인공지능응용 논문리뷰

Created By	
Status	
Type	

Towards Federated Learning at Scale: System Design

Abstract

FL은 탈중앙화된 데이터에서 모델 트레이닝을 가능하게 하는 분산 머신러닝 방법이다. 저자는 텐서플로우로 모바일 환경에서 작동하는 scalable한 시스템을 설계하였다. 이 논문에서는 high-level 설계, 몇몇 도전과제와 해결책, 아직 남아있는 문제들과 미래 방향을 제시한다.

Protocol

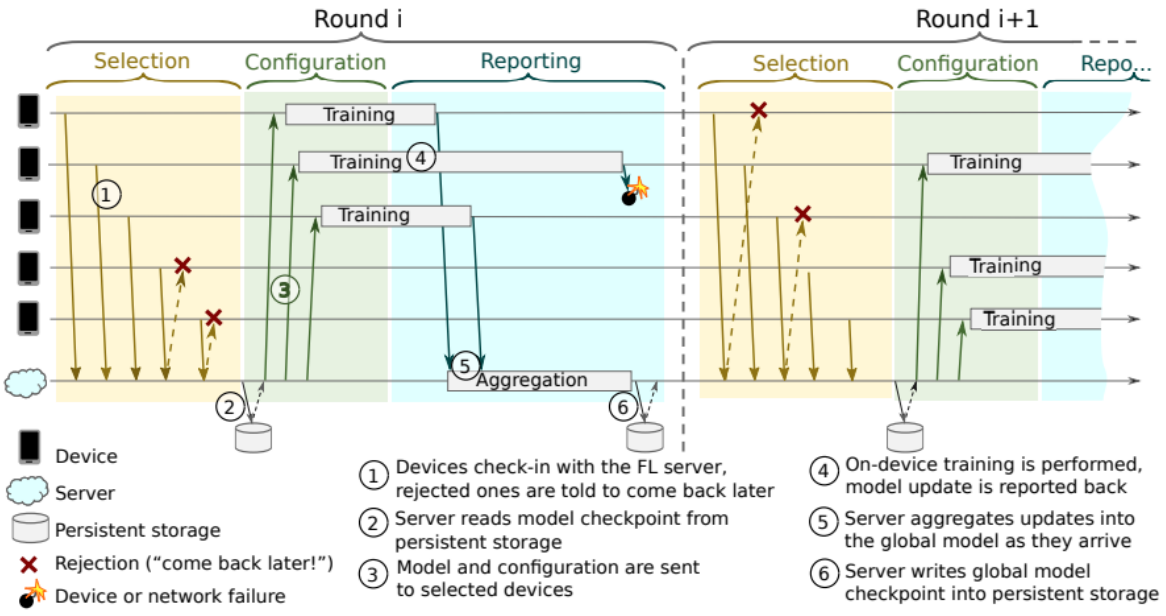

Figure 1: Federated Learning Protocol

프로토콜에서의 참가자는 *devices*와 분산형 클라우드 기반의 *FL server* 이다. device는 서버에게 FL task에 대한 준비가 되었다고 알린다. FL task는 특정한 연산인데, 예를 들면 주어진 하이퍼파라미터로의 모델 훈련, 로컬 데이터로 훈련된 모델에 대한 평가 등이 있다.

서버는 선택된 디바이스에 어떤 연산을 수행하라고 전한다. 한번 라운드가 시작되면 서버는 다음으로, 각 참가자에 현재의 global model parameters와 *FL checkpoint* (텐서플로우 세션 정보 등)를 보낸다. 각 참가자는 로컬 연산을 수행하고 FL checkpoint라는 형태로 서버에 업데이트를 보낸다. 서버는 이러한 업데이트를 글로벌 상태로 합치며 계속 진행된다.

Phases

- **Selection:** 디바이스는 양방향 통신을 시작하며 서버에 체크인한다. 서버는 목적에 따라 현재 연결되어있는 클라이언트 중 일부 클라이언트를 선택한다. 선택되지 않은 디바이스에 서버는 나중에 다시 연결하라고 응답을 보낸다.

- **Configuration:** 서버는 aggregation mechanism으로 설정된다. FL plan과 FL checkpoint를 글로벌 모델과 함께 각 디바이스에 보낸다.
- **Reporting:** 서버는 참가한 디바이스가 결과를 업데이트하길 기다린다. 업데이트를 받는대로 서버는 FedAvg를 사용해 aggregation하고, 디바이스에게 언제 재연결할지 보낸다. 충분한 디바이스가 제 시간에 도착한 경우 라운드는 성공적으로 끝난 것이며 서버는 자신의 글로벌 모델을 업데이트 할 것이다. 반대의 경우 라운드는 무시된다.

Pace Steering

Pace Steering은 디바이스의 연결 패턴을 제한하는 흐름 제어 메커니즘이다. 이 것은 FL server가 FL 참여자를 줄이거나 매우 크게 늘릴 수 있도록 한다.

Pace Steering은 서버가 디바이스에게 재연결할 최적의 시간범위를 제안하며, 디바이스가 이를 지키고자 하는 것이다.

FL 참여자가 적을 경우, pace steering은 적정한 숫자의 디바이스를 서버에 동시에 연결하는 데에 쓰이는데, 이는 task의 진행속도 및 secure aggregation protocol에도 중요하다.

FL 참여자가 많을 경우, pace steering은 디바이스의 체크인 타임을 랜덤화할 수 있다. 이는 “thundering herd”* 문제를 피하도록 하며 디바이스가 필요한만큼 자주 연결해 모든 FL task를 수행하게 한다.

> 💡 Thundering Herd Problem은 많은 수의 클라이언트가 동일한 요청을 서버에 하게 될 때 Cache Miss가 발생하게 되는 현상.

