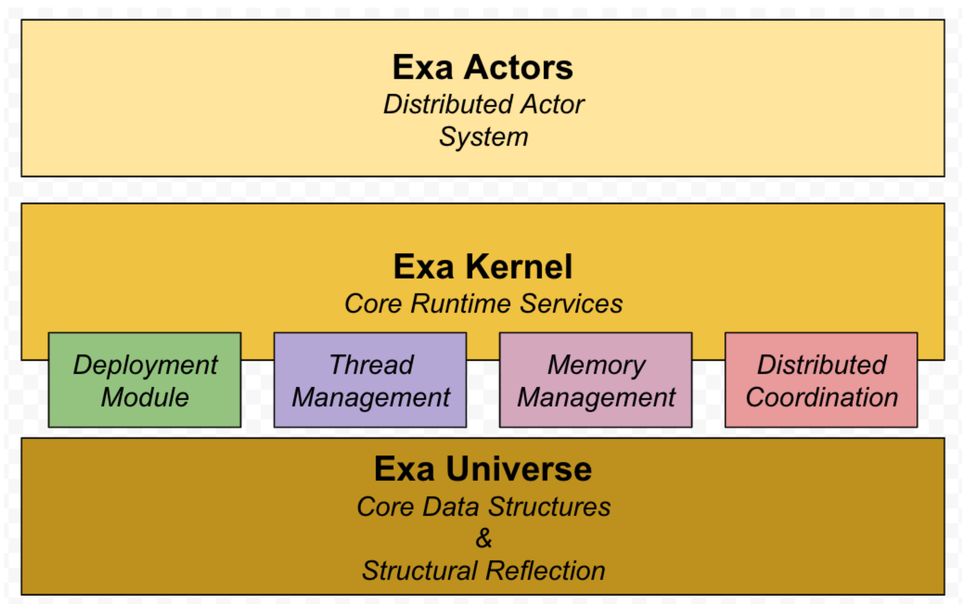
EXA - A distributed computation environment

Tobias Herb, Daniel Thilo Schroeder

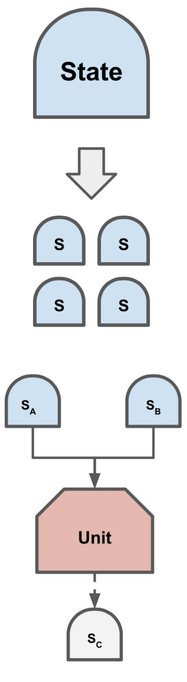
Introduction

The EXA-Platform is an open programming environment for distributed computational tasks. It provides first-class concepts to model concurrent and parallel computation based on the Java platform. The Design follows the philosophy of multi-resolution which means strongly tiered building blocks which allow to solve problems on the appropriate level of abstraction.

Architecture



Concepts



Computation-Unit

- Encapsulates the computation.
- Operate on state container(s).
- Unit views local context.
- Provided as user defined-function (UDF).
- State update/transformation logic.
- Units are scheduled to state containers.
- Unit receives the flow of control from the state container (IoC).
- State container views global context.

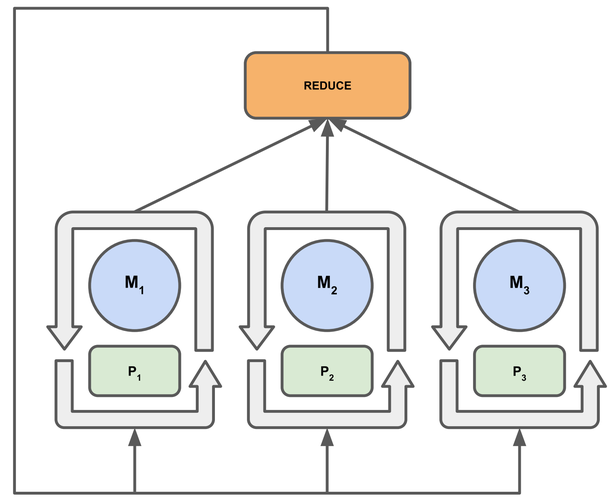
State-Container

- Made explicit as first-class citizen.
- Modeled as Distributed Abstract Data Type
- Storing data/state in an organized way with specific distribution & access rules.
- Parallelization & distribution behavior: singleton, replicated, partitioned (range-, hash-, ...)
- Access & consistency behavior parallel updates, state reconcilation

