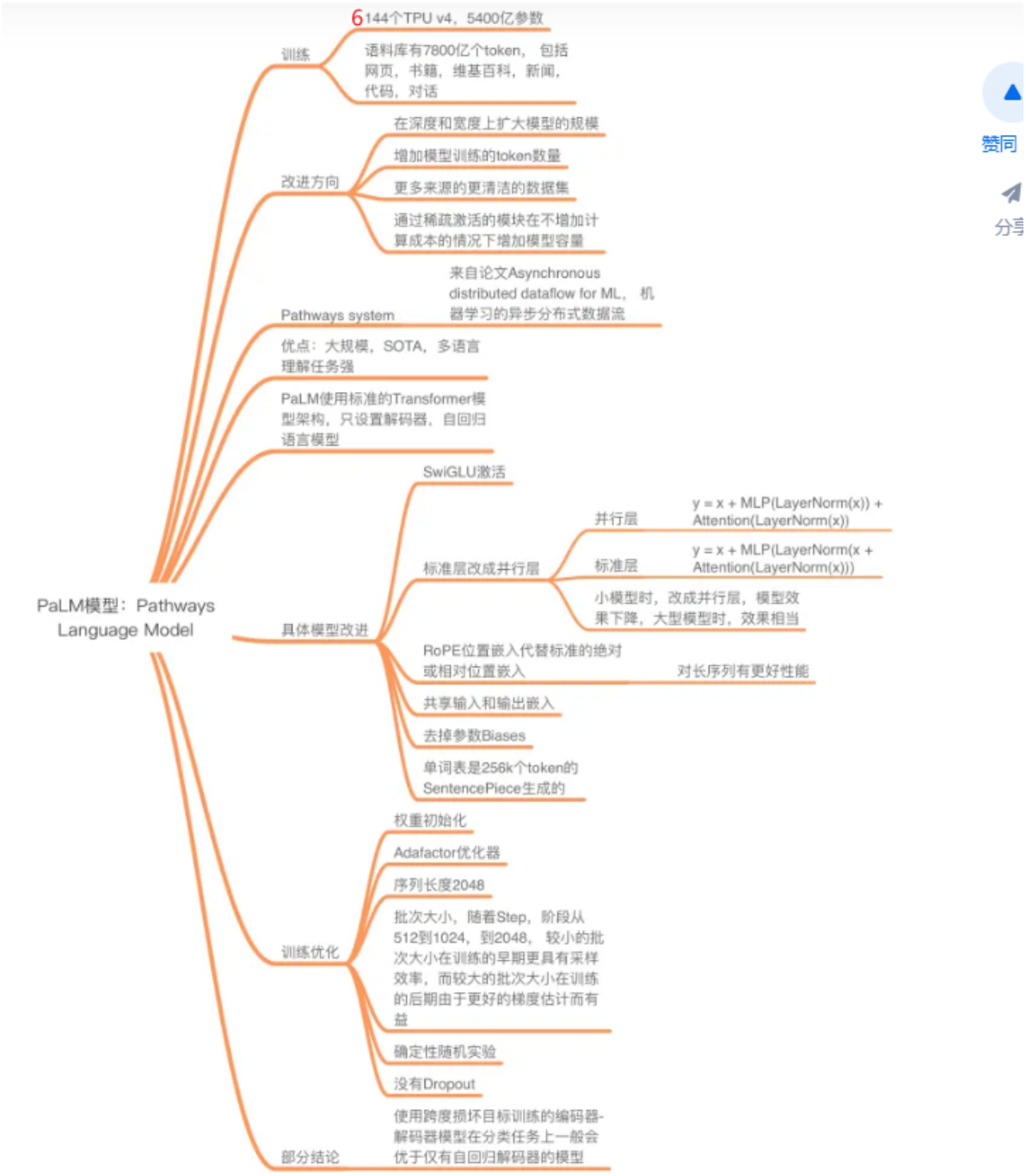
PaLM (Scaling Language Modeling with Pathways)

从 Jeff Dean 去年十月提出 Pathways 这个架构起。前段时间正式放出了 Pathways 的论文 **[1]**，但主要是讲底层的设计和性能。谷歌公布了这个架构所训出的第一个大模型 **PaLM**。

PaLM 是一个 5400 亿参数的单向语言模型。

为啥用单向而不是谷歌经典的 T5 呢？作者的解释是 **GPT3 这种结构的 few-shot 会更好**，而带有 **encoder 的模型得 finetune 才能有不错的表现，这样既需要很多数据，又得为每个任务改变模型权重，和 Pathways 的万能大模型初衷有些背离。**



OK，那就用 GPT3 的结构吧，但怎么效果好这么一大截？？？

Model	Avg NLG	Avg NLU
GPT-3 175B	52.9	65.4
GLaM 64B/64E	58.4	68.7
PaLM 8B	41.5	59.2
PaLM 62B	57.7	67.3
PaLM 540B	63.9	74.7

