并行编程

笔记本:	study		
创建时间:	2021/1/7 16:48	**更新时间:**	2021/1/7 20:41
作者:	hopezhou		
URL:	https://developer.aliyun.com/article/371636		

> 系统实现
>
> - Iterative Map Reduce
> - Parameter Server
> - Data FLow

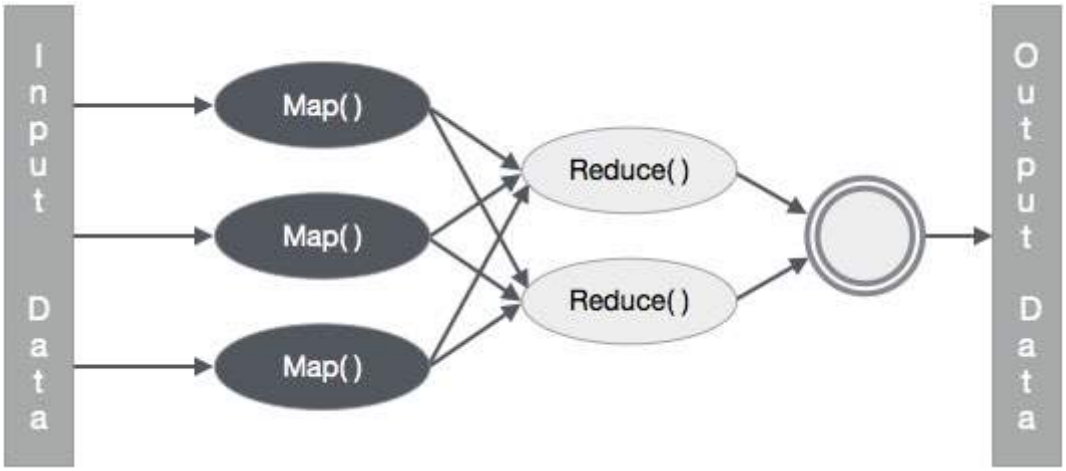
Iterative Map Reduce

- Hadoop
- Spark

在当前最流行的高性能并行体系结构中比较常用的并行编程环境分为两类: **消息传递和共享存储。**
MPI是基于消息传递的经典代表，是消息传递并行程序设计的标准，用于构建高可靠的、可伸缩的、灵活的分布式应用程消息传递并行处理开销比较大，适合于大粒度的进程级并行计算，相对其他并行编程环境，它具有很好的可移植性，几乎能被所有的并行环境支持;还具有很好的可扩展性，具有完备的异步通信功能，能按照用户的要求很好地分解问题，组织不同进程之间进行数据交换，适合大规模可扩展性的并行算法。
MPI模式在学术研究领域应用较多，而在商业领域，云计算系统大多采用的是**Google云计算系统中的MapReduce并行编程模型。**
云计算强调的就是简单的编程模型，而MapReduce就是一种高效的、简单的并行编程模式，也是一种高效的任务调度器。MapReduce这种编程模型不仅适用于云计算，在多核和多处理器、Cell processor以及异构机群上同样有良好的性能。利用MapReduce ,程序员能够轻松地编写紧耦合的程序，在运行时能高效地调度和执行任务，在实现时，在Map函数中指定对各分块数据的处理过程，在Reduce函数中指定如何对分块数据处理的中间结果进行归约。用户只需要指定Map和Reduce函数来编写分布式的并行程序，不需要关心如何将输人的数据分块、分配和调度，同时系统还将处理集群内节点失败及节点间通信的管理等。
而MPI仅仅是一个并行计算标准，没有相应的分布式文件系统的支撑，在大数据场景下大文件的存储及访问都会成为一个问题，同时用户还需要考虑集群节点之间的通信协调、容错等问题，这些使得MPI的编程难度比较大，集群本身的规模也很难做到像MapReduce那样的超大规模。

A MapReduce framework (or system) is usually composed of three operations (or steps):