Application - Distributed Machine Learning

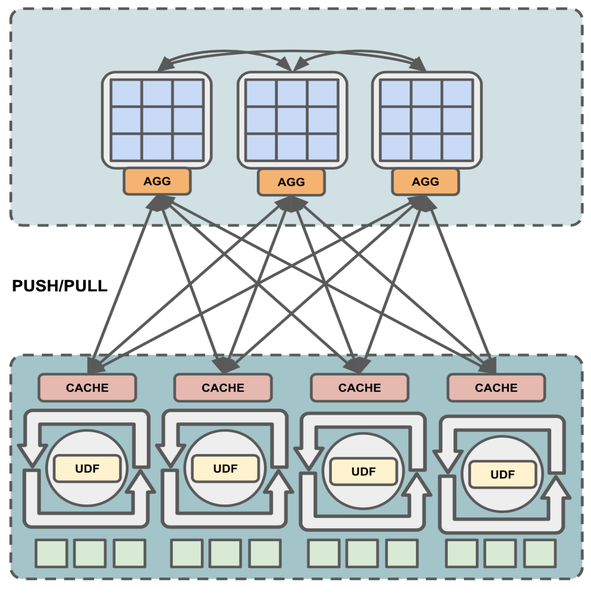
Dataflow Parallelization

Independent (M1 - M3) model instances are trained on different data partitions (P1 - P3). After each training iteration a global reduce operation combines all the independent models. This simple approach is supported by major platforms but the Reduce leads to communication bursts. Further the execution is bulk synchronous parallel (BSP)



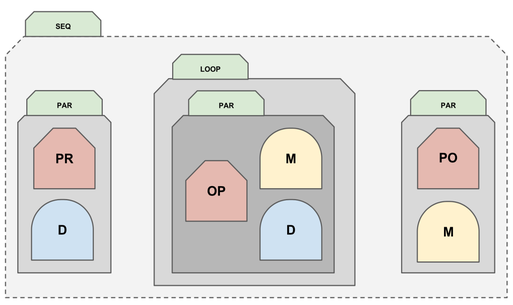
Parameter Server

- Built for very-large industry size machine learning problems.
- Parameter storage and parameter update mechanism is broken out as a primary component!
- Allows efficient distribution of state over many machines. Globally shared parameters are represented as dense or sparse vectors and matrices.
- Workers store training data and run ML algorithm.
- Asynchronous data communication between workers and the server nodes.
- Communicating over fine-grained push/pull interface.

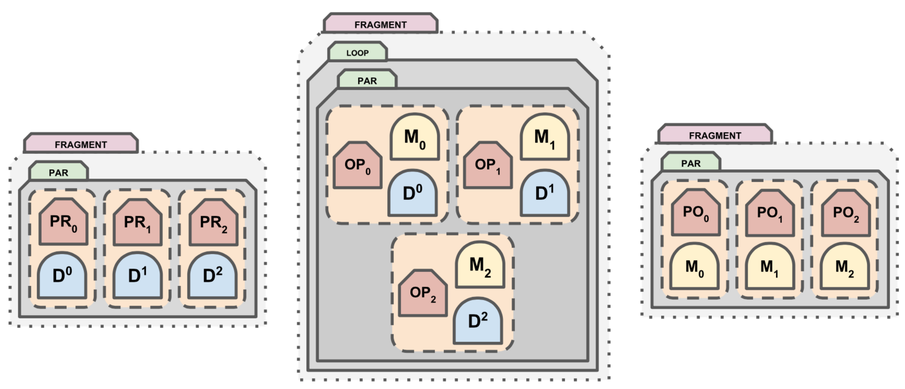


In Action

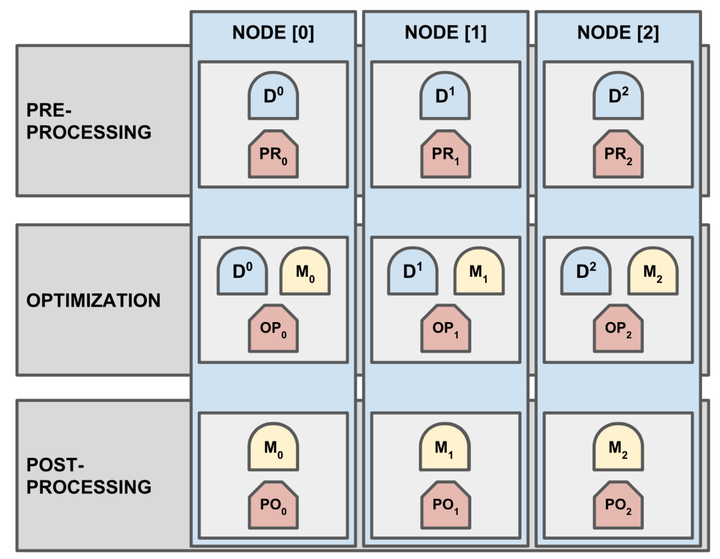
Logical Representation



Physical Representation



State/Unit Scheduling



A typical parallelization pipeline for the machine learning use case. A program in the logical representation consisting of State-Container, Computation-Units and the control flow primitive Loop, Seqential and Parallel is continuously compiled into a physical representation. All of this takes place just in time. After compilation, the physical representations are continuously distributed over the cluster and executed on the individual nodes.