Software Architektur

JAN HOFMEIER, KRISTINA ALBRECHT

2018-04-20

Contents

List of Tables

List of Figures

List of Listings

1 TensorFlow-Architektur

Was ist TensorFlow

TensorFlow ist eine Machine Learning Bibliothek, welche 2015 von Google als Open-Source veröffentlicht wurde. Der Schwerpunkt der Bibliothek liegt auf neuronalen Netzen und tiefen neuronalen Netzen, die in der letzen Zeit eine umfangreiche Anwendung in vielen Bereichen der künstlichen Intelligenz wie Bilderkennung und Spracheanalyse gefunden haben.

TensorFlow wurde als Nachfolger einer anderen Bibliothek für Machine Learning, **DistBelief**, entwickelt. DistBelief wurde im Rahmen des Google Brain Projekts im Jahr 2011 entwickelt, um die Nutzung von hochskalierbaren tiefen neuronalen Netzen (DNN) zu erforschen. Die Bibliothek wurde unter anderem für unsupervised Lernen, Bild- und Spracherkennung und auch bei der Evaluation von Spielzügen im Brettspiel Go eingesetzt. (TensorFlow 2018)

Trotz des erfolgreichen Einsatzes hatte DistBelief einige Einschränkungen:

- die NN-Schichten mussten (im Gegensatz zum genutzten Python-Interface) aus Effizienz-Gründen mit C++ definiert werden.
- die Gradientenfunktion zur Minimierung des Fehlers erforderte eine Anpassung der Implementierung des integrierten Parameter-Servers.
- nur vorwärtsgerichtete Algorithmen möglich rekurrente KNN oder Reinforcement Learning möglich.
- wurde nurfür die Anwendung auf großen Clustern von Multi-Core-CPU-Servern entwickelt, keine Unterstützung von GPUs oder anderen Prozessoren.

Aus den Erfahrungen mit DistBelief wurde gelernt und diese Erfahrungen wurden bei der Entwicklung von TensorFlow berücksichtigt. Interessant ist, dass DistBelief zwar als Prototyp für TensorFlow genommen wurde, an dem verschiedene Funktionalitäten ausprobiert und getestet wurden, allerdings wurde TensorFlow komplett neu entwicklelt. Das ist ein Beispiel dafür, dass Prototype sehr praktisch sind, dass es jedoch auch wichtig ist, deren Vor- und Nachteile zu bewerten und im Laufe der Entwicklung Prototype auch zu verwerfen.

Im Weiteren werden die Anforderungen verschiedener Benutzergruppen beschrieben und die Architektur der Bibliothek ausführlich erläutert.

Anforderungsanalyse

TensorFlow wird von verschiedenen **Benutzergruppen** verwendet:

- Forscher, Studenten, Wissenschaftler
- Architekten und Software Engineure
- Software Entwickler
- Hardware Hersteller.

Die Bibliothek wird vor allem zur Entwicklung der Anwendungen mit AI-Funktionalitäten eingesetzt. Zusätzlich wir sie zur Forschungszwecken im Bereich Machine Learning zur Entwicklung der neuen Algorithmen und Modelle verwendet. Außerdem gehören auch Hardware-Hersteller zu einer der Benutzergruppen von TensorFlow, die ihre Produkte (zB. CPUs, GPUs etc.) für Machine Learning-Zwecke optimieren wollen.

Aus diesen Anwendungsfällen lassen sich die **Anforderungen** an die Bibliothek ableiten, die nach FURPS-Merkmalen aufgeteilt werden können. FURPS steht für:

- Functionality (Funktionalität)
- Usability (Benutzbarkeit)
- Reliability (Zuverlässigkeit)
- Performance (Effizienz)
- Supportability (Änderbarkeit, Wartbarkeit)