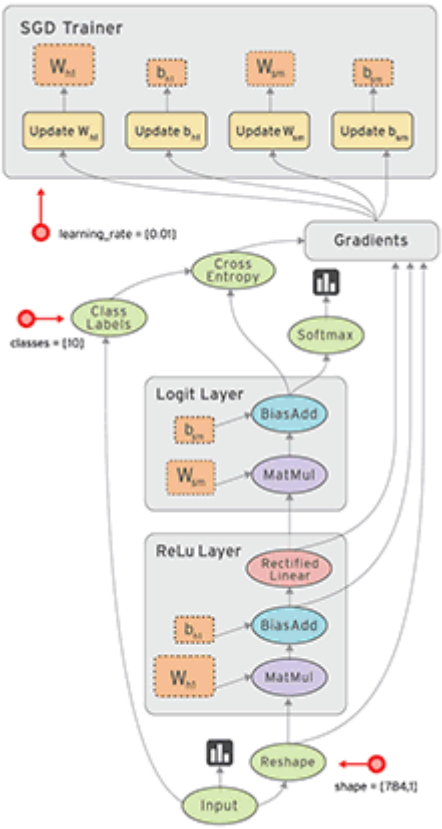
- Map: each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of the redundant input data is processed.
- Shuffle: worker nodes redistribute data based on the output keys (produced by the map function), such that all data belonging to one key is located on the same worker node.
- Reduce: worker nodes now process each group of output data, per key, in parallel.

