1. Set MAXDOP to 1 if you're seeing CXPACKET waits as the prevalent wait type
2. Set MAXDOP to 1 for OLTP systems, and don't do anything else.
3. Old Microsoft guidance to set MAXDOP to half the number of physical processors.
4. Set MAXDOP to the number of cores in the NUMA node.
5. For OLTP systems, it can often be beneficial to set MAXDOP to 1 and then use the MAXDOP query hint to override the server-wide setting for queries that can benefit from parallelism.
6. For mixed-workload systems, you need to be careful how you set MAXDOP so you don't inadvertently penalize one of the workloads. Judicious use of the MAXDOP query hint can help here. A more powerful solution for mixed workloads is to use resource governor and have a workload group for each portion of the workload, with a different MAXDOP for each workload group.
7. For systems with high CXPACKET waits, investigate why this is the case before reducing MAXDOP. It's easy to come up with a demo where there are lots of CXPACKET waits, and while reducing MAXDOP (for the server or the query) reduces the CXPACKET waits, it also makes the query take a lot longer. CXPACKET waits can be because the statistics are incorrect and the query execution system divides up the work by the out-of-date statistics
8. Consider using the cost threshold for parallelism setting
9. [Microsoft suggests](http://blogs.msdn.com/b/jimmymay/archive/2008/11/28/case-study-part-1-cxpacket-wait-stats-max-degree-of-parallelism-option-introduction-to-using-wait-stats-to-identify-remediate-query-parallelism-bottlenecks.aspx) that if your CXPACKET wait consists of more than 5% of your total wait time, then you may have a parallelism bottleneck.

Note: - if you change the MAXDOP setting for the server, it will flush the plan cache when you do. It shouldn't, but it does. Be careful when doing this on a production server…