

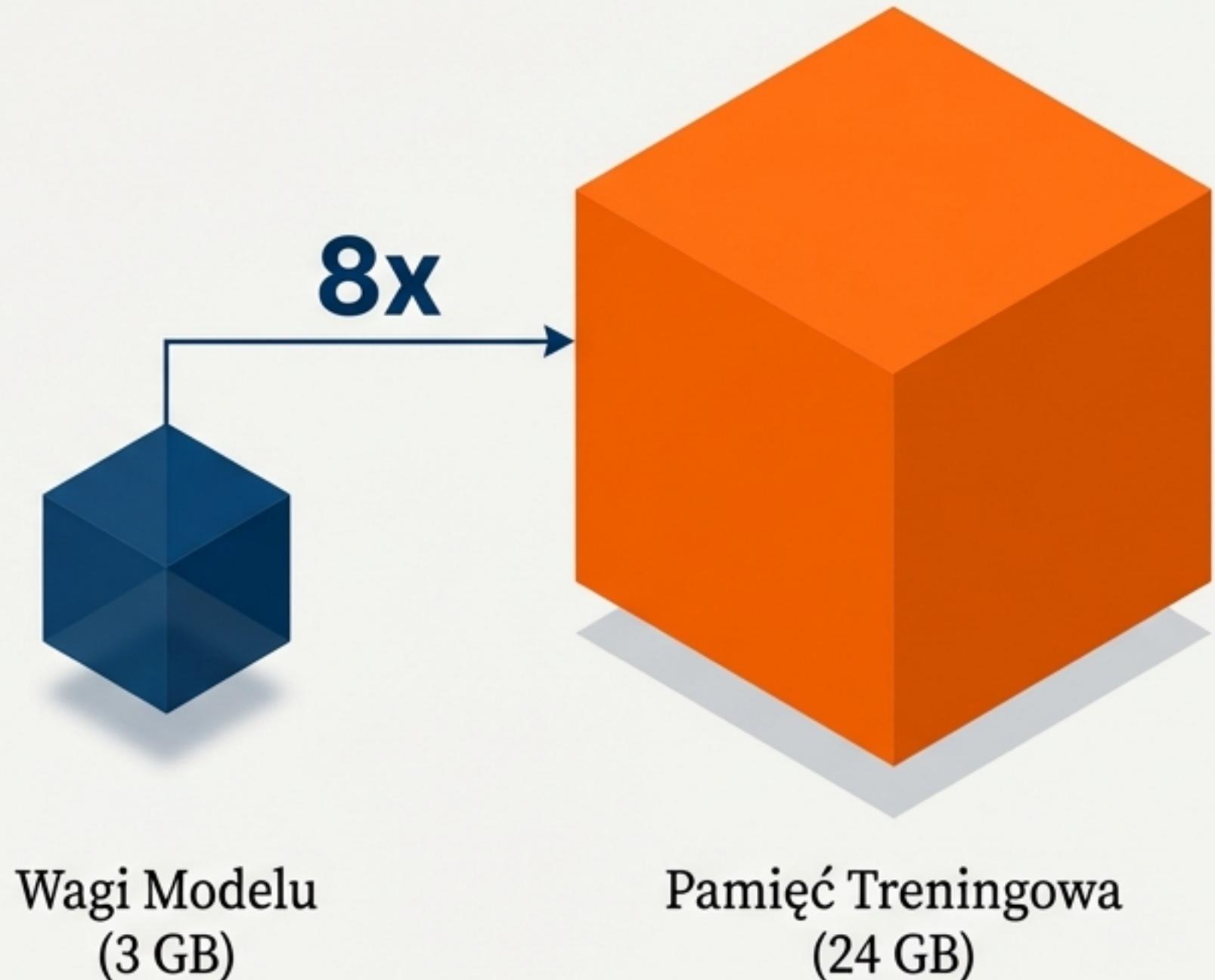
# Problem: Mur Pamięci – Ukryty Wróg Skalowania AI

- Pamięć, a nie moc obliczeniowa, staje się fundamentalną barierą dla gigantycznych modeli. Samo dodanie sprzętu nie rozwiązuje problemu – potrzebujemy innowacji algorytmicznej.

**Studium Przypadku: GPT-2 (1.5 mld parametrów):**

- Wagi modelu (FP16): **3 GB**
- Pamięć do treningu (Adam, mixed precision): **24 GB**
- To ośmiokrotny wzrost zapotrzebowania  
To ośmiokrotny wzrost zapotrzebowania na pamięć w stosunku do samego rozmiaru modelu.

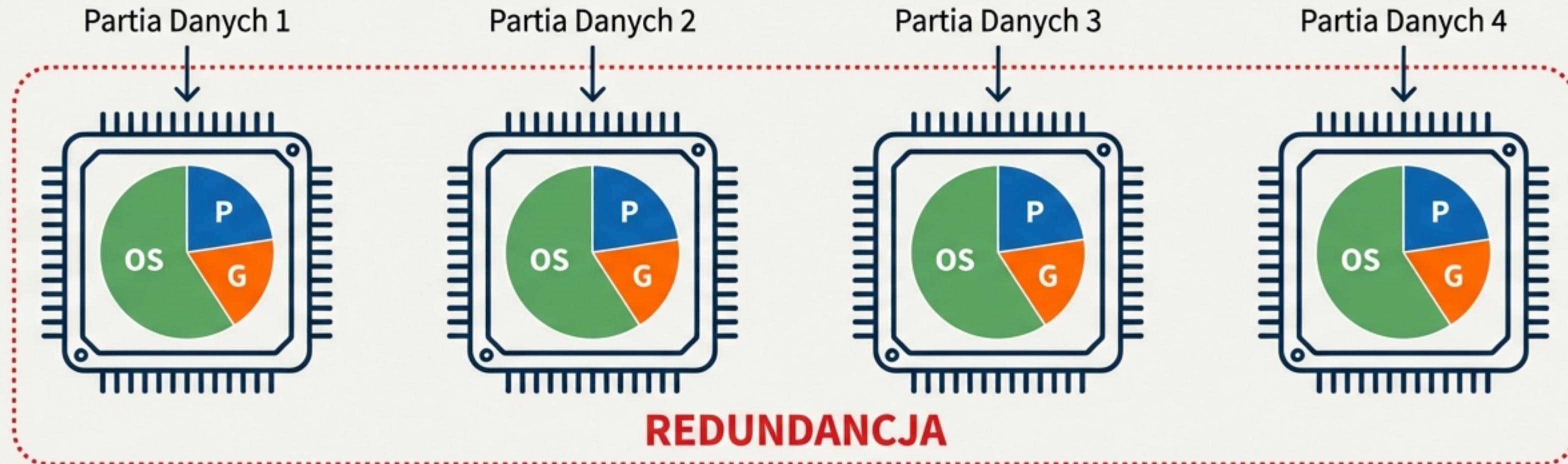
**Cel projektu ZeRO:** Umożliwić trening modeli o bezprecedensowej skali na istniejącym sprzęcie, przełamując tę barierę.