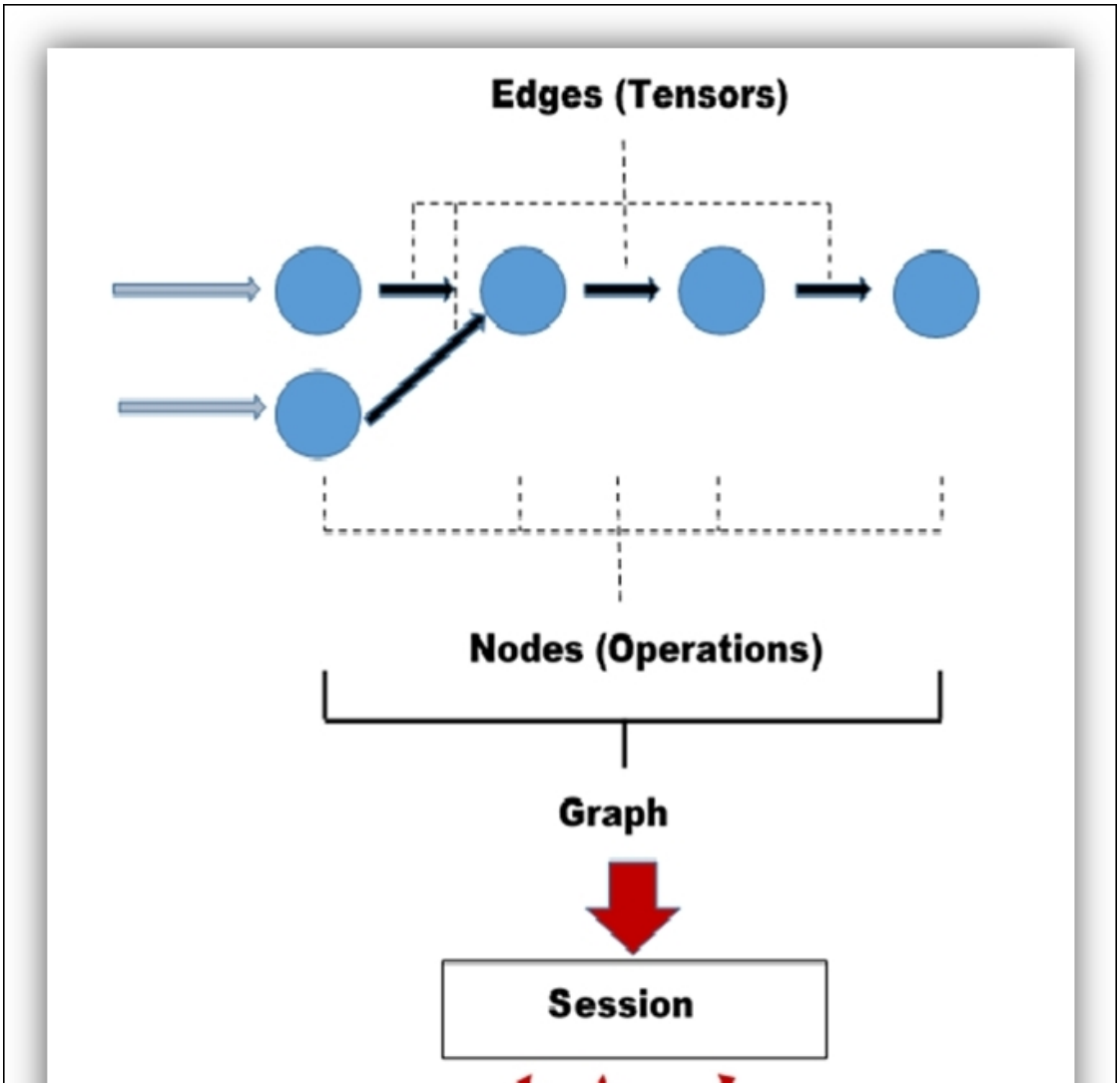


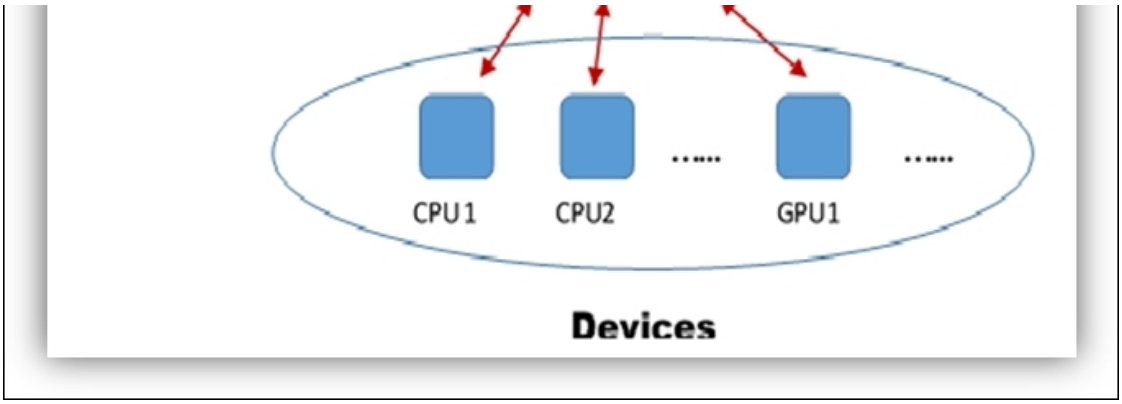
Parameter Server

Data Flow

TensorFlow™ 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个CPU（或GPU），服务器，移动设备等等。


SGD Trainer
Update W_hl
Update b_hl
Update W_sm
Update b_sm
learning_rate = [0.01]
Gradients
Cross Entropy
Class Labels
classes = [10]
Softmax
Logit Layer
BiasAdd
MatMul
ReLu Layer
Rectified Linear
BiasAdd
MatMul
Reshape
shape = [784,1]
Input


