensive data analysis
  - Data compression
  - Solution visualization

Online Learning with SmartSim

Data Transporter and In-Memory Database

- Both components provided by SmartSim
- Open source tool developed at Hewlett Packard Enterprise (https://github.com/CrayLabs/SmartSim)
- Infrastructure library (IL):
  - Provides API to start, stop and monitor HPC applications from Python
  - Deploy a distributed in-memory database
- SmartRedis client library:
  - Provides clients that can connect to the database from Fortran, C, C++ and Python
  - Client API enables data transfer to/from database, commands to act on data stored in database (e.g., delete), and invoking JIT-traced Python and ML runtimes

Online Learning with SmartSim

Features of Infrastructure

- SmartSim/Redis API and database offer:
  - Asynchronous communication between data producer and consumer
  - Ability to store and communicate training data and useful metadata for duration of HPC job
  - Relatively easy implementation in many existing simulation and ML codes
  - Launching multiple data consumers simultaneously querying the same data (e.g., train multiple models simultaneously)
  - Model inference directly on database (can use GPU backends) using JIT-traced Python and ML runtimes