Wagi Modelu  
(3 GB)

Pamięć Treningowa  
(24 GB)

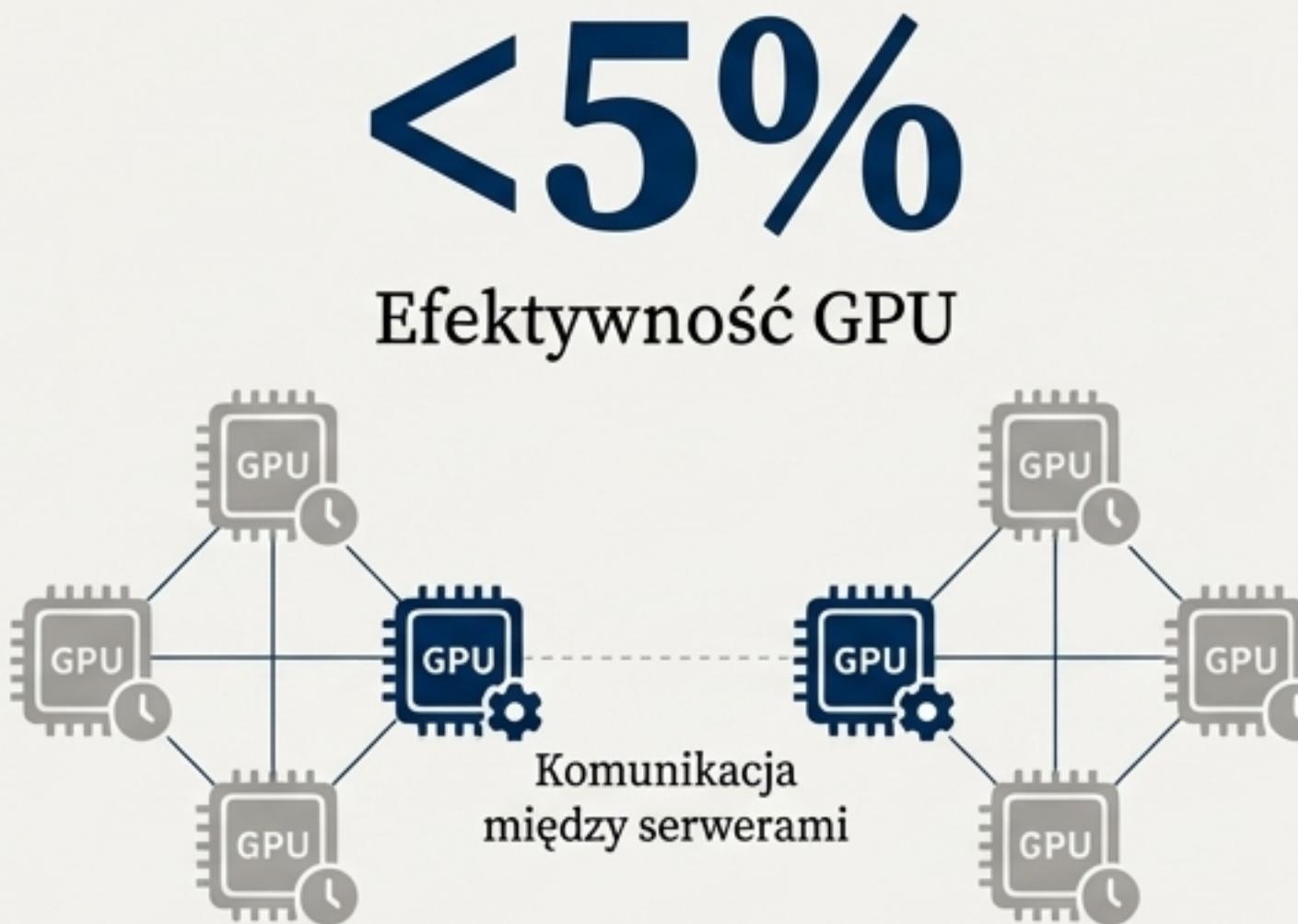
# Stare Podejście #1: Równoległość Danych (DP) – Prostota Okupiona Marnotrawstwem



- **Jak to działa:** Każde GPU otrzymuje pełną kopię całego modelu (parametrów, gradientów, stanów optymalizatora). Następnie każde GPU przetwarza inną partię danych.
- **Problem w skali:** Na 64 GPU mamy 64 identyczne, gigantyczne kopie wszystkich stanów modelu. To definicja redundancji.
- **Analogia:** Wyobraźmy sobie 64 fabryki, z których każda przechowuje kompletne części do budowy całego samochodu, ale jej zadaniem jest tylko montaż jednych drzwi.
- **Wedykt:** Proste w implementacji, ale skrajnie nieefektywne pamięciowo. Trening modeli większych niż ~1.4 mld parametrów staje się niemożliwy na typowych GPU (32GB).