ID	Kurzbeschreibung	Anforderung
F1{#af1}	ML und DL Funktionalitäten	Da Machine Learning auf mathematischen Berechnungen beruht, muss TensorFlow Vektor- bzw Matrizen-Operationen und andere Rechenoperationen aus Linearen Algebra und Statistik bereitstellen. Viele Trainingsalgorithmen benötigen Gradienten, deshalb muss TensorFlow diese selbst bestimmen können.
U1{#au1}	Protoyping	TensorFlow muss eine Möglichkeit zum schnellen Definieren und Testen von Modellen bereitstellen.
U2{#au2}	Produktiver Einsatz	TensorFlow muss für den produktiven Einsatz (vor allem Inference) geeignet sein.
P1{#ap1}	Performance	Da Maschine Learning durch Rechenleistung limitiert ist, muss TensorFlow die verfügbaren Ressourcen effizent nutzen.
P2{#ap}	Skalierbarkeit	TensroFlow muss mit sehr großen Datenmengen umgehen können.
S1{#as1}	Portabilität	Die Bibliothek muss auf verschiedene Systeme portierbar sein und unterschiedliche Acceleratoren (GPU, TPU) untersützten.
R1{#ar}	Recovery	Der Trainingsfortschritt soll nach einem Absturtz wiederherstellbar sein

Einflussfaktoren

Folgende organisatorische, technische und Produkt-bezogene Faktoren können einen Einfluss auf die Architektur-Entscheidung beeinflusst haben.

Faktor-Index	Beschreibung
O1	Community, welche verwendet und Contributions macht
O2	Entwickler und interne Nutzer haben Kenntnisse in C++und Python
T1	Es werden neue Acceleratoren entwickelt
T2	Die Rechenleistung einer Maschine ist begrenzt, weshalb horizontal skaliert werden muss

Faktor-Index	Beschreibung
T3	Auf Clustern und im Produktiven Einsatz kommt vor allem Linux zum Einsatz, die User verwenden oft OS X oder Windows. Es muss aber auch auf noch unbekannten Betriebsystem einsetzbar sein.
P1	Neue Bibliotheken können integriert werden
P2	Modell kann sehr komplex werden und viele Daten involvieren

Architekturentwurf

Im Weiteren werden 4+1 Sichten der TensorFlow-Architektur dargestellt: Szenarien, Kontext-Sicht, Verhaltenssicht, Struktursicht, Abbildungssicht.

