**Advantages of threads**

* Thread switching does not require Kernel mode privileges.
* User level thread can run on any operating system.
* Scheduling can be application specific in the user level thread.
* User level threads are fast to create and manage.

**Disadvantages of threads**

* In a typical operating system, most system calls are blocking.
* Multithreaded application cannot take advantage of multiprocessing.

**Advantages of a Multithreaded Application**

Multithreaded and multicontexted applications offer the following advantages:

* ***Improved performance and concurrency***

For certain applications, performance and concurrency can be improved by using multithreading and multicontexting together. In other applications, performance can be unaffected or even degraded by using multithreading and multicontexting together. How performance is affected depends on your application.

* ***Simplified coding of remote procedure calls and conversations***

In some applications it is easier to code different remote procedure calls and conversations in separate threads than to manage them from the same thread.

* ***Simultaneous access to multiple applications***

Your BEA Tuxedo clients can be connected to more than one application at a time.

* ***Reduced number of required servers***

Because one server can dispatch multiple service threads, the number of servers to start for your application is reduced. This capability for multiple dispatched threads is especially useful for conversational servers, which otherwise must be dedicated to one client for the entire duration of a conversation.

For applications in which client threads are created by the Microsoft Internet Information Server API or the Netscape Enterprise Server interface (that is, the NSAPI), the use of multiple threads is essential if you want to obtain the full benefits afforded by these tools. This may be true of other tools, as well.

**Disadvantages of a Multithreaded Application**

Multithreaded and multicontexted applications present the following disadvantages:

* ***Difficulty of writing code***

Multithreaded and multicontexted applications are not easy to write. Only experienced programmers should undertake coding for these types of applications.

* ***Difficulty of debugging***

It is much harder to replicate an error in a multithreaded or multicontexted application than it is to do so in a single-threaded, single-contexted application. As a result, it is more difficult, in the former case, to identify and verify root causes when errors occur.

* ***Difficulty of managing concurrency***

The task of managing concurrency among threads is difficult and has the potential to introduce new problems into an application.

* ***Difficulty of testing***

Testing a multithreaded application is more difficult than testing a single-threaded application because defects are often timing-related and more difficult to reproduce.

* ***Difficulty of porting existing code***

Existing code often requires significant re-architecting to take advantage of multithreading and multicontexting. Programmers need to:

* + Remove static variables
  + Replace any function calls that are not thread-safe
  + Replace any other code that is not thread-safe