# Stare Podejście #2: Równoległość Modelu (MP) – Komunikacyjny Koszmar

- **Jak to działa:** Model jest dzielony na części (np. warstwy), a każda część jest umieszczona na innym GPU.



- **Obietnica vs. Rzeczywistość:**  
Obietnicą jest efektywność pamięci, ale rzeczywistością jest zapaść wydajności.

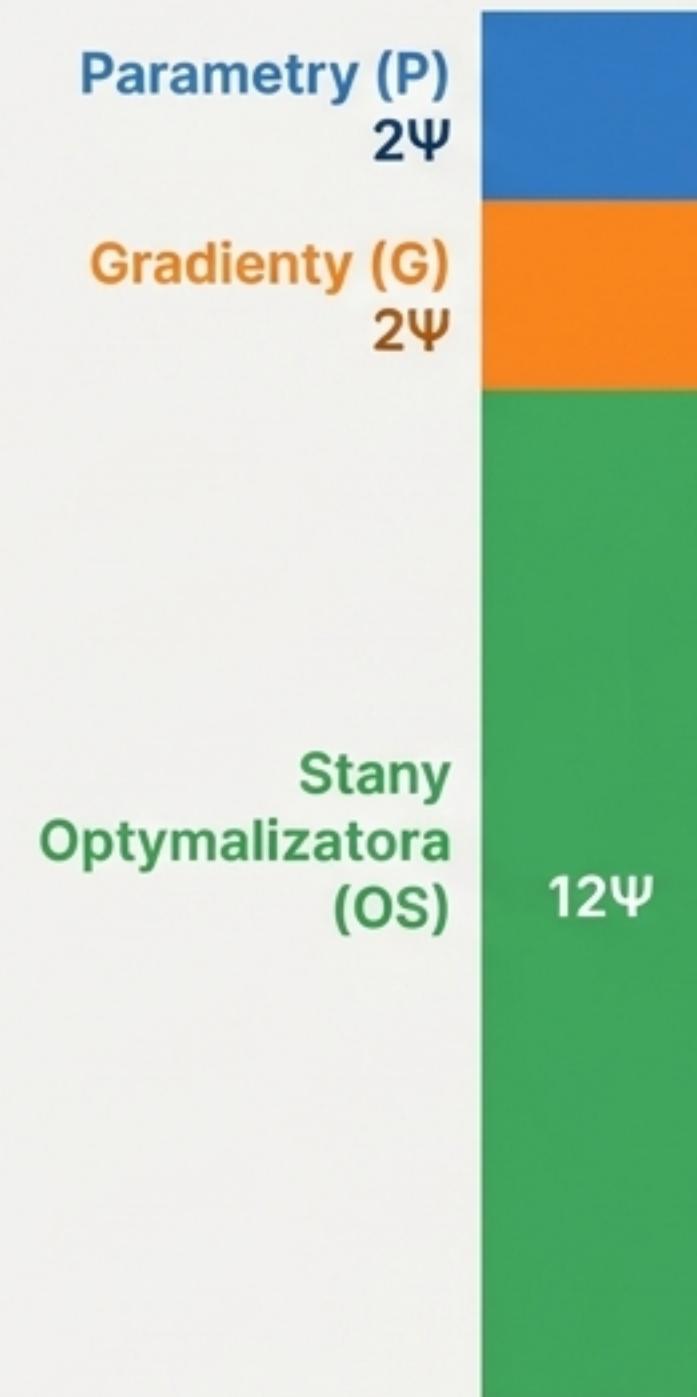
- **Twarde Dane z Testów:** Model 40 mld parametrów trenowany przy użyciu Megatron-LM na dwóch serwerach DGX-2 osiągnął **poniżej 5% teoretycznej wydajności GPU** (zaledwie ~5 TFlops).

- **Wniosek:** 95% czasu GPU bezczynnie czeka na dane z innych GPU. Koszt komunikacji rośnie wykładniczo i staje się wąskim gardłem, zwłaszcza przy komunikacji między serwerami.

# Anatomia Pamięci: Gdzie Dokładnie Znikają Gigabajty?

- **1. Stany Modelu (Główny winowajca):**

- **Parametry (P):** Wagi modelu (np. 2 bajty w FP16). FP16).
- **Gradienty (G):** Pochodne używane do aktualizacji (np. 2 bajty w FP16).
- **Stany Optymalizatora (OS):** W przypadku optymalizatora Adam i treningu mixed-precision, to kopia parametrów w FP32 (4 bajty) oraz pęd (momentum) w FP32 (4 bajty) i wariancja w FP32 (4 bajty).



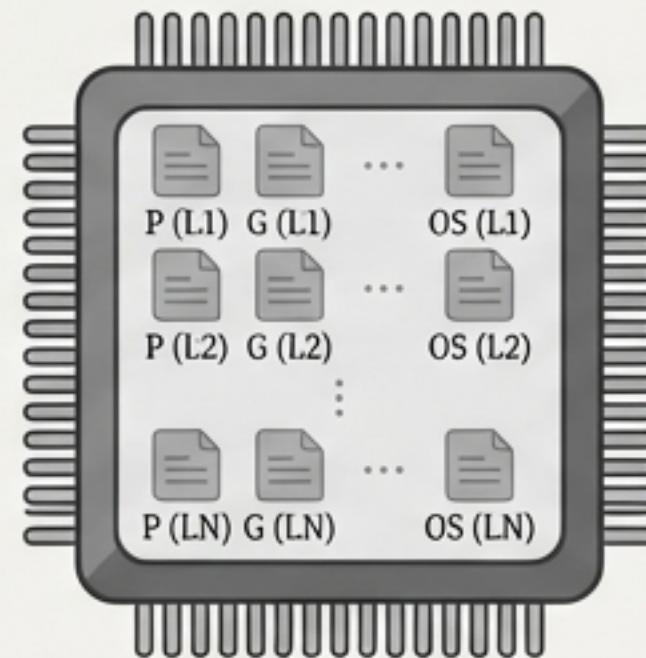
- **2. Stany Resztkowe (Problem poboczny):**  
Aktywacje, bufore tymczasowe, fragmentacja pamięci.