Szenarien

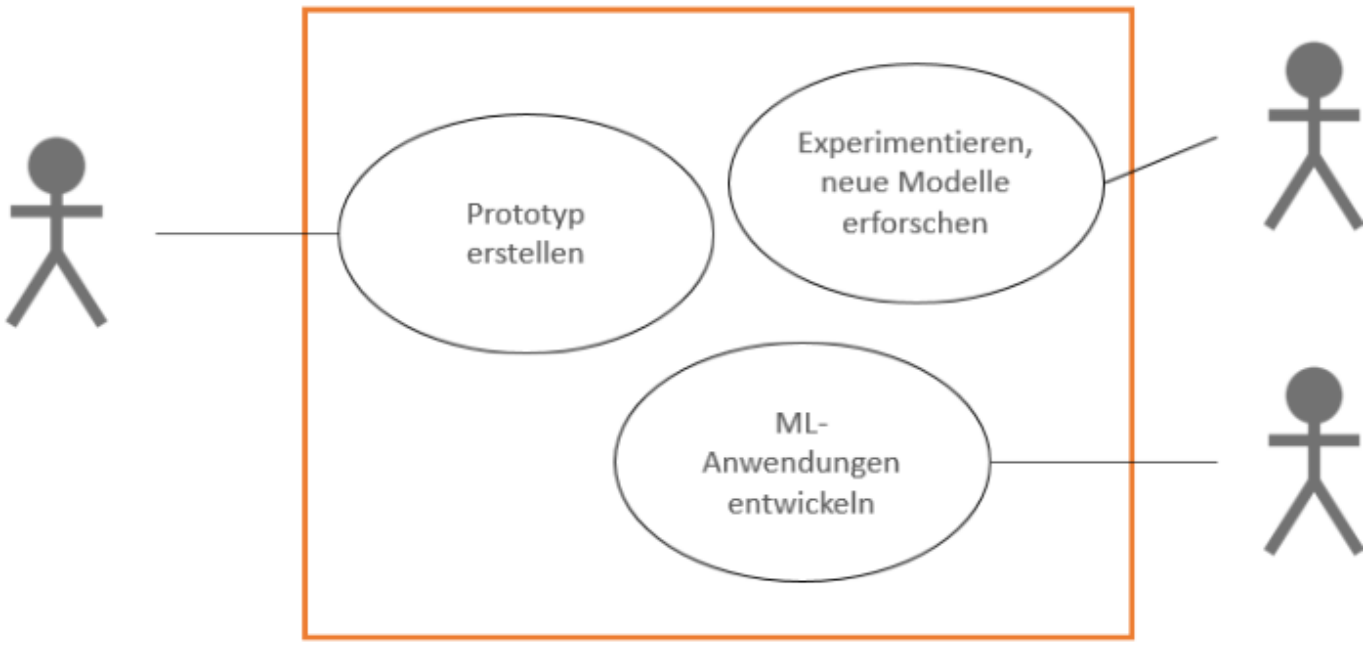

Figure 1.1: Szenarien

Kontext-Sicht

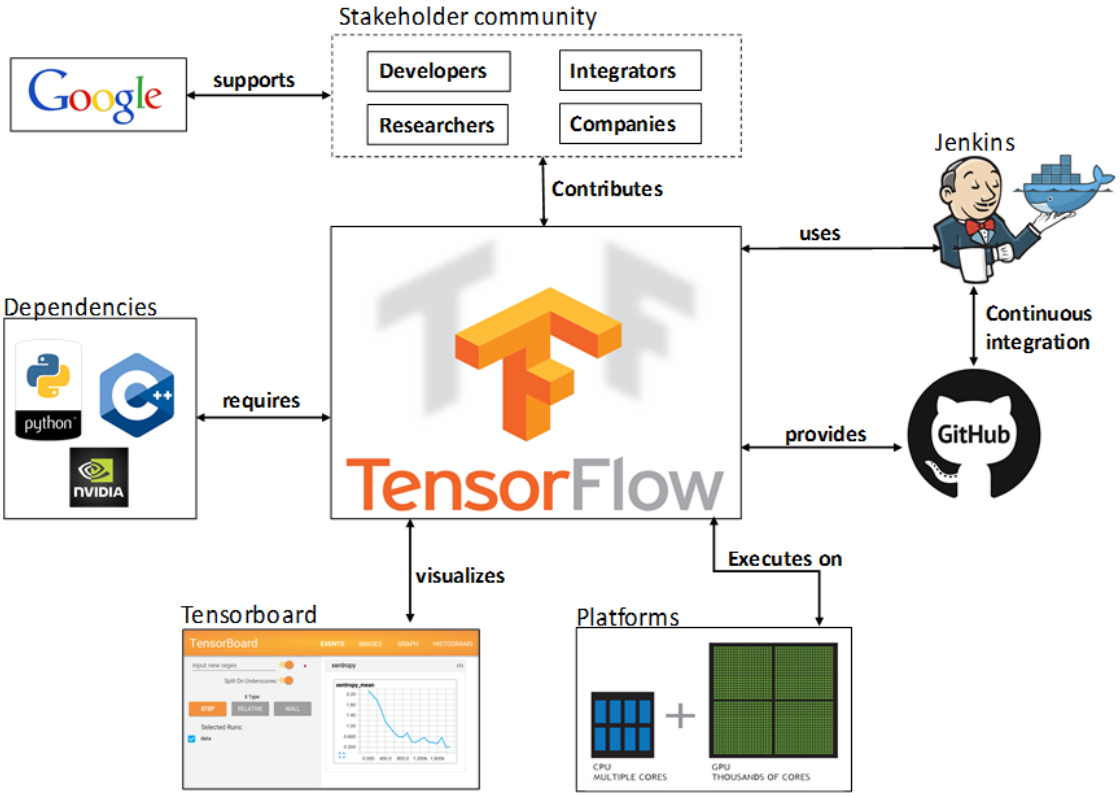

Figure 1.2: Dependencies

###Verhaltenssicht (Architekturbausteine)

