同步服务为每个请求创建单一线程，由此线程完成整个请求的处理：接收消息，处理消息，返回数据；这种情况下服务器资源对所有入栈请求开放，服务器资源被所有入栈请求竞争使用，如果入栈请求过多就会导致服务器资源耗尽宕机，或者导致竞争加剧，资源调度频繁，服务器资源利用效率降低。

异步服务则可以分别设置两个线程队列，一个专门负责接收消息，另一个专门负责处理消息并返回数据，另有一些值守线程负责任务派发和超时监控等工作。在这种情况下无论入栈请求有多少，服务器始终依照自己的能力处理请求，服务器资源消耗始终在一个可控的范围。这种模式的一个问题就是这两个线程队列的大小如何根据机器负载情况动态调整。

对于异步服务模式：

这种情况下，虽然入栈请求以消息队列的方式被异步处理但每个请求内部却是采用阻塞的方式访问外部资源，如果外部资源访问速度过慢，可能导致请求处理队列中的所有线程均处于阻塞状态，此时CPU使用率虽然很低但是却因为队列中线程已满而无法处理消息队列中的新消息，此时若能调整线程队列最大线程数将可提高CPU利用率。但另一个问题是如果线程数被调高之后所有线程的IO处理突然结束并且接下来每个线程都将进行大量计算的话那么CPU可能出现过载。

在系统运行的每个时间点上，当时正在进行IO的线程数量和正在进行计算的线程数量是不断变化着的，那么如何才能设计出一个可以根据系统当时情况自动适应负载变化的高度自适应的系统呢?

在这方面采用反应式计算模型确实能设计出适应负载能力很强的系统，系统利用率和吞吐量可以大幅提高，但这种系统仍然可能会出现系统局部负载过高的风险。

采用反应式计算模型，不仅系统中的入栈请求以消息队列的方式得以异步化，而且系统中所有的IO任务也必需依照此法行之，这些IO任务的处理需要采用异步模型（如NIO）。另外要考虑的就是如何划分异步IO消息并为其配置线程队列了，比如是要将所有IO任务放入统一的队列还是为某类IO任务设置单独的队列。

服务器资源虽然由系统分配但大多以线程为持有者被线程持有并使用，如线程堆栈，被线程持有的各类锁等资源。