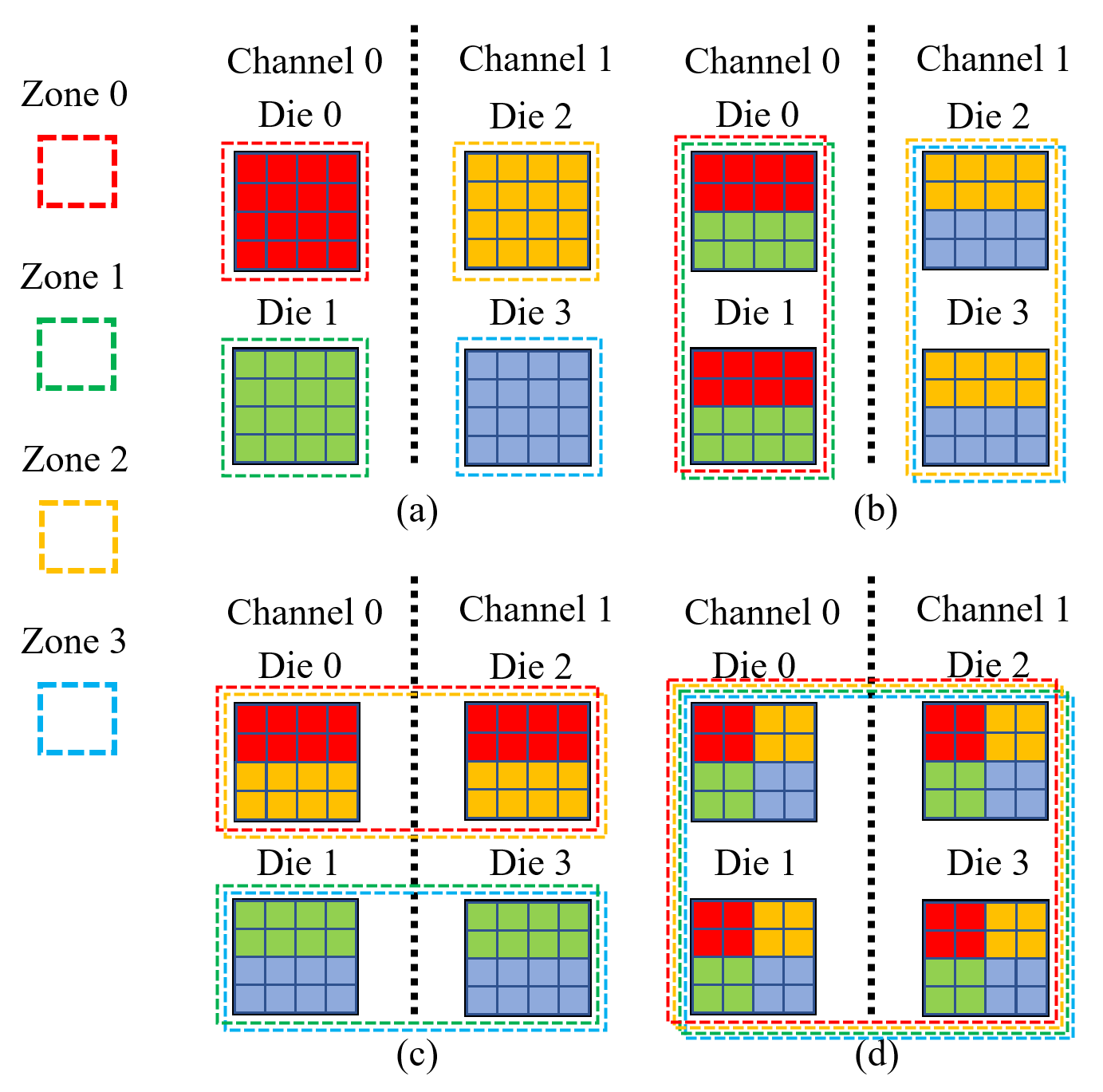
固态盘的并行单元为通道级并行，芯片级并行和晶圆级并行。ZNS传统的固态硬盘性能隔离方案，是单纯地将来自不同应用程序的工作负载占用的存储空间映射到不同的通道，以避免不同负载之间的相互干扰。但这种固定的按通道划分方式一定程度浪费了SSD并行性。