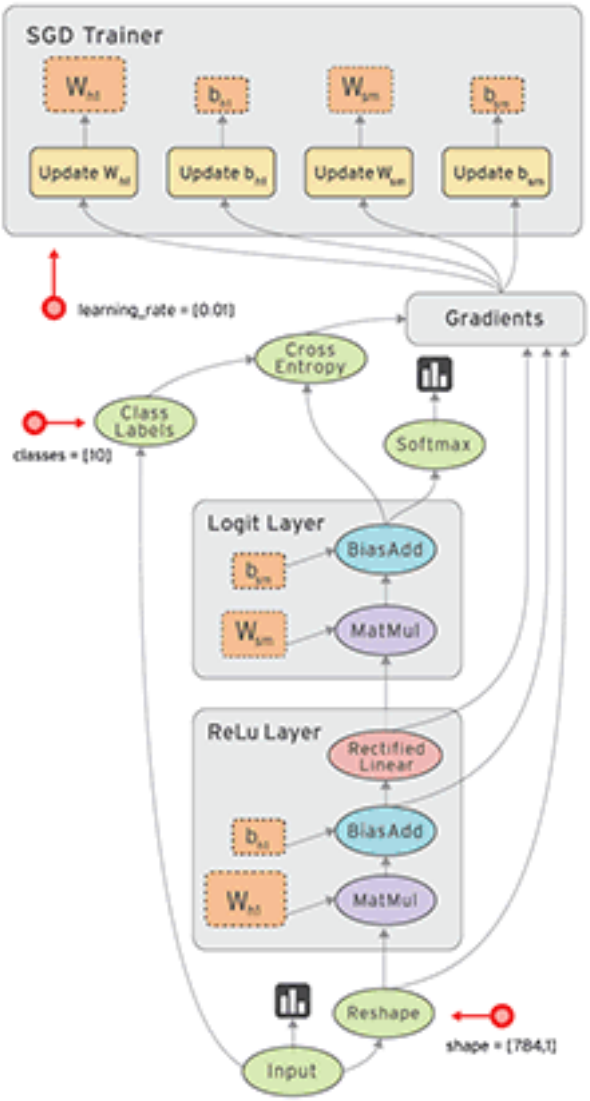

Figure 1.3: Datenfluss in TensorFlow

In TensorFlow werden Berechnungen als Graphen dargestellt. Diese Graphen lassen sich innerhalb einer Session ausführen. In den Graphen gibt drei Grundarten von Knoten: Operationen, Placeholder und Variablen. Placeholder, werden beim ausführen durch einen konkreten Wert ersetzt. Sie stellen Eingabewerte (hier der Input Node) dar. Variablen halten Werte, die in der Session gespeichtert werden und verändert werden können. Hier sind das beispielsweise die Weights W und Biaseses b. Operationen sind hier Beispielsweise Matrix Multiplikation $MatMul$ oder $BiasAdd$. Die ein und Ausgabewerte der Nodes sind jeweils Tensoren, welche entlang der Kanten "fließen". Ein Subgraph lässt sich zu einer Komponente zusammenfassen, hier sind das $ReLuLayer$, $LogitLayer$ und $SDGTrainer$.

Der Graph stellt ein Neuronales Netzwerk mit zwei Layern da, welches Classification macht und über Statistic Gradient Descent trainiert wird. Die Trainingsdaten werden im $Input$ Placeholder eingegeben und und in Labels und die Feature Matrix aufgeteilt. Die Feature Matrix wird stellt die Eingabedaten für den Forwardpass durch das

Neuronale Netz genutzt. In vectorisierter Form lässt sich die Multiplikation jedes Features mit dem entsprechenden Gewicht für alle Samples als Matrixmultiplikation ausdrücken. Der erste Layer nutzt ReLu als activation Function, deren Ausgabe Matrix die Eingabe für den zweiten Layer ist.
Für das Training werden die Labels One Hot Kodiert $ClassLabels$ und über die $CrossEntropie$ Cost Function mit der Ausgabe des letzten Layers Netzes verglichen. Der Trainingsprozess besteht darin, dass mit Hilfe von Gradient Descent die Variablen die Cost Function für die Trainingsdaten minimiert wird, indem die W und b in kleinen Schritten angepasst werden. Da ein Minimum der Cost Function gesucht wird. Anhand der Partiellen Ableitungen (Gradients) nach den Variablen ergibt sich die Richtung in der Sich das Minimum befindet und entsprechend werden die Variablen geupdated in den $Update$ angepasst. Das geschieht in mehren Iterationen, sodass man entsprechend Gradients sich einem (lokalen) Minimum annähert.
Bei Inference wird die Ausgabe des letzten Layers durch die Softmax Funktion in den Interval [0,1] gebracht, was der Wahrscheinlichkeit entspricht, mit der das Netz ein Sample einer Klasse zuordnet. Das Netz nutzt dazu die in der Session gespeicherten trainierten Weights und Biases. Der Gleiche Graph kann auch in unterschiedlichen Session für unterschiedliche Daten Trainiert werden.

