## Think: Ways to ensure performance of my service

1. **correct communication mechanism**

Currently, there are three main technologies for remoting a method call: Enterprise Services, .NET remoting, and ASP.NET Web services. The best choice depends upon various factors, including the source and target platforms, whether you need to communicate across an intranet or the Internet, whether you require additional services such as distributed transactions, your security requirements, deployment considerations (such as whether your communication must pass through a firewall), other port limitations, and so on.

2. **proper choice of parameters**

Improper choice of parameters can lead to a number of issues, including increased serialization costs and potential versioning problems for the Web service (for example where a custom type is updated). Your choice of parameters depends upon various factors, such as interoperability, the varying platforms used by the clients, maintainability, the type of encoding format used, and so on.

3. **Proper serialization**

Serializing large amounts of data and passing it to and from Web services can cause performance-related issues, including network congestion and excessive memory and processor overhead. Selecting an appropriate data transfer strategy — such as using a SOAP extension that performs compression and decompression or offloading data transfer to other services — is critical to the performance of your Web services solution.

4. **proper choice of encoding format**

use either literal or SOAP encoding. SOAP encoding involves more SOAP-processing overhead as compared to literal encoding.

5. **efficient caching**

Caching-related issues that can significantly affect Web services performance include failure to use caching for Web methods, caching too much data, caching inappropriate data, and using inappropriate expiration settings.

6. **efficient state management**

Inefficient state management design in Web services can lead to scalability bottlenecks because the server becomes overloaded with state information that it must maintain on a per-user basis. Common pitfalls for Web services state management include using stateful Web methods, using cookie container-based state management, and choosing an inappropriate state store. The most scalable Web services maintain no state.

7. **use multithread correctly**

unnecessarily implementing a Web method asynchronously can cause more worker threads to be used and blocked, which affects the performance of the Web server. Therefore, you should avoid doing so unless your client application needs to do something else while the service is invoked.

8. **efficient Web method processing**

using a schema to validate input upfront. This issue can be significant because the Web method may de-serialize the incoming message and then throw exceptions later on while processing the input data.