Dlatego model 1 mld parametrów potrzebuje nie 2 GB, a co najmniej 16 GB pamięci.

**Złota Reguła (Adam + Mixed Precision):**  
1 parametr modelu ≈ **16 bajtów** pamięci na stany modelu.

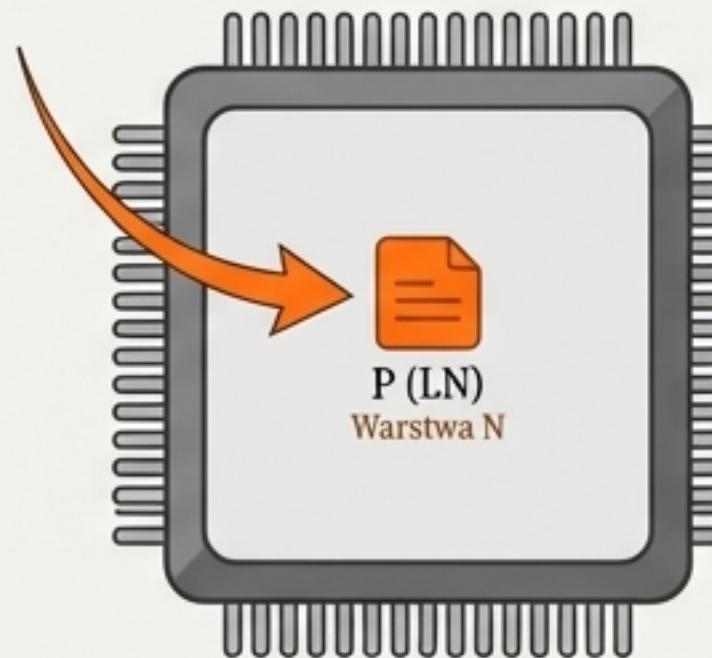
# Przełomowa Idea ZeRO: Koniec z Redundancją, Czas na Dynamiczną Dostawę

Standardowe DP



Każdy “kucharz” (GPU) dostaje na stałe pełną listę wszystkich składników z całej książki kucharskiej.

Podejście ZeRO-DP

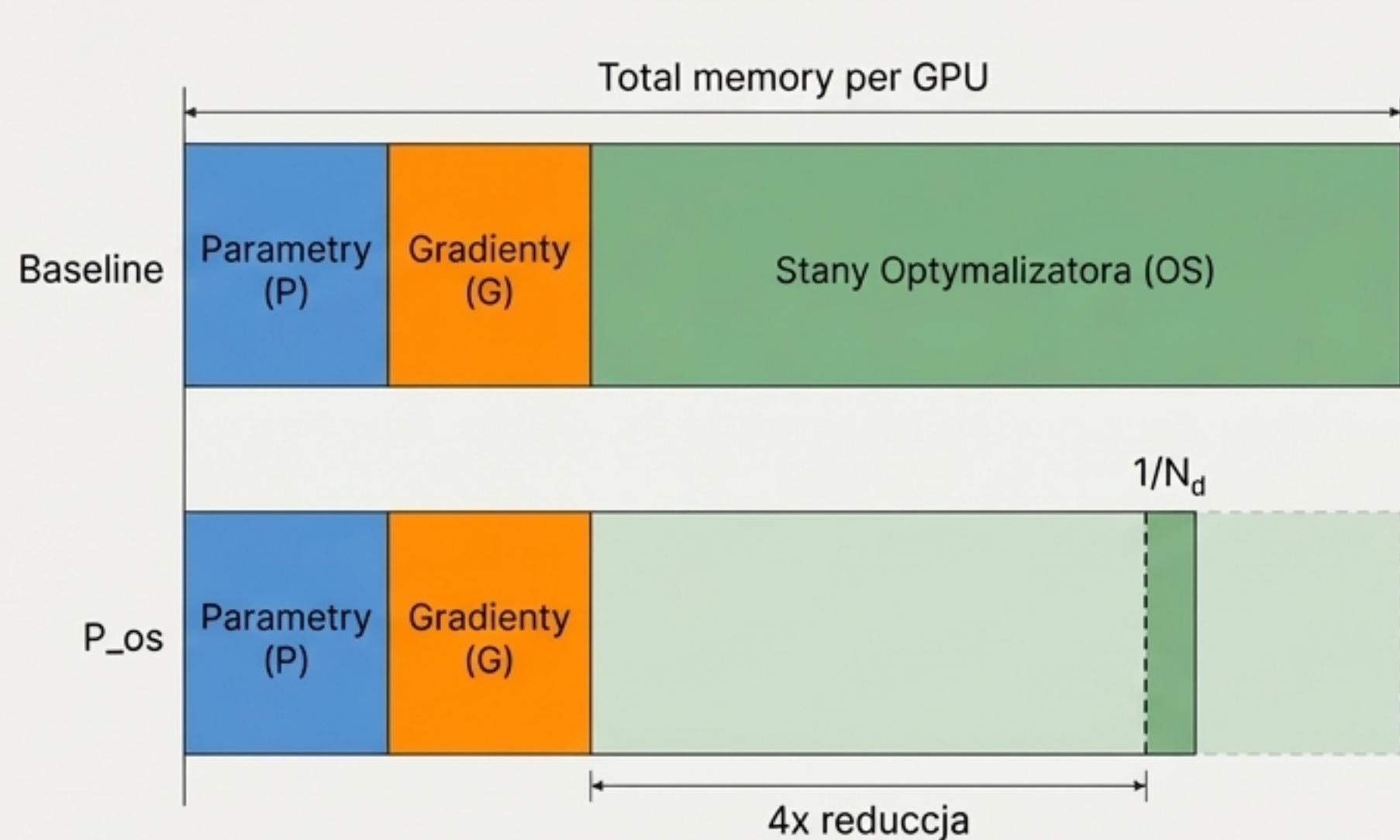


Dostarczaj tylko to, co potrzebne, dokładnie wtedy, gdy jest potrzebne.