###Struktursicht

Tensorflow ist in mehren Schichten Organisiert. Diese reichen von einer Gerätespezifischen Schicht (unten) bis zu High Level Training und Inference Bibliotheken (oben). Der Tensorflow Core stellt seine Funktionalität über eine Low-Level C API bereit. Diese Low-Level API wird durch High-Level APIs für verschiedene Client Sprachen, wie Python, C++, Java und Go gekpaselt. Unter Verwendung der Sprachspezifischen APIs gibt es High-Level Bibliotheken für Training und Inference.

Diese Schichtenarchitektur erlaubt ein hohes Maß and Flexibilität und Portabilität. So kann auf dem Device Layer beispielsweise Unterstützung für Acceleratoren wie GPUs und TPUs hinzugefügt werden, Kernelimplementationen hinzugefügt oder ersetzt werden und eine zusätztliche Schnittstelle für eine andere Sprache hinzugefügt werden.

High Level Python Bibliotheken erlauben das schnelle prototyping und Training von Modellen und Algorithmen. Die C++ und Java Clients dagegen untersützt das einbinden trainierter Modelle in Produktivsysteme, auf welchen hohe Inference Performance gefordert ist.

Der Distribution master ist dafür verantwortlich die Berechnungen auf mehre Dataflow Executer aufzuteilen. Dabei werden transparent Send und Recive Nodes an den Kanten eingefügt, an denen der Graph partitioniert wird. Die Dataflow Executer können dabei auf mehre Madschienen verteilt werden, die passende Kommunikation wird vom Network Layer abstrahiert.

