# Przetwarzanie danych z użyciem Ray

[https://docs.ray.io](https://docs.ray.io/)

Autorzy:

Monika Etrych

Sebastian Hec

#### 1. Cel projektu

Technologia **Ray** jest wykorzystywana do rozproszonego przetwarzania danych i obliczeń równoległych. Jest to platforma, która umożliwia tworzenie i skalowanie aplikacji opartych na sztucznej inteligencji, uczeniu maszynowym i innych zadaniach wymagających intensywnych obliczeń. Ray pozwala na łatwe rozdzielanie zadań obliczeniowych na wiele węzłów w klastrze, co umożliwia szybsze wykonywanie obliczeń oraz skalowanie aplikacji na dużą skalę.

Celem projektu jest zaprezentowanie wybranych możliwości technologii Ray, szczególnie jej wykorzystania do rozproszonego przetwarzania zadań, zarządzania zadaniami równoległymi oraz w kontekście uczenia maszynowego.

#### 2. Do czego służy wybrana technologia?

Ray to framework do rozproszonego obliczania, który obsługuje różne modele programowania, takie jak obliczenia równoległe, obliczenia rozproszone i uczenie maszynowe. Może być używany w takich obszarach jak:

* **Rozproszone obliczenia**: Ray umożliwia rozproszenie zadań obliczeniowych na wiele maszyn, co pozwala na równoległe wykonywanie skomplikowanych zadań.
* **Zarządzanie zadaniami asynchronicznymi**: Dzięki wbudowanej obsłudze zadań asynchronicznych, Ray pozwala na łatwe zarządzanie i harmonogramowanie obliczeń, w tym obliczeń równoległych i podzielonych.
* **Uczenie maszynowe i sztuczna inteligencja**: Ray jest wykorzystywany do rozproszonych treningów modeli, wspiera takie biblioteki jak Ray Tune (do optymalizacji hiperparametrów), Ray RLLib (do uczenia wzmocnionego), a także Ray Serve (do wdrażania modeli).
* **Przetwarzanie danych w czasie rzeczywistym**: Ray może obsługiwać rozproszone procesy, które potrzebują przetwarzać dane w czasie rzeczywistym, np. aplikacje streamingowe i analityczne.

#### 3. Wybrane możliwości technologii i przykłady

W ramach projektu skoncentrujemy się na następujących możliwościach Ray:

* **Rozproszone wykonywanie zadań obliczeniowych**: Pokażemy, jak za pomocą Ray można rozdzielać obliczenia na wiele węzłów, co pozwala na równoległe przetwarzanie danych.
* **Zarządzanie zadaniami asynchronicznymi i harmonogramowanie**: Zademonstrujemy, jak Ray może być użyty do uruchamiania zadań asynchronicznych w sposób, który pozwala na optymalne wykorzystanie zasobów obliczeniowych.
* **Uczenie maszynowe w rozproszonym środowisku**: Pokażemy, jak Ray może być użyty do rozproszonego trenowania modeli AI oraz optymalizacji hiperparametrów (Ray Tune).

#### 4. Demo będzie prezentować:

1. **Użycie Ray do rozproszonego treningu modelu AI**: W pierwszej części projektu uruchomimy Ray Cluster na wybranym środowisku (np. lokalny serwer lub chmurowe rozwiązanie). Model AI, np. sieć neuronowa do rekomendacji, będzie trenowany w sposób rozproszony na kilku węzłach.
2. **Optymalizacja hiperparametrów z Ray Tune**: W drugiej części zademonstrujemy, jak Ray Tune pomaga w automatycznej optymalizacji hiperparametrów modelu w trakcie treningu.
3. **Wdrażanie modelu za pomocą Ray Serve**: Na koniec, model AI będzie wdrożony na Ray Serve, umożliwiając generowanie rekomendacji w czasie rzeczywistym.

#### 5. Moduły, z których skorzystamy

* **Ray Core**: Moduł podstawowy do obsługi zadań równoległych i rozproszonych.
* **Ray Tune**: Narzędzie do optymalizacji hiperparametrów modeli uczenia maszynowego.
* **Ray Serve**: Umożliwia łatwe wdrożenie modelu AI jako serwis RESTful.
* **Ray RLLib** (opcjonalnie): Jeśli projekt będzie obejmował uczenie maszynowe wzmocnione.

#### 6. Schemat architektury

[Klient]

V

[API (Ray Serve)]

V

[Ray Cluster (Węzły, Scheduler)]

V

[Model AI (Trening w rozproszonym środowisku)]

V

[Zbiór danych]

#### 7. Przepływ sterowania i przepływ danych

**Przepływ sterowania**:

1. Klient wysyła zapytanie o rekomendację do API (Ray Serve).
2. API przekazuje zapytanie do Ray Serve, który zarządza wnioskami.
3. Ray Serve uruchamia zadanie na jednym z węzłów klastra Ray.
4. Na węzłach Ray obliczane są wyniki (rekomendacje).
5. API zwraca rekomendację do klienta.