- **Kluczowa Obserwacja:** Nie wszystkie stany modelu są potrzebne na każdym GPU przez cały czas.
- **Filozofia:** Eliminacja statycznej replikacji na rzecz intelligentnej, dynamicznej komunikacji. ZeRO-DP wprowadza trzy progresywne etapy optymalizacji.

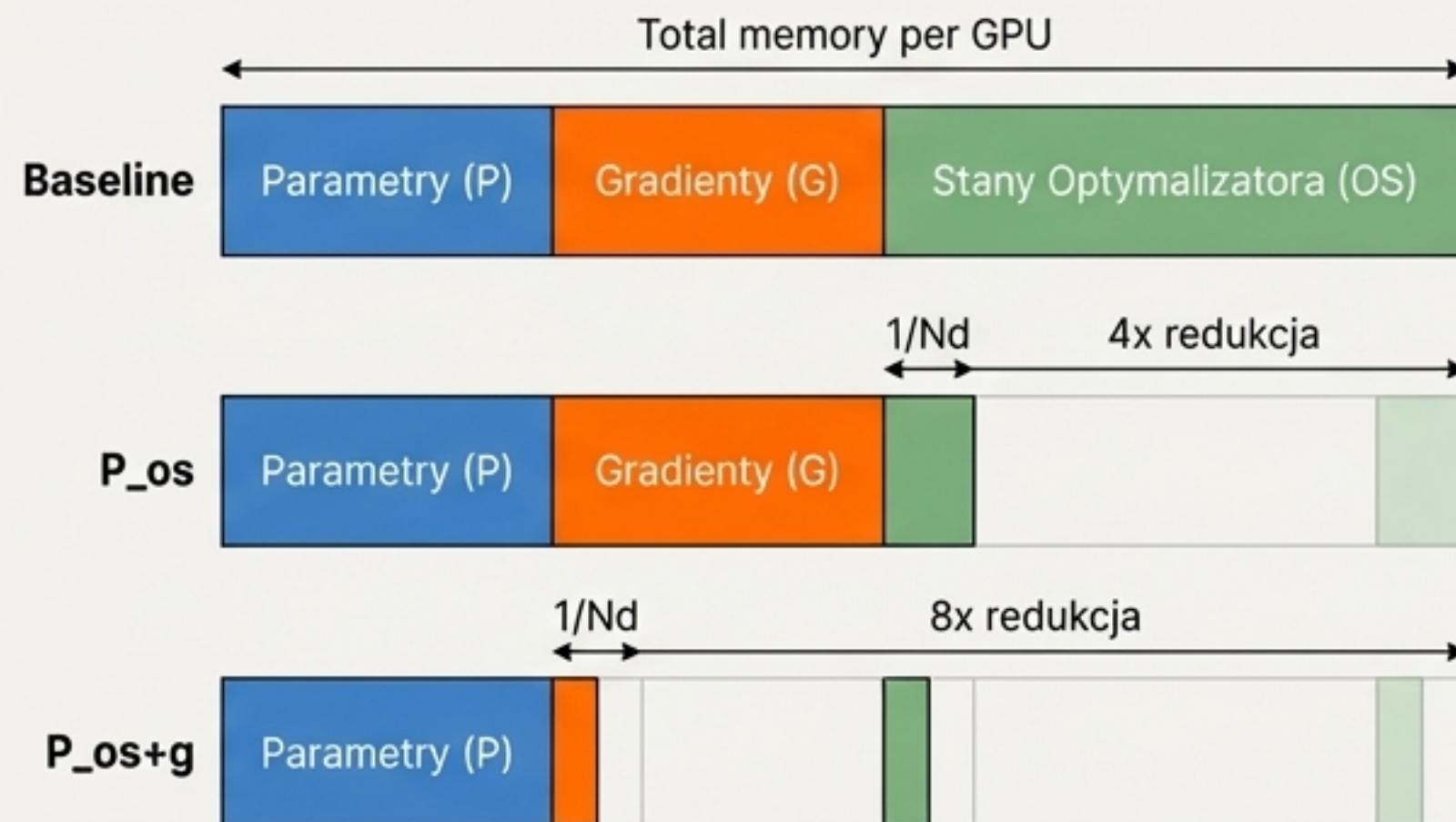
# Etap 1: Partycjonowanie Stanów Optymalizatora (P\_os)

## – "Darmowa" 4x Redukcja Pamięci



- **Mechanizm:** Zamiast replikować stany optymalizatora (największy zjadacz pamięci) na każdym GPU, dzielimy je. Przy 8 GPU, każde przechowuje i aktualizuje tylko 1/8 stanów.
- **Rezultat: 4x redukcja** całkowitego zapotrzebowania na pamięć dla stanów modelu.
- **Genialny Trick:** Osiągnięte przy **zerowym dodatkowym koszcie komunikacji** w porównaniu do standardowego DP. Operacja `All-gather` na koniec kroku treningowego zapewnia, że wszystkie GPU mają zaktualizowane parametry, co efektywnie zastępuje część standardowej operacji `All-reduce`.