Edges (Tensors)
Nodes (Operations)
Graph
Session

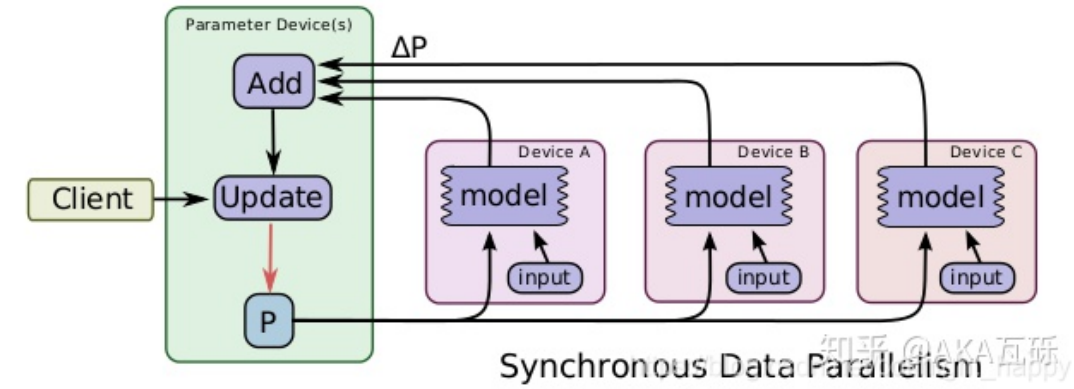


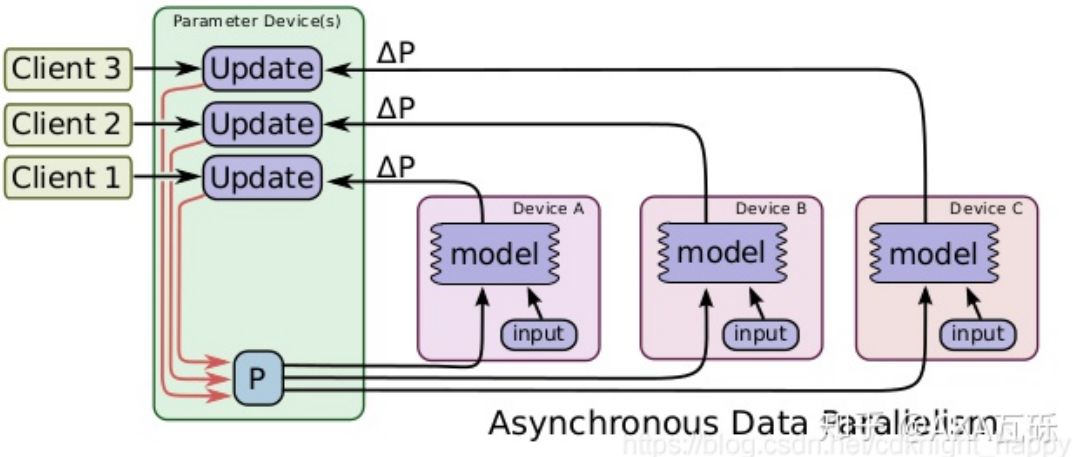
同步vs异步更新

Synchronize vs asynchronous
同步更新 vs 异步更新：
数据并行中，每个显卡通过自己拥有的那一份数据进行前向推理和梯度计算，根据梯度进行模型权重更新时，就涉及到了参数更新方式使用同步更新还是异步更新。

- 同步更新：所有的GPU都计算完梯度后，累加到一起求均值进行参数更新，再进行下一轮的计算；
- 异步更新：每个GPU计算完梯度后，无需等待其他更新，立即更新参数并同步。