ZNS SSD 引入的新的逻辑单元zone让我们为不同应用程序划分存储空间，提供了全新的方式。就 ZNS SSD 硬件而言，在将区域映射到内部硬件资源策略方面因厂家、型号不同而异，并不存在固定的标准，而逻辑zone到硬件资源的映射或者称为——zone的构造方式，决定着zone的 I/O 并行能力以及与其他zone性能隔离的能力。例如，

如果区域与多个通道和芯片关联，当工作负载的读写请求IO\_size较大时，即使与单个zone进行交互，也能活得较大的读写性能。同时，zone覆盖的范围越广也意味着，需要与更多的其他zone共享并行资源，也就是更容易收到其他zone上负载的干扰。 另一方面，如果不同的区域映射到单独且不同的通道，则访问不同区域的应用程序彼此独立运行，具有更好的隔离性。图1展示四种不同并行度的zone构造方式。



图表 1 zone的不同构造方式示意

图1场景中的物理资源为2个channel，每个channel 2个die共4个。描述zone的工造方式时，我们关注channel 与die两级并行单元，channel用于数据传输，die用于数据读写操作。我们用二元组来表示(num\_channel,num\_die)来表示一个逻辑zone在物理上面的构造。 上面的例子我们假设zone的容量与die的容量一致。在(a)中 每个zone都单独占据了一个一个die，它的构造方式就是(1,1)。