STRADS-AP: Simplifying Distributed Machine Learning Programming without Introducing a New Programming Model

Jin Kyu Kim¹, Abutalib Aghayev¹, Garth A. Gibson¹,²,³, Eric P. Xing¹,⁴

¹Carnegie Mellon University, ²Vector Institute, ³University of Toronto, ⁴Petuum Inc.
Distributed Machine Learning (ML) development flow

ML researchers → New Model → Sequential Algorithm → Sequential Program in Python, Java, R, Matlab, C/C++ → Gap → Distributed Program on high-level frameworks such as Spark/Hadoop/GraphLab/PS

The cost of using high-level frameworks
- Mold a sequential ML program to a framework-specific programming model
- Change data structure design and computation routine
- Often deliver suboptimal performance

STRADS-AP aims to simplify conversion of sequential ML program into a distributed ML program almost mechanically
Easy Conversion of Seq. ML into Dist. ML

// part1: pretraining
declare data structure D for input
declare data structure P for parameters

// part2: training
for(iter=0; iter<MAX; iter++)
  for(i=0; i<N; i++){
    read a part of input D and parameter P
    write to a part of parameters P
  }

Structure pattern of targeting ML programs

Sequential data structures (i.e. map, vector) for input data and model parameters
→ Challenge1: scalability limit

Source of parallelism: repetitive, static control flow, reorderable
→ Challenge2: data dependencies among loop bodies
Easy Conversion of Seq. ML into Dist. ML

// part1: pretraining
declare data structure D for input
declare data structure P for parameters

// part2: training
for (iter=0; iter<MAX; iter++)
    for (i=0; i<N; i++) {
        read a part of input D and parameter P
        write to a part of parameters P
    }

Structure pattern of targeting ML programs

distribute declared data structures D

distribute declare data structures P

for (iter=0; iter<MAX; iter++)
    ParallelFor(N, [D, P](int i){
        read a part of input D and parameter P
        write to a part of parameters P
    })

STRADS-AP Distributed Program

Sequential data structures (i.e. map, vector) for input data
and model parameters
→ Challenge1: scalability limit

Source of parallelism: repetitive, static control flow, reorderable
→ Challenge2: data dependencies among loop bodies

Solution1: distributed data structures (i.e. dmap, dvector)
→ allows index based random R/W access from any node

Solution2: parallel loops (Sync/AsyncFor)
→ parallelize loop bodies with different consistency level
STRADS-AP Workflow

```cpp
stradsap::dvector<T1> D;
stradsap::dmap<T2> P,Q;
float alpha(0.1);
for(i=0; i<N; i++){
    stradsap::parallel_for
    (N, [i, alpha, &D, &P, &Q](int j){
        - optimization routine
        - read i,j, alpha, elements of D
        - read/write elements of P,Q
    }, stradsap::ConsistencyModel);
    alpha *= 0.99;
}
```

fill the lack of C++ language’s reflection capability

Native compiler

Binary code

Add Language specific augmentations

STRADS-AP runtime

STRADS-AP debugging

STRADS-AP code
Presentation schedule

STRADS-AP presentation at 3pm Wed July 10 in Track II