同步更新有等待，速度取决于最慢的那个GPU；异步更新没有等待，但是会出现loss异常抖动等问题，实践中，一般使用的是同步更新。



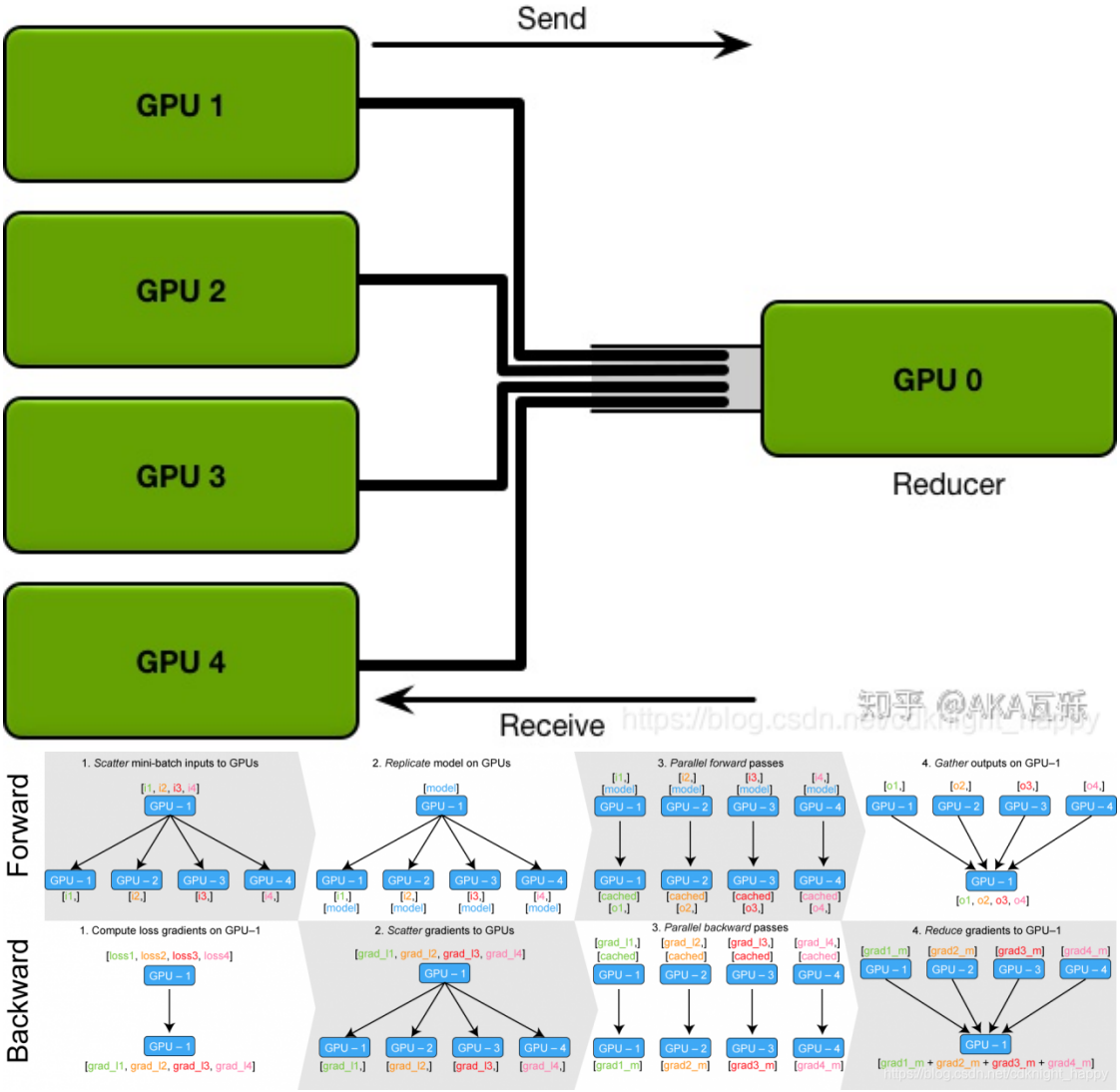


PS算法 vs Ring AllReduce算法

https://blog.csdn.net/cdknight_happy/article/details/108262595

PS算法，parameter server，GPU 0将**数据**分为五份发到各GPU上，各GPU用自己分到的数据计算输出，返回给GPU 0进行**loss累加**求均值，然后将梯度返回给每个gpu，每个gpu计算梯度（因为输入不同），然后在返回给GPU0 ，进行参数更新，再由GPU 0将**更新后的参数发给各GPU**

