Functional Federated Learning in Erlang

Gregor Ulm, Emil Gustavsson, Mats Jirstrand
Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Gothenburg, Sweden

FRAUNHOFER CHALMERS
RESEARCH CENTRE FOR INDUSTRIAL MATHEMATICS

Problem

There is no limit to the amount of data a connected vehicle produces every day. Even a very conservative estimate of a few dozen gigabytes per vehicle per hour for a small fleet of cars shows that it is infeasible to perform centralized data processing. The reason is that it is not efficient to transfer big data via the network. Instead, alternative approaches for distributed data analysis are needed. Federated Learning is one such example of decentralized data processing.

Contribution

- Creating an Erlang-based prototype of a framework for distributed data processing
- Highlighting the feasibility of a purely functional style for the aforementioned framework
- Creating a purely functional implementation of an artificial neural network in Erlang
- Conducting a performance comparison of a Federated Learning implementation exclusively in Erlang with one in which client processes depend on computations being executed in C
- Demonstrating the suitability of Erlang for decentralized machine learning tasks via performance comparisons

Solution

Federation Learning The Federated Learning algorithm performs the following steps:

1. Select a subset c of the set of clients C, i.e. edge devices
2. Send the current model from the server to each client $\in c$
3. For each client $\in c$, update the provided model based on local data by performing iterations of a machine learning algorithm
4. For each client $\in c$, send the updated model to the server
5. Aggregate all received local models, for instance by averaging, in order to construct a new global model

Implementation We implemented a general framework in Erlang for distributing machine learning tasks and added an implementation of an artificial neural network to it. We created four prototypes, using the following approaches: (1) concurrent Erlang, (2) concurrent Erlang plus NIFs, (3) distributed Erlang, (4) distributed Erlang and C nodes.
As the source code on the right shows, message passing is a very good fit for distributed machine learning in general, and Federated Learning in particular. The code is discussed further in our paper. Here, we draw attention to how elegantly the general framework can be implemented in Erlang.

The client process receives the current global model from the server, trains it with locally available data, and sends the updated local model to the server:

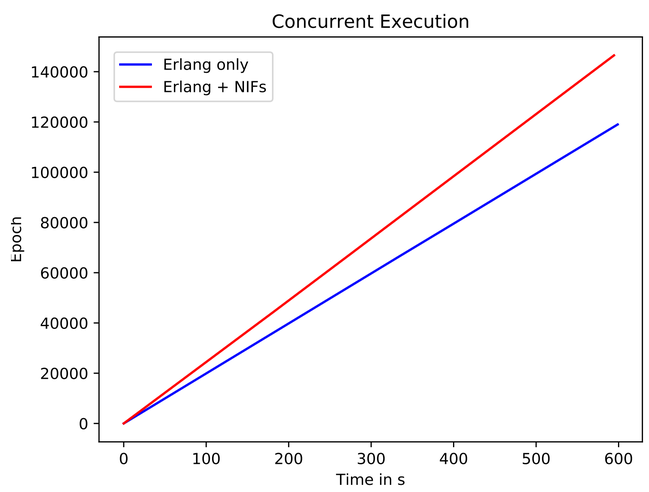
```
client() ->

  receive
  {assignment, Model, Server_Pid} ->

    Val = train(Model), % computes w_j
    Server_Pid ! { update, self(), Val },
    client()

  end.
```