同时，zone0与zone1共享channel 0的传输带宽，zone2与zone3共享channel1的传输带宽。 在图(b)中zone 0与zone2 共享channel 0中的两个die，zone1 和zone3共享channel 1中的两个die，构造方式为（1，2）。这意味着zone0与zone1 除了共享着channel 0的传输带宽，还共享着两个die资源的读写带宽。zone 2 与zone3亦是如此。在图（c）中，zone 0和zone2 ，zone1和zone3共享不同channel的两个die，构造方式为（2，1）。Zone 0和zone2，zone1和zone3分别共享着两个die的读写带宽，且每个channel的传输带宽 都被4个zone所共享。图(d)中，每个zone在每个die上都有资源分布，每个die的读写带宽都被4个zone共享，每个channel的传输带宽也被4个zone共享。从并行度上限来看，(a)<(b)<(c)<(d),而并行度越高也说明与更多的zone共享资源，有更大的概率收到其他zone上的负载干扰，因此隔离性正好相反。因为

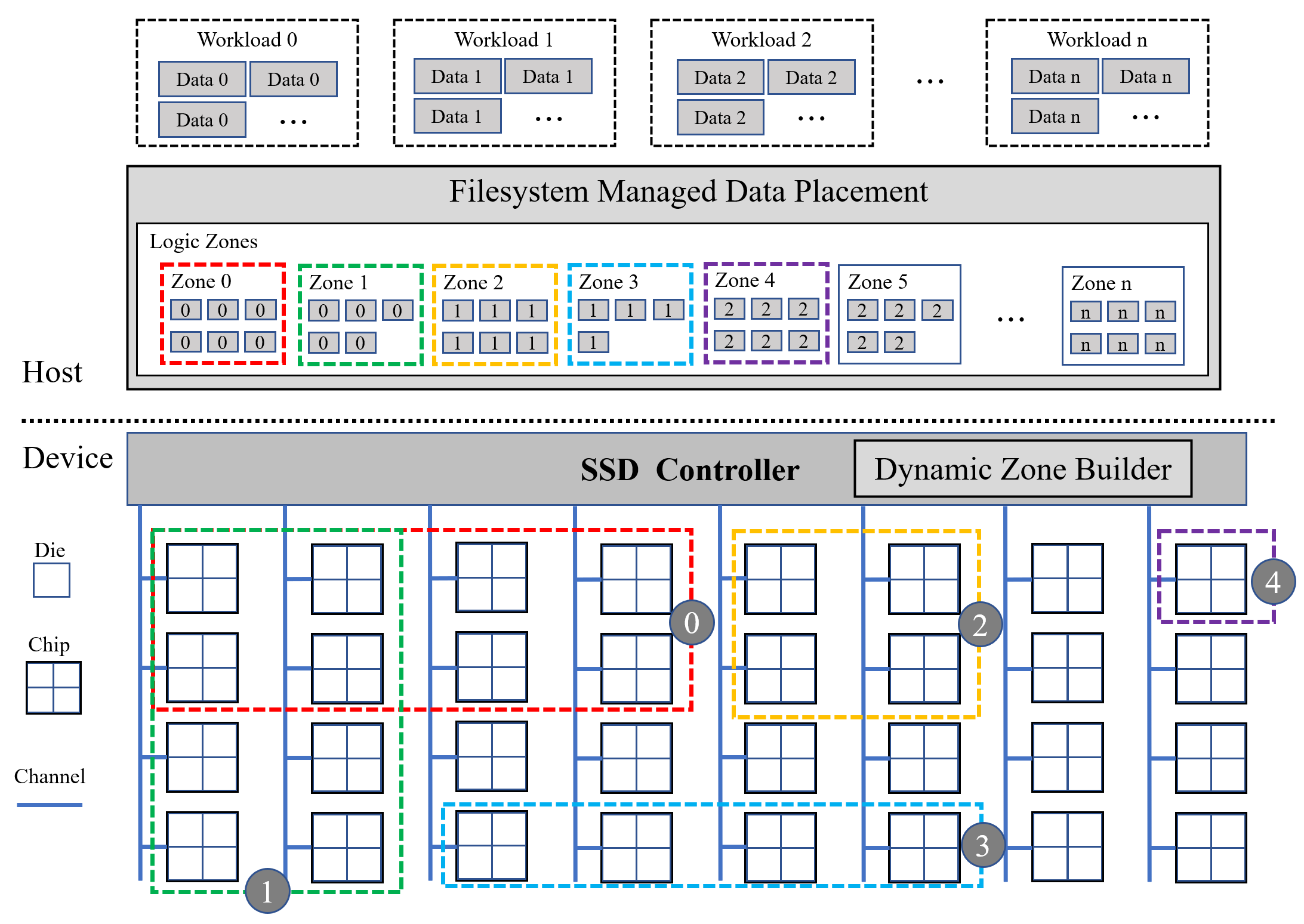
而之所以在zone 跨度大的时候，不通过扩大zone容量以独占更多的并行资源这种方式来提升隔离性是因为，越大的zone意味着更大的垃圾回收开销与更严重的写放大问题。因此，通常同一ZNS SSD设备通常会固定zone\_size的大小,以方便管理。