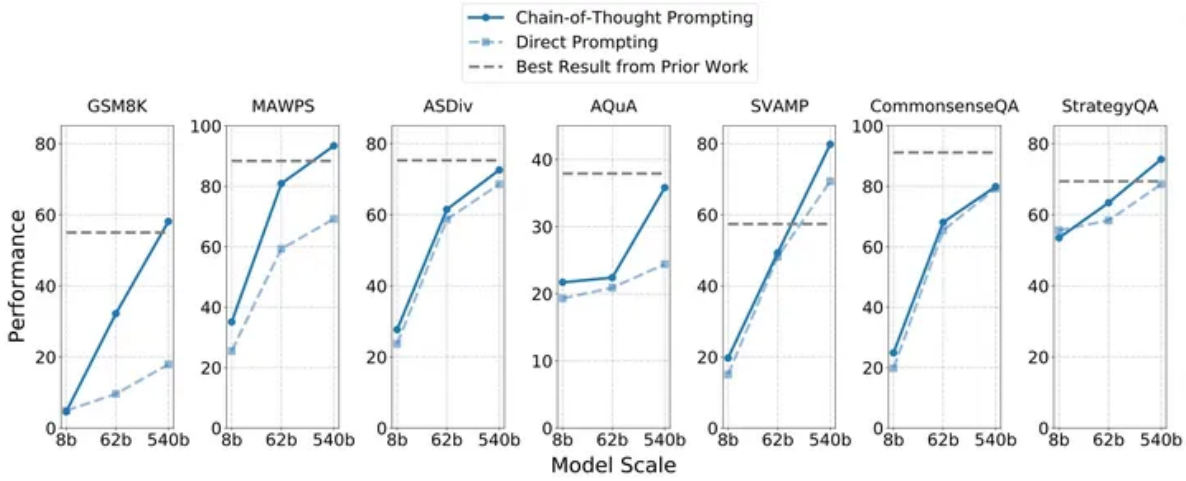
作者没有多做解释，个人猜测数据占大头，GPT3 用了近 500B token 的数据，而 PaLM 有 780B，多了不少。但同时 PaLM 的模型拟合能力也小了一半，所以真的挺难说。作者的原话也挺有意思：

```
Interestingly, the PaLM 62B outperforms the GPT-3 175B in both categories
```

类似 chinchilia deepmind 说了模型越大所需要的tokens越多

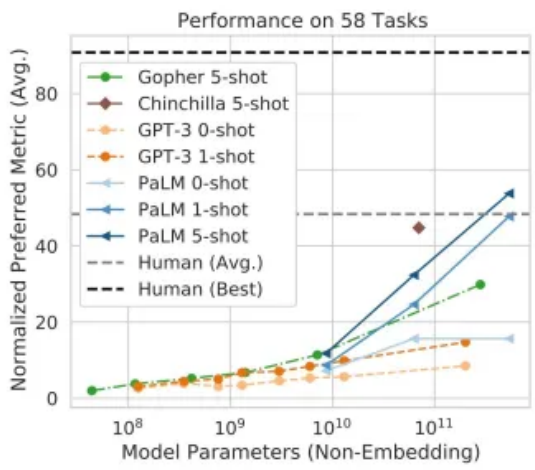
除了在 finetune 方面 pk 掉 GPT3，还让人注意到的就是推理能力的大幅提升。

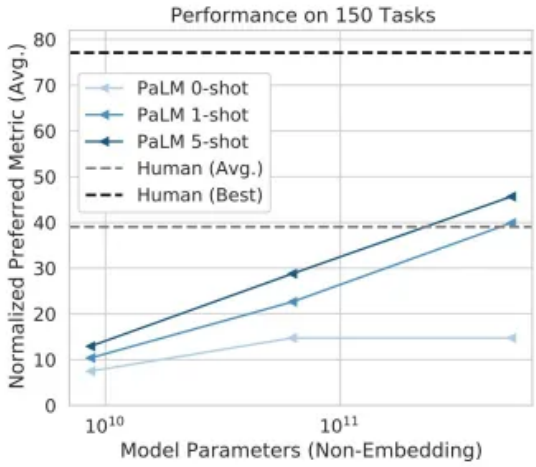
在 7 个算术应用题 / 常识推理数据集上，有 4 个超越了当前 SOTA，而且是以 8-shot 的 PaLM 跟精调的 SOTA 比，很多 SOTA 的结构都是特殊设计的。



这个震撼程度不知传达到没有，以前是一顿魔改调参还干不过大模型直接精调，现在马上就要变成**一顿魔改调参却干不过大模型 few-shot 了。**

除了早就被刷爆的 GLUE 和 SuperGLUE 之外，还有一个新的 **BIG-bench 评测，包含 150 个任务，专为大模型准备。**可以看到， **PaLM 的 1-shot 已经接近人类平均水平了，不过距离最好水平还马达马达达内。**同时，随着参数量的扩大，few-shot 模型的效果也展现了更大的提升，但 GPT3 为什么没展现同样的特性就有点玄学了。