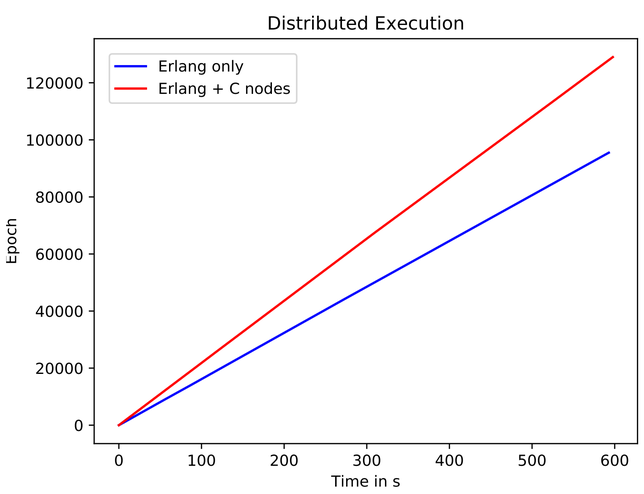
The server sends each client in a subset of all clients an assignment containing the current global model, and afterwards awaits their respective updated models. Subsequently, a new global model is computed, for instance by averaging.

```
server(Client_Pids, Model) ->

  Subset = select_subset(Client_Pids, []),

  % send assignment
  lists:map(
    fun(X) ->
      X ! {assignment, Model, self()} end,
    Subset),

  % receive values
  Vals = [ receive
           {update, Pid, Val} -> Val
           end
         || Pid <- Subset ],

  % update model, i.e. compute global 'w'
  Model_ = update_model(
             Model, Vals, length(Client_Pids)),
  % Note: it is a simplification to
  % use the number of clients

  server(Client_Pids, Model_).
```

Results

The key takeaway is that Erlang is fairly competitive with C for our use case. The plots below show that the performance penalty of Erlang amounts to only a modest constant factor.





Future Work

Building on this research, future avenues to pursue are, first, to scale the performance comparision and determine at which point a collection of edge devices outperforms a powerful server. Second, one could implement other machine learning algorithms and assess how well they map to distributed computing. Third, there is the issue of scaling when using very large data sets, which one may want to look into. Lastly, we could like to point out that the results of this paper led to currently ongoing work on designing and implementing a real-world framework for distributed data analysis in our research lab, written in Erlang and Python. We intend to share details in a future paper.

Acknowledgements

Our interns Adrian Nilsson and Simon Smith assisted with the implementation. This research was supported by the project Onboard/Offboard Distributed Data Analytics (OODIDA) in the funding program FFI: Strategic Vehicle Research and Innovation (DNR 2016-04260), which is administered by VINNOVA, the Swedish Government Agency for Innovation Systems.

Source Code

Source code artifacts accompanying this paper are available at https://gitlab.com/fraunhofer_chalmers_centre/federated_learning_erlang, released under the MIT license.

References

[1] McMahan, H Brendan et al. (2016). Communication-efficient learning of deep networks from decentralized data, arXiv preprint arXiv:1602.05629.

[2] Yu, Tina et al. (1998). PolyGP: A polymorphic genetic programming system in Haskell, Genetic Programming 98.

[3] Bauer, Harald et al. (2012). The supercomputer in your pocket. McKinsey on Semiconductors, pp. 14–27.

[4] Chen, Deyan et al. (2012). Data security and privacy protection issues in cloud computing. Proc International Conference on Computer Science and Electronics Engineering (ICCSEE), pp. 647–651.

[5] Tene, Omer et al. (2011). Privacy in the age of big data: a time for big decisions. Stan. L. Rev. Online 64, pp. 63–69.

[6] Evans-Pughe, Christine (2005). The connected car. IEE Review 51 (1), pp. 42–46.

[7] Lee, Junghoon et al. (2010). A Roadside Unit Placement Scheme for Vehicular Telematics Networks. Advances in Computer Science and Information Technology, pp. 196-202.

[8] Orr, Genevieve B et al. (2003). Neural networks: tricks of the trade (Springer).

[9] Gybenko, G (1989). Approximation by superposition of sigmoidal functions. Mathematics of Control, Signals and Systems 2 (4), pp. 303–314.

[10] Hornik, Kurt (1991). Approximation capabilities of multilayer feedforward networks. Neural networks 4 (2), pp. 251–257.

[11] Srihari, Sargur N et al. (1997). Integration of hand-written address interpretation technology into the united states postal service remote computer reader system. Proc Fourth International Conference on Document Analysis and Recognition 2, pp. 892–896.

[12] Silver, David et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), pp. 484–489.

[13] Allison, Lloyd (2005). Models for machine learning and data mining in functional programming. Journal of Functional Programming 15 (1), pp. 15–32.

[14] Fisher, RA et al. (1936). Iris data set, UC Irvine Machine Learning Repository.

[15] LeCun, Yann et al. (2010). MNIST handwritten digit database, AT&T Labs.