现有ZNS SSD 在生产时就已确定其内部zone到硬件存储资源的映射关系，这种固定的zone构造，以及忽视工作负载特性地盲目的资源划分，都将使得不同应用程序负载的并行度以及相互之间的隔离性变得不可控。

因此我们提出, DZB(Dynamic Zone Builder),一个工作在设备端且对应用工作负载感知的动态zone构造器，主要包括两个方面：

·不同的分区构造

·动态的资源分配与回收

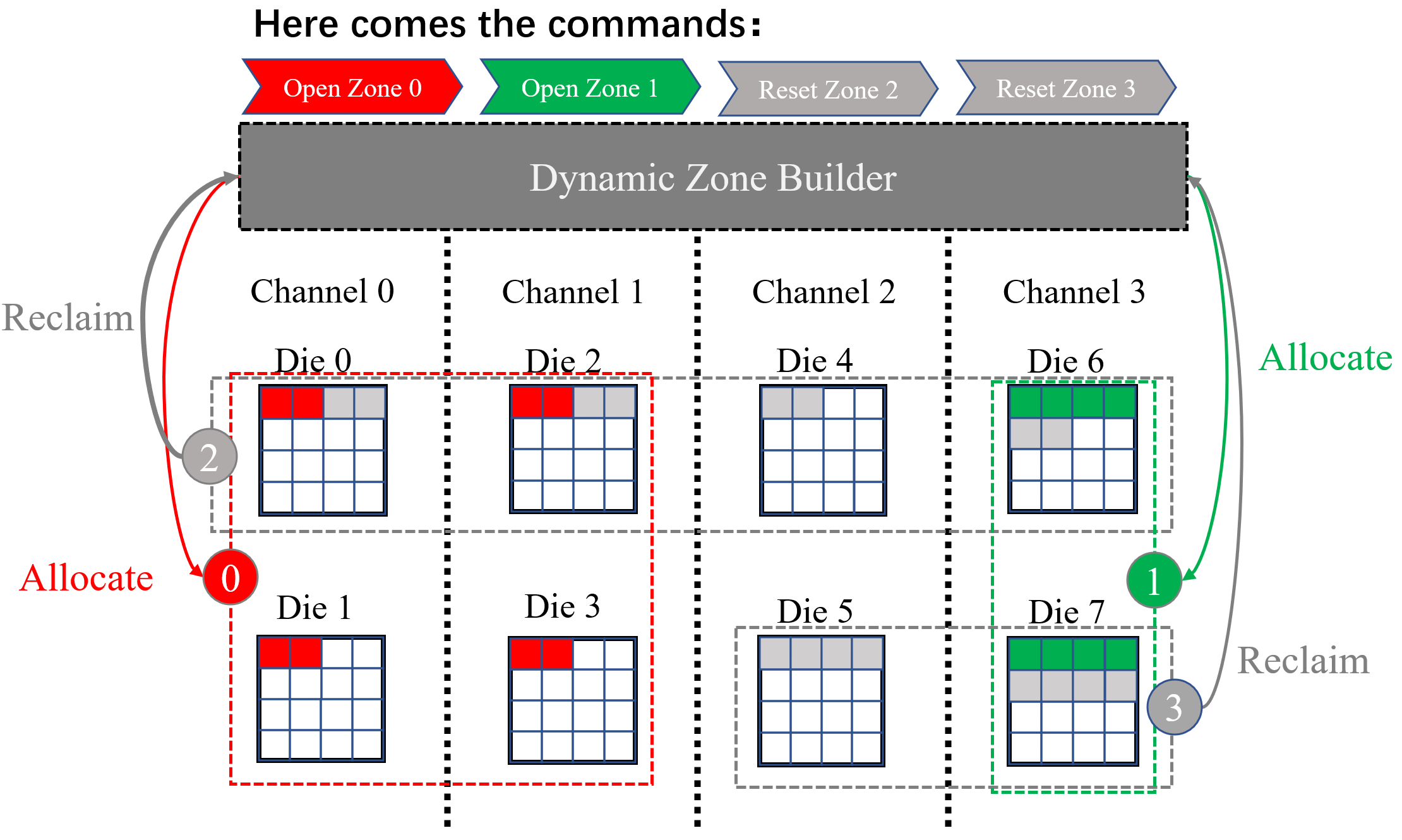


图表 2 数据分布以及Zone构造方式示意图with DZB

图2是加入了DZB的ZNS SSD设备工作时，数据分布以及Zone构造方式示意图。

首先，由主机控制的数据放置方式方便实现将不同应用程序的工作负载划分到不同的Logic Zone内。DZB工作在设备端，负载管理Logic Zone 逻辑地址到物理资源的映射关系。不同于现有的ZNS SSD 一致的Zone 构造方式（虽然Zone构造方式因ZNS SSD的不同厂家型号而异，但同类型的ZNS SSD内部所有Zone的划分实际上是保持一致的），DZB管理下的Zone构造方式是灵活变化的，使得不同Zone可以有不同的构造方式使得每个Zone有了灵活的并行度，同时Zone之间的干扰也因共享的并行资源的不同而产生差异。例如上图中，Zone 0与Zone 1 的构造分别为（4，2）与（2，4），虽然在范围上跨越了相同数量的die，但Zone 0 跨越了更多的通道，并行性上会优于Zone 1。同时二者共享了两个通道内共4个die的资源。Zone 2与Zone 3的构造分别为2×2与4×1，Zone 3跨越了更多的通道，并行性上优于2，同时二者共享了两个通道但是并无die上的资源共享。只考虑图中现实的zone的话，图中的4号zone独占一块die，虽然并行度性上较差，但是在数据读写和数据传输上不会受到图中任何其他zone的干扰。

其次，zone的构造方式并不是在生产时觉确定且固定不变的，而是在zone第一次被用到时根据通过分析申请开辟新空间的应用程序负载特性，以及硬盘压力状况进行为zone进行动态的资源划分。并且在不再被使用时进行资源回收等待分配给其他将要被使用的zone。具体的，当empty状态的zone通过显式或者隐式的方式被打开，DZB会根据申请该zone资源的应用程序工作负载信息以及硬盘各并行资源的压力信息，综合分析决定该zone的的硬件资源分配。并且当收到某个zone的reset指令，即从其他状态转移到empty状态，说明该zone中的数据均已失效。此时会将之前分配给该zone的资源重新回收。如图3所示