那么，上面讲了那么多 PaLM 的效果，作者除了 Pathways 还做了啥呢？

模型层面主要有如下改动：

1. 使用 SwiGLU 激活函数，有研究证明在同等计算量下 SwiGLU 的效果更好
2. 把 FFN 和 Attention 并行

- 以前:

$$y = x + MLP(LayerNorm(x + Attention(LayerNorm(x))))$$

- 现在:

$$y = x + MLP(LayerNorm(x)) + Attention(LayerNorm(x))$$

1. Multi-Query Attention：以往做 attention 前我们都会把 Q 和 K 隐层映射到 `[head_num, head_size]`，而 PaLM 让所有头共享参数矩阵，只映射到 `[1, head_size]`，对训练速度和效果没什么影响，但却**提升了 decode 的速度**
2. 使用 RoPE**[2]** 位置编码：RoPE 是苏神的工作，主要利用三角函数的恒等变换来优化相对位置编码
3. 输入和输出共享 embedding 矩阵
4. 去掉所有的 Bias 项
5. 使用 256K 个 token 的 SentencePiece

但每个改动对于 PaLM 到底有多少提升，并没有消融实验。

pathways for this paper

Pathways 系统如何执行双向的 pod 级数据并行化。一个 Python client 构建了一个分片数据流程序（如图 2 左侧所示），该程序在远程服务器上**启动 JAX/XLA（XLA，2019）工作**，这些服务器各自构成一个 TPU pod。该程序包含一个用于 Pod 内前向 + 反向计算（包括 Pod 内梯度减少）的组件 A，用于跨 Pod 梯度迁移的迁移子图，以及用于优化器更新（包括本地和远程梯度的求和）的组件 B。Pathways 程序在每个 pod 上执行组件 A，然后将输出梯度迁移到另一个 pod，最后在每个 pod 上执行组件 B。Pathways 系统设计有几个特点，使其能够将程序执行扩展到数千个加速器芯片 -- 首先，它通过每个 pod 调度器的异步群组调度，掩盖了将 JAX/XLA 工作从单个 python client 调度到远程服务器的延迟（如图 2 右侧所示），其次，它通过分片数据流执行模型摊销了管理数据传输的成本（详情请参考 Barham 等人（2022））。

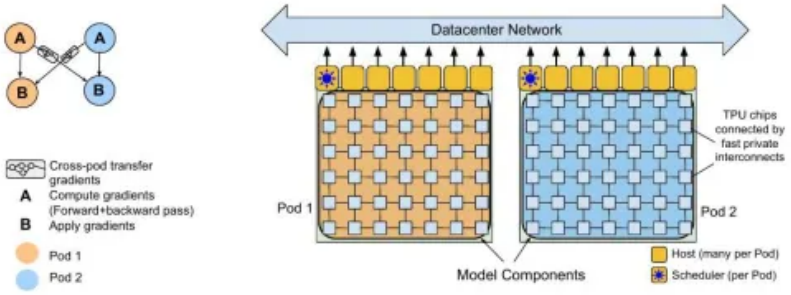


Figure 2: The Pathways system (Barham et al., 2022) scales training across two TPU v4 pods using two-way data parallelism at the pod level.

双向 Pod 级数据并行的一个有趣方面是，在 6144 个 TPU v4 芯片连接到**两个 Pod 共 1536 个主机的规模上，实现跨 Pod 梯度传输的高训练吞吐量是一个挑战。**请注意，跨 Pod 梯度传输只需要在两个 pod 的相应主机之间进行 1:1 的传输，因为每个核心只需要其模型托管参数的远程梯度。此外，两个 pod 之间的主机是通过谷歌数据中心网络连接的（Singh 等人，2015）。由于在每个核心完成梯度计算后才开始传输（如图 2 所示），这导致了一个非常突发性的工作负载，所有的主机在同一时间通过数据中心 - 网络链接传输他们的梯度。特别是，每对主机在每个训练步骤中交换大约 1.3GB 的梯度，相当于所有主机的总爆发量为 81Tbps。这一工作负载的突发特性带来了挑战，我们通过对 Pathways 网络堆栈的精心设计，使 DCN 链路得到最佳利用。例如，为了减轻拥堵的影响，梯度传输的数据被分解成较小的块，并通过多个较小的流在一组不同的 DCN 链路上进行路由。通过这些优化，在训练期间，相对于单个 Pod 的吞吐量，我们实现了约 1.95 倍的训练吞吐量（相当于 97% 的完美弱化扩展，因为相对于单个 Pod，我们将两个 Pod 的批次量翻倍）。与理论上 2 倍的吞吐量相比，性能上的差距是由于反向传递和跨 Pod 梯度减少之间缺乏重叠。我们希望在未来的工作中解决这个问题。