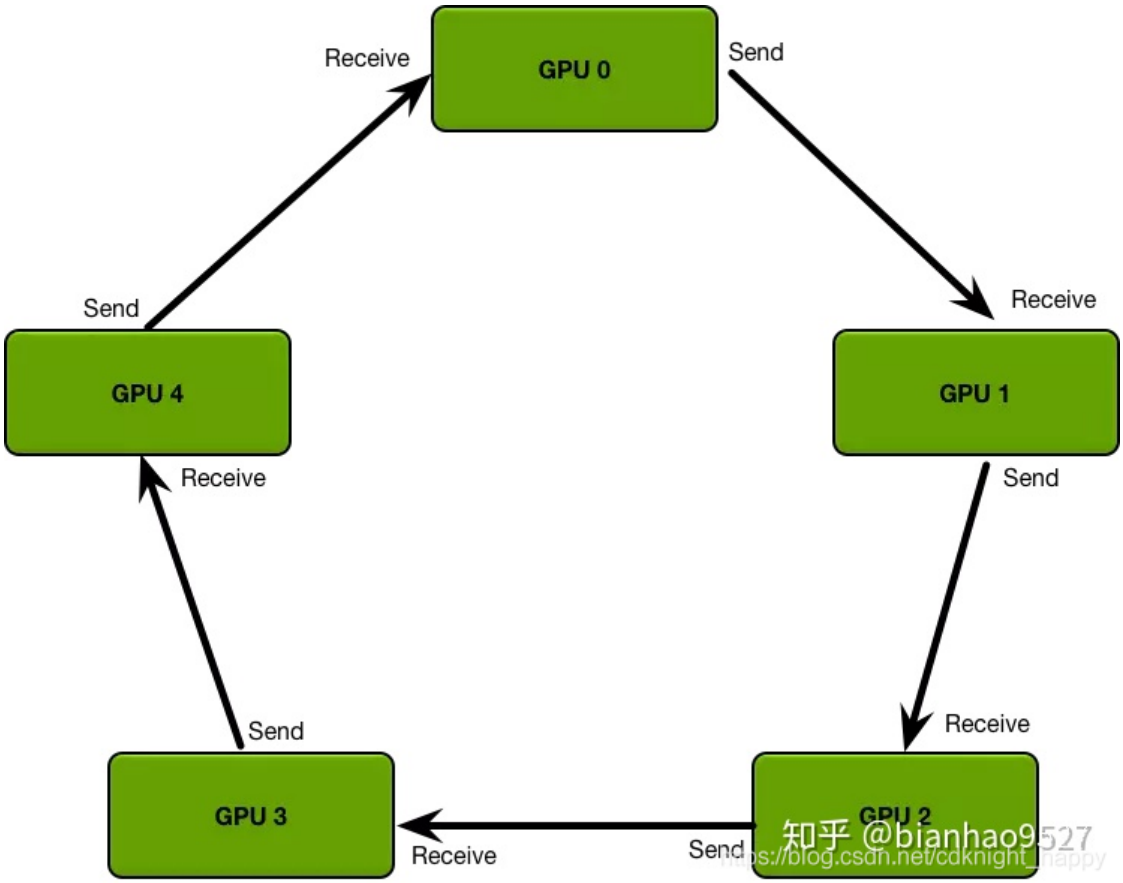
ps算法类似于map-reduce



Ring AllReduce算法：

Ring AllReduce算法把所有的GPU组成一个逻辑环，每个GPU都**从它的左邻居接收数据，向它的右邻居发送数据。**Ring AllReduce分为两个步骤：Scatter Reduce和All Gather，**Scatter Reduce用于在不同的GPU间交换数据，All Gather则用于在不同的GPU间同步数据。**

前向和后向过程均通过ring allreduce方式实现?



Scatter Reduce：
为简单起见，让我们假设目标是对一个浮点数的大数组求和; 系统中有N个GPU，每个GPU都有一个相同大小的数组，并且在allreduce的末尾，每个GPU都应该有一个相同大小的数组，其中包含原始数组中数字的总和。首先，gpu**将数组划分为N个更小的块**(其中N是环中的gpu数)。
相邻的GPU传递不同的参数，每次传递都是把当前GPU的数据累加到其右邻居上，同时接收其左邻居的输入数据累加到自身，在传递N-1次之后，可以得到每一份参数的累积（在不同的GPU上）。

All Gather：
得到每一份参数的累积之后，再做一次传递，将累加结果同步复制到所有的GPU上。All Reduce的通信成本和GPU数量无关。

相比PS架构，Ring-allreduce架构是带宽优化的，因为集群中每个节点的带宽都被充分利用。此外，在深度学习训练过程中，计算梯度采用BP算法，其特点是后面层的梯度先被计算，而前面层的梯度慢于后面层，Ring-allreduce架构可以充分利用这个特点，在后面层梯度计算的同时进行前面层梯度的传递，从而进一步减少训练时间。在百度的实验中，他们发现训练速度基本上线性正比于GPUs数目（worker数）。

GPU

GPUs are built on SIMD (Single Instruction Multiple Data) architecture. This means each streaming multiprocessor (SMP) (each GPU has multiple SMPs) can execute only one instruction on all the threads at a time but on potentially different data.

GPUs are designed for very high compute-density (computations per area). Thus, they have more ALUs (Arithmetic Logic Units) on the same die compared a CPU. To achieve this high compute-density, GPUs need to make some sacrifices. For example, in a GPU, a full-warp, up to 32 threads, shares the same fetch unit. This implies that all MUST work on the same task in lock step.