**Przepływ danych**:

1. Dane treningowe są ładowane do klastra Ray.
2. Modele AI są trenowane na rozproszonych węzłach.
3. Po zakończeniu treningu, model jest zapisany i może być używany do generowania rekomendacji w czasie rzeczywistym.

#### 8. Planowany test

**Metryki**:

* **Czas wykonania**: Zmierzymy czas wykonania treningu modelu, porównując wydajność przy użyciu klastra Ray i na jednym węźle.
* **Dokładność modelu**: Ocena jakości rekomendacji (np. precision, recall, F1-score).
* **Skalowalność**: Zbadamy, jak system skaluje się przy dodawaniu kolejnych węzłów do klastra Ray.

#### 9. Dataset:

#### 10. FAQ

* **Czy technologia Ray jest darmowa?**

Tak, Ray jest open-source'ową technologią, która jest dostępna za darmo. Projekt jest rozwijany przez społeczność i udostępniany na licencji Apache 2.0.

* **Czym wyróżnia się Ray?**

1. Łatwość w skalowaniu: Ray pozwala na łatwe skalowanie aplikacji w poziomie, rozdzielając obliczenia na wiele maszyn lub węzłów. Umożliwia to płynne zarządzanie dużymi obciążeniami obliczeniowymi w sposób, który jest bardziej efektywny niż tradycyjne podejścia.
2. Optymalizacja dla AI i ML: Ray został zaprojektowany z myślą o wsparciu zaawansowanych technik AI i ML, oferując takie narzędzia jak Ray Tune (do optymalizacji hiperparametrów), Ray RLLib (do uczenia wzmocnionego) oraz Ray Serve (do wdrażania i skalowania modeli). To czyni go bardzo odpowiednim narzędziem dla środowisk, które intensywnie wykorzystują technologię sztucznej inteligencji.

* **Plusy Ray**

Łatwość integracji z popularnymi bibliotekami AI/ML: Ray jest dobrze zintegrowany z popularnymi narzędziami i bibliotekami, takimi jak TensorFlow, PyTorch, Horovod czy Apache Spark. Dzięki temu jest bardzo użyteczny w środowiskach, które wymagają pracy z dużymi zbiorami danych i treningiem modeli AI.

Rozproszona optymalizacja: Dzięki narzędziu Ray Tune, użytkownicy mogą automatycznie optymalizować hiperparametry swoich modeli, co przyspiesza proces eksperymentowania z różnymi ustawieniami algorytmów AI.

Open-source i szeroka społeczność: Jako projekt open-source, Ray cieszy się dużą popularnością i aktywną społecznością, która przyczynia się do jego dalszego rozwoju, udostępniając liczne zasoby, tutoriale i wsparcie.

* **Minusy Ray**

Krzywa uczenia się: Choć Ray jest potężnym narzędziem, może być trudny do opanowania dla osób, które nie mają doświadczenia w pracy z rozproszonymi systemami obliczeniowymi. Wymaga to zazwyczaj dobrej znajomości programowania równoległego oraz rozproszonych zasobów.

Wymaga dużych zasobów: Ray jest zaprojektowany z myślą o dużych obciążeniach obliczeniowych i może wymagać znacznych zasobów, zwłaszcza przy rozproszonej architekturze. Przy małych projektach może to prowadzić do nieefektywnego wykorzystania zasobów.

Brak wbudowanego wsparcia dla wszystkich przypadków użycia: Ray jest bardzo elastyczny, ale może nie mieć gotowych narzędzi do obsługi wszystkich typów aplikacji. W niektórych przypadkach konieczne może być napisanie własnych rozwiązań do obsługi specyficznych problemów.

* **Na co jest nałożony nacisk?**

Skalowalność: Ray został zaprojektowany z myślą o rozproszonych obliczeniach, z naciskiem na możliwość łatwego skalowania aplikacji na wiele węzłów w klastrze. Dzięki temu może obsługiwać zadania obliczeniowe wymagające dużej mocy obliczeniowej i pamięci.

Integracja z AI/ML: Jednym z głównych obszarów, w którym Ray wyróżnia się na tle innych frameworków, jest jego wsparcie dla sztucznej inteligencji i uczenia maszynowego. Dzięki narzędziom takim jak Ray Tune, Ray RLLib i Ray Serve, Ray umożliwia łatwe skalowanie i optymalizację treningu modeli AI, co czyni go bardzo wartościowym w tym obszarze.

Wydajność: Ray stawia na szybkie wykonywanie zadań w sposób równoległy. Jego architektura pozwala na efektywne rozdzielanie zadań i zarządzanie nimi, co przekłada się na większą wydajność w obliczeniach.