# Google Mapreduce 论文

## Mapreduce 编程模型

总的来讲，Google MapReduce 所执行的分布式计算会以一组键值对作为输入，输出另一组键值对，用户则通过编写 Map 函数和 Reduce 函数来指定所要进行的计算。

由用户编写的Map 函数将被应用在每一个输入键值对上，并输出若干键值对作为中间结果。之后，MapReduce 框架则会将与同一个键 fig: 相关联的值都传递到同一次 Reduce 函数调用中。

同样由用户编写的 Reduce 函数以键 fig: 以及与该键相关联的值的集合作为参数，对传入的值进行合并并输出合并后的值的集合。

形式化地说，由用户提供的 Map 函数和 Reduce 函数应有如下类型：



值得注意的是，在实际的实现中 MapReduce 框架使用 Iterator 来代表作为输入的集合，主要是为了避免集合过大，无法被完整地放入到内存中。

## 函数式编程模型

了解函数式编程范式的读者不难发现，MapReduce 所采用的编程模型源自于函数式编程里的 Map 函数和 Reduce 函数。后起之秀 Spark 同样采用了类似的编程模型。

使用函数式编程模型的好处在于这种编程模型本身就对并行执行有良好的支持，这使得底层系统能够轻易地将大数据量的计算并行化，同时由用户函数所提供的确定性也使得底层系统能够将函数重新执行作为提供容错性的主要手段。

## MapReduce 容错机制

由于 Google MapReduce 很大程度上利用了由 Google File System 提供的分布式原子文件读写操作，所以 MapReduce 集群的容错机制实现相比之下便简洁很多，也主要集中在任务意外中断的恢复上。

## Worker 失效

在 MapReduce 集群中，Master 会周期地向每一个 Worker 发送 Ping 信号。如果某个 Worker 在一段时间内没有响应，Master 就会认为这个 Worker 已经不可用。

任何分配给该 Worker 的 Map 任务，无论是正在运行还是已经完成，都需要由 Master 重新分配给其他 Worker，因为该 Worker 不可用也意味着存储在该 Worker 本地磁盘上的中间结果也不可用了。Master 也会将这次重试通知给所有 Reducer，没能从原本的 Mapper 上完整获取中间结果的 Reducer 便会开始从新的 Mapper 上获取数据。

如果有 Reduce 任务分配给该 Worker，Master 则会选取其中尚未完成的 Reduce 任务分配给其他 Worker。鉴于 Google MapReduce 的结果是存储在 Google File System 上的，已完成的 Reduce 任务的结果的可用性由 Google File System 提供，因此 MapReduce Master 只需要处理未完成的 Reduce 任务即可。