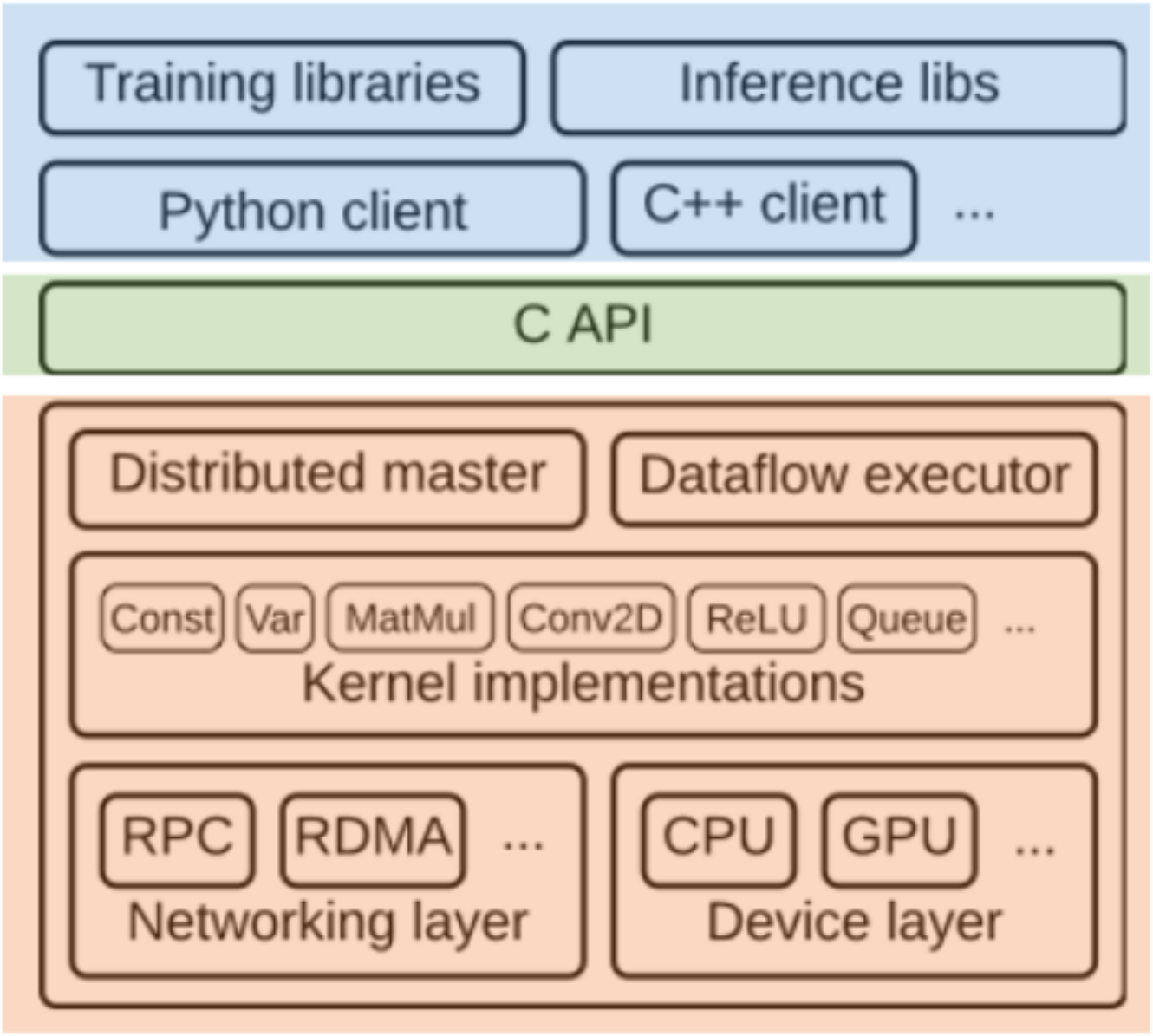

Figure 1.4: Schichtenarchitektur von TensorFlow

Kernels

- Kernels sind C++ Implementierungen von Operationen, die speziell für die Ausführung auf einer bestimmten Recheneinheit wie CPU oder GPU entwickelt wurden. Da sie zu nativiem Maschinencode kompiliert werden und sehr Hardware nah geschrieben sind, können sie diese optimal ausnutzten und sehr hohe Performance erzielen.

- Die TensorFlow enthält mehrere solche eingebaute Operationen/Kernels. Beispiele dafür sind:

Kategorie	Beispiele
Elementweise mathematische Operationen	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal
Array-Operationen	Concat, Slice, Split, Constant, Rank, Shape, Shuffle

Kategorie	Beispiele
Matrix-Operationen	MatMul, MatrixInverse, MatrixDeterminant
Variablen und Zuweisungsoperationen	Variable, Assign, AssignAdd
Elemente von Neuronalen Netzen	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool
Checkpoint-Operations	Save, Restore
Queue und Synchronisations- operationen	Enqueue, Dequeue, MutexAcquire, MutexRelease
Flusskontroll-Operationen	Merge, Switch, Enter, Leave, NextIteration

- TensorFlow untersützt das einbinden eigener Kernel.

Source-Code-Hierarchie

–TensorFlow™ 's root directory at GitHub is organized in five main subdirectories: google, tensorflow, third-party, tools and util/python. Additionally, the root directory provides information on how to contribute to the project, and other relevant documents. In figure 3, the source code hierarchy is illustrated.

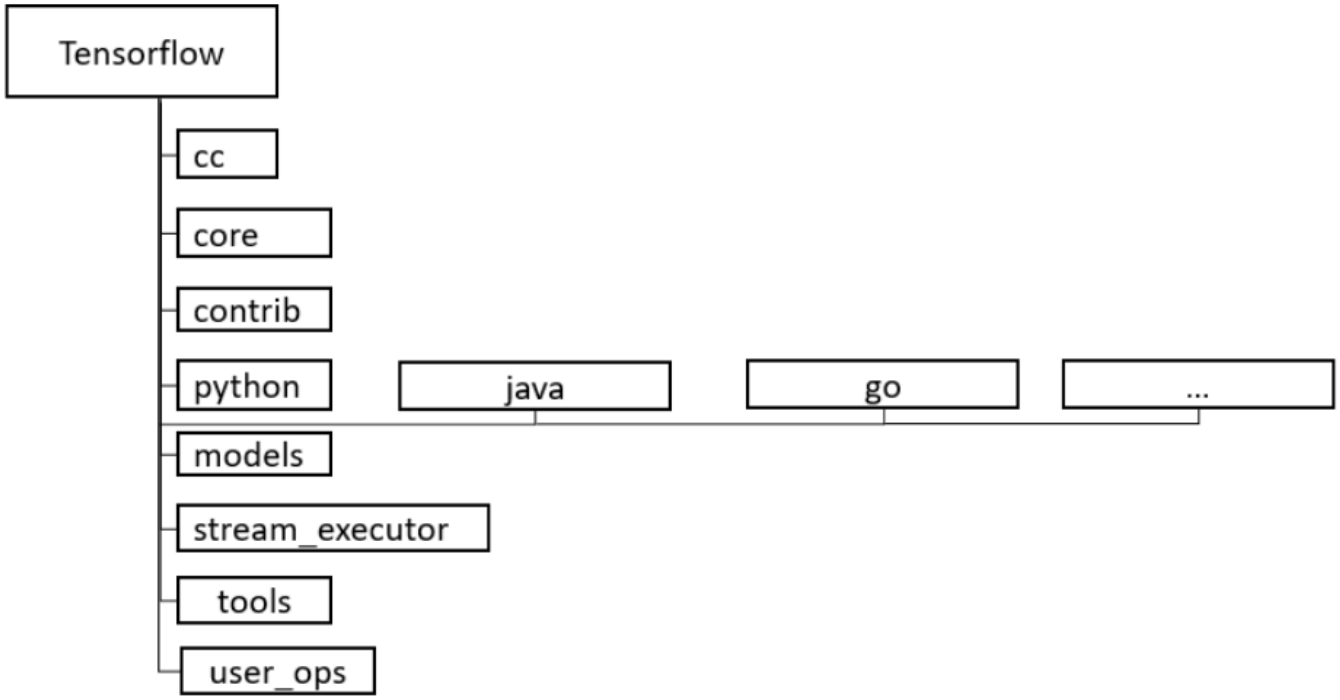

Figure 1.5: Source-Code-Hierarchie

Abblidungssicht (Ausführungseinheiten)

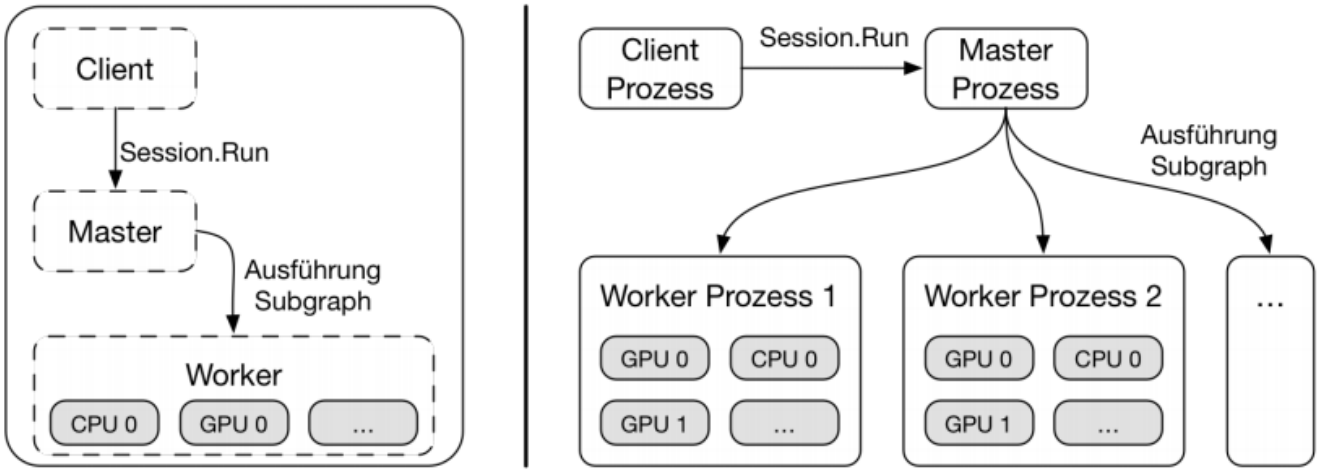

Figure 1.6: Single-Machine vs. verteilte Ausführung

Fazit

Portabilität => Device Layer, Kernel implementations

Skalierbarkeit => verteilt, mehrere Worker (Distributed Master, Dataflow Executor, Worker Services)

Perfomence => C++ Client, Kernel implementations

–Runs on CPUs, GPUs, desktop, server, or mobile computing platforms. That make it very suitable in several fields of application, for instance medical, finance, consumer electronic, etc.

Flexibilität: => High & Low Level APIs

Vielfältige Einsatzmöglichkeit: Forschung, Prototypen und Produktion => Python Client, High Level Libraries
–TensorFlow™ allows industrial researchers a faster product prototyping. It also provides academic researchers with a development framework and a community to discuss and support novel applications.

–Provides tools to assemble graphs for expressing diverse machine learning models. New perations can be written in Python and low-level data operators are implemented using in C++.
Portabilität

TensorFlow. 2018. “TensorFlow Architecture.” https://www.tensorflow.org/extend/architecture.