# Etap 2: Partycjonowanie Gradientów (P\_os+g) – Podwajamy Stawkę do 8x



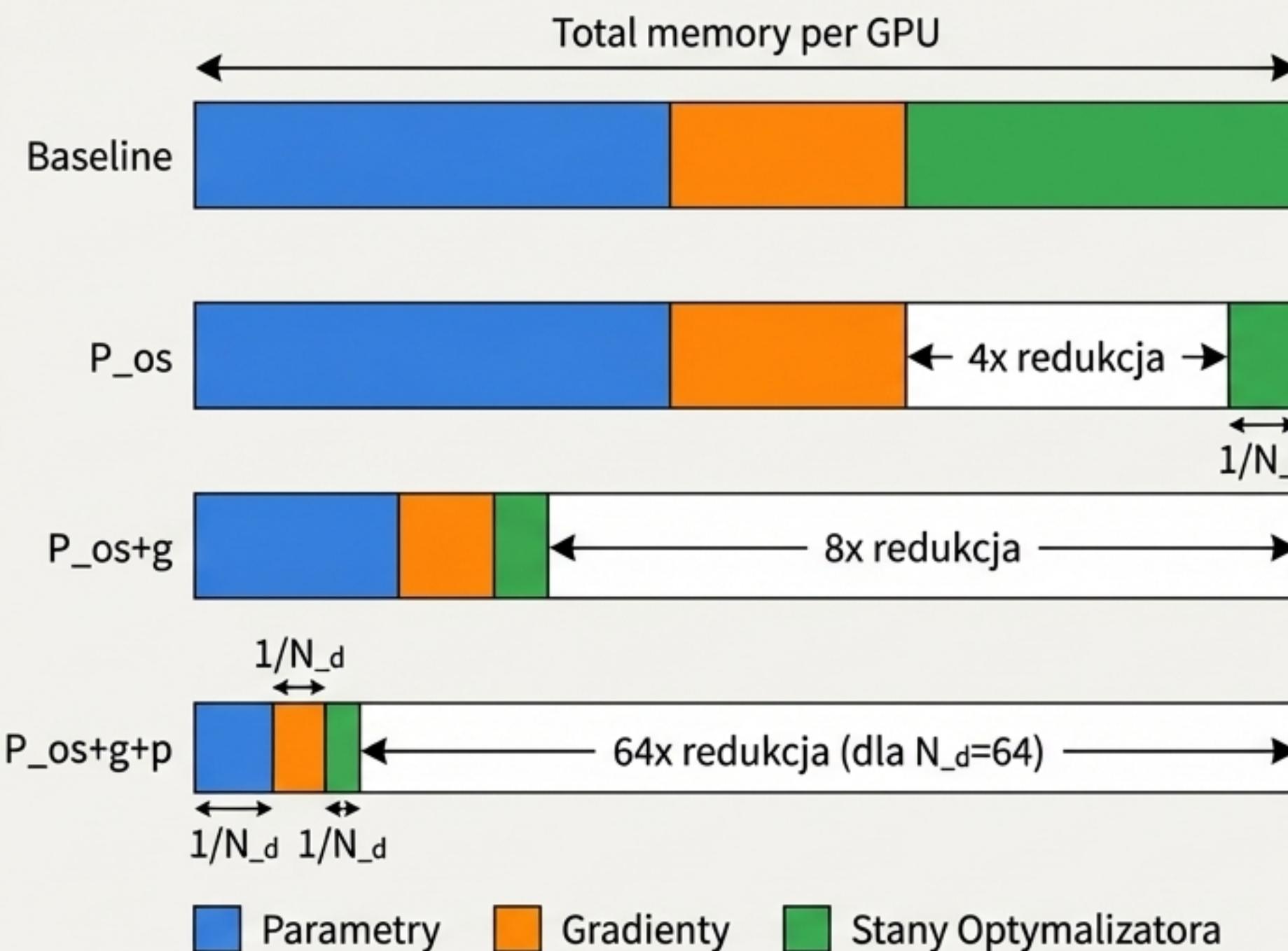
**Mechanizm:** Oprócz stanów optymalizatora, dzielmy również gradienty. Każde GPU jest odpowiedzialne tylko za gradienty przypisane do jego części parametrów.

**Jak:** Używana jest operacja **Reduce-Scatter** zamiast pełnego **All-Reduce**. Zamiast wysyłać wszystkie gradienty do wszystkich, każdy proces otrzymuje tylko sumę gradientów, za które jest odpowiedzialny.

**Komunikacja:** Całkowita ilość przesyłanych danych jest identyczna jak w standardowym All-Reduce – to po prostu mądrzejszy routing.

**Rezultat:** **8x redukcja** całkowitego zapotrzebowania na pamięć dla stanów modelu.

# Etap 3: Partycjonowanie Parametrów (P\_os+g+p) – Ostateczne Uderzenie



**Mechanizm:** Najbardziej radykalny krok – partycjonujemy same parametry modelu. Każde GPU przechowuje tylko fragment parametrów, za który jest odpowiedzialne.

**Dynamiczna Komunikacja:** Parametry dla danej warstwy są dynamicznie pobierane operacją **AllGather** tuż przed obliczeniami (forward/backward), a następnie zwalniane z pamięci.

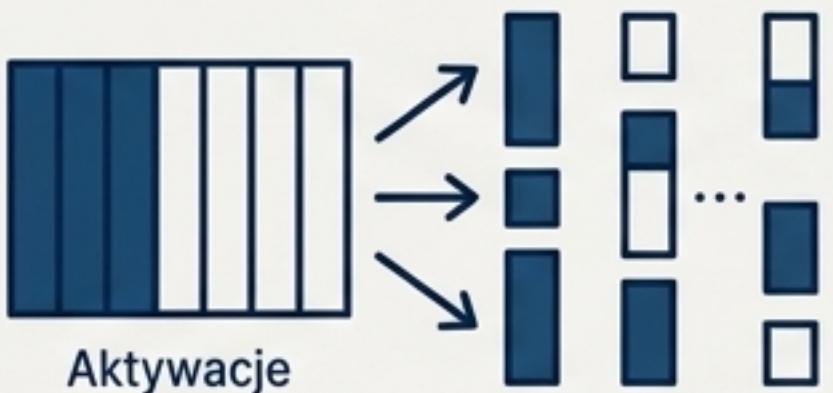
**Rezultat:** Redukcja pamięci staje się **liniowa w stosunku do liczby GPU**. Dla  $N_d=64$ , uzyskujemy 64x redukcję pamięci na stany modelu.

