overhead at runtime (jitting) what is have done to reduce this?

Jitting, or Just-in-Time (JIT) compilation, is the process where the .NET runtime compiles Intermediate Language (IL) code into native machine code at runtime. While this allows for platform independence and optimizations, it can introduce overhead, especially during the initial execution of an application.

To reduce this overhead, several techniques and improvements have been implemented:

1. **Tiered Jitting**: .NET Core introduced tiered jitting, which allows the runtime to use different levels of optimization based on the method's execution frequency. Frequently executed methods are optimized more aggressively, reducing the overhead over time1.
2. **Native AOT (Ahead-of-Time Compilation)**: With .NET 9, Native AOT compilation was introduced, allowing developers to compile .NET applications into native machine code before execution. This eliminates the need for JIT compilation at runtime, significantly reducing startup time and improving performance for specific workloads2.
3. **Improved Garbage Collection**: Optimizations in the garbage collection algorithms help reduce latency and improve throughput. This makes memory-intensive operations faster and reduces the impact of garbage collection on application performance2.
4. **Background Garbage Collection**: Background garbage collection runs more smoothly, minimizing interruptions during application execution. This helps maintain application responsiveness, especially in high-concurrency environments2.
5. **Threading Enhancements**: Improvements in threading management and async performance optimize resource utilization and multitasking, particularly in multicore systems.

These advancements collectively help reduce the runtime overhead associated with jitting, leading to faster and more efficient .NET applications.