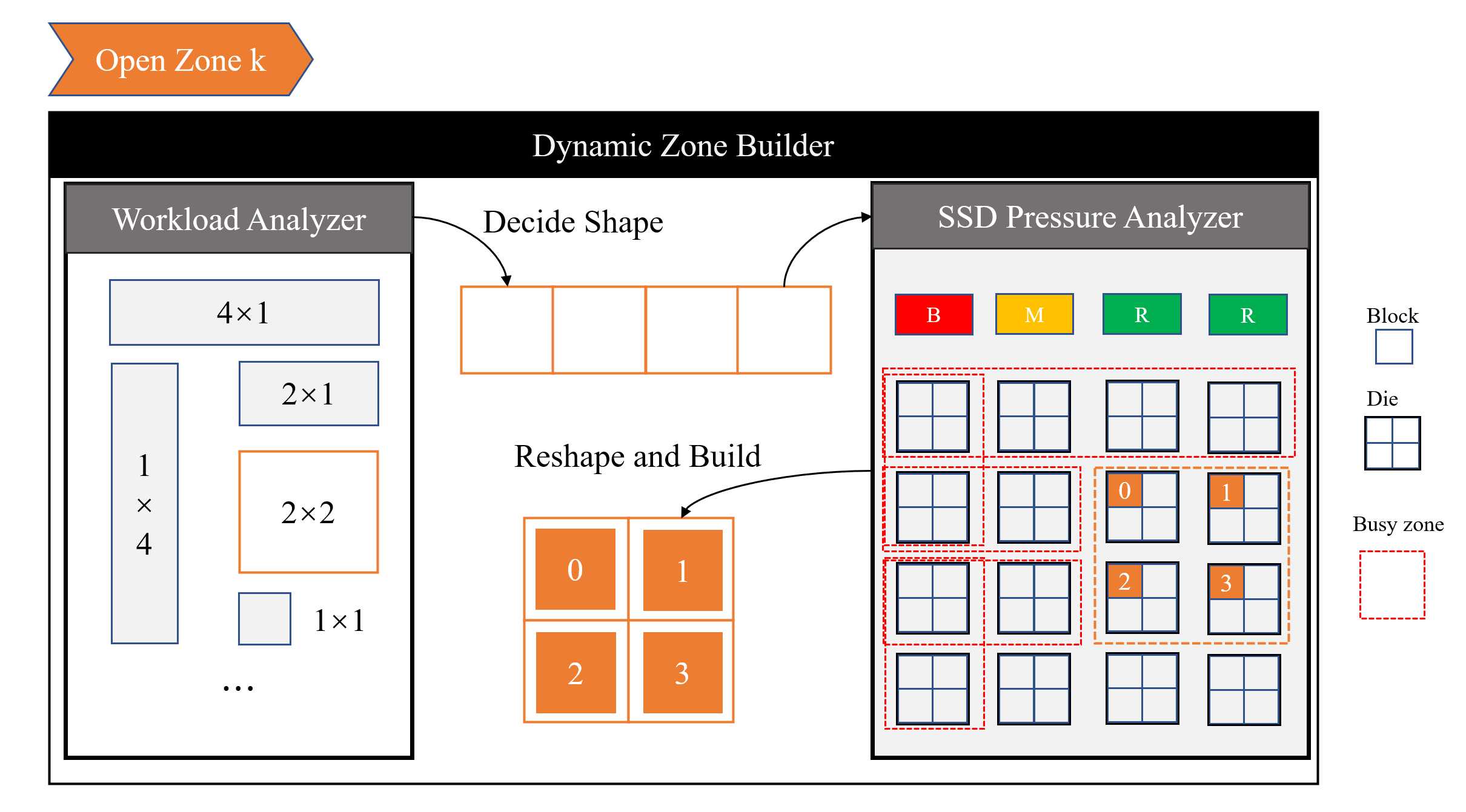
图表 3 zone资源动态分配回收过程示意

（在背景部分要讲解zone的状态转移）。如图3所示，在该场景中，设备收到zone 0 与 zone 1 首次被打开的命令，DZB根据该二者所属的应用程序负载特征，为zone 0 构造了（2，2）的二元组即认为适合为zone1分配两个channel的传输资源以及每channel两个die的读写资源，为zone1构造了（1，2）的二元组，即认为适合为zone1分配一个channel的传输资源以及两个die的读写资源。然后根据当前硬盘各部分资源承受复杂压力以及当前空闲资源池进行具体的资源划分，最终将zone 0映射在channel 0 channel 1上的4个die上，将zone1映射在channel 3的两个die上。设备在收到zone 2与 zone 3的reset命令后，就会将之前划分给这两个zone的资源回收进空闲资源池。