**Kompromis:** Objętość komunikacji wzrasta o 50%. Jest to jednak kompensowane przez możliwość użycia znacznie większych partii danych, co poprawia wykorzystanie GPU.

**Próg:** Model z 1 bilionem parametrów staje się możliwy do wytrenowania na 1024 GPU.

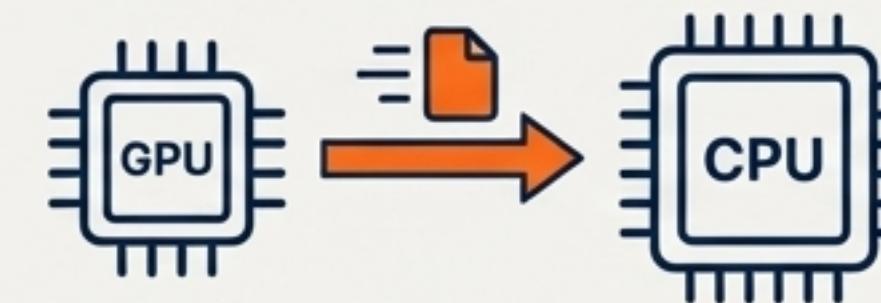
# Poza Stanami Modelu: ZeRO-R Ujarzmia Resztę Pamięci

ZeRO to holistyczny system optymalizacji. Po zredukowaniu pamięci na stany modelu, pozostałe elementy stają się wąskim gardłem. ZeRO-R rozwiązuje te problemy:

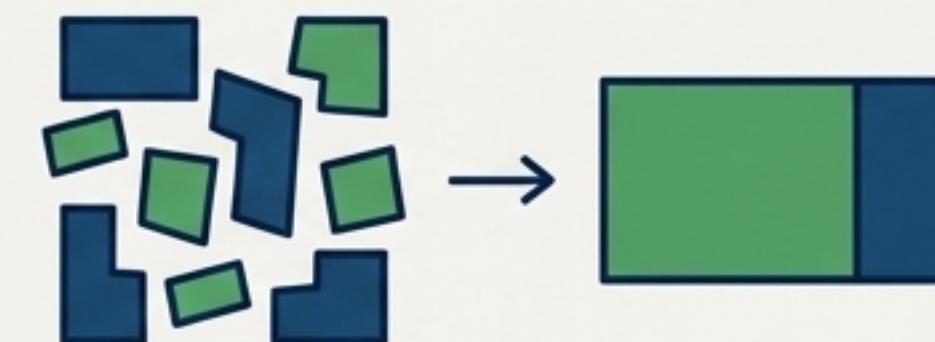


**Partycjonowanie Checkpointów Aktywacji (P\_a):** Zamiast replikować aktywacje w Model Parallelism, ZeRO-R je partycjonuje, redukując zużycie pamięci proporcjonalnie do stopnia MP.

**CPU Offload:** Inteligentne przerzucanie partycjonowanych checkpointów aktywacji do pamięci RAM. W przypadku dużych modeli, koszt transferu jest ukryty za obliczeniami.