또한 pace steering은 시간대를 임의로 조정하거나 피크시간대의 불필요한 활동을 FL 성능의 저하 없이 줄일 수 있다.

Device

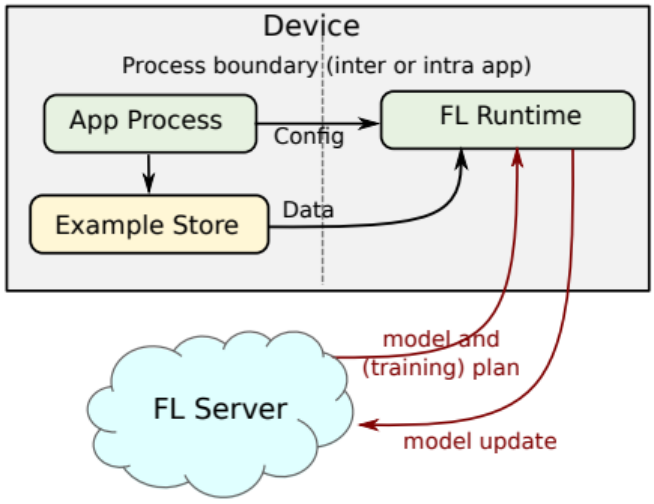

Figure 2: Device Architecture

Device의 첫번째 역할은 훈련과 검증을 위해 로컬 데이터를 유지하는 것이다. App은 API를 통해 데이터를 FL 런타임에서 활용 가능한 형태로 만들어야 한다. App에서는 전체 저장 용량을 제한하고 오래된 데이터를 삭제하는 것을 권장한다.

Programmatic Configuration

UX, data usage, battery life에 부정적인 영향을 주지 않도록 FL runtime job을 스케줄링 해야한다. FL 런타임은 오직 폰이 idle 상태, 충전중, WiFi 등의 네트워크 상태일 때만 job scheduler에 요청한다.

Job Invocation

FL 런타임은 FL server에 연결해 주어진 참가자에 대한 task 수행이 가능하다고 알려줘야한다. 서버는 어떤 FL task 수행이 가능한지 결정한다.

Task Execution

디바이스가 선택되면, FL 런타임은 FL plan을 받아 app의 데이터를 질의하고 모델과 메트릭에 따른 연산을 진행한다.

Reporting

FL plan 실행이 끝난 후, FL 런타인은 서버에 보고하며 임시 저장된 자원들을 지운다.

Multi-Tenancy

한 서비스로 수많은 사용자의 학습 등 활동을 대응할 수 있다.

Server

서버는 수십개부터 수억개의 디바이스에 대응할 수 있어야 한다. 그리고 수집되고 통신하는 업데이트의 사이즈는 각 라운드에 kilobytes 부터 수십 megabytes까지 가능하다.

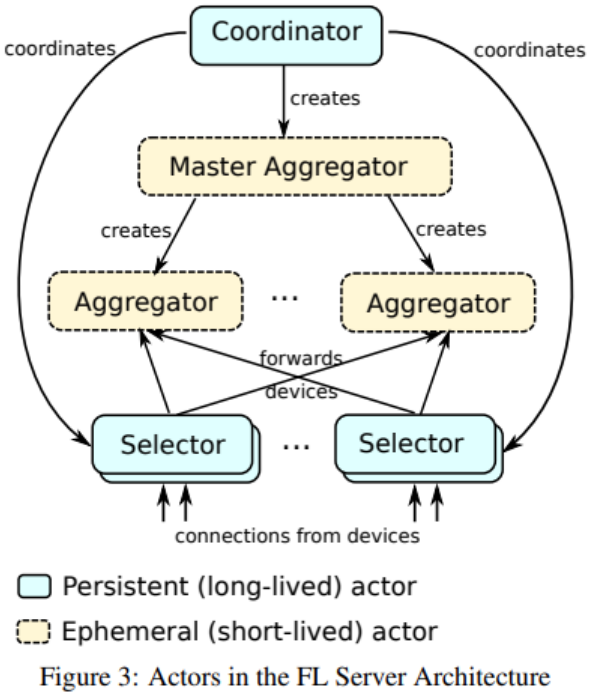

Figure 3: Actors in the FL Server Architecture

Coordinators

global synchronization과 라운드 진행을 가능하게 한다. 여러 Coordinators가 있는데, 각각은 FL 디바이스의 수를 책임진다. 하나의 Coordinator는 자신의 주소와 FL population을 등록한다. 각 Selector에 얼마나 많은 디바이스가 연결되어 있는지 정보를 받고, 얼만큼의 디바이스가 선택될지 지정한다. Coordinator는 Master Aggregator를 생성하여 각 FL task의 라운드를 관리하도록 한다.

Selectors

디바이스의 연결을 수락하고 포워딩하는 역할이다. 주기적으로 Coordinator로부터 각 FL population에 얼마나 많은 디바이스가 필요한지 정보를 받아, 각 디바이스를 수락할지 말지 로컬에서 결정한다. Master Aggregator와 set of Aggregators가 생성된 이후에는, Coordinator가 Selectors에게 연결된 디바이스를 Aggregators에 포워딩해주라고 명령한다. 그렇게 하면 Coordinator가 얼마나 많은 디바이스가 사용 가능한지에 관계없이 효과적으로 디바이스를 FL task에 할당할 수 있다. 이러한 접근은 Selectors가 분산되어 있으며 디바이스에 가깝고, Coordinator와는 제한된 통신을 하도록 한다.

Master Aggregators

각 FL task의 라운드를 관리한다. device와 update size를 스케일하기 위해 동적으로 Aggregators를 생성한다.