DZB的核心功能是决定zone的构造并合理分配硬件资源，以实现保证性能隔离的同时，最大程度得发挥硬件的并行性优势。此时，需要获取两方面的关键信息，一是zone所属应用程序的工作负载信息，二是当前设备各部分并行资源所承载的负载压力信息。因此DZB包含了两部分关键组件：

·工作负载分析器，通过分析特定工作负载以及与其他负载进行比较，根据相对负载特征，保证并行性与隔离性的均衡，为将要分配的zone决定其构造二元组(num\_channel,num\_die)。具体的，当zone被打开时会携带着所属工作负载标识以方便DZB后续识别被打开的zone都属于哪些工作负载。当zone承载读写请求时，DZB便会统计一段时间内zone上承载的请求大小,请求强度，读写比例信息，以此DZB便可维护这一段时间内不同工作负载的特征。工作负载特征并不会一直保持不变会随着时间变化，因此属于同一个应用程序的zone有可能有着不同的构造二元组。当携带着某一个工作负载标识的zone打开命令到来时，DZB便会第一时间知晓该工作负载近一段时间的特征，已分配合适构造二元组的zone，在此阶段更注重考虑充分利用硬件并行性资源。

·设备压力分析器，在确定了zone的构造



图表 4 DZB两组件工作简易示意

（在背景部分要讲一下数据条带化存储）

首先，为充分利用并行性资源，当zone分配到某一个die上的存储资源的时候，这部分资源会在die中再次细粒度划分，被均匀地划分给die上的每一个plane。For example，一个die上有4个plane，一个zone占用了一个die上400MB的容量，那么die上的每个plane都会提供100MB容量。然后我们对数据进行条带化存储，那么一个zone的条带宽度stripe\_wideth就是4×num\_channel×num\_die。 stripe\_size为每次在plane上写的数据块大小，那么当 zone的二元组与条带设置满足：

(IO\_size)/ stripe\_size ≈ stripe\_wideth

= num\_die 总数

便可以充分利用并行性资源，因此首先根据IO\_size 依照此公式可以初步确定num\_channel×num\_die的乘积，也就是占用的die的总数量。假设num\_channel×num\_die为c。

然后，根据IO强度（单位时间内请求数据量）初步进行num\_channel的确认，由于num\_channel和num\_die当前成反比关系，num\_channel越小，那么单个channel承载的die数量就会越多，也意味着更好的隔离性。我们提出用下列公式用以维护num\_channel与num\_die的数量关系:

α\*IO\_Intensities/num\_channel≤β\*(num\_die/die\_per\_channel)\*channel 理论带宽

其中channel 理论带宽可以通过硬盘参数计算，α（0~1）为该负载的 ，由DZB自身统计分析得出，用以代表当这个工作负载的所有请求中落在新zone上的概率。因此公式左侧可以计算得出当num\_channel为某值时，这个zone落在单通道上的工作强度。

β强度阈值 由管理员手动设置 由工作负载的读写比例来确定，用以表示公式左边计算得出的单通道请求强度不可超过单channel理论带宽的百分之多少。β越大则channel压力越大。由于die对于读操作的处理速度远超过写，因此die上相同的请求数据量，读比例高的负载会给channel更大的压力，因此我们希望读比例大的时候β应该尽量小，因此取：

num\_channel的取值应该为

β=1/(read\_ratio+0.01)

当百分百读的时候，β为1,当百分百写的时候我们可以认为β为100，对于这个场景作用等同于无限大。 (这个公式估计要改)

设备压力分析器，根据YCSB 以及 RocksDB benchmark的论文我们可以得知，工作负载对于数据访问的分布并不是均匀的，大家常说对于内容的访问遵循80/20原则，也就是20%的内容，会占有80%的访问量。因此我们可以认为，同一个工作负载的zone将会也会有冷热之分，（可能需要我们自己把热键和冷键区分）。

排序

干扰 落在die上的每秒的数据量可算 那么(channel理论带宽)

读写干扰

Zone b 受zone a干扰

(a b重叠die数)/b die数 ×（a单位die请求强度/ die理论带宽）

Zone a 受zone b干扰系数 (a b重叠die数)/a die数×

传输干扰系数

Zone b 受zone a干扰系数 (a b重叠channel数)/b channel数× （a单位channel请求强度/ channel理论带宽）

Zone a 受zone b干扰系数 (a b重叠channel数)/a channel数×