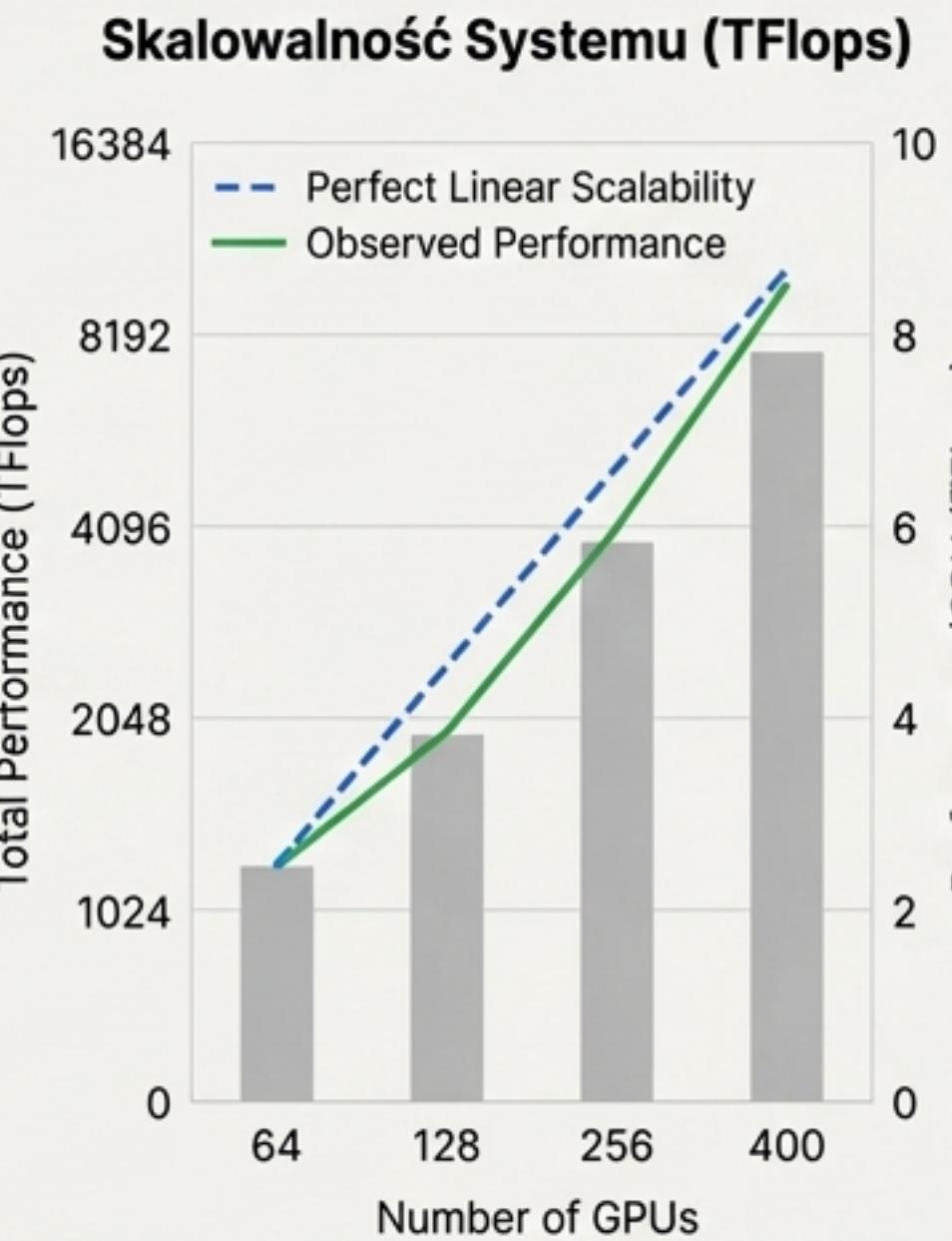
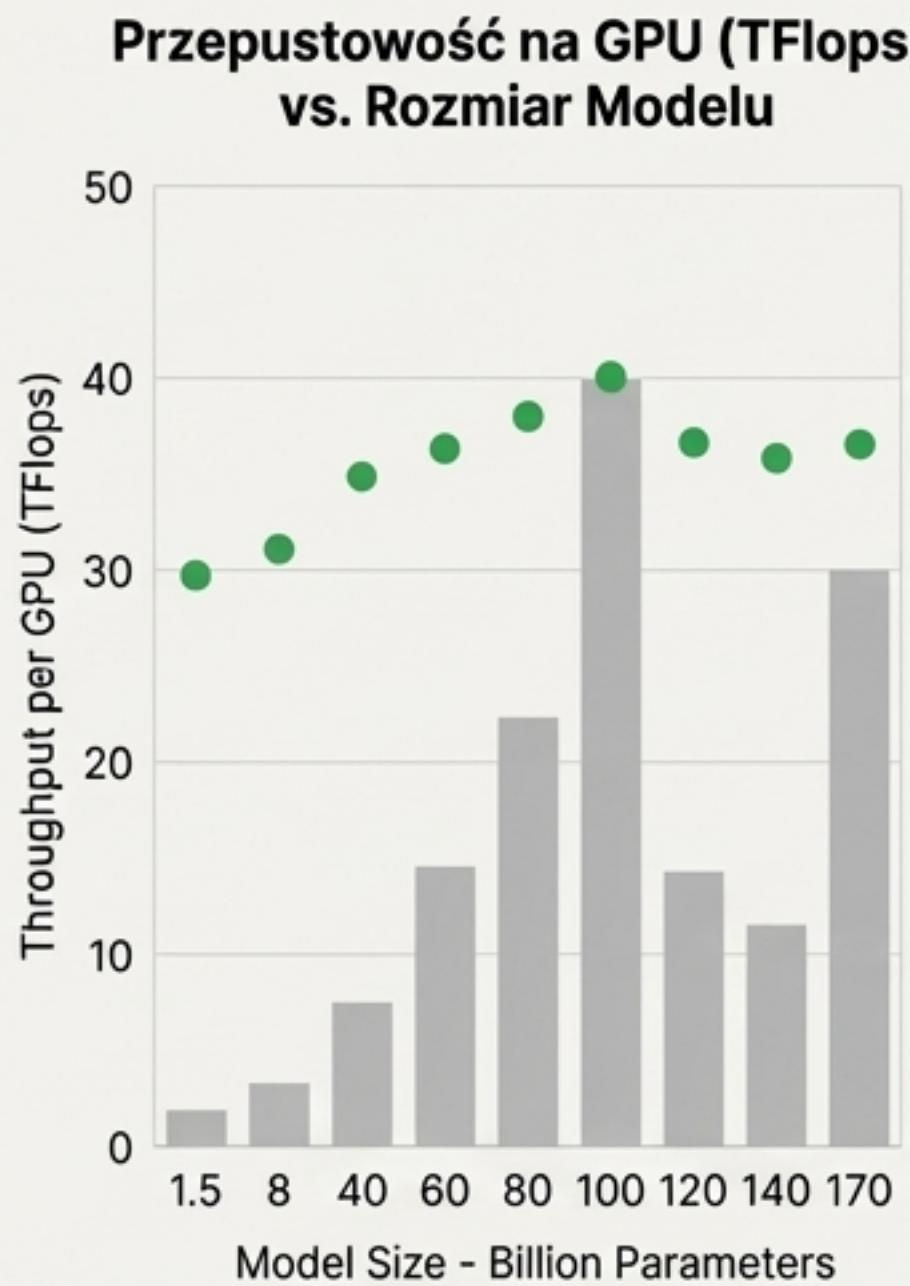


**Bufory o Stałym Rozmiarze (C\_B):** Zapobiega "puchnięciu" buforów tymczasowych (np. na spłaszczone gradienty) wraz z rozmiarem modelu.



**Defragmentacja Pamięci (M\_D):** Aktywne zarządzanie pamięcią zapobiega błędom "Out of Memory" poprzez przenoszenie długo żyjących obiektów do ciągłych bloków pamięci.

# Rewolucja ZeRO: Wyniki i Wpływ na Świat AI



**Wydajność:** W porównaniu do SOTA (Megatron-LM) na modelach 100B parametrów, ZeRO osiąga **do 10x wyższą przepustowość** (TFLOPs).

**Skalowalność:** ZeRO wykazuje **superliniową skalowalność**. Podwojenie liczby GPU daje ponad dwukrotny wzrost wydajności, ponieważ oszczędność pamięci pozwala na użycie większych, bardziej efektywnych partii danych.

**Demokratyzacja:** Umożliwia trening modeli do 13B parametrów przy użyciu samego Data Parallelism, otwierając drzwi dla grup uniwersyteckich i mniejszych laboratoriów.

**Dowód:** Technologia ZeRO umożliwiła stworzenie **Turing-NLG (17B parametrów)** – w momencie publikacji największego modelu językowego na